Formale Systeme

P. H. Schmitt

Winter 2013/2014
Version: 11. August 2016

Vorwort

Formale Methoden und die zu ihrer Umsetzung notwendigen formalen Systeme spielen in der Informatik von Anfang an eine wichtige Rolle. Umfang und Komplexität der einzelnen Computeranwendungen sind in den letzten Jahren enorm angewachsen, was die Kontrolle ihrer Korrektheit immer schwieriger macht. Neben den traditionellen Methoden zur Qualitätssicherung einschließlich extensiver Testläufe wird der Wunsch immer stärker, eine formale Verifikation der logischen Korrektheit durchführen zu können. Die Fortschritte in der Entwicklung formaler Beweismethoden und die Verfügbarkeit immer schnellerer Rechner hat diesen Wunsch der Verwirklichung ein Stück näher gebracht. Untrennbar verbunden mit dem operationalen Verhalten ist die Spezifikation und Analyse komplexer Systeme mittels deklarativer formaler Sprachen.
Die Fülle der in der Informatik benutzten formalen Systeme ist schier unübersehbar. Wir machen keinen Versuch einen Überblick oder auch nur den Ansatz zu einer umfassenden Definition zu geben. Statt dessen betrachten wir ausführlich einen Kernbereich für das Arbeiten mit formalen Systemen: Logik, mit Betonung ihrer Funktion in der Informatik.
Wir befassen uns in Kapitel 1 zunächst mit der *Aussagenlogik*, insbesondere, weil die Grundidee für Vollständigkeitsbeweise hier in einer besonders einfachen Form bereits realisiert ist. Einen wesentlich stärker algorithmisch geprägten Bereich betritt man im speziellen Fall der *Gleichungslogik*: nämlich das Gebiet der *Termerzeugungssysteme*, das die zentrale Methodik für symbolisches Rechnen und für abstrakte Datentypen liefert.

Das Ziel dieser Vorlesung ist dabei die exemplarische Vermittlung zentraler Ideen. Das geschieht dann gründlich, wobei nicht nur die gültigen Aussagen, wie Korrektheits- und Vollständigkeitssätze, genannt sondern auch vollständige Beweise gegeben werden. Im Vordergrund stehen dabei die Fragestellungen:

1. wie bekommen Zeichen eine Bedeutung ?

2. was ist ein Beweis ?

Die erste Frage findet ihre Antwort in der Beschreibung des Zusammenhangs zwischen Syntax und Semantik der Prädikatenlogik, die zweite in der Angabe formaler Beweiskalküle, die der Strenge der Begriffbildungen der mathematischen Logik standhalten. Der Autor ist der Auffassung, daß ein Student der Informatik, wenigstens einmal während seines Studiums, auch die technischen Details in einem solchen Vorgehen kennenlernen sollte. Auch das geschieht wieder exemplarisch an einigen Stellen. Danach kehrt die Darstellung wieder auf eine etwas informellere Ebene zurück. Damit nicht durch zufällige Eigenheiten eines Beispiels ein falscher Eindruck entsteht, werden mehrere Logikkalküle vorgestellt.

Eine umfangreiche Sammlung von Übungsaufgaben, teilweise mit Lösungen, bietet dem Leser die Gelegenheit, neben der passiven Rezeption des Materials sich auch aktiv mit ihm auseinanderzusetzen.
Anwendungsmöglichkeiten werden an nicht trivialen Beispielen aufgezeigt. Realistische Anwendungsszenarien erwiesen sich dagegen als zu umfangreich. Notwendigerweise gibt es Themen, die in diesem Skript nicht behandelt werden konnten. So werden Logiken höherer Stufe und nichtklassische Logiken nur sehr knapp behandelt, und der Lambda-Kalkül kommt überhaupt nicht vor.

Danksagungen

Der vorliegende Text ist eine Fortentwicklung des gleichnamigen Vorlesungsskripts, das meinem Kollegen Wolfram Menzel und mir in den Jahren 1988 bis 2001 als gemeinsame Grundlage für unsere Vorlesungen diente. Ich bin Wolfram Menzel dankbar für die offene und fruchtbare Kooperation und insbesondere für die Vorleistungen, von denen er mir erlaubt hat zu profitieren.

Ein herzliches Dankeschön ergeht an alle Studentinnen und Studenten, es sind zu viele um sie hier namentlich aufzuführen, die durch ihre Kritik und Anregungen zur stetigen Verbesserung des Manuskripts beigetragen haben.

Referenzen

Verweise auf relevante oder benutzte Literatur werden an den entsprechenden Stellen im Skriptum gegeben werden. Wir listen hier jedoch einige Bücher auf, die in einem allgemeineren Sinne zur Vorbereitung der Vorlesung *Formale Systeme* herangezogen wurden.

M. Fitting: First Order Logic and Automated Theorem Proving, [Fit90]

U. Schöning: Logik für Informatiker, [Sch00]

V. Sperschneider/G. Antoniou: Logic: a Foundation for Computer Science, [SA91]

P. H. Schmitt: Theorie der logischen Programmierung, [Sch92]

Huth und Ryan: Logic in Computer Science, [HR00].

Ehrig, Mahr et al: Mathematische Grundlagen der Informatik, [EMC+99]

E. Börger: Logik, Berechenbarkeit, Komplexität, [B9̈8]

Ebbinghaus/Flum/Thomas: Mathematische Logik, [EFT92]

R. Smullyan: First Order Logic, [Smu95]

John Harrison: Handbook of Pratical Logic and Automated Reasoning, [Har09]

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Vorausetzungen

Die erfolgreiche Lektüre des Vorlesungsskripts *Formale Systeme* setzt Vertrautheit mit den elementaren Vorgehensweisen der Theoretischen Informatik voraus. Grundbegriffe wie Menge, Relation, Funktion, sollten bekannt sein. Der Leser sollte ganz allgemein in der Lage sein Definition zu verstehen, auch induktive, mathematischen Argumentationen folgen können und symbolische Notationen lesen können.

Diese Fähigkeiten können in der Regel durch den Besuch der Vorlesung *Theoretische Grundlagen der Informatik* oder der vorgeschriebenen Vorlesungen in der Mathematik erworben werden.

Es folgt eine Zusammenfassung einiger Definitionen mathematischer Grundbegriffe, die im Rest des Textes häufig oder gelegentlich benutzt werden, zusammen mit einigen elementaren Eigenschaften. Der Text geht davon aus, daß der Leser diesen Teil zunächst überspringt und darauf zurück greift, wenn er im Text einen Begriff findet, über dessen Definition er sich nicht im Klaren ist.

Definition 1.1 (Notation für Zahlbereiche)
Wir benutzen die folgende, weit verbreitete, Notation:

$\mathbb{N}$	für	die Menge der natürlichen Zahlen, inklusive 0
$\mathbb{Z}$	für	die Menge der ganzen Zahlen
$\mathbb{Q}$	für	die Menge der rationalen Zahlen
$\mathbb{R}$	für	die Menge der reellen Zahlen

Definition 1.2 (Eigenschaften von Relationen)
Sei R eine zweistellige Relation auf einer Menge D. R ist:

reflexiv Für alle $d \in D$ gilt $R(d, d)$

irreflexiv Für alle $d \in D$ gilt nicht $R(d, d)$

transitiv Für $d_1, d_2, d_3 \in D$ mit $R(d_1, d_2)$ und $R(d_2, d_3)$, gilt auch $R(d_1, d_3)$

symmetrisch Für alle $a, b \in D$ mit$R(a, b)$ gilt auch $R(b, a)$.

antisymmetrisch Für alle $d_1, d_2 \in D$ $R(d_1, d_2)$ und $R(d_2, d_1)$ gilt $d_1 = d_2$.

asymmetrisch Für alle $d_1, d_2 \in D$ kann nicht $R(d_1, d_2)$ und $R(d_2, d_1)$ gleichzeitig gelten.

funktional Aus $R(a, b_1)$ und $R(a, b_2)$ folgt $b_1 = b_2$.
Insbesondere ist die *leere* Relation $R_\emptyset$ funktional.

Ordnung R ist reflexiv, transitiv und antisymmetrisch.
Für Ordnungsrelationen wird häufig die Notation $a \preceq b$ benutzt anstelle von $R(a, b)$.

strikte Ordnung R ist transitiv, asymmetrisch.
Strikte Ordnungsrelationen werden häufig mit $a \prec b$ notiert anstelle von $R(a, b)$.

Quasiordnung R ist reflexiv und transitiv.

Strikte Quasiordnung R ist transitiv und irreflexiv.

totale Ordnung R ist eine Ordnungsrelation und zusätzlich gilt für alle $d_1, d_2 \in D$ mit $d_1 \neq d_2$ entweder $R(d_1, d_2)$ oder $R(d_2, d_1)$.
Totale Ordnungen werden auch **lineare Ordnungen** genannt.

Äquivalenzrelation R ist reflexiv, transitiv und symmetrisch.

Wenn wir Ordnung oder strikte Ordnung sagen, meinen wir eine nicht notwendig lineare Ordnung. Wenn wir diese Tatsache betonen wollen, fügen wir das Attribut **partiell** hinzu, reden also von einer partiellen Ordnung oder einer strikten partiellen Ordnung.

Lemma 1.3

1. Sei $(D, \preceq)$ eine Ordnung. Wir definieren für $a, b \in D$:

$$a \prec b \text{ gdw } a \preceq b \text{ und } a \neq b$$

   Dann ist $(D, \prec)$ eine strikte Ordnung.

2. Sei $(D, \prec)$ eine strikte Ordnung. Wir definieren für $a, b \in D$:

$$a \preceq b \text{ gdw } a \prec b \text{ oder } a = b$$

   Dann ist $(D, \preceq)$ eine Ordnung.

Beweis: Einfach.

∎

Lemma 1.4
Sei $\prec$ eine irreflexive Relation und $\preceq$ die wie in Lemma 1.3 zugeordnete reflexive Relation, i.e. $a \preceq b$ gdw $a \prec b$ oder $a = b$.

Dann ist $\prec$ genau dann asymmetrisch wenn $\preceq$ antisymmetrisch ist.

Beweis Sei $\prec$ asymmetrisch und gelte $a \preceq b$ und $b \preceq a$. Nach Definition von $\preceq$ heißt das $a \prec b$ oder $a = b$ und $b \prec a$ oder $b = a$. Da $a \prec b$ und $b \prec a$ nicht beide wahr sein können, folgt $a = b$ und damit die Antisymmetrie von $\prec$.

Sei jetzt umgekehrt $\preceq$ antisymmetrisch und gelte $a \prec b$ und $b \prec a$. Nach Definition gilt dann auch $a \preceq b$ und $b \preceq a$. Aus der Antisymmetrie von $\preceq$ folgt $a = b$ und damit $a \prec a$. Das steht aber im Widerspruch zur Irreflexivität von $\prec$. Also können $a \prec b$ und $b \prec a$ nicht beide wahr sein.

∎

Lemma 1.5
Für eine transitive Relation R fallen die Begriffe *irreflexive* und *asymmetrisch* zusammen.

Beweis: Jede, nicht notwendig transitive, asymmetrische Relation R ist auch irreflexive. Asymmetrie kann durch die Formel $\forall x \forall y(\neg(R(x,y) \wedge R(y,x)))$ formalisiert werden, Irreflexivität durch $\forall x(\neg R(x,x))$. Aus der ersten Formel folgt insbesondere der Spezialfall $x \equiv y$, also $\forall x(\neg(R(x,x) \wedge R(x,x)))$, was logisch äquivalent zu $\forall x(\neg R(x,x))$ ist.

Sei jetzt R transitiv und irreflexiv und gelte $R(a,b)$ und $(R(b,a)$. Mit Hilfe der Transitivität folgt $R(a,a)$, im Widerspruch zur Irreflexivität. Also kann $R(a,b)$ und $(R(b,a)$ nicht auftreten.

∎

Beispiel 1.6
Die folgenden Strukturen sind Beispiel für strikte, totale Ordnungen:

1. $(\mathbb{N}, <)$, die natürlichen Zahlen mit der üblichen strikten Ordnung
   Es gibt ein kleinstes Element, jedes Element hat einen nächsten Nachfolger und es gibt kein größtes Element.

2. $(\{0, \ldots, n\}, <)$, die natürlichen Zahlen von 0 bis einschließlich n mit der üblichen strikten Ordnung
   Es gibt ein kleinstes Element, es gibt ein größtes Element und jedes Element außer dem größten hat einen nächsten Nachfolger.

3. $(\mathbb{Z}, <)$, die ganzen Zahlen mit der üblichen strikten Ordnung
   Es gibt kein kleinstes Element, es gibt kein größtes Element und jedes Element hat einen nächsten Nachfolger und Vorgänger.

4. $(\mathbb{Q}, <)$, die rationalen Zahlen mit der üblichen strikten Ordnung
   Es gibt kein kleinstes Element, es gibt kein größtes Element und zwischen je zwei Elementen liegt ein drittes Element.

5. $(\mathbb{N} \times \mathbb{N}, <_{lex})$, wobei
   $(a_1, b_1) <_{lex} (a_2, b_2)$ gdw $a_1 < a_2$ oder $a_1 = a_2$ und $b_1 < b_2$.
   Es gibt ein kleinstes Element, es gibt kein größtes Element, jedes Element hat einen nächsten Nachfolger und alle Elemente außer $\{(n, 0) \mid n \in \mathbb{N}\}$ haben einen unmittelbaren Vorgänger. Im Unterschied zu 3 gibt es jedoch ein Element x (in der Tat sehr viele), so daß unendliche viele Elemente kleiner als x sind.

Beispiel 1.7
Die folgende Struktur ist ein Beispiel für eine strikte, partielle Ordnungen, die nicht total ist.

$$(\mathbb{N} \times \mathbb{N}, <_{comp})$$

wobei $(a_1, b_1) <_{comp} (a_2, b_2)$ gdw $a_1 < a_2$ und $b_1 < b_2$.

Definition 1.8 (Transitive Hülle)
Sei R eine beliebige binäre Relation mit Definitionsbereich D.
Die **transitive Hülle** von R ist die kleinste Relation R^+ auf D mit den Eigenschaften:

1. $R \subseteq R^+$,

2. R^+ ist eine transitive Relation.

Entsprechend stehen R^* und R^s für die kleinste, R umfassende transitive, reflexive und für die transitive, reflexive und symmetrische Relation.

Lemma 1.9
Zu jeder Relation R gibt stets eine

1. transitive Hülle

2. transitive, reflexive Hülle

3. transitive, reflexive, symmetrische Hülle

Beweis Wir betrachten die erste Behauptung.
Sei R eine binäre Relation mit Definitionsbereich D.
Sei $\mathcal{R} = \{R' \mid R \subseteq R' \subseteq D \times D \text{ und } R' \text{ ist transitive } \}$. Die Menge $\mathcal{R}$ von Relationen ist nicht leer, da auf jeden Fall $D \times D$ in ihr liegt. Man sieht leicht, daß $R^+ = \bigcap \mathcal{R}$ gilt. Der Nachweis für die beiden restlichen Behauptungen erfolgt mit den offensichtlichen Variationen.

∎

Lemma 1.10
Sei R eine Relation mit Definitionsbereich D.
Die Relation S sei definiert durch

$$aSb \Leftrightarrow \begin{array}{l} \text{es gibt } n \in \mathbb{N} \text{ und } d_1, \ldots, d_n \in D \text{ mit} \\ d_1 = a \text{ und } d_n = b \text{ und} \\ d_i R d_{i+1} \text{ gilt für alle } 1 \leq i < n \end{array}$$

Dann ist S die transitive Hülle von R.

Beweis Einfach.

∎

Lemma 1.11
Jede Ordnungsrelation $(D, \leq)$ kann zu einer totalen Ordnung $(D, \leq_t)$ erweitert werden.

Beweis Wenn $(D, \leq)$ noch keine totale Ordnung ist, dann gibt es $a, b \in D$ mit $a \not\leq b$ und $b \not\leq a$. Wir definieren

$$x \leq_1 y \leftrightarrow (x \leq y \text{ oder } (x \leq a \text{ und } b \leq y)$$

Wir zeigen, daß $(D, \leq_1)$ wieder eine Ordnungsrelation, also reflexiv, transitiv und antisymmetrisch, ist.

Reflexivität Nach Definition ist $\leq_1$ eine Erweiterung von $\leq$, aus $d_1 \leq d_2$ folgt also $d_1 \leq_1 d_2$. Aus der Reflexivität von $\leq$ folgt somit die Reflexivität von $\leq_1$.

Transitivität Gelte $d_1 \leq_1 d_2$ und $d_2 \leq_1 d_3$. Wir wollen $d_1 \leq_1 d_3$ zeigen. Dazu unterscheiden wir die folgenden vier Fälle:

1. $d_1 \leq d_2$ und $d_2 \leq d_3$
2. $d_1 \leq d_2$ und $d_2 \leq a$ und $b \leq d_3$
3. $d_1 \leq a$ und $b \leq d_2$ und $d_2 \leq d_3$
4. $d_1 \leq a$ und $b \leq d_2$ und $d_2 \leq a$ und $b \leq d_3$

(1) Aus der Transitivität von $\leq$ folgt $d_1 \leq d_3$ und damit auch $d_1 \leq_1 d_3$.
(2) Aus der Transitivität von $\leq$ folgt $d_1 \leq a$ und $b \leq d_3$ und damit nach Definition von $\leq_1$ auch $d_1 \leq d_3$.
(3) Aus der Transitivität von $\leq$ folgt $b \leq d_3$ und und damit nach Definition von $\leq_1$ wieder $d_1 \leq d_3$.
(4) Aus der Transitivität von $\leq$ folgt $b \leq a$ im Widerspruch zur Wahl von a und b.

Antisymmetrie Gelte $d_1 \leq_1 d_2$ und $d_2 \leq_1 d_1$. Wir wollen $d_1 = d_2$ zeigen. Wir unterscheiden die folgenden Fälle.

1. $d_1 \leq d_2$ und $d_2 \leq d_1$
2. $d_1 \leq d_2$ und $d_2 \leq a$ und $b \leq d_1$
3. $d_1 \leq a$ und $b \leq d_2$ und $d_2 \leq d_1$
4. $d_1 \leq a$ und $b \leq d_2$ und $d_2 \leq a$ und $b \leq d_1$

(1) Aus der Antisymmetrie von $\leq$ folgt unmittelbar $d_1 = d_2$.
(2) Aus der Transitivität von $\leq$ folgt zunächst $d_1 \leq a$ und dann $b \leq a$ im Widerspruch zur Wahl von a und b.
(3) Aus der Transitivität von $\leq$ folgt zunächst $b \leq d_1$ und dann $b \leq a$ im Widerspruch zur Wahl von a und b.
(4) Aus der Transitivität von $\leq$ folgt wieder der Widerspruch $b \leq a$.

Ist D eine endliche Menge, dann führt die Wiederholung dieser Operation nach endlich vielen Schritten zu einer totale Ordnung.

Für unendliches D benötigt man Hilfsmittel aus der Mengenlehre, wie z.B. das Zornsche Lemma oder transfinite Induktion. Im letzten Fall beginnt man mit $(D, \leq^0) = (D, \leq)$. Falls $(D, \leq^\alpha)$ noch nicht total ist setzt man

$(D, \leq^{\alpha+1}) = (D, (\leq^\alpha)_1)$ und für Limesordinalzahlen λ setzt man $(D, \leq^\lambda) = \bigcup_{\alpha<\lambda}(D, \leq^\alpha)$.

∎

Das nächste Lemma beschreibt eine Standardmethode um von einer Quasiordnung zu einer Ordnung zu kommen, allerdings nicht auf derselben Grundmenge.

Lemma 1.12
Sei eine Quasiordnung $(D, \preceq)$ gegeben.

1. Dann wird durch
$$a \sim b \text{ gdw } a \preceq b \text{ und } b \preceq a$$
eine Äquivalenzrelation auf D definiert.

2. Bezeichne $[a]_\sim = \{b \in D : a \sim b\}$ die Äquivalenzklasse eines Elements a. Setze $D_\sim = \{[a]_\sim : a \in D\}$ und $[a]_\sim \preceq [b]_\sim$ gdw $a \preceq b$, dann ist $(D_\sim, \preceq)$ wohldefiniert und eine Ordnungsrelation.

Wir benutzen dasselbe Symbol $\preceq$ für die Relation zwischen Elementen $a, b \in D$ und Elementen $[a]_\sim, b_\sim \in D_\sim$. Aus der Art der Argumente geht eindeutig hervor welche der beiden Relationen jeweils gemeint ist.

Beweis: Um $\sim$ als Äquivalenzrelation zu identifizieren müssen wir Reflexivität, Symmetrie und Transitivität nachweisen. Die Reflexivität $a \sim a$ folgt aus der Reflexivität der Quasiordnung $\preceq$. Die Symmetrie folgt direkt, da die rechte Seite der Definition symmetrisch in a und b ist. Aus $a \sim b$ und $b \sim c$ folgt nach Definition $a \preceq b$ und $b \preceq c$ und ebenfalls $c \preceq b$ und $b \preceq a$. Wegen der Transitivität von $\preceq$ folgt $a \preceq c$ und $c \preceq a$. Aus der Definition von $\sim$, jetzt von rechts nach links gelesen, folgt schließlich $a \sim c$.

Um die Wohldefiniertheit nachzuweisen gehen wir aus von $a \sim a'$, $b \sim b'$ und $a \preceq b$ und müssen $a' \preceq b'$ zeigen. Nach Definition können wir aus $a \sim a'$ auf $a' \preceq a$ und aus $b \sim b'$ auf $b \preceq b'$ schließen. Insgesamt haben wir also $a' \preceq a \preceq b \preceq b'$ woraus mit der Transitivität $a' \preceq b'$ folgt, wie gewünscht.

Reflexivität und Transitivität der Relation $(D_\sim, \preceq)$ folgt unmittelbar aus der Reflexivität und Transitivität von $\preceq$. Es bleibt die Antisymmetrie nachzuweisen. Aus $[a]_\sim \preceq [b]_\sim$ und $[b]_\sim \preceq [a]_\sim$ folgt $a \preceq b$ und $b \preceq a$, also $a \sim b$ und damit $[a]_\sim = [b]_\sim$.

∎

Es folgt ein typischen Beispiel für eine Quasiordnung.

Beispiel 1.13
Sei $D = \{\langle n,m\rangle \mid n,m \in \mathbb{Z}, m \neq 0\}$ und

$$\langle n,m\rangle \preceq \langle n',m'\rangle \Leftrightarrow \frac{n}{m} \leq \frac{n'}{m'}$$

$(D,\preceq)$ ist eine Quasiordnung, wegen $\langle 4,2\rangle \preceq \langle 2,1\rangle$, $\langle 2,1\rangle \preceq \langle 4,2\rangle$ und $\langle 2,1\rangle \neq \langle 4,2\rangle$ aber keine Ordnungsrelation.

Die nach Lemma 1.12 konstruierte Ordnung $(D_{\sim},\preceq)$ ist in diesem Fall isomorph zu der Ordnung auf der rationalen Zahlen $(\mathbb{Q},\leq)$.

Definition 1.14 (Kongruenzrelation)
Sei R eine zweistellige Relation auf dem Definitionsbereich D und seien $h_1,\ldots,h_k$ Funktionen auf D. Dabei sei h_i eine n_i-stellige Funktion. R heißt eine **Kongruenzrelation** auf $(D,h_1,\ldots,h_k)$ wenn:

1. R eine Äquivalenzrelation ist und
2. für jedes Funktion h_i und jede Wahl von zweimal n_i Elementen $a_1,\ldots a_{n_{i]}}$, $b_1,\ldots b_{n_i}$ aus D gilt
   aus $a_1Rb_1\ldots a_{n_i}Rb_{n_i}$ folgt $h(a_1,\ldots a_{n_i})Rh(b_1,\ldots b_{n_i})$.

Aus dieser Definition folgt insbesondere für 0-stellige Funktionen h, d.h. für Konstanten, stets hRh.

Definition 1.15 (Homomorphismus)
Es seien $\mathcal{A}_1 = (A_1,I_1)$ und $\mathcal{A}_2 = (A_2,I_2)$ Interpretationen über Σ. Ein *Homomorphismus* von $\mathcal{A}_1$ nach $\mathcal{A}_2$ ist eine Abbildung

$$\varphi : A_1 \to A_2,$$

so daß für alle $n \in I\!N$, $f \in F_\Sigma$ mit $\alpha_\Sigma(f) = n$, $p \in P_\Sigma$ mit $\alpha_\Sigma(p) = n$ und $d_1,\ldots,d_n \in D$ gilt:

1. $\varphi(I_1(f)(d_1,\ldots,d_n)) = (I_2(f))(\varphi(d_1),\ldots,\varphi(d_n))$
   im Falle $n = 0$ also: $\varphi(I_1(f)) = I_2(f)$
2. $(d_1,\ldots,d_n) \in I_1(p) \Leftrightarrow (\varphi(d_1),\ldots,\varphi(d_n)) \in I_2(p)$
   im Falle $n = 0$: $I_1(p) = I_2(p)$.

Ein *Isomorphismus* ist ein bijektiver Homomorphismus.

$\mathcal{A}_1 = (A_1,I_1)$ und $\mathcal{A}_2 = (A_2,I_2)$ heißen *isomorph*, geschrieben $\mathcal{A}_1 \cong \mathcal{A}_2$ wenn es einen Isomorphismus von $\mathcal{A}_1$ auf $\mathcal{A}_2$ gibt.

Definition 1.16 (Partition)
Sein X eine beliebige Menge. Eine Menge $P_1, \ldots, P_n$ von Teilmengen von X heißt eine *Partition* von X, wenn gilt

1. $\bigcup_{i=1}^{i=n} P_i = X$
2. für alle $1 \leq i < j \leq n$ gilt $P_i \cap P_j = \emptyset$.

Kapitel 2

Aussagenlogik: Syntax und Semantik

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Abbildung 2.1: Eine Sudoku Probem

Als Einführung in die Aussagenlogik beginnen wir mit einem konkreten Problem und dessen Lösung. Wir nehmen an, daß der Leser schon auf einer informellen Ebene mit den elementaren aussagenlogischen Operationen, oder wie man auch sagt mit den elementaren Booleschen Operationen, vertraut ist.

2.1 Einleitende Beispiele

2.1.1 Sudoku

Sudoku Probleme, ein aus Japan stammende Art von Denksportaufgaben, sind in letzter Zeit sehr populär geworden. Abb. 2.1 zeigt ein typisches, einfaches Sudoku Problem. Die Aufgabenstellung besteht darin, das vorgegebene Diagramm so zu vervollständigen, daß

1. in jeder Zeile jede Ziffer zwischen 1 und 9 mindestens einmal vorkommt,
2. in jeder Spalte jede Ziffer zwischen 1 und 9 mindestens einmal vorkommt,
3. in jeder der neun Teilregionen jede Ziffer zwischen 1 und 9 mindestens einmal vorkommt.

Da in jeder Zelle höchstens eine Ziffer stehen kann, folgt, daß in jeder der oben beschriebenen Situationen jede Ziffer sogar genau einmal vorkommt. Eine Lösung des Sudokus aus Abb. 2.1 ist in Abb. 2.2 abgebildet.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Abbildung 2.2: Die Lösung des Problems aus Abb. 2.1

Es gibt viele unterschiedliche Lösungmethode für Sudokus. Uns interessiert hier die Formulierung von Sudoku Problemen als aussagenlogische Erfüllbarkeitsprobleme. Das besondere an dieser Formulierung ist, daß die dabei auftretenden Variablen, sogenannte Boolesche Variablen, nur die Werte *wahr* oder $falsch$ annehmen dürfen. Wir führen für jede Zellenposition (i, j) des Sudoku und jede Zahl k zwischen 1 und 9 eine Boolesche Variable

$$D^k_{i,j}$$

ein, mit der Vorstellung, daß $D^k_{i,j}$ den Wert *wahr* hat, wenn auf dem Feld (i, j) die Zahl k steht. Die linke untere Zelle soll dabei die Koordinaten $(1, 1)$ haben. So ist z.B. $D^9_{9,1}$ wahr, wenn in der rechten unteren Ecke die Zahl 9 steht. Die Sudoku Regeln lassen sich mit diesem Vokabular als aussagenlogische Formeln schreiben. Wir betrachten dazu einige Beispiele.

$$D^1_{1,9} \vee D^1_{2,9} \vee D^1_{3,9} \vee D^1_{4,9} \vee D^1_{5,9} \vee D^1_{6,9} \vee D^1_{7,9} \vee D^1_{8,9} \vee D^1_{9,9}$$

sagt, daß die Ziffer 1 mindestens einmal in der ersten Zeile vorkommen muß.

$$D^1_{1,1} \vee D^1_{1,2} \vee D^1_{1,3} \vee D^1_{1,4} \vee D^1_{1,5} \vee D^1_{1,6} \vee D^1_{1,7} \vee D^1_{1,8} \vee D^1_{1,9}$$

sagt, daß die Ziffer 1 mindestens einmal in der ersten Spalte vorkommen muß.

$$D^1_{1,1} \vee D^1_{1,2} \vee D^1_{1,3} \vee D^1_{2,1} \vee D^1_{2,2} \vee D^1_{2,3} \vee D^1_{3,1} \vee D^1_{3,2} \vee D^1_{3,3}$$

sagt, daß die Ziffer 1 mindestens einmal in der Region links unten vorkommen muß.

Die bisherigen Formeln genügen jedoch noch nicht, um das Problem korrekt zu beschreiben. Man muß noch ausdrücken, daß in jeder Zelle höchstens eine Ziffer stehen kann. die Konjunktion der folgenden Formel stellt das für die Zelle $(1,1)$ sicher:
$\neg(D^1_{1,1} \wedge D^2_{1,1})$, $\neg(D^1_{1,1} \wedge D^3_{1,1})$, $\neg(D^1_{1,1} \wedge D^4_{1,1})$, $\neg(D^1_{1,1} \wedge D^5_{1,1})$, $\neg(D^1_{1,1} \wedge D^6_{1,1})$, $\neg(D^1_{1,1} \wedge D^7_{1,1})$, $\neg(D^1_{1,1} \wedge D^8_{1,1})$, $\neg(D^1_{1,1} \wedge D^9_{1,1})$, $\neg(D^2_{1,1} \wedge D^3_{1,1})$, $\neg(D^2_{1,1} \wedge D^4_{1,1})$, $\neg(D^2_{1,1} \wedge D^5_{1,1})$, $\neg(D^2_{1,1} \wedge D^6_{1,1})$, $\neg(D^2_{1,1} \wedge D^7_{1,1})$, $\neg(D^2_{1,1} \wedge D^8_{1,1})$, $\neg(D^2_{1,1} \wedge D^9_{1,1})$, $\neg(D^3_{1,1} \wedge D^4_{1,1})$, $\neg(D^3_{1,1} \wedge D^5_{1,1})$, $\neg(D^3_{1,1} \wedge D^6_{1,1})$, $\neg(D^3_{1,1} \wedge D^7_{1,1})$,
$\neg(D^3_{1,1} \wedge D^8_{1,1})$, $\neg(D^3_{1,1} \wedge D^9_{1,1})$, $\neg(D^4_{1,1} \wedge D^5_{1,1})$, $\neg(D^4_{1,1} \wedge D^6_{1,1})$, $\neg(D^4_{1,1} \wedge D^7_{1,1})$, $\neg(D^4_{1,1} \wedge D^8_{1,1})$, $\neg(D^4_{1,1} \wedge D^9_{1,1})$, $\neg(D^5_{1,1} \wedge D^6_{1,1})$, $\neg(D^5_{1,1} \wedge D^7_{1,1})$, $\neg(D^5_{1,1} \wedge D^8_{1,1})$, $\neg(D^5_{1,1} \wedge D^9_{1,1})$, $\neg(D^6_{1,1} \wedge D^7_{1,1})$, $\neg(D^6_{1,1} \wedge D^8_{1,1})$, $\neg(D^6_{1,1} \wedge D^9_{1,1})$, $\neg(D^7_{1,1} \wedge D^8_{1,1})$, $\neg(D^7_{1,1} \wedge D^9_{1,1})$ $\neg(D^8_{1,1} \wedge D^9_{1,1})$

Kompakter können wir diese Formelmenge beschreiben durch

$$\neg(D^s_{i,j} \wedge D^t_{i,j})$$

für alle $1 \leq i, j, s, t \leq 9$ mit $s < t$.

Auf diese Weise ergeben sich $81 * 36 = 2916$ Formeln.

Auch die Sudoku Regeln selbst lassen sich in dieser kompakten Weise beschreiben:

Für Zeilen:

$$D^k_{1,j} \vee D^k_{2,j} \vee D^k_{3,j} \vee D^k_{4,j} \vee D^k_{5,j} \vee D^k_{6,j} \vee D^k_{7,j} \vee D^k_{8,j} \vee D^k_{9,j}$$

für alle $1 \leq k, j \leq 9$.

Für Spalten:

$$D^k_{i,1} \vee D^k_{i,2} \vee D^k_{i,3} \vee D^k_{i,4} \vee D^k_{i,5} \vee D^k_{i,6} \vee D^k_{i,7} \vee D^k_{i,8} \vee D^k_{i,9}$$

für alle $1 \leq k, i \leq 9$.

Für Regionen:

$$\begin{array}{l} D^k_{1+3*i,1+3*j} \vee D^k_{2+3*i,1+3*j} \vee D^k_{3+3*i,1+3*j} \vee D^k_{1+3*i,2+3*j} \vee D^k_{2+3*i,2+3*j} \\ \vee D^k_{3+3*i,2+3*j} \vee D^k_{1+3*i,3+3*j} \vee D^k_{2+3*i,3+3*j} \vee D^k_{3+3*i,3+3*j} \end{array}$$

für $1 \leq k \leq 9$ und $0 \leq i, j \leq 2$.

Insgesamt braucht man $2916 + 3 * 81 = 3159$ Formeln.

2.1.2 Das Acht-Damen-Problem

Wir betrachten als ein weiteres einführendes Beispiel ein bekanntes kombinatorisches Problem. Bei dem sogenannten *8-Damen-Problem* geht es darum acht Damen so auf einem Schachbrett (mit den üblichen 64 Feldern) zu plazieren, daß sie sich gegenseitig nicht bedrohen.

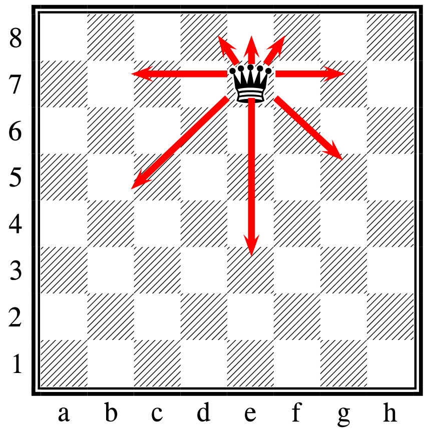


Abbildung 2.3: Die Dame im Schachspiel

Nach den Schachregeln bedroht die Dame alle Felder die sie horizontal, vertikal und diagonal in beide Richtungen erreichen kann, siehe Abb. 2.3. Man sieht, daß die Dame in der in der Abbildung gezeigten Position schon 24 Felder (einschließlich des von ihr besetzten Feldes) beherrscht. Eine der insgesamt 92 (12 wenn man symmetrische Lösungen nur einmal zählt) möglichen Lösungen des 8-Damen-Problems ist in Abb. 2.4 zu sehen.

Uns interessiert hier ein ausagenlogischer Lösungsansatz. Dazu führen wir für jedes Feld des Schachbretts eine Boolesche Variable $D_{i,j}$ ein, mit der Vorstellung, daß $D_{i,j}$ den Wert *wahr* hat, wenn auf dem Feld (i, j) eine Dame steht, sonst hat $D_{i,j}$ den Wert $falsch$. Wir benutzen hier zur Notation von Positionen auf dem Schachbrett nicht die in der Schachliteratur übliche Bezeichnung durch Buchstaben und Ziffern, sondern benutzen (diskrete) kartesische Koordinaten. Die Dame in Abb. 2.3 steht in dieser Notation auf dem Feld $(5, 7)$. Als nächstes beschreiben wir die durch die Problemstellung geforderten Einschränkungen. Steht z.B. auf dem Feld $(1, 1)$ eine Dame, dann kann auf den Feldern der ersten Spalte $(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8)$, der ersten Reihe $(2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1)$ und der Diagonalen $(2, 2)$, $(3, 3)$, $(4, 4)$, $(5, 5)$, $(6, 6)$, $(7, 7)$, $(8, 8)$ keine weitere Dame stehen. Aussagenlogisch formuliert heißt das, daß die drei folgenden Formeln den

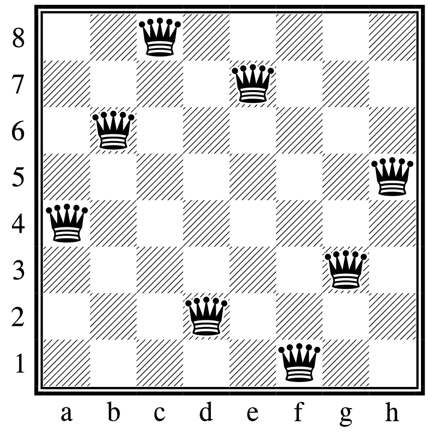


Abbildung 2.4: Eine Lösung des 8-Damenproblems

Wahrheitswert *wahr* erhalten sollen:

$$
\begin{array}{l}
D_{1,1} \to \neg D_{1,2} \wedge \neg D_{1,3} \wedge \neg D_{1,4} \wedge \neg D_{1,5} \wedge \neg D_{1,6} \wedge \neg D_{1,7} \wedge \neg D_{1,8} \\
D_{1,1} \to \neg D_{2,1} \wedge \neg D_{3,1} \wedge \neg D_{4,1} \wedge \neg D_{5,1} \wedge \neg D_{6,1} \wedge \neg D_{7,1} \wedge \neg D_{8,1} \\
D_{1,1} \to \neg D_{2,2} \wedge \neg D_{3,3} \wedge \neg D_{4,4} \wedge \neg D_{5,5} \wedge \neg D_{6,6} \wedge \neg D_{7,7} \wedge \neg D_{8,8}
\end{array}
$$

Für $D_{5,7}$ ergeben sich, jetzt gleich in aussagenlogischer Notation geschrieben, die folgenden Einschränkungen:

$$
\begin{array}{l}
D_{5,7} \to \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1} \\
D_{5,7} \to \neg D_{4,7} \wedge \neg D_{3,7} \wedge \neg D_{2,7} \wedge \neg D_{1,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7} \\
D_{5,7} \to \neg D_{6,8} \wedge \neg D_{4,6} \wedge \neg D_{3,5} \wedge \neg D_{2,4} \wedge \neg D_{1,3} \\
D_{5,7} \to \neg D_{4,8} \wedge \neg D_{6,6} \wedge \neg D_{7,5} \wedge \neg D_{8,4}
\end{array}
$$

Für jedes Feld (i, j) sei $FE_{i,j}$ die Konjunktion der Formeln, welche, wie in den beiden betrachteten Beispielen $(i, j) = (1, 1)$ und $(i, j) = (5, 7)$, die Problemeinschränkungen für dieses Feld beschreiben. Wir müssen noch die Forderung ausdrücken, daß für genau acht Positionen (i, j) die Boolesche Variable $D_{i,j}$ wahr sein soll. Das erreichen wir, indem wir für jedes k, $1 \leq k \leq 8$ verlangen daß die Formeln R_k wahr sein soll:

$$D_{1,k} \vee D_{2,k} \vee D_{3,k} \vee D_{4,k} \vee D_{5,k} \vee D_{6,k} \vee D_{7,k} \vee D_{8,k}$$

Ein Lösung des 8-Damen-Problems besteht nun darin eine Belegung aller 64 Booleschen Variablen $D_{i,j}$ zu finden, so daß alle Formeln $FE_{i,j}$ und R_k wahr werden. In aussagenlogischer Terminologie nennt man diese Aufgabenstellung ein *Erfüllbarkeitsproblem*. Wir werden im folgenden Methoden zur Lösung solcher Probleme kennen lernen.

2.2 Syntax der Aussagenlogik

Nach den einführenden Beispielen in Unterkapitel 2.1 beginnen wir mit einer systematischen Einführung in die Aussagenlogik.

Wir beginnen ganz am Anfang, mit den Zeichen, aus denen aussagenlogische Formeln aufgebaut werden. Diese Zeichen lassen sich in zwei Kategorien unterteilen: in logische Zeichen und in einen *Signatur* genannten Teil. Die logischen Zeichen kommen in jeder Aussagenlogik vor. Erst durch die Festlegung einer Signatur entsteht eine *bestimmte* aussagenlogische Sprache.

Definition 2.1 (Logische Zeichen)

1	Symbol für den Wahrheitswert „wahr“
0	Symbol für den Wahrheitswert „falsch“
$\neg$	Negationssymbol („nicht“)
$\wedge$	Konjunktionssymbol („und“)
$\vee$	Disjunktionssymbol („oder“)
$\rightarrow$	Implikationssymbol („wenn ... dann“, „impliziert“)
$\leftrightarrow$	Symbol für beiderseitige Implikation („genau dann, wenn“)
(,)	die beiden Klammern

Wir nennen $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ *Boolesche* oder *aussagenlogische Operatoren* und **1**, **0** *Boolesche Konstanten.*

Definition 2.2 (Signatur)
Eine (aussagenlogische) *Signatur* ist eine abzählbare Menge Σ.

Die Elemente von Σ heißen *atomare Aussagen*, *Atome*, *Aussagevariablen* oder *aussagenlogische Variable.*

In der Regel wird $\Sigma = \{P_0, P_1, \ldots\}$ oder $\Sigma = \{P_0, P_1, \ldots P_{n-1}\}$ benutzt werden. Wir erlauben uns aber von Fall zu Fall davon abzuweichen und andere Signaturen zu vereinbaren.

Das im folgenden in Bezeichnungen verwendete Suffix „0“ zeigt an, daß wir uns in der Aussagenlogik befinden.

Definition 2.3 (Aussagenlogische Formeln)
Zur Signatur Σ ist die Menge, $For0_\Sigma$, aller *Formeln über* Σ induktiv definiert durch

1. $\mathbf{1}, \mathbf{0} \in For0_\Sigma$, $\Sigma \subseteq For0_\Sigma$

2. Mit A, B sind auch

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B)$$

Elemente von $For0_\Sigma$

Wenn klar ist, um welches Σ es sich handelt, schreiben wir oft einfach $For0$ statt $For0_\Sigma$.

So simpel Definition 2.3 auch ist, gibt es jedoch zwei nützliche Hinweise.

Die Klammern spielen eine wichtige Rolle für die eindeutige Lesbarkeit einer aussagenlogischen Formel. Die ungeklammerte Zeichenkette $P_1 \wedge P_2 \vee P_3$ könnte sowohl als die Formel $((P_1 \wedge P_2) \vee P_3)$ als auch als $(P_1 \wedge (P_2 \vee P_3))$ gelesen werden. Größere voll geklammerte Formeln sind allerdings für den Menschen schwer zu lesen. Wir vereinbaren deswegen die folgenden *Abkürzungen*.

1. Ganz außen stehende Klammern in einer Formel dürfen weggelassen werden.

2. $\wedge$, $\vee$ binden stärker als $\rightarrow$, $\leftrightarrow$.
   $P_1 \wedge P_2 \rightarrow P_3$ steht für $((P_1 \wedge P_2) \rightarrow P_3)$
   $P_1 \leftrightarrow P_2 \vee P_2$ steht für $(P_1 \leftrightarrow (P_2 \vee P_2))$

3. $\neg$ bindet stärker als alle anderen Operatoren.
   Die Formel $\neg(P_1 \wedge P_2)$ ist verschieden von $(\neg P_1 \wedge P_2)$.

Weitere Abkürzungen folgen in Unterkapitel 2.3.4. Man beachte, daß dies nur Abkürzungen sind. Die „offizielle“ Sprache bleibt die voll geklammerte Form.

Es ist in der Informatik üblich Sprachen durch Grammatiken zu definieren. Grammatiken können leicht in eine maschinenlesbare Form gebracht werden, die z.B. als Eingabe für einen Parsergenerator dienen kann. Wir bevorzugen in diesem Text eine vom Menschen leicht lesbare und für Beweise taugliche Form der Sprachdefinition. Es ist außerdem ein Leichtes diese Form der Definition in eine Grammatik zu übersetzen, siehe Übungsaufgabe 2.2.1.

Wenn wir über Formeln reden oder sie analysieren müssen wir häufig auch ihre Teilformeln betrachten. Wir stellen deswegen hier schon die formale Definition einer Teilformel bereit.

Definition 2.4 (Teilformeln)
Für jede Formel $A \in For0_\Sigma$ ist die Menge $TF(A)$ aller Teilformeln von A induktiv definiert durch

$$\begin{array}{lcll}
TF(\mathbf{1}) & = & \{\mathbf{1}\} & \\
TF(\mathbf{0}) & = & \{\mathbf{0}\} & \\
TF(P_i) & = & \{P_i\} & P_i \in \Sigma \\
TF((A_1 \wedge A_2)) & = & \{(A_1 \wedge A_2)\} \cup TF(A_1) \cup TF(A_2) & \\
TF((A_1 \vee A_2)) & = & \{(A_1 \vee A_2)\} \cup TF(A_1) \cup TF(A_2) & \\
TF((A_1 \rightarrow A_2)) & = & \{(A_1 \rightarrow A_2)\} \cup TF(A_1) \cup TF(A_2) & \\
TF((A_1 \leftrightarrow A_2)) & = & \{(A_1 \leftrightarrow A_2)\} \cup TF(A_1) \cup TF(A_2) & \\
TF(\neg A) & = & \{\neg A\} \cup TF(A) &
\end{array}$$

B heißt eine *echte* Teilformel von A, wenn $B \in TF(A)$ und B verschieden von A ist, i.e. $B \neq A$.

2.2.1 Strukturelle Induktion

In Kapitel 1 wurde als eine der Voraussetzungen für die erfolgreiche Arbeit mit diesem Vorlesungsskript das Verständnis für induktive Definitionen genannt. Wir verdeutlichen in diesem Unterkapitel was wir damit meinen.

Eine Definition wie Definition 2.3 heißt eine Definition durch *strukturelle Induktion*. Sie besteht aus Anfangsfällen, in diesem Fall aus den Klauseln $\mathbf{1}, \mathbf{0} \in For0_\Sigma$ und $\Sigma \subseteq For0_\Sigma$. Die restlichen Klauseln der Definition sind die Induktionsschritte. Eine Funktion f mit Definitionsbereich $For0_\Sigma$ kann nun definiert werden, wie im nächsten Lemma beschrieben.

Lemma 2.5
Ist eine Funktion f

1. eindeutig definiert auf $\mathbf{1}$, $\mathbf{0}$ und den Atomen.
2. sind $f(\neg A)$, $f((A \wedge B))$, $f((A \vee B))$, $f((A \rightarrow B))$, $f((A \leftrightarrow B))$ eindeutig definiert unter der Annahme, es seien $f(A)$, $f(B)$ schon definiert

dann ist f auf der gesamten Menge $For0_\Sigma$ eindeutig definiert.

Ein Beispiel haben wir schon in Definition 2.4 kennen gelernt.

Ein wichtiges Prinzip um Sätze in der Aussagenlogik zu beweisen ist die

Lemma 2.6 (Strukturelle Induktion)
Gilt für eine Eigenschaft Eig

1. **1**, **0** und jedes Atom $P_i \in \Sigma$ haben die Eigenschaft Eig

2. Für beliebige $A, B \in For0_\Sigma$:

   - Hat A die Eigenschaft Eig, dann auch $\neg A$.
   - Haben A, B die Eigenschaft Eig, dann auch $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$.

dann gilt Eig für alle $A \in For0_\Sigma$.

Man nennt die *strukturelle Induktion* auch Induktion nach dem Aufbau der Formeln.

Wir geben ein Beispiel, das Definition und Beweis durch strukturelle Induktion demonstriert. Der Leser kann selbst überprüfen, wie gut er folgen kann.

Die folgenden werden nur für die Formulierung von Lemma 2.7 gebraucht. Zunächst numerieren wir die Positionen eine Formel mit 1 beginnend durch, z.B. für die Formel $((P_1 \wedge P_2) \vee \neg(P_3 \vee (P_1 \wedge P_2)))$

(	(	P_1	$\wedge$	P_2	)	$\vee$	$\neg$	(	P_3	$\vee$	(	P_1	$\wedge$	P_2	)	)	)
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Wir können dann z.B. sagen, daß an der 9. Stelle eine öffnende Klammer steht. Jedem Vorkommen B einer Teilformel von A ordnen wir das Intervall $IV(B)$ der Stellen vor, die von B belegt werden. So kommt $(P_1 \wedge P_2)$ zweimal in der obigen Beispielformel vor, einmal mit dem Intervall $[2, 6]$ und ein zweites Mal mit dem Intervall $[12, 16]$. Wie sagen, zwei Vorkommen von Teilformeln B_1, B_2 von A sind disjunkt, wenn $IV(B_1) \cap IB(B_2) = \emptyset$. Wir sagen, B ist ein Präfix von A, wenn $IV(B)$ mit 0 beginnt. Wir werden nachher Gebrauch machen von der folgenden offensichtlichen Eigenschaft: ist B eine Teilformelvorkommen in A und C selbst wieder eine Teilformel von B, dann gilt $IV(C) \subseteq IV(B)$.

Lemma 2.7
Sei $A \in For0_\Sigma$ und B_1, B_2 Vorkommen von Teilformeln in A, dann gilt

1. entweder B_2 ist Teilformel von B_1

2. oder B_1 ist echte Teilformel von B_2

3. oder B_1, B_2 liegen disjunkt.

Beweis Der Beweis wird durch strukturelle Induktion über A geführt.

Ist $A = \mathbf{0}$, $A = \mathbf{1}$ oder $A = P_i$ für $P_i \in \Sigma$ dann gibt es keine echte Teilformeln von A und die Behauptung des Lemmas ist trivialerweise erfüllt.

Für den Induktionsschritt gibt es nach Definition 2.3 fünf Fälle zu betrachten. Wir führen nur den Fall $A = (A_1 \wedge A_2)$ aus; die vier restlichen folgen vollkommen analog. Für das Vorkommen B_i einer Teilformel von A gibt es nach Definition 2.4 die drei Möglichkeiten: $B_i = A$, B_i ist Teilformelvorkommen in A_1 oder B_i ist Teilformelvorkommen in A_2. Insgesamt sind also 9 Kombinationen zu betrachten. Ist $B_i = A$ für ein $i \in \{1, 2\}$ dann gilt trivialerweise 1. oder 2. In den verbleibenden 4 Fälle liegen die Vorkommen B_i

a. entweder beide in derselben Teilformel A_j oder

b. in unterschiedlichen Teilformeln.

Im Fall (a) folgt aus der Induktionsvoraussetzung das B_1 und B_2 disjunkt liegen.

Im Fall (b), sagen wir B_1 kommt in A_1 vor und B_2 in A_2, gilt $IV(B_i) \subseteq IV(A_i)$, was zusammen mit $IV(A_1) \cap IV(A_2) = \emptyset$ auch $IV(B_1) \cap IV(B_2) = \emptyset$ ergibt.

■

Abschlußbemerkungen Alle aussagenlogischen Operatoren, die in diesem Unterkapitel aufgetreten sind, ware 0-stellig, 1-stellig oder 2-stellig. Gibt es auch interessante 3-stellige aussagenlogische Operatoren? Die Antwort ist *Ja* und wird in Unterkapitel 2.4.4 präzisiert.

Es gibt eine Unzahl von Möglichkeiten formale Sprachen zu definieren. Darunter auch viele, die nicht auf eine textuelle Darstellung fixiert sind, sondern z.B. Diagramme oder Graphen benutzen. Wir werden, ebenfalls in Unterkapitel 2.4.5, eine graphische Repräsentation Boolescher Funktionen kennen lernen.

2.2.2 Übungsaufgaben

Übungsaufgabe 2.2.1

1. Σ sei endlich.
   Geben Sie eine kontextfreie Grammatik für $For0_\Sigma$ an.

2. Lösen Sie dieselbe Aufgabe für unendliches $\Sigma = \{P_0, P_1, \ldots\}$.
   Hinweis: Schreibe P_i als $P\underbrace{+\ldots+}_{i\ mal\ +}$.

3. Beschränkt man sich auf die Sonderzeichen $\{(,), \mathbf{0}, \rightarrow\}$ (siehe 2.3.2), so findet man auch eine SLL1-Grammatik (zur Wahl der richtigen Produktion der Grammatik wird jeweils nur das vorderste Zeichen gebraucht).

Übungsaufgabe 2.2.2
In dieser Aufgabe betrachten wir nur Formeln in der Signatur $\Sigma_{nf} = \{\neg, \wedge, \vee\}$.

Die folgende Definition ist in vielen Zusammenhängen nützlich.

Definition 2.8 (Schachtelungstiefe von Formeln)
Für jede Formel $A \in For0_{\Sigma_{nf}}$ ist die *Schachtelungstiefe* oder einfach nur *Tiefe* $d(A)$ eine Formel wie folgt definiert.

1. $d(P_i) = 0$ für aussagenlogische Atome P_i.
2. $d(\neg A) = d(A) + 1$
3. $d(A_1 \wedge A_2) = d(A_1 \vee A_2) = max\{d(A_1), d(A_2\} + 1$

Sei A, A', B, B' Formeln in $For0_{\Sigma_{nf}}$, so daß B ein Teilformelvorkommen in A ist und A' aus A entsteht indem B durch B' ersetzt wird. Zeigen Sie

1. Gilt $d(B') = d(B)$, dann auch $d(A') = d(A)$
2. Gilt $d(B') \leq d(B)$, dann auch $d(A') \leq d(A)$

Übungsaufgabe 2.2.3
Für eine aussagenlogische Formel A definieren wir die Komplexitätsmaße

$$\begin{array}{lcl} L(A) & = & \text{Anzahl der Atomvorkommen in } A \\ Op(A) & = & \text{Anzahl der Vorkommen von } \wedge \text{ oder } \vee \text{ in } A \end{array}$$

Zeigen Sie, daß für jede Formel A, die nur die Operatoren $\wedge$ oder $\vee$ enthält die Gleichung gilt

$$L(A) = Op(A) + 1$$

.

2.3 Semantik der Aussagenlogik

Die in Unterkapitel 2.2 eingeführten aussagenlogischen Formeln sind bisher nur bedeutungslose Zeichenketten. Aus dem Kontext, aus den Beispielen aus Unterkapitel 2.1 und aus früheren Erfahrungen werden die meisten Leser eine mehr oder weniger konkrete Vorstellung haben, was Formeln bedeuten sollen. Es folgt eine präzise Definition.

Wir stellen uns Formeln als Aussagen vor, die wahr oder falsch sein können. Wenn man für jedes aussagenlogische Atom $P_i \in \Sigma$, weiß ob es wahr oder falsch ist, dann erklärt die folgende Definition 2.9, wie man den Wahrheitswert für zusammengesetzte Formeln $A \in For0_\Sigma$ ausrechnen kann.

Definition 2.9 (Semantik der Aussagenlogik)
Es sei Σ eine aussagenlogische Signatur.
Eine *Interpretation über* Σ ist eine Abbildung $I : \Sigma \to \{\mathbf{W}, \mathbf{F}\}$.
$\{\mathbf{W}, \mathbf{F}\}$ nennen wir die Menge der Wahrheitswerte.
Zu jeder Interpretation I über Σ ist die zugehörige *Auswertung* definiert als die Abbildung

$$val_I : For0_\Sigma \to \{\mathbf{W}, \mathbf{F}\}$$

mit:

$$\begin{array}{lcl} val_I(\mathbf{1}) & = & \mathbf{W} \\ val_I(\mathbf{0}) & = & \mathbf{F} \\ val_I(P) & = & I(P) \qquad \text{für jedes } P \in \Sigma \\ val_I(\neg A) & = & \begin{cases} F & \text{falls} \quad val_I(A) = \mathbf{W} \\ W & \text{falls} \quad val_I(A) = \mathbf{F} \end{cases} \end{array}$$

$val_I(A)$	$val_I(B)$	$val_I(A \wedge B)$	$val_I(A \vee B)$	$val_I(A \to B)$	$val_I(A \leftrightarrow B)$
W	**W**	**W**	**W**	**W**	**W**
W	**F**	**F**	**W**	**F**	**F**
F	**W**	**F**	**W**	**W**	**F**
F	**F**	**F**	**F**	**W**	**W**

Man beachte, daß auch Definition 2.9 wieder strukturelle Induktion benutzt.

Beispiel 2.10
Sei $\Sigma = \{P, Q, R\}$. Wir berechnen $val_I((P \wedge \neg R) \to \neg(R \vee Q))$ für die Interpretation I mit $I(P) = \mathbf{W}, I(Q) = \mathbf{F}, I(R) = \mathbf{W}$. Der leichteren Nachvollziehbarkeit wegen enthält die Tabelle auch die Auswertung aller Teilformeln.

$I(P)$	$I(Q)$	$I(R)$	$val_I(\neg R)$	$val_I(P \wedge \neg R)$	$val_I(R \vee Q)$	$val_I(\neg(R \vee Q))$	$val_I((P \wedge \neg R) \rightarrow \neg(R \vee Q))$
W	**F**	**W**	**F**	**F**	**W**	**F**	**W**

Die Definition der Auswertung zusammengesetzter Formeln folgt im Wesentlichen dem Gebrauch der logischen Operatoren in der Alltagssprache. Für $\neg$, $\wedge$, $\leftrightarrow$ ist das vollkommen zutreffend. Beim logischen *oder* ist die Alltagssprache nicht präzise genug, manchmal will man darunter das ausschließende *oder* verstanden wissen, das in unserem System durch $(P_1 \vee P_2) \wedge \neg(P_1 \wedge P_2)$ formalisiert werden kann. Bei der Implikation $P_1 \rightarrow P_2$ sind auch alle Fälle in Übereinstimmung mit dem informellen Gebrauch, bis auf den Fall $I(P_1) = I(P_2) = \mathbf{F}$. Nach Definition 2.9 ist die Aussage *wenn 0=1 ist, dann bin ich der Papst* wahr. Man kann darüber streiten, ob das unsere Intuition richtig wiedergibt. Aber die Alternative, diese Aussage als falsch zu klassifizieren, ist auch nicht überzeugend. Die in Definition 2.9 gegebene Auswertung der Implikation hat sich als Normierung durchgesetzt.

2.3.1 Boolesche Funktionen

Definition 2.11
Eine *Boolesche Funktion* ist eine Funktion von $\{\mathbf{W}, \mathbf{F}\}^n$ nach $\{\mathbf{W}, \mathbf{F}\}$ für ein $n \in \mathbb{N}$.

Man kann eine aussagenlogische Formel auch als Notation für eine Boolesche Funktion auffassen. Diese Sichtweise ist vor allem in der Technischen Informatik zuhause.

Definition 2.12 (Boolesche Funktion einer Formel)
Sei $A \in For0_\Sigma$. Um die A zugeordnete Boolesche Funktion f_A zu erklären braucht man als erstes eine Zuordnung der Argumentpositionen von f_A zu

den aussagenlogischen Atomen in Σ. Wir nehmen an dieser Stelle $\Sigma = \{P_1, \ldots, P_n\}$ an und ordnen die i-te Argumentposition von f_A dem Atom P_i zu. Für jedes Argument $\bar{b} \in \{\mathbf{W}, \mathbf{F}\}^n$ setzen wir

$$f_A(\bar{b}) = val_I(A)$$

wobei $I(P_i) = \bar{b}[i]$ für alle $1 \leq i \leq n$.

Nicht jede Funktion f_A muß von allen ihren Argumentpositionen abhängen, wie man das auch aus der Mathematik kennt, z.B. anhand der Projektionsfunktionen.

Beispiel 2.13
Die Funktion f_A für die Formel $P_1 \wedge P_3 \in For0_\Sigma$ für $\Sigma = \{P_1, P_2, P_3\}$ wird durch folgende Wahrheitstabelle wiedergegeben.

P_1	P_2	P_3	$f_A(P_1, P_2, P_3)$
W	**W**	**W**	**W**
W	**W**	**F**	**F**
W	**F**	**W**	**W**
W	**F**	**F**	**F**
F	**W**	**W**	**F**
F	**W**	**F**	**F**
F	**F**	**W**	**F**
F	**F**	**F**	**F**

Wir erinnern an ein wichtiges Resultat aus der Theorie Boolescher Funktionen, und weil es so einfach ist, liefern wir auch den Beweis dazu.

Lemma 2.14 (Funktionale Vollständigkeit)
Für jedes $n \in \mathbb{N}$ und jede Boolesche Funktion $f : \{\mathbf{W}, \mathbf{F}\}^n \to \{\mathbf{W}, \mathbf{F}\}$ gibt es eine Formel $A \in For0_\Sigma$, für $\Sigma = \{P_1, \ldots, P_n\}$ mit

$$f_A \quad = \quad f$$

Beweis Seien $\bar{b_1}, \ldots \bar{b_k}$ genau die n-Tupel aus $\{\mathbf{W}, \mathbf{F}\}^n$ mit $f(\bar{b_i}) = \mathbf{W}$. Für $1 \leq i \leq k$ und $\bar{b_i} = \langle b_{i,1}, \ldots b_{i,n} \rangle$ bilden wir die aussagenlogische Formel $A = A_1 \vee \ldots \vee A_k$ mit $A_i = A_{i,1} \wedge \ldots \wedge A_{i,n}$ mit $A_{i,j} = \begin{cases} P_j & \text{falls } b_{i,j} = \mathbf{W} \\ \neg P_j & \text{falls } b_{i,j} = \mathbf{F} \end{cases}$.

Für jede Interpretation I gilt

$$val_I(A_i) = \mathbf{W} \quad \Leftrightarrow \quad I(P_j) = b_{i,j} \text{ für alle } 1 \leq j \leq n$$

Insgesamt

$$\begin{array}{lll} f_A(\bar{b}) = \mathbf{W} & \Leftrightarrow & val_I(A) = \mathbf{W} \text{ mit } I(P_j) = \bar{b}[j] \text{ für alle } 1 \leq j \leq n \\ & \Leftrightarrow & \text{es gibt } i \text{ mit } val_I(A_i) = \mathbf{W} \\ & \Leftrightarrow & \text{es gibt } i \text{ mit } I(P_j) = b_{i,j} \text{ für alle } 1 \leq j \leq n \\ & \Leftrightarrow & \text{es gibt } i \text{ mit } \bar{b} = \bar{b}_i \\ & \Leftrightarrow & f(\bar{b}) = \mathbf{W} \end{array}$$

Falls in der obigen Konstruktion $k = 0$ ist, also für alle n-Tupel $\vec{b}$ gilt $f(\vec{b}) = \mathbf{F}$, dann ist die Disjunktion $A = A_1 \vee \ldots \vee A_k$ die leere Disjunktion, die immer als die Konstante $\mathbf{0}$ interpretiert wird.

■

In der Regel werden wir in diesem Text Formeln aus Aussagen auffassen. Gelegentlich, z.B. in Unterkapitel 2.4.4, wird aber auch die Auffassung als Boolesche Funktion vorteilhafter sein.

2.3.2 Basen

Die in Definition 2.3 gewählten aussagenlogischen Konstanten und Operatoren, $\mathbf{1}, \mathbf{0}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$, sind hochgradig redundant.

Man könnte z.B. allein mit den Operatoren $\neg$ und $\wedge$ auskommen. Das soll heißen Formeln der Form $\mathbf{1}$, $\mathbf{0}$, $A \vee B$, $A \rightarrow B$ und $A \leftrightarrow B$ lassen sich durch Formeln ersetzen, welche dieselbe Boolesche Funktion haben und nur mit den Operatoren $\neg$ und $\wedge$ aufgebaut sind, etwa so

$\mathbf{0}$	wird ersetzt durch	$P_0 \wedge \neg P_0$
$A \vee B$	wird ersetzt durch	$\neg(\neg A \wedge \neg B)$
$\mathbf{1}$	wird ersetzt durch	$\neg P_0 \vee P_0)$
$A \rightarrow B$	wird ersetzt durch	$\neg A \vee B$
$A \leftrightarrow B$	wird ersetzt durch	$(A \rightarrow B) \wedge (B \rightarrow A)$

In dieser Tabelle wird in jeder Zeile schon von den in den darüber stehenden Zeile etablierten Ersetzungen Gebrauch gemacht.

Definition 2.15
Eine Menge KOp von aussagenlogischen Konstanten und Operatoren, so daß für jede Boolesche Funktion $f : \{\mathbf{W}, \mathbf{F}\}^n \rightarrow \{\mathbf{W}, \mathbf{F}\}$ eine Formel A existiert, die nur mit Konstanten und Operatoren aus KOp aufgebaut ist, und $f_A = f$ erfüllt, nennt man eine (aussagenlogische) *Basis*.

Wir haben bisher die Basen $\{\mathbf{1}, \mathbf{0}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ und $\{\neg, \wedge\}$ kennen gelernt. Eine weitere Basis ist $\{\mathbf{0}, \rightarrow\}$ wie man an den folgenden Ersetzungen sieht:

$\neg A$	wird ersetzt durch	$A \rightarrow \mathbf{0}$
$A \vee B$	wird ersetzt durch	$\neg A \rightarrow B$
$\mathbf{1}$	wird ersetzt durch	$\neg P_0 \vee P_0)$
$A \wedge B$	wird ersetzt durch	$\neg(\neg A \vee \neg B)$
$A \leftrightarrow B$	wird ersetzt durch	$(A \rightarrow B) \wedge (B \rightarrow A)$

Es gibt auch aussagenlogische Basen, die nur aus einer einzigen Operation bestehen. Ein Beispiel ist die Operation $\downarrow$, die durch die folgende Wahrheitstabelle definiert wird:

$val_I(A)$	$val_I(B)$	$valI(A \downarrow B)$
W	**W**	**F**
W	**F**	**F**
F	**W**	**F**
F	**F**	**W**

Der Wahrheitsverlauf von $A \downarrow B$ stimmt überein mit demjenigen der Formel $\neg(A \vee B)$, was häufig auch $A\ NOR\ B$ geschrieben wird. Die funktionale Vollständigkeit der einelementigen Operatorenmenge $\{\downarrow\}$ ersieht man aus der folgenden Übersetzungstabelle:

$\neg A$	wird ersetzt durch	$A \downarrow A$
$A \wedge B$	wird ersetzt durch	$\neg A \downarrow \neg B$
	also durch	$(A \downarrow A) \downarrow (B \downarrow B)$

Der Rest folgt, da wir schon wissen, daß $\{\neg, \wedge\}$ eine Basis ist.

2.3.3 Logisches Schließen

Wir kommen zum ureigensten Begriff jeder Logik, dem des logischen Schließens. Wir beginnen mit einer terminologischen Vereinbarung:

Definition 2.16 (Modell)
Eine Interpretation I mit $val_I(A) = \mathbf{W}$ für eine Formel $A \in For0_\Sigma$ nennen wir ein *Modell* von A. Zu einer Formelmenge $M \subseteq For0_\Sigma$ ist ein *Modell* von M eine Interpretation I, welche Modell von jedem $A \in M$ ist.

Man beachte, daß die leere Menge jede Interpretation zum Modell hat.

Definition 2.17 (Logische Folgerung)
Es seien: Σ eine Signatur, $M \subseteq For0_\Sigma$, $A, B \in For0_\Sigma$. Wir sagen,

$$A \text{ ist eine logische Folgerung aus } M\text{, in Symbolen } M \models A$$

genau dann, wenn

$$\text{Jedes Modell von } M \text{ ist auch Modell von } A$$

Statt $\emptyset \models A$ schreiben wir kurz $\models A$, statt $\{B\} \models A$ schreiben wir $B \models A$.

Eng mit der logischen Folgerung verwandt sind die Begriffe der Allgemeingültigkeit und Erfüllbarkeit.

Definition 2.18 (Allgemeingültigkeit und Erfüllbarkeit)
Sei $A \in For0_\Sigma$, $M \subseteq For0_\Sigma$.

A heißt allgemeingültig	gdw	$\models A$
	d.h.	$val_I(A) = \mathbf{W}$ für alle Interpretation I
A heißt erfüllbar	gdw	es gibt einer Interpretation I
		mit $val_I(A) = \mathbf{W}$
M heißt erfüllbar	gdw	es gibt einer Interpretation I
		mit $val_I(B) = \mathbf{W}$ für alle $B \in M$

Man nennt die allgemeingültigen Formeln auch *Tautologien.* Erst in der Prädikatenlogik werden beide Begriffe differieren.

Das folgende Lemma zeigt, daß man Fragen nach logischer Folgerbarkeit auf Fragen nach Allgemeingültigkeit zurückführen kann.

Lemma 2.19 (Deduktionslemma)
Für $A, B \in For0_\Sigma$ gilt

$$A \models B \quad \text{gdw} \quad \models A \to B$$

Beweis	$A \models B$	gdw	für alle Interpretation I gilt:
			wenn $val_I(A) = \mathbf{W}$ dann $val_I(B) = \mathbf{W}$
		gdw	für alle Interpretation I gilt:
			$val_I(A \to B) = \mathbf{W}$
		gdw	$\models A \to B$

∎

Der Zusammenhang zwischen Allgemeingültigkeit und Erfüllbarkeit wird durch das folgende, offensichtliche, Lemma formuliert.

Lemma 2.20
Für $A, B \in For0_\Sigma$ gilt

A ist allgemeingültig	gdw	$\neg A$ ist nicht erfüllbar.
A ist erfüllbar	gdw	$\neg A$ ist nicht allgemeingültig

Lemma 2.20 ist die Grundlage für die Beweistechnik des Widerspruchsbeweises: um die Allgemeingültigkeit einer Formel A zu zeigen, nimmt man an ihre Negation $\neg A$ sei wahr und leitet daraus einen Widerspruch ab. Somit ist $\neg A$ als nicht erfüllbar erkannt und A nach dem Lemma allgemeingültig. Fast alle Beweissysteme, die wir in diesem Skriptum kennen lernen werden benutzen die Technik des Widerspruchsbeweises.

Wir sind jetzt auch in der Lage, die logische Äquivalenz zweier Formeln zu definieren, die wir bisher nur umschrieben haben.

Definition 2.21 (Logische Äquivalenz)
$A, B \in For0_\Sigma$
A und B heißen *logisch äquivalent*, in Symbolen $A \equiv B$
gdw
für alle Interpretationen I gilt $val_I(A) = val_I(B)$

Lemma 2.22
Für $A, B \in For0_\Sigma$ sind die folgenden Aussagen äquivalent

1. $A \equiv B$
2. $A \leftrightarrow B$ ist eine Tautologie.
3. $f_A = f_B$
4. Für jede Interpretation I gilt $val_I(A) = \mathbf{W}$ gdw $val_I(B) = \mathbf{W}$

Beweis Einfache Umformulierungen der Definitionen.

Lemma 2.23 (Ersetzungslemma)
Es seien $A, A', B, B' \in For0_\Sigma$, B eine Teilformel von A und B' logisch äquivalent zu B. Ferner entstehe A' aus A, indem an irgendwelchen Stellen, wo B in A auftritt, B durch B' ersetzt wird. Dann ist A' logisch äquivalent zu A.

Beweis Übung in struktureller Induktion.

Beispiel 2.24
Es gilt $(A \downarrow B) \equiv \neg(A \vee B)$. Nach dem Ersetzunglemma gelten auch die folgenden Äquivalenzen.

$$\begin{aligned}(A \downarrow B) \wedge (A \downarrow B) &\equiv (A \downarrow B) \wedge \neg(A \vee B)\\ &\equiv \neg(A \vee B) \wedge \neg(A \vee B)\end{aligned}$$

Bemerkung 1 In der Definition 2.17 hätten wir, wenn wir wirklich penibel wären, $\models_\Sigma$ anstelle von $\models$ schreiben müssen. Dann hätten wir aber im nächsten Satz auch erklären müssen, daß es auf das Σ nicht ankommt. Genauer, wenn $M \subseteq For0_{Sigma}$, $A \in For0_{Sigma}$ und $\Sigma \subseteq \Sigma^*$ dann gilt

$$M \models_\Sigma A \text{ gdw } M \models_{\Sigma^*} A$$

Diese Aussage formuliert ein wichtiges logisches Prinzip: Die Bedeutung einer Aussage, soll nur von den Symbolen abhängen, die in ihr vorkommen. Man kann sich schwer vorstellen, wie dieses Prinzip überhaupt verletzt werden kann.

Bemerkung 2 Um zu testen, ob A allgemeingültig (bzw. erfüllbar) ist, genügt es, die Wahrheitstafel von A aufzustellen und abzulesen, ob jede Zeile – jede Belegung der Atome in A mit Werten $\in \{\mathbf{W}, \mathbf{F}\}$ – den Wert $\mathbf{W}$ ergibt (bzw. dies für mindestens eine Zeile der Fall ist). Die Allgemeingültigkeit (Erfüllbarkeit) ist also entscheidbar, und zwar in einer Zeit, welche exponentiell ist, in der Anzahl der in der gegebenen Formel auftretenden Signatursymbole, oder auch in der Länge der Formel.

Zu gegebenem I ist die Berechnung von $val_I(A)$ linear (in der Anzahl der Signatursymbole in A, oder der Länge von A). Also liegt das Problem SAT, die Erfüllbarkeit einer aussagenlogischen Formel zu entscheiden, in der Klasse NP der nicht deterministisch polynomialen Probleme. SAT ist sogar *NP-vollständig*: Gäbe es einen (deterministischen) polynomialen Entscheidungsalgorithmus für die Erfüllbarkeit, dann wäre $NP = P$, d.h. jedes nichtdeterministisch-polynomiale Entscheidungsproblem wäre auch schon deterministisch-polynomial. Dieses Ergebnis wurde 1971 veröffentlicht in [Coo71].

2.3.4 Tautologien

Nachdem wir die Bedeutung von Tautologien für das logische Schließen erkannt haben, stellen wir in Tabelle 2.3.4 einige davon zusammen.

Genau genommen handelt es sich um eine Liste von Schemata für allgemeingültige Formeln. Für jede Ersetzung von A, B, C unten durch irgendwelche Formeln – natürlich jeweils dieselbe Formel für jedes Vorkommen von A, B, C im selben Schema – entsteht eine allgemeingültige Formel.

Wir wollen nicht zu jeder Tautologie etwas sagen, sondern uns auf einige Kommentare beschränken.

Tautologie (1) ist der einfachste logische Schluß. Wenn man A schon weiß, so kann man, sozusagen in 0 Schritten, daraus A ableiten.

Tautologie (2) spiegelt eine wichtige Voraussetzung für die Aussagenlogik, wie wir sie in diesem Text präsentieren, wieder. Eine Aussage ist entweder wahr oder sie ist falsch, es gibt keine dritte Möglichkeit, tertium non datur. Dabei gäbe es schon die Möglichkeit zu sagen, der Wahrheitswert einer Aussagen ist undefiniert, eine Aussagen ist mit großer Wahrscheinlichkeit wahr, oder ähnliches. Solche Szenarien werden in der dreiwertigen Logik, in der Wahrscheinlichkeitslogik oder fuzzy Logik untersucht. Es bleibt aber, zumindest zum heutigen Zeitpunkt, festzustellen, daß keine dieser mehrwertigen Logiken an die Bedeutung der zweiwertigen Logik heranreicht.

Tautologie (4) ist letzten Endes eine Konsequenz aus der Festlegung der Wahrheitstafel für die Implikation $\rightarrow$. Auch hier gibt es Versuche Logiken aufzubauen, die nicht dieses Phänomen aufweisen, z.B. para-konsistente Logiken, oder default Logiken. Die Motivation dafür kommt von dem Vergleich mit Alltagssituationen: enthält eine Faktenbasis einen Fehler, so ist sie deswegen noch nicht ganz nutzlos. Solange man bei der Arbeit mit der Faktenbasis die fehlerhafte Stelle nicht benutzt ist alles noch in Ordnung. Unsere Logik ist hier strikter, enthalten die Voraussetzungen einen Fehler, so gibt es keine Garantie, daß eine Folgerung daraus noch korrekt ist.

Die Kontraposition, Tautologie (6), ist eine häufig benutzte Beweisfigur, auch in diesem Vorlesungsskriptum. Wir werden, mehrfach, die Vollständigkeit eines Beweissystems beweisen. Damit ist die folgende Aussage gemeint: wenn eine Formel A allgemeingültig ist, dann gibt es einen Beweis in dem System. Um nachzuweisen, daß das stimmt, werden wir annehmen, das System findet keinen Beweis und daraus eine Interpretation konstruieren, für die A falsch ist.

Die Tautologien (7) haben wir schon im Unterkapitel 2.3.2 implizit benutzt.

Tautologie (8) erlaubt es die Implikation $\rightarrow$ durch die anderen Booleschen Operatoren zu ersetzen. Das wird im Unterkapitel über Normalformen wichtig sein.

Tautologie (9) ist die naheliegende Definition der Äquivalentoperation als

Implikation in beide Richtungen. Tautologie (10) gibt eine alternative, ebenso einleuchtende, Definition, die in manchen Situationen hilfreicher ist.

Tautologien (11) bis (15) sind strukturelle Eigenschaften der Booleschen Operationen $\wedge$ und $\vee$, die eine fast vollständige Übereinstimmung mit den Eigenschaften der Operationen $\times$ (Multiplikation) und $+$ (Addition) auf den vertrauten Zahlbereichen, z.B. $(\mathbb{N}, \times, +)$ hat. Nur bei den Distributivgesetzen ist ein Unterschied zu beobachten. Das Distributivgesetze (14) gilt noch für $(\mathbb{N}, \times, +)$, aber (15) nicht.

Das Assoziativgesetz für $\wedge$ und $\vee$ gibt uns auch die Berechtigung für eine weitere Klammereinsparung: Wir schreiben $A_1 \wedge A_2 \wedge A_3$ anstelle von $(A_1 \wedge A_2) \wedge A_3$ oder $A_1 \wedge (A_2 \wedge A_3)$. Ebenso für $\vee$ und natürlich auch für mehr als 3 Teilformeln.

Tautologie (16), das Assoziativgesetz für $\leftrightarrow$, ist vielleicht die einzige etwas verblüffende in der ganzen Liste. Wir haben bisher kein Wort darüber verloren, wie man beweist, daß alle Formeln in Tabelle 2.3.4 wirklich Tautologien sind. Wir haben es dem Leser überlassen, im Zweifelsfall eine Wahrheitstabelle zu erstellen. Für Tautologie (16) machen wir eine Ausnahme und zeigen für jede Interpretation I

$$val_I(A \leftrightarrow (B \leftrightarrow C)) = val_I((A \leftrightarrow B) \leftrightarrow C)$$

Fall: $\mathbf{I}(\mathbf{A}) = \mathbf{W}$
In diesem Fall gilt einerseits $val_I(A \leftrightarrow (B \leftrightarrow C)) = val_I(B \leftrightarrow C)$, andererseits $val_I((A \leftrightarrow B) = val_I(B)$ und damit ebenfalls $val_I((A \leftrightarrow B) \leftrightarrow C) = val_I(B) \leftrightarrow C)$.
Fall: $\mathbf{I}(\mathbf{A}) = \mathbf{F}$
In diesem Fall ist $val_I(A \leftrightarrow (B \leftrightarrow C)) = \mathbf{W}$ genau dann wenn $val_I(B \leftrightarrow C) = \mathbf{F}$. Ebenso ist $val_I(A \leftrightarrow B) = \mathbf{W}$ genau dann, wenn $val_I(B) = \mathbf{F}$ und damit auch $val_I((A \leftrightarrow B) \leftrightarrow C) = \mathbf{W}$ genau dann, wenn $val_I(B \leftrightarrow C) = \mathbf{F}$.

Tautologien (17),(18) formulieren Regeln, die für Boolesche Algebren typisch sind.

Für die Tautologie (19) wollen wir ebenfalls einen Beweis angegeben. Wir zeigen, daß für alle Interpretationen I gilt

$$val_I(A \rightarrow (B \rightarrow C)) = \mathbf{F} \text{ gdw } val_I((A \rightarrow B) \rightarrow (A \rightarrow C)) = \mathbf{F}$$

Für die linke Seite erhalten wir

$$\begin{array}{lll} val_I(A \rightarrow (B \rightarrow C)) = \mathbf{F} & \text{gdw} & val_I(A) = \mathbf{W} \text{ und } val_I(B \rightarrow C)) = \mathbf{F} \\ & \text{gdw} & val_I(A) = val_I(B) = \mathbf{W}, val_I(C) = \mathbf{F} \end{array}$$

Für die rechte Seite

$$
\begin{array}{lll}
val_I((A \to B) \to (A \to C)) = \mathbf{F} & \text{gdw} & val_I(A \to B) = \mathbf{W} \text{ und} \\
& & \quad val_I(A \to C) = \mathbf{F} \\
& \text{gdw} & val_I(A \to B) = \mathbf{W} \text{ und} \\
& & \quad val_I(A) = \mathbf{W} \text{ und } val_I(C) = \mathbf{F} \\
& \text{gdw} & val_I(A) = val_I(B) = \mathbf{W} \text{ und} \\
& & \quad val_I(C) = \mathbf{F}
\end{array}
$$

Die Tautologien (20) und (21) werden bei der Normalisierung von aussagenlogischen Formeln eine Rolle spielen. Sie erlauben es das Vorkommen von Negationszeichen zu *normieren*. Mit Hilfe der De Morganschen Regeln kann man Negationszeichen *nach innen* verschieben, bis sie nur noch vor aussagenlogischen Atomen vorkommen.

Die Äquivalenzen in (22) zeigen, daß die Konstanten **1** und **0** eine reine Bequemlichkeit sind. Man könnte auch gut ohne sie auskommen.

2.3.5 Interpolanten

Die Untersuchung von Interpolanten, sowohl für die Aussagenlogik als auch für die Prädikatenlogik, gehört zum Standardrepertoire der theoretischen Logik. Daß das Interpolationslemma, gelegentlich, auch praktische Anwendungen hat, wurde z.B. in [McM03] für Modellprüfungsverfahren gezeigt.

Wir haben dieses Unterkapitel in das Skriptum mit aufgenommen um anhand einer interessanten Fragestellung eine größere Vertrautheit mit den Begriffen der Aussagenlogik zu fördern.

Definition 2.25 (Interpolante)
Seien A,B aussagenlogische Formeln, so daß $A \to B$ eine Tautologie ist. Eine Formel C heißt eine *Interpolante* von $A \to B$, falls

1. $A \to C$ und $C \to B$ Tautologien sind und

2. in C nur solche aussagenlogischen Atome $P \in \Sigma$ vorkommen, die sowohl in A als auch in B vorkommen.
   Wegen (22) in Tabelle 2.3.4 können wir annehmn, daß **1** und **0** in C nicht vorkommen.

Lemma 2.26 (Craigsches Interpolationslemma)
Zu je zwei aussagenlogischen Formeln A,B, so daß $A \to B$ eine Tautologie ist, existiert eine Interpolante.

Beweis Der Beweis hat zwei Teile. Im ersten Teil wird ein Kandidat für eine Interpolante vorgeschlagen. Im zweiten Teil wird dann gezeigt, daß der Kandidat tatsächlich eine Interpolante ist.

Teil 1 Seien $P_1, \dots P_n$ alle in A vorkommenden aussagenlogischen Atome, die nicht in B vorkommen. Für Konstanten $c_i \in \{\mathbf{1}, \mathbf{0}\}$ bezeichnen wir mit $A[c_1, \dots, c_n]$ die Formeln, die aus A hervorgeht, indem P_i durch c_i ersetzt wird für alle $1 \leq i \leq n$. Wir setzen

$$C \equiv \bigvee_{(c_1,\dots,c_n) \in \{\mathbf{1},\mathbf{0}\}^n} A[c_1, \dots, c_n]$$

Bevor wir mit dem Beweis fortfahren betrachten wir zwei Beispiele.

Beispiel 2.27
Offensichtlich ist $P_1 \wedge P_2 \to P_1 \vee P_3$ eine Tautologie. Da sowohl $P_1 \wedge P_2 \to P_1$ als auch $P_1 \to P_1 \vee P_3$ Tautologien sind, ist P_1 als Interpolante erkannt.

Wir wollen die Konstruktion im Beweis von Lemma 2.26 verfolgen. Zunächst ist P_2 die einzige Variable, die in A aber nicht in B vorkommt. Wir erhalten $A[\mathbf{1}] = P_1 \wedge \mathbf{1} \leftrightarrow P_1$, $A[\mathbf{0}] = P_1 \wedge \mathbf{0} \leftrightarrow \mathbf{0}$. Als Interpolante also $C = A[\mathbf{1}] \vee A[\mathbf{0}] \leftrightarrow P_1 \vee \mathbf{0} \leftrightarrow P_1$.

Beispiel 2.28
Wir präsentieren ein etwas komplizierteres Beispiel zur Interpolantenberechnung. Dieses Beispiel stammt aus [D'S10]. Dazu betrachten wir

$$\begin{array}{rcl} A & = & (P_1 \vee \neg P_2) \wedge (\neg P_1 \vee \neg P_3) \wedge P_2 \\ B & = & \neg((\neg P_2 \vee P_3) \wedge (P_2 \vee P_4) \wedge \neg P_4) \end{array}$$

Durch elementare Umformungen sieht man, daß A äquivalent ist zu $P_1 \wedge P_2 \wedge \neg P_3$ und B äquivalent ist zu $\neg P_2 \vee \neg P_3 \vee P_4$. Damit sieht man auch, daß $A \to B$ eine Tautologie ist. Die Voraussetzung von Lemma 2.26 ist also erfüllt.

P_1 ist die einzige aussagenlogische Variable, die in A aber nicht in B vorkommt. In dem Beweis des Lemmas wird die Interpolante

$$C = A[\mathbf{1}] \vee A[\mathbf{0}]$$

angegeben, mit

$$\begin{array}{rcl} A[\mathbf{1}] & = & (\mathbf{1} \vee \neg P_2) \wedge (\neg \mathbf{1} \vee \neg P_3) \wedge P_2 \\ & \leftrightarrow & \mathbf{1} \wedge \neg P_3 \wedge P_2 \\ & \leftrightarrow & \neg P_3 \wedge P_2 \\ A[\mathbf{0}] & = & (\mathbf{0} \vee \neg P_2) \wedge (\neg \mathbf{0} \vee \neg P_3) \wedge P_2 \\ & \leftrightarrow & \neg P_2 \wedge \mathbf{1} \wedge P_2 \\ & \leftrightarrow & \mathbf{0} \end{array}$$

Also $C = (\neg P_3 \wedge P_2) \vee \mathbf{0} \leftrightarrow \neg P_3 \wedge P_2$. Die Überprüfung

$$P_1 \wedge P_2 \wedge \neg P_3 \rightarrow \neg P_3 \wedge P_2 \qquad \neg P_3 \wedge P_2 \rightarrow P_1 \wedge P_2 \wedge \neg P_3$$

liefert einen positiven Testfall für die Korrektheit von Lemma 2.26.

In Übungsaufgabe 2.3.7 wird eine duale Methode zur Interpolantenberechung vorgestellt. Als Interpolante wird

$$D = B[\mathbf{1}] \wedge B[\mathbf{0}]$$

vorgeschlagen, wobei $B[c]$ entsteht, in dem überall in B die Variable P_4, das ist die einzige Variable, die in B aber nicht in A vorkommt, ersetzt wird durch c. Wir rechnen:

$$\begin{array}{rcl}
B[\mathbf{1}] & = & \neg((\neg P_2 \vee P_3) \wedge (P_2 \vee \mathbf{1}) \wedge \neg \mathbf{1}) \\
& \leftrightarrow & \neg(\mathbf{0}) \\
& \leftrightarrow & \mathbf{1} \\
B[\mathbf{0}] & = & \neg((\neg P_2 \vee P_3) \wedge (P_2 \vee \mathbf{0}) \wedge \neg \mathbf{0}) \\
& \leftrightarrow & \neg((\neg P_2 \vee P_3) \wedge P_2 \wedge \mathbf{1}) \\
& \leftrightarrow & \neg((\neg P_2 \vee P_3) \wedge P_2) \\
& \leftrightarrow & \neg((P_3 \wedge P_2) \\
& \leftrightarrow & \neg P_3 \vee \neg P_2
\end{array}$$

Damit ist $D = \mathbf{1} \wedge (\neg P_3 \vee \neg P_2) \leftrightarrow \neg P_3 \vee \neg P_2$.

Der Test

$$P_1 \wedge P_2 \wedge \neg P_3 \rightarrow \neg P_3 \vee \neg P_2 \qquad \neg P_3 \vee \neg P_2 \rightarrow P_1 \wedge P_2 \wedge \neg P_3$$

fällt wieder positiv aus.

Auch $\neg P_3$ ist eine Interpolante für $A \rightarrow B$, aber keine der Berechnungen findet sie.

Teil 2 des Beweises von Lemma 2.26

Wir wollen diesen Teil des Beweises nicht präsentieren, sondern den Leser durch kleinschrittige Fragen dazu führen, den Beweis selbst zu finden. Diese Methode wurde z.B., von dem Topologen R.L.Moore propagiert und erfolgreich benutzt. Die Antworten sind am Ende der Lösungen zu den Übungsaufgaben zu diesem Kapitel auf Seite 419 zu finden. Der Idealfall ist aber, daß Sie so überzeugt sind von Ihren Antworten, daß ein Nachschlagen nicht nötig ist.

1. Wieso kommen in jedem $A[c_1, \ldots, c_n]$ nur aussagenlogische Variablen vor, die A und B gemeinsam sind?

2. Wieso kommen in jedem C nur aussagenlogische Variablen vor, die A und B gemeinsam sind?

3. Wie oben definiert, gehe die Formel $A[c_1]$ aus A hervor, indem jedes Vorkommen von P_1 durch die Konstante $c_1 \in \{\mathbf{1}, \mathbf{0}\}$ ersetzt wird. Sei weiterhin I eine Interpretation mit $val_I(A) = \mathbf{W}$. Wie muß man c_1 wählen, damit $val_I(A[c_1]) = \mathbf{W}$ gilt?

4. Die Situation sei wie in Frage 3. Wie muß man c_i, $1 \leq i \leq n$ wählen, damit $val_I(A[c_1, \ldots, c_n]) = \mathbf{W}$ gilt?

5. Wieso folgt $\models \mathbf{A} \rightarrow \mathbf{C}$ aus der Antwort auf Frage 4?

6. Wenn für eine beliebige Wahl von c_1 und eine beliebige Interpretation I gilt $val_I(A[c_1]) = \mathbf{W}$ gilt dann auch $val_I(A) = \mathbf{W}$?

7. Für eine beliebige Konstante c_1 und eine beliebige Interpretation gelte $val_I(A[c_1]) = \mathbf{W}$. Finden Sie eine Interpretation I_1, so daß $val_{I_1}(A) = \mathbf{W}$ gilt.

8. Erweitern Sie die Konstruktion aus der vorangegangenen Frage auf die Situation mit beliebig vielen Konstanten $c_1, \ldots, c_n$.

9. Seien I und J Interpretationen, so daß $val_J(B) = \mathbf{W}$ gilt und für alle aussagenlogischen Variablen P in B gilt $I(P) = J(P)$. Was können Sie über $val_I(B)$ daraus schließen?

10. Wie helfen Ihnen die Antworten zu 8 und 9 um $\models \mathbf{C} \rightarrow \mathbf{B}$ zu zeigen? Hinweis: Irgendwann einmal muß man von der Voraussetzung des Lemmas Gebrauch machen.

■

Abschlußbemerkung zur Notation Im Laufe des bisherigen Textes sind die Symbole $\leftrightarrow$, $=$, $\equiv$, $\Leftrightarrow$ eingeführt worden, die in verschiedenen Zusammenhängen die Gleichheit oder Gleichwertigkeit bezeichnen. Es lohnt sich einen Moment über die Unterschiede nachzudenken.

Das einfachste Symbol ist $\leftrightarrow$. Es kommt nur in Formeln der Aussagenlogik vor. Wenn A und B Formeln sind, dann gilt das auch für $A \leftrightarrow B$. Die übrigen Symbole $=$, $\equiv$, $\Leftrightarrow$ kommen nicht in Formeln vor. Die Benutzung des Gleichheitszeichens $=$ ist selbst dem gebildeten Laien geläufig, das Symbol $\equiv$ wurde in Definition 2.21 als symbolische Abkürzung für die Äquivalenz von Formeln eingeführt. Anstelle des Satzes „Stehen A und B für die gleiche Formel, dann sind sie trivialerweise logisch äquivalent“kann man dann auch kürzer schreiben „Für $A = B$ gilt trivialerweise $A \equiv B$. Das Zeichen $\Leftrightarrow$ ist eine Abkürzung für die Redewendung *genau dann wenn*, die häufig auch durch *gdw* abkürzt wird.

Einzig $\leftrightarrow$ ist unter den betrachteten Symbolen ein Zeichen der *Objektebene*; die anderen gehören der *Metaebene* an.

2.3.6 Übungsaufgaben

Übungsaufgabe 2.3.1
Zeigen Sie, daß die folgenden Operatormengen aussagenlogische Basen sind

1. $\{\neg, \vee\}$

2. $\{\neg, \rightarrow\}$

3. $\{|\}$
   Dabei ist die Operation | durch die folgende Wahrheitstabelle definiert.

$val_I(A)$	$val_I(B)$	$valI(A|B)$
W	**W**	**F**
W	**F**	**W**
F	**W**	**W**
F	**F**	**W**

   Der Wahrheitsverlauf von $A|B$ stimmt überein mit demjenigen der Formel $\neg(A \wedge B)$, was häufig auch $A\ NAND\ B$ geschrieben wird.

4. $\{\mathbf{1}, \oplus, \wedge\}$
   wobei die Operation $\oplus$ durch die folgende Wahrheitstabelle definiert.

$val_I(A)$	$val_I(B)$	$valI(A \oplus B)$
W	**W**	**F**
W	**F**	**W**
F	**W**	**W**
F	**F**	**F**

   Man sieht, $A \oplus B$ ist das ausschließende *oder*, das manchmal auch mit $A\ XOR\ B$ notiert wird und auch durch $(A \vee B) \wedge \neg(A \wedge B)$ definiert werden könnte.

Übungsaufgabe 2.3.2
Wir betrachten Formeln, die ausschließlich mit den Operatoren $\{\mathbf{1}, \mathbf{0}, \oplus, \wedge\}$ aufgebaut sind, siehe auch die vorangegangene Übungsaufgabe 2.3.1(4).
Zeigen Sie, daß die folgenden Formeln Tautologien sind

$P \wedge P$	$\leftrightarrow$	P	$P \wedge \mathbf{0}$	$\leftrightarrow$	$\mathbf{0}$	$P \wedge \mathbf{1}$	$\leftrightarrow$	P
$P \oplus P$	$\leftrightarrow$	$\mathbf{0}$	$P \oplus \mathbf{0}$	$\leftrightarrow$	P			
$P \wedge Q$	$\leftrightarrow$	$Q \wedge P$	$(P \wedge Q) \wedge R$	$\leftrightarrow$	$Q \wedge (P \wedge R)$			
$P \oplus Q$	$\leftrightarrow$	$Q \oplus P$	$(P \oplus Q) \oplus R$	$\leftrightarrow$	$Q \oplus (P \oplus R)$			
$P \wedge (Q \oplus R) \leftrightarrow (P \wedge Q) \oplus (P \wedge R)$								

Übungsaufgabe 2.3.3
Seien $A, A', B, B' \in For0_\Sigma$ mit $A \equiv A'$ und $B \equiv B'$. Dann gilt

1. $\neg A \equiv \neg A'$
2. $(A \wedge B) \equiv (A' \wedge B')$
3. $A \vee B \equiv A' \vee B'$

Übungsaufgabe 2.3.4

1. A ist allgemeingültig gdw $A \equiv \mathbf{1}$
2. A ist unerfüllbar gdw $A \equiv \mathbf{0}$

Übungsaufgabe 2.3.5
Wir nennen zwei aussagenlogische Formeln A,B *wechselseitig inkonsistent* wenn $\neg(A \wedge B)$ eine Tautologie ist, i.e. $\models \neg(A \wedge B)$.
Zeigen Sie die folgende Behauptung.
Zu zwei wechselseitig inkonsistenten aussagenlogische Formeln A,B gibt es eine Formel C die nur Atome enthält, die in A und B vorkommen mit
$\models A \rightarrow C$
C und B sind wechselseitig inkonsistent

Das ist die Formulierung, in der das Interpolationslemma in der Arbeit von McMillan [McM03] benutzt wird. Auch die Terminologie der wechselseitigen Inkonsistenz ist daraus entnommen.

Übungsaufgabe 2.3.6
Seien C_1, C_2 beides Interpolanten für die Implikation

$$A \rightarrow B.$$

Dann sind auch $C_1 \wedge C_2$ und $C_1 \vee C_2$ Interpolanten für $A \rightarrow B$..

Übungsaufgabe 2.3.7
Sei $A \to B$ eine Tautologie.
Seien $Q_1, \ldots, Q_k$ alle Variablen in B, die nicht in A vorkommen. Sind c_i Konstanten aus $\{\mathbf{1}, \mathbf{0}\}$ dann bezeichne $B[c_1, \ldots, c_k]$, wie im Beweis von Lemma 2.26, die Formel, die aus B entsteht, wenn man alle Vorkommen von Q_i durch c_i ersetzt. Außerdem sei:

$$D \equiv \bigwedge_{(c_1,\ldots,c_n)\in\{\mathbf{1},\mathbf{0}\}^n} B[c_1, \ldots, c_n]$$

Zeigen Sie, daß D eine Interpolante von $A \to B$ ist.

Übungsaufgabe 2.3.8
Sei $A \to B$ eine Tautologie. Sei C die im Beweis von Lemma 2.26 konstruierte Interpolante von $A \to B$ und D die Formel aus der vorangegangen Übungsaufgabe 2.3.7. Zeigen Sie, daß C die stärkste und D die schwächste Interpolante von $A \to B$ ist, d.h. zeigen Sie, daß für jede Interpolante U

$$C \to U \text{ und } U \to B$$

Tautologien sind.

Übungsaufgabe 2.3.9
Gegeben sei eine Landkarte mit L Ländern, die mit den Zahlen von 0 bis $L-1$ bezeichnet werden. Die binäre Relation $Na(i, j)$ trifft auf zwei Länder i und j zu $(0 \leq i, j < L)$, wenn sie benachbart sind. Die Landkarte soll nun mit den **drei** Farben *rot*, *grün* und *blau* so eingefärbt werden, dass keine zwei benachbarten Länder dieselbe Farbe erhalten.

Geben Sie – in Abhängigkeit von L und Na – eine aussagenlogische Formel F an, so dass F genau dann erfüllbar ist, wenn eine Färbung der geforderten Art möglich ist.

Hinweis: Die Landkarte muss nicht zwei-dimensional sein, d. h. der durch $Na(i, j)$ gegebene Graph muss nicht planar einbettbar sein.

Übungsaufgabe 2.3.10

Definition 2.29
Eine unerfüllbare Klauselmenge S heißt eine *minimal unerfüllbare* Klauselmenge, wenn jede echte Teilmenge $S_0 \subset S$ erfüllbar ist.

Sei S eine minimal unerfüllbare Klauselmenge dann gilt

1. S enthält keine tautologische Klausel, d.h. keine Klausel C, die für eine aussagenlogische Variable P sowohl P selbst als auch $\neg P$ enthält.

2. für jede Interpretation I gibt es Klauseln $C, D \in S$ und ein Literal L, so daß $I(C) = W$, $I(D) = F$, L kommt in C und $\bar{L}$ kommt in D vor.

1.	$A \to A$	Selbstimplikation
2.	$\neg A \vee A$	Tertium non datur
3.	$A \to (B \to A)$	Abschwächung
4.	$\mathbf{0} \to A$	Ex falso quodlibet
5.	$\neg\neg A \leftrightarrow A$	Doppelnegation
6.	$(A \to B) \leftrightarrow (\neg B \to \neg A)$	Kontraposition
7.	$A \wedge B \leftrightarrow \neg(\neg A \vee \neg B)$, $A \vee B \leftrightarrow \neg(\neg A \wedge \neg B)$	Dualität
8.	$(A \to B) \leftrightarrow \neg A \vee B$	Elimination von $\to$
9.	$(A \leftrightarrow B) \leftrightarrow (A \to B) \wedge (B \to A)$	Def. von $\leftrightarrow$
10.	$(A \leftrightarrow B) \leftrightarrow (A \wedge B) \vee (\neg A \wedge \neg B)$	Alternative Def. von $\leftrightarrow$
11.	$A \wedge B \leftrightarrow B \wedge A$, $A \vee B \leftrightarrow B \vee A$	Kommutativität
12.	$(A \wedge B) \wedge C \leftrightarrow (A \wedge (B \wedge C))$	Assoziativität
13.	$((A \vee B) \vee C) \leftrightarrow (A \vee (B \vee C))$	Assoziativität
14.	$A \wedge (B \vee C) \leftrightarrow (A \wedge B) \vee (A \wedge C)$	Distributivität
15.	$A \vee (B \wedge C) \leftrightarrow (A \vee B) \wedge (A \vee C)$	Distributivität
16.	$(A \leftrightarrow (B \leftrightarrow C)) \leftrightarrow (A \leftrightarrow B) \leftrightarrow C)$	Assoziativität
17.	$A \wedge (A \vee B) \leftrightarrow A$, $A \vee (A \wedge B) \leftrightarrow A$	Absorption
18.	$A \wedge A \leftrightarrow A$, $A \vee A \leftrightarrow A$	Idempotenz
19.	$(A \to (B \to C)) \leftrightarrow ((A \to B) \to (A \to C))$	Klammerverteilung
20.	$\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$	De Morgansches Gesetz
21.	$\neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$	De Morgansches Gesetz
22.	$A \vee \mathbf{1} \leftrightarrow \mathbf{1}$, $A \wedge \mathbf{1} \leftrightarrow A$, $A \vee \mathbf{0} \leftrightarrow A$, $A \wedge \mathbf{0} \leftrightarrow \mathbf{0}$	**1**,**0**-Elimination

Tabelle 2.1: Tautologien

2.4 Normalformen

Die Benutzung aussagenlogischer Formeln stößt man auf zwei, miteinander im Konflikt stehenden, Anforderungsprofile. Bei der Formalisierung von Problemstellungen durch aussagenlogische Formeln wünscht man sich einen möglichst flexiblen Rahmen: alle aussagenlogischen Operatoren sollen zu Verfügung stehen, und es sollte keine Einschränkung an die Kombination von Teilformel geben. So hatten wir das in Definition 2.3 auch organisiert. Will man andererseits Werkzeuge zur Bearbeitung aussagenlogischer Formelmengen bauen, z.B. SAT-solver oder Theorembeweiser, dann ist es vorteilhaft ein möglichst simples, restriktives Eingabeformat zu verlangen. Um diesen Konflikt aufzulösen benutzt man die Idee von Normalformen. Jede Formel kann in eine, im Idealfall, logisch äquivalente Formel in Normalform transformiert werden. Für praktische Belange ist es auch wichtig, daß diese Transformation möglichst effizient durchgeführt werden kann.

2.4.1 Disjunktive und konjunktive Normalform

In diesem Unterkapitel benutzen wir nur die aussagenlogischen Operatoren in $Op_{nf} = \{\neg, \wedge, \vee\}$. Nach den Überlegungen aus Unterkapitel 2.3.2 ist das keine Einschränkung der Allgemeinheit.

Wir beginnen mit der einfachsten Normalform.

Definition 2.30 (Negationsnormalform)
Eine aussagenlogische Formel $A \in For0_\Sigma$ heißt eine *Negationsnormalform* wenn in ihr das Negationszeichen höchsten in der Form $\neg P_i$ für ein aussagenlogisches Atom P_i vorkommt.

Damit ist auch $\neg\neg P_i$ ausgeschlossen.

Lemma 2.31 (Negationsnormalform)
Zu jeder aussagenlogische Formel $A \in For0_\Sigma$ gibt es eine logisch äquivalente Negationsnormalform $A_{nnf} \in For0_\Sigma$.

Beweis Wir werden einen Pseudoalgorithmus präsentieren der eine beliebige Formel $A \in For0_\Sigma$ in eine Negationsnormal transformiert.

Für die Zwecke dieses Beweises führen wir die folgende Notation ein. Sei weiterhin $A \in For0_\Sigma$. Dann steht $N(A)$ für die Menge aller Vorkommen von Teilformeln von A von der Form $\neg A'$, so daß A' kein Atom ist. Offensichtlich ist A eine Negationsnormalform, genau dann wenn $N(A) = \emptyset$.

Der Pseudoalgorithmus sieht jetzt so aus:

> **while** $N(A) \neq \emptyset$
> **do**
> **choose** $\neg B \in N(A)$;
> **if** $B = \neg B_0$ **replace** $\neg\neg B_0$ **in** A **by** B_0;
> **if** $B = B_1 \wedge B_1$ **replace** $\neg B$ **in** A **by** $(\neg B_1 \vee \neg B_2)$;
> **if** $B = B_1 \vee B_1$ **replace** $\neg B$ **in** A **by** $(\neg B_1 \wedge \neg B_2)$;
> **od**

Wegen der Tautologien

$$\begin{array}{lcl} \neg\neg A & \leftrightarrow & A \\ \neg(A_1 \wedge A_2) & \leftrightarrow & \neg A_1 \vee \neg A_2) \\ \neg(A_1 \vee A_2) & \leftrightarrow & \neg A_1 \wedge \neg A_2) \end{array}$$

sind wir auf jeden Fall sicher, daß nach jedem Schleifendurchlauf die neue Formel A logisch äquivalent zur Anfangsformel A ist. Es bleibt noch die Terminierung sicherzustellen. Einerseits wird das kaum jemand in Frage stellen, andererseits ist ein strikter Beweis, den z.B. auch ein Theorembeweiser nachvollziehen könnte, nicht ganz einfach. Eine ideale Übungsaufgabe für den engagierten Leser, siehe Übungsaufgabe 2.4.1 auf Seite 67.

■

Definition 2.32 (Konjunktive und disjunktive Normalform)

1. Ein *Literal* ist ein Atom oder ein negiertes Atom.

2. Eine Formel ist in *disjunktiver Normalform* (DNF), wenn sie Disjunktion von Konjunktionen von Literalen ist.

3. Eine Formel ist in *konjunktiver Normalform* (KNF), wenn sie Konjunktion von Disjunktionen von Literalen ist.

Eine Disjunktionen von Literalen wird häufig auch eine *Klausel* genannt. Eine Formel in konjunktiver Normalform ist also eine Konjunktion von Klauseln. Die konjunktiver Normalform wird deswegen auch *Klauselnormalform* genannt.

Lemma 2.33
Zu jeder Formel $A \in For0_\Sigma$ gibt es eine logische äquivalente konjunktive Normalform und eine logische äquivalente disjunktive Normalform.

Beweis In einem ersten Schritt stellt man die Negationsnormalform A_{nnf} von A her. Für die weitere Transformation in eine konjunktive Normalform werden wir wieder einen Pseudoalgorithmus präsentieren.

Für den nachfolgenden Pseudoalgorithmus brauchen wir noch die beiden folgenden Vereinbarungen. Für eine Formeln $A \in For0_\Sigma$ in Negationsnormalform sei $K(A)$ die Menge aller Teilformelnvorkommen von A von der Form $B \vee (C \wedge D)$ oder $(C \wedge D) \vee B$. Eine Formel $F \in K(A)$ heißt minimal, wenn es keine Teilformel F' von F gibt, die auch in $K(A)$ liegt. Offensichtlich ist A genau dann eine konjunktive Normalform, wenn $K(A) = \emptyset$. Wenn $K(A) \neq \emptyset$, dann gibt es auch (mindestens) eine minimale Formel $F \in K(A)$.

Der Pseudoalgorithmus sieht jetzt so aus:

> **while** $K(A) \neq \emptyset$
> **do**
> **choose minimal** $F \in K(A)$;
> **if** $F = B \vee (C \wedge D)$ **replace** F **in** A **by** $(B \vee C) \wedge (B \vee D)$;
> **if** $F = (C \wedge D) \vee B$ **replace** F **in** A **by** $(C \vee B) \wedge (D \vee B)$;
> **od**

Die Formel A' nach Terminierung des Algorithmus ist eine konjunktive Normalform, wegen $K(A) = \emptyset$ und wegen der Distributivgesetze aus Tabelle 2.3.4 auch logisch äquivalent zur Ausgangsformel.

Durch Anwendung der Kommutativität, Idempotenz, Absorption läßt sich die Formel gegebenenfalls noch vereinfachen.

Der Nachweis der Terminierung bleibt wieder dem engagierten Leser als Übungsaufgabe 2.4.2 überlassen.

Den reinen Existenzbeweis einer disjunktiven Normalform können wir uns jetzt einfach machen. Wir wissen, daß es eine konjunktive Normalform für $\neg A$ gibt, $\neg A \leftrightarrow \bigwedge_i \bigvee_j L_{ij}$. Daraus folgt $A \leftrightarrow \bigvee_i \bigwedge_j \neg L_{ij}$. Wenn wir jetzt noch doppelte Negationen beseitigen haben wir eine disjunktive Normalform für A.

■

Man kann zu einer gegebenen Booleschen Funktion f auch direkt eine Formel A mit $f_A = f$ konstruieren, so daß A in Normalform ist. Die im Beweis von Lemma 2.14 konstruierte Formel war schon eine disjunktive Normalform.

Definition 2.34 (Vollständige Normalformen)
Eine konjunktive Normalform $\bigwedge_i \bigvee_j L_{i,j}$ in der Signatur Σ heißt *vollständige Normalform* wenn in jeder Klausel $C_i = \bigvee_j L_{i,j}$ für jedes Atom $P \in \Sigma$ entweder P oder $\neg P$ unter den Literalen $L_{i,j}$ vorkommt.

Analog ist eine vollständige disjunktive Normalform definiert.

Lemma 2.35 (Eindeutigkeit vollständiger Normalformen)
Sei $\Sigma = \{P_0, \ldots, P_n\}$ und $A \in For0_\Sigma$.
Die vollständige konjunktive Normalform von A, ohne triviale Klauseln, ist eindeutig bis auf die Reihenfolge der Klauseln und der Literalen in den Klauseln.
Eine triviale Klausel C ist eine Klausel, in der für ein Atom sowohl P, als auch $\neg P$ vorkommt. Eine triviale Klausel trägt nicht zum Wahrheitswert eine konjunktiven Normalform bei.

Ebenso ist die vollständige disjunktive Normalform ohne triviale Konjunktionen eindeutig.

Beweis Wir präzisieren zunächst, was zu beweisen ist. Seien A_1, A_1 zwei vollständige konjunktive Normalformen ohne triviale Klauseln in $For0_\Sigma$. Sind A_1, A_2 logisch äquivalent, dann sind sie, bis auf die Reihenfolge, gleich. Wir benutzen Lemma 2.22, das besagt, daß A_1, A_2 logisch äquivalent sind, genau dann wenn $f_{A_1} = f_{A_2}$ ist. Außerdem definieren wir für eine vollständige Klausel $C = L_0 \vee \ldots \vee L_n$, wobei $L_i = P_i$ oder $L_i = \neg P_i$ ist das Tupel $V_C = \langle W_o, \ldots, W_n \rangle$ von Wahrheitswerten durch $W_i = \mathbf{1} \Leftrightarrow L_i = P_i$. Dann gilt automatisch auch $W_i = \mathbf{0} \Leftrightarrow L_i = \neg P_i$. Der Kern des Beweises liegt in der Beobachtung: C kommt in A_i vor genau dann wenn $f_{A_i}(V_C) = \mathbf{1}$.

■

Vollständige Normalformen sind für praktische Zwecke untauglich. Damit ist auch die Eindeutigkeit nur von geringer Bedeutung. Die vollständige konjunktive Normalform von $P_1 \wedge P_2$ in $For0_{\{P_1,P_2\}}$ ist $(P_1 \vee P_2) \wedge (P_1 \vee \neg P_2) \wedge (P_1 \vee P_2) \wedge (\vee P_1 \vee P_2)$. In der Signatur $\Sigma = \{P_1, P_2, P_3\}$ hätte die vollständige konjunktive Normalform von $P_1 \wedge P_2$ sogar 8 Klauseln.

Definition 2.36 (Minimale Normalform)
Eine disjunktive Normalform A heißt *minimal*, wenn durch Weglassen einer Klausel oder durch Weglassen eines Literals in einer Klausel eine Formel entsteht, die nicht mehr logisch äquivalent zu A ist.

Analog ist eine minimale disjunktive Normalform definiert.

Verfahren zur Konstruktion minimaler Normalformen werden in der Schaltkreistheorie behandelt, siehe z.B. [Lie06, Kapitel 1.2]. Im nächsten Kapitel sehen wird, daß Primimplikanten in dieser Konstruktion eine wesentliche Rolle spielen, gehen in diesem Text aber nicht weiter darauf ein.

2.4.2 Primimplikanten

Definition 2.37 (Primimplikanten)
Sei C eine beliebige aussagenlogische Formel.

Ein Primimplikant P von C ist

1. eine Konjunktion von Literalen,
2. so daß $P \to C$ eine Tautologie ist, und
3. für jede Teilkonjunktion P' von P ist $P' \to C$ keine Tautologie.

Definition 2.38 (Essentielle Primimplikanten)
Sei C eine beliebige aussagenlogische Formel und P ein Primimplikant von C.

P heißt ein essentieller Primimplikant von C, falls für jede Teilmenge $\mathcal{Q}$ von Primimplikanten von C

- für die $C \leftrightarrow \bigvee \mathcal{Q}$ gilt
- $P \in \mathcal{Q}$ gelten muß.

Lemma 2.39
Sei C eine beliebige aussagenlogische Formel, dann ist

$$C \leftrightarrow \bigvee \{P \mid P \text{ ist Primimplikant von } C\}$$

eine Tautologie.

Beweis
Wir haben zu zeigen, daß

$$\begin{array}{c} C \leftarrow \bigvee \{P \mid P \text{ ist Primimplikant von } C\} \\ \text{und} \\ C \rightarrow \bigvee \{P \mid P \text{ ist Primimplikant von } C\} \end{array}$$

beides Tautologien sind. Für die erste Formel ist das eine unmittelbare Konsequenz aus der Definition. Zum Beweis der zweiten Implikation betrachten wir eine Interpretation I mit $val_I(C) = \mathbf{W}$ und wollen $val_I(RS) = \mathbf{W}$ für die rechte Seite RS der Implikation zeigen. Seien $A_1, \ldots A_n$ alle in C vorkommenden Atome. Für jedes i mit $1 \leq i \leq n$ sei

$$L_i = \begin{cases} A_i & \text{falls } val_I(A_i) = \mathbf{W} \\ \neg A_i & \text{falls } val_I(A_i) = \mathbf{F} \end{cases}$$

Außerdem sei $P_I = \bigwedge_i L_i$. Offensichtlich gilt $val_I(P_I) = \mathbf{W}$. Für jede Interpretation J mit $val_J(P_I) = \mathbf{W}$ stimmt J mit I auf allen Atomen, die in C vorkommen, überein. Also gilt auch $val_J(C) = \mathbf{W}$. Damit ist $P_I \rightarrow C$ eine Tautologie. Sei P_I^0 die kürzeste Teilkonjunktion von P_I, so daß $P_I^0 \rightarrow C$ immer noch eine Tautologie ist. Nach Definition ist P_I^0 ein Primimplikant von C und da P_I^0 eine Teilkonjunktion von P_I ist, gilt auch $val_I(P_I^0) = \mathbf{W}$. Damit gilt aber auch für die rechte Seite RS der obigen Implikation $val_I(RS) = \mathbf{W}$.

■

2.4.3 Kurze Konjunktive Normalform

Ein Problem, das in jeder Situation auftritt, in der aus einem umfangreicheren Satz von Operatoren eine Normierung mit nur wenigen Operatoren vorgenommen wird, ist die wachsende Größe der Normalform. Hier ein Beispiel.

Beispiel 2.40 (Exponentielle Konjunktive Normalform)
Wir beginnen mit der konkjunktiven Normalform Die konjunktive Normalform von

$$B = (P_{1,1} \wedge P_{1,2}) \vee \ldots \vee (P_{n,1} \wedge P_{n,2})$$

berechnet sich zu

$$\bigwedge \{P_{1,f(1)} \vee \ldots \vee P_{n,f(n)} \mid f : \{1, \ldots, n\} \rightarrow \{1, 2\}\}.$$

Für $n = 3$ ist das:

$$\begin{array}{c}
(P_{1,1} \vee P_{2,1} \vee P_{3,1}) \wedge (P_{1,1} \vee P_{2,1} \vee P_{3,2}) \wedge \\
(P_{1,1} \vee P_{2,2} \vee P_{3,1}) \wedge (P_{1,1} \vee P_{2,2} \vee P_{3,2} \wedge \\
(P_{1,2} \vee P_{2,1} \vee P_{3,1}) \wedge (P_{1,2} \vee P_{2,1} \vee P_{3,2}) \wedge \\
(P_{1,2} \vee P_{2,2} \vee P_{3,1}) \wedge (P_{1,2} \vee P_{2,2} \vee P_{3,2})
\end{array}$$

Das Beispiel 2.40 zeigt eine Schwierigkeit des vorgeschlagenen Tautologienachweises: in B treten $2 * n$ Literale auf, aber in der konjunktiven Normalform sind es $n * 2^n$.

Wir wollen hier eine Transformation vorstellen, die zu einer beliebigen aussagenlogischen Formel B mit n Literalen eine konjunktive Normalform B_{kcnf} mit $c * n$ Literalen herstellt für eine kleine Konstante c. Der Preis, der dafür zu zahlen ist, liegt darin, daß B_{kcnf} nur erfüllbarkeitsäquivalent zu B ist. Die Idee kann in vielen Kontexten eingesetzt werden. Als erste Veröffentlichung wird meist Tseitin zitiert [Tse68, Tse83]. Deswegen spricht man manchmal auch von der *Tseitin Normalform*. Auch die Terminologie *definitorische Normalform* ist in Gebrauch. Wir haben uns für die Bezeichnung *kurze konjunktive Normalform* entschieden. Unsere Darstellung orientiert sich an [Min90]. Eine ausführliche Behandlung findet man auch in [Ede92, pp. 48 - 52].

Wir erklären zunächst die Methode anhand des Beispiels 2.40. Der allgemeine Fall wird dann in Definition 2.42 behandelt.

Beispiel 2.41 (Kurze konjunktive Normalform)

In einem ersten Schritt wird für jede Teilformel der Form $C \lor D$ oder $C \land D$, eine neue aussagenlogische Variable eingeführt und mit einer Biimplikation $\leftrightarrow$ definiert, daß diese Variable für die entsprechende Teilformel steht. Dabei werden von innen nach außen fortschreitend die schon eingeführten Abkürzungen wiederbenutzt. Schließlich wird noch die Einerklausel Q_{2n-1} hinzugefügt. Alle anderen so erhaltenen Formeln sind von der Form $Q \leftrightarrow (L_1 \lor L_2)$ oder $Q \leftrightarrow (L_1 \land L_2)$ für eine neue Variable Q und Literale L_i, die natürlich auch die neuen Variablen enthalten können.

In einem zweiten Schritt werden die Äquivalenzen $Q \leftrightarrow (L_1 \text{ op } L_2)$ ersetzt durch $(\neg Q \lor (L_1 \text{ op } L_2)) \land (Q \lor \neg(L_1 \text{ op } L_2)$.

Im dritten Schritt wird für jede der bisher erhaltenen Formeln $(\neg Q \lor (L_1 \text{ op } L_2)) \land (Q \lor \neg(L_1 \text{ op } L_2)$ durch maximal zweimalige Anwendung eines Distributivgesetzes eine konjunktive Normalform erreicht.

$$
\begin{array}{llll}
 & \multicolumn{2}{l}{1.\mathit{Schritt}} & 2.\mathit{Schritt} \\
Q_1 & \leftrightarrow & P_{1,1} \land P_{1,2} & (\neg Q_1 \lor (P_{1,1} \land P_{1,2})) \land (Q_1 \lor \neg(P_{1,1} \land P_{1,2})) \\
\vdots & & & \\
Q_i & \leftrightarrow & P_{i,1} \land P_{i,2} & (\neg Q_i \lor (P_{i,1} \land P_{i,2})) \land (Q_i \lor \neg(P_{i,1} \land P_{i,2})) \\
\vdots & \vdots & & \\
Q_n & \leftrightarrow & P_{n,1} \land P_{n,2} & (\neg Q_n \lor (P_{n,1} \land P_{n,2})) \land (Q_n \lor \neg(P_{n,1} \land P_{n,2})) \\
Q_{n+1} & \leftrightarrow & Q_1 \lor Q_2 & (\neg Q_{n+1} \lor Q_1 \lor Q_2) \land (Q_{n+1} \lor \neg(Q_1 \lor Q_2)) \\
Q_{n+2} & \leftrightarrow & Q_{n+1} \lor Q_3 & (\neg Q_{n+2} \lor Q_{n+1} \lor Q_3) \land (Q_{n+2} \lor \neg(Q_{n+1} \lor Q_3)) \\
\vdots & \vdots & & \\
Q_{2n-1} & \leftrightarrow & Q_{2n-2} \lor Q_n & (\neg Q_{2n-1} \lor Q_{2n-2} \lor Q_n) \land (Q_{2n-1} \lor \neg(Q_{2n-2} \lor Q_n)) \\
Q_{2n-1} & & & Q_{2n-1}
\end{array}
$$

$$3.Schritt$$

$(\neg Q_1 \vee P_{1,1}) \wedge (\neg Q_1 \vee P_{1,2}) \wedge (Q_1 \vee \neg P_{1,1}) \wedge (Q_1 \vee P_{1,2})$

$\vdots$

$(\neg Q_n \vee P_{n,1}) \wedge (\neg Q_n \vee \neg P_{n,2}) \wedge (Q_n \vee \neg P_{n,1}) \wedge (Q_n \vee \neg P_{n,2})$

$(\neg Q_{n+1} \vee Q_1 \vee Q_2) \wedge (Q_{n+1} \vee \neg Q_1) \wedge (Q_{n+1} \vee \neg Q_2)$

$(\neg Q_{n+2} \vee Q_{n+1} \vee Q_3) \wedge (Q_{n+2} \vee \neg Q_{n+1}) \wedge (Q_{n+2} \vee \neg Q_3)$

$\vdots$

$(\neg Q_{2n-1} \vee Q_{2n-2} \vee Q_n) \wedge (Q_{2n-1} \vee \neg Q_{2n-2}) \wedge (Q_{2n-1} \vee \neg Q_n)$

Q_{2n-1}

Im vorangegangenen Beispiel 2.41 wurde die Idee der kurzen KNF deutlich. Hier folgt jetzt die Definition im allgemeinen Fall.

Definition 2.42 (Kurze konjunktive Normalform)
Sei A eine aussagenlogische Formel in Negationsnormalform.
Es ist geschickter für diese Definition und den nachfolgenden Satz die neuen Atome Q nicht mit Zahlen zu indizieren sondern als unterscheidende Indizes die Teilformeln zu benutzen, für die sie als Abkürzung dienen sollen.
Sie als TF_A die Menge alle Teilformeln von A, die nicht Literale sind. Für jedes $F \in TF_A$ führen wir eine neues aussagenlogisches Atom Q_F ein. Wie im Beispiel beschreiben wir die Konstruktion in drei Schritten.

1.Schritt K sei die Konjunktion der folgenden Formeln

$$Q_F \leftrightarrow B_F^1 \circ B_F^2$$

für $F \in TF_A$ mit $F = F^1 \circ F^2$ und $B_F^i = \begin{cases} F^i & \text{falls } F^i \text{ ein Literal ist, und} \\ Q_{F^i} & \text{sonst} \end{cases}$
Schließlich wird noch die Einerklausel Q_A der Konjunktion hinzugefügt.
B_F^i ist also auf jeden Fall ein Literal, ein altes oder ein neues.

2.Schritt Jedes Konjunktionsglied $Q_F \leftrightarrow B_F^1 \circ B_F^2$ in K wird ersetzt durch die logisch äquivalente Konjunktion

$$(\neg Q_F \vee (B_F^1 \circ B_F^2)) \wedge (Q_F \vee \neg(B_F^1 \circ B_F^2))$$

3.Schritt Die Konjunktionsglieder $(\neg Q_F \vee (B_F^1 \circ B_F^2))$ und $(Q_F \vee \neg(B_F^1 \circ B_F^2))$ werden durch ihre konjunktive Normalform ersetzt. Im Detail treten die beiden Fälle auf.
$\circ = \wedge$

$(\neg Q_F \vee (B_F^1 \wedge B_F^2))$ wird ersetzt durch $(\neg Q_F \vee B_F^1) \wedge (\neg Q_F \vee B_F^2))$ und
$(Q_F \vee \neg(B_F^1 \wedge B_F^2))$ durch $(Q_F \vee \neg B_F^1) \wedge (Q_F \vee B_F^2)$.
$\circ = \vee$
$(\neg Q_F \vee (B_F^1 \vee B_F^2))$ bleibt $(\neg Q_F \vee B_F^1 \vee B_F^2)$
$(Q_F \vee \neg(B_F^1 \vee B_F^2))$ wird ersetzt durch $(Q_F \vee \neg B_F^1) \wedge (Q_F \vee \neg B_F^2)$.

Da alle B^i Literale und die Q's sowie so Atome sind haben wir eine konjunktive Normalform A_{kKNF} erreicht.

Wir hätten natürlich Schritt 2 und 3 zu einem zusammenfassen können. Es ist aber übersichtlicher sie auseinander zu halten.

Als Maß für die Größe einer Formel A wollen wir die Anzahl der Literalvorkommen $L(A)$ in A betrachten. Die Konjunktion K in Schritt 1 enthält so viele Konjunktionsglieder wie es Vorkommen der zweistelligen Operatoren in A gibt. Das sind $L(A) + 1$ viele (davon überzeugt man sich wie in Übungsausgabe 2.2.3). Jede dieser Konjunktion wird in Schritt 2 und 3 durch ein Formel mit höchstens 8 Literalvorkommen ersetzt und schließlich kommt noch die Einerklausel hinzu, insgesamt enthält A_{kKNF} also $8 * L(A) + 9$ Literalvorkommen.

Satz 2.43 (Kurze konjunktive Normalform)
Die Formel A ist erfüllbar gdw A_{kKNF} erfüllbar ist.

Beweis: Es genügt zu zeigen, daß A genau dann erfüllbar ist, wenn die in Schritt 1 erhaltene Konjunktion K erfüllbar ist, da alle weiteren Umformungen die logische Äquivalenz erhalten.

Ist A durch die Interpretation I erfüllt, d.h. $val_I(A) = \mathbf{W}$, dann setzen wir I zu einer Interpretation J fort, indem wir für die neuen Atome $J(Q_F) = val_I(F)$ definieren. Offensichtlich gilt $val_J(K_i) = \mathbf{W}$ für alle Konjunktionsglieder K_i von K.

Sei jetzt J eine Interpretation mit $val_I(K) = \mathbf{W}$. Sei I die Einschränkung von J auf die Atome, die in A vorkamen. Man zeigt für jede Teilformel F von A durch strukturelle Induktion $val_J(Q_F) = val_I(F)$. Da die Einerklausel Q_A in K vorkommt, gilt insbesondere $val_J(Q_A) = val_I(A) = \mathbf{W}$.

∎

Die konjunktive Normalform spielt auch in dem weiter unten vorzustellenden Resolutionskalkül eine wichtige Rolle. Will man in diesem Kalkül zeigen, daß eine Formel A eine Tautologie ist, so bringt man die Formel $\neg A$ in konjunktive Normalform und zeigt ihre Unerfüllbarkeit.

2.4.4 Shannon Formeln

Wie in den Abschlußbemerkungen zu Unterkapitel 2.2 angekündigt wollen wir in diesem Unterkapitel aussagenlogische Formeln mit einem dreistelligen Operator betrachten. Das aktuelle Unterkapitel wird für das nachfolgende über Shannon Graphen nicht benötigt. Der eilige Leser kann direkt zu Unterkapitel 2.4.5 vorspringen.

Definition 2.44 (Shannon-Formeln)
Wir betrachten aussagenlogische Formeln, die aufgebaut sind aus

- dem dreistelligen Operator sh
- den Konstanten 0 und 1
- Aussagevariablen $P_1, \dots, P_n, \dots$

Wir wollen solche Formeln Shannon-Formeln nennen und bezeichnen die Menge aller Shannon-Formeln mit $shFor0$. Das ist unsere eigene Terminologie.

Die Auswertung einer Formel der Form $sh(A_1, A_2, A_3)$ bezüglich einer Interpretation I wird gegeben durch

$$val_I(sh(A_1, A_2, A_3)) = \begin{cases} val_I(A_2) & \text{falls} \quad val_I(A_1) = \mathbf{F} \\ val_I(A_3) & \text{falls} \quad val_I(A_1) = \mathbf{W} \end{cases}$$

oder in Tabellenform:

P_1	**W**	**W**	**W**	**W**	**F**	**F**	**F**	**F**
P_2	**W**	**W**	**F**	**F**	**W**	**W**	**F**	**F**
P_3	**W**	**F**	**W**	**F**	**W**	**F**	**W**	**F**
$sh(P_1, P_2, P_3)$	**W**	**F**	**W**	**F**	**W**	**W**	**F**	**F**

Lemma 2.45
Es gelten die folgenden Äquivalenzen:

1. $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \wedge P_2) \vee (P_1 \wedge P_3)$
2. $sh(P_1, P_2, P_3) \leftrightarrow (\neg P_1 \rightarrow P_2) \wedge (P_1 \rightarrow P_3) \leftrightarrow (P_1 \vee P_2) \wedge (\neg P_1 \vee P_3)$
3. $sh(0, P_2, P_3) \leftrightarrow P_2, sh(1, P_2, P_3) \leftrightarrow P_3$
4. $sh(P, 0, 1) \leftrightarrow P$

5. $sh(P, 1, 0) \leftrightarrow \neg P$

6. $sh(P_1, P_2, P_2) \leftrightarrow P_2$

7. $sh(sh(P_1, P_2, P_3), P_4, P_5) \leftrightarrow sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$

8. $A \leftrightarrow sh(P, A_{P=\mathbf{0}}, A_{P=\mathbf{1}})$

Hierbei ist $A_{P=\mathbf{0}}$ die Formel, die aus A entsteht, indem jedes Vorkommen der Aussagenvariablen P durch die Konstante $\mathbf{0}$ ersetzt wird. Analog ist $A_{P=\mathbf{1}}$ zu verstehen.

Beweis
zu 1 Zeigt man durch Vergleich der Wahrheitstafeln.
zu 2 Zeigt man am besten durch Fallunterscheidung.
Fall $I(P_1) = \mathbf{W}$

$$\begin{array}{rcl} val_I(P_1 \vee P_2) \wedge (\neg P_1 \vee P_3)) & = & val_I(\mathbf{1} \vee P_2) \wedge (\mathbf{0} \vee P_3)) \\ & = & val_I(P_3) \\ & = & val_I(sh(\mathbf{1}, P_2, P_3)) \\ & = & val_I(sh(P_1, P_2, P_3)) \end{array}$$

Fall $I(P_1) = \mathbf{F}$

$$\begin{array}{rcl} val_I(P_1 \vee P_2) \wedge (\neg P_1 \vee P_3)) & = & val_I(\mathbf{0} \vee P_2) \wedge (\mathbf{1} \vee P_3)) \\ & = & val_I(P_2) \\ & = & val_I(sh(\mathbf{0}, P_2, P_3)) \\ & = & val_I(sh(P_1, P_2, P_3)) \end{array}$$

(3) bis (6) und (8) sind offensichtlich.
zu 7 siehe Übungsaufgabe 2.4.14.

Definition 2.46 (Normierte Shannon-Formeln)
Wir fixieren eine Ordnung auf der Menge der Aussagevariablen, etwa die durch die Ordnung der Indizes gegebene. Eine Shannon-Formel A heißt eine *normierte Shannon-Formel* (wenn wir genauer sein wollen, sagen wir eine bzgl. der fixierten Ordnung der Aussagevariablen normierte Shannon-Formel), wenn sie der folgenden rekursiven Definition genügt.

1. Die Konstanten $0, 1$ sind normierte Shannon-Formeln.

2. Sei P_i eine Aussagevariable, seien A, B normierte Shannon-Formeln, so daß keine Aussagevariable P_j mit $i \geq j$ in A oder B auftritt. Dann ist $sh(P_i, A, B)$ eine normierte Shannon-Formel.

Mit dieser Definition meinen wir, wie üblich, daß die Menge der normierten Shannon-Formeln die kleinste Menge ist, die 1. und 2. erfüllt.

Man beachte, daß P_i selbst, für ein aussagenlogisches Atom, nach dieser Definition keine normierte Shannon-Formel ist, wohl aber $sh(P_i, 0, 1)$. Ebenso ist $\neg P_i$ keine normierte Shannon-Formel, wohl aber $sh(P_i, 1, 0)$.

Lemma 2.47
Zu jeder aussagenlogischen Formel A gibt es eine äquivalente normierte Shannon-Formel B.

Beweis: Induktion nach der Anzahl n der in A vorkommenden Aussagevariablen.

Für $n = 0$ kann A logisch äquivalent auf eine der Konstanten 0 oder 1 reduziert werden. Konstanten sind normierte Shannon-Formeln.

Im Induktionsschritt wählen wir die in A vorkommende Aussagevariable P_i mit dem kleinsten Index. Mit A_0 bezeichnen wir die Formel, die aus A entsteht, indem jedes Vorkommen von P_i durch 0 ersetzt wird. Entsprechend wird A_1 gebildet. Nach Induktionsvoraussetzung gibt es normierte Shannon-Formeln B_0, B_1, die logisch äquivalent (siehe Def. 2.21) sind zu A_0, A_1. Offensichtlich ist A äquvalent zu $sh(P_i, B_0, B_1)$, und $sh(P_i, B_0, B_1)$ ist eine normierte Shannon-Formel.

∎

2.4.5 Shannon Graphen

Neben der DNF und KNF gibt es noch eine Vielzahl von Normalformen. In Übungsaufgabe 2.4.18 wird z.B. die Reed-Muller-Normalform vorgestellt. In diesem Unterabschnitt gehen wir auf eine von R. E. Bryant in [Bry86] behandelte graphbasierte Normalform eingehen. Sie geht zurück auf einen Ansatz von C. E. Shannon (siehe [Sha38]) und wurde außerdem schon in dem Buch [Chu56] von A. Church in §24, Seite 129 ff behandelt.

Definition 2.48
Ein *sh-Graph* ist ein gerichteter, binärer, zusammenhängender Graph.

1. Jedem nichtterminalen Knoten v ist eine natürliche Zahl $index(v)$ zugeordnet.

2. Von jedem nichtterminalen Knoten v gehen zwei Kanten aus. Eine davon ist mit **F**, die andere mit **W** gekennzeichnet.

3. Jedem terminalen Knoten v ist eine der Zahlen **F** oder **W** zugeordnet, bezeichnet mit $wert(v)$.

4. Ist der nichtterminale Knoten w ein unmittelbarer Nachfolger von v, dann gilt $index(v) < index(w)$.

5. Es gibt genau einen Wurzelknoten.

Wegen der vierten Forderung ist jeder sh-Graph zyklenfrei.

Jedem sh-Graphen G wird eine m-stellige Boolesche Funktion f_G zuordnen, wobei m die Anzahl der in G vorkommenden verschiedenen Indizes $i_1, \ldots, i_m$ ist. Wir fassen f_G als eine Funktion mit den Eingabevariablen $P_{i_1}, \ldots, P_{i_m}$ auf und bestimmen den Funktionswert $f_G(P_{i_1}, \ldots, P_{i_m})$, indem wir an der Wurzel von G beginnend einen Pfad durch G wählen. Am Knoten v folgen wir der Kante **F**, wenn die Eingabevariable $P_{index(v)}$ den Wert F hat, sonst der Kante **W**. Ist der Wert des terminalen Knotens **W**, dann der gesuchte Funktionswert W sonst F. Das Beispiel in Abb. 2.5 zeigt für den dort angegeben Graphen G_{ex}, die Berechnung der drei-stelligen Booleschen Funktion $f_{G_{ex}}$ für das Argumenttriple $(\mathbf{F}, \mathbf{W}, \mathbf{W})$. Folgt man wie beschrieben, dem in der Abbildung rot gefärbten Pfad von der Wurzel, so endet man in grünen Blattknoten. Damit hat man $f_{G_{ex}}(\mathbf{F}, \mathbf{W}, \mathbf{W}) = \mathbf{F}$ gefunden.

Das führt zu der folgenden induktiven Definition:

Definition 2.49 (f_G)
Sei G ein sh-Graph, $i_1, \ldots, i_m$ seien die in G auftretenden Indizes mit dem kleinsten beginnend der Größe nach geordnet.
Die von G induzierte Boolesche Funktion $f_G : \{W, F\}^m \rightarrow \{W, F\}$ ist definiert durch:

1. $f_G = F$ falls G aus dem einzigen Knoten v besteht, mit $wert(v) = \mathbf{F}$. In diesem, wie im nächsten Fall, ist notwendigerweise $m = 0$ und f_G eine Konstante.

2. $f_G = W$ falls G aus dem einzigen Knoten v besteht, mit $wert(v) = \mathbf{W}$,

3. $f_G(b_1, \ldots, b_m) = \begin{cases} f_{G_\mathbf{F}}(b_2, \ldots, b_m) & \text{falls } b_1 = F \\ f_{G_\mathbf{W}}(b_2, \ldots, b_m) & \text{falls } b_1 = W \end{cases}$

   Hierbei ist $G_\mathbf{X}$ der Teilgraph von G, der mit dem Knoten beginnt, der von der Wurzel aus entlang der mit **X** markierten Kante erreicht wird.

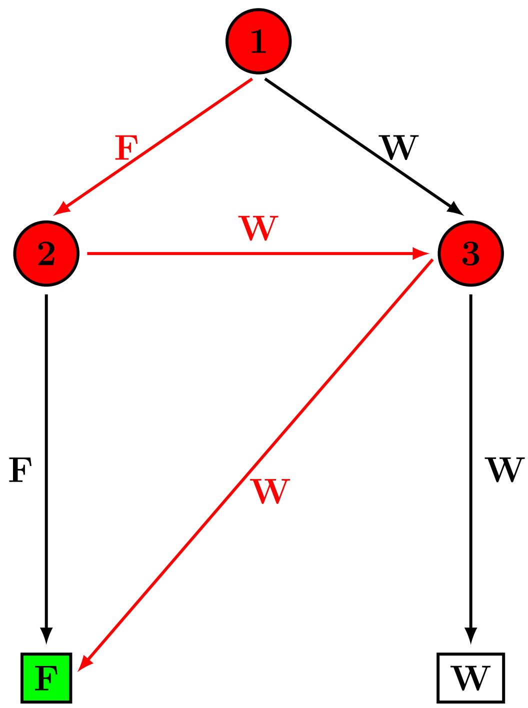


Abbildung 2.5: Berechnung von $f_{G_{ex}}(F, W, W)$

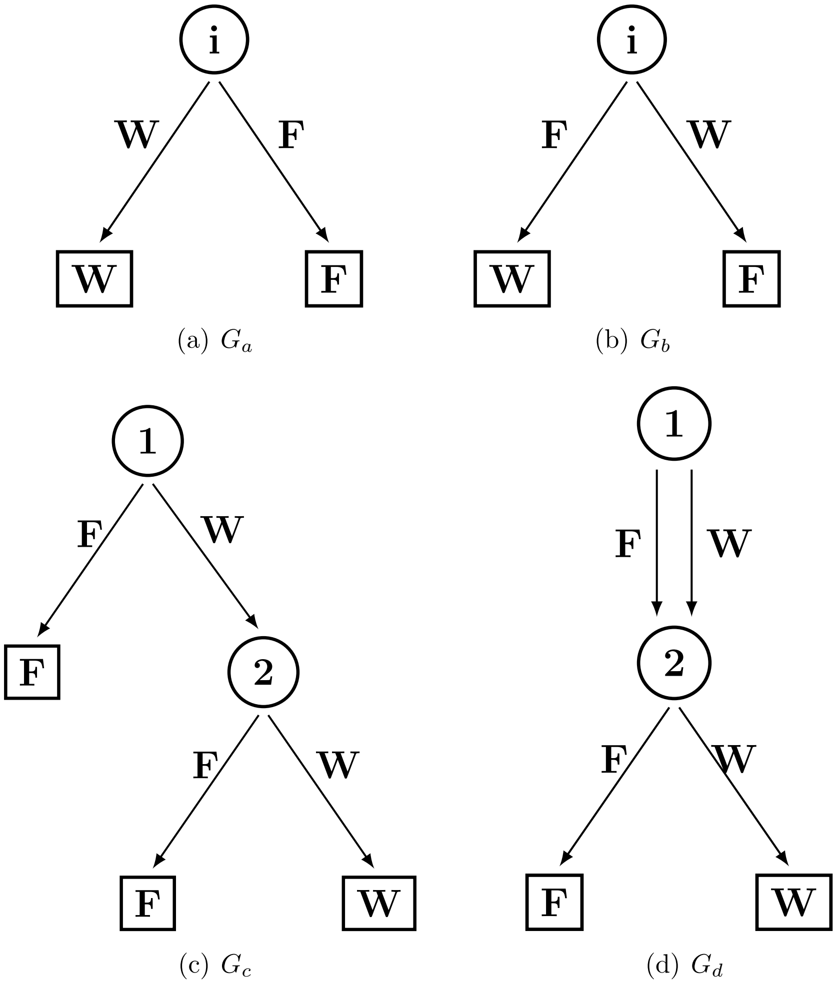


Abbildung 2.6: Beispiele von sh-Graphen

Abbildung 2.6 zeigt einige weitere Beispiele von sh-Graphen. Die zugehörigen Booleschen Funktionen sind $f_{G_a}(P_i) = P_i$, $f_{G_b}(P_i) = \neg P_i$, $f_{G_c} = f_{P_1 \wedge P_2}$, $f_{G_d}(P_1, P_2) = P_2$.

Lemma 2.50
Zu jeder Booleschen Funktion $f : \{W, F\}^m \rightarrow \{W, F\}$ und jeder aufsteigenden Folge $i_1 < \ldots < i_m$ von Indizes gibt es einen sh-Graphen G mit

$$f_G = f.$$

Beweis: Der Beweis geschieht durch Induktion über m.

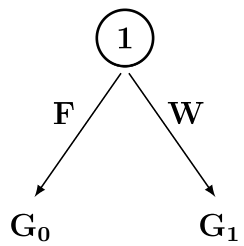
Ist $m = 0$ und f die konstante Funktion mit Wert F, dann besteht G nur aus einem Knoten v, der dann notwendigerweise ein terminaler Knoten ist, mit $wert(v) = \mathbf{F}$.

Ist $m = 0$ und f die konstante Funktion mit Wert W, dann besteht G nur aus einem Knoten v mit $wert(v) = \mathbf{W}$.

Sei jetzt $m > 0$. Wir definieren die m-1 stelligen Funktionen f_0 und f_1 durch

$$f_i(b_2, \dots, b_m) = \begin{cases} f(F, b_2, \dots, b_m) & \text{falls } i = 0 \\ f(W, b_2, \dots, b_m) & \text{falls } i = 1 \end{cases}$$

Nach Induktionsvoraussetzung gibt es sh-Graphen G_i mit $f_{G_i} = f_i$ für $i = 0, 1$. Sei G der sh-Graph



dann gilt offensichtlich $f_G = f$.

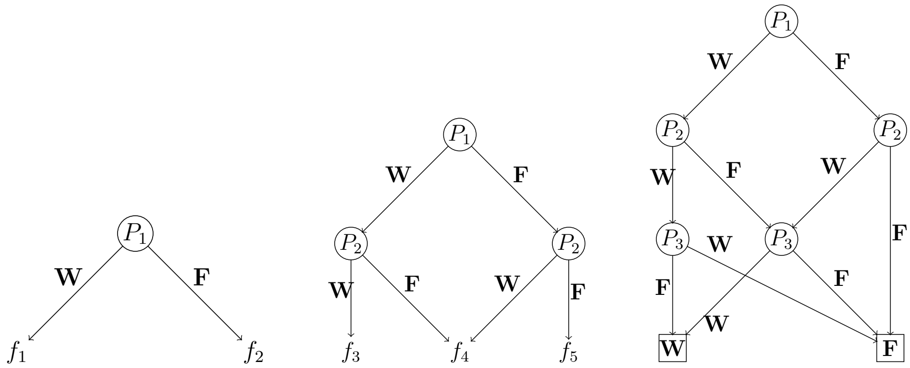
■

Beispiel 2.51
Wir betrachten die Boolesche Funktion f

$$f(P_1, P_2, P_3) = \begin{cases} W & \text{falls } \operatorname{card}(\{1 \leq i \leq 3 \mid P_i = W\}) = 2 \\ F & \text{sonst} \end{cases}$$

Wir wollen einen Shannongraph G konstruieren mit $f = F_G$. Die Variablenordnung sei $P_1 < P_2 < P_3$. Wir beginnen mit dem Knoten für P_1, siehe Abbildung 2.7(a). Dabei steht f_1, f_2 für die Funktion, die aus f entsteht, wenn man $P_1 = W$, $P_1 = F$ in Rechnung stellt. Wir schreiben dafür $f_1 = f_{P_1=W}$ und $f_2 = f_{P_1=F}$.

$$\begin{aligned} f_1(P_2, P_3) &= \begin{cases} W & \text{falls } (P_2 = W \text{ und } P_3 = F) \text{ oder} \\ & (P_2 = F \text{ und } P_3 = W) \\ F & \text{sonst} \end{cases} \\ f_2(P_2, P_3) &= \begin{cases} W & \text{falls } P_2 = P_3 = W \\ F & \text{sonst} \end{cases} \end{aligned}$$



(a) Erster Schritt (b) Zweiter Schritt (c) Endergebnis

Abbildung 2.7: Konstruktionsschritte in Beispiel 2.51

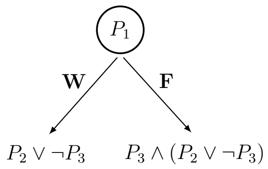
Der Anfang ist gemacht. Jetzt müssen nur noch in Abbildung 2.7(a) die Funktionen f_1, f_2 durch ihre Shannongraphen ersetzt werden. Das führt zu dem Graphen in Abbildung 2.7(b) wobei

$$\begin{array}{rcl}(f_1)_{P_2=W}(P_3) = f_3(P_3) & = & \neg P_3 \\ (f_1)_{P_2=F}(P_3) = f_4(P_3) & = & P_3 \\ (f_2)_{P_2=F}(P_3) = f_5(P_3) & = & F \\ (f_2)_{P_2=W}(P_3) = f_6(P_3) & = & P_3\end{array}$$

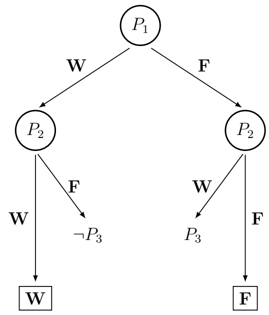
Die Tatsache, daß $f_4 = f_6$ ist wurde in Abbildung 2.7(b) schon benutzt um einen Knoten einzusparen.

Im letzten Schritt müssen die Funktionen f_3 bis f_5 durch ihre Shannongraphen ersetzt werden. Das Endergebnis ist in Abbildung 2.7(c) zu sehen

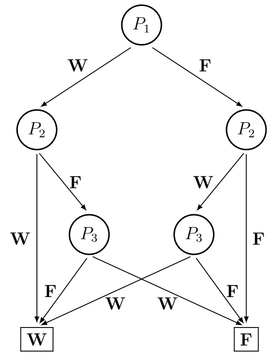
Beispiel 2.52
Als ein weiteres Beispiel wollen wir einen Shannon Graphen G_2 konstruieren zur Formel $F_2 = (P_1 \vee P_3) \wedge (P_2 \vee \neg P_3)$. Die Funktion f_{G_1} soll also übereinstimmen mit f_{F_2}. Wir benutzen dieselbe rekursive Herangehensweise wie im vorangegangen Beispiel 2.51. Im ersten Schritt, siehe Abbildung 2.8(a), wird das Problem aufgespalten in die Konstruktion eines Graphen G_{21} für $(1 \vee P_3) \wedge (P_2 \vee \neg P_3) \leftrightarrow (P_2 \vee \neg P_3)$, was sich logisch äquivalent vereinfachen läßt zu $P_2 \vee \neg P_3$, und eines Graphen G_{20} für $(0 \vee P_3) \wedge (P_2 \vee \neg P_3) \leftrightarrow P_3 \wedge (P_2 \vee \neg P_3)$, was sich vereinfachen läßt zu $P_3 \wedge (P_2 \vee \neg P_3)$. Der Gesamtgraph G_2 wird erhalten, indem ausgehend vom Wurzelknoten, der mit P_1 markiert ist, G_{21} an die **W**-Kante und G_{20} an die **F**-Kante *angehängt* wird. Im zweiten Schritt, siehe Abbildung 2.8(b), wird die Konstruktion von G_{21} weiter zerlegt in die Konstruktion eines Shannon Graphen für $1 \vee \neg P_3 \leftrightarrow 1$



(a) Erster Schritt



(b) Zweiter Schritt



(c) Endergebnis

Abbildung 2.8: Konstruktionsschritte in Beispiel 2.52

und $0 \vee \neg P_3 \leftrightarrow \neg P_3$. Die Konstruktion von G_{20} wird rekursiv zerlegt in die Fälle $P_3 \wedge (1 \vee \neg P_3) \leftrightarrow P_3$ und $P_3 \wedge (0 \vee \neg P_3) \leftrightarrow P_3 \wedge \neg P_3 \leftrightarrow 0$.

Das Endergebnis ist in Abbildung 2.8(c) zu sehen

An den Beispielgraphen in Abbildung 2.6 kann man sehen, daß die Definition von *sh*-Graphen Redundanz erlaubt. In Graph 2.6(c) könnte man einen mit **F**markierten Terminalknoten einsparen und in 2.6(d) könnte man ohne die induzierte Funktion zu ändern den Wurzelknoten samt der beiden von ihm ausgehenden Kanten einsparen. Wir werden jetzt systematisch Reduktionsmöglichkeiten für *sh*-Graphen untersuchen.

Als Vorbereitung dazu müssen wir zunächst über die Isomorphie von Graphen reden.

Definition 2.53 (Isomorphie von Shannongraphen)
Seien zwei *sh*-Graphen H, G gegeben. Ihre Knotenmengen seien V_1, V_2. H, G heißen zueinander *isomorph* ($H \cong G$) genau dann, wenn es eine bijektive Abbildung π von V_1 nach V_2 gibt mit:

1. $index(k) = index(\pi(k))$ für jeden Nichtterminalknoten $k \in V_1$
2. $wert(k) = wert(\pi(k))$ für jeden Terminalknoten $k \in V_1$
3. Für jeden Nichtterminalknoten $k \in V_1$, dessen **F**-Kante/**W**-Kante zu dem Knoten k_0/k_1 führt, gilt: die **F**-Kante von $\pi(k)$ führt zu $\pi(k_0)$, die **W**-Kante zu $\pi(k_1)$.

Im simpelsten Fall besagt die Definition 2.53, daß es auf die Anordnung der Kanten des Graph G in der Zeichenebene nicht ankommt. So sind die

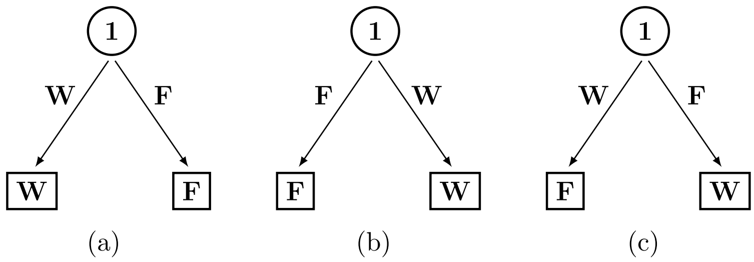


Abbildung 2.9: Einfachstes Beispiel isomorpher Shannon-Graphen

Graphen in 2.9(a) und 2.9(b) isomorph, nicht aber 2.9(a) und 2.9(c). Im Graphen in 2.9(a) führt die **W**-Kante von der Wurzel zum **W**-Terminalknoten, in 2.9(c) dagegen führt die **W**-Kante von der Wurzel zum **F**-Terminalknoten. Das ist aber der trivialste Fall isomorpher Graphen. Der wichtigere Fall wird in Abbildung 2.9 deutlich. Nennen wir den Knoten, der von der Wurzel aus mit der **F**-Kante erreicht wird für den Augenblick N_{11} und den Knoten, der von der Wurzel aus mit der **W**-Kante erreicht wird für den Augenblick N_{12}. Der Teilgraph G_{11} mit Wurzel N_{11} stimmt überein mit dem Teilgraph G_{12} mit Wurzel N_{12}. G_{11} und G_{12} sind unterschiedliche Vorkommen desselben Graphen im Gesamtgraphen. In unserer Terminologie sind G_{11} und G_{12} verschiedene aber isomorphe Teilgraphen.

[inner sep=0pt,outer sep=0pt]

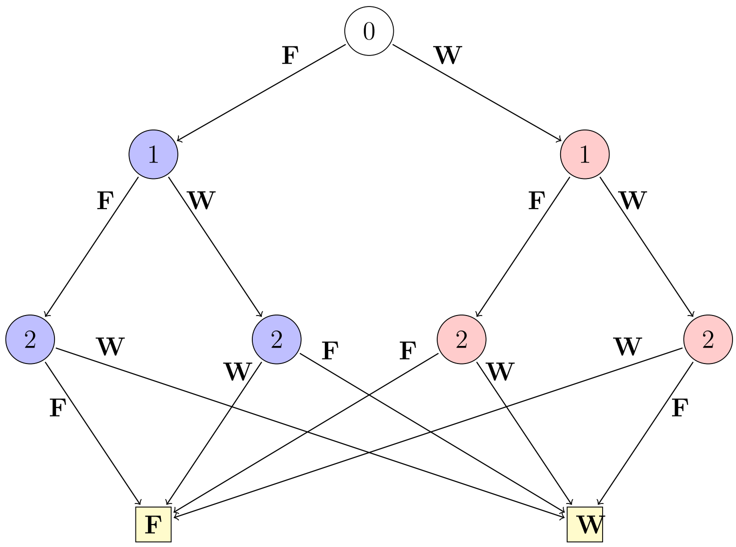


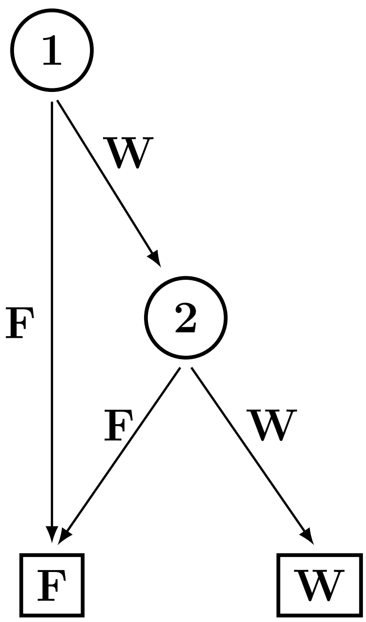
Abbildung 2.10: Beispiel isomorphe Shannon-Graphen

Wenn zwei sh-Graphen G_1, G_2 isomorph sind, dann ist die isomorphe Abbildung π eindeutig bestimmt: π muß den Wurzelknoten von G_1 auf den Wurzelknoten von G_2 abbilden, den **W** Nachfolger eines Urbildes auf den **W** Nachfolger des Bildes, usw.

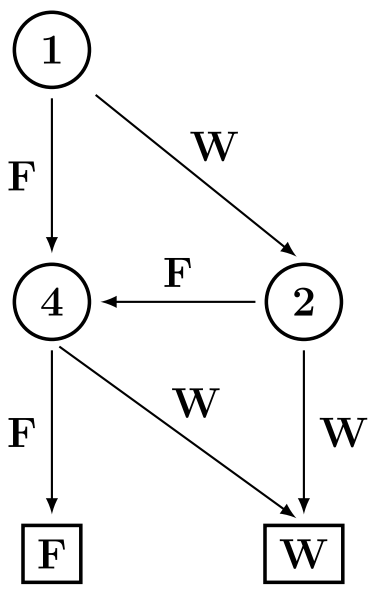
Definition 2.54
Ein sh-Graph heißt *reduziert*, wenn

1. es keine zwei Knoten v und w ($v \neq w$) gibt, so daß der in v verwurzelte Teilgraph G_v mit dem in w verwurzelten Teilgraph G_w isomorph ist.

2. es keinen Knoten v gibt, so daß beide von v ausgehenden Kanten zum selben Nachfolgerknoten führen.



(a)



(b)

Abbildung 2.11: Beispiele reduzierter sh-Graphen

Ein reduzierter Shannongraph heißt auch BDD bzw. OBDD *(ordered binary decision diagram)*.

Abbildung 2.11 zeigt zwei reduzierte sh-Graphen.

Der Graph in Abbildung 2.6(d) ist nicht reduziert weil die zweite Klausel in der Definition 2.54 verletzt ist.

Ein etwas komplexeres Beispiel eines nicht reduzierten sh-Graphen zeigt Abbildung 2.10. Wir hatten diese Abbildung schon auf Seite 59 betrachtet und den Knoten, der von der Wurzel aus mit der **F**-Kante erreicht wird N_{11} genannt und den Knoten, der von der Wurzel aus mit der **W**-Kante erreicht wird N_{12}. Der Teilgraph G_{11} mit Wurzel N_{11} ist isomorph zum Teilgraph G_{12} mit Wurzel N_{12}, aber $N_{11} \neq N_{12}$. Wir prüft man nun, ob ein Graph reduziert ist? Sucht man nach verschiedenen Knoten, so daß die dort beginnenden Teilgraphen isomorph sind, so kann das, wie das betrachtete Beispiel 2.10 zeigt recht kompliziert werden, besonders, wenn man das für alle Knotenpaare wiederholen muß. Ds folgende Kriterium schafft hier Linderung.

Lemma 2.55
Sei G ein Shannongraph, so daß für jedes Paar von Knoten v, w gilt

wenn	die **W**-Nachfolger von v und w gleich sind und
	die **F**-Nachfolger von v und w gleich sind
dann	$v = w$

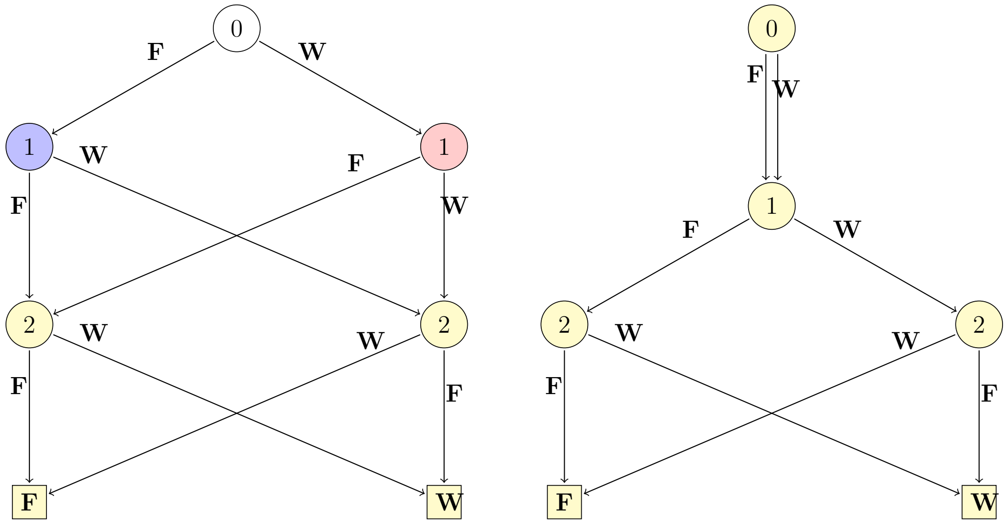
Dann erfüllt G die Bedingung (1) aus Definition 2.54.

Beweis: Seinen x, y zwei Knoten und π ein Isomorphismus von dem Teilgraphen G_x unterhalb von x auf den Teilgraphen G_y unterhalb von y. Wir müssen $x = y$ zeigen. Das geschieht durch Induktion nach der Anzahl der Knoten in G_x.

Sei $N_{x\mathbf{W}}$ der den Knoten am Ende der **W**-Kante von x und $G_{x\mathbf{W}}$ der Teilgraph von G_x mit Wurzel $N_{x\mathbf{W}}$. Ebenso sei $N_{x\mathbf{F}}$ der den Knoten am Ende der **F**-Kante von x und $G_{x\mathbf{F}}$ der Teilgraph von G_x mit Wurzel $N_{x\mathbf{F}}$. Analog seien $N_{y\mathbf{W}}$ $N_{y\mathbf{F}}$, $G_{y\mathbf{W}}$ und $G_{y\mathbf{F}}$ definiert. Die jeweiligen Einschränkungen von π liefern Isomorphismen von $G_{x\mathbf{W}}$ nach $G_{y\mathbf{W}}$ und ebenfalls von $G_{x\mathbf{F}}$ nach $G_{y\mathbf{F}}$. Nach Induktionsvorausetzung muß $N_{x\mathbf{W}} = N_{y\mathbf{W}}$ und $N_{x\mathbf{F}} = N_{y\mathbf{F}}$. Jetzt greift aber die Voraussetzung des Lemmas und liefert $x = y$, wie gewünscht.

∎

Lemma 2.55 liefert jetzt ein praktikables Verfahren um einen nicht reduzierten *sh*-Graphen, schrittweise, in einen reduzierte zu transformieren. Man sucht nach verschiedenen Knoten N_1, N_2, deren **W**-Nachfolger und **F**-Nachfolger jeweils übereinstimmen. Man reduziert den Graphen, indem man die beiden Knoten N_1, N_2 durch einen einzigen ersetzt und die Kanten entsprechend umlenkt. Abbildung 2.12 zeigt die schrittweise Reduktion des Graphen aus Abbildung 2.10. In den ersten beiden Schritten werden jeweils zwei der mit 2 markierten Knoten identifiziert. Das Ergebnis ist in 2.12(a) zu sehen. In einem weiteren Schritt werden die mit 1 markierten Knoten identifiziert, siehe Abbildung 2.12(a). In einem letzten Schritt, der nicht mehr abgebildet ist,



(a) nach 2 Reduktionsschritten

(b) nach 3 Reduktionsschritten

Abbildung 2.12: Reduktion des Shannongraphen aus Abb.2.10

wird der mit 0 markierte Wurzelknoten samt der beiden ausgehenden Kanten weggelassen.

Lemma 2.56 (Eindeutigkeit reduzierter *sh*-Graphen)
Sind G, H reduzierte sh-Graphen, dann gilt

$$f_G = f_H \Leftrightarrow G \cong H.$$

(Zu jeder Booleschen Funktion f gibt es bis auf Isomorphismus genau einen reduzierten sh-Graphen H mit $f = f_H$).

Beweis Die Idee des Existenzbeweises eines äquivalenten reduzierten sh-Graphen ist einfach: Man geht von einem entsprechenden sh-Graphen aus und beseitigt nach und nach alle Situationen, die der Reduziertheit entgegenstehen. Bei jedem solchen Schritt wird die Anzahl der Knoten um mindestens eins kleiner, so daß die Terminierung gesichert ist.

Beweis der Eindeutigkeit: Seien H und G zwei reduzierte sh-Graphen mit $f_H = f_G = f$. Wir müssen $H \cong G$ zeigen. Das geschieht durch Induktion über $k =$ die Länge des längsten Pfades in G und H.

$k = 0$:
In diesem Anfangsfall bestehen G und H nur aus einem Blatt, das zugleich

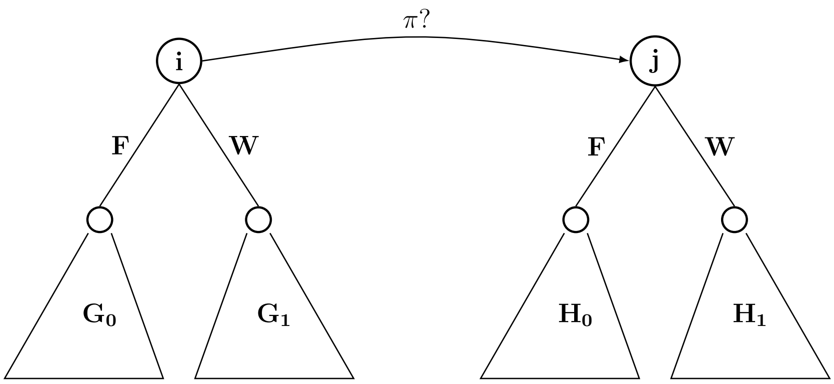


Abbildung 2.13: Konstruktion eines Isomorphismus zwischen G und h

Wurzel ist; d.h., G und H haben die Form $\boxed{\mathbf{F}}$ oder $\boxed{\mathbf{W}}$. f_H und f_G sind 0-stellige Funktionen, d.h. Konstanten und aus $f_H = f_G$ folgt, daß $H = G = \boxed{\mathbf{F}}$ oder $H = G = \boxed{\mathbf{W}}$.

$k > 0:$
Die Induktionsvoraussetzung garantiert für alle Graphen G', H' mit maximaler Pfadlänge $< k$ und $f_{G'} = f_{H'}$, daß G' und H' isomorph sind.

Es sei
$$\begin{array}{ll} v_0 \text{ Wurzel von } G, & index(v_0) = i \\ w_0 \text{ Wurzel von } H, & index(w_0) = j \end{array}$$

Siehe Abbildung 2.13

Wir zeigen i ist der kleinste Index, so daß f_G von P_i abhängt.

Zunächst argumentieren wir, daß f_G von P_i abhängt. Andernfalls wäre $f_{P_i=\mathbf{F}} = f_{P_i=\mathbf{W}}$. Sind G_0, G_1 die Teilgraphen von G, die über die **F**-Kante bzw. die **W**-Kante direkt unter v_0 hängen, so gilt $f_{G_0} = f_{P_i=\mathbf{F}} = f_{P_i=\mathbf{W}} = f_{G_1}$; da die Länge des längsten Pfades in G_0 und $G_1 < k$ ist, folgt nach Induktionsannahme $G_0 \cong G_1$. Das widerspricht der Reduziertheit von G, siehe Definition 2.54(1).

Da i überhaupt der kleinste Index in G ist, ist i auch die kleinste Zahl, so daß f_G von P_i abhängt.

Genauso zeigt man, daß $index(w_0) = j$ der kleinste Index ist, so daß f_H von P_j abhängt. Aus der Voraussetzung $f_G = f_H$ folgt somit $i = j$.

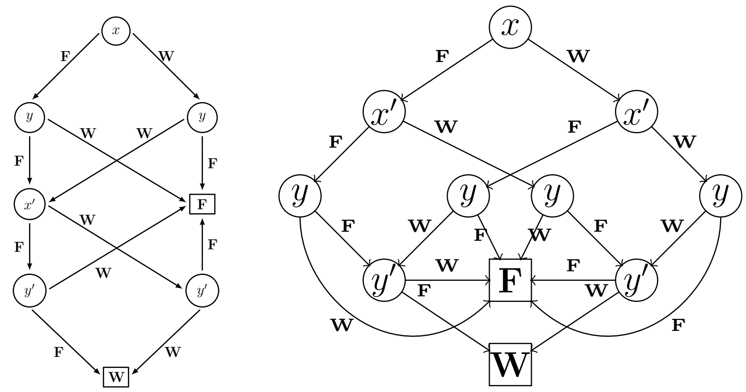
Man betrachte nun G_0, G_1, H_0, H_1, die Teilgraphen aus G bzw. H, die an den Knoten v_0 bzw. w_0 an den **F**-Kanten bzw. **W**-Kanten hängen. Aus $f_G = f_H = f$ folgt $f_{G_0} = f_{P_i=\mathbf{F}} = f_{H_0}$ und $f_{G_1} = f_{P_i=\mathbf{W}} = f_{H_1}$. Nach Induktionsannahme

gilt also $G_0 \cong H_0$ und $G_1 \cong H_1$.
Seien π_0, π_1 Isomorphismen $\pi_0 : G_0 \to H_0$, $\pi_1 : G_1 \to H_1$. Es gilt:
(*) Wenn v Knoten in G_0 und G_1 ist, dann $\pi_0(v) = \pi_1(v)$.
Angenommen, das wäre nicht wahr. Dann wählen wir ein v in $G_0 \cap G_1$, so daß $\pi_0(v) \neq \pi_1(v)$ gilt. Die in $\pi_0(v)$ und $\pi_1(v)$ verwurzelten Teilgraphen von H definieren dieselbe Funktion, wären also nach Induktionsannahme isomorph. Das widerspricht der Reduziertheit von H.
Definiere $\pi : G \to H$ durch

$$\pi(v) = \begin{cases} w_0 & \text{falls } v = v_0 \\ \pi_0(v) & \text{falls } v \text{ in } G_0 \\ \pi_1(v) & \text{falls } v \text{ in } G_1 \end{cases}$$

π ist injektiv: Begründung wie für (*).
π ist surjektiv: klar.
π erhält Kanten und Marken: klar.
Also ist π ein Isomorphismus und damit $G \cong H$.

∎



(a) Ordnung $x < y < x' < y'$

(b) Ordnung $x < x' < y < y'$

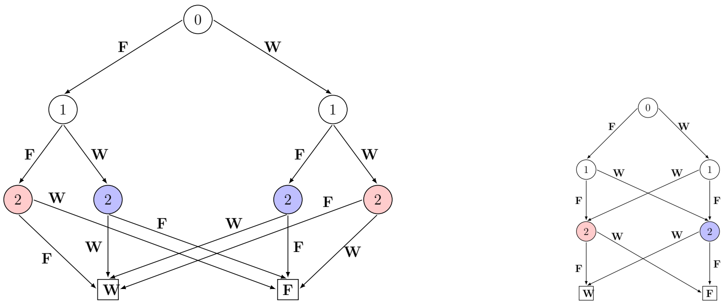
Abbildung 2.14: Shannon Graphen für $(x \leftrightarrow y) \wedge (x' \leftrightarrow y')$

Die Größe eines Shannongraphen für eine vorgegebene Boolesche Funktion hängt manchmal entscheidend von der Variablenordnung ab, wie man in Abbildung 2.14 sehen kann. Heuristiken für eine geschickte Wahl der Variablenordnung sind daher in vielen Anwendungen OBDDs von großer Wichtigkeit. Allerdings gibt es auch hart Nüsse, die bezüglich jeder Variablenordnung große Shannongrpahen liefern.

Beispiel 2.57
X sei die Menge der $2k$ Variablen $\{x_0, \ldots, x_{k-1}, y_0, \ldots, y_{k-1}\}$. $x = x_0 \ldots x_{k-1}$ und $y = y_0 \ldots y_{k-1}$ bezeichnen k-stellige Binärzahlen. Für $0 \leq i < 2k$ bezeichne $Mult_i$ die boolsche Funktion, die das i-te Bit des Produktes von x mit y beschreibt.

Lemma 2.58
Für jede Ordnung $<$ der Variablen in X gibt es einen Index $0 \leq i < 2k$, so dass der BDD $B_{Mult_i,<}$ mindestens $2^{k/8}$ Knoten besitzt.

`Zitat fehlt`



(a) Graph zu F_{bei} (b) Reduzierter Graph

Abbildung 2.15: Graph zu Formel

Bemerkung Es besteht eine offensichtliche Zuordnung zwischen normierten Shannon-Formeln und sh-Graphen.

Ist eine Formel F gegeben, so sind die Knoten des zugehörigen sh-Graphen G_F alle Vorkommen von aussagenlogischen Atomen in F plus die beiden Blattknoten für **W** und **F**. Sind P, Q zwei Knoten, also Vorkommen in F, dann gibt es eine **W**-Kante von P nach Q, wenn $sh(P, X, sh(Q, Y; Z)$ Teilformelvorkommen von F ist für geeignete X, Y, Z. Es gibt eine **F**-Kante von P nach Q, wenn $sh(P, sh(Q, Y; Z), X)$ Teilformelvorkommen von F ist für geeignete X, Y, Z. Für die normierte Shannon-Formel F_{bei} :

$$sh(A_0, sh(A_1, sh(A_2, \mathbf{1}, \mathbf{0}), sh(A_2, \mathbf{0}, \mathbf{1})), sh(A_1, sh(A_2, \mathbf{0}, \mathbf{1}), sh(A_2, \mathbf{1}, \mathbf{0})))$$

ergibt sich so der Graph in Abbildung 2.15(a).

Ist ein sh-Graph G gegeben, so ist die zugehörige normierte Shannon-Formel F_G induktiv definiert. Besteht G nur aus dem Blatt für **W** (**F**) dannist $F_G = \mathbf{1}$ ($F_G = \mathbf{0}$). Besteht G aus der Wurzel mit Markierung i und sind G_0, G_1 die

Graphen am Ende der **F**-, bzw **W**-Kante, so berechnet man induktiv die Formeln F_0 für G_0 und F_1 für G_1 und setzt $F = sh(P_i, F_0, F_1)$.

In Abbildung 2.15(b) ist der reduzierte Graph aus Abbildung2.15(a) zu sehen. Wendet man auf diesen das beschriebene Verfahren an, so erhält man wieder die ursprüngliche Formel F_{bei}. In der graphischen Darstellung ist es möglich ein Vorkommen eines Teilgraphen einzusparen, das ist in der linearen Formelnotation nicht möglich. Will man eine ähnliche Einsparung bei Formeln erreichen, kann man F_{bei} z.B. ersetzen durch

$$sh(A_0, sh(A_1, Y, X), sh(A_1, X, Y)) \land X \leftrightarrow sh(A_2, \mathbf{0}, \mathbf{1}) \land Y \leftrightarrow sh(A_2, \mathbf{1}, \mathbf{0})$$

Als nächstes wollen wir uns um die folgende, praktisch relevante, Frage kümmern:

> Seien G_i sh-Graphen für die Formeln F_i für $i = 1, 2$.
> Wie kann man einen sh-Graphen für $F_1 \land F_2$ erhalten?
> oder für andere Boolesche Kombinationen von F_1 und F_2?

Die theoretische Grundlagen für die Beantwortung dieser Frage läßt sich bequem in einer Aussagenlogik beschreiben, die neben dem 3-stelligen sh Operator auch noch die klassischen Operatoren $\neg$, $\land$, $\lor$, etc. enthält. Das wollen wir für den Rest dieses Unterabschnitts annehmen.

Lemma 2.59
Für beliebige Formeln $G_1 = sh(P, F_1^1, F_2^1)$, $G_2 = sh(Q, F_1^2, F_2^2)$ für beliebige Formeln F_j^i und und aussagenlogische Atomen P, Q und $\circ$ eine beliebige zweistellige Boolesche Operation gilt

1. $\neg G_1 \leftrightarrow sh(P, \neg F_1^1, \neg F_2^1)$
2. $G_1 \circ G_2 \leftrightarrow sh(P, F_1^1 \circ F_1^2, F_2^1 \circ F_2^2)$ falls $P \equiv Q$.
3. $G_1 \circ G_2 \leftrightarrow sh(P, F_1^1 \circ G_2, F_2^1 \circ G_2)$
4. $G_1 \circ G_2 \leftrightarrow sh(Q, F_1^2 \circ G_1, F_2^2 \circ G_1)$

Beweis Die Beweise sind einfach. Wir geben dennoch die notwendigen Argumente ausführlich an. Wir betrachten nur die Situation, daß $\circ = \land$. Alle anderen verlaufen vollkommen parallel.

zu 1

$$\begin{array}{rcll} \neg sh(P, F_1^1, F_2^1) & \leftrightarrow & \neg((P \land F_2^1) \lor (\neg P \land F_1^1)) & \textit{Lemma } 2.45(1) \\ & \leftrightarrow & (\neg P \lor \neg F_2^1) \land (P \lor \neg F_1^1) & \\ & \leftrightarrow & sh(P, \neg F_1^1, \neg F_2^1) & \textit{Lemma } 2.45(2) \end{array}$$

zu 2 nach Voraussetzung gilt $P \equiv Q$

$I(P) = \mathbf{W}$

$$\begin{array}{rcl} val_I(G_1 \wedge G_2) & = & val_I(G_1) \wedge val_I(G_2) \\ & = & val_I(sh(P, F_1^1, F_2^1)) \wedge val_I(sh(P, F_1^2, F_2^2)) \\ & = & val_I(F_2^1) \wedge val_I(F_2^2) \\ & = & sh(P, F_1^1 \wedge F_1^2, F_2^1) \wedge F_2^2) \end{array}$$

$I(P) = \mathbf{F}$

$$\begin{array}{rcl} val_I(G_1 \wedge G_2) & = & val_I(G_1) \wedge val_I(G_2) \\ & = & val_I(sh(P, F_1^1, F_2^1)) \wedge val_I(sh(P, F_1^2, F_2^2)) \\ & = & val_I(F_1^1) \wedge val_I(F_1^2) \\ & = & sh(P, F_1^1 \wedge F_1^2, F_2^1) \wedge F_2^2) \end{array}$$

zu 3

$I(P) = \mathbf{W}$

$$\begin{array}{rcl} val_I(G_1 \wedge G_2) & = & val_I(G_1) \wedge val_I(G_2) \\ & = & val_I(sh(P, F_1^1, F_2^1)) \wedge val_I(G_2) \\ & = & val_I(F_2^1) \wedge val_I(G_2) \\ & = & sh(P, F_1^1 \wedge G_2, F_2^1 \wedge G_2) \end{array}$$

$I(P) = \mathbf{F}$

$$\begin{array}{rcl} val_I(G_1 \wedge G_2) & = & val_I(G_1) \wedge val_I(G_2) \\ & = & val_I(sh(P, F_1^1, F_2^1)) \wedge val_I(G_2) \\ & = & val_I(F_1^1) \wedge val_I(G_2) \\ & = & sh(P, F_1^1 \wedge G_2, F_2^1 \wedge G_2) \end{array}$$

zu 4 Analog zu **3**.

■

2.4.6 Übungsaufgaben

Übungsaufgabe 2.4.1
Zeigen Sie, daß der im Beweis zu Lemma 2.31 angegebene Pseudoalgorithmus zur Berechnung der Negationsnormalform terminiert.

Übungsaufgabe 2.4.2
Zeigen Sie, daß der im Beweis zu Lemma 2.33 angegebene Pseudoalgorithmus zur Berechnung der konjunktiven Normalform terminiert.

Übungsaufgabe 2.4.3
Berechnen Sie die konjunktive und disjunktive Normalform der Formel $A \leftrightarrow (B \leftrightarrow C)$.

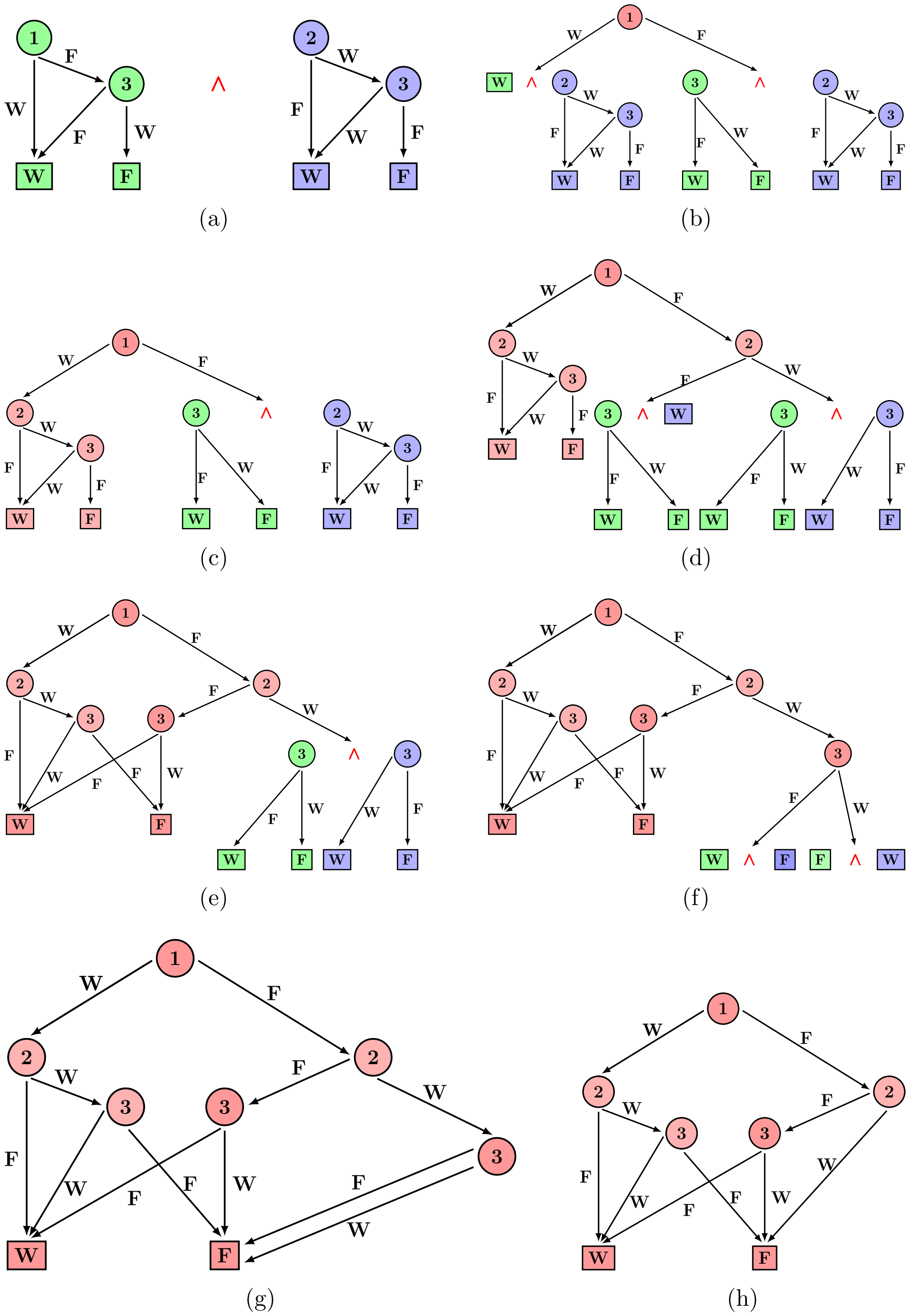


Abbildung 2.16: Konjunktion von *sh*-Graphen

Übungsaufgabe 2.4.4
Betrachten Sie die DNF

$$A = (\neg P \wedge Q \wedge \neg R) \vee (P \wedge Q \wedge \neg R) \vee (P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge R)$$

Zeigen Sie, daß die Normalformen

$$\begin{array}{rcl} A' & = & (Q \wedge \neg R) \vee (P \wedge R) \vee (\neg Q \wedge R) \\ A'' & = & (Q \wedge \neg R) \vee (P \wedge Q) \vee (\neg Q \wedge R) \end{array}$$

äquivalent zu A sind.

Übungsaufgabe 2.4.5
Der Begriff *Primimplikant* wurde in Definition 2.37 auf Seite 45 eingeführt.

Zeigen Sie: alle Disjunktionsglieder von

$$A_1 = (Q \wedge \neg R) \vee (P \wedge R) \vee (P \wedge Q) \vee (\neg Q \wedge R)$$

sind Primimplikanten der Formel A aus Aufgabe 2.4.4.

Übungsaufgabe 2.4.6
Zeigen Sie: Ist A eine DNF für B und ist A von minimaler (Zeichen-)Länge, dann sind alle Disjunktionsglieder von A Primimplikanden von B.

Übungsaufgabe 2.4.7
In Lemma 2.26 wurde das Craigsche Interpolationslemma formuliert und bewiesen.

Sei jetzt eine aussagenlogische Formeln A in disjunktiver Normalform $A = \bigvee_i \bigwedge_j A_{ij}$ und eine Formel B in konjunktive Normalform $B = \bigwedge_r \bigvee_s B_{rs}$ gegeben, so daß

$$A \rightarrow B$$

eine Tautologie ist. Finden Sie ein Verfahren um eine Formel C mit den Eigenschaften:

1. $A \rightarrow C$ und $C \rightarrow B$ sind Tautologien
2. In C kommen nur Atome vor, die sowohl in A als auch in B vorkommen.

direkt aus den Normalformen von A und B abzulesen.

W. Craig hat die in Lemma 2.26 formulierte Aussage als erster entdeckt und bewiesen, sogar für den prädikatenlogischen Fall. Der Beweis der hier genannten aussagenlogischen Variante ist wesentlich einfacher.

Übungsaufgabe 2.4.8
Für jedes n sei

$$C_n := \neg \overbrace{(P_1 \leftrightarrow \underbrace{(P_2 \leftrightarrow (P_3 \leftrightarrow \cdots (P_n \leftrightarrow \neg P_1) \cdots))}_{a_2})}^{a_1}$$

1. Ist C_n für jedes n eine Tautologie ?
2. Bringen Sie $\neg C_n$ in KNF.
3. Geben Sie $(C_n)_{kKNF}$ an.

Übungsaufgabe 2.4.9
Kann es eine kurze DNF geben ?

Übungsaufgabe 2.4.10
Sei n die Anzahl der Literalvorkommen in C.
Geben Sie in Abhängigkeit von n eine obere Schranke für die Anzahl der Literalvorkommen in C_{kKNF} an.

Übungsaufgabe 2.4.11
Finden Sie äquivalente normierte Shannon-Formeln zu:

1. P_i
2. $P_1 \wedge P_2$
3. $\neg P_i$
4. $P_1 \vee P_2$

Übungsaufgabe 2.4.12
1. Finde den reduzierten *sh*-Graphen für die von der Formel $(P_1 \wedge P_2) \vee P_4$ erzeugte Funktion.
2. Finde den reduzierten *sh*-Graphen für die folgende Funktion

$$f_0(P_1, P_2, P_3, P_4) = \begin{cases} 1 & \text{falls die Summe } P_1 + \ldots + P_4 \text{ ungerade ist} \\ 0 & \text{sonst} \end{cases}$$

Übungsaufgabe 2.4.13
Sei A eine normierte sh-Formel, B entstehe aus A durch Vertauschung der Konstanten 0 und 1. Zeigen Sie $\neg A \leftrightarrow B$.

Übungsaufgabe 2.4.14
Zeigen Sie:

$$sh(sh(P_1, P_2, P_3), P_4, P_5) \equiv sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$$

Übungsaufgabe 2.4.15
Sei G ein reduzierter Shannongraph. Jedem Pfad in G von der Wurzel zu einem Blattknoten kann in natürlicherweise eine Konjunktion von Literalen zugeordnet werden. Beschreiben wir einen Pfad als Folge von Indizes und Knotenmarkierungen, beispielsweise $1 - 0 - 3 - 1 - 4 - 0 - w$, so ist die zugeordnete Konjunktion $\neg P_1 \wedge P_3 \wedge \neg P_4$. Für jeden Pfad in G der zu dem Blattknoten 1 führt ist die so zugeordnete Konjunktion von Literalen offensichtlich ein Implikant für die durch G dargestellte aussagenlogische Formel. Man könnte vermuten, daß diese Implikant sogar ein Primimplikant ist. Stimmt das?

Übungsaufgabe 2.4.16
Berechnen Sie eine kurze konjunktive Normalform für die Formel

$$A = ((A_1 \wedge A_2) \vee (A_1 \wedge A_3)) \vee \neg(A_2 \wedge A_3)$$

Übungsaufgabe 2.4.17
Sei S eine Menge von Klauseln. Für ein Literal L sei $\bar{L}$ das zu L *komplementäre* Literal, d.h.

$$\bar{L} = \begin{cases} \neg A & \text{falls } L = A \\ A & \text{falls } L = \neg A \end{cases}$$

Ein Literal L heißt *isoliert* in S (im Englischen *pure literal in* S), wenn $\bar{L}$ in keiner Klausel in S vorkommt. Eine Klausel C in S heißt *isoliert*, wenn sie ein isoliertes Literal enthält.

Ist S eine unerfüllbare Klauselmenge und C eine isolierte Klausel in S, dann ist auch $S \setminus \{C\}$ unerfüllbar.

Übungsaufgabe 2.4.18
Diese Übungsaufgabe knüpft an Aufgaben 2.3.1 und 2.3.2 an.

1. Sei F eine Formel, die nur die Operatoren $\wedge, \oplus, 1$ benutzt. Dann läßt sich F logische äquivalent auf die folgende Form bringen

$$\bigoplus_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq r_i} A_{ij}$$

Dabei sind die ausgearteten Fälle $k = 0$ und $r_i = 0$ erlaubt. Die leere Summe steht dabei abkürzend für die Konstante 0, die leere Konjunktion für die Konstante 1. Außerdem kommt für jedes i jedes Atom höchstens einmal in $\{A_{i,1}, \dots A_{i,r_i}\}$ vor und für $1 \leq i_1, i_2 \leq k$ mit $i_1 \neq i_2$ sind F_{i_1} und F_{i_2} auch nach Änderung der Reihenfolge verschieden.

Diese Normalform ist unter den Bezeichnungen *Reed-Muller-Normalform* oder *Ringsummennormalform* seltener als *Schegalkinsches Polynom* bekannt.

2. Die Reed-Muller Normalform einer Formel ist eindeutig.

Übungsaufgabe 2.4.19

Definition 2.60
Eine Boolesche Funktion $f : \{\mathbf{W}, \mathbf{F}\}^n \to \{\mathbf{W}, \mathbf{F}\}$ heißt *symmetrisch* wenn ihr Wert nicht von der Reihenfolge der Argumente abhängt, d.h. $f(b_1, \dots, b_n) = f(c_1, \dots, c_n)$ falls $(c_1, \dots, c_n)$ eine Permutation von $(b_1, \dots, b_n)$ ist.

Da die b_i und c_i nur die beiden Werte $\mathbf{W}$ und $\mathbf{F}$ annehmen können ist $(c_1, \dots, c_n)$ eine Permutation von $(b_1, \dots, b_n)$ genau dann wenn unter den c_i der Wert $\mathbf{W}$ genau so häufig vorkommt, wie unter den b_i. Für $\bar{b} = (b_1, \dots, b_n) \in \{\mathbf{W}, \mathbf{F}\}^n$, bezeichne $\#_W(\bar{b})$, die Anzahl der in $\bar{b}$ vorkommenden $\mathbf{W}$, d.h. $\#_W(\bar{b}) = \#\{i \mid 1 \leq i \leq n, b_i = \mathbf{W}\}$. Eine n-stellige symmetrische Funktion f läßt sich somit vollständig beschreiben durch eine Menge $X \subseteq \{0, \dots, n\}$, so daß $f(\bar{b}) = \mathbf{W}$ genau dann wenn $\#_{\mathbf{W}}(\bar{b}) \in X$ gilt. Das führt zu folgender Notation f^n_X für n-stellige symmetrischer Boolscher Funktionen, die auch für die Lösung der Aufgabe nützlich sein kann.

$$f^n_X(\bar{b}) = \begin{cases} \mathbf{W} & \text{falls } \#_{\mathbf{W}}(\bar{(b)}) \in X \\ \mathbf{F} & \text{sonst} \end{cases}$$

Daraus folgt insbesondere, daß es 2^{n+1} symmetrische n-stellige Boolesche Funktionen gibt, während die Anzahl aller n-stellige Boolesche Funktionen 2^{2^n} ist.

Aufgabe:
Zeigen Sie, daß jede n-stellige symmetrische Boolesche Funktion f einen *sh*-Graphen G_f besitzt mit höchstens n^2 Knoten.

Übungsaufgabe 2.4.20
Finden Sie eine aussagenlogische Formel A, deren Boolesche Funktion f_A (siehe Definition 2.12) übereinstimmt mit der Booleschen Funktion $f_{G_{ex}}$ des Shannon Graphen G_{ex} aus Abbildung 2.5.

Übungsaufgabe 2.4.21
Wir nennen eine n-stellige Boolesche Funktion $f : \{\mathbf{W}, \mathbf{F}\}^n \rightarrow \{\mathbf{W}, \mathbf{F}\}$ *schwer fassbar* (engl. elusive) wenn für jede Variablenordnung in dem reduziert Shannon Graph G_f für f für jeden der beiden Blattknoten mindestens ein Pfad der Länge n existiert vom Wurzelknoten zum dem Blattknoten. Die Länge eines Pfades ist dabei die Anzahl der nicht-Blatt-Knoten.
Da es keinen Pfad der Länge $n+1$ geben kann ist das der komplizierteste Fall. Am anderen Ende des Spektrums steht die tautologische Funktion f_T, die für jedes Eintabetupel den Wert $\mathbf{W}$ hat. In dem zugehörigen Shannongraphen G_{f_T} hat jeder Pfad die Länge 0. Schwer fassbare Funktionen sind also solche, bei den die Reduktion des zugehörigen Shannon Graphen schon zu einer Verkleinerung des Graphen führen kann, aber nicht zu einer Verringerung der *Tiefe des Graphen.*

Zeigen Sie: Ist für eine Boolesche Funktion $f : \{\mathbf{W}, \mathbf{F}\}^n \rightarrow \{\mathbf{W}, \mathbf{F}\}$ die Anzahl der Eingabetupeln $\bar{b} \in \{\mathbf{W}, \mathbf{F}\}^n$ mit $f(\bar{b}) = \mathbf{W}$ ungerade, dann ist f schwer fassbar.

Kapitel 3

Aussagenlogik: Erfüllbarkeitstests

In diesem Kapitel wollen wir uns intensiv der Frage widmen:

> Ist eine Menge F aussagenlogischer Formeln erfüllbar?

Dieses Problem ist bekannt als das Erfüllbarkeitsproblem der Aussagenlogik. Im Englischen heißt es *satisfiability problem* oder kurz *SAT problem*. In diesem Kapitel wird das Erfüllbarkeitsproblem ausschließlich für endliche Formelmengen untersucht. Dann macht es keine Unterschied ob wir eine Menge von Formeln betrachten oder eine einzige Formel, die Konjunktion aller Elemente der Menge.

Die Frage nach der Allgemeingültigkeit einer Formel F kann zurückgeführt werden auf die Untersuchung der Erfüllbarkeit von $\neg F$.

Die Erfüllbarkeit einer Formel F ist einfach festzustellen: Man erstellt die Wahrheitstabelle für F und prüft, ob in der Ergebnisspalte wenigstens einmal der Wahrheitswert **W** vorkommt. Dann ist die Formel erfüllbar, andernfalls unerfüllbar. Der Haken dabei ist der Aufwand dieses Verfahrens, der exponentiell mit der Anzahl der aussagenlogischen Atome in F wächst. Es ist höchst unwahrscheinlich, daß sich ein exponentielles Wachstum vermeiden läßt, da das SAT Problem NP-vollständig ist, wie wir schon zu Ende des Unterkapitels 2.3.3 bemerkte hatten.

Vielleicht gibt es aber effizientere Erfüllbarkeitstest für Formeln in einer bestimmten Normalform? Ist $F = \bigvee_i K_i$ eine Formeln in DNF, dann ist F erfüllbar genau dann wenn es ein I gibt, so daß K_i erfüllbar ist und K_i ist erfüllbar, genau dann wenn es keine Atom P gibt, so daß P und $\neg P$ in K_i vorkommt. Das liefert einen Erfüllbarkeitstest in linearer Zeit, $O(n)$, wobei n die Länge der Eingabeformel ist. Dieses Problem hilft leider nicht viel, da die Transformation in DNF wieder von exponentielle Komplexität ist. (siehe auch Übungsaufgabe 3.5.1).

Wie steht es mit der KNF, der konjunktiven Normalform? Hierfür kennen wir ein Verfahren, daß mit quadratischem Aufwand eine beliebige Formeln in KNF transformiert.

Definition 3.1
Eine Formel F heißt eine n-KNF wenn F eine konjunktive Normalform ist $F = \bigwedge_i C_i$ und jede Klausel C_i höchstens n Literale enthält.

Besonders wichtig sind 3-KNF und 2-KNF. 2-KNF nennen wir auch Krom-Formeln.

Satz 3.2
1. Das Erfüllbarkeitsproblem für 3-KNF ist NP-vollständig.

2. Die Erfüllbarkeit von Krom-Formeln ist in polynomialer Zeit entscheidbar.

Beweis (1) wird in [Coo71] bewiesen, siehe auch [GJ79, p.48ff]. Zu (2) siehe Übungsaufgabe 5.3.3 auf Seite 175.

∎

Die Einschränkung auf konjunktive Normalformen, sogar 3-KNF führt also nicht zu einer Reduktion der Komplexität des Entscheidungsproblems. Wir werden in Unterkapitel 3.2 ein Formelfragment kennen lernen, das eine polynomiell entscheidbares Erfüllbarkeitsproblem hat. Zuerst werden wir aber ein Verfahren für uneingeschränkte konjunktive Normalformen präsentieren.

3.1 Davis-Putnam-Logemann-Loveland Verfahren

Das **Davis–Putnam–Logemann–Loveland** (DPLL) Verfahren ist das zur Zeit schnellste und fast ausschließlich benutzte Verfahren in implementierten Erfüllbarkeitstests.

Im Englischen nennt man solche Programme *SAT solver*. Inzwischen gibt es einen jährlichen Wettbewerb, *SAT competition* (siehe `http://www.satcompetition.org/`), in dem die verschiedenen SAT solver gegeneinander antreten. Alle benutzen sie Varianten des DPLL Verfahrens.

Das DPLL Verfahren benötigt als Eingabe eine KNF, also eine Konjunktion von Klauseln. Wobei Klauseln als Disjunktionen von Literalen definiert waren (Definition 2.32). Da es in einer Klausel nicht auf die Reihenfolge der Literale ankommt und auch das Weglassen mehrfach auftretender Literale die logische Äquivalenz erhält, wollen wir in diesem Unterkapitel Klauseln als Mengen von Literalen schreiben.

Dasselbe gilt für konjunktive Normalenformen selbst: es kommt nicht auf die Reihenfolge der darin auftretenden Disjunktionen an und das Weglassen mehrfach auftretender Klauseln erhält die logische Äquivalenz. Wir repräsentieren daher konjunktive Normalformen als Mengen von Klauseln, d.h. als Mengen von Mengen von Literalen.

Da jetzt nicht mehr klar ist, ob die leere Menge für die leere Menge von Klauseln oder die leere Menge von Literalen steht führen wir das Symbol $\Box$ für die leere Klausel ein.

Ist $I : \Sigma \rightarrow \{W, F\}$ eine Interpretation der Atome, die in den Literalen eine Klauselmenge S vorkommen, so ist nach der obigen Erklärung für welche Formeln die mengentheoretische Notationen stehen klar, wie man den Wahrheitswert $val_I(S)$ berechnet und ebenso $val_I(C)$, wenn C eine Menge von Literalen ist.

Um etwas Geläufigkeit mit der neuen Notation zu gewinnen schreiben wir die rekursiven Regeln für die Auswertung der Interpretation I explizit auf.

Ist S eine Menge von Klauseln, dann gilt

$$val_I(S) = \begin{cases} \mathbf{W} & \text{falls für alle } C \in S \text{ gilt: } val_I(C) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$$

Ist C eine Menge von Literalen, dann gilt

$$val_I(C) = \begin{cases} \mathbf{W} & \text{falls ein } L \in C \text{ existiert mit } val_I(L) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}$$

Es folgt also $val_I(\Box) = \mathbf{F}$ und $val_I(\emptyset) = \mathbf{W}$ für die leere Klauselmenge $\emptyset$ und $val_I(S) = \mathbf{F}$ falls $\Box \in S$.

Eine Klausel C, die nur ein einziges Literal L enthält, also $C = \{L\}$ heißt eine *Einerklausel* (im Englischen *unit clause*). Für die Einerklausel $\{L\}$ ergibt sich nach den obigen Erklärungen $val_I(\{L\}) = val_I(L)$.

Ist L ein Literal, so ist das komplementäre Literal $\bar{L}$ (siehe auch Übungsaufgabe 2.4.17 auf Seite 71) definiert durch

$$\bar{L} = \begin{cases} \neg A & \text{falls } L = A \\ A & \text{falls } L = \neg A \end{cases}$$

Der DPLL Algorithmus wird durch den Pseudocode in Abbildung 3.1 beschrieben. Bevor wir ein Beispiel durchrechnen kommentieren wir die kompakte Beschreibung. Die Eingabe ist eine Klauselmenge S. Das Ergebnis ist entweder $\text{DPLL}(S) = 1$, was heißen soll, daß die Klauselmenge S erfüllbar ist oder $\text{DPLL}(S) = 0$, was heißen soll, daß S unerfüllbar ist. Vorausgesetzt natürlich, daß der Algorithmus immer terminiert. Alle diese Aussagen werden in Satz 3.4 bewiesen werden.

Mit $\text{atom}(S)$ bezeichnen wir die Menge aller Atome P, die in S vorkommen, und zwar unabhängig davon ob P ein Literal in einer Klausel von S ist oder $\neg P$. Die Operation $red_K(S)$ für $K = \{L\}$ kann in Worten so beschrieben werden: Lasse alle Klauseln, in denen L vorkommt weg und lasse in den verbliebenen Klauseln das komplementäre Literal $\bar{L}$ weg. Die Klauselmenge

if $S = \emptyset$ **then** DPLL(S) $= 1$; **stop**
if $\square \in S$ **then** DPLL(S) $= 0$; **stop**
if S enthält Einerklausel
    then choose Einerklausel $K \in S$;
        DPLL(S) $=$ DPLL($\mathrm{red}_K(S)$);
    else choose $P \in \mathrm{atom}(S)$
        DPLL(S) $=$ **max**$\{$DPLL(S_P), DPLL($S_{\neg P}$)$\}$;

Dabei ist

- $\mathrm{atom}(S) = \bigcup_{C \in S} \mathrm{atom}(C)$,
  wobei $\mathrm{atom}(C) = \{P \in \Sigma \mid P \in C \text{ oder } \neg P \in C\}$
- $\mathrm{red}_{\{L\}}(S) = \{\mathrm{red}_{\{L\}}(C) \mid L \notin C\}$
  $\mathrm{red}_{\{L\}}(C) = \{L' \mid L' \in C \text{ und } L' \neq \bar{L}\} = C \setminus \{\bar{L}\}$
- $S_P = S \cup \{\{P\}\}$ und $S_{\neg P} = S \cup \{\{\neg P\}\}$.

Abbildung 3.1: Pseudoalgorithmus für DPLL(S)

S_P entsteht, indem zu S die Einerklausel $\{P\}$ hinzugefügt wird, entsprechend entsteht $S_{\neg P}$, indem die Einerklausel $\{\neg P\}$ zu S hinzugefügt wird.

Der DPLL Algorithmus ist, wie man sieht, ein rekursiver Algorithmus.

Zeit für ein erstes Beispiel.

Beispiel 3.3 (DPLL)

$\mathbf{S}:$
$P_1 \vee P_2 \vee P_3$
$\neg P_1 \vee P_2 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$
$\neg P_2$

$\mathbf{S_1 = red_{\neg P_2}(S)}$
$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$

$\mathbf{(S_1)_{P_1} = S_{1,0}}:$
$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$
P_1

$\mathbf{red_{P_1}(S_{1,0})}$
$\neg P_4$
P_3
$\neg P_3 \vee P_4$

$\mathbf{red_{P_3}(red_{P_1}(S_{1,0}))}$
$\neg P_4$
P_4

$\mathbf{red_{P_4}(red_{P_3}(red_{P_1}(S_{1,0})))}$
$\square$

$\mathbf{DPLL((S_1)_{P_1}) = 0}$

$\mathbf{(S_1)_{\neg P_1} = S_{1,1}}:$
$P_1 \vee P_3$
$\neg P_1 \vee \neg P_4$
$\neg P_1 \vee P_3$
$\neg P_1 \vee \neg P_3 \vee P_4$
$P_1 \vee \neg P_3$
$\neg P_1$

$\mathbf{red_{\neg P_1}(S_{1,1})}$
P_3
$\neg P_3$

$\mathbf{red_{P_3}(red_{\neg P_1}(S_{1,1}))}$
$\square$

$\mathbf{DPLL((S_1)_{\neg P_1}) = 0}$

$\mathbf{DPLL(S_1) = 0}$

Die Klauselmenge S wurde z.B. auch in [Hoo88] benutzt.

Satz 3.4

1. Der DPLL Algorithmus aus Abbildung 3.1 terminiert für jede Eingabe.

2. Der DPLL Algorithmus aus Abbildung 3.1 ist korrekt und vollständig.

Beweis
(1) Es ist ziemlich offensichtlich, daß die Reduktionsoperation, falls erforderlich nach vorheriger Einführung einer neuen Einerklauseln, ein Atom aus der Klauselmenge eliminiert. Das garantiert Terminierung.

Wir geben ein detaillierteres Argument, das auch von einem automatischen Theorembeweiser behandelt werden könnte. Dazu brauchen wir ein geeignetes Maß für die Komplexität von Klauselmengen S. Die Menge der Atome wird aufgeteilt: $\text{atom}(S) = a_u(S) \cup a_{nu}(S)$. In $a_u(S)$ liegen alle Atome P, so daß die Einerklausel $\{P\}$ oder $\{\neg P\}$ in S vorkommt. Dann ist $a_{nu}(S) = \text{atom}(S) \setminus a_u(S)$. Mit $u(S)$ und $nu(S)$ bezeichnen wir die Anzahl der Atome in $a_u(S)$ und $a_{nu}(S)$. Das gesuchte Maß ist jetzt

$$m(S) = u(S) + 2 * nu(S)$$

Falls $\{P\}$ oder$\{\neg P\}$ einer Einerklausel in S ist gilt $m(\text{red}_P(S)) < m(S)$ und $m(\text{red}_{\neg P}(S)) < m(S)$, denn P kommt in $a_u(\text{red}_P(S))$ und $a_u(\text{red}_{\neg P}(S))$ nicht mehr vor.

Für $P \in a_{nu}(S)$ gilt $m(S_P) < m(S)$ und $m(S_{\neg P}) < m(S)$, da P von $a_{nu}(S)$ nach $a_u(S_P)$, bzw $a_u(S_{\neg P})$ wandert und damit in m nur noch einfach gezählt wird. Bei der Berechnung von DPLL(S) erfolgen die rekursiven Aufrufe in dem Zeilen 5 und 7 des Pseudocodes von Abbildung 3.1 also jeweils für eine Klauselmenge mit echt kleinerer Komplexität als S.

(2) Wir müssen zeigen:

1. aus DPLL(S) $= 1$ folgt, daß S erfüllbar ist und
2. ist S erfüllbar dann gilt DPLL(S) $= 1$.

Das geschieht durch Induktion über das oben definierte Maß $n = m(S)$. Im Fall $n = 0$ gibt es genau die beiden Fälle $S = \emptyset$ oder $S = \{\square\}$. In diesen beiden Anfangsfällen sind die Behauptungen (1) und (2) wegen Zeile 1 und 2 in Abbildung 3.1 erfüllt. Im Rekursionsschritt müssen wir die beiden rekursiven Aufrufe in Zeilen 5 und 7 des Codes betrachten. Wir zeigen

3. Ist $\{L\} \in S$ dann gilt: S ist erfüllbar gdw $\text{red}_{\{L\}}(S)$ ist erfüllbar
4. Für $P \in \text{atom}(S)$ gilt
   S ist erfüllbar gdw S_P oder $S_{\neg P}$ erfüllbar ist.

zu 3 Gilt $I(S) = \mathbf{W}$, dann muß $I(L) = \mathbf{W}$, und damit auch $I(\bar{L}) = \mathbf{F}$, gelten, da $\{L\}$ Einerklausel in S ist. Für jede Klausel C in S, die L nicht enthält, muß auch $I(C \setminus \{\bar{L}\}) = \mathbf{W}$ gelten. Somit gilt $I(\text{red}_{\{L\}}(S)) = \mathbf{W}$.

Gelte umgekehrt $I(\mathrm{red}_{\{L\}}(S)) = \mathbf{W}$. Sei J eine Interpretation, die mit I übereinstimmt, bis auf $J(L) = \mathbf{W}$. Da das Literal L in $\mathrm{red}_{\{L\}}(S)$ nicht vorkommt, gilt auch $J(\mathrm{red}_{\{L\}}(S)) = \mathbf{W}$. Für die Klauseln $C \in S$ mit $l \in C$ gilt nach Definition von J sicher $J(C) = \mathbf{W}$. Für jede andere Klausel $C' \in S$ gibt es ein $C'' \in \mathrm{red}_{\{L\}}(S)$ mit $C'' \subseteq C'$. Aus $J(C'') = \mathbf{W}$ folgt umso mehr $J(C') = \mathbf{W}$. Insgesamt also $J(S) = \mathbf{W}$.

zu 4 Gelte $I(S) = \mathbf{W}$ für eine Interpretation I. Es muß entweder $I(P) = \mathbf{W}$ oder $I(\neg P) = \mathbf{W}$ gelten. Im ersten Fall ist auch $I(S_P) = \mathbf{W}$ wahr, im zweiten $I(S_{\neg P}) = \mathbf{W}$.

Die umgekehrte Implikation ist trivial, da $S \subseteq S_P$ und $S \subseteq S_{\neg P}$ gilt.

■

3.2 Horn-Formeln

Während im vorangegangenen Unterkapitel ein Verfahren vorgestellt wurde, das für beliebige KNF, trotz NP-Vollständigkeit, sehr brauchbare Resultate liefert, stellen wir in diesem Unterkapitel einen Erfüllbarkeitstest für eine Fragment konjunktiver Formeln vor, dem Fragment der aussagenlogischen Hornformeln.

Definition 3.5
Eine *Horn-Formel* ist eine aussagenlogische Formel in KNF, in der jede Disjunktion höchstens ein positives Literale enthält. Eine solche Disjunktion heißt eine *Horn-Klausel*.

Beispiel 3.6
$\neg P \wedge (Q \vee \neg R \vee \neg S) \wedge (\neg Q \vee \neg S) \wedge R \wedge S \wedge (\neg Q \vee P)$ ist eine Horn-Formel.

Eine Horn-Klausel schreibt man häufig logisch äquivalent als Implikation:

Klauselnotation	Hornklauselnotation
$\neg B_1 \vee \ldots \vee \neg B_m \vee A$	$B_1 \wedge \ldots \wedge B_m \to A$
$\neg B_1 \vee \ldots \vee \neg B_m$	$B_1 \wedge \ldots \wedge B_m \to \mathbf{0}$
A	A

Dabei heißt $B_1 \wedge \ldots \wedge B_m$ der *Rumpf* und A der *Kopf* der Horn-Klausel $B_1 \wedge \ldots \wedge B_m \to A$

Unser Beispiel lautet in dieser Schreibweise:

$$(P \to \mathbf{0}) \wedge (R \wedge S \to Q) \wedge (Q \wedge S \to \mathbf{0}) \wedge R \wedge S \wedge (Q \to P)$$

Satz 3.7
Für Horn-Formeln ist die Erfüllbarkeit in quadratischer Zeit entscheidbar.

Beweis

Wir analysieren den in Abbildung 3.2 angegebenen Algorithmus HornM(H). Die Behauptung ist, daß HornM(H) genau dann mit HornM(H)=1 terminiert, wenn die Hornklauselmenge H erfüllbar ist.

1 **for all** A **in** $\mathit{atom}(H)$ **set** : $m(A) = \mathbf{F}$;
2 **while** $\{\mathit{aktiv}(H, m) \neq \emptyset\}$ **do**
3 **if exists** $R \to \mathbf{0}$ **in** $\mathit{aktiv}(H, m)$
4 **then** HornM$(H) = 0$; **return**;
5 **else for all** $R \to K$ **in** $\mathit{aktiv}(H, m)$ **set** : $m(K) = \mathbf{W}$;
6 **endwhile**;
7 HornM$(H) = 1$;

Dabei ist

- $\mathit{atom}(H)$ die Menge der in H auftretenden aussagenlogischen Atome.
- $m : \mathit{atom}(H) \to \{\mathbf{F}, \mathbf{W}\}$ die *Markierungsfunktion*. Wir sagen, ein Atom A ist markiert, wenn $m(A) = \mathbf{W}$ gilt.
- $\mathit{aktiv}(H, m)$ ist die Menge aller Horn-Klauseln $D = A_1 \wedge \ldots A_n \to B$ in H, so daß für alle $1 \leq i \leq n$ gilt $m(A_i) = \mathbf{W}$ aber $m(B) = \mathbf{F}$. Dabei kann B ein Atom sein oder die Konstante $\mathbf{0}$.

  Beim ersten Erreichen von Zeile 2 besteht $\mathit{aktiv}(H, m)$ aus allen *Fakten* in H, d.h. allen Horn-Klauseln in H mit leerem Rumpf.
- Die Anweisung **return** in Zeile 4 beendet den ganzen Algorithmus und nicht nur die `while`-Schleife.

Abbildung 3.2: Pseudoalgorithmus für HornM(H)

Terminierung Wir bemerken zunächst, daß im Laufe des Algorithmus höchstens ein Wechsel von $m(A) = \mathbf{F}$ zu $m(A) = \mathbf{W}$ vorkommen kann, niemals umgekehrt. Bei jedem vollendeten Schleifendurchlauf findet für mindestens ein Atom A dieser Wechsel von $m(A) = \mathbf{F}$ zu $m(A) = \mathbf{W}$ statt. Da in H nur endlich viele Atome vorkommen teminiert HornM(H).

Laufzeit Gemessen an der Länge der Eingabemenge H kann jede Zeile des Pseudocodes von *HornM* in Abb. 3.2 in lineare Laufzeit ausgeführt werden, einschließlich der Berechnung von $aktiv(H, m)$. Da höchstens so viele Schritte zu machen sind, wie es Atome in H gibt, hat der Algorithmus einen quadratischen Zeitaufwand.

Korrektheit Wir beginnen mit dem einfacheren Teil. Wenn der Algorithmus mit HornM(H)=1 terminiert, dann ist H erfüllbar. In diesem Fall terminiert der Algorithmus weil Zeile 7 erreicht wurde, d.h. die `while`-Schleife wurde verlassen weil $aktiv(H, m) = \emptyset$ gilt. Wir behaupten, daß durch m eine erfüllende Belegung für H gegeben wird. Zum Beweis betrachten wir eine beliebige Hornklausel $D = A_1 \wedge \ldots \wedge A_n \rightarrow B$ in H. Gilt $val_m(A_1 \wedge \ldots \wedge A_n) = \mathbf{F}$, dann gilt auch $val_m(D) = \mathbf{W}$ und wir sind fertig. Gilt $val_m(A_1 \wedge \ldots \wedge A_n) = \mathbf{W}$, dann muß auch $m(B) = \mathbf{W}$ gelten, denn anderenfalls wäre $D \in aktiv(H, m)$, was wegen $aktiv(H, m) = \emptyset$ nicht möglich ist.

Für den zweiten Teil des Beweises zeigen wir zunächst, daß die folgende Aussage *inv* eine Schleifeninvariante für die Schleife von Zeile 2 bis 6 ist. Das heißt die Aussage *inv* gilt jedesmal, wenn die Zeile 2 erreicht wird.

inv ist für jede Interpretation I mit $val_I(D)$ für alle D in H gilt
für alle Atome A:
wenn $m(A) = \mathbf{W}$, dann auch $I(A) = \mathbf{W}$.

Wenn Zeile 2 zum ersten Mal erreicht wird, gilt $m(A) = \mathbf{F}$ für alle Atome A und *inv* ist trivialerweise wahr.

Sei jetzt m_0 die Markierung zu Beginn eines beliebigen Schleifendurchlaufs und m_1 die Markierung nach Beendigung der Schleife. (Wird die Schleife nicht vollendet und damit Zeile 2 nicht wieder erreicht, so ist nichts zu zeigen.) Die Voraussetzung ist, daß für $m = m_0$ die Aussage *inv* zutrifft. Zeigen müssen wir, daß *inv* auch für $m = m_1$ wahr ist. Dazu genügt es die Atome B zu betrachten für die in dem betrachteten Schleifendurchlauf $m_1(B) = \mathbf{W}$ gesetzt wurde. Das geschieht nur wenn es ein $D = A_1 \wedge \ldots \wedge A_n \rightarrow B$ gibt mit $D \in aktiv(H, m_0)$. Nach Definition von $aktiv(H, m_0)$ gilt $m_0(A_i) = \mathbf{W}$ für alle $1 \leq i \leq n$. Nach Voraussetzung gilt somit auch $I(A_i) = \mathbf{W}$ für alle $1 \leq i \leq n$ für jede erfüllende Belegung I von H. Wegen $I(D) = \mathbf{W}$ muß somit auch $I(B) = \mathbf{W}$ gelten.

Wir sind jetzt soweit den zweiten Teil des Korrektheitsbeweises in Angriff zu nehmen: wenn der Algorithmus mit HornM(H)=0 terminiert, dann ist H nicht erfüllbar. HornM(H)=0 kann nur auftreten, wenn Zeile 4 erreicht wird. In dieser Situation gibt es $R \rightarrow \mathbf{0}$ in $aktiv(H, m)$. Nach Definition von $aktiv(H, m)$ gilt $val_m(R) = \mathbf{W}$. Außerdem galt die Invariante *inv* zu Beginn des betreffenden Schleifendurchlaufs. Angenommen es gäbe eine erfüllende

Belegung I für ganz H. Die Aussage inv impliziert, daß ebenfalls $val_I(R) = \mathbf{W}$ gilt. Nach Wahl von I müßte auch $val_I(R \rightarrow \mathbf{0}) = \mathbf{W}$ gelten, das ist aber nicht möglich. Die Annahme es gäbe eine erfüllende Belegung ist damit auf einen Widerspruch gestoßen, H ist nicht erfüllbar.

■

3.3 Äquivalenzformeln

Das im vorangegangen Abschnitt betrachtete Fragment der Hornformeln ist durchaus von praktischer Relevanz. Prädikatenlogische Hornformeln mit genau einem positiven Literal bilden die theoretische Grundlage der Programmiersprache *PROLOG*. Das in diesem Abschnitt betrachtete Fragment der Äquivalenzformeln ist von rein didaktischem Interesse. Es ist ein konstruiertes Beispiel einer Formelklasse, für die Erfüllbarkeit extrem einfach zu entscheiden ist.

Definition 3.8
Das Fragment $\ddot{A}qFor$ aussagenlogischer Formeln besteht aus Formeln die einzig mit den Operatoren

$$\leftrightarrow, \mathbf{1}, \mathbf{0}$$

und aussagenlogischen Variablen aufgebaut sind.

Satz 3.9
Eine Formel A aus $\ddot{A}qFor$ ist eine Tautologie
gdw.
jede Aussagenvariable hat eine gerade Anzahl von Vorkommen in A
und die Konstante $\mathbf{0}$ hat eine gerade Anzahl von Vorkommen in A.

Man beachte, daß über die Häufigkeit des Vorkommens der Konstanten $\mathbf{1}$ nichts verlangt wird.

Beispiel 3.10
Nach Satz 3.9 ist

$$\Big((P \leftrightarrow \mathbf{1}) \leftrightarrow (\mathbf{0} \leftrightarrow Q)\Big) \leftrightarrow \Big((P \leftrightarrow Q) \leftrightarrow \mathbf{0}\Big)$$

eine Tautologie,

$$\Big((P \leftrightarrow \mathbf{0}) \leftrightarrow P\Big) \leftrightarrow \Big(Q \leftrightarrow (\mathbf{0} \leftrightarrow P) \leftrightarrow P\Big)$$

aber nicht.

Beweis: Wir beginnen mit einem schnellen Beweis. Sei A eine Äquivalenzformel, in der jede aussagenlogische Variable und die Konstante $\mathbf{0}$ in gerader Anzahl vorkommen. Wir zeigen, daß A eine Tautologie ist.

1. Unter Ausnutzung der Assoziativität

$$(A \leftrightarrow (B \leftrightarrow C)) \quad \leftrightarrow \quad ((A \leftrightarrow B) \leftrightarrow C)$$

   des Operators $\leftrightarrow$ können wir in A alle Klammern weglassen.

2. durch wiederholte Anwendung der Kommutativität von $\leftrightarrow$ bringen wir alle gleichen Variablen und Konstanten unmittelbar nebeneinander.

3. indem wir $(X \leftrightarrow X) \leftrightarrow \mathbf{1}$ mehrfach ausnutzen eliminieren wir paarweise Atome und Vorkommen von $\mathbf{0}$.

4. nach Voraussetzung über A bleibt eine Äquivalenzformel übrig, in der nur noch die Konstante $\mathbf{1}$ vorkommt. Diese ist äquivalent zu $\mathbf{1}$.

In dem ersten Beispiel, sieht das so aus:

1. $\Big((P \leftrightarrow \mathbf{1}) \leftrightarrow (\mathbf{0} \leftrightarrow Q)\Big) \leftrightarrow \Big((P \leftrightarrow Q) \leftrightarrow \mathbf{0}\Big)$
2. $P \leftrightarrow \mathbf{1} \leftrightarrow \mathbf{0} \leftrightarrow Q \leftrightarrow P \leftrightarrow Q \leftrightarrow \mathbf{0}$
3. $\mathbf{1} \leftrightarrow P \leftrightarrow P \leftrightarrow Q \leftrightarrow Q \leftrightarrow \mathbf{0} \leftrightarrow \mathbf{0}$
4. $\mathbf{1}$

Das ist sehr überzeugend und für das mathematische Beweisen als ein sozialer Prozeß ausreichend. Ein strenger Beweis ist das nicht. Das Hauptproblem ist, daß in unserer Notation eine klammerfreie Schreibweise nicht vorgesehen ist. Wollte man den obigen Beweis einem maschinellen *proof checker* zur Kontrolle geben, würde der schon mit einem Syntaxfehler beim Parsen abbrechen.

Hier folgt eine mathematisch rigoroser Beweis.

Wir betrachten zunächst den Spezialfall, daß in A keine Aussagevariablen vorkommen. A ist also nur aus $\leftrightarrow$, $\mathbf{1}$ und $\mathbf{0}$ aufgebaut.

Wir zeigen:

1. $val(A) = \mathbf{W}$, wenn die Konstante $\mathbf{0}$ eine gerade Anzahl von Vorkommen in A hat

2. $val(A) = \mathbf{F}$ sonst

Das läßt sich durch Induktion über die Anzahl n der in A vorkommenden $\leftrightarrow$-Zeichen beweisen.

Im Fall $n = 0$ besteht A nur aus einer Konstanten, die im Teil 1 die Konstante **1** und im Teil 2 die Konstante **0** sein muß.

Im Fall $n > 0$ enthält A eine Teilformel der Form $c_1 \leftrightarrow c_2$, wobei c_1, c_2 Konstanten sind. Es sind drei Fälle möglich:

1. Fall: c_1 und c_2 sind beide **1**

2. Fall: c_1 und c_2 sind beide **0**

3. Fall: sonst.

A_0 sei die Formel, die aus A hervorgeht, indem wir $c_1 \leftrightarrow c_2$ ersetzen durch **1** in den beiden ersten Fällen und durch **0** im dritten Fall.

Offensichtlich gilt $val(A) = val(A_0)$, und die Anzahl der Vorkommen von **0** ist entweder unverändert geblieben oder hat sich um 2 verringert. Die gewünschte Behauptung für A folgt somit aus der Induktionshypothese für A_0.

Wir kommen jetzt zum Beweis der allgemeinen Behauptung.

Zunächst nehmen wir an, daß jede Aussagenvariable und die Konstante **0** geradzahlig in A vorkommt. Formel A_I entstehe, indem man jede Variable P in A ersetzt durch **1** falls $I(P) = \mathbf{W}$ bzw. durch **0** falls $I(P) = \mathbf{F}$. A_I ist offensichtlich eine Formel vom Typ 1. Nach dem Resultat des Spezialfalls gilt also $val_I(A) = val(A_I) = \mathbf{W}$.

Kommt die Konstante **0** ungeradzahlig vor, so belege I alle Aussagevariablen mit **W**. Nach Teil 2 des Spezialfalls erhalten wir $val_I(A) = \mathbf{F}$ und A kann keine Tautologie sein.

Kommt die Konstante **0** zwar geradzahlig vor, aber eine Aussagevariable P ungeradzahlig, so sei I eine Belegung, die P mit **F** und alle anderen Aussagevariablen durch **W** belegt. Wieder gilt $val_I(A) = \mathbf{F}$ und A kann keine Tautologie sein.

■

3.4 Numerische Verfahren

Die Frage nach der Erfüllbarkeit einer aussagenlogischen Formel, oder einer Menge aussagenlogischer Formeln, kann aufgefasst werden als die Lösbarkeit einer Gleichung oder eines Gleichungssystems, wobei die auftretenden Variablen die Booleschen Werten **W** und **F** annehmen dürfen.

Dieses Unterkapitel stellt eine Verbindung her zwischen der Lösbarkeit eines Booleschen Gleichungssystems mit Booleschen Werte und der Lösbarkeit mit ganzen oder reellen Zahlen.

Definition 3.11
Sei

$$A = D_1 \wedge \cdots \wedge D_k$$

einer Formel in konjunktiver Normalform. Die vorkommenden aussagenlogischen Atome seien unter den P_j, $0 \leq j \leq n$ enthalten.

$U(A)$ ist definiert als die Menge der Ungleichungen

$$U_i \geq 1 \text{ für } 1 \leq i \leq k$$

und

$$0 \leq X_j \leq 1 \text{ für } 0 \leq j \leq n$$

Dabei wird U_i aus D_i erhalten indem

- P_j ersetzt wird durch X_j,
- $\neg P_j$ durch $(1 - X_j)$ und
- $\vee$ durch $+$.

Satz 3.12
A ist erfüllbar
gdw
U(A) in den ganzen Zahlen lösbar ist.

Beweis Einfach.

■

Beispiel 3.13
Für
$E = (P_1 \vee P_2) \wedge (P_1 \vee \neg P_2) \wedge (\neg P_1 \vee P_2) \wedge (\neg P_1 \vee \neg P_2)$
ergibt sich U(E):

$$
\begin{array}{ll}
X_1 + X_2 \geq 1 & X_1 + (1 - X_2) \geq 1 \\
(1 - X_1) + X_2 \geq 1 & (1 - X_1) + (1 - X_2) \geq 1 \\
0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1
\end{array}
$$

Vereinfacht:

$$
\begin{array}{ll}
X_1 = X_2 & X_1 + X_2 = 1 \\
0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1
\end{array}
$$

Unlösbar mit ganzen Zahlen.

Die Gleichungen

$$
\begin{array}{ll}
X_1 = X_2 & X_1 + X_2 = 1 \\
0 \leq X_1 \leq 1 & 0 \leq X_2 \leq 1
\end{array}
$$

sind allerdings lösbar für rationale Zahlen.

Satz 3.14
U(S) besitzt keine rationale Lösung
gdw
aus S ist mit 1-Resolution $\square$ herleitbar.

Beweis: Dieser Satz ist der Publikation [BJL88] entnommen.

Sei S eine Menge von Klauseln, aus der mit 1-Resolution die leere Klausel $\square$ nicht herleitbar ist. Sei S_1 die Klauselmenge, die durch iterierte 1-Resolution entsteht. Damit sind S und S_1 erfüllbarkeitäquivalent. Wir schreiben $S_1 = ML \cup S_0$, wobei ML alle Literale aus Einerklauseln sind und S_0, die Klauseln in S_1 mit zwei oder mehr Literalen sind. ML ist konsistent, d.h. es kann nicht p und $\neg p$ in ML vorkommen, sonst könnte man mit 1-Resolution weitermachen und die leere Klausel ableiten. Ordnet man den positiven Literalen in ML den Wert 1, den negativen Literalen in ML den Wert 0 und allen anderen Atomen den Wert $\frac{1}{2}$ zu, so sind alle Gleichungen in U($ML \cup S_0$) erfüllt und damit auch alle Gleichungen in dem ursprünglichen U(S).

Kann man aus S mit 1-Resolution die leere Klausel herleiten und ist ML die Menge der Literale, die dabei als Einerklauseln auftreten, dann ist eine Belegung b der Variablen mit rationalen Zahlen eine Lösung für U(S) genau dann, wenn b eine Lösung für U(ML) ist. U(ML) enthält aber nur Gleichungen der Form $x = 1$ oder $x = 0$, so daß jede Lösung eine ganzzahlige Lösung ist. Da aus ML außerdem $\square$ ableitbar ist, existiert auch keine ganzzahlige Lösung.

Als ein weiteres Beispiel betrachten wir die Übersetzung der Klauselmenge aus 3.15 in Gleichungen, für die wir eine Lösung in den ganzen Zahlen suchen.

Beispiel 3.15
Gleichungssystem aus [Hoo88]

$$\begin{array}{rcrcrcrcr}
X_1 & + & X_2 & + & X_3 & & & \geq & 1 \\
-X_1 & + & X_2 & & & - & X_4 & \geq & -1 \\
-X_1 & + & & & X_3 & & & \geq & 0 \\
-X_1 & & & - & X_3 & + & X_4 & \geq & -1 \\
X_1 & & & - & X_3 & & & \geq & 0 \\
 & - & X_2 & & & & & \geq & 0
\end{array}$$

$0 \leq X_1, X_2, X_3, X_4 \leq 1$ Aus der dritten und vorletzten Ungleichung folgt $X_1 = X_3$. Eingesetzt in das Gleichungssystem ergibt sich:

$$\begin{array}{rcrcr}
2X_1 & & & \geq & 1 \\
-X_1 & - & X_4 & \geq & -1 \\
-2X_1 & + & X_4 & \geq & -1
\end{array}$$
$$0 \leq X_1, X_4 \leq 1$$

Aus der ersten Ungleichung folgt $X_1 = 1$. Eingesetzt in die beiden folgenden Ungleichungen ergibt sich

$$\begin{array}{rrcr}
- & X_4 & \geq & 0 \\
+ & X_4 & \geq & 1
\end{array}$$
$$0 \leq X_4 \leq 1$$

Aus der letzten Ungleichung folgt sofort $X_2 = 0$. Dieses System ist sicherlich in $\mathbb{Z}$ unerfüllbar.

3.5 Übungsaufgaben

Übungsaufgabe 3.5.1
Zeigen Sie, daß für Formeln in DNF die Erfüllbarkeit in linearer Zeit, d.h. in $O(n)$, entscheidbar, wobei n die Länge der Eingabeformel ist.

Übungsaufgabe 3.5.2
Formulieren Sie ein Kriterium für die Erfüllbarkeit einer Formel in $\ddot{A}qFor$.

Definition 3.16
Seien I, J zwei aussagenlogische Interpretationen. Wir sagen I ist kleiner als J, $I \leq J$, wenn für jede aussagenlogische Variable P mit $I(P) = W$ auch $J(P) = W$ gilt.

Übungsaufgabe 3.5.3
Zeigen Sie, daß es für jede erfüllbare Horn-Formel eine kleinste erfüllende Interpretation gibt.

Definition 3.17
Eine *definite Horn-Formel* ist eine aussagenlogische Formel in KNF, in der jede Disjunktion genau ein positives Literal enthält. Eine solche Disjunktion heißt eine *definite Horn-Klausel*.

$P_1 \wedge \ldots \wedge P_k \to \mathbf{0}$ ist eine Horn-Formel, aber keine definite Horn-Formel.

Übungsaufgabe 3.5.4
Zeigen Sie, daß jede definite Horn-Formel erfüllbar ist.

Übungsaufgabe 3.5.5
In seinem Buch [Car58] stellt Lewis Carrol, der Autor von *Alice in Wonderland*, die folgende *Logelei* vor.

The Lion and the Unicorn

> When Alice entered the forest of forgetfullness, she did not forget everything, only certain things. She often forgot her name, and the most likely thing for her to forget was the day of the week. Now, the lion and the unicorn were frequent visitors to this forest. These two are strange creatures. The lion lies on Mondays, Tuesdays and Wednesdays and tells the truth on the other days of the week. The unicorn, on the other hand, lies on Thursdays, Fridays and Saturdays, but tells the thruth on all the other days of the week.
> One day Alice met the lion and the unicorn resting under a tree. They made the following statements:
>
> lion : Yesterday was one of my lying days.
> unicorn: Yesterday was one of my lying days.
>
> From these statements, Alice, who was a bright girl, was able to deduce the day of the week. What was it?

Geben Sie eine Menge aussagenlogischer Formeln *LU* and, so daß aus einer erüllenden Belegung der aussagenlogischen Atome in *LU* die Lösung des Rätsels abgelesen werden kann. Überlegen Sie sich, welche aussagenlogischen Atome Sie benutzen wollen.

Übungsaufgabe 3.5.6
Gegeben sei eine Landkarte mit L Ländern. Der Einfachheit halber seien die Länder mit den Zahlen von 0 bis $L-1$ bezeichnet. Die binäre Relation $Na(i,j)$ trifft zu auf zwei Länder i,j wenn sie benachbart sind. Die Landkarte soll mit den Farben *rot*, *blau* und *grün* so eingefärbt werden, daß keine zwei benachbarten Länder dieselbe Farbe erhalten.
Finde Sie eine Menge aussagenlogischer Formeln FF, so daß FF erfüllbar ist, genau dann wenn eine Färbung der geforderten Art möglich ist. Aus eine erfüllenden Belegung für FF, soll außerdem eine korrekte Färbung der Landkarte ablesbar sein.

Kapitel 4

Prädikatenlogik erster Ordnung: Syntax und Semantik

4.1 Einführende Beispiele

4.1.1 Alltagslogik

Auch nur ein klein wenig interessantere Schlüsse der „Alltagslogik“ lassen sich nicht mehr in der Aussagenlogik wiedergeben. Im klassischen Beispiel

> Alle Menschen sind sterblich.
> Sokrates ist ein Mensch.
> Also ist Sokrates sterblich.

kann man die letzten beiden Aussagen direkt durch Aussagevariable repräsentieren. Für die erste Aussage ist das schwierig. Rein theoretisch könnte man für jeden Menschen eine eigene Variable einführen. Offensichtlich ist das aber sehr ineffizient und würde die Aussage auch auf die jetzt lebenden Menschen beschränken.

In der Sprache der Prädikatenlogik (erster Ordnung) könnten wir die obigen Sätze schreiben:

> $\forall x$ (Mensch(x) $\to$ sterblich(x))
> Mensch(Sokrates)
> sterblich(Sokrates)

Das Symbol „$\forall$“ lesen wir „für alle“, und es ist mit Hilfe der *Variablen* x das umgangssprachliche

> Alle Menschen sind sterblich

ausgedrückt in der Weise

> Für alle x: Wenn x Mensch ist, dann ist x sterblich.

Genaueres zur Syntax sagen wir sofort. Haben wir nun die Formel

> $\forall x$ (Mensch(x) $\to$ sterblich(x)) $\to$ (Mensch(Sokrates) $\to$ sterblich(Sokrates))

zur Verfügung, so läßt sich aus

> {$\forall x$ (Mensch(x) $\to$ sterblich(x)), Mensch(Sokrates) }

mittels Modus Ponens ableiten

> sterblich(Sokrates).

4.1.2 Spezifikationen im Java Card API

Die Java Card Platform Specification v2.2.1 (siehe `http://java.sun.com/products/javacard/specs.html`) enthält unter vielen anderen die Klasse `Util`.

public class Util
extends Object

In der Klasse kommt, neben einigen anderen, die Methode `arrayCompare` vor, die zunächst übersichtsmäßig in einer Tabelle erklärt wird.

Method Summary	
static byte	arrayCompare(byte[] src, short srcOff, byte[] dest, short destOff, short length) Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right.

Darauf folgt eine detailierte Erklärung.

Method Detail arrayCompare

— Java + JML —
```
public static final byte arrayCompare(byte[] src,
                                      short srcOff,
                                      byte[] dest,
                                      short destOff,
                                      short length) throws
                                        ArrayIndexOutOfBoundsException,
                                        NullPointerException
```
— Java + JML —

Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right. Returns the ternary result of the comparison : less than(-1), equal(0) or greater than(1).

Note:

- If `srcOff` or `destOff` or `length` parameter is negative an `ArrayIndexOutOfBoundsException` exception is thrown.
- If `srcOff+length` is greater than `src.length`, the length of the `src` array an
`ArrayIndexOutOfBoundsException` exception is thrown.

- If `destOff+length` is greater than `dest.length`, the length of the `dest` array an `ArrayIndexOutOfBoundsException` exception is thrown.
- If `src` or `dest` parameter is `null` a `NullPointerException` exception is thrown.

Parameters:

`src`	-	source byte array
`srcOff`	-	offset within source byte array to start compare
`dest`	-	destination byte array
`destOff`	-	offset within destination byte array to start compare
`length`	-	byte length to be compared

Returns: the result of the comparison as follows:

- 0 if identical
- -1 if the first miscomparing byte in source array is less than that in destination array
- 1 if the first miscomparing byte in source array is greater that that in destination array

Throws:

`ArrayIndexOutOfBoundsException`	-	if comparing all bytes would cause access of data outside array bounds
`NullPointerException`	-	if either src or dest is null

Wir wollen versuchen diese natürlich-sprachlichen Aussagen in einer formalen Sprache wiederzugeben. Wir formulieren dazu eine Nachbedingung und verlangen, daß jeder Aufruf der Methode `arrayCompare` terminiert und nach Beendigung die Formel

$$(\phi_0 \to \phi_1) \land \phi_2$$

wahr ist, wobei

Abkürzungen

s	für	src	source
d	für	$dest$	destination
sO	für	$srcOff$	source offset
dO	für	$destOff$	destination offset
l	für	$length$	Länge
E	für	$java :: lang :: Exception$	
NPE	für	$java :: lang :: NullPointerException$	
OBE	für	$java :: lang :: ArrayIndexOutOfBoundsException$	

$$
\begin{array}{lll}
\phi_0 & \equiv & \begin{array}[t]{ll}
s \neq null & \wedge \\
sO \geq 0 & \wedge \\
sO + l \leq size(s) & \wedge \\
d \neq null & \wedge \\
dO \geq 0 & \wedge \\
dO + l \leq size(d) & \wedge \\
l \geq 0 &
\end{array}
\end{array}
$$

$$
\begin{array}{lll}
\phi_1 & \equiv & \begin{array}[t]{ll}
\neg excThrown(E) & \wedge \\
(result = -1 \vee result = 0 \vee result = 1) & \wedge \\
(subSeq(s, sO, sO + l) = subSeq(d, dO, dO + l) \to result = 0) & \wedge \\
(\exists i : Int(i \leq l \wedge (at(s, sO + i) < at(d, dO + i) & \wedge \\
\forall j : Int(1 \leq j < i \to at(s, sO + j) = at(d, dO + j)))) \to result = -1) & \wedge \\
(\exists i : Int(i \leq l \wedge (at(s, sO + i) > at(d, dO + i) & \wedge \\
\forall j : Int(1 \leq j < i \to at(s, sO + j) = at(d, dO + j)))) \to result = 1) &
\end{array}
\end{array}
$$

$$
\begin{array}{lll}
\phi_2 & \equiv & \begin{array}[t]{ll}
\neg excThrown(E) & \vee \\
excThrown(NPE) \wedge (s = null \vee d = null) & \vee \\
excThrown(OBE) \wedge & \\
(sO < 0 \vee dO < 0 \vee l < 0 \vee sO + l > size(s) \vee dO + l > size(d)) &
\end{array}
\end{array}
$$

4.2 Syntax der Prädikatenlogik

Die jetzt aufzubauende Prädikatenlogik erster Ordnung nennen wir kurz auch: PL1.

4.2.1 Terme und Formeln

Definition 4.1 (Logische Zeichen)
Die folgenden Zeichen sind in jeder Sprache der PL1 vorhanden. Sie heißen *Logische Zeichen* (manchmal auch Sonderzeichen).

wie Aussagenlogik	neu	
(	,	Komma
)	$\doteq$	objektsprachliches Gleichheitssymbol
1	$\forall$	Allquantor
0	$\exists$	Existenzquantor
$\neg$	v_i	Individuenvariablen, $i \in \mathbb{N}$
$\wedge$		
$\vee$		
$\to$		
$\leftrightarrow$		

$Var := \{v_i \mid i \in I\!N\}$ ist die Menge der *Individuenvariablen* oder kurz *Variablen.* Var ist disjunkt zur Menge der übrigen logischen Zeichen.

Einige Zeichen, wie z.B. das Komma, kommen sowohl in der Objektsprache als auch in der Metasprache vor. Aber das wird kaum Anlaß zu Verwechslungen geben. Bei dem Zeichen für die Gleichheit machen wir hingegen explizit einen Unterschied. Wir benutzen ein eigenes Zeichen $\doteq$ für die Gleichheit in der Objektsprache und $=$ in der Metasprache.

Definition 4.2 (Signatur)
Eine *Signatur* der PL1 ist ein Tripel $\Sigma = (F_\Sigma, P_\Sigma, \alpha_\Sigma)$ mit:

- F_Σ, P_Σ sind endliche oder abzählbar unendliche, effektiv gegebene Mengen
- F_Σ, P_Σ und die Menge der logischen Zeichen sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \to I\!N$.

Die $f \in F_\Sigma$ heißen *Funktionssymbole*, die $p \in P_\Sigma$ *Prädikatssymbole*. α_Σ ordnet jedem Funktions- oder Prädikatsymbol seine *Stelligkeit* zu: f ist *n-stelliges Funktionssymbol*, wenn $\alpha_\Sigma(f) = n$; entsprechend für $p \in P_\Sigma$.

Ein nullstelliges Funktionssymbol heißt auch *Konstantensymbol* oder kurz *Konstante*, ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom.*

Definition 4.3 (Terme)
Term$_\Sigma$, die Menge der *Terme über* Σ, ist induktiv definiert durch

1. $Var \subseteq Term_\Sigma$
2. Mit $f \in F_\Sigma$, $\alpha_\Sigma(f) = n$, $t_1, \ldots, t_n \in Term_\Sigma$ ist auch $f(t_1, \ldots, t_n) \in Term_\Sigma$
3. Ein Term heißt *Grundterm*, wenn er keine Variablen enthält. Mit $Term_\Sigma^0$ bezeichnen wir die Menge aller Grundterme in der Signatur Σ.

Insbesondere is jede Konstante ein Term. Variablen und Konstanten sind damit die kleinsten Bausteine aus denen komplexere Terme aufgebaut werden.

Definition 4.4 (Atomare Formeln)
At_Σ, die Menge der *atomaren Formeln* – oder *Atome* – *über* Σ, ist definiert als

$$\begin{array}{rcl} At_\Sigma & := & \{s \doteq t \mid s, t \in Term_\Sigma\} \cup \\ & & \{p(t_1, \ldots, t_n) \mid p \in P_\Sigma, \alpha_\Sigma(p) = n, t_1, \ldots, t_n \in Term_\Sigma\} \end{array}$$

Definition 4.5 (Formeln)
For_Σ, die Menge der *Formeln über* Σ, ist induktiv definiert durch

1. $\{\mathbf{1}, \mathbf{0}\} \cup At_\Sigma \subseteq For_\Sigma$
2. Mit $x \in Var$ und $A, B \in For_\Sigma$ sind ebenfalls in For_Σ:

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B), \forall x A, \exists x A$$

Bemerkungen

Wir schreiben in der Regel

$x, y, z, \ldots$	für Variable
$a, b, c, \ldots$	für Konstanten
$f, g, h, \ldots$	für Funktionssymbole allgemein
$P, Q, \ldots$	für aussagenlogische Atome
$p, q, r, \ldots$	für Prädikatensymbole allgemein
$s, t, u, \ldots$	für Terme
$A, B, C, \ldots$	für Formeln

ggf. mit Indizes oder ähnlichen Unterscheidungszeichen.

Den Index Σ lassen wir oft weg. Zur kürzeren Notation von Formeln werden dieselben Klammereinsparungsregeln verwendet wie in der Aussagenlogik.

Die Anwendung von Funktionssymbolen geben wir manchmal auch in Infixschreibweise wieder: $(s + t)$ statt $+(s, t)$.

Wir denken uns die Mengen F_Σ, P_Σ stets *effektiv* gegeben (vgl.: Grundlagen der Berechenbarkeit) und nehmen entsprechend an, daß α_Σ *berechenbar* sei.

Strukturelle Induktion
Die Definition oder der Beweis einer Eigenschaft von Termen oder Formeln durch *strukturelle Induktion* geschieht völlig entsprechend zur Situation in der Aussagenlogik, siehe 2.2.1.

Definition 4.6
Ein *Teilterm* oder *Unterterm* eines Terms t ist ein Teilwort von t, das selbst Term ist; entsprechend sind die Teilterme (Unterterme) einer Formel definiert. Eine *Teilformel* einer Formel A ist ein Teilwort von A, das selbst Formel ist.

4.2.2 Gebundene und freie Variable. Substitutionen

Definition 4.7

1. Wenn wir im Folgenden vom Auftreten einer Variablen, z.B. x in einer Formel reden, so schließen wir das Vorkommen von x direkt nach ei-

nem Quantor, also $\forall x$ oder $\exists x$, aus. Dieses Vorkommen von x wird als Bestandteil des Quantors angesehen.

2. Mit $Var(A)$, $Var(t)$ bezeichnen wir alle in der Formel A, bzw. in dem Term t vorkommenden Variablen.

3. Hat eine Formel A die Gestalt $\forall xB$ oder $\exists xB$ so heißt B der *Wirkungsbereich* des *Quantors* $\forall x$ bzw. $\exists x$ von A.

4. Ein Auftreten einer Variablen x in einer Formel A heißt *gebunden*, wenn es innerhalb des Wirkungsbereichs eines Quantors $\forall x$ oder $\exists x$ einer Teilformel von A stattfindet.

5. Mit $Bd(A)$ bezeichnen wir die Menge der Variablen, die in A mindestens einmal gebunden auftreten.

6. Ein Auftreten einer Variablen x in einer Formel A heißt *frei*, wenn es nicht gebunden ist.

7. Mit $Frei(A)$ bezeichnen wir die Menge der Variablen, die in A mindestens einmal frei auftreten.

Man könnte hierbei das *Auftreten* eines Zeichens ζ in A definieren als ein Paar (ζ, i) mit: $1 \leq i \leq |A|$, und an der Position i in A steht ζ. Für unsere Zwecke genügt jedoch ein intuitives Verständnis dieses Konzepts.

In Aufgabe 4.2.4 wird eine formale, rekursive Definition von $Frei(A)$ und $Bd(A)$ nachgefragt.

Man beachte, daß eine Variable in einer Formel auch frei *und* gebunden auftreten kann, d.h. es kann $Frei(A) \cap Bd(A) \neq \{\}$ gelten. Ferner gilt, daß im Wirkungsbereich eines Quantors, etwa $\forall x$, derselbe Quantor wieder auftreten kann.

Beispiel 4.8
In der Formel

$$\forall x(p_0(x,y) \rightarrow \forall z(\exists y\ p_1(y,z) \vee \forall x\ p_2(f(x),x)))$$

treten x und z nur gebunden auf, y tritt frei und gebunden auf.

Für Terme gibt es keine Bindungen von Variablen, so daß wir jedes Auftreten einer Variablen in einem Term als dort freies Auftreten ansehen wollen.

$t \in Term_\Sigma$ heißt *Grundterm*, wenn $Var(t) = \{\}$.

Definition 4.9
A heißt *geschlossen*, wenn $Frei(A) = \{\}$. Ist allgemein $Frei(A) = \{x_1, \ldots, x_n\}$, so heißt

$\forall x_1 \ldots \forall x_n A$ *Allabschluß*
$\exists x_1 \ldots \exists x_n A$ *Existenzabschluß*

von A. Abkürzend schreiben wir $Cl_\forall A$ bzw. $Cl_\exists A$.

Ist A geschlossen, dann gilt also $Cl_\forall A = Cl_\exists A = A$.

Definition 4.10 (Substitutionen)
Eine *Substitution* (über Σ) ist eine Abbildung

$$\sigma : Var \to Term_\Sigma$$

mit $\sigma(x) = x$ für fast alle $x \in Var$.

Sind $x_1, \ldots, x_m$ so, daß gilt $\{x \mid \sigma(x) \neq x\} \subseteq \{x_1, \ldots, x_m\}$, und ist $\sigma(x_i) = s_i$ für $i = 1, \ldots, m$, so geben wir σ auch an in der Schreibweise

$$\{x_1/s_1, \ldots, x_m/s_m\}.$$

σ heißt *Grundsubstitution*, wenn für alle $x \in Var$ gilt: $\sigma(x) = x$ oder $\sigma(x)$ ist Grundterm.

Mit *id* bezeichnen wir die identische Substitution auf Var, d.h. $id(x) = x$ für alle $x \in Var$.

Definition 4.11 (Anwendung von Substitutionen)
Wir setzen eine Substitution σ fort zu Abbildungen

$$Term_\Sigma \to Term_\Sigma$$

$$For_\Sigma \to For_\Sigma$$

die beide wieder mit σ bezeichnet werden, mittels:

- $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$
- $\sigma(p(t_1, \ldots, t_n) = p(\sigma(t_1), \ldots, \sigma(t_n))$
- $\sigma(t \doteq s) = \sigma(t) \doteq \sigma(s))$
- $\sigma(\neg A) = \neg\sigma(A)$

- $\sigma(A \circ B) = \sigma(A) \circ \sigma(B)$ für jeden zweistelligen aussagenlogischen Operator $\circ$.
- $\sigma(QxA) = Qx\sigma_x(A)$, wobei $\sigma_x(x) = x$ und $\sigma_x(y) = \sigma(y)$ für $y \neq x$, für $Q \in \{\forall, \exists\}$

$\sigma(A)$ entsteht aus A, indem simultan für jedes $x \in Var$ an jeder Stelle, wo x frei in A auftritt, x ersetzt wird durch $\sigma(x)$.

Beispiel 4.12

1. Für $\sigma = \{x/f(x,y), y/g(x)\}$ gilt $\sigma(f(x,y)) = f(f(x,y), g(x))$.
2. Für $\mu = \{x/c, y/d\}$ gilt $\mu(\exists yp(x,y)) = \exists yp(c,y)$.
3. Für $\sigma_1 = \{x/f(x,x)\}$ gilt $\sigma_1(\forall yp(x,y)) = \forall yp(f(x,x),y)$.
4. Für $\mu_1 = \{x/y\}$ gilt $\mu_1(\forall yp(x,y)) = \forall yp(y,y)$.

Das Beispiel 4 unterscheidet sich von den vorangegangenen dadurch, daß an einer Position der Formel vor der Substitution eine ungebundene Variable steht, nämlich x, und nach der Substitution eine gebundene Variable, nämlich y. Man hat schon jetzt das Gefühl, daß eine solche Substitution die Bedeutung einer Formel in unerlaubter Weise verändert, was wir in Lemma 4.35 konkretisieren können. Wir wollen jedoch schon hier für die in Beispiel 4 aufgetretene Situation eine Bezeichnung einführen.

Definition 4.13 (kollisionsfreie Substitutionen)
Eine Substitution σ heißt *kollisionsfrei* für eine Formel A, wenn für jede Variable z und jede Stelle freien Auftretens von z in A gilt: Diese Stelle liegt nicht im Wirkungsbereich eines Quantors $\forall x$ oder $\exists x$, wo x eine Variable in $\sigma(z)$ ist.

Nach dieser Definition ist $\{x/y\}$ keine kollisionsfreie Substitution für $\forall yp(x,y)$.

Definition 4.14 (Komposition von Substitutionen)
Sind σ, τ Substitutionen, dann definieren wir die Komposition von τ mit σ durch

$$(\tau \circ \sigma)(x) = \tau(\sigma((x)).$$

Man beachte, daß auf der rechten Seite τ als die zugehörige, gleichnamige Abbildung $Term_\Sigma \to Term_\Sigma$ verstanden werden muß.

Lemma 4.15

1. Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, daß $\sigma(t) = \tau(t)$, dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t.

2. Wenn $\sigma(t) = t$, dann $\sigma(s) = s$ für jeden Teilterm s von t.

Beweis

1. Strukturelle Induktion nach t.

   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \dots, t_n)$. Dann gilt

     $$\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n))$$

     und

     $$\tau(f(t_1, \dots, t_n)) = f(\tau(t_1), \dots, \tau(t_n)).$$

     Es folgt also $\sigma(t_i) = \tau(t_i)$ für $i = 1, \dots, n$. Nach Definition der Teiltermrelation ist s ist mit t identisch oder Teilterm eines t_i, nach Induktionsvoraussetzung folgt also die Behauptung.

2. Spezialfall von 1 mit $\tau = id$.

∎

Für spätere Zwecke besprechen wir noch eine spezielle Sorte von Substitutionen.

Definition 4.16
Eine *Variablenumbenennung* ist eine Substitution σ mit

1. $\sigma(x) \in Var$ für alle $x \in Var$

2. σ ist injektiv

Korollar 4.17
Gilt für Substitutionen σ, τ, daß $\tau \circ \sigma = id$, dann ist σ eine Variablenumbenennung.

Beweis Es ist $\tau(\sigma(x)) = x$ für jedes $x \in Var$, woraus folgt: $\sigma(x) \in Var$. Ferner haben wir: Wenn $\sigma(x) = \sigma(y)$, dann $x = \tau(\sigma(x)) = \tau(\sigma(y)) = y$.

∎

4.2.3 Unifikation

Für eine Menge $T \subseteq Term_\Sigma$ schreiben wir kurz

$$\sigma(T)$$

für

$$\{\sigma(t) \mid t \in T\}.$$

Entsprechend für $M \subseteq For_\Sigma$.

Definition 4.18 (Unifikator)
Es sei $T \subseteq Term_\Sigma$, $T \neq \{\}$, und σ eine Substitution über Σ. σ *unifiziert* T, oder: ist *Unifikator von* T, genau dann, wenn $\#\sigma(T) = 1$. T heißt *unifizierbar*, wenn T einen Unifikator besitzt. Insbesondere sagen wir für zwei Terme s, t daß s *unifizierbar* sei *mit* t, wenn $\{s, t\}$ unifizierbar ist, d.h. in diesem Fall, wenn $\sigma(t) = \sigma(s)$.

Die Begriffe werden auf Formelmengen übertragen durch: σ *unifiziert* M genau dann, wenn $\#\sigma(M) = 1$. (Hierbei liefert # zu einer Menge ihre Mächtigkeit; insbesondere zu einer endlichen Menge die Anzahl ihrer Elemente.)

Beispiel 4.19
$\{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$ wird unifiziert durch $\{x/a, y/v, z/g(a, a), w/b\}$.

Wir beschränken uns im Folgenden auf die Betrachtung der Unifikation von Termmengen und überlasssen die (einfache) Übertragung auf Formelmengen dem Leser.

Wir halten einen Moment inne und betrachten die folgenden simplen Feststellungen:

1. Jeder Term ist mit sich selbst unifizierbar. Jede beliebige Substitution ist in diesem Fall ein Unifikator. Die identische Substitution id ist gefühlsmäßig der *einfachste* Unifikator.

2. Zwei Terme der Gestalt

$$f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)$$

(mit demselben Kopf) sind genau dann unifizierbar, wenn es eine Substitution σ gibt mit $\sigma(s_i) = \sigma(t_i)$ für $i = 1, \ldots, n$.

3. Ist $x \in Var$ und t ein Term, der x nicht enthält, dann sind x und t unifizierbar.

4. Ist $x \in Var$ und t ein Term $\neq x$, der x enthält, dann sind x und t nicht unifizierbar

Diese Beobachtungen liefern die Idee zu einem Algorithmus, der die Unifizierbarkeit einer endlichen Menge von Termen testet und im positiven Fall einen Unifikator liefert. Bevor wir diesen Algorithmus hinschreiben, gehen wir noch auf die Frage der Eindeutigkeit eines Unifikators ein.

Beispiel 4.20

$$\{f(x, g(y)), f(g(a), g(z))\}$$

wird unifiziert durch

$$\sigma = \{x/g(a), z/y\},$$

aber auch durch

$$\tau = \{x/g(a), y/a, z/a\}.$$

σ ist *allgemeiner* als τ – oder τ *spezieller* als σ – insofern, als sich aus dem Resultat der Anwendung von σ auf eine beliebige Termmenge T immer noch, durch die nachgeschaltete Substitution $\{y/a\}$, das Resultat der Anwendung von τ auf T gewinnen läßt:

$$\tau(T) = \{y/a\}(\sigma(T)) \text{ für alle } T \subseteq Term_\Sigma$$

d. h.

$$\tau = \{y/a\} \circ \sigma.$$

Uns interessieren insbesondere Unifikatoren einer Termmenge, die in diesem Sinne so allgemein wie möglich sind.

Definition 4.21
Es sei $T \subseteq Term_\Sigma$. Ein *allgemeinster Unifikator* oder mgu (*most general unifier*) von T ist eine Substitution μ mit

1. μ unifiziert T
2. Zu jedem Unifikator σ von T gibt es eine Substitution σ' mit $\sigma = \sigma' \circ \mu$.

Lemma 4.22
Es sei T eine nichtleere Menge von Termen. Dann ist jeder allgemeinste Unifikator von T bis auf Variablenumbenennung eindeutig bestimmt, d. h.: Sind μ, μ' allgemeinste Unifikatoren von T mit $\mu(T) = \{t\}$ und $\mu'(T) = \{t'\}$, dann gibt es eine Variablenumbenennung π mit $t' = \pi(t)$.

Beweis

Nach der Definition allgemeinster Unifikatoren gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt $\mu = \sigma'\sigma\mu$ und insbesondere für jeden Term $t_0 \in T$ $\mu(t_0) = \sigma'\sigma\mu(t_0)$. Also auch $t = \sigma'\sigma(t)$. Das kann nur sein, wenn für jede Variable $x \in Var(t)$ gilt $\sigma'\sigma(x) = x$. Daraus folgt insbesondere, daß für jedes $x \in Var(t)$ $\sigma(x)$ wieder eine Variable sein muß und für $x, y \in Var(t)$ mit $x \neq y$ auch $\sigma(x) \neq \sigma(y)$ gilt. Wir definieren

$$\pi(x) = \begin{cases} \sigma(x) & \text{falls } x \in Var(t) \\ \varphi(x) & \text{falls } x \notin Var(t). \end{cases}$$

wobei φ eine beliebige bijektive Funktion von $Var \setminus Var(t)$ auf $Var \setminus \{\sigma(x) \mid x \in Var(t)\}$ ist mit: $\varphi(y) = y$ für fast alle $y \in Var$. Offensichtlich ist π eine Variablenumbenennung und nach Wahl von σ gilt für jeden Term $t_0 \in T$ $\mu'(t_0) = \sigma\mu(t_0)$, also

$$t' = \mu'(t_0) = \sigma\mu(t_0) = \sigma(t) = \pi(t)$$

■

4.2.4 Unifikationsalgorithmus

Wir stellen nun den von Robinson stammenden Algorithmus vor, der zu einer endlichen Menge T von Termen entscheidet, ob T unifizierbar ist, und im positiven Fall einen allgemeinsten Unifikator liefert. Wir benötigen die

Definition 4.23
Zu $t \in Term_\Sigma$ und $i \in \mathbb{N}$ sei

$t^{(i)}$:= der an Position i in t (beim Lesen von links nach rechts) beginnende Teilterm von t, wenn dort eine Variable oder ein Funktionssymbol steht;
undefiniert sonst.

Es sei nun $T \subseteq Term_\Sigma$ endlich und $\neq \emptyset$. Die *Differenz* von T ist die folgende Menge $D(T) \subseteq Term_\Sigma$

1. $D(T) := T$ falls $\#T = 1$

2. Falls $\#T \geq 2$, sei i das kleinste j mit: Mindestens zwei Terme $\in T$ unterscheiden sich an Position j. Setze $D(T) := \{t^{(i)} \mid t \in T\}$.

(Beachte: Nach Übung 4.2.1, 6 (Seite 109) ist $t^{(i)}$ wirklich ein Teilterm von t.)

Gegeben sei $T \subseteq Term_\Sigma$, T endlich und $\neq \emptyset$.

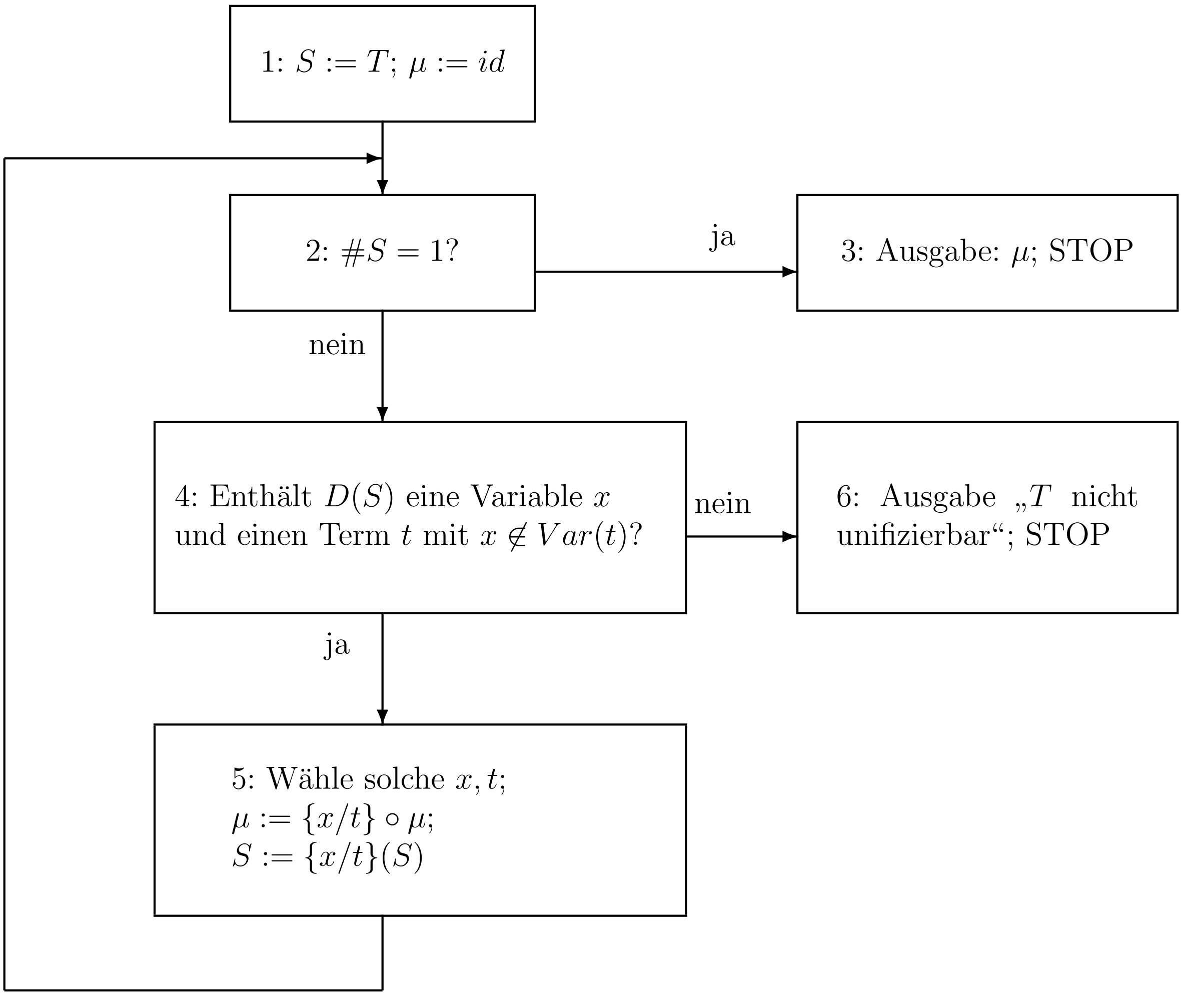


Abbildung 4.1: Algorithmus von Robinson

Man bestätige zur Übung, daß gilt:
Es sei $\#T \geq 2$ und T unifizierbar. Dann

1) $\#D(T) \geq 2$

2) Jeder Unifikator von T unifiziert auch $D(T)$

3) $D(T)$ enthält eine Variable x und einen Term t mit $x \notin Var(t)$.

Satz 4.24
Der Algorithmus von ROBINSON (Bild 4.1, Seite 106) terminiert für jedes endliche, nichtleere $T \subseteq Term_\Sigma$. Wenn T unifizierbar ist, liefert er einen allgemeinsten Unifikator von T. Wenn T nicht unifizierbar ist, liefert er die Ausgabe „T nicht unifizierbar“.

Beweis

Wir zeigen

1. Der Algorithmus terminiert.
2. Wenn er eine Substitution μ ausgibt, dann ist μ Unifikator von T.
3. Ist σ ein beliebiger Unifikator von T, dann gilt: Es gibt μ, σ' so daß
   - der Algorithmus mit Ausgabe μ terminiert,
   - $\sigma = \sigma' \circ \mu$
   - *Somit*: μ ist mgu von T.
4. Wenn der Algorithmus ausgibt „T nicht unifizierbar“, dann ist T nicht unifizierbar.

Unter einem *Schleifendurchlauf* in dem obigem Algorithmus verstehen wir einen *vollständigen* Durchlauf der Schleife, d. h. der Befehlsfolge 2–4–5. Ist S eine Menge von Termen und x eine Variable, so sagen wir kurz, daß x *in* S *auftritt*, wenn x in einem Term in S auftritt.

Wir setzen

- $S_0 := T$, $\mu_0 := id$
- $S_{k+1} :=$ Wert von S nach dem $(k+1)$-ten Schleifendurchlauf
- $\mu_{k+1} :=$ entsprechend mit μ

sofern die Schleife tatsächlich so oft durchlaufen wird. Ferner seien dann x_k, t_k die im $(k+1)-$ten Schleifendurchlauf ausgewählten x, t.

Ad 1:

k sei so, daß die Schleife mindestens $(k+1)-$mal durchlaufen wird. Dann gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei tritt x_k in S_k, aber nicht in t_k auf, jede Variable in S_{k+1} aber auch in S_k. Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k. Es folgt, daß die Schleife nur endlich oft durchlaufen wird. Das Programm terminiert.

Im Folgenden sei m die Anzahl der Schleifendurchläufe bis zur Terminierung. (Es wir also 5 genau $m-$mal, 2 genau $(m+1)-$mal ausgeführt.)

Ad 2:

Durch Induktion über k ergibt sich unmittelbar, daß

$$S_k = \mu_k(T)$$

für alle $k \leq m$. Hält das Programm mit Ausgabe einer Substitution μ an, dann hat μ den Wert μ_m, und es ist

$$\#\mu_m(T) = \#S_m = 1.$$

μ_m ist Unifikator von T.

Ad 3:

Es sei σ ein Unifikator von T. Wir zeigen zunächst

$(*)$ Für alle $k \leq m$: es gibt σ_k mit $\sigma = \sigma_k \circ \mu_k$.

$k = 0$:

Setze $\sigma_0 := \sigma$.

$k + 1$:

Nach Induktionsannahme existiert σ_k mit $\sigma = \sigma_k \circ \mu_k$. Wir haben

$$\#\sigma_k(S_k) = \#\sigma_k(\mu_k(T)) = \#\sigma(T) = 1$$

da σ Unifikator von T ist. Wegen $k+1 \leq m$ wurde im $(k+1)-$ten Durchlauf noch Test 2 mit „Nein“ und Test 4 mit „Ja“ verlassen: es ist $\#S_k \geq 2$, und in $D(S_k)$ gibt es x_k, t_k mit $x_k \notin Var(t_k)$. Da σ_k Unifikator von S_k ist, muß gelten $\sigma_k(x_k) = \sigma_k(t_k)$. Wir setzen

$$\sigma_{k+1}(x) = \begin{cases} \sigma_k(x) & \text{falls } x \neq x_k \\ x_k & \text{falls } x = x_k \end{cases}$$

Wir haben für alle x:

Falls $x \neq x_k$:
$$\sigma_{k+1}(\{x_k/t_k\}(x)) = \sigma_{k+1}(x) = \sigma_k(x),$$
falls $x = x_k$:
$$\begin{aligned}&\sigma_{k+1}(\{x_k/t_k\}(x)) = \sigma_{k+1}(\{x_k/t_k\}(x_k))\\ &= \sigma_{k+1}(t_k) = \sigma_k(t_k) \quad (\text{da } x_k \notin Var(t_k))\\ &= \sigma_k(x_k) = \sigma_k(x).\end{aligned}$$
Somit
$$\sigma_{k+1} \circ \{x_k/t_k\} = \sigma_k.$$
Es folgt
$$\begin{aligned}&\sigma_{k+1} \circ \mu_{k+1} = \sigma_{k+1} \circ \{x_k/t_k\} \circ \mu_k\\ &= \sigma_k \circ \mu_k = \sigma\end{aligned}$$
d. h. $(*)$.

Insbesondere gilt
$$\sigma = \sigma_m \circ \mu_m.$$

Angenommen nun, bei der $(m+1)-$ten Durchführung des Tests 2 würde der „Nein"-Ausgang gewählt, d. h. es wäre $\#S_m \geq 2$.

σ_m unifiziert S_m (da σ T unifiziert). Also muß $D(S_m)$ eine Variable x und einen Term t enthalten mit $x \notin Var(t)$ (siehe die obige Übung zum Operator D). Test 4 würde mit „Ja" verlassen, im Widerspruch zur Definition von m. Hiermit ist 3 bewiesen.

Ad 4: Folgt unmittelbar aus 3.

∎

4.2.5 Übungsaufgaben

Übungsaufgabe 4.2.1
Beweisen Sie die folgenden Aussagen

1. $Term_\Sigma \cap For_\Sigma = \{\}$
2. Kein Teilwort eines Terms s ist eine Formel.
3. Ist s ein Term und sind t, u Teilterme von s, dann gilt genau eine der Aussagen
   (a) u ist Teilterm von t
   (b) t ist echter Teilterm von u
   (c) t, u liegen disjunkt

Entsprechendes gilt für Teilterme einer Formel und Teilformeln einer Formel.

4. Ist der Term t ein Präfix des Terms s, dann $s = t$. Ist die Formel B ein Präfix der Formel A, dann $A = B$. Keine Formel kann Präfix eines Terms sein.

5. Es sei t ein Term und i eine Position in t (d. h. $1 \leq i \leq |t|$; t wird von links nach rechts gelesen). Genau dann beginnt bei i ein Teilterm von t, wenn dort eine Variable oder ein Funktionssymbol steht.

6. Es seien $s, t \in Term_\Sigma$, $s \neq t$. Dann gibt es eine erste Position i, $1 \leq i \leq min(|s|, |t|)$, an der s und t sich unterscheiden. Für dieses i gilt: Sowohl in s wie in t steht dort eine Variable oder ein Funktionssymbol. (Nach 5 also: dort beginnt ein Teilterm.)

Es sei nochmals betont, daß diese Aussagen sich auf unsere „offizielle" Sprache (ohne Klammereinsparungen) beziehen.

Übungsaufgabe 4.2.2
Definieren Sie den Begriff eines „Teilterms zu einem Term s" durch strukturelle Induktion. Entsprechend zu einer Formel A: „Teilformel von A".

Übungsaufgabe 4.2.3
Im Falle einer endlichen Signatur Σ sind $Term_\Sigma$, For_Σ kontextfreie Sprachen, man gebe erzeugende Grammatiken an. Auch bei einer unendlichen Signatur lassen sich in manchen Fällen kontextfreie Grammatiken für $Term_\Sigma$ bzw. For_Σ angeben, wenn man die Funktions- und Prädikatsymbole geeignet codiert: etwa, wenn es zu jeder Stelligkeit n unendlich viele $n-$stellige Funktions- und Prädikatsymbole gibt; oder wenn dies, mit festem m, für alle $n \leq m$ der Fall ist und insgesamt nur für endlich viele Stelligkeiten solche Symbole vorhanden sind. Weshalb hat man im allgemeinen Fall Schwierigkeiten?

Übungsaufgabe 4.2.4
Man definiere $Frei(A)$ und $Bd(A)$ direkt durch strukturelle Induktion.

Übungsaufgabe 4.2.5
Zu jeder Substitution σ und jeder Formel A gibt es eine Formel A', die aus A durch Umbenennung gebundener Variablen entsteht, so daß σ für A' kollisionsfrei ist.

Übungsaufgabe 4.2.6
Geben Sie (falls existent) für die folgenden Paare von Termen einen *allgemeinsten Unifikator* σ an. Falls kein mgu existiert, begründen Sie dies.

1. $\{p(f(x_1, x_1), x_1, f(x_2, x_2), x_2),$
   $p(y_1, f(y_2, y_2), y_2, f(y_3, y_3))\}$
2. $\{p(f(x, y), g(x, y)),$
   $p(f(h(z), h(z)), g(h(z), z))\}$
3. $\{p(f(y), w, g(z)),\ p(u, u, v)\}$
4. $\{p(f(y), w, g(z)),\ p(v, u, v)\}$

Übungsaufgabe 4.2.7
Dual zur Definition 4.18 definieren wir den Begriff der konkretesten Verallgemeinerung zweier Terme.

Definition 4.25
Seien t_1, t_2 Terme.

1. ein Term t heißt eine (gemeinsame) Verallgemeinerung von t_1, t_2 wenn es Substutionen σ_1 und σ_2 gibt mit
$$\sigma_1(t) = t_1 \quad \text{und} \quad \sigma_2(t) = t_2$$
2. t heißt eine konkreteste Verallgemeinerung von t_1, t_2, wenn t zunächst eine Verallgemeinerung von t_1, t_2 ist und für jeden Term s, für den es Substitutionen τ_1, τ_2 gibt mit $\tau_1(s) = t_1$ und $\tau_2(s) = t_2$ eine Substitution ρ existiert mit $\rho(s) = t$.

Die Definition wird durch das folgende Bild veranschaulicht:

$$\begin{tikzcd}
 & s \arrow[d, "\rho"] \arrow[ddl, bend right, "\tau_1"'] \arrow[ddr, bend left, "\tau_2"] & \\
 & t \arrow[dl, "\sigma_1"'] \arrow[dr, "\sigma_2"] & \\
t_1 & & t_2
\end{tikzcd}$$

Wir geben eine Vorschrift an, die aus zwei Termen t_1, t_2 einen dritten Term $kVg(t_1, t_2)$ konstruiert. Als Hilfsfunktion brauchen wir eine Abbildung nv die zwei beliebigen Termen s_1, s_2 eindeutig eine neue Variable $nv(s_1, s_2)$ zuordnet. Die Einzelheiten der Definition von nv spielen keine Rolle, solange nur aus $nv(s_1, s_2) = nv(s'_1, s'_2)$ folgt $s_1 = s'_1$ und $s_2 = s'_2$.

$$kVg(t_1, t_2) = \begin{cases} f(kVg(t_{11}, t_{21}), \ldots, kVg(t_{1k}, t_{2k})) & \text{falls } t1 = f(t_{11}, \ldots t_{1k}) \\ & \text{und} \\ & t2 = f(t_{21}, \ldots t_{2k}) \\ nv(t_1, t_2) & \text{sonst} \end{cases}$$

Zeigen Sie, daß $kVg(t_1, t_2)$ eine konkreteste Verallgemeinerung von t_1 und t_2 ist.

Die konkreteste Verallgemeinerung von t_1 und t_2 wird manchmal auch als die *Anti-Unifikation* von t_1 und t_2 bezeichnet.

Übungsaufgabe 4.2.8
Gegeben seien die Substitutionen σ, τ.
In Korrolar 4.17 wurde gezeigt, daß aus $\tau \circ \sigma = id$ folgt, daß σ eine Variablenumbenennung ist.
Zeigen Sie, daß auch τ eine Variablenumbenennung ist.

4.3 Semantik der PL1

Betrachten wir die Formel

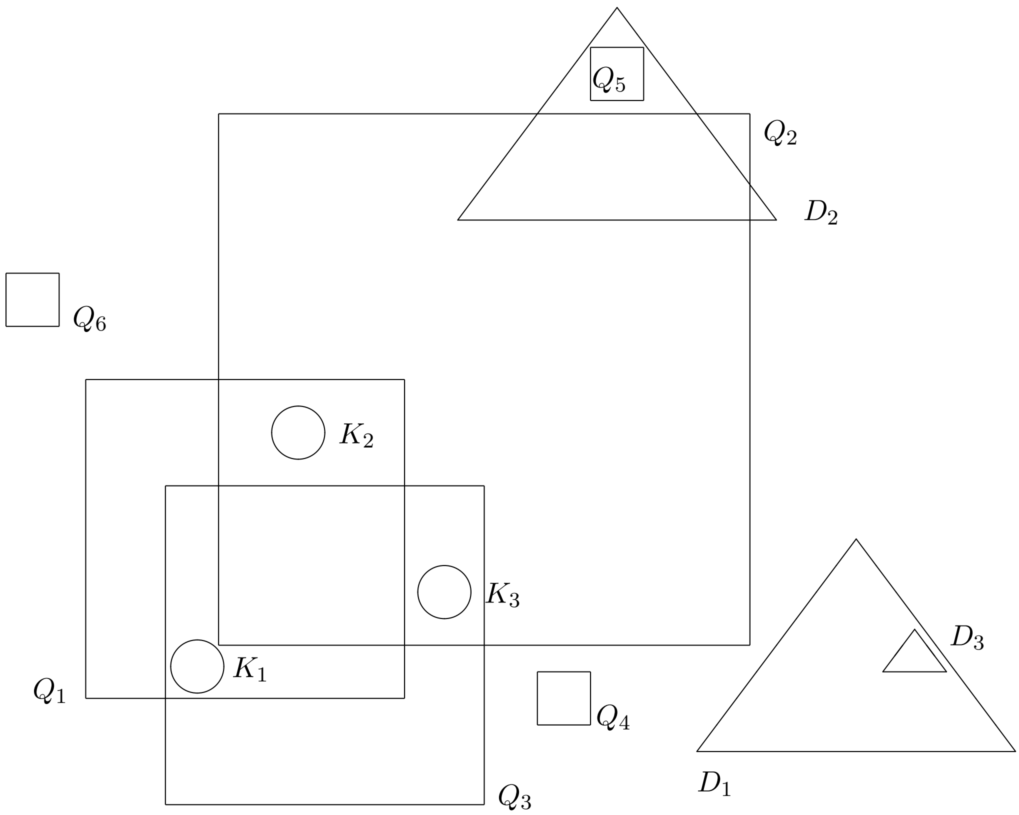
$$q(x) \rightarrow \exists y(\ in(y, x) \wedge kl(y)),$$

so können wir bisher nur sagen, daß sie eine korrekt gebildetet Formel über der durch die Auflistung der Prädikatszeichen $\{k(\ ),\ q(\ ),\ d(\ ),\ kl(\ ),\ gr(\ ),\ in(\ ,\ )\}$ gegebene Signatur Σ ist. Es macht keinen Sinn zu fragen, ob diese Formel, wahr oder falsch ist. Dazu müßten wir zuerst wissen, welche Bedeutung die auftretenden Relationszeichen q, kl und in haben, wir müßten wissen über welchem Bereich von Elementen wir nach einem Beispiel für die durch $\exists y$ signalisierte Existenzbehauptung suchen sollen. Diese Informationen werden durch die im nächsten Abschnitt definierten *Interpretationen* festgelegt. Die Angabe einer Interpretation genügt aber noch immer nicht um der Formel einen Wahrheitswert zuordnen zu können, solange wir nicht wissen, auf welches Element die freie Variable x referenziert. Diese Information wird durch die sogenannten *Variablenbelegungen* festgelegt. In Abhängigkeit von einer Interpretation $\mathcal{D}$ und einer Variablenbelegung β liegt jetzt die Wahrheit oder Falschheit der obigen Formel fest.

4.3.1 Interpretationen

Definition 4.26 (Interpretation)
Es sei Σ eine Signatur der PL1. Eine *Interpretation* $\mathcal{D}$ von Σ ist ein Paar (D, I) mit

1. D ist eine beliebige, nichtleere Menge
2. I ist eine Abbildung der Signatursymbole, die
   - jeder Konstanten c ein Element $I(c) \in D$
   - für $n \geq 1$: jedem $n-$stelligen Funktionssymbol f eine Funktion $I(f) : D^n \rightarrow D$
   - jedem $0-$stelligen Prädikatsymbol P einen Wahrheitswert $I(P) \in \{W, F\}$
   - für $n \geq 1$: jedem $n-$stelligen Prädikatsymbol p eine $n-$stellige Relation $I(p) \subseteq D^n$ zuordnet.



$P_\Sigma = \{k(\ ),\ q(\ ),\ d(\ ),\ kl(\ ),\ gr(\ ),\ in(\ ,\ )\}$
$D_{Bsp} = \{Q_i : 1 \leq i \leq 6\} \cup \{K_1, K_2, K_3, D_1, D_2, D_3\}$ $I_{Bsp}(q) = \{Q_i : 1 \leq i \leq 6\}$, $I_{Bsp}(k) = \{K_1, K_2, K_3\}$, $I_{Bsp}(d) = \{D_1, D_2, D_3\}$
$I_{Bsp}(in) = \{(K_1, Q_1), (K_1, Q_3), (K_2, Q_1), (K_2, Q_2), (K_3, Q_2), (K_3, Q_3), (D_3, D_1), (Q_5, D_2)\}$

Tabelle 4.1: Beispiel einer Interpretation

Wir werden manchmal eine Interpretatin $\mathcal{D} = (D, I)$ auch eine *prädikatenlogische Struktur*, oder einfach eine *Struktur* nennen. Der Funktion I haben wir hier keinen eigenen Namen gegeben. Falls notwendig nennen wir I die *Interpretationsfunktion*. Wir nennen D das Universum von $\mathcal{D}$.

Tabelle 4.1 zeigt ein Beispiel einer Interpretation Σ mit den Prädikatszeichen $P_\Sigma = \{k(\ ),\ q(\ ),\ d(\ ),\ kl(\ ),\ gr(\ ),\ in(\ ,\ )\}$. Die Interpretation des einstelligen Prädikatszeichens k ist dabei die Menge aller Kreise, die von q die Menge aller Quadrate, die von d die Menge aller Dreiecke. Das Universum besteht in diesem Beispiel aus der Menge aller aufgezeichneten geometrischen Objekte. Die Interpretation von kl ist die Menge aller kleinen Objekte, die von gr

entsprechend die Menge aller großen Objekte. Da man sich darüber streiten kann, was groß und was klein ist, geben wir explizit an:

$I_{Bsp}(kl) = \{K_1, K_2, K_3, Q_4, Q_5, Q_6, D_3\}$
$I_{Bsp}(gr) =$ Komplement von $I_{Bsp}(kl)$ in D.

Die Interpretation der binären Relation *in* wird in der Abbildung indirekt durch die geometrische Enthaltenseinsrelation dargestellt.

Definition 4.27 (Variablenbelegung)
Es sei (D, I) eine Interpretation von Σ. Eine *Variablenbelegung* (oder kurz *Belegung* über D) ist eine Funktion

$$\beta : Var \to D.$$

Zu β, $x \in Var$ und $d \in D$ definieren wir die *Modifikation* von β an der Stelle x zu d:

$$\beta_x^d(y) = \begin{cases} d & \text{falls } y = x \\ \beta(y) & \text{falls } y \neq x \end{cases}$$

Definition 4.28 (Auswertung von Termen und Formeln)
Es sei (D, I) eine Interpretation von Σ und β eine Variablenbelegung über D. Wir definieren eine Funktion $val_{D,I,\beta}$, welche jedem Term ein Element in D und jeder Formel einen Wahrheitswert zuordnet. Also:

$$val_{D,I,\beta} : Term_\Sigma \cup For_\Sigma \to D \cup \{W, F\}$$

mit

$$val_{D,I,\beta}(t) \in D \text{ für } t \in Term_\Sigma$$
$$val_{D,I,\beta}(A) \in \{W, F\} \text{ für } A \in For_\Sigma$$

1. $val_{D,I,\beta}$ auf $Term_\Sigma$:
   $val_{D,I,\beta}(x) = \beta(x)$ für $x \in Var$
   $val_{D,I,\beta}(f(t_1, \ldots, t_n)) = (I(f))(val_{D,I,\beta}(t_1), \ldots, val_{D,I,\beta}(t_n))$

2. $val_{D,I,\beta}$ auf For_Σ:

   (a) $val_{D,I,\beta}(\mathbf{1}) = W$
   $val_{D,I,\beta}(\mathbf{0}) = F$
   $val_{D,I,\beta}(s \doteq t) := \begin{cases} W \text{ falls } val_{D,I,\beta}(s) = val_{D,I,\beta}(t) \\ F \text{ sonst} \end{cases}$
   $val_{D,I,\beta}(P) := I(P)$ für 0-stellige Prädikate P
   $val_{D,I,\beta}(p(t_1, \ldots, t_n)) := \begin{cases} W \text{ falls } (val_{D,I,\beta}(t_1), \ldots, val_{D,I,\beta}(t_n)) \in I(p) \\ F \text{ sonst} \end{cases}$

(b) Ist $val_{D,I,\beta}$ für Formeln A, B bereits definiert, so wird $val_{D,I,\beta}$ auf $\neg A$, $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$ festgelegt wie in der Aussagenlogik. Ferner:

$$val_{D,I,\beta}(\forall x A) := \begin{cases} W & \text{falls für alle } d \in D : val_{D,I,\beta_x^d}(A) = W \\ F & \text{sonst} \end{cases}$$

$$val_{D,I,\beta}(\exists x A) := \begin{cases} W & \text{falls ein } d \in D \text{ existiert mit } val_{D,I,\beta_x^d}(A) = W \\ F & \text{sonst} \end{cases}$$

Insbesondere ist also $val_{D,I,\beta}(c) = I(c)$ für jede Konstante c.
Ferner erkennt man:
Ist t ein Grundterm, dann hängt $val_{D,I,\beta}(t)$ von β gar nicht ab. Wir schreiben dann $I(t)$ statt $val_{D,I,\beta}(t)$.

Beispiel 4.29
Wir wollen die Formel

$$q(x) \rightarrow \exists y(\ in(y,x) \wedge kl(y)),$$

in der Interpretation $\mathcal{D}_{Bsp}$ aus Abbildung 4.1 mit der Variablenbelegung $\beta(x) = Q1$ auswerten.
Wir beginnen mit der links des Implikationspfeils $\rightarrow$ stehenden Teilformel und bemerken $val_{\mathcal{D}_{Bsp},\beta}(x) = Q1 \ \in I(q)$. Nach der Definition von val folgt also $val_{\mathcal{D}_{Bsp},\beta}(q(x)) = \ W$.
Wir behaupten, daß auch für die rechts von $\rightarrow$ stehende Teilformel gilt:
$val_{\mathcal{D}_{Bsp},\beta}(\exists y(\ in(y,x) \wedge kl(y)) = \ W$
Dazu müssen wir ein geeignetes Element aus D für die Belegung der existentiell quantifizierten Variablen y finden. Im vorliegenden Fall eignet sich K_1.
Wir behaupten also
$val_{\mathcal{D}_{Bsp},\beta_y^{K_1}}((\ in(y,x) \wedge kl(y)) = \ W$
Durch weitere Anwendung der Definition von val kann diese Behauptung reduziert werden zu
$(K_1, Q_1) \in I_{Bsp}(in)$ und $K_1 \in I_{Bsp}(kl)$,
was offensichtlich nach Definition des Beispiels zutrifft. Insgesamt haben wir damit

$$val_{\mathcal{D}_{Bsp},\beta}(q(x) \rightarrow \exists y(\ in(y,x) \wedge kl(y)) = \ W$$

gezeigt.

Korollar 4.30
Es gelten die aus der Aussagenlogik bekannten Abhängigkeiten zwischen den Bedeutungen der Operatoren $\mathbf{1}$, $\mathbf{0}$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, so daß hier wieder die bekannten Teilmengen dieser Menge ausreichen. Ferner gilt für alle x, A:

$$val_{D,I,\beta}(\exists x A) = val_{D,I,\beta}(\neg\forall x \neg A)$$
$$val_{D,I,\beta}(\forall x A) = val_{D,I,\beta}(\neg\exists x \neg A)$$

d. h. $\exists$ ist durch $\forall$ und $\neg$, $\forall$ ist durch $\exists$ und $\neg$ ausdrückbar. Als *Basis* der Sprachen der PL1 können wir also etwa nehmen

$$\mathbf{0}, \rightarrow, \forall$$

oder

$$\neg, \rightarrow, \forall$$

oder

$$\neg, \wedge, \forall$$

und viele andere mehr. Die übrigen Operatoren lassen sich dann als Abkürzungen auffassen. Insbesondere genügt es bei struktureller Induktion über For_Σ, die Formeln über irgendeiner solchen Basis zu betrachten.

Zur **notationellen Vereinfachung** werden wir manchmal eine Interpretation (D, I) durch einen Buchstaben $\mathcal{D}$ bezeichnen und schreiben:

$val_{\mathcal{D},\beta}$	für	$val_{D,I,\beta}$.
$\mathcal{D} \models A[a_1, \ldots, a_n]$	für	$val_{\mathcal{D},\beta}(A) = W$, wobei $\beta(x_i) = a_i$ und $x_1, ..., x_n$ die in A frei vorkommenden Variablen sind, oder eine Obermenge davon
$t^{\mathcal{D}}[a_1, \ldots, a_n]$	für	$val_{\mathcal{D},\beta}(t)$ entsprechend

Wir gewinnen in dieser Notation an Übersichtlichkeit durch die Einsparung von β, müssen aber voraussetzen, daß die Zuordnung, welches der in eckigen Klammern angeführten Elemente als Belegung für welche Variable dient, aus dem Kontext ersichtlich ist. Wir belegen in der Regel die Variablen in ihrer alpha-numerischen Reihenfolge.

Das folgende Lemma faßt einige offensichtliche Konsequenzen aus der Definition der Auswertungsfunktion *val* zusammen. Zum Beispiel hängt der Wert von $val_{\mathcal{D},\beta}(A)$ nicht ab von den Funktionswerten $\beta(x)$ für Variable x, die in A nicht frei vorkommen. Diese Tatsachen werden im folgenden stillschweigend benutzt. Ein ähnliches Lemma ließe sich formulieren über die Unabhängigkeit von $val_{D,I,\beta}(A)$ von den Werten $I(P)$ für Prädikate P, die in A nicht vorkommen. Wir verzichten darauf und vertrauen der natürlichen Intelligenz des Lesers solche und ähnliche Offensichtlichkeiten im folgenden zu erkennen.

Lemma 4.31 (Koinzidenzlemma)
$\mathcal{D}$ sei Interpretation über Σ.

1. Gilt für den Term t und die Variablenbelegungen β, γ, daß

$$\beta(x) = \gamma(x) \text{ für alle } x \in Var(t),$$

dann

$$val_{\mathcal{D},\beta}(t) = val_{\mathcal{D},\gamma}(t).$$

2. Gilt für die Formel A und die Variablenbelegungen β, γ, daß

$$\beta(x) = \gamma(x) \text{ für alle } x \in Frei(A),$$

dann

$$val_{\mathcal{D},\beta}(A) = val_{\mathcal{D},\gamma}(A).$$

3. Ist $A \in For_\Sigma$ geschlossen, dann gilt

$$val_{\mathcal{D},\beta}(A) = val_{\mathcal{D},\gamma}(A)$$

für alle Belegungen β, γ, d. h. der Wert von A hängt nicht von der Belegung ab.

Beweis: Durch strukturelle Induktion unter Ausnutzung der Definition von val.

■

Für die beiden folgenden Beispiele legen wir die arithmetische Signatur Σ_{arith} fest bestehend aus den Zeichen für die Addition (+), Multiplikation ($*$), Ordnungsrelation ($\leq$).

Beispiel 4.32
Die Struktur $\mathcal{Z}$ stehe für die gewohnte Interpretation der Signatur Σ_{arith} mit Universum $\mathbb{Z}$ und der Additionsfunktion ($+_\mathcal{Z}$), Multiplikationsfunktion ($*_\mathcal{Z}$) und der Ordnungsrelation ($\leq_\mathcal{Z}$), zusammenfassend: $\mathcal{Z} = (\mathbb{Z}, +_\mathcal{Z}, *_\mathcal{Z}, \leq_\mathcal{Z})$.

Hier, wie häufig im Folgenden, schreiben wir anstelle von $\mathcal{Z} = (\mathbb{Z}, I)$ auch $(\mathbb{Z}, +_\mathcal{Z}, *_\mathcal{Z}, \leq_\mathcal{Z})$ mit $I(+) = +_\mathcal{Z}$ etc.

Beispiel 4.33
Die Struktur $\mathcal{Z}_{Jint}$ steht für den Java Datentyp `int`. Das Universum $\mathbb{Z}_{Jint}$ besteht aus den ganzen Zahlen zwischen $minInt$ und $maxInt$, d.h. $\mathbb{Z}_{Jint}$ ist das Interval $[minInt, maxInt] = [-2147483648, 2147483647]$. Die Ordnungsrelation $\leq_{Jint}$ ist die Einschränkung von $\leq_\mathcal{Z}$ auf das Intervall $\mathcal{Z}_{Jint}$. Liegt das Ergebnis einer Addition oder Multiplikation in $\mathcal{Z}_{Jint}$ im Intervall $[minInt, maxInt]$, dann stimmt es mit dem entsprechenden Ergebnis in $\mathcal{Z}$ überein. Anderenfalls wird modulo gerechnet, wie unten beschrieben. Die Länge des Intervalls $[minInt, maxInt]$ ist gleich $-2 * minInt$ mit dem Zahlenwert 4294967296. Präzise gilt

$$
\begin{array}{lcll}
x +_{Jint} y & = & x +_{\mathcal{Z}} y & \text{falls } minInt \leq x +_{\mathcal{Z}} y \leq maxInt \\
 & & mod_{Jint}(x +_{\mathcal{Z}} y) & \text{sonst} \\
x *_{Jint} y & = & x *_{\mathcal{Z}} y & \text{falls } minInt \leq x *_{\mathcal{Z}} y \leq maxInt \\
 & & mod_{Jint}(x *_{\mathcal{Z}} y) & \text{sonst}
\end{array}
$$

wobei

$$mod_{Jint}(z) = (z - minInt)mod(-2 * minInt) + minInt$$

Anschaulich kann man sich vorstellen, daß in $\mathcal{Z}_{Jint}$ nach Erreichen von $maxInt$ mit $minInt$ weitergezählt wird, die ganzen Zahlen in Java sind also in gewisser Weise zyklisch. Daß diese anschauliche Beschreibung mit der mathematischen übereinstimmt bestätigt die Rechnung:

$$
\begin{array}{lcl}
maxInt +_{Jint} 1 & = & mod_{Jint}(maxInt + 1) \\
 & = & (maxInt + 1 - minInt)mod(-2 * minInt) + minInt \\
 & = & (-2 * minInt)mod(-2 * minInt) + minInt \\
 & = & 0 + minInt \\
 & = & minInt.
\end{array}
$$

Hier sind einige Beispiel, welche die beiden arithmetischen Strukturen miteinander vergleichen:

Formel ϕ	$\mathcal{Z} \models \phi$	$\mathcal{Z}_{Jint} \models \phi$
$\forall x \exists y(x < y)$	ja	nein
$\forall x, y((x + 1) * y = x * y + y)$	ja	ja
$\exists x(0 < x \land x + 1 < 0)$	nein	ja

Die folgenden beiden Lemmata sind technischer Natur, spielen aber eine Schlüsselrolle bei Beweisen durch strukturelle Induktion. Darüberhinaus sagen sie Grundsätzliches über die Substitution aus. Es geht um folgendes: Der Wert, der bei der Auswertung eines Terms oder einer Formel einer freien Variablen x zukommt, kann auf zwei Arten beeinflußt werden. Man kann durch eine Substitution σ an die Stelle der freien Variablen x einen Term $\sigma(x)$ setzen, oder man kann die Variablenbelegung β an der Stelle x modifizieren. Die Substitutionslemmas beschreiben den Zusammenhang zwischen diesen beiden Möglichkeiten.

Lemma 4.34 (Substitutionslemma für Terme)
Σ sei eine Signatur, $\mathcal{D}$ eine Interpretation für Σ, β eine Belegung, σ eine Substitution und $t \in Term_{\Sigma}$. Dann gilt

$$val_{\mathcal{D},\beta}(\sigma(t)) = val_{\mathcal{D},\beta'}(t).$$

wobei $\beta'(x) = val_{\mathcal{D},\beta}(\sigma(x))$ für alle $x \in Var$.

Beweis

Strukturelle Induktion nach t.

$t = x \in Var$:

$$\begin{array}{rcll} val_{\mathcal{D},\beta}(\sigma(x)) & = & \beta'(x) & \text{nach Def. von } \beta' \\ & = & val_{\mathcal{D},\beta'}(x) & \text{nach Def. von } val \text{ für Variable} \end{array}$$

$t = f(t_1, \ldots, t_n)$:

$$\begin{aligned} & val_{\mathcal{D},\beta}(\sigma(f(t_1, \ldots, t_n))) \\ & \quad = val_{\mathcal{D},\beta}(f(\sigma(t_1), \ldots, \sigma(t_n))) \\ & \quad = I(f)(val_{\mathcal{D},\beta}(\sigma(t_1)), \ldots, val_{\mathcal{D},\beta}(\sigma(t_n))) \\ & \quad = I(f)(val_{\mathcal{D},\beta\prime}(t_1), \ldots, (val_{\mathcal{D},\beta\prime}(t_n))) \text{ (nach Induktionsannahme)} \\ & \quad = val_{\mathcal{D},\beta'}(f(t_1, \ldots, t_n)). \end{aligned}$$

■

Lemma 4.35 (Substitutionslemma für Formeln)
Σ sei eine Signatur, $\mathcal{D}$ eine Interpretation für Σ, β eine Belegung, $A \in For_\Sigma$ und σ eine für A kollisionsfreie Substitution. Dann gilt:

$$val_{\mathcal{D},\beta}(\sigma(A)) = val_{\mathcal{D},\beta'}(A),$$

wobei $\beta'(x) = val_{\mathcal{D},\beta}(\sigma(x))$ für alle $x \in Var$.

Beweis: Geschieht durch strukturelle Induktion nach A. Wir führen hier exemplarisch den Induktionsschritt von A nach $\exists x A$ vor. Wir schreiben val_β anstelle von $val_{\mathcal{D},\beta}$. Außerdem sei σ_x definiert durch $\sigma_x(x) = x,\ \sigma_x(y) = \sigma(y)$ für $y \neq x$.

$$\begin{array}{lllr} & val_\beta(\sigma(\exists x A)) = W & & \\ \text{gdw} & val_\beta(\exists x \sigma_x(A)) = W & & \text{Anwendung von } \sigma \\ \text{gdw} & val_{\beta_x^d}(\sigma_x(A)) = W \text{ für ein } d \in D & & \text{Def. von } val \\ \text{gdw} & val_{(\beta_x^d)''}(A) = W & & \text{Ind.Voraussetz.} \\ & & \multicolumn{2}{r}{\text{wo } (\beta_x^d)''(y) = val_{\beta_x^d}(\sigma_x(y)) \text{ für all } y.} \\ \text{gdw} & val_{(\beta')_x^d}(A) = W & & \text{Lücke} \\ \text{gdw} & val_{\beta'}(\exists x A) = W & & \text{Def. von } val \end{array}$$

Man beachte, daß die Induktionsvoraussetzung für die Formel A mit der Variablenbelegung β_x^d und der Substitution σ_x benutzt wird.
Der Beweis wird vollständig geführt sein, wenn wir die Lücke

$$(\beta_x^d)'' = (\beta')_x^d$$

schließen können. Wir müssen für jede Variable $y \in Frei(A)$ zeigen $(\beta_x^d)''(y) = (\beta')_x^d(y)$.

$y = x$:

$$\begin{array}{lcll}
(\beta_x^d)''(x) & = & val_{\beta_x^d}(\sigma_x(x)) & \text{Def. von } (\beta_x^d)'' \\
& = & val_{\beta_x^d}(x) & \text{Def. von } \sigma_x \\
& = & \beta_x^d(x) & \text{Def. von } val \text{ für Variable} \\
& = & d & \text{Def. der modifizierten Belegung} \\
& = & (\beta')_x^d(x) & \text{Def. der modifizierten Belegung}
\end{array}$$

$y \neq x$, y frei in A:

$$\begin{array}{lcll}
(\beta_x^d)''(y) & = & val_{\beta_x^d}(\sigma_x(y)) & \text{Def. von } (\beta_x^d)'' \\
& = & val_{\beta_x^d}(\sigma(y)) & \text{Def. von } \sigma_x \\
& = & val_{\beta}(\sigma(y)) & \text{da } x \text{ nicht in } \sigma(y) \text{ vorkommt} \\
& & & \text{Kollisionsfreiheit von } \sigma \\
& = & \beta'(y) & \text{Def. von } \beta' \\
& = & (\beta')_x^d(y) & \text{Def. der modifizierten Belegung}
\end{array}$$

■

Wir beenden das Thema „Substitutionslemma“ mit zwei Anwendungsbeispielen.

Beispiel 4.36 (Hoare-Kalkül)
Die Zuweisungsregel im Hoare-Kalkül ([dB80], Seite 38) lautet:

$$\{\{x/s\}A\} \text{ x := s } \{A\}$$

wobei die Substitution kollisionsfrei sein muß und ausdrücken soll, daß ausgehend von einem Zustand, in dem die Formel $\{x/s\}A$ wahr ist, nach Ausführung der Programmstücks $x := s$ ein Zustand erreicht wird, in dem die Formel A gilt. Die Struktur, in welcher die Wahrheit der Formeln auszuwerten ist, wird in der Regel nicht explizit angegeben. Sie ergibt sich aus dem Kontext und legt die übliche Bedeutung der verwendeten Datentypen fest, z.B. der natürlichen Zahlen, der Listen von Zahlen oder der Bäume samt der üblichen Funktionen und Relationen auf diesen Datentypen. Wir nennen diese Hintergrund-Interpretation $\mathcal{H}$. Ein Zustand ist eindeutig bestimmt durch die augenblickliche Belegung der Programmvariablen und entspricht somit unserem Begriff einer Variablenbelegung. Wir betrachten eine Variablenbelegung (einen Zustand) β, in dem $val_{\mathcal{H},\beta}(\{x/s\}A) = W$ gilt. Die Vorbedingung der Zuweisungsregel ist also erfüllt. Nach Ausführung der Zuweisung $x := s$ wird ein Zustand β' erreicht mit

$$\beta'(y) := \begin{cases} val_{\mathcal{H},\beta}(s) \text{ falls } x = y \\ \beta(y) \text{ sonst} \end{cases}$$

Behauptet wird, daß in dem neuen Zustand, β', die Formel A gilt, oder in formaler Notation aufgeschrieben, daß $val_{\mathcal{H},\beta'}(A) = W$. Ein genauer Vergleich zeigt, daß das gerade die Aussage des Substitutionslemmas für Formeln ist für die Substitution $\sigma = \{x/s\}$.

Das zweite Anwendungsbeispiel tritt im Beweis des folgenden Lemmas auf:

Lemma 4.37
Sei Σ eine Signatur, $\mathcal{D}$ eine Interpretation für Σ, β eine Belegung und σ eine für A kollisionsfreie Substitution mit $\sigma(y) = y$ für alle Variablen $y \neq x$, dann gilt:

- $val_{\mathcal{D},\beta}(\forall x A \rightarrow \sigma(A)) = W$
- $val_{\mathcal{D},\beta}(\sigma(A) \rightarrow \exists x A) = W$.

Beweis
Wir nehmen an, daß $val_{\mathcal{D},\beta}(\forall x A) = W$ gilt, d.h.

$$val_{\mathcal{D},\beta_x^d}(A) = W \text{ für alle } d \in D.$$

Zu zeigen ist

$$val_{\mathcal{D},\beta}(\sigma(A)) = W$$

Nach dem Substitutionslemma ist das gleichbedeutend mit

$$val_{\mathcal{D},\beta'}(A) = W$$

wobei

$$\beta'(y) = val_{\mathcal{D},\beta}(\sigma(y)) = \begin{cases} \beta(y) \text{ falls } x \neq y \\ val_{\mathcal{D},\beta}(\sigma(x)) \text{ falls y = x} \end{cases}$$

Also $\beta' = \beta_x^d$ für $d = val_{\mathcal{D},\beta}(\sigma(x))$.

Die zweite Aussage läßt sich analog beweisen.

■

4.3.2 Allgemeingültigkeit, logische Äquivalenz

Wir wollen uns bei der Behandlung der logischen Folgerbarkeit beschränken auf Formeln, die keine freien Variablen enthalten. Das ist mit Abstand der typischste Anwendungsfall. Der Fall mit freien Variablen wird in den Übungsaufgaben 4.3.6 bis 4.3.9 ausführlich behandelt.

Wenn wir im folgenden keine genauen Abgaben machen ist jede Formel als eine Formel ohne freie Variablen zu verstehen.

Definition 4.38 (Modell)
Eine Interpretation $\mathcal{D}$ über Σ heißt **Modell** einer Formel A ohne freie Variaben über Σ, wenn gilt $val_{\mathcal{D}}(A) = W$.
$\mathcal{D}$ heißt **Modell** einer Formelmenge M ohne freie Variable, wenn für jede Formel $B \in M$ gilt $val_{\mathcal{D}}(B) = W$.

Definition 4.39 (Folgerbarkeit)
Es sei M eine Menge von Formeln aus For_Σ und A eine einzelne Formel aus For_Σ, wobei weder in M noch in A freie Variablen vorkommen.

$$M \models_\Sigma A \quad :\Leftrightarrow \quad \text{Jedes Modell von } M \text{ ist auch Modell von } A.$$

Lies: **Aus** M **folgt** A (über Σ).

Wir werden meistens kurz $\models$ statt $\models_\Sigma$ schreiben. Ferner stehe wieder $\models A$ für $\emptyset \models A$ und $B \models A$ für $\{B\} \models A$.

Definition 4.40
$A \in For_\Sigma$ ohne freie Variablen heißt

- **allgemeingültig** gdw $\models A$
- **erfüllbar** gdw $\neg A$ ist nicht allgemeingültig.

Lemma 4.41
Sei A eine variablenfreie Formel.

1. Die folgenden Aussagen sind äquivalent:
   (a) A allgemeingültig
   (b) Jede Interpretation $\mathcal{D}$ ist Modell von A.
   (c) $val_{\mathcal{D}}(A) = W$ für alle Interpretationen $\mathcal{D}$.
2. Die folgenden Aussagen sind äquivalent:
   (a) A erfüllbar
   (b) Es gibt eine Interpretation $\mathcal{D}$ mit $val_{\mathcal{D}}(A) = W$
3. Falls M keine freien Variablen enthält gilt:
   $M \cup \{\neg A\}$ ist nicht erfüllbar gdw $M \models A$.

Beweis: Übung

Wir werden im Folgenden viele Beweise allgemeingültiger Formeln kennenlernen. Ein besonders einfacher Fall liegt vor, wenn die Allgemeingültigkeit „schon allein aufgrund der aussagenlogischen Struktur besteht“:

Definition 4.42 (Tautologie)
$A \in For_\Sigma$ ohne freie Variablen heißt *Tautologie*, wenn es eine endliche aussagenlogische Signatur $\Sigma' = \{P_0, \ldots, P_{n-1}\}$, ein $A' \in For^0_{\Sigma'}$ und Formeln $A_0, \ldots, A_{n-1} \in For_\Sigma$ gibt, so daß

- A' ist (aussagenlogisch) allgemeingültig über Σ'
- A entsteht aus A', indem man dort P_i durch A_i ersetzt (für $i = 0, \ldots, n-1$).

Beispiel 4.43
(wir lassen keine Klammern weg):

$$\begin{array}{l}(((\forall x \exists y(p(x) \land \neg q(y,c,x)) \to p(c)) \\ \land(\neg\forall x \exists y(p(x) \land \neg q(y,c,x)) \to p(c))) \to p(c))\end{array}$$

ist eine Tautologie, denn

$$(((P_0 \to P_1) \land (\neg P_0 \to P_1)) \to P_1)$$

ist aussagenlogisch allgemeingültig, und hieraus entsteht durch Ersetzen von
P_0 durch $\forall x \exists y(p(x) \land \neg q(y,c,x))$
P_1 durch $p(c)$
die Ausgangsformel.

Lemma 4.44
Jede Tautologie ist allgemeingültig.

Beweis

Zur Tautologie A seien $A_0, \ldots, A_{n-1}$ Teilformeln, wie in der Definition festgelegt. Es sei D, I gegeben. Gemäß der Definition der A_i ist dann $val_{D,I}(A)$ eindeutig festgelegt durch die Werte $val_{D,I}(A_i)$, $i = 0, \ldots, n-1$, und ist für jede Verteilung dieser Werte W.

■

Besonders wichtig jedoch sind – wie in der Aussagenlogik – allgemeingültige Formeln der Gestalt $A \leftrightarrow B$.

Definition 4.45
$A, B \in For_\Sigma$ heißen *logisch äquivalent* gdw $\models A \leftrightarrow B$ (d. h. $A \leftrightarrow B$ ist allgemeingültig)

Wie es wünschenswert ist und wie wir gleich festhalten, ist logische Äquivalenz eine Kongruenz auf For_Σ.

Korollar 4.46
Logische Äquivalenz ist eine Äquivalenzrelation auf For_Σ. Sie ist darüberhinaus eine Kongruenz, d. h.: gilt

$$\models A \leftrightarrow A', \models B \leftrightarrow B'$$

dann auch

$$\models \neg A \leftrightarrow \neg A'$$
$$\models (A \; op \; B) \leftrightarrow (A' \; op \; B') \text{ für } op \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$
$$\models QxA \leftrightarrow QxA' \text{ für } x \in Var, Q \in \{\forall, \exists\}$$

Beweis

Unmittelbar aus der Definition.

■

Korollar 4.47 (Ersetzungstheorem)
Es seien $A, A', B, B' \in For_\Sigma$ mit

- B ist Teilformel von A
- B ist logisch äquivalent zu B'
- A' entsteht aus A, indem man dort an irgendwelchen Stellen (nicht notwendig überall) B durch B' ersetzt.

Dann ist A logisch äquivalent zu A'.

Beweis Klar.

■

Satz 4.48
Für $x, y \in Var$, $A, B \in For_\Sigma$ und $Q \in \{\forall, \exists\}$ sind logisch äquivalente Formelpaare:

1. A und B, wenn $A \leftrightarrow B$ Tautologie ist,
2. $\neg\forall xA$ und $\exists x\neg A$, $\neg\exists xA$ und $\forall x\neg A$
3. $\forall x\forall yA$ und $\forall y\forall xA$, $\exists x\exists yA$ und $\exists y\exists xA$
4. $\forall x(A \wedge B)$ und $\forall xA \wedge \forall xB$

5. $\exists x(A \vee B)$ und $\exists xA \vee \exists xB$
6. QxA und $Qy\{x/y\}(A)$, falls $\{x/y\}$ kollisionsfrei ist für A und $y \notin Frei(A)$
7. $A \wedge QxB$ und $Qx(A \wedge B)$, falls $x \notin Frei(A)$
8. $QxA \wedge B$ und $Qx(A \wedge B)$, falls $x \notin Frei(B)$
9. $A \vee QxB$ und $Qx(A \vee B)$, falls $x \notin Frei(A)$
10. $QxA \vee B$ und $Qx(A \vee B)$, falls $x \notin Frei(B)$
11. $A \rightarrow \forall xB$ und $\forall x(A \rightarrow B)$, falls $x \notin Frei(A)$
12. $\forall xA \rightarrow B$ und $\exists x(A \rightarrow B)$, falls $x \notin Frei(B)$
13. $A \rightarrow \exists xB$ und $\exists x(A \rightarrow B)$, falls $x \notin Frei(A)$
14. $\exists xA \rightarrow B$ und $\forall x(A \rightarrow B)$, falls $x \notin Frei(B)$

Man bemerke, daß diese Liste teilweise redundant ist (5 ergibt sich aus 4 etc.). Wir haben sie absichtlich in dieser Reichhaltigkeit aufgeschrieben. Die Äquivalenz 6 nennt man *gebundene Umbenennung*.

Beweis

1 wissen wir schon, 2 bis 5 rechnet man unmittelbar nach. Man mache 6 als Übung (wobei man das Substitutionslemma verwende); ohnehin wird die Allgemeingültigkeit der gebundenen Umbenennung aus späteren Resultaten folgen. Stellvertretend für 7 bis 14 zeigen wir 11. (Die anderen Aussagen dieser Gruppe folgen hieraus oder werden entsprechend bewiesen.)

Es ist zu zeigen, daß für alle D, I, β gilt:

$$val_{D,I,\beta}(A \rightarrow \forall xB) = val_{D,I,\beta}(\forall x(A \rightarrow B))$$

wenn $x \notin Frei(A)$. Falls $val_{D,I,\beta}(A \rightarrow \forall xB) = W$, hat man $val_{D,I,\beta}(\forall x(A \rightarrow B)) = W$ unmittelbar aus der Definition von $val_{\ldots}$ (Übung).

Sei umgekehrt $val_{D,I,\beta}(\forall x(A \rightarrow B)) = W$, d. h.

1. Für alle $d \in D$: $(val_{D,I,\beta_x^d}(A) = W \Rightarrow val_{D,I,\beta_x^d}(B) = W)$.
   Angenommen, es wäre $val_{D,I,\beta}(A \rightarrow \forall xB) = F$. Dann gilt also
2. $val_{D,I,\beta}(A) = W$
3. $val_{D,I,\beta}(\forall xB) = F$, d. h.: es gibt ein $e \in D$ mit $val_{D,I,\beta_x^e}(B) = F$.
   Da $x \notin Frei(A)$, liefern 2 und das Koinzidenzlemma für dieses e:

4. $val_{D,I,\beta_x^e}(A) = W$.
   Aus 4 und 1 hat man

5. $val_{D,I,\beta_x^e}(B) = W$
   im Widerspruch zu 3.

■

4.3.3 Übungsaufgaben

Übungsaufgabe 4.3.1
Ist $x \notin Frei(A)$, dann

$$val_{D,I,\beta}(A) = val_{D,I,\beta_x^d}(A) \text{ für alle } d \in D.$$

Insbesondere: Ist A geschlossen, dann

$$val_{D,I,\beta}(A) = val_{D,I,\beta_x^d}(A) \text{ für alle } x \in Var \text{ und } d \in D.$$

Übungsaufgabe 4.3.2
Geben Sie eine geschlossene prädikatenlogische Formel A an, für die gilt:

Jedes Modell (D, I) von A hat einen unendlich großen Grundbereich D.

Übungsaufgabe 4.3.3
Der Beweis von Teil 2 aus Lemma 4.31 wird durch strukturelle Induktion über A geführt. Beweisen Sie den Induktionsschritt von B auf $\forall x B$.

Übungsaufgabe 4.3.4
Für beliebige Mengen P, Q, R, S gilt:

$$\begin{array}{rcl} S \cap Q & = & \emptyset \\ P & \subseteq & Q \cup R \\ P = \emptyset & \Rightarrow & Q \neq \emptyset \\ Q \cup R & \subseteq & S \end{array} \quad \Rightarrow \quad P \cap R \neq \emptyset$$

Geben Sie eine geschlossene prädikatenlogische Formel A über der Signatur

$$(\emptyset, \{p, q, r, s\}, \alpha(p) = \alpha(q) = \alpha(r) = \alpha(s) = 1\}$$

an, die diesen Sachverhalt modelliert, d.h. jede Interpretation, die die Extensionen von p, q, r, s als Mengen interpretiert, ist ein Modell von A.

Übungsaufgabe 4.3.5 (First–Order n–Damen Problem)
Auf einem $n \times n$ Felder großen Schachbrett sind n Damen so zu verteilen, daß keine eine andere schlagen kann.

Finden Sie geschlossene prädikatenlogischee Formeln q_n so, daß für alle $n \geq 1$ gilt:

q_n ist erfüllbar
gdw
das n–Damen Problem hat wenigstens eine Lösung.

Versuchen Sie eine Lösung zu finden, bei der die Länge von q_n linear mit n wächst.

Übungsaufgabe 4.3.6
Ist M eine Menge von Formeln, in denen auch freie Variablen vorkommen können und A eine Formel, die ebenfalls freie Variablen enthalten kann. Es gibt zwei Möglichkeiten die Definition der logischen Folgerbarkeit, Def.4.39, auf diesen Fall auszudehnen.

Möglichkeit 1 Man erweitert den Modellbegriff auf diese Situation

Definition 4.49 (Modell)
Eine Interpretation $\mathcal{D}$ über Σ heißt **Modell** einer Formel A über Σ, wenn für jedes β gilt $val_{\mathcal{D},\beta}(A) = W$.
$\mathcal{D}$ heißt **Modell** einer Formelmenge M, wenn für jedes β und jede Formel $B \in M$ gilt $val_{\mathcal{D},\beta}(B) = W$.

Möglichkeit 2 Wir bilden den *universellen Abschluß* $Cl_\forall(A)$ einer Formel (A) mit freien Variablen. Sind $x_1, \ldots, x_n$ alle freien Variablen in A, dann ist $Cl_\forall(A) = \forall x_1 \ldots \forall x_n A$. Mit $Cl_\forall(M)$ bezeichnen wir die Menge $\{Cl_\forall(B) \mid B \in M\}$. Jetzt kann die Definition $M \models A$ zurückführen auf den variablenfreien Fall durch $Cl_\forall(M) \models Cl_\forall(A)$.

Zeigen Sie, daß die beiden Möglichkeiten zur selben Folgerungsrelation führen.

Übungsaufgabe 4.3.7
Die folgende Aufgabe ist nur sinnvoll, wenn in A freie Variablen vorkommen. Es soll die Definition 4.49 benutzt werden.
Wenn A ein Modell besitzt, dann ist A erfüllbar. Zeigen Sie, daß die Umkehrung nicht richtig sein muß.

Übungsaufgabe 4.3.8
Wie die vorangegangene Aufgabe 4.3.7 ist auch diese nur sinnvoll, wenn in A freie Variablen vorkommen.

Wenn zwei Formeln logisch äquivalent sind, dann haben sie dieselben Modelle. Geben Sie zwei Formeln an, die dieselben Modelle haben, aber nicht logisch äquivalent sind.

Übungsaufgabe 4.3.9
In der einschlägigen Literatur findet sich gelegentlich eine zweite Definition der Folgerbarkeitsbeziehung von Formeln mit freien Variablen, die von der Definition 4.3.6. Da meist explizit oder stillschweigend angenommen wird, daß in Aussagen über Ableitbarkeit keien freien Variablen vorkommen, findet man in der Regel in diesen Texten keinen Hinweis auf die zwei Varianten der Folgerbarkeitsbeziehung. Hier jetzt die zweite Definition der Folgerbarkeitsbeziehung:

> **Definition 4.50 (Lokale Folgerbarkeit)**
> Es sei $M \subseteq For_\Sigma, A \in For_\Sigma$.
>
> $M \models^{\circ}_{\Sigma} A \quad :\Leftrightarrow$ Für jede Interpretation von $\mathcal{D}$ und jede Belegung β gilt wenn $val_{\mathcal{D},\beta}(M) = W$, dann gilt auch $val_{\mathcal{D},\beta}(A) = W$.

Wenn eine verbale Unterscheidung nötig ist, nennen wir die in Definition 4.39 definierte Relation $\models$ die **globale** und $\models^{\circ}$ die **lokale** Folgerungsbeziehung.

Zeigen Sie:

1. $\models^{\circ}$ schärfer als $\models$: $M \models^{\circ} A \quad \Rightarrow \quad M \models A$, aber i. a. nicht umgekehrt.
2. $A \models^{\circ} B \quad \Leftrightarrow \quad \models^{\circ} A \to B \quad \Leftrightarrow \quad \models A \to B$
3. $\models A \to B \quad \Rightarrow \quad A \models B$ aber i. a. nicht umgekehrt.
4. Ist M variablenfrei, dann gilt für alle A:
$$M \models^{\circ} A \quad \Leftrightarrow \quad M \models A$$

Übungsaufgabe 4.3.10
Diese Aufgabe soll zeigen, wie sich die beiden Folgerungsrelationen auf den variablenfreien Fall zurückführen lassen.

1. $A \models B$ gdw $Cl_\forall A \models^{\circ} Cl_\forall B$
2. $A \models^{\circ} B$ gdw $\bar{A} \models \bar{B}$
Seien $x_1, \ldots, x_k$ alle freien Variablen in A und B und $c_1, \ldots, c_k$ neue paarweise verschiedene Konstanten. $\bar{A}$ entsteht aus A, indem jede Variable x_i durch c_i ersetzt wird. Entsprechend entsteht $\bar{B}$ aus B.

Übungsaufgabe 4.3.11
Beweisen Sie die folgende Behauptung:
A und B sind logisch äquivalent
gdw Für alle $\mathcal{D}$ und β gilt $val_{\mathcal{D},\beta}(A) = val_{\mathcal{D},\beta}(B)$.
gdw $A \models^{\circ} B$ und $B \models^{\circ} A$

Übungsaufgabe 4.3.12

1. Finden Sie eine Formel ϕ_3 der Prädikatenlogik erster Stufe mit leeren Vokabular, so daß $\mathcal{M} \models \phi_3$ genau dann gilt, wenn $\mathcal{M}$ genau drei Elemente hat.
   Die Formel ϕ_3 enthält also als einziges Relationszeichen das Symbol $\doteq$ für die Gleichheit.
2. Geben Sie ϕ_n für beliebige $n \geq 1$ an.

Übungsaufgabe 4.3.13

Definition 4.51
Sei A eine Formel der PL1 ohne freie Variablen. Das **Spektrum** von A, $spec(A)$ ist die folgende Menge natürlicher Zahlen:

$$\begin{array}{rcl} spec(A) & = & \{n \in I\!N : \text{ es gibt ein Modell } \mathcal{D} \text{ von } A, \\ & & \text{dessen Universum genau } n \text{ Elemente hat}\} \end{array}$$

Es kann A auch noch unendliche Modelle besitzen, aber diese spielen für die Definition des Spektrums keine Rolle.

Nach Aufgabe 4.3.12 ist klar, daß es für jede endliche Teilmenge $E \subset \mathbb{N}$ eine erster Stufe Formel Sp_E gibt mit $spec(Sp_E) = E$. Finden Sie

1. eine Formel A mit $spec(A) =$ die Menge aller geraden Zahlen.
2. eine Formel B mit $spec(B) =$ die Menge aller Quadratzahlen.
3. eine Formel C mit $spec(C) =$ die Menge aller Zahlen, die keine Primzahlen sind.

In allen drei Fällen steht Ihnen die Wahl einer passenden Signatur völlig frei.
Kommentar: Das Spektrumsproblem, also die Frage welche Teilmengen von $\mathbb{N}$ können als Spektren von Formeln erster Stufe auftreten, hat in der mathematischen Logik und theoretischen Informatik viel Interesse gefunden. Eine komplexitätstheoretische Charakterisierung wurde bereits 1974 von Jones und Selman gefunden [JS74]: Die Klasse $SPEC$ der Spektren stimmt überein mit der Klasse der Teilmengen von $\mathbb{N}$ die von einer nichtdeterministischen Turingmaschine in 2^{ct} Schritten erkannt werden können, wobei c

eine Konstante und t die Länge der Eingabe ist. Dabei wird eine natürliche Zahl n in Binärdarstellung eingegeben, also $t \approx log_2(n)$.
Die Frage ob $SPEC$ unter mengentheoretischen Komplementen abgeschlossen ist, Vermutung von Asser, ist bis heute ungelöst.

Übungsaufgabe 4.3.14
Finden Sie Formeln A und B, die freie Variablen enthalten können, so daß $A \models B$ gilt aber nicht $\models A \rightarrow B$.

Übungsaufgabe 4.3.15
Sei $\phi(x, y)$ die Abkürzung für die Formel

$$0 \leq y \wedge 0 \leq x \wedge x * x \leq y \wedge y < (x + 1) * (x + 1).$$

Für ganze Zahlen a, b gilt $\mathcal{Z} \models \phi[a, b]$ genau dann wenn a positiv ist und b die positive ganzzahlige Quadratwurzel von a ist. So ist z.B. 2 die positive ganzzahlige Quadratwurzel von 5.
Geben Sie ein Beispiel, daß diese Aussage für $\mathcal{Z}_{Jint} \models \phi[a, b]$ nicht mehr richtig ist.

Zur Definition von $\mathcal{Z}$, $\mathcal{Z}_{Jint}$ siehe Beispiele 4.32 und 4.33 auf Seite 118.
Die Notation $\mathcal{Z} \models \phi[a, b]$ ist eine abkürzende Schreibweise für $\mathcal{Z}, \beta \models \phi$ mit $\beta(x) = a$ und $\beta(y) = b$, siehe Seite 117.

Die Formel ϕ war, in anderer Notation, in [LBR03] als Spezifikation für die ganzzahlige Quadratwurzel benutzt worden. Auf den Fehler in der Spezifikation wurde in [Cha03] aufmerksam gemacht.

Übungsaufgabe 4.3.16
In vielen Programmiersprachen gibt es die Möglichkeit Datenwerte abhängig von einer Bedingung zu referenzieren, z.B. $v =$ `if` $x = 0$ `then` 5 `else` w. Konstrukte der Art `if` ϕ `then` t_1 `else` t_2 nennt man *bedingte Terme* (*conditional terms*). In der Definition 4.3 von Seite 97 kommen bedingte Terme nicht vor. Ihre Einführung würde sogar zu einem völlig neuen Phenomän führen: bisher konnten Terme in Formeln vorkommen, aber umgekehrt keine Formeln in Termen. Das passiert aber in `if` ϕ `then` t_1 `else` t_2. In dieser Aufgabe wollen wir die die formale Einführung bedingter Terme untersuchen. Für die Zwecke dieser Übungsaufgabe erweitern wir die Definition 4.3 um die folgende Zeile

3. Sind t_1, t_2 Terme und ist ϕ eine Formel, so ist `if` ϕ `then` t_1 `else` t_2 ein Term.

Die zugehörige Semantikdefinition 4.28 wird ebenfalls ergänzt. Sei dazu (D, I) eine Interpretation des Vokabulars Σ und β eine Variablenbelegung über D.

3. $val_{D,I,\beta}(\texttt{if } \phi \texttt{ then } t_1 \texttt{ else } t_2) = \begin{cases} val_{D,I,\beta}(t_1) \text{ falls } val_{D,I,\beta}(\phi) = \mathbf{1} \\ val_{D,I,\beta}(t_2) \text{ falls } val_{D,I,\beta}(\phi) = \mathbf{0} \end{cases}$

Zeigen Sie:

1. Finden Sie eine zu

   $$\forall n \forall d(jdiv(n, d) = \texttt{if } (n \geq 0) \texttt{ then } div(n, d) \texttt{ else } -div(-n, d))$$

   äquivalente Formel ohne `if then else` .

2. Sei s beliebiger Term in der erweiterten Syntax und *occ* ein Vorkommen eines bedingten Terms $\texttt{if } \phi \texttt{ then } t_1 \texttt{ else } t_2$ in s. Mit s_1, bzw. s_2 bezeichnen wir die Terme, die aus s entstehen, wenn man *occ* ersetzt durch t_1, bzw. t_2. Dann gilt für jede Interpretation (D, I) und Variablenbelegung β
   $val(s) = \begin{cases} val(s_1) & \text{falls} \quad val(\phi) = W \\ val(s_2) & \text{falls} \quad val(\phi) = F \end{cases}$
   wobei wir der Kürze halber nur *val* geschrieben haben anstelle von $val_{D,I,\beta}$.

3. Sei F eine quantorenfreie Formel und *occ* ein Vorkommen eines bedingten Terms $\texttt{if } \phi \texttt{ then } t_1 \texttt{ else } t_2$ in F. Sei F_1 die Formel, die entsteht, wenn man *occ* in F ersetzt durch t_1 and F_2 die analog entstehende Formel, wenn *occ* durch t_2 ersetzt wird. Dann gilt

   $$F \leftrightarrow (\phi \wedge F_1) \vee (\neg\phi \wedge F_2)$$

4. Zu jeder Formel F in der erweiterten Syntax gibt es eine logisch äquivalente Formel G, die keine bedingten Terme enthält.

Übungsaufgabe 4.3.17
In einem Tutorium zur Vorlesung *Formale System* wurde die folgende Vermutung diskutiert:

> Sei F eine Formel, in der genau die Variablen x, y vorkommen. Außerdem wird angenommen, daß in F nur einstellige Prädikatszeichen und keine Funktionszeichen vorkommen. Dann ist
>
> $$\forall x \exists y F \rightarrow \exists y \forall x F$$
>
> allgemeingültig.

Stimmt diese Vermutung?

4.4 Normalformen

Wie in der Aussagenlogik wollen wir die aufgeführten Fälle logischer Äquivalenz benutzen, um für prädikatenlogische Formeln Normalformen herzustellen. Wir beginnen mit der Einfachsten.

Definition 4.52 (Negationsnormalform)
Eine Formel $A \in For_\Sigma$ ist in **Negationsnormalform**, wenn jedes Negationszeichen in A vor einer atomaren Teilformel steht.
(Insbesondere kommt keine Teilformel der Form $\neg\neg B$ in A vor.)

Lemma 4.53
Zu jeder Formel A gibt es eine logisch äquivalente Formel B in Negationsnormalform.

Definition 4.54
Eine Formel $A \in For_\Sigma$ heißt *bereinigt*, wenn

- $Frei(A) \cap Bd(A) = \emptyset$
- die hinter Quantoren stehenden Variablen paarweise verschieden sind.

Satz 4.55
Zu jeder Formel A gibt es eine äquivalente bereinigte. Sie läßt sich aus A algorithmisch herstellen.

Beweis

Man wende die gebundene Umbenennung an nebst dem Ersetzungstheorem.

■

Definition 4.56 (Pränexe Normalform)
$A \in For_\Sigma$ hat *Pränex-Normalform*, wenn A die Gestalt hat

$$Q_1 x_1 \dots Q_n x_n B$$

mit $Q_i \in \{\forall, \exists\}$, $x_i \in Var (i = 1, \dots, n)$ und: B ist eine quantorenfreie Formel. Man nennt B auch die Matrix von A.

Satz 4.57
Zu jeder Formel A gibt es eine äquivalente in Pränex-Normalform. Sie läßt sich aus A algorithmisch ableiten.

Beweis

Nachdem man A bereinigt hat, wende man von innen nach außen die Äquivalenzen 7 bis 14 in Satz 4.48 (Seite 125) auf Teilformeln an, um sukzessive Quantoren „nach links zu schieben“.

▪

Bemerkungen

Beim geschilderten Verfahren bleibt die Eigenschaft „bereinigt“ erhalten. Man beachte, daß die Pränex-Normalform einer Formel A i. a. nicht eindeutig bestimmt ist. Abhängig von der Reihenfolge der angewandten Äquivalenzen kann man z. B. aus

$$\forall x p(x) \rightarrow \forall y q(y)$$

sowohl

$$\exists x \forall y (p(x) \rightarrow q(y))$$

als auch

$$\forall y \exists x (p(x) \rightarrow q(y))$$

erhalten.

Beispiel 4.58 (Pränex Normalform)

Aus $\forall y\ (\forall x\ \forall y\ p(x,y) \rightarrow \exists x\ r(x,y))$ erhält man sukzessive:

$$\forall y(\forall x\ \forall z\ p(x,z) \rightarrow \exists u\ r(u,y))$$

$$\forall y \exists x(\forall z\ p(x,z) \rightarrow \exists u\ r(u,y))$$

$$\forall y\ \exists x\ \exists z(p(x,z) \rightarrow \exists u\ r(u,y))$$

$$\forall y\ \exists x\ \exists z\ \exists u(p(x,z) \rightarrow r(u,y))$$

Eine Formel in Pränex-Normalform läßt sich noch weiter normieren. Für quantorenfreie Formeln können wir in unmittelbarer Übertragung der Definition aus der Aussagenlogik sagen, wann eine solche in *disjunktiver* bzw. *konjunktiver* Normalform ist. Mit Hilfe der Tautologien läßt sich eine Formel in Pränex-Normalform dann in eine äquivalente überführen, deren Matrix in DNF oder KNF ist.

4.4.1 Skolem-Normalform

Wir gehen zum Abschluß auf eine Normalform ein, die noch wesentlich restriktiver ist als etwa die Pränex-Normalform mit Matrix in KNF. Allerdings kann man jetzt nicht mehr zu beliebig vorgelegter Formel A eine Formel A' in dieser neuen Normalform erhalten, welche logisch äquivalent zu A wäre. Nur eine viel schwächere Äquivalenz zwischen A und A' läßt sich erreichen: A' besitzt genau dann ein Modell, wenn A eines hat. Für das wichtigste Anwendungsgebiet dieser Normierung reicht das aber auch aus: Die Überführbarkeit einer Formel in „Skolem-Normalform" ist grundlegend für die Theorie des automatischen Beweisens.

Als Vorbereitung auf das nächste Lemma betrachten wir anhand dreier Formeln die alternativen Möglichkeiten einen Sachverhalt unter Benutzung von Existenzquantoren auszudrücken (Beispiel 4.59) oder durch Einführung entsprechender Funktionszeichen (Beispiel 4.60). Das Vokabular Σ sei durch die Relations- und Funktionszeichen $\{<, +, 0, 1\}$ gegeben, die in der Interpretation $\mathcal{N}$ über dem Universum der natürlichen Zahlen in der üblichen Weise interpretiert werden. Mit Σ_1, Σ_2, Σ_3 bezeichnen wir die jeweilige Erweiterung von Σ um ein einstelliges Funktionszeichen *do*, um ein einstelliges Funktionszeichen *gr* bzw. um ein zweistelliges Funktionszeichen *diff*.

Beispiel 4.59 (Darstellung mit Existenzquantor)

1. $\forall x \exists y (y \doteq x + x)$
2. $\forall x \exists y (x < y)$
3. $\forall x \forall y \exists z (x < y \to x + z \doteq y)$

Beispiel 4.60 (Darstellung mit Funktionszeichen)

1. $\forall x (do(x) \doteq x + x)$
2. $\forall x (x < gr(x))$
3. $\forall x \forall y (x < y \to x + diff(x, y) \doteq y)$

Alle drei Formeln aus Beispiel 4.59 sind offensichtlich in der Interpretation $\mathcal{N}$ wahr. Damit die Formeln aus Beispiel 4.60 in den entsprechenden Erweiterungen auf die Signaturen Σ_i, bezeichnet mit $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ wahr werden, müssen die neuen Funktionszeichen richtig interpretiert werden:

Beispiel 4.61

1. Dieser Fall ist einfach, da wir für die Definition der Interpretation des Funktionszeichens *do* in $\mathcal{N}_1$ keine Wahl haben:
   Setze $do^{\mathcal{N}_1}(d) = d + d$

2. Hier gibt es sehr viele Möglichkeiten für die Wahl einer Interpetation von gr. Die einzige Forderung, die erfüllt sein muß ist, daß der Funktionswert grösser als das Argument ist. Also etwa:
   Setze $gr^{\mathcal{N}_2}(d) = d + 1$

3. Hier setzen wir etwa:
   $diff^{\mathcal{N}_3}(d_1, d_2) = \begin{cases} d_2 - d_1 \text{ falls } d_1 < d_2 \\ 0 \text{ sonst} \end{cases}$
   Der Wert im Fall $d_2 \leq d_1$ ist hier willkürlich gewählt, aber das stört nicht die Äquivalenz von $\forall x \forall y \exists z (x < y \rightarrow x + z \doteq y)$ zu $\forall x \forall y (x < y \rightarrow x + diff(x, y) \doteq y)$, denn wenn die Prämisse der Implikation nicht erfüllt ist, dann ist die ganze Formel wahr, unabhängig davon, was rechts des Implikationspfeiles steht.

Lemma 4.62 (Beseitigen von Existenzquantoren)
Gegeben sei eine geschlossene Formel über Σ der Gestalt

$$\forall x_1 \ldots \forall x_n \exists y B,$$

in der $x_1, \ldots, x_n, y$ paarweise verschieden sind und $x_1, \ldots, x_n \notin Bd(B)$ ($n = 0$ ist erlaubt, die Formel ist dann $\exists y B$). Wir erweitern Σ um ein neues, $n-$stelliges Funktionssymbol g zu einer Signatur Σ_g (d. h. $F_{\Sigma_g} = F_\Sigma \cup \{g\}$ und $\alpha_{\Sigma_g}(g) = n$).

Dann gilt

$$\begin{aligned} &\forall x_1 \ldots \forall x_n \exists y B \text{ hat ein Modell über } \Sigma \\ &\Leftrightarrow \forall x_1 \ldots \forall x_n \{y/g(x_1, \ldots, x_n)\}(B) \text{ hat ein Modell über } \Sigma_g. \end{aligned}$$

Beweis

$\Rightarrow$:

(D, I) sei Modell von $\forall x_1 \ldots \forall x_n \exists y B$. Das bedeutet: Zu jedem $(d_1, \ldots, d_n) \in D^n$ gibt es ein $e \in D$, so daß für jede Variablenbelegung β gilt

$$val_{D,I,\beta^{d_1,\ldots,d_n,e}_{x_1,\ldots,x_n,y}}(B) = W$$

(Dabei steht $\beta^{d_1,\ldots,d_n,e}_{x_1,\ldots,x_n,y}$ kurz für $((\ldots(\beta^{d_1}_{x_1})\ldots)^{d_n}_{x_n})^e_y$. Im Falle $n = 0$ lautet die obige Aussage einfach: Es gibt e mit $val_{D,I,\beta^e_y}(B) = W$ für alle β.)

Es sei $\bar{g} : D^n \rightarrow D$ eine Funktion, die zu jedem $(d_1, \ldots, d_n)$ ein solches e festlegt. Also:

$$val_{D,I,\beta^{d_1,\ldots,d_n,\bar{g}(d_1,\ldots,d_n)}_{x_1,\ldots,x_n,y}}(B) = W$$

für alle $(d_1, \ldots, d_n)$ und β.

Zur erweiterten Signatur Σ_g definieren wir die Erweiterung I_g von I durch

$$I_g(g) = \bar{g}.$$

Nach dem Substitutionslemma gilt dann für (D, I_g), jede Variablenbelegung β und alle $(d_1, \ldots, d_n) \in D^n$:

$val_{D,I_g,\beta^{d_1,\ldots,d_n}_{x_1,\ldots,x_n}}(\{y/g(x_1, \ldots, x_n)\}(B))$
$= val_{D,I_g,\beta^{d_1,\ldots,d_n,\bar{g}(d_1,\ldots,d_n)}_{x_1,\ldots,x_n,y}}(B)$
$= val_{D,I,\beta^{d_1,\ldots,d_n,\bar{g}(d_1,\ldots,d_n)}_{x_1,\ldots,x_n,y}}(B)$
da I_g und I auf den Signatursymbolen in B übereinstimmen
$= W$

Es folgt, daß (D, I_g) Modell von $\forall x_1 \ldots \forall x_n\{y/g(x_1, \ldots, x_n)\}(B)$ ist.

$\Leftarrow$:

Die Formel

$$\forall x_1 \ldots \forall x_n\{y/g(x_1, \ldots, x_n)\}(B) \to \forall x_1 \ldots \forall x_n \exists y B$$

ist allgemeingültig. (Korollar zum Lemma 4.37 auf Seite 122). Jedes Modell (D, I_g) von $\forall x_1 \ldots \forall x_n\{y/g(x_1, \ldots, x_n)\}(B)$ (über Σ_g) ist also auch Modell von $\forall x_1 \ldots \forall x_n \exists y B$. Da die letztere Formel das neue Symbol g nicht enthält, hat sie auch (D, I) mit

$$I := \text{Einschränkungen von } I_g \text{ auf } F_\Sigma \cup P_\Sigma$$

zum Modell. (D, I) ist Interpretation über Σ.

■

Definition 4.63 (Skolem-Normalform)
Eine Formel ist in *Skolem-Normalform*, wenn sie

- geschlossen ist
- die Gestalt hat
  $\forall x_1 \ldots \forall x_n B$
  (nur Allquantoren im Präfix), mit quantorenfreiem B
- die Matrix B in KNF ist.

Satz 4.64
Zu jedem $A \in For_\Sigma$ gibt es eine endliche Erweiterung Σ_{sk} von Σ und eine Formel $A_{sk} \in For_{\Sigma_{sk}}$ mit

- A_{sk} ist in Skolem-Normalform
- A_{sk} hat ein Modell genau dann, wenn A ein Modell hat.

A_{sk} läßt sich aus A algorithmisch erhalten.

Beweis

Nachdem wir A bereinigt und in Pränex-Normalform gebracht haben, bilden wir den Allabschluß und wenden dann – im Präfix von links nach rechts fortschreitend – Lemma 4.62 an, um der Reihe nach alle Existenzquantoren zu beseitigen. Die Matrix der erhaltenen Formel wird durch die bekannten Tautologien in KNF gebracht.

■

Bemerkung

Die bei Anwendung von Lemma 4.62 neu eingeführten Funktionssymbole nennt man *Skolemfunktionen*, im nullstelligen Fall *Skolemkonstanten*. Das Verfahren selber heißt *Skolemisierung*. Man beachte, daß die neue Signatur Σ_{sk} von der vorgelegten Formel A abhängt.

Beispiel 4.65

Gegeben:
$$\forall x(\exists y(p(y)) \land \exists z(q(x,z)))$$
Pränex Normalform:
$$\forall x\exists y\exists z(p(y) \land q(x,z))$$
Skolem Normalform:
$$\forall x(p(f_1(x)) \land q(x, f_2(x)))$$

Beispiel 4.66

Gegeben:
$$\exists x(p(w,x) \lor \forall y(q(w,x,y) \land \exists z\; r(y,z)))$$
All-Abschluß:
$$\forall w\; \exists x(p(w,x) \lor \forall y(q(w,x,y) \land \exists z\; r(y,z)))$$
Pränex Normalform:
$$\forall w\; \exists x\; \forall y\; \exists z(p(w,x) \lor (q(w,x,y) \land r(y,z)))$$
Skolemisierung:

$$\forall w\, \forall y(p(w, f_1(w)) \vee (q(w, f_1(w), y) \wedge r(y, f_2(w, y))))$$

Matrix in KNF, Skolem Normalform:

$$\forall w\, \forall y((p(w, f_1(w)) \vee q(w, f_1(w), y)) \wedge (p(w, f_1(w)) \vee r(y, f_2(w, y))))$$

4.4.2 Übungsaufgaben

Übungsaufgabe 4.4.1
Weshalb haben wir die Voraussetzung machen müssen, daß $\doteq$ nicht in den Formeln in M auftritt? Was wäre eine naheliegende Verallgemeinerung des Begriffs der Herbrand-Algebra, bei der der obige Satz auch bei Formeln mit $\doteq$ richtig bleibt?

Mengen von variablenfreien Formeln ohne $\doteq$ – wie oben Grundinstanzen(M) – haben besonders schöne Eigenschaften, auf die wir zum Schluß noch hinweisen wollen. Wir wollen etwa annehmen, daß die Formeln in konjunktiver oder disjunktiver Normalform vorliegen. Das nachfolgende Lemma zeigt dann, weshalb solche Formeln günstig sind: insbesondere für automatisches Beweisen mit der Resolutionsregel.

Definition 4.67
Ein *Literal* ist ein Atom oder negiertes Atom. Es heißt *positives* Literal im ersten und *negatives* im zweiten Fall. Ein *Grundliteral* ist ein Literal ohne Variable. Ein *komplementäres Paar* von Literalen ist ein Paar $\{L, \neg L\}$ (L Atom).

Lemma 4.68
Für eine Konjunktion $L_1 \wedge \ldots \wedge L_k, k \geq 1$, von Grundliteralen gilt:

1. $L_1 \wedge \ldots \wedge L_k$ ist nicht allgemeingültig
2. $L_1 \wedge \ldots \wedge L_k$ hat ein Modell $\Leftrightarrow \{L_1, \ldots, L_k\}$ enthält kein komplementäres Paar

Für eine Disjunktion $L_1 \vee \ldots \vee L_k, k \geq 1$ von Grundliteralen gilt:

3. $L_1 \vee \ldots \vee L_k$ hat ein Modell
4. $L_1 \vee \ldots \vee L_k$ ist allgemeingültig $\Leftrightarrow \{L_1, \ldots, L_k\}$ enthält ein komplementäres Paar

Beweis

1. ist trivial.

Zu 2.:

Falls die Konjunktion $L_1 \wedge \ldots \wedge L_k$ ein komplementäres Paar enthält, kann sie sicherlich kein Modell haben.

$L_1 \wedge \ldots \wedge L_k$ enthalte kein komplementäres Paar. Über der gegebenen Signatur Σ betrachten wir die Herbrand-Algebra $(Term^0_\Sigma, I)$, in der festgelegt wird

$$I(p) := \{(t_1, \ldots, t_n) | t_1, \ldots, t_n \in Term^0_\Sigma, p(t_1, \ldots, t_n) \in \{L_1, \ldots, L_k\}\}$$

für Prädikatsymbole p einer Stelligkeit $n \geq 1$;

$$I(P) = W :\Leftrightarrow P \in \{L_1, \ldots, L_k\}$$

für aussagenlogische Atome P.

Wir zeigen, daß $(Term^0_\Sigma, I)$ Modell jedes L_i ist ($i = 1, \ldots, k$), mithin Modell von $L_1 \wedge \ldots \wedge L_k$.

Wenn L_i Atom ist, etwa $L_i = p(t_1, \ldots, t_n)$, dann $(t_1, \ldots, t_n) \in I(p)$ nach Definition von I, d. h. $val_{Term^0_\Sigma, I}(p(t_1, \ldots, t_n)) = val_{Term^0_\Sigma, I}(L_i) = W$. Ist L_i negatives Literal, etwa $L_i = \neg p(t_1, \ldots, t_n)$, dann gilt $p(t_1, \ldots, t_n) \notin \{L_1, \ldots, L_k\}$, da es keine komplementäre Paare gibt. Also ist $val_{Term^0_\Sigma, I}(p(t_1, \ldots, t_n)) = F$ und damit $val_{Term^0_\Sigma, I}(L_i) = val_{Term^0_\Sigma, I}(\neg p(t_1, \ldots, t_n)) = W$. Entsprechend ist die Situation für aussagenlogische Atome P.

Zu 3.:

Insbesondere hat jedes einzelne Grundliteral stets ein Modell, also auch jede Disjunktion $L_1 \vee \ldots \vee L_k$.

Zu 4.:

Schließlich gilt

$$L_1 \vee \ldots \vee L_k \text{ist allgemeingültig} \Leftrightarrow \neg(L_1 \vee \ldots \vee L_k) \text{hat kein Modell}$$

da die L_i geschlossen sind und 4. ergibt sich aus 2..

■

Übungsaufgabe 4.4.2

1. Wo haben wir im Beweis von Lemma 4.62 verwendet, daß die Ausgangsformel geschlossen ist?

2. Wo haben wir die Voraussetzung verwendet, daß $x_1, \ldots, x_n \notin Bd(B)$?

Übungsaufgabe 4.4.3
Natürlich braucht A_{sk} durch A bei weitem nicht eindeutig bestimmt zu sein. Man mache sich das an Beispielen klar. Ferner finde man Beispiele dafür, daß A_{sk} und A nicht logisch äquivalent zueinander sind.

Übungsaufgabe 4.4.4
Die Elimination von Existenzquantoren kann auch ohne vorherigen Übergang zur pränexen Normalform erhalten werden. Dazu definieren wir rekursiv für Formeln A in Negationsnormalform:

- $sk(A) = A$, falls A ein Atom oder das Negat eines Atoms ist.
- $sk(A \wedge B) = sk(A) \wedge sk(B)$
- $sk(A \vee B) = sk(A) \vee sk(B)$
- $sk(\forall x A) = \forall x sk(A)$
- $sk(\exists x A) = sk(x/f(y_1, \ldots y_n))$, wobei $y_1, \ldots y_n$ alle freien Variablen in $\exists x A$ sind.

1. Zeigen Sie, daß A und $sk(A)$ erfüllbarkeitsäquivalent sind.
2. Berechnen Sie $sk(A)$ für die Formeln aus Beispiel 4.65 und 4.66

Übungsaufgabe 4.4.5
Berechnen Sie zuerst die Pränex Normalform und dann die Skolem Normalform für folgende Formeln:

1. $(\forall x\ p(x) \rightarrow \forall x\ q(x)) \rightarrow \forall x(p(x) \rightarrow q(x))$
2. $\exists x(\forall y\ p(x, y) \vee \exists z(p(x, z) \wedge \forall x\ p(z, x)))$

Übungsaufgabe 4.4.6
In der zweiten Version des Satzes von Herbrand, Satz 5.64, war die Allgemeingültigkeit einer existentiellen Formel charakterisiert worden.
Gilt auch die folgende Aussage über die Erfüllbarkeit einer existentiellen Formel. Die Voraussetzungen seien wie in Satz 5.64:

$\exists x \phi$ ist erfüllbar

gdw

es gibt eine natürliche Zahl n und Grundterme $t_1, \ldots, t_n$, sodaß
$\phi(t_1) \vee \ldots \vee \phi(t_n)$ erfüllbar ist.

Kapitel 5

Beweistheorie

5.1 Einleitendes Beispiel

Eine der Wurzeln der modernen Logik ist das Interesse an einer systematischen Analyse des menschlichen Denkens. Nun kann man sich viel denken. Wir fokussieren in dieser Vorlesung auf den Teil des menschlichen Denkens, den man mit *logischem Schließen* enger eingrenzen kann. Als ein Beispiel betrachten wir die Herleitung einer Antwort auf die Frage *Kann mein Bruder mein Schwager sein?*[1] Nehmen wir mal für den Augenblick an, mein Bruder wäre mein Schwager. Als nächstes brauchen wir eine Erklärung, was ein Schwager ist. Nach allgemeinem Sprachgebrauch ist das der Bruder meiner Frau. Wenn aber mein Bruder auch der Bruder meiner Frau ist, dann ist meine Frau meine Schwester. Das ist nach dem deutschen Eherecht verboten. Also kann mein Bruder nicht mein Schwager sein.

Betrachten wir das Argument etwas systematischer. Geben wir als erstes meinem Bruder den Namen *Bruno* und schreiben Zeile um Zeile auf, was wir auf jeder Stufe der Argumentation wissen.

1. *Bruno* ist mein Bruder.
2. *Bruno* ist mein Schwager.
3. Wenn jemand mein Schwager ist, dann ist er ein Bruder meiner Frau.
4. Nach deutschem Eherecht darf niemand mit seiner Schwester verheiratet sein.

Dabei beschreiben 1. und 2. eine aktuelle Situation, 3. ist eine Definition des Begriffs *Schwager* und 4 beschreibt ein allgemeines Faktum. Aus 3 und 2 können wir dann schließen auf

5. *Bruno* ist ein Bruder meiner Frau.

In der informellen Argumentation haben wir aus 1. und 5. ohne Zwischenschritte erschlossen, daß meine Frau meine Schwester sein muß. Das beruht darauf, daß wir ein unmittelbares Verständnis haben, was Bruder und Schwester bedeutet. Um die gesamte Argumentation transparent zu machen, sollten wir dieses Hintergrundwissen explizit in der Liste aufführen.

6. Wenn jemand gleichzeitig mein Bruder und der Bruder einer Frau ist, dann ist diese Frau meine Schwester.

[1] Ich bedanke mich bei meinem früheren Kollegen Wolfgang Schönfeld, von dem ich dieses Beispiel zum ersten Mal gehört habe.

Daraus können wir dann schließen

7. Meine Frau ist meine Schwester.

Die Aussagen 4 und 7 stehen in einem Widerspruch. Die eingangs für einen Augenblick gemachte Annahme ist widerlegt. Zusammenfassend stellen wir fest, daß die Aussagen in den Zeilen 1, 2, 3, 4 und 6 Fakten sind, entweder allgemeine Fakten oder in der augenblicklichen Situation angenommene. Nur die Zeilen 5 und 7 werden aus vorangegangen Zeilen durch einen logischen Schluß gewonnen. Beide Schlüsse folgen demselben Muster.

Schluß vom Allgemeinen zum Besonderen In Zeile 3 wird über eine beliebige Person gesprochen, durch das Wort *jemand* umgangssprachlich ausgedrückt. Daraus folgt dann insbesondere

wenn *Bruno* mein Schwager ist, dann ist *Bruno* der Bruder meiner Frau.

Ebenso wird in 6 unspezifisch von *jemand* und *einer Frau* gesprochen. Daraus folgt die spezielle Aussage

wenn *Bruno* gleichzeitig mein Bruder ist und der Bruder meiner Frau, dann ist meine Frau meine Schwester.

Modus Ponens Wenn eine Aussage der Form *wenn A, dann B* bekannt ist und außerdem A, dann kann man auf B schließen. Diese seit der Antike bekannte logische Schlußfigur ist unter dem Namen *modus ponens* in der philosophischen Logik bekannt. Zeile 5 unserer Beispielargumentation wird aus Zeile 2 und dem obigen Zwischenschritt

wenn *Bruno* mein Schwager ist, dann ist *Bruno* der Bruder meiner Frau

durch Anwendung des *modus ponens* gewonnen.

Die Zeile 7 folgt aus den Zeilen 1 und 5 und dem Zwischenschritt

wenn *Bruno* gleichzeitig mein Bruder ist und der Bruder meiner Frau, dann ist meine Frau meine Schwester

Wir wollen noch einen letzten Schritt gehen und die obige Argumentation in einer formalen Sprache aufschreiben. In der bisherigen Darstellung wurde bewußt nur natürliche Sprache benutzt um zu dokumentieren, daß die Analyse logischen Schließens zunächst nicht an eine formale Sprache gebunden ist. Aber wir haben mit der Prädikatenlogik erster Stufe eine geeignete formale Sprache zur Verfügung und wollen sie jetzt auch einsetzen. Das dabei benutzte Vokabular umfaßt die Konstantensymbole *Bruno* als Bezeichnung für meinen Bruder , und i als Bezeichnung für mich. Außerdem die zweistelligen Relationszeichen $bruder(a, b)$, $schwager(a, b)$, $schwester(a, b)$ und das Funktionszeichen fr für *Frau.* Bei der Formalisierung fällt uns auf, daß in Aussage 6 von einer Frau gesprochen wird. Wir brauchen also noch eine einstelligen Relationszeichen w für weiblich. An dieser stelle fällt uns noch etwas auf, was in einer informellen Argumentation als offensichtlich angenommen wird, daß die Ehefrau einer Person weiblich ist. Diese Faktum fügen wir hinzu. Damit können wir die obige Beweisführung wie folgt formalisieren, wobei wir jetzt auch die in der Analyse identifizierten Zwischenschritte mit aufführen.

1. $bruder(Bruno, i)$
2. $schwager(Bruno, i)$
3. $w(fr(i))$
4. $\forall x(schwager(x, i) \rightarrow bruder(x, fr(i)))$
5. $\forall x(\neg schwester(fr(x), x))$
6. $\forall x \forall y(bruder(x, i) \wedge bruder(x, y) \wedge w(y) \rightarrow schwester(y, i))$
7. $schwager(Bruno, i) \rightarrow bruder(Bruno, fr(i))$
8. $bruder(Bruno, fr(i))$
9. $bruder(Bruno, i) \wedge bruder(Bruno, fr(i)) \wedge w(fr(i))$
   $\rightarrow schwester(fr(i), i)$
10. $schwester(fr(i), i)$
11. $\neg schwester(fr(i), i)$
12. Widerspruch!

Die Beweisregel, welche als Ergebnis Zeile 7 aus Zeile 4 liefert kann anschaulich dargestellt werden als

$$\frac{\forall x(schwager(x, i) \rightarrow bruder(x, fr(i)))}{schwager(Bruno, i) \rightarrow bruder(Bruno, fr(i))}$$

oder allgemein

$$\frac{\forall x(\phi(x))}{\phi(t)}$$

wobei t ein Term ist.

Die modus ponens Regel, welche Zeile 8 aus den Zeilen 2 und 7 liefert stellen wir gleich in allgemeiner Form vor

$$\frac{A \rightarrow B \quad A}{B}$$

Die beiden Regeln *modus ponens* und der Schluß vom Allgemeinen zu Besonderen sind als Beispiele gedacht zur Vorbereitung der systematischen Untersuchung von Regelsystemen in Rest dieses Kapitels.

Vorher wollen wir aber noch einmal zu dem einleitenden Beispiel zurückkehren und es um ein paar Schritte erweitern. Die Aussage 6 auf Seite 143 wirkt ein bisschen gekünstelt. Sie ist so konstruiert, daß sie in den Beweis passt. Eine konsequente Formalisierung von Hintergrundwissen könnte z.B. eine Definition der Bruder- bzw. Schwesterrelation sein. Genauer wird es die Reduktion dieser beiden Begriffe auf die drei grundlegendere Begriffe sein:

$kind(k, m, v)$	soll heißen	k ist Kind von m (Mutter) und v (Vater)
$weibl(a)$	soll heißen	a ist weiblich
$maenn(a)$	soll heißen	a ist männlich

Die gewünschten Definitionen könnten dann so aussehen:

8. $schw(a_1, a_2) \quad \leftrightarrow \quad \exists y \exists z(weibl(a_1) \wedge kind(a_1, y, z) \wedge kind(a_2, y, z))$
9. $brd(a_1, a_2) \quad \leftrightarrow \quad \exists y \exists z(maenn(a_1) \wedge kind(a_1, y, z) \wedge kind(a_2, y, z))$

Aus diesen Definition kann man ableiten.

10. $schw(a_1, a_2) \wedge schw(a_1, a_3) \wedge weibl(a_2) \rightarrow schw(a_2, a_3)$
11. $schw(a_1, a_2) \wedge schw(a_1, a_3) \wedge maenn(a_2) \rightarrow brd(a_2, a_3)$
12. $brd(a_1, a_2) \wedge brd(a_1, a_3) \wedge maenn(a_2) \rightarrow brd(a_2, a_3)$
13. $brd(a_1, a_2) \wedge brd(a_1, a_3) \wedge weibl(a_2) \rightarrow schw(a_2, a_3)$

Die Aussage 6 auf Seite 143 folgt jetzt aus 13 und dem nächsten Stück Hintergrundwissen, daß eine Ehefrau weiblich ist.

Im folgenden Text betrachten die Aussage 10 genauer. Die Analyse der übrigen drei Aussagen verläuft analog und bleibt dem Leser überlassen.

Welche logischen Schlussregeln braucht man um, z.B., 10 aus der Definition abzuleiten? Als erstes ersetzen wir eine atomare Formel durch ihre Definition. Das führt zu:

14. $\exists y \exists z(weibl(a_1) \wedge kind(a_1, y, z) \wedge kind(a_2, y, z)) \wedge$
$\exists y \exists z(weibl(a_1) \wedge kind(a_1, y, z) \wedge kind(a_3, y, z)) \wedge$
$weibl(a_2) \quad \rightarrow \quad schw(a_2, a_3)$

Als nächstes machen wir die beiden Existenzquantoren explizit. Wenn ein Element existiert, dann geben wir ihm einen Namen und ersetzen jedes Vorkommen der quantifizierten Variablem durch diesen Namen. Natürlich müssen wir einen Namen wählen, der neu ist, in dem bisherigen Kontext noch nicht vorkommen ist. Wir entscheiden uns für m_1, m_2 für die $\exists y$ Quantoren und v_1, v_2 für die $\exists z$ Quantoren.

16. $weibl(a_1) \wedge kind(a_1, m_1, v_1) \wedge kind(a_2, m_1, v_1) \wedge$
$weibl(a_1) \wedge kind(a_1, m_2, v_2) \wedge kind(a_3, m_2, v_2) \wedge$
$weibl(a_2) \quad \rightarrow \quad schw(a_2, a_3)$

An dieser Stelle treffen wir auf ein häufig auftretendes Phänomen: wir kommen nicht weiter weil ein weiteres Stück der Formalisierung des Kontextes fehlt: die Eltern eines Kindes sind eindeutig, oder als Formel:

17. $\forall x \forall y_1 \forall y_2 \forall z_1 \forall z_1($
$kind(x, y_1, z_1) \wedge kind(x, y_2, z_2) \rightarrow (y_1 \doteq y_2 \wedge z_1 \doteq z_2))$

Mit Hilfe von 17 können wir aus 16 folgern:

18. $weibl(a_1) \wedge kind(a_1, m_1, v_1) \wedge kind(a_2, m_1, v_1) \wedge$
$weibl(a_1) \wedge kind(a_1, m_2, v_2) \wedge kind(a_3, m_2, v_2) \wedge$
$weibl(a_2) \quad \rightarrow m_1 \doteq m_2 \wedge v_1 \doteq v_2$

Der nächste Schritt kommt aus der Gleichheitslogik. Wir machen Gebrauch von 18 und ersetzen in 16 Gleiches durch Gleiches, d.h. m_2 durch m_1 und v_2 durch v_1 und erhalten die neue Beweisverpflichtung:

19. $weibl(a_1) \wedge kind(a_1, m_1, v_1) \wedge kind(a_2, m_1, v_1) \wedge$
$weibl(a_1) \wedge kind(a_1, m_1, v_1) \wedge kind(a_3, m_1, v_1) \wedge$
$weibl(a_2) \quad \rightarrow \quad schw(a_2, a_3)$

Die linke Seite der Implikation kann man abschwächen, in dem die konkreten Namen m_1, v_1 dort, wo sie vorkommen, durch Existenzaussagen ersetzt werden. Das führt zu

19. $\exists y \exists z(kind(a_2, y, z) \wedge kind(a_3, y, z) \wedge weibl(a_2))$
$\wedge weibl(a_1) \wedge kind(a_1, m_1, v_1) \quad \rightarrow \quad schw(a_2, a_3)$

Jetzt können wir die Definition aus 8, in der umgekehrten Richtung wie beim ersten Mal und erhalten die Tautologie:

21. $schw(a_2, a_3) \wedge weibl(a_1) \wedge kind(a_1, m_1, v_1) \quad \rightarrow \quad schw(a_2, a_3)$

Hiermit schließen wir das motivierende Beispiel ab.

Im weiteren Verlauf diese Kapitels wollen wir, wie angekündigt, Regelsysteme für das logische Schließen untersuchen. Solche Regelsysteme nennt man in der formalen Logik auch *Kalküle*. In der philosophischen Logik wurden Kalkülregeln gerechtfertigt durch einen Appell an die Intuition. Es gibt eine erstaunliche Übereinstimmungen darüber, was verschiedene Personen für einen korrekten logischen Schluß halten. In der mathematischen Logik nennen wir einen Kalkül *korrekt*, wenn jede aus der leeren Menge ableitbare Formel allgemeingültig ist. Ein Kalkül heißt vollständig, wenn, umgekehrt, jede allgemeingültige Aussage in ihm hergeleitet werden kann. Es gibt eine erstaunliche Fülle verschiedener korrekter und vollständiger Kalküle. Wir wollen vier davon behandeln:

- Hilbert-Kalkül
- Resolutionskalkül
- Tableaukalkül
- Sequenzenkalkül

Der Hilbertkalkül ist wegen seine wissenschaftsgeschichtlichen Bedeutung interessant und von allen vier Kalkülen dem menschlichen Schließen am nächsten. Der Resolutionskalkül ist für das automatische Beweisen erfunden worden und auch heute noch der Kern vieler Theorembeweiser. Der Tableaukalkül bildet in besonders einfacher und direkter Weise das semantikbezogene Schließen nach. Sequenzenkalküle, manchmal auch Gentzen-Systeme genannt, wurden von Gerhard Gentzen in [Gen35] (siehe auch [Gen69]) eingeführt für beweistheoretische Untersuchungen. Eine umfassende Darstellung unter besonderer Berücksichtigung der Bedeutung des Kalküls für das automatische Theorembeweisen findet man in [Gal86]. Der Sequenzenkalkül ist eine einfache Variante eines Tableaukalküls, der als Vorbild für viele Implementierungen von Theorembeweisern dient. Der Tableaukalkül ist aus pädagogischer Sicht für die Präsentation in einer Vorlesung besser geeignet.

Alle vier Kalküle präsentieren wir die Kalkülreglen zunächst für die Aussagenlogik und danach für die Prädikatenlogik. Für Hilbertkalkül und Resolutionskalkül beschränken wir Korrektheits- und Vollständigkeitsuntersuchungen ausschließlich auf die aussagenlogische Version. Der Schwerpunkt liegt auf dem Tableaukalkül, wo wir auch Korrektheit und Vollständigkeit für die Prädikatenlogik betrachten. Für den Sequenzenkalkül führen wir nur vor, in welcher Weise er als Variante des Tableaukalküls verstanden werden kann.

5.2 Hilbertkalkül

David Hilbert ist einer der wesentlichen Begründer der *axiomatische Logik*. Seine Arbeiten markieren den Übergang der Logik von einer philosophischen Disziplin zur *Mathematischen Logik*.

Kalküle vom Hilbert-Typ sind weniger auf Effizienz (Kürze der Beweise) und Komfort (leichtes Finden von Beweisen, Flexibilität im Formulieren von Beweisideen) hin ausgelegt, als auf Sparsamkeit.

Die nachfolgende Darstellung folgt im wesentlichen [Men87].

5.2.1 Axiome und Regeln

Der folgende Kalkül arbeitet nicht auf der vollen Sprache $For0_\Sigma$, sondern auf ihrer Einschränkung auf die Basis $\{\neg, \to, \forall\}$. Will man Formeln der vollen Sprache behandeln, muß man **1**, **0**, $\wedge$, $\vee$, $\leftrightarrow$ und $\exists$ durch ihre Definitionen in dieser Basis ersetzen.

Definition 5.1 (Hilbertkalkül)
Der Hilbertkalkül (für eine Signatur Σ) bezeichnet mit $\mathbf{H}$ (genauer mit $\mathbf{H}_\Sigma$)wird durch die folgenden Axiome und Regeln gegeben:
Axiome und Regeln werden als Schemata angegeben. $x, y, z, \ldots$ stehen für Variablen, f für ein Funktionssymbol, p für ein Relationssymbol, t für einen Term, A, B, C stehen für Formeln.

Axiome

Ax1	$A \to (B \to A)$	(Abschwächung)
Ax2	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$	(Verteilung von $\to$)
Ax3	$(\neg A \to \neg B) \to (B \to A)$	(Kontraposition)
Ax4	$\forall x A \to \{x/t\}(A)$ wobei $\{x/t\}$ kollisionsfrei für A	(Instantiierung)
Ax5	$\forall x(A \to B) \to (A \to \forall x B)$ wobei $x \notin$ *Frei(A)*	($\forall$-Verschiebung)
Gl1	$x \doteq x$	(Reflexivität
Gl2	$x \doteq y \to y \doteq x$	(Symmetrie)
Gl3	$x \doteq y \to (y \doteq z \to x \doteq z)$	(Transitivität)
Gl4	$x_1 \doteq y_1 \to (x_2 \doteq y_2 \to (\ldots(x_n \doteq y_n \to f(x_1,\ldots,x_n) \doteq f(y_1,\ldots,y_n))\ldots))$	
Gl5	$x_1 \doteq y_1 \to (x_2 \doteq y_2 \to (\ldots(x_n \doteq y_n \to p(x_1,\ldots,x_n) \to p(y_1,\ldots,y_n))\ldots))$	

Regeln

Mp: $\dfrac{A, A \to B}{B}$ (Modus ponens)

Gen: $\dfrac{A}{\forall x A}$ (Generalisierung)

Ableitbarkeit im Hilbertkalkül wird bezeichnet mit $\vdash_{\mathbf{H}}$.

Lemma 5.2
Jedes Axiom von **H** ist allgemeingültig.

Beweis *Direktes Nachrechnen*

■

Beispiel 5.3
Für jede Formel A gilt: $\vdash_{\mathbf{H}} A \to A$.

Beweis Wir geben eine Ableitung aus $\emptyset$ an.

1. $(A \to ((\underbrace{A \to A}_{B}) \to \underbrace{A}_{C})) \to ((A \to (\underbrace{A \to A}_{B})) \to (A \to \underbrace{A}_{C}))$ (Ax2)
2. $A \to ((A \to A) \to A)$ Ax1
3. $(A \to (A \to A)) \to (A \to A)$ Mp auf (2),(1)
4. $A \to (A \to A)$ Ax1
5. $A \to A$ Mp auf (3),(4)

H eignet sich nicht besonders gut, um Ableitungen zielgerichtet zu finden. Einiges hierzu läßt sich gleichwohl sagen.

Zunächst einmal haben wir das folgende „Kompositionslemma“

Lemma 5.4 (Komposition mittels Modus ponens)

$$M_1 \vdash_{\mathbf{H}} A, \quad M_2 \vdash_{\mathbf{H0}} A \to B \quad \Rightarrow \quad M_1 \cup M_2 \vdash_{\mathbf{H}} B.$$

Das wichtigste Instrument jedoch für das Finden einer Ableitung bildet der folgende Satz, der eine bestimmte Austauschbarkeit zwischen dem metasprachlichen Operator $\vdash_{\mathbf{H0}}$ und dem objektsprachlichen Operator $\to$ zum Inhalt hat.

Satz 5.5 (Deduktionstheom)
Für beliebige Formelmengen M und Formeln A, B, wobei A eine geschlossene Formel (siehe Def. 4.9 auf Seite 100) ist, gilt:

$$M \vdash_{\mathbf{H}} A \to B \quad \Leftrightarrow \quad M \cup \{A\} \vdash_{\mathbf{H}} B$$

Beweis Wir lassen den Index **H** weg.

$\Rightarrow$.

$$\begin{array}{llcll} \text{Es gelte} & M & \vdash & A \to B. & \text{Dann} \\ & M \cup \{A\} & \vdash & A \to B. & \text{(erst recht)} \\ & M \cup \{A\} & \vdash & A & \text{(trivialerweise)} \\ & M \cup \{A\} & \vdash & B & \text{(Mp)} \end{array}$$

$\Leftarrow$.

Es gelte: $M \cup \{A\} \vdash B$. Sei $(A_1, \ldots, A_m)$ Ableitung von B aus $M \cup \{A\}$. Wir wollen zeigen: $M \vdash A \to B$, d. h. $M \vdash A \to A_m$. Wir zeigen sogar für alle $i \in \{1, \ldots, m\}$, daß $M \vdash A \to A_i$, und zwar durch Induktion über i. Sei i mit $1 \leq i \leq m$ gegeben. Nehmen wir also an, daß für alle $j < i$ schon gezeigt ist: $M \vdash A \to A_j$.

Fall 1:

$$\begin{array}{ll} A_i \in M \cup \text{ Ax1} \cup \text{Ax2} \cup \text{Ax3}. & \text{Dann gilt:} \\ M \vdash A_i & \text{(trivial)} \\ M \vdash A_i \to (A \to A_i) & \text{(Ax1)} \\ M \vdash A \to A_i & \text{(Mp)} \end{array}$$

Fall 2:

$A_i = A$. In Beispiel 5.3 haben wir schon gezeigt: $M \vdash A \to A_i$.

Fall 3:

Es gibt $j < i$ und $k < i$ mit $A_k = A_j \to A_i$. Nach obiger Annahme (Induktionsvoraussetzung) wissen wir:

$M \vdash A \to A_j$

$M \vdash A \to (A_j \to A_i)$

Man hat ferner

$M \vdash (A \to (A_j \to A_i)) \to ((A \to A_j) \to (A \to A_i))$ (Ax2)

also

$M \vdash A \to A_i$ (2mal Mp).

Fall 4:

Es gibt $j < i$ mit $A_i = \forall x A_j$.

$M \vdash A \to A_j$	*Induktionsvoraussetzung*
$M \vdash \forall x(A \to A_j)$	*Generalisierungsregel*
$M \vdash \forall x(A \to A_j) \to (A \to \forall x A_j)$	Axiom $Ax5$ hier machen wir Gebrauch von der Voraussetzung an A
$M \vdash A \to \forall x A_j$	MP

∎

Wir erinnern daran, daß $M \models A$ für die semantische Folgerbarkeit von A aus M steht (siehe Definition 4.39 auf Seite 123).

Beispiel 5.6 (Eine Ableitung in H)
Wir schreiben in diesem Beispiel der Einfachheit halber $\vdash$ statt $\vdash_{\mathbf{H}}$.

Sei $M = \{r(x,x), r(x,y) \wedge r(x,z) \to r(y,z)\}$. Wir wollen

$$M \vdash r(x,y) \to r(y,x)$$

zeigen. (Dasselbe Beispiel wird auch in dem Buch [HW90], S. 82ff benutzt, allerdings mit einem anderen Kalkül.)

1	$M \vdash r(x,y) \wedge r(x,z) \to r(y,z)$	$(Vor.)$
2	$M \vdash \forall z(r(x,y) \wedge r(x,z) \to r(y,z))$	$(Gen.)[1]$
3	$\emptyset \vdash \forall z(r(x,y) \wedge r(x,z) \to r(y,z)) \to (r(x,y) \wedge r(x,x) \to r(y,x))$	$(Ax4)$
4	$M \vdash r(x,y) \wedge r(x,x) \to r(y,x)$	$(MP)[2,3]$
5	$\emptyset \vdash (r(x,y) \wedge r(x,x) \to r(y,x)) \to$ $(r(x,x) \to ((r(x,y) \wedge r(x,x) \to r(y,x)) \wedge r(x,x)))$	(AL-Tautologie $A \to (B \to A \wedge B)$
6	$M \vdash r(x,x) \to ((r(x,y) \wedge r(x,x) \to r(y,x)) \wedge r(x,x))$	$(MP)[4,5]$
7	$M \vdash r(x,x)$	$(Vor.)$
8	$M \vdash (r(x,y) \wedge r(x,x) \to r(y,x)) \wedge r(x,x)$	$(MP)[6,7]$
9	$\emptyset \vdash ((r(x,y) \wedge r(x,x) \to r(y,x)) \wedge r(x,x))$ $\to (r(x,y) \to r(y,x))$	(AL-Tautologie)
10	$M \vdash r(x,y) \to r(y,x)$	$(MP)[8,9]$

5.2.2 Beispielableitungen

Wir geben weitere Beispiele ableitbarer Formeln an (aus [Men87]). Die hingeschriebenen Beweise sind nicht unmittelbar Ableitungen, sondern Folgen von Ableitbarkeitsaussagen, die mit Hilfe von

- Axiomen
- dem Kompositionslemma (Mp)
- dem Deduktionstheorem

gewonnen sind und aus denen sich Ableitungen dann unmittelbar herstellen lassen.

Mit **H**0 bezeichnen wir den aussagenlogischen Teil von **H**. Er besteht aus den Axiomen $Ax1$ bis $Ax3$ und der Modus Ponens Regel.

Beispiel 5.7
Für beliebige Formeln A, B, C sind in **H**0 aus $\emptyset$ ableitbar

(1)	$(A \to B) \to ((B \to C) \to (A \to C))$	(Transitivität)
(2)	$A \to ((A \to B) \to B)$	(Modus ponens)
(3)	$\neg\neg A \to A$	(Doppelte Negation)
(4)	$A \to \neg\neg A$	
(5)	$(A \to B) \to (\neg B \to \neg A)$	(Kontraposition)
(6)	$A \to (\neg B \to \neg(A \to B))$	(Semantik von $\to$)
(7)	$\neg A \to (A \to B)$	
(8)	$A \to (B \to (A \to B))$	
(9)	$\neg(A \to B) \to A$	
(10)	$\neg(A \to B) \to \neg B$	
(11)	$(A \to \neg A) \to \neg A$	(Selbstwiderlegung)
(12)	$(\neg A \to A) \to A$	
(13)	$(A \to B) \to ((\neg A \to B) \to B)$	(Fallunterscheidung)
(14)	$\neg(A \to A) \to B$	(Ex falso quodlibet)

Beweis

Wir sollten versuchen, so oft wie möglich das Deduktionstheorem (DT) zu verwenden. Statt $\vdash_{\mathbf{H0}}$ schreiben wir kurz $\vdash$.

(1)	$\{A \to B, B \to C, A\} \vdash A$	(trivial)
	$\{A \to B, B \to C, A\} \vdash A \to B$	(trivial)
	$\{A \to B, B \to C, A\} \vdash B$	(Mp)
	$\{A \to B, B \to C, A\} \vdash B \to C$	(trivial)
	$\{A \to B, B \to C, A\} \vdash C$	(Mp)
	$\{A \to B, B \to C\} \vdash A \to C$	(DT)
	$\{A \to B\} \vdash (B \to C) \to (A \to C)$	(DT)
	$\vdash (A \to B) \to ((B \to C) \to (A \to C))$	(DT)

(Versuchen Sie (eine Zeit lang) direkt, also ohne Verwendung des Deduktionstheorems, eine Ableitung $(A_1, \ldots, A_n)$ von $(A \to B) \to ((B \to C) \to (A \to C))$ aus $\emptyset$ hinzuschreiben.)

$$
\begin{array}{lll}
(2) & \{A, A \to B\} \vdash A & \text{(trivial)} \\
& \{A, A \to B\} \vdash A \to B & \text{(trivial)} \\
& \{A, A \to B\} \vdash B & \text{(Mp)} \\
& \vdash A \to ((A \to B) \to B) & \text{(2mal DT)}
\end{array}
$$

$$
\begin{array}{lll}
(3) & \vdash \neg\neg A \to (\neg\neg\neg\neg A \to \neg\neg A) & \text{(Ax1)} \\
& \{\neg\neg A\} \vdash \neg\neg\neg\neg A \to \neg\neg A & \text{(DT)} \\
& \{\neg\neg A\} \vdash (\neg\neg\neg\neg A \to \neg\neg A) \to (\neg A \to \neg\neg\neg A) & \text{(Ax3)} \\
& \{\neg\neg A\} \vdash \neg A \to \neg\neg\neg A & \text{(Mp)} \\
& \{\neg\neg A\} \vdash (\neg A \to \neg\neg\neg A) \to (\neg\neg A \to A) & \text{(Ax3)} \\
& \{\neg\neg A\} \vdash \neg\neg A \to A & \text{(Mp)} \\
& \{\neg\neg A\} \vdash A & \text{(DT)} \\
& \vdash \neg\neg A \to A & \text{(DT)}
\end{array}
$$

$$
\begin{array}{lll}
(4) & \vdash \neg\neg\neg A \to \neg A & \text{(mit (3))} \\
& \vdash (\neg\neg\neg A \to \neg A) \to (A \to \neg\neg A) & \text{(Ax3)} \\
& \vdash A \to \neg\neg A & \text{(Mp)}
\end{array}
$$

$$
\begin{array}{lll}
(5) & \{A \to B\} \vdash (\neg\neg A \to \neg\neg B) \to (\neg B \to \neg A) & \text{(Ax3)} \\
& \{A \to B\} \vdash \neg\neg A \to A & \text{(mit (3))} \\
& \{A \to B\} \vdash A \to B & \text{(trivial)} \\
& \{A \to B\} \vdash \neg\neg A \to B & \text{(mit (1), 2mal Mp)} \\
& \{A \to B\} \vdash B \to \neg\neg B & \text{(mit (4))} \\
& \{A \to B\} \vdash \neg\neg A \to \neg\neg B & \text{(mit (1), 2mal Mp)} \\
& \{A \to B\} \vdash \neg B \to \neg a & \text{(Mp)} \\
& \vdash (A \to B) \to (\neg B \to \neg A) & \text{(DT)}
\end{array}
$$

$$
\begin{array}{lll}
(6) & \vdash A \to ((A \to B) \to B) & \text{(mit (2))} \\
& \{A\} \vdash (A \to B) \to B & \text{(DT)} \\
& \{A\} \vdash ((A \to B) \to B) \to (\neg B \to \neg(A \to B)) & \text{(siehe (5))} \\
& \{A\} \vdash \neg B \to \neg(A \to B) & \text{(Mp)} \\
& \vdash A \to (\neg B \to \neg(A \to B)) & \text{(DT)}
\end{array}
$$

$$
\begin{array}{lll}
(7) & \vdash \neg A \to (\neg B \to \neg A) & \text{(Ax1)} \\
& \{\neg A\} \vdash \neg B \to \neg A & \text{(DT)} \\
& \{\neg A\} \vdash (\neg B \to \neg A) \to (A \to B) & \text{(Ax3)} \\
& \{\neg A\} \vdash (A \to B) & \text{(Mp)} \\
& \vdash \neg A \to (A \to B) & \text{(DT)}
\end{array}
$$

(8) $\{A\} \vdash B \to (A \to B)$ (Ax1)
$\vdash A \to (B \to (A \to B))$ (DT)

(9) $\vdash \neg A \to (A \to B)$ (mit (7))
$\vdash \neg(A \to B) \to \neg\neg A$ (mit (5), Mp)
$\vdash \neg\neg A \to A$ (mit (3))
$\vdash \neg(A \to B) \to A$ (mit (1), 2mal Mp)

(10) $\vdash B \to (A \to B)$ (Ax1)
$\vdash \neg(A \to B) \to \neg B$ (mit (5), Mp)

(11) $\vdash A \to (\neg\neg A \to \neg(A \to \neg A))$ (mit (6))
$\{A\} \vdash \neg\neg A \to \neg(A \to \neg A)$ (mit DT)
$\{A\} \vdash \neg\neg A$ (mit (4), Mp)
$\{A\} \vdash \neg(A \to \neg A)$ (Mp)
$\vdash A \to \neg(A \to \neg A)$ (DT)
$\vdash (A \to \neg(A \to \neg A)) \to (\neg\neg(A \to \neg A) \to \neg A)$ (mit (5))
$\vdash \neg\neg(A \to \neg A) \to \neg A$ (Mp)
$\vdash (A \to \neg A) \to \neg\neg(A \to \neg A)$ (mit (4))
$\vdash (A \to \neg A) \to \neg A$ (mit (1), 2mal Mp)

(12) $\vdash (\neg A \to \neg\neg A) \to \neg\neg A$ (mit (11))
Verwende nun (3) und (4).

(13) $\{(A \to B), (\neg A \to B)\} \vdash (A \to B)$ (trivial)
$\{(A \to B), (\neg A \to B)\} \vdash (\neg B \to \neg A)$ (mit (5), Mp)
$\{(A \to B), (\neg A \to B)\} \vdash (\neg A \to B)$ (trivial)
$\{(A \to B), (\neg A \to B)\} \vdash (\neg B \to B)$ (mit (1), 2mal Mp)
$\vdash (\neg B \to B) \to B$ (mit (12))
$\{(A \to B), (\neg A \to B)\} \vdash B$ (Mp)
$\vdash (A \to B) \to ((\neg A \to B) \to B)$ (2mal DT)

(14) $\{\neg B\} \vdash A \to A$
$\vdash (A \to A) \to \neg\neg(A \to A)$ (mit (4))
$\{\neg B\} \vdash \neg\neg(A \to A)$ (Mp)
$\vdash \neg B \to \neg\neg(A \to A)$ (DT)
$\vdash \neg(A \to A) \to B$ (Ax3, Mp)

Für die folgenden Formelschemata versuche der Leser selbst, die Ableitbarkeit zu beweisen. Wie schon gesagt, sind $\mathbf{1}$, $\mathbf{0}$, $\vee$, $\wedge$, $\leftrightarrow$ als Abkürzungen aufzufassen.

(15)	$A \rightarrow \mathbf{1}$	(20)	$A \rightarrow (A \vee B)$
(16)	$\mathbf{0} \rightarrow A$	(21)	$A \wedge B \leftrightarrow B \wedge A$
(17)	$A \rightarrow (B \rightarrow A \wedge B)$	(22)	$A \wedge A \leftrightarrow A$
(18)	$A \vee B \leftrightarrow B \vee A$	(23)	$A \wedge A \rightarrow A$
(19)	$A \vee A \leftrightarrow A$	(24)	$\neg A \leftrightarrow (A \rightarrow \mathbf{0})$

■

5.2.3 Metatheoreme

Satz 5.8 (Vollständigkeit und Korrektheit von H)
Σ sei eine Signatur der PL1. Für jede Formelmenge $M \subseteq For_\Sigma$ und jede Formel $A \in For_\Sigma$ gilt:

1. $M \vdash_{\mathbf{H}} A \Rightarrow M \models A$
2. $M \models A \Rightarrow M \vdash_{\mathbf{H}} A$

Beweis Siehe z.B. [Men87].

■

5.2.4 Übungsaufgaben

Übungsaufgabe 5.2.1
Zeigen Sie unter Verwendung von Ax3 und Ax4 (siehe Def. 5.1 auf Seite 150) $\vdash_H Cl_\forall \neg A \rightarrow \neg Cl_\forall A$.

Übungsaufgabe 5.2.2
Beweisen Sie im Hilbert-Kalkül: $A, \neg A \vdash B$.

Übungsaufgabe 5.2.3
Der Lambdakalkül ist ein wichtiges theoretisches Werkzeug für das Studium typisierter Programmiersprachen. Eine gute Darstellung gibt [Pie02]. Eine interessantes Phänomen, das Typsysteme und Logik verbindet, ist der *Curry-Howard Isomorphismus.* Diese Aufgabe ist diesem Isomorphismus gewidmet.

Als Vorbereitung für die eigentliche Aufgabenstellung definieren wir die Mengen $lambdaTyp$ der Typen und die Menge $lambdaTerm$ der Lambdaterme des einfach getypten Lambda-Kalküls, siehe [Pie02, Chapter 9]

Definition 5.9
Die Menge der Typen $lambdaTyp$ des einfach getypten Lambda-Kalküls ist gegeben durch

1. $U \subseteq\ lambdaTyp$ für eine Menge U atomarer Typen
2. mit $\tau_1, \tau_2 \in\ lambdaTyp$ gilt auch $\tau_1 \to \tau_2 \in\ lambdaTyp$

Die Menge der Terme $lambdaTerm$ des einfach getypten Lambda-Kalküls ist gegeben durch

1. jede Variable x vom Type τ ist eine Lambdaterm vom Typ τ.
2. ist x eine Variable vom Type τ_1 und M ein Lambdaterm vom Typ τ_2, dann ist $\lambda x.M$ ein Lambdaterm vom Typ $\tau_1 \to \tau_2$.
3. Ist M ein Lambdaterm vom Typ $\tau_1 \to \tau_2$ und N ein Lambdaterm vom Typ τ_1, dann ist MN ein Lambdaterm vom Typ τ_2.

Man kann sich - in groben Näherung - atomare Typen als (unstrukturierte) Mengen vorstellen und $\tau_1 \to \tau_2$ als die Menge aller totalen Funktionen von τ_1 nach τ_2. Ebenso kann man sich $\lambda x.M$ vorstellen als die Funktion, die als Wert an der Stelle a die Auswertung von M annimmt, wobei in dieser Auswertung x durch a ersetzt wird. Schließlich kann man sich MN als Auswertung der durch M repräsentierten Funktion auf das durch N repräsentierte Argument vorstellen. Die nachfolgenden Aufgaben, und überhaupt große Teile der Theorie des Lambdakalküls machen nicht Gebrauch von einer formalen Semantik der involvierten Begriffe.

Definition 5.10
Die Menge $FV(M)$ der freien Variablen eines Lambdaterms M sind alle in M vorkommenden Variablen, die nicht im Bereich eines λ-Operators liegen. Formal

1. $FV(x) = \{x\}$
2. $FV(\lambda x.M) = FV(M) \setminus \{x\}$
3. $FV(MN) = FV(M) \cup FV(N)$

U sei die Menge der atomaren Typen. Benutzt man dasselbe U zur Bezeichnung aussagenlogischer Atome, dann kann man jeden Lambdatyp τ als aussagenlogische Formel auffassen. Umgekehrt ist dann jede aussagenlogische Formel, die nur das Implikationszeichen als Operator benutzt ein Lambdatyp.

Nach dieser Vorbereitung können wir die folgenden Aufgabe stellen.

1. Sei F eine aussagenlogische Formel, die nur das Implikationszeichen als Operator und nur Atomen aus U enthält. Falls es einen Lambdaterm M ohne freie Variablen mit Typ F gibt, dann ist F eine aussagenlogische Tautologie.

2. Geben Sie Lambdaterme an, die mit Hilfe von Teil 1 zeigen, daß die Axiome $Ax1$ und $Ax2$ des Hilbertkalküls (siehe Definition 5.1 auf Seite 150) Tautologien sind.

3. Geben Sie Lambdaterme an, die wie in der vorangegangenen Aufgabe zeigen, daß
   $$(A \to B) \to ((B \to C) \to (A \to C))$$
   $$((A \to B) \to C) \to (A \to (B \to C))$$
   Tautologien sind.

5.3 Resolutionskalkül

Merkmale des Resolutionskalküls

- Widerlegungskalkül
- Voraussetzung: Alle Formeln sind in konjunktiver Normalform.
- Es gibt eine einzige Regel (Resolution), die eine Modifikation des Modus ponens ist
- Es sind keine logischen Axiome mehr notwendig; insbesondere entfällt die Suche nach („langen“) passenden Instantiierungen der Axiomenschemata Ax1, Ax2, Ax3.

5.3.1 Der aussagenlogische Kalkül

Wir benutzen die zu Beginn des Unterkapitels 3.1 eingeführte mengentheoretische Schreibweise für konjunktive Normalformen.

Definition 5.11 (Resolutionsregel)
Die *aussagenlogische Resolutionregel* ist die Regel

$$\frac{C_1 \cup \{P\}, C_2 \cup \{\neg P\}}{C_1 \cup C_2} \quad \text{mit: } P \in \Sigma, C_1, C_2 \text{ Klauseln über } \Sigma.$$

$C_1 \cup C_2$ heißt *Resolvente* von $C_1 \cup \{P\}$, $C_2 \cup \{\neg P\}$.

Die Resolutionsregel ist eine Modifikation des modus ponens. Es ist nämlich auch die Erweiterung von MP,

$$(\text{MP}') \qquad \frac{A \vee C,\ A \to B}{C \vee B}, \quad \text{d.h.} \quad \frac{A \vee C,\ \neg A \vee B}{C \vee B}$$

eine gültige Regel. Setzt man $A := P$, $C_1 := C$ und $C_2 := B$, so erhält man gerade die Resolutionsregel.

Definition 5.12 (Resolutionskalkül)
Sei M eine Klauselmenge.

1. Mit $Res(M)$ bezeichnen wir die Menge aller Resolventen von Klauseln aus M.

2. $R_0^0(M) = M$

3. $R_0^{n+1}(M) = Res(R^n) \cup R_0^n$

4. $M \vdash_{\mathbf{R}0} A$ gilt genau dann, wenn es ein n gibt mit $A \in R_0^n(M)$

Der Index 0 soll wieder daran erinnern, daß wir es hier mit der aussagenlogischen Variante des Resolutionskalküls zu tun haben. Die Erweiterung auf die Prädikatenlogik erfolgt später, siehe Def. 5.19.

Da **R**0 als Widerlegungskalkül gedacht ist, interessieren wir uns für Situationen $M \vdash_{\mathbf{R}0} \Box$, wo also aus der Klauselmenge M die leere Klausel durch Resolution ableitbar ist. Wenn aus dem Kontext ersichtlich schreiben wir $\vdash$ einfach anstelle von $\vdash_{\mathbf{R}0}$.

Beispiel 5.13

Gegeben sei die Klauselmenge
$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$

$$\frac{\{P_1, P_2\}, \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\frac{\{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}}{\{\neg P_1\}}$$

$$\frac{\{P_1\}, \{\neg P_1\}}{\Box}$$

Insgesamt:

$$M \vdash_{Res} \Box$$

5.3.2 Korrektheit und Vollständigkeit

Vorbemerkung

Im Gegensatz zu **H**0 enthält **R**0 keine logischen Axiome. Im Sinne des direkten Beweisens kann **R**0 also nicht vollständig sein. Z. B. gilt für die Formel $P \to P$ – oder gleichbedeutend für die Klausel $\{\neg P, P\}$ – sicherlich

$$\emptyset \not\vdash_{\mathbf{R}0} \{\neg P, P\}.$$

R0 soll ja auch als Widerlegungskalkül arbeiten, und zwar auf Klauselmengen, d. h. Formeln in KNF. Es wird also (wenn wir zunächst wieder in unserer vertrauten Sprache mit Formeln aus $For0_\Sigma$ formulieren) eine Folgerbarkeitsaussage

$$M \models A$$

($M \subseteq For0_\Sigma$, $A \in For0_\Sigma$) zuerst einmal umformuliert in die gleichbedeutende

$$M \cup \{\neg A\} \text{ ist unerfüllbar}$$

oder

$$M \cup \{\neg A\} \models \underline{false}.$$

Bringt man die Formeln $M \cup \{\neg A\}$ in KNF und schreibt sie dann als Klauselmenge, etwa $\overline{M}$, schreibt man ferner wieder $\Box$ statt $\underline{false}$, so erhält man als wiederum gleichbedeutende Aussage

$$\overline{M} \text{ ist unerfüllbar}$$

oder

$$\overline{M} \models \Box.$$

R0 wäre als Widerlegungskalkül korrekt und vollständig, wenn dies, für beliebige $\overline{M}$, äquivalent wäre zur Ableitbarkeit von $\Box$ aus $\overline{M}$ in **R**0.

In unserem Beispiel $P \to P$ haben wir die $\{\neg[P \to P]\}$ entsprechende Klauselmenge zu bilden, also $\{\{\neg P\}, \{P\}\}$. Sie liefert in einem einzigen Resolutionsschritt $\Box$.

Satz 5.14 (Korrektheit und Vollständigkeit)
Es sei M eine Menge von Klauseln über einer aussagenlogischen Signatur Σ. Dann gilt

$$M \text{ unerfüllbar} \quad \Leftrightarrow \quad M \vdash_{\mathbf{R0}} \Box.$$

Beweis

Korrektheit:

Es ist zu zeigen, daß aus $M \vdash_{\mathbf{R0}} \Box$ folgt, daß M unerfüllbar ist.

Wir zeigen dazu: wenn eine Menge M_0 von Klauseln erfüllbar ist, dann ist auch $M_1 = M_0 \cup \{R\}$ erfüllbar, wobei R eine Resolvente von M_0 ist.

Wir überlegen uns zuerst, daß diese Aussage unser Problem tatsächlich löst. Eine Klauselmenge die $\Box$ enthält, kann nicht erfüllbar sein. Erreichen wir also von einer Klauselmenge M ausgehend durch iterierte Anwendung der Resolutionsregel eine Klauselmenge, die $\Box$ enthält, dann ist auch M nicht erfüllbar.

Sei jetzt also M_0 erfüllbar, $C_1 \cup \{P\}$ und $C_2 \cup \{\neg P\}$ Klauseln in M_0 und $M_1 = M_0 \cup \{C_1 \cup C_2\}$. Sei I eine erfüllende Interpretation für M_0. Dann gilt $I(C_1 \cup \{P\}) = \mathbf{W}$ und $I(C_2 \cup \{\neg P\}) = \mathbf{W}$. Je nachdem, ob $I(P) = \mathbf{F}$ oder $P(\neg P) = \mathbf{F}$ gilt, gilt $I(C_1) = \mathbf{W}$ oder $I(C_2) = \mathbf{W}$. Auf jeden Fall aber $I(C_1 \cup C_2) = \mathbf{W}$.

Vollständigkeit:
Wir zeigen die gewünschte Behauptung durch Kontraposition, d.h. wir gehen von der Voraussetzung aus, daß $\Box$ aus M mit Hilfe des Kalküls **R**0 nicht herleitbar ist und beweisen die Erfüllbarkeit von M.
Wir fixieren eine feste Reihenfolge der Atome, z. B. die durch den Index gegebene, $P_0, \dots, P_n, \dots$.
Mit M_0 bezeichnen wir die Menge aller Klauseln C, für die $M \vdash_{R0} C$ gilt. Es gilt also $M \subseteq M_0$ und $\Box \notin M_0$. Wir werden im folgenden eine Interpretation I angeben, so daß $val_I(M_0) = W$ ist, d.h. für alle $C \in M_0$ ist $val_I(C) = W$. Insbesondere ist dann M als erfüllbar nachgewiesen.
Wir definieren I induktiv. Angenommen $I(P_0), \dots, I(P_{n-1})$ sind schon festgelegt, dann setzen wir

$$I(P_n) = F,$$

genau dann, wenn gilt:

> M_0 enthält eine Klausel $C = C_1 \cup \{\neg P_n\}$, so daß in C_1 nur Atome P_i mit $i < n$ auftreten, und $val_I(C_1) = F$.

Für den Indunktionsanfang besagt das:

$$I(P_0) = W \Leftrightarrow \{\neg P_0\} \notin M_0.$$

Es bleibt nachzuweisen, daß für alle $C \in M_0$ gilt $val_I(C) = W$. Für jede Klausel C sei $index(C) = i+1$, wobei i die größte natürliche Zahl, so daß P_i, negiert oder unnegiert, in C vorkommt. Mit anderen Worten, $index(C)$ ist die kleinste Zahl, so daß für alle in C vorkommenden Atome P_i gilt $i < index(C)$. Beachten Sie, daß $\Box \notin M_0$. Die Behauptung wird durch Induktion über $index(C)$ bewiesen.
Im Induktionsanfang, $index(C) = 0$, muß $C = \Box$ gelten. Die Behauptung ist erfüllt weil $\Box \notin M_0$ gilt.

Sei also für alle $C \in M_0$ mit $index(C) < n$, $n \geq 1$ schon $val_I(C) = W$ nachgewiesen und $D \in M_0$ mit $index(D) = n$ vorgelegt.
1.Fall $D = D_1 \cup \{\neg P_{n-1}\}$, $index(D_1) < n$
Falls schon $val_I(D_1) = W$ gilt, dann gilt natürlich umso mehr $val_I(D) = W$. Falls $val_I(D_1) = F$, dann sind wir genau in der Situation, die $I(P_{n-1}) = F$

in der Definition von I erzwungen hat. Woraus auch $val_I(D) = W$ folgt.
2.Fall $D = D_1 \cup \{P_{n-1}\}$, $index(D_1) < n$
Falls schon $val_I(D_1) = W$ gilt, dann sind wir natürlich fertig. Also betrachten wir im folgenden nur noch die Möglichkeit $val_I(D_1) = F$. Falls $I(P_{n-1}) = W$ gilt, dann haben wir sofort $val_I(D) = W$, wie gewünscht. Wenn $I(P_{n-1}) = F$ gilt, dann muß es dafür nach Definition von I einen Grund geben, d.h. eine Klausel $C = C_1 \cup \{\neg P_{n-1}\} \in M_0$, so daß $index(C_1) < n$ und $val_I(C_1) = F$. Da M_0 abgeschlossen ist unter Ableitbarkeit in R0 muß auch die Resolvente $R = C_1 \cup D_1$ als das Ergebnis der Resolution von C mit D in M_0 liegen. Da offensichtlich $index(R) < n$ gilt, muß nach Induktionsvoraussetzung $val_I(R) = W$ sein. Das steht aber im Widerspruch zu $val_I(D_1) = F$ und $val_I(C_1) = F$. Der zuletzt betrachtete Fall kann also nicht auftreten und $val_I(D) = W$ ist in allen Fällen nachgewiesen.
3.Fall $D = D_1 \cup \{P_{n-1}, \neg P_{n-1}\}$, $index(D_1) < n$
In diesem Fall ist D eine Tautologie und $val_I(D) = W$ ist in jedem Fall wahr.

∎

Beispiel 5.15

Als ein erstes Beispiel für das Arbeiten mit dem Resolutionskalkül stellen wir uns die Aufgabe zu zeigen, daß

$$(A \to B) \to ((B \to C) \to (A \to C))$$

eine Tautologie ist.

Dazu werden wir zeigen, daß die Negation dieser Formel

$$\neg((A \to B) \to ((B \to C) \to (A \to C)))$$

nicht erfüllbar ist.

Als erstes muß diese Formel in Klauselnormalform gebracht werden. Das Ergebnis ist

$$M = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}$$

Aus M läßt sich wie folgt die leere Klausel ableiten

(1)	[]	$\{\neg A, B\}$
(2)	[]	$\{\neg B, C\}$
(3)	[]	$\{A\}$
(4)	[]	$\{\neg C\}$
(5)	$[1, 3]$	$\{B\}$
(6)	$[2, 5]$	$\{C\}$
(7)	$[4, 6]$	$\Box$

Um Resolutionsbeweise leichter lesbar zu machen numerieren wir alle darin vorkommenden Klauseln durch und geben in eckigen Klammern jeweils die Nummern der Elternklauseln an.

Notwendigkeit der Mengenschreibweise
Die Menge von Formeln

$$E = \{P_1 \vee \neg P_2, \neg P_1 \vee P_2, \neg P_1 \vee \neg P_2, P_1 \vee P_2\}$$

ist sicherlich nicht erfüllbar.
Resolutionsmöglichkeiten:

$$\frac{\neg P_1 \vee \neg P_2, P_1 \vee \neg P_2}{\neg P_2 \vee \neg P_2} \qquad \frac{\neg P_1 \vee \neg P_2, \neg P_1 \vee P_2}{\neg P_1 \vee \neg P_1}$$
$$\frac{\neg P_1 \vee \neg P_1, P_1 \vee \neg P_2}{\neg P_1 \vee \neg P_2} \qquad \frac{\neg P_2 \vee \neg P_2, \neg P_1 \vee P_2}{\neg P_1 \vee \neg P_2}$$

Auf diese Weise ist $\Box$ nicht herleitbar.

5.3.3 Der prädikatenlogische Kalkül

Um unsere Darstellung möglichst knapp zu halten, verzichten wir auf die Behandlung der Gleichheit.

Im Übrigen sei auf die Literatur verwiesen, insbesondere auf die beiden Bücher von [HK89] und [SA91]. Aber auch die Klassiker auf diesem Gebiet, [CL73] und [Lov78], sind auch heute noch mit Gewinn zu lesen.

Definition 5.16 (Klausel)
Ein **Literal** ist eine atomare oder eine negierte atomare Formel.
Eine **Klausel** ist eine endliche Menge von Literalen.
Die **leere Klausel** wird mit $\Box$ bezeichnet.

Eine Klausel wird interpretiert wie die Disjunktion ihrer Literale. Eine Menge von Klauseln wird, wie jede Menge von Formeln, interpretiert wie die (unendliche) Konjunktion der den Klauseln entsprechenden Disjunktionen. Alle auftretenden Variablen werden als universell quantifiziert interpretiert. Die leere Klausel, $\Box$, erhält immer den Wahrheitswert F.

Definition 5.17 (Variante)
Ist A eine quantorenfreie Formel und σ eine Variablenumbenennung, dann heißt $\sigma(A)$ eine *Variante* von A.

Notation

Zu einem Literal L sei $\sim L$ das Literal

$$\sim L := \begin{cases} \neg L & \text{wenn } L \text{ eine atome Formel ist} \\ L' & \text{wenn } L = \neg L' \text{ für eine atomar Formel } L' \text{ ist.} \end{cases}$$

Zu einer Klausel C sei $\sim C := \{\sim L | L \in C\}$.

Definition 5.18 (Resolutionsregel)
Über der Menge der Klauseln (über der prädikatenlogischen Signatur Σ) ist die *Resolution* die folgende Regel

$$\frac{C_1 \cup K_1 \qquad C_2 \cup K_2}{\mu(C_1 \cup C_2)}$$

wobei gilt:

- C_1, C_2, K_1, K_2 sind Klauseln
- K_1, $K_2 \neq \Box$
- $Var(C_1 \cup K_1) \cap Var(C_2 \cup K_2) = \emptyset$
- μ ist allgemeinster Unifikator von $K_1 \cup \sim K_2$.

Eine Klausel C heißt *Resolvente* zweier Klauseln, wenn sie durch Anwendung der Resolutionsregel auf diese entsteht.

Definition 5.19 (Ableitbarkeit im Resolutionskalkül)
Sei M eine Klauselmenge.

1. Mit $Res(M)$ bezeichnen wir die Menge aller Resolventen von Klauseln aus M. Genauer:

$$\begin{array}{rl} Res(M) = & \{B \mid \text{ es gibt Varianten } D_1, D_2 \text{ von Klauseln aus } M, \\ & \text{so daß } B \text{ eine Resolvente von } D_1, D_2 \text{ ist.}\} \end{array}$$

2. $R^0(M) = M$
3. $R^{n+1}(M) = Res(R^n) \cup R^n$
4. $M \vdash_{\mathbf{R}} A$ gilt genau dann, wenn es ein n gibt mit $A \in R^n(M)$

Beispiel 5.20 (Anwendung der Resolutionsregel)
Gegeben seien die beiden Klauseln

$$\{p(x), q(f(x)), q(f(g(c))\} \text{ und } \{r(y,z), \neg q(z), \neg q(f(y))\}$$

Zur Vorbereitung der Anwendung der Resolutionsregel zerlegen wir die Klauseln in komplementäre Klauseln K_1 und K_2 und die jeweiligen Restklauseln C_1 und C_2.

$$\begin{array}{ll} K_1 = \{q(f(x)), q(f(g(c))\} & C_1 = \{p(x)\} \\ K_2 = \{\neg q(z), \neg q(f(y))\} & C_2 = \{r(y,z)\} \end{array}$$

Die Menge $K_1 \cup \sim K_2$ ist also in diesem Fall $\{q(z), q(f(y)), q(f(x)), q(f(g(c))\}$ mit $\sigma = \{x/g(c),\ y/g(c),\ z/f(g(c))\}$ als allgemeinstem Unifikator. Die Resolvente ist daher $\{p(g(c)),\ r(g(c), f(g(c)))\}$.

Am häufigsten wird die Resolutionsregel für einelementige komplementäre Klauseln angewandt. Diese spezielle Form heißt **binäre Resolution**.

Beispiel 5.21 (Binäre Resolution)
Gegeben seien die beiden Klauseln $\{p(x), q(f(x))\}$ und $\{r(y,c), \neg q(f(c))\}$.

$$\begin{array}{ll} K_1 = \{q(f(x))\} & C_1 = \{p(x)\} \\ K_2 = \{\neg q(f(c))\} & C_2 = \{r(y,c)\} \end{array}$$

Die Menge $K_1 \cup \sim K_2$ ist also in diesem Fall $\{q(f(x)), q(f(c))\}$ und unifizierbar durch den allgemeinsten Unifikator $\sigma = \{x/c\}$. Die Resolvente ist daher $\{p(c), r(y,c)\}$.

Satz 5.22 (Korrektheit und Vollständigkeit von R)
Sei M eine Menge von Klauseln in der Signatur Σ.

1. Gilt $M \vdash_{\mathbf{R}} \Box$, dann ist M nicht erfüllbar.
2. Ist M nicht erfüllbar, dann folgt $M \vdash_{\mathbf{R}} \Box$

Beweis
zu 1: Wir zeigen für eine beliebige Klauselmenge M für alle $n \geq 0$

wenn $R^n(M)$ erfüllbar ist, dann ist auch $R^{n+1}(M)$ erfüllbar

Daraus folgt dann:

$$\begin{array}{lll} \text{wenn } M \vdash_{\mathbf{R}} \Box & \text{dann} & \Box \in R^n(M) \text{ für ein } n, \\ & \text{dann} & R^n(M) \text{ unerfüllbar für ein } n, \\ & \text{dann} & R^0(M) = M \text{ unerfüllbar.} \end{array}$$

Kommen wir zum Beweis der induktiven Behauptung. Sei dazu (D, I) ein Modell von $R^n(M)$. Da alle Variablen als implizit universell quantifiziert interpretiert werden, gilt für alle Klauseln $K \in R^n(M)$ und alle Variablenbelegungen β $(D, I, \beta) \models K$. Wir streben an, für jede Resolvente C von Klauseln aus $R^n(M)$ auch $(D, I, \beta) \models C$ für jedes β nachzuweisen. Damit wären wir dann auch fertig.
Sei also C eine Resolvente von $C_1, C_2 \in R^n(M)$. Genauer heißt das nach der allgemeinen Definition einer Resolvente, daß $C_1 = C_{1,1} \cup K_1$, $C_2 = C_{2,1} \cup K_2$, μ ein allgemeinster Unifikator von $K_1 \cup \sim K_2$ und $C = \mu(C_{1,1} \cup C_{2,1})$. Nach Voraussetzung gilt für eine beliebige Variablenbelegung β

$$(D, I, \beta) \models \mu(C_{1,1}) \vee \mu(K_1) \quad \text{und} \quad (D, I, \beta) \models \mu(C_{2,1}) \vee \mu(K_2)$$

Gilt $(D, I, \beta) \models \mu(C_{1,1})$ dann gilt um so mehr $(D, I, \beta) \models \mu(C_{1,1}) \vee \mu(C_{2,1})$ und wir sind fertig. Somit bleibt noch der Fall $(D, I, \beta) \models \mu(K_1)$ zu betrachten. Da μ Unifikator von $K_1 \cup \sim K_2$ ist, folgt daraus $(D, I, \beta) \not\models \mu(K_2)$ und damit muß nach Voraussetzung $(D, I, \beta) \models \mu(C_{2,1})$ gelten. Was wieder $(D, I, \beta) \models C$ nach sich zieht.
Das obige Argument mit dem Unifikator kann man sich klar machen, wenn man den einfachsten Fall $K_1 = \{L_1\}$, $K_2 = \{\neg L_2\}$ betrachtet mit $\mu(L_1) = \mu(L_2)$.
Bemerkungen: Da Variablen implizit als universell quantifiziert angesehen werden macht es für das vorliegende Argument keinen Unterschied, ob wir ein $C \in R^n(M)$ betrachten oder einer Variante C' für ein $C \in R^n(M)$. Offensichtlich spielte die Variablendisjunktheitsbedingung aus der Definition einer Resolvente für obigen Beweis keine Rolle. Ebenso die Forderung, daß die K_i nicht trivial sind.

zu 2: Die Beweisstrategie zielt darauf ab, den Vollständigkeitsbeweis für die aussagenlogische Resolution, Satz 5.14, zu benutzen. Wir beginnen mit der Beobachtung, daß aus der vorausgesetzten Unerfüllbarkeit von M nach dem Herbrandschen Satz (Satz 5.66, Implikation von (1) nach (3)), auch die Unerfüllbarkeit der Menge $\bar{M}$ aller Grundinstanzen von M folgt. Wir ersetzen in $\bar{M}$ jede atomare Formel durch ein aussagenlogisches Atom - natürliche alle Vorkommen einer atomaren Formel durch dasselbe Atom. Die entstehende Menge aussagenlogischer Formel M_0 ist ebenfalls nicht erfüllbar. Warum? Aus einer erfüllenden Belegung I für M_0 könnten wir ein Herband-Modell $\mathcal{H}$ für $\bar{M}$ bauen, indem wir $\mathcal{H} \models p(t_1, \ldots, t_n)$ genau dann setzen wenn $I(A) = \mathbf{1}$ gilt für das der atomaren Formel $p(t_1, \ldots, t_n)$ zugeordnete aussagenlogische Atome A. Nach Satz 5.14 kann man mit aussagenlogischer Resolution aus M_0 die leere Klausel $\Box$ herleiten. Die Frage, ob man aus einer aussagenlogischen Herleitung eine prädikatenlogische Herleitung gewinnen kann, ist unter dem

Namen *lifting* bekannt. Die Antwort ist natürlich positiv. Genauer werden wie zeigen:

$$LiftingLemma$$

für jedes n und jede aussagenlogische Klausel $C_0 \in R_0^n(M_0)$
gibt es eine prädikatenlogische Klausel $C \in R^n(M)$
und eine Grundsubstitution $\sigma,$ so daß
C_0 aus $\sigma(C)$ durch die oben beschriebene Ersetzung
von Grundliteralen durch aussagenlogische Atome entsteht.

Wir wissen bisher schon, daß es ein n gibt mit $\Box \in R_0^n(M_0)$. Nach dem Lifting Lemma gibt es dann eine Klausel $C \in R^n(M)$ und eine Substitution σ mit $\sigma(C) = \Box$. Dann kann nur $C = \Box$ sein und wir sind fertig. Es bleibt also nur noch das Lifting Lemma zu beweisen.
Der Beweis geschieht durch Induktion über n. Für $n = 0$ reduziert sich die Behauptung des Lemmas auf die Definition von M_0. Für den Schritt von n nach $n+1$ betrachten wir zwei Klauseln $C_{1,0}, C_{2,0} \in R_0^n(M_0)$ und eine ihrer Resolventen $C_0 \in R_0^{n+1}(M_0)$. Etwa $C_0 = \{L_1^0, \ldots, L_k^0\}$ mit $C_{1,0} = \{L_1^0, \ldots, L_k^0, L^0\}$ und $C_{2,0} = \{L_1^0, \ldots, L_k^0, \neg L^0\}$. Nach Induktionsvoraussetzung gibt es $C_1, C_2 \in R^n(M)$ und Substitutionen σ_1, σ_2 mit $\sigma_1(C_1) = C_{1,0}$ und $\sigma_2(C_2) = C_{2,0}$. Wir unterschlagen hier die Ersetzung von Grundliteralen durch aussagenlogische Atome. Das ist für den Beweis unerheblich. Wir können außerdem annehmen, daß C_1 und C_2 keine gemeinsamen Variablen haben. Denn einerseits läßt sich eine Variablenumbennenung die C_1 und C_2 variablendisjunkt macht durch entsprechende Änderungen von σ_1 und σ_2 kompensieren anderseits ist die Bildung von Varianten der Ausgangsklauseln eines Resolutionsschritts explizit erlaubt und die Variablendisjunktheit sogar gefordert. Wegen der Variablendisjunktheit von C_1 und C_2 können wir σ_1 und σ_2 zu einer Substitution μ zusammenfassen mit $\mu(C_1) = C_{1,0}$ und $\mu(C_2) = C_{2,0}$. Schauen wir uns C_1 und C_2 etwas genauer an $C_1 = C_{1,1} \cup C_{1,2}$ und $C_2 = C_{2,1} \cup C_{2,2}$ mit $\mu(C_{1,2}) = \mu(\sim C_{2,2}) = L$ und $C_0 = \mu(C_{1,1}) \cup \mu(C_{2,1})$. Man beachte, daß $C_{1,2}$ und $C_{2,2}$ keine Einerklauseln sein müssen. Die allgemeiner Form der prädikatenlogischen Resolution fordert das auch nicht. Wir haben noch eine kleine Klippe zu überwinden. Wir wissen, daß μ ein Unifikator von $C_{1,2}$ und $\sim C_{2,2}$ ist. Der Resolutionsschritt wird mit einem allgemeinsten Unfikator μ_a durchgeführt, wobei etwa $\mu = \mu' \circ \mu_a$ gilt. Das Ergebnis der prädikatenlogischen Resolution ist $C = \mu_a(C_{1,1} \cup C_{2,1})$ mit $C \in R^{n+1}(M)$. Somit gilt $\mu'(C) = \mu(C_{1,1} \cup C_{2,1}) = C_0$ und wir sind fertig.

■

Arbeiten mit dem Resolutionskalkül

Der Resolutionskalkül ist ein Widerlegungskalkül, d.h. der Beweis einer logischen Folgerbarkeit wird in einen Nachweis der Widersprüchlichkeit umgeformt. Wir fassen die Vorgehensweise zusammen.

Gegeben: Eine endliche Menge $\{B_1, \ldots, B_n\} \subseteq For_\Sigma$ und eine Formel $A \in For_\Sigma$.

Gesucht: Für den Fall, daß $\{B_1, \ldots, B_m\} \models A$ ein Beweis dieser Tatsache unter Verwendung von **R**.

Verfahren:

1. Bilde $E := \{Cl_\forall B_1, \ldots, Cl_\forall B_m, \neg Cl_\forall A\}$
2. Bringe jede Formel in E in Pränex-Normalform, erhalte E',
3. Bringe jede Formel in E' in Skolem-Normalform, erhalte E''. (Die Matrizen der Formeln sind in KNF.)
4. Lasse die $\forall$-Präfixe weg; schreibe die verbleibenden Konjunktionen von Disjunktionen (von Literalen) als Mengen von Klauseln; bilde die Vereinigung dieser Klauselmengen; erhalte $\overline{E}$.
5. Versuche, aus $\overline{E}$ mittels Variantenbildung von Klauseln und Resolution die leere Klausel herzuleiten.

Zu Schritt 3 beachte man, daß die bei der Skolemisierung eingeführten Funktionssymbole paarweise verschieden sind und verschieden sind von allen Symbolen in E und allen bisher eingeführten Skolemfunktionen.

Die Variantenbildung in Schritt 5 hat den Zweck, die für die Anwendung der Resolution erforderliche Variablendisjunktheit von Klauseln herzustellen.

Das nächste Beispiel zeigt die Notwendigkeit der Variantenbildung für die Vollständigkeit des Resolutionskalküls.

Beispiel 5.23
Die Menge der beiden Klauseln $\{p(x)\}$ und $\{\neg p(f(x))\}$ hat sicherlich kein Modell, da sie der prädikatenlogischen Formel $(\forall x p(x)) \wedge (\forall x \neg p(f(x)))$ entspricht. Aber $p(x)$ und $p(f(x))$ sind nicht unifizierbar, also ist auch die leere Klausel so nicht herleitbar.

Weitere Beispiele zum Resolutionskalkül

Wir wollen die folgende logische Folgerung beweisen:

Beispiel 5.24

$$\forall x \forall y (x \subseteq y \leftrightarrow \forall u (u \epsilon x \rightarrow u \epsilon y)) \models \forall x \forall y \forall z (x \subseteq y \wedge y \subseteq z \rightarrow x \subseteq z)$$

Es handelt sich dabei um eine Aussage aus der elementaren Mengenlehre zur Transitivität der Teilmengenbeziehung. Bemerkenswert ist vielleicht, daß die Transitivität allein aus der Definition der Teilmengenrelation gefolgert werden soll, ohne zusätzliche mengentheoretische Axiome über die ϵ–Relation.

Bevor wir mit einem Beweis im Resolutionskalkül beginnen können, müssen wir die Prämisse und die Negation der Behauptung in Klauselnormalform transformieren. Dazu zerlegen wir die Prämisse in die beiden Teile

$$\forall x \forall y (x \subseteq y \rightarrow \forall u (u \epsilon x \rightarrow u \epsilon y))$$

$$\forall x \forall y (\forall u (u \epsilon x \rightarrow u \epsilon y) \rightarrow x \subseteq y)$$

Benutzen wir die Relationszeichen *conteq* und *memb* anstelle der Infixzeichen $\subseteq$ und ϵ, so läßt sich die erste Formel direkt in die Klausel

$$\{\neg conteq(x, y), \neg memb(u, x), memb(u, y)\}$$

umschreiben.

Die zweite Formel wird nach Elimination von $\rightarrow$ zunächst zu

$$\forall x \forall y (\exists u (u \epsilon x \wedge \neg u \epsilon y) \vee x \subseteq y)$$

und nach Skolemisierung zu

$$\forall x \forall y ((f(x, y) \epsilon x \wedge \neg f(x, y) \epsilon y) \vee x \subseteq y)$$

Nach Anwendung des Distributivgesetzes entstehen die beiden Klauseln:

$$\{memb(f(x, y), x), conteq(x, y)\}$$

$$\{\neg memb(f(x, y), y), conteq(x, y)\}$$

Die Negation der Behauptung führt zu

$$\exists x \exists y \exists z (x \subseteq y \wedge y \subseteq z \wedge \neg x \subseteq z)$$

und nach Einführung von Skolemkonstanten zu den drei Einerklauseln:

$conteq(a, b)$
$conteq(b, c)$
$\neg conteq(a, c)$

Der Resolutionsbeweis (wir lassen zur Vereinfachung die Mengenklammern weg)

(1)	$\neg conteq(x, y), \neg memb(u, x), memb(u, y)$	[Vor.]
(2)	$memb(f(x, y), x), conteq(x, y)$	[Vor.]
(3)	$\neg memb(f(x, y), y), conteq(x, y)$	[Vor.]
(4)	$conteq(a, b)$	[¬ Beh.]
(5)	$conteq(b, c)$	[¬ Beh.]
(6)	$\neg conteq(a, c)$	[¬ Beh.]
(7)	$\neg memb(u, a), memb(u, b)$	[4,1]
(8)	$\neg memb(u, b), memb(u, c)$	[5,1]
(9)	$\neg memb(f(a, c), c)$	[6,3]
(10)	$memb(f(a, c), a)$	[6,2]
(13)	$memb(f(a, c), b)$	[7,10]
(19)	$memb(f(a, c), c)$	[8,13]
(20)	$\square$	[19,9]

Dieser Beweis wurde von dem automatischen Beweissystem OTTER gefunden. Die Lücken in der Nummerierung weisen darauf hin, daß zwischendurch Klauseln erzeugt wurden, die zum Beweis nichts beitragen.

Die binäre Resolutionsregel alleine liefert keinen vollständigen Kalkül, wie das nächste Beispiel zeigt.

Beispiel 5.25
Die Menge bestehend aus den beiden Klauseln

$$\{P(x), P(y)\}, \{\neg P(u), \neg P(v)\}$$

ist sicherlich unerfüllbar. Die binäre Resolutionsregel liefert, selbst bei beliebiger Wiederholung, als Resolventen nur Varianten der Klausel

$$\{P(y), \neg P(v)\}$$

und nie die leere Klausel. Entscheidend für dieses Phänomen ist der Teil der Definition 5.18, der die Variablendisjunktheit der Elternklauseln verlangt.

Die Regel

$$\frac{\{L_1, \ldots, L_k, L_{k+1}, \ldots, L_n\}}{\{\sigma(L_1), \ldots, \sigma(L_k)\}},$$

wobei σ so gewählt ist, daß $\sigma(L_k) = \sigma(L_{k+1}) = \ldots = \sigma(L_n)$, heißt **Faktorisierungsregel**.
Binäre Resolution zusammen mit der Faktorisierungsregel ergibt wieder einen vollständige, häufig eingesetzten Kalkül.

Auch das zweite Beispiel eines Resolutionsbeweises stammt aus der elementaren Mengenlehre. Es soll bewiesen werden

Beispiel 5.26

$$\forall x \forall y(x \subseteq y \land y \subseteq x \to x = y).$$

Als Voraussetzungen stehen dazu zur Verfügung

- die Definition der Teilmengenrelation

  $$\forall x \forall y(x \subseteq y \leftrightarrow \forall u(u \in x \to u \in y))$$

  und

- die Definition der Gleichheit (in der Mengenlehre ist die Gleichheit eine definierte Relation)

  $$\forall x \forall y(x = y \leftrightarrow \forall u(u \in x \leftrightarrow u \in y))$$

Die Transformation liefert zunächst die schon bekannten Klauseln
(1) $\neg conteq(x, y) \lor \neg memb(u, x) \lor memb(u, y)$
(2) $memb(f(x, y), x) \lor conteq(x, y)$
(3) $\neg memb(f(x, y), y) \lor conteq(x, y)$
Ersetzen wir das Infixzeichen $=$ durch eq, so liefert die Definition der Gleichheit die Klausel
(4) $eq(x, y) \lor memb(g(x, y), x) \lor memb(g(x, y), y)$
(5) $eq(x, y) \lor \neg memb(g(x, y), x) \lor \neg memb(g(x, y), y)$
(6) $\neg eq(x, y) \lor memb(u, x) \lor memb(u, y)$
(7) $\neg eq(x, y) \lor \neg memb(u, x) \lor \neg memb(u, y)$
mit der neuen Skolemfunktion $g(x, y)$. Die Negation der Behauptung schließlich führt zu den Klauseln
(8) $conteq(a, b)$
(9) $conteq(b, a)$
(10) $\neg eq(a, b)$
mit den Skolemkonstanten a, b, c

(11)	$\neg memb(x,a) \vee memb(x,b)$	[8,1]
(12)	$\neg memb(x,b) \vee memb(x,a)$	[9,1]
(18)	$memb(g(a,x),b) \vee eq(a,x) \vee memb(g(a,x),x)$	[11,4]
(23)	$\neg memb(g(x,b),a) \vee eq(x,b) \vee \neg memb(g(x,b),x)$	[11,5]
(28)	$memb(g(a,b),b) \vee eq(a,b)$	[Fak 18]
(29)	$\neg memb(g(a,b),a) \vee eq(a,b)$	[Fak 23]
(61)	$memb(g(a,b),b)$	[28,10]
(62)	$memb(g(a,b),a)$	[61,12]
(69)	$eq(a,b)$	[29,62]
(70)	$\square$	[69,10]

Der Beweis wurde wieder mit dem Beweiser OTTER gefunden, der nicht die Mengennotation verwendet und außerdem die Faktorisierungsregel (Fak) benutzt. Man erhält aus dem obigen Beweis einen Beweis ohne Faktorisierung, wenn man die Mengenschreibweise benutzt und die Beweisschritte (28) und (29) einfach wegläßt. Aus (18) und (10) entsteht direkt (61), ebenso kommt man von (23) und (62) direkt zu (69).

Bemerkung

Die Behandlung der Gleichheit erfolgt prinzipiell durch Einbeziehen einer Axiomatisierung der Gleichheit. Für das genauere Verfahren (Paramodulation) sei auf die Literatur verwiesen.

5.3.4 Übungsaufgaben

Übungsaufgabe 5.3.1
Beweisen Sie mit Resolution:

$$\begin{array}{l} (p1h1 \rightarrow \neg p2h1) \\ \wedge (p1h1 \rightarrow \neg p3h1) \\ \wedge (p2h1 \rightarrow \neg p3h1) \\ \wedge (p1h2 \rightarrow \neg p2h2) \\ \wedge (p1h2 \rightarrow \neg p3h2) \\ \wedge (p2h2 \rightarrow \neg p3h2) \end{array} \vdash \begin{array}{l} (\neg p1h1 \wedge \neg p1h2) \\ \vee (\neg p2h1 \wedge \neg p2h2) \\ \vee (\neg p3h1 \wedge \neg p3h2) \end{array}$$

Diese Aufgabe ist die aussagenlogische Formulierung eines kombinatorischen Prinzips, des sog. Dirichletschen Schubfachprinzip. Es thematisiert die offensichtliche Tatsache, daß man $n+1$ verschiedene Dinge nur dann in n Schubladen unterbringen kann, wenn in mindestens einer Schublade mindestens zwei Dinge untergebracht sind. Im Englischen redet man von $n+1$ Tauben und n Taubenschlägen und nennt das ganze ¸*pigeon hole* principle. Dieses Prinzip ist für den Fall $n = 2$ so in aussagenlogische Form umgesetzt, daß ein Atom $p1h1$ wahr ist, wenn die 1. Taube im 1.Taubenschlag untergebracht ist, und analog für die anderen Atome $pihj$. Die Voraussetzung besagt dann, daß keine zwei Tauben im selben Taubenschlag untergebracht sind und die zu beweisende Disjunktion besagt, daß dann eine Taube nicht untergebracht ist.

Übungsaufgabe 5.3.2
Man könnte versucht sein, zur Verkürzung von Beweisen im Resolutionskalkül zwei Resolutionsanwendungen in einer neuen Regel zusammenzufassen:

$$\frac{C_1 \cup \{P, Q\}, C_2 \cup \{\neg P, \neg Q\}}{C_1 \cup C_2}$$

Zeigen Sie, daß diese Regel nicht korrekt ist.

Übungsaufgabe 5.3.3
Eine *Krom-Formel* ist eine aussagenlogische Formel in KNF, in der jede Disjunktion höchstens zwei Literale enthält.

Beweisen Sie, daß Erfüllbarkeit für Krom-Formeln in polynomialer Zeit entscheidbar ist.

Übungsaufgabe 5.3.4

Definition 5.27 (Geordnete Resolution)
Sei $< \subseteq At \times At$ eine beliebige Ordnungsrelation auf der Menge At, siehe Definition 1.2 auf Seite 1

Die Ordnungsrelation wird fortgesetzt auf die Menge aller Literale, indem die Negationszeichen schlichtweg ignoriert werden, d.h. für zwei Literale $L_1 \in \{A_1, \neg A_1\}$, $L_2 \in \{A_2, \neg A_2\}$ gilt

$$L_1 < L_2 \quad gdw \quad A_1 < A_2$$

.

Ein Literal L ist $<$-maximal in einer Klausel C gdw. es kein $L' \in C$ gibt mit $L < L'$.

Eine Anwendung der Resolutionsregel

$$\frac{C_1 \cup \{P\}, C_2 \cup \{\neg P\}}{C_1 \cup C_2}$$

heißt ein geordneter Resolutionsschritt bezüglich $<$, wenn P maximal in $C_1 \cup \{P\}$ und $\neg P$ maximal in $C_2 \cup \{\neg P\}$ ist.

Beweisen oder widerlegen Sie die Vollständigkeit einer Variante des Resolutionskalküls, bei der nur geordnete Resolutionsschritte erlaubt sind.

Hinweis: Beachten Sie den Beweis des Vollständigkeitssatzes 5.14 von Seite 162.

Übungsaufgabe 5.3.5

In dieser Aufgabe wollen wir eine Variante des Vollständigkeitsbeweises des Resolutionskalküls, siehe Satz 5.14, untersuchen.

Wir ändern die induktive Definition der Belegung I folgendermaßen ab:

$$I(P_n) = W,$$

genau dann, wenn gilt:

> M_0 enthält keine Klausel $C = C_1 \cup \{\neg P_n\}$, so daß in C_1 nur negative Literale $\neg P_i$ mit $i < n$ auftreten, und $val_I(C_1) = F$.

Zeigen Sie, daß auch für diese Belegung I für alle Klauseln $C \in M_0$ gilt $val_I(C) = W$.

Übungsaufgabe 5.3.6

Wir nennen einen Resolutionsschritt

$$\frac{C_1 \cup \{P\}, C_2 \cup \{\neg P\}}{C_1 \cup C_2}$$

einen *negativen Resolutionschritt* wenn die Klausel $C_2 \cup \{\neg P\}$ nur negative Literale enthält.

Beweisen oder widerlegen Sie die Vollständigkeit einer Variante des Resolutionskalküls, bei der nur negative Resolutionsschritte erlaubt sind.

Übungsaufgabe 5.3.7

Was würde sich ändern wenn man in Definition 5.18 nicht verlangen würde, daß $C_1 \cup K_1$ und $C_2 \cup K_2$ variablendisjunkt sind? Vollständigkeit, Korrektheit, keins von beiden?

Übungsaufgabe 5.3.8

In Fortsetzung der vorangegangen Übungsaufgabe 5.3.7 betrachtet wir - nur für diese Übungsaufgabe - die folgende geänderte Version der Definition von $Res(M)$ aus Definition 5.19:

$$\begin{array}{rl} Res'(M) = & \{B \mid \text{ es gibt Klauseln } C_1, C_2 \text{ aus } M, \\ & \text{ so daß } B \text{ eine Resolvente von } C_1, C_2 \text{ ist.}\} \end{array}$$

Gegenüber der offiziellen Definition ist die Variantenbildung, d.h. die Umbennung der Variablen in C_1, C_2 weggefallen.
Wie wird dadurch Korrektheit und Vollständigkeit des Kalküls beeinflußt?

5.4 Aussagenlogische Tableaukalkül

Vorbemerkung

Auch der im folgenden vorgestellte Tableaukalkül ist ein Widerlegungskalkül, und er ist korrekt und vollständig:

$$M \models A \quad \Leftrightarrow \quad M \cup \{\neg A\} \vdash_{\mathbf{T0}} \mathbf{0}.$$

Er leistet aber noch mehr: wenn $M \not\models A$ und M endlich, dann wird eine Interpretation I, so daß val_I auf M wahr und auf A falsch ist, tatsächlich geliefert.
Literatur: [Smu68], [Fit90].

5.4.1 Syntax und Semantik

Wir betrachten in diesem Abschnitt aussagenlogische Formeln aus $A \in For0_\Sigma$, wie sie in Definition 2.3 eingeführt wurden. Die Operatoren sind also $\neg$, $\wedge$, $\vee$, $\rightarrow$ und $\mathbf{1}$ und $\mathbf{0}$ treten als Konstanten auf.

Definition 5.28 (Vorzeichenformeln)
Eine *Vorzeichenformel* ist eine Zeichenkette der Gestalt $0A$ oder $1A$ mit $A \in For0_\Sigma$. Dabei sind 0, 1 neue Sonderzeichen, die Vorzeichen, im Alphabet der Objektsprache.

Vorzeichen treten nur einmal am Beginn einer Formel auf, deswegen ist auch eine Verwechslung mit den Booleschen Konstanten ausgeschlossen.

Mit der Kuriosität, daß $0\mathbf{1}$,$1\mathbf{1}$, $0\mathbf{0}$,$1\mathbf{0}$ legitime Vorzeichenformeln sind wollen wir leben,.

Bemerkung

Die Spracherweiterung ist nicht wirklich notwendig, sondern dient nur der Durchsichtigkeit und Anschaulichkeit des Tableauverfahrens und bietet außerdem einen Ansatzpunkt für Verallgemeinerungen auf Tableauverfahren für nichtklassische Logiken. Das Vorzeichen und die aussagenlogische Verknüpfung führen zu einer Unterscheidung von zehn Typen von Formeln. Um spätere Beweise durch Fallunterscheidung zu vereinfachen, übernehmen wir die von R.Smullyan eingeführte **uniforme Notation**.

Definition 5.29 (Typen von Vorzeichenformeln)

Typ ϵ	(elementarer Typ):	$0P$, $1P$ für $P \in \Sigma$
Typ α	(konjunktiver Typ):	$0\neg B$, $1\neg B$, $1(B \wedge C)$, $0(B \vee C)$, $0(B \rightarrow C)$
Typ β	(disjunktiver Typ):	$0(B \wedge C)$, $1(B \vee C)$, $1(B \rightarrow C)$

Definition 5.30 (Abkömmlinge)

Zu jeder nicht elementaren Vorzeichenformel V definieren wir zwei *Abkömmlinge* V_1, V_2 (die auch identisch sein können) gemäß den folgenden Tabellen

Vom Typ α	V_1	V_2
$0\neg B$	$1B$	$1B$
$1\neg B$	$0B$	$0B$
$1(B \wedge C)$	$1B$	$1C$
$0(B \vee C)$	$0B$	$0C$
$0(B \rightarrow C)$	$1B$	$0C$

Vom Typ β	V_1	V_2
$0(B \wedge C)$	$0B$	$0C$
$1(B \vee C)$	$1B$	$1C$
$1(B \rightarrow C)$	$0B$	$1C$

Elementare Vorzeichenformeln haben keine Abkömmlinge.

Ist I eine Interpretation der Atome und val_I die zugehörige Auswertung der Formeln ohne Vorzeichen in $For0_\Sigma$. Wir setzen val_I fort auf die Menge aller Vorzeichenformeln durch

$$val_I(0A) = val_I(\neg A),$$

und

$$val_I(1A) = val_I(A).$$

Lemma 5.31
Für eine Vorzeichenformel V gilt

vom Typ α:	$val_I(V) = W$	$\Leftrightarrow$	$val_I(V_1) = W$	und	$val_I(V_2) = W$
vom Typ β:	$val_I(V) = W$	$\Leftrightarrow$	$val_I(V_1) = W$	oder	$val_I(V_2) = W$

Hieran orientiert sich nun das nachfolgende Tableauverfahren.

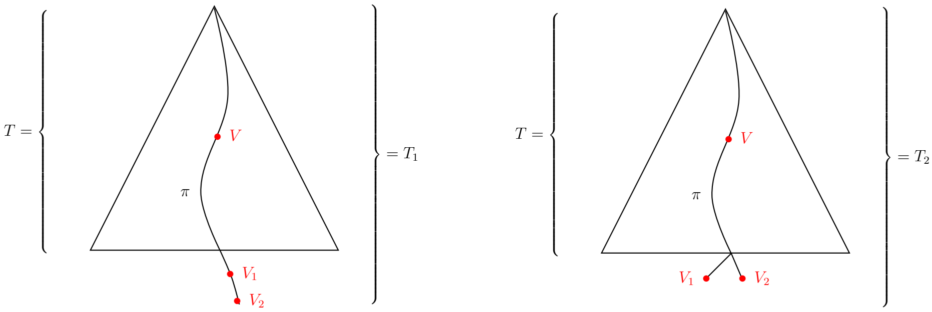
5.4.2 Kalkül

Definition 5.32 (Pfade)
In der nachfolgenden Definition betrachten wir Bäume, deren Knoten durch Vorzeichenformeln beschriftet sind. Ein **Pfad** in einem Baum ist eine maximale, linear geordnete Menge von Knoten. Wollen wir die Maximalität nicht verlangen, so sprechen wir von einem **Teilpfad**.

Bevor wir die formale Definition des Tableaukalküls geben, betrachten wir Abb. 5.2 als Einstiegsbeispiel. Es soll bewiesen werden, daß $(P \vee (Q \wedge R)) \rightarrow ((P \vee Q) \wedge (P \vee R))$ eine Tautologie ist. Der Tableaukalkül ist ein Widerspruchskalkül. Wir nehmen also an, daß $(P \vee (Q \wedge R)) \rightarrow ((P \vee Q) \wedge (P \vee R))$ keine Tautologie ist, also bei geeigneter Interpretation falsch werden kann. Diese Annahme deuten wir durch das Voranstellen der 0 an. Die betrachtete Implikation kann offensichtlich nur falsch sein, wenn die Prämisse, $P \vee (Q \wedge R)$, wahr und die Konklusion, $(P \vee Q) \wedge (P \vee R)$, falsch ist. Das notieren wir, indem wir die beiden Formeln unter die Anfangsannahme schreiben, mit dem Vorzeichen 1, bzw. 0, versehen. Die Formel $P \vee (Q \wedge R)$ ist wahr, wenn P oder $Q \wedge R$ wahr ist. Diese Alternative deuten wir im Tableau durch eine Verzweigung an, der linke Pfad führt zur Formel $1P$, der rechte zu $1(Q \wedge R)$. Wir führen zuerst die Argumentationskette auf dem linken Pfad weiter. Unter den Folgerungen, zu denen uns die Anfangsannahme bisher geführt hat, ist auch die Aussage, daß $(P \vee Q) \wedge (P \vee R)$ falsch ist. Eine Konjunktion ist falsch, wenn eines ihrer Konjunktionglieder falsch ist. Das führt also zu einer weiteren Verzweigung des linken Pfades mit den in der Abbildung angegebenen Markierungen. Die bisherigen Überlegungen haben auf dem linkesten Pfad zur Annahme geführt, daß die Formel $P \vee Q$ falsch ist, das bedeutet, daß P und Q falsch sein müssen. Alle Formeln auf diesem Ast sind jetzt abgearbeitet worden und auf Atome mit Vorzeichen reduziert worden. An dieser Stelle erinnern wir uns, daß unser Ansatz ein Widerspruchbeweis ist. Wir erwarten, im positiven Fall, daß unsere Anfangsannahme zu Widersprüchen führt. Für den Pfad, den wir gerade betrachten, ist das schon der Fall. Er enthält die Formeln $1P$ und $0P$, die besagen, daß P wahr ist und P auch falsch ist. In entsprechender Weise müssen noch die übrigen offenen Alternativen in dem Tableau betrachtet werden. In dem Beispiel in Abbildung 5.2 erweisen sich alle Pfade als geschlossen. Damit ist die Anfangsannahme zu einem Widerspruch geführt worden.
Es ist wichtig zu bemerken, daß bei der Verlängerung eines Pfades nicht notwendigerweise nur die letzte Formel auf dem Pfad betrachtet wird.

Definition 5.33 (Tableaux)
Es sei M eine Menge von Formeln und A eine Formel. Ein (aussagenlogisches) **Tableau** für A über M ist ein Baum mit Beschriftung der Knoten durch markierte Formeln, gemäß der folgenden rekursiven Definition:

1. Ein einziger mit $0A$ markierter Knoten ist ein Tableau für A über M.
2. Es sei T ein Tableau für A über M und π ein Pfad in T, so daß auf π ein Knoten liegt, der mit einem V vom Typ α beschriftet ist. Dann ist auch T_1 ein Tableau für A über M, welches dadurch aus T entsteht,



(a) α – *Erweiterung* (b) β – *Erweiterung*

Abbildung 5.1: Tableau-Erweiterungen

daß an π zwei neue Knoten angefügt werden, welche mit V_1 und mit V_2 beschriftet sind, siehe Abbildung 5.1 (linke Seite). Wir sagen in diesem Fall, daß T_1 aus T durch eine Anwendung der **α-Regel** aus T hervorgeht.

3. Es sei T ein Tableau für A über M und π ein Pfad in T, so daß auf π ein Knoten liegt, der mit einem V vom Typ β beschriftet ist. Dann ist auch T_2 ein Tableau für A über M, welches dadurch entsteht, daß an π zwei verzweigende Kanten angefügt werden, welche zu neuen Blättern mit Beschriftungen V_1 bzw. V_2 führen, siehe Abbildung 5.1 (rechte Seite). Wir sagen, T_2 geht aus T durch eine Anwendung der **β-Regel** aus T hervor.

4. Es sei T ein Tableau für A über M und π ein Pfad in T, dann ist auch T_3 ein Tableau für A über M, welches durch Verlängerung von π um einen mit $1B$ beschrifteten Knoten entsteht, für eine Formel B aus M. Wir sagen, T_3 geht aus T durch eine Anwendung der **Voraussetzungsregel** aus T hervor.

Der Verzweigungsgrad eines Tableaubaumes (die maximale Anzahl der Nachfolger eines Knotens) hängt von der benutzten Regeln ab. Die Regeln, die wir bisher kennen gelernt haben, führen zu Tableaubäumen mit maximalem Verzweigungsgrad 2. Wir werden in Unterabschnitt 5.4.3ein allgemeineres Regelformat betrachten.

Definition 5.34 (Geschlossenes Tableau)
Ein Tableau T heißt **geschlossen**, wenn alle seine Pfade geschlossen sind. Ein Pfad π heißt **geschlossen**, wenn zwei Knoten auf π liegen, welche respektive beschriftet sind mit $0A$ und $1A$, für eine beliebige Formel A. Auch ein Pfad,

auf dem die Formel 0**1** oder 1**0** vorkommt zählt als geschlossen. Gelegentlich deuten wir die Abgeschlossenheit eines Pfades π an durch Anfügen einer Kante an π, welche zu einem mit $*$ beschrifteten Blatt führt.

Wir verwenden ggf. die folgende abkürzende Notation ("uniforme Notation"). Schreibt man kurz α für eine α-Formel, so heißen ihre Abkömmlinge α_1, α_2. Entsprechend für β. Hiermit definieren wir

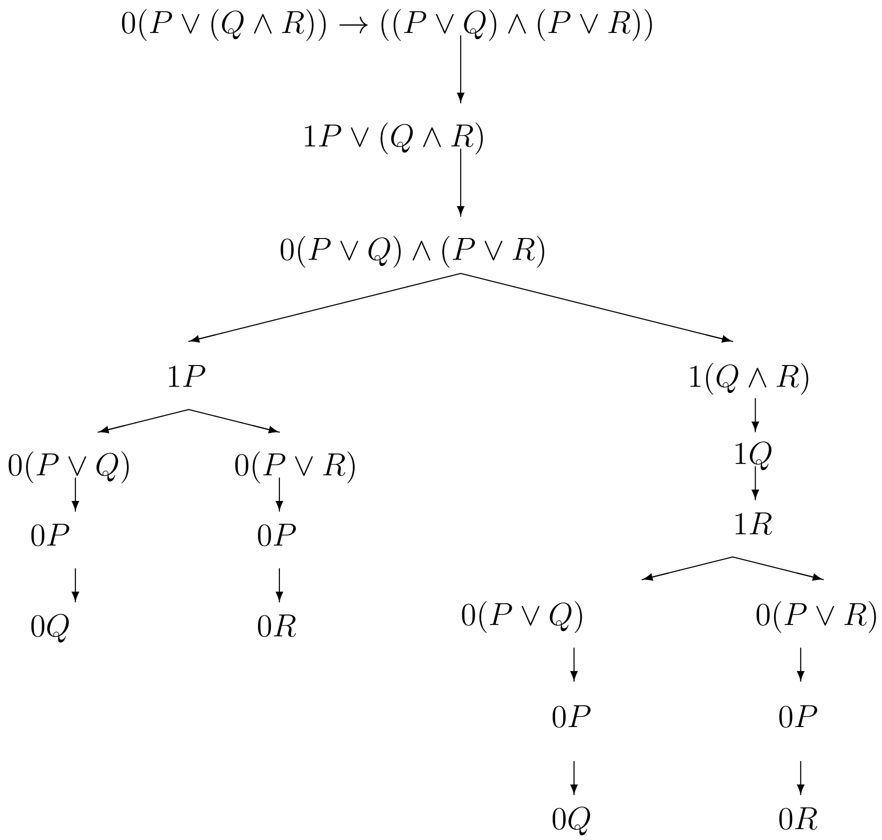


Abbildung 5.2: Beispiel eines Tableau-Beweises

Wir benutzen die folgende Kurzschreibweise für Tableauregeln

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2}$$

Definition 5.35 (Ableitbarkeit)
Es sei $A \in For0_\Sigma$ und $M \subseteq For0_\Sigma$. A ist aus M *ableitbar in* $\mathbf{T}0$,

$$M \vdash_{\mathbf{T}0} A,$$

g.d.w. es ein geschlossenes Tableau für $0A$ über M gibt.

Man beachte, daß zur Definition der Ableitbarkeit einer Formel aus einer Formelmenge hinzugenommen sind:

> die *Initialisierung:* Erzeuge einen mit $0A$ beschrifteten Knoten;
> die *V_M-Regel* (Voraussetzungsregel bzg. M):
> Vergrößere ein Tableau durch Anhängen einer Kante von einem Blatt zu einem neuen, mit $1B$ für ein $B \in M$ beschrifteten Knoten.

Bemerkung 5.36
(Beziehung zum Begriff des abstrakten Kalküls)
Muss noch ergänzt werden

Definition 5.37 (Semantik von Tableaux)
Für eine Interpretation I der Atome haben wir bereits oben (5.30, 5.31) die Auswertung der Vorzeichenformeln definiert. Ist T ein Tableau und π ein Pfad in T, so setzen wir

$$\begin{array}{lll} val_I(\pi) = W & :\Leftrightarrow & val_I(V) = W \text{ für alle } V \text{ auf } \pi \\ val_I(T) = W & :\Leftrightarrow & \text{Es gibt einen Pfad } \pi \text{ in } T \text{ mit } val_I(\pi) = W \end{array}$$

Eine Interpretation I mit $val_I(T) = W$ heißt **Modell** von T.

Satz 5.38 (Korrektheit und Vollständigkeit)
Es seien $A \in For0_\Sigma$ und $M \subseteq For0_\Sigma$. Dann gilt

$$M \vdash_{\mathbf{T0}} A \quad \Leftrightarrow \quad M \models A.$$

Beweis der Korrektheit:
Aus der Semantik der Tableau folgt, daß ein geschlossenes Tableau kein Modell haben kann (Definitionen 5.34, 5.37). Wenn nun $M \not\models A$, d.h. $M \cup \{\neg A\}$ hat ein Modell, dann zeigen wir:

- Jedes Tableau für A über M hat ein Modell. Somit kann kein solches Tableau geschlossen sein.
- folgt aus dem

Lemma 5.39 (Korrektheitslemma)
Ist I Modell von $M \cup \{\neg A\}$, dann ist I Modell eines jeden Tableau für A über M.

Beweis:
Initialisierung: I ist Modell von $\neg A$, also von $0A$.
Im Induktionsschritt unterscheiden wir nach dem Typ der auf ein Tableau T angewandten Regel.

α-Fall:
Nach Induktionsannahme gilt $val_I(T) = W$, es gibt also einen Pfad π in T mit $val_I(\pi) = W$. Zur Anwendung der α-Regel wird ein Pfad π_1 in T und eine α-Formel V auf π_1 gewählt, π_1 wird verlängert um V_1, V_2. Wenn $\pi_1 \neq \pi$, gilt für das neu entstehende Tableau T' trivialerweise $val_I(T') = W$. Wenn $\pi_1 = \pi$, haben wir aus $val_I(\pi) = W$, daß $val_I(V) = W$, also $val_I(V_1) = W$ und $val_I(V_2) = W$ nach Lemma 5.31. Somit $val_I(\pi') = W$ für den neu entstehenden Pfad π', d.h. $val_I(T') = W$

β-Fall:
Entsprechend mit „oder“ statt „und“. (Es entstehen zwei Pfade, von denen mindestens einer I zum Modell haben muß.

V_M-Fall:
Nach Voraussetzung ist I Modell von M, also von jedem $1B$, $B \in M$.

∎

Beweis der Vollständigkeit
Wir zeigen etwas mehr. Definieren wir zunächst *erschöpfte* Tableaux als solche, die sich nur „auf trivial wiederholende Weise“ noch vergrößern lassen:

Definition 5.40 (Erschöpftes Tableau)
Das Tableau T für A über der Formelmenge M heißt **erschöpft**, wenn auf jedem offenen Pfad π von T jede α- und β-Formel benutzt wurde (d.h. die entsprechende Regel auf sie angewandt wurde), und jedes $1B$, $B \in M$, auf π vorkommt.

Wenn M unendlich ist, ist auch jedes erschöpfte Tableau über M unendlich. Offensichtlich gibt es mindestens ein erschöpftes Tableaux für A über M, man muß nur in systematischer Weise auf jedem Pfad jede mögliche Regelanwendung vornehmen. Ausgehend von A und M gibt es nicht nur ein erschöpftes Tableau, sondern abhängig von der Reihenfolge der Regelanwendungen mehrere. Durch sorgfältige Buchführung könnte man sich ein Verfahren ausdenken, das alle Variationsmöglichkeiten durchprobiert. Wie die nachfolgenden Ergebnisse zeigen, ist das jedoch nicht nötig. Wir zeigen (Lemma 5.41), daß für jedes erschöpfte Tableau T über M gilt: Hat T einen offenen Pfad, so läßt sich aus ihm ein Modell von $M \cup \{\neg A\}$ ablesen. Insbesondere hat man dann:

$$\begin{array}{rcl} M \not\vdash_{\mathbf{T0}} A & \Leftrightarrow & \text{kein Tableau für } A \text{ über } M \text{ ist geschlossen} \\ & \Rightarrow & \text{kein erschöpftes Tableau für } A \text{ über } M \text{ ist geschlossen} \\ & \Rightarrow & M \cup \{\neg A\} \text{ hat ein Modell, d.h. } M \not\models A. \end{array}$$

Aus der Korrektheit von **T**0 folgt auch noch: Entweder alle erschöpften Tableaux für A über M sind geschlossen, oder keins.

Lemma 5.41 (Vollständigkeitslemma)
Wenn es ein erschöpftes, nicht geschlossenes Tableau für A über M gibt, dann hat $M \cup \{\neg A\}$ ein Modell.

Beweis:
Sei π ein offener Pfad im erschöpften Tableau T für A über M. Wir schreiben $V \in \pi$, wenn V auf π liegt. Da T erschöpft ist, hat man

- Für jede α-Formel $V \in \pi$: $V_1 \in \pi$ und $V_2 \in \pi$
- für jede β-Formel $V \in \pi$: $V_1 \in \pi$ oder $V_2 \in \pi$
- Für jedes $B \in M$: $1B \in \pi$.

Da π nicht abgeschlossen ist liegen für jedes in A oder einem $B \in M$ vorkommenden Atom P nicht beide $0P$ und $1P$ auf π. Wir definieren

$$I(P) := \begin{cases} W & \text{falls } 1P \in \pi \\ F & \text{falls } 0P \in \pi \\ & \text{beliebig sonst} \end{cases}$$

Durch Induktion zeigt man leicht, daß $I(V) = W$ für jede Formel V auf π. Wir zeigen das für den Fall, daß $V = V_1 \wedge V_2$. Da π erschöpft ist, folgt aus $V \in \pi$ auch $V_1 \in \pi$ und $V_2 \in \pi$. Nach Induktionsvoraussetzung folgt $I(V_1) = W$ und $I(V_2) = W$, woraus nach der Wahrheitstabelle für die Konjunktion $I(V_1 \wedge V_2) = W$ folgt.

Es folgt, daß I Modell von $M \cup \{\neg A\}$ ist.

∎

5.4.3 Generierung von Tableauregeln

Woher kommen die Tableauregeln? Für die bisher betrachteten einfachen logischen Verknüpfungen waren die zugehörigen Regeln zu naheliegend, als daß man dieser Frage große Aufmerksamkeit geschenkt hätte. Die beiden Regeln für die Konjunktion

$$\begin{array}{c} 1(A \land B) \\ \hline 1A \\ 1B \end{array} \qquad \begin{array}{c} 0(A \land B) \\ \hline 0A \;\mid\; 0B \end{array}$$

bieten sich direkt an. Wie müsste aber eine Tableauregel für die Formel $1(A \leftrightarrow B)$ aussehen? oder für den Shannon-Operator? (siehe Definition 2.44). Bevor wir diese Frage beantworten, wollen wir das Format für Tableauregeln verallgemeinern. Statt einer Verzweigung in zwei Fortsetzungen wollen wir beliebige endliche Verzweigungen zulassen. Auch soll pro Fortsetzung erlaubt sein endliche viele Formeln anzugeben. In den bisherigen Beispielen wurden höchstens zwei Formeln pro Fortsetzung benötigt. Eine allgemeine Tableauregel hat also die Form

$$\begin{array}{c|c|c} \multicolumn{3}{c}{\phi} \\ \hline \psi_{1,1} & \dots & \psi_{n,1} \\ \vdots & \dots & \vdots \\ \vdots & \dots & \vdots \\ \psi_{1,K_1} & \dots & \psi_{n,k_n} \end{array}$$

Um die Teilformeleigenschaft des Tableaukalküls zu gewährleisten, wird gefordert, daß alle Vorzeichenformeln $\psi_{i,j}$ Teilformeln der Vorzeichenformel ϕ sind.

Eine genaue Inspektion des Vollständigkeits- und Korrektheitsbeweises für den Tableaukalkül zeigt, daß allein die Aussagen von Lemma 5.31 notwendig waren. Die Verallgemeinerung der Aussage dieses Lemmas führt uns zu der Definition

Definition 5.42

Eine allgemeine Tableauregel

$$\begin{array}{c|c|c} \multicolumn{3}{c}{\phi} \\ \hline \psi_{1,1} & \dots & \psi_{n,1} \\ \vdots & \dots & \vdots \\ \vdots & \dots & \vdots \\ \psi_{1,k_1} & \dots & \psi_{n,k_n} \end{array}$$

heißt vollständig und korrekt, wenn für jede Interpretation I gilt

$$\begin{array}{lll} val_I(\phi) = W & \text{gdw} & \text{es gibt ein } i,\ 1 \leq i \leq n, \text{ so daß} \\ & & \text{für alle } j, 1 \leq j \leq k_i \text{ gilt} \\ & & val_I(\psi_{i,j}) = W \end{array}$$

Satz 5.43

Ein Tableaukalkül der nur korrekte und vollständige Regeln benutzt ist selbst korrekt und vollständig.

Beweis siehe Literatur.

■

Beispiel 5.44

Wir schauen uns noch einmal die Wahrheitstabelle für den logischen Äquivalenzoperator an.

$\leftrightarrow$	1	0
1	1	0
0	0	1

Man prüft leicht nach, daß die Tableauregel

$$\frac{1(A \leftrightarrow B)}{\begin{array}{l|l} 1A & 0A \\ 1B & 0B \end{array}}$$

die Forderungen der Definition 5.42 erfüllt.

5.4.4 Übungsaufgaben

Übungsaufgabe 5.4.1
In Anlehnung an [DGHP99, pp.70] definieren wir

Definition 5.45
Ein Tableau T heißt *regulär*, wenn keine Vorzeichenformel auf demselben Pfad zweimal vorkommt.

Das Tableau aus Abb.5.2 ist ein Beispiel eines regulären Tableaux.

1. Geben Sie Formeln A,B an, so daß das es ein geschlossenes Tableau für A über B gibt, aber keine reguläres geschlossenes.
2. Wie könnte man das Tableauverfahren aus Definition 5.33 abändern, so daß nur reguläre Tableaux enstehen und Korrektheit und Vollständigkeit erhalten bleiben?

Übungsaufgabe 5.4.2
Geben Sie vollständige und korrekte Tableauregeln an für

1. die Äquivalenz $\leftrightarrow$,
2. den dreistelligen Shannon-Operator sh.

Übungsaufgabe 5.4.3
Beweisen Sie im Tableau-Kalkül:

$$\begin{array}{l} \,(p1h1 \rightarrow \neg p2h1) \\ \wedge\,(p1h1 \rightarrow \neg p3h1) \\ \wedge\,(p2h1 \rightarrow \neg p3h1) \\ \wedge\,(p1h2 \rightarrow \neg p2h2) \\ \wedge\,(p1h2 \rightarrow \neg p3h2) \\ \wedge\,(p2h2 \rightarrow \neg p3h2) \end{array} \quad \vdash \quad \begin{array}{l} \,(\neg p1h1 \wedge \neg p1h2) \\ \vee\,(\neg p2h1 \wedge \neg p2h2) \\ \vee\,(\neg p3h1 \wedge \neg p3h2) \end{array}$$

Übungsaufgabe 5.4.4
Beweisen Sie im Tableau-Kalkül:

$$((A \wedge B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$$

Übungsaufgabe 5.4.5
Beweisen Sie den β-Fall von Lemma 5.39.

Übungsaufgabe 5.4.6
Anstelle der β-Regel werde die folgende β^+-Regel benutzt:

$$\frac{\beta}{\begin{array}{c|c} \beta_1 & \neg\beta_1 \\ & \beta_2 \end{array}}$$

Zeigen Sie, daß der so entstehende Tableaukalkül auch korrekt und vollständig ist.

Übungsaufgabe 5.4.7
Im Kapitel *Tableau Methods for Classical Propositional Calculus* in [DGHP99] von Marcello D'Agostino wird ein **KE** genannter Tableaukalkül vorgestellt. Die α-Regeln sind dieselben wie in Abschnitt 5.4.2 beschrieben. Die uniforme β-Regel sieht dagegen so aus:

$$\frac{\begin{array}{c}\beta_1^c \\ \beta\end{array}}{\beta_2} \qquad \frac{\begin{array}{c}\beta_2^c \\ \beta\end{array}}{\beta_1}$$

wobei V^c die zu V komplementäre Vorzeichenformel ist, d.h. $V^c = 0F$ für $V = 1F$ und $V^c = 1F$ für $V = 0F$. In unserem Kalkül aus Abschnitt 5.4.2 besaßen die Regeln nur eine Prämisse, mit Ausnahme der Pfadabschlussregel. Im **KE** Kalkül besitzen alle β-Regeln zwei Voraussetzungen. Die generische β-Regel umfasst die folgenden Instanzen.

$$\frac{\begin{array}{c}0F_1\\1(F_1 \vee F_2)\end{array}}{1F_2} \qquad \frac{\begin{array}{c}0F_2\\1(F_1 \vee F_2)\end{array}}{1F_1} \qquad \frac{\begin{array}{c}1F_1\\1(F_1 \to F_2)\end{array}}{1F_2} \qquad \frac{\begin{array}{c}0F_2\\1(F_1 \to F_2)\end{array}}{0F_1}$$

$$\frac{\begin{array}{c}1F_1\\0(F_1 \wedge F_2)\end{array}}{0F_2} \qquad \frac{\begin{array}{c}1F_2\\0(F_1 \wedge F_2)\end{array}}{0F_1}$$

Die bisher präsentierten Regeln sind nicht vollständig, sie würden auch nur die Konstruktion von Tableaux mit einem einzigen Pfad erlauben. Hinzu kommt noch die eingeschränkte Schnittregel

$$\frac{}{1F \mid 0F}$$

Die Einschränkung besteht darin, daß die Schnittregel zu Verzweigung eines Tableaupfades π nur angewendet werden darf wenn es eine β-Formel β auf π gibt, so daß $\beta_1 = 1F$ oder $\beta_1 = 0F$. Die Pfadabschlussregel bleibt wie bisher.

Beweisen Sie

1. Korrektheit und
2. Vollständigkeit des **KE** Kalküls

5.5 Prädikatenlogischer Tableaukalkül (ohne Gleichheit)

Wir beschränken uns aus didaktischen Gründen auf einen Tableaukalkül ohne Gleicheit

Eine ausführliche Darstellung des Tableaukalküls findet man in den Büchern [Smu68] und [Fit90]. Abgesehen von der Orientierung zum automatischen Beweisen, enthält das Buch von Fitting gegenüber dem von Smullyan eine neue Behandlung des Allquantors, der es ermöglicht auch im Tableaukalkül die Technik der Unifikation zu nutzen. Eine umfassende Darstellung der Tableau Methode wird in dem Handbuch [DGHP99] gegeben. Wir folgen der Darstellung in [Fit90] mit den Verbesserungen aus [HS94].

Wir arbeiten über der Basis $\{\neg, \wedge, \vee, \rightarrow, \forall, \exists\}$. Eine Transformation der am Beweis beteiligten Formeln in eine Normalform ist nicht erforderlich.

5.5.1 Tableauregeln

Prädikatenlogische Vorzeichenformeln sind prädikatenlogischen Formeln, vor denen 0 oder 1 als Vorzeichen stehen, wie in 5.4. Vorzeichenformeln werden wieder in Typen eingeteilt, und entsprechend dieser Einteilung werden Abkömmlinge definiert:

Uniforme Notation

Typ ϵ: $1A, 0A$ für atomare Formeln A

Typ α:

V	V_1	V_2
$1\neg A$	$0A$	–
$0\neg A$	$1A$	–
$1A \wedge B$	$1A$	$1B$
$0A \vee B$	$0A$	$0B$
$0A \rightarrow B$	$1A$	$0B$

Typ β:

V	V_1	V_2
$0A \wedge B$	$0A$	$0B$
$1A \vee B$	$1A$	$1B$
$1A \rightarrow B$	$0A$	$1B$

Typ γ:

V	V_1
$1\forall xA(x)$	$1A(x)$
$0\exists xA(x)$	$0A(x)$

Typ δ:

V	V_1
$1\exists xA(x)$	$1A(x)$
$0\forall xA(x)$	$0A(x)$

Durch die uniforme Notation wird in den nachfolgenden Beweisen eine Fallunterscheidung in elf Fälle auf vier Fälle reduziert. Das ist einzig und allein der Grund für ihre Einführung. Wesentlich dafür sind die folgenden Gemeinsamkeiten von Formeln desselben Typs:

Lemma 5.46
Sei (D, I) eine beliebige Interpretation, β eine beliebige Variablenbelegung für (D, I). Wir schreiben val_β statt $val_{D,I,\beta}$. Dann gilt

1. für jede α-Formel V:
$$val_\beta(V) = W \quad \Leftrightarrow \quad val_\beta(V_1) = W \text{ und } val_\beta(V_2) = W$$

2. für jede β-Formel V:
$$val_\beta(V) = W \quad \Leftrightarrow \quad val_\beta(V_1) = W \text{ oder } val_\beta(V_2) = W$$

3. für jede γ-Formel V:
$$val_\beta(V) = W \quad \Leftrightarrow \quad \text{Für alle } d \in D : val_{\beta_x^d}(V_1(x)) = W$$

4. für jede δ-Formel V:
$$val_\beta(V) = W \quad \Leftrightarrow \quad \text{Es gibt } d \in D \text{ mit: } val_{\beta_x^d}(V_1(x)) = W.$$

(Hierbei ist $V_1(x)$ der Rumpf und x die quantifizierte Variable von V.)

Definition 5.47 (Tableaux)
Sei A eine Formel und M eine Menge von Formeln, alle ohne freie Variablen. Ein **Tableau für A über M** ist wie folgt rekursiv definiert.

1. Ein einziger mit $0A$ markierter Knoten ist ein Tableau für A über M.

2. α-Regel (wie im AL-Fall).

3. β-Regel (wie im AL-Fall).

4. Es sei T ein Tableau für A über M und π ein Pfad in T, so daß auf π ein Knoten liegt, der mit einem V vom Typ γ beschriftet ist, $V = 1\forall x B_1(x)$ oder $V = 0\exists x B_1(x)$. Dann ist auch T' ein Tableau für A über M, welches dadurch aus T entsteht, daß an π ein neuer Knoten angefügt wird, welcher mit $V_1(y)$ (d.h. $1B_1(y)$ bzw. $0B_1(y)$) beschriftet ist für eine Variable y, die noch nicht auf π frei vorkommt. Wir sagen in diesem Fall, daß T' aus T durch eine Anwendung der **γ-Regel** hervorgeht.

5. Es sei T ein Tableau für A über M und π ein Pfad in T, so daß auf π ein Knoten liegt, der mit einem V vom Typ δ beschriftet ist, $V = 0\forall x B_1(x)$ oder $V = 1\exists x B_1(x)$. Dann ist auch T' ein Tableau für A über M, welches dadurch aus T entsteht, daß an π ein neuer Knoten angefügt wird, welcher mit $V_1(f(x_1, \ldots, x_n))$ beschriftet ist, wobei $x_1, \ldots, x_n$ alle freien Variablen in V sind und f ein neues Funktionszeichen ist. Wir sagen, T' geht aus T durch eine Anwendung der **δ-Regel** hervor.

6. Es sei T ein Tableau für A über M und π ein Pfad in T, dann ist auch T' ein Tableau für A über M, welches durch Verlängerung von π um einen mit $1B$ beschrifteten Knoten entsteht, für eine Formel B aus M. Wir sagen, T' geht aus T durch eine Anwendung der **V-Regel** aus T hervor.

Diese Definition ist eine Fortsetzung der entsprechenden aussagenlogischen Definition 5.33 auf Seite 180.

Zusammenfassung der Tableauregeln in Kurzform

α-Regel	$\dfrac{V}{\begin{array}{c}V_1\\V_2\end{array}}$	für α-Formeln V
β-Regel	$\dfrac{V}{V_1 \mid V_2}$	für β-Formeln V
γ-Regel	$\dfrac{V}{V_1(y)}$	für γ-Formeln V und jede Variable y, die auf dem Pfad noch nicht vorkommt
δ-Regel	$\dfrac{V}{V_1(f(x_1, \ldots, x_n))}$	für δ-Formeln V, wobei $x_1, \ldots, x_n$ alle freien Variablen in V sind und f ein neues n-stelliges Funktionssymbol
Anfangsregel	$\dfrac{}{0A}$	für die zu beweisende Formel A A ohne freie Variable
V-Regel	$\dfrac{}{1B}$	für jedes $B \in M$ B ohne freie Variable

Definition 5.48
Es sei π ein Pfad in T und σ eine Substitution. σ *schließt* π, wenn es

- Formeln B, C gibt, so daß $\sigma(B) = \sigma(C)$, σ kollisionsfrei für B und C ist und $1B, 0C$ auf π liegen oder
- eine der Formeln $0\mathbf{1}$ oder $1\mathbf{0}$ liegt auf π.

Definition 5.49 (Abschlußregel)
C-Regel:

Aus einem Tableau T erzeuge ein Tableau T_1 dadurch, daß für einen Pfad π und eine kollisionsfreie Substitution σ, die π schließt, σ auf ganz T angewandt wird.

Nach Anwendung der C-Regel ist der zugehörige Pfad π geschlossen.
Ein Tableau T heißt *geschlossen*, wenn alle seine Pfade geschlossen sind.

Beispiel 5.50
Ein Tableau, das mit der C-Regel geschlossen wird:

$$
\begin{array}{ccccc}
0\ \forall x\ p(x) \to \exists y\ p(y) & & & & 0\ \forall x\ p(x) \to \exists y\ p(y) \\
| & & & & | \\
1\ \forall x\ p(x) & (\alpha\text{-Regel}) & & & 1\ \forall x\ p(x) \\
| & & \text{C-Regel} & & | \\
0\ \exists y\ p(y) & (\alpha\text{-Regel}) & \Longrightarrow & & 0\ \exists y\ p(y) \\
| & & \{Y/X\} & & | \\
1\ p(X) & (\gamma\text{-Regel}) & & & 1\ p(X) \\
| & & & & | \\
0\ p(Y) & (\gamma\text{-Regel}) & & & 0\ p(X) \\
 & & & & \star
\end{array}
$$

Aus der zu beweisenden Aussage ist durch einmalige Anwendung der α-Regel und zweimalige Anwendung der γ-Regel das linke Tableau entstanden. Aus diesem ist dann das rechts stehende durch die Anwendung der C-Regel entstanden. Zur Verdeutlichung der freien Variablen verwendet man üblicherweise Großbuchstaben für die Variablen, die bei Anwendung der γ-Regel substituiert werden.

Definition 5.51 (Der Kalkül T)
Sei A eine Formel, M eine Formelmenge, alle ohne freie Variable.

Wir sagen, A ist im Tableaukalkül aus M ableitbar, in Symbolen

$$M \vdash_{\mathbf{T}} A$$

wenn es ein geschlossenes Tableau für A über M gibt.

Der Kalkül **T** arbeitet auf Tableaux als Objekte für die Regeln. Die Regeln von **T** stellen jeweils aus einem Tableau T ein Tableau T_1 her gemäß einer der Regeln (α, β, γ, δ und C) anzuwenden auf eine oder zwei Vorzeichenformeln auf einem Pfad π in T.

Man beachte, daß zum Zwecke des Herstellens eines geschlossenen Tableaus für A über M zu den Regeln von **T** die *Anfangsregel* und die *V-Regel* hinzuzunehmen sind.

In dieser Terminologie ist eine *Ableitung* in **T** eine Folge von Tableaux, beginnend mit einem Ein-Knoten-Tableau der Art $0A$ und jeweils fortgesetzt durch Anwenden einer der angegebenen Regeln.

Bemerkung 5.52

Um $M \vdash_{\mathbf{T}} A$ zu etablieren, versuchen wir, ein geschlossenes Tableau für A über M zu finden. Ausgehend vom Ein-Knoten-Tableau $0A$ werden durch Anwendung der Regeln von **T** und der V-Regel für M sukzessive Tableaux gebildet. Ist man bei einem solchen T angelangt, so sind Fortsetzungsmöglichkeiten

1. das Schließen eines Pfades in T durch die C-Regel, also der Übergang zu $\sigma(T)$ (in offensichtlicher Notation) und

2. das Verlängern von T mittels einer der weiteren Regeln von **T** oder der V-Regel.

In der Regel wird man zunächst versuchen, Fortsetzung 1 zu wählen. Ist dies nicht möglich, setzt man also mit 2 fort, so ist hierzu folgendes zu bedenken. Es ist nicht verboten, eine Regel auf denselben Knoten – also die dort stehende Vorzeichenformel – zweimal anzuwenden. Man erkennt jedoch, daß dies für alle Regeln *außer der γ-Regel* unnötig ist, der bearbeitete Knoten könnte im Prinzip gelöscht werden (das folgt aus Lemma 5.46). Die γ-Regel spielt eine Sonderrolle: Im Gegensatz zu allen anderen Tableauregeln muss die mehrfache Anwendung auf dieselbe Formel $1\forall x\ B(x)$ mit unterschiedlichen Substitutionen $1B(t_1),\ 1B(t_2), ...$ ermöglicht werden. Dazu müssen Formeln $1B(X_1),\ 1B(X_2), ...$ als Ausgangspunkt für diese Substitutionen bereitgehalten werden. Um keine Möglichkeit auszulassen, sollte eine Strategie zur Wahl der Formel, mit deren Bearbeitung fortgesetzt wird, *fair* sein: Solange ein Pfad nicht geschlossen ist, wird versucht, jede α, β, δ-Formel auf ihm genau einmal zu benutzen, jede γ-Formel "immer wieder" (unbeschränkt oft), mit immer neuen eingesetzten Variablen.

1[]	$0\exists y\forall x p(x,y) \to \forall x\exists y p(x,y)$
2[1]	$1\exists y\forall x p(x,y)$
3[1]	$0\forall x\exists y p(x,y)$
4[2]	$1\forall x p(x,a)$
5[3]	$0\exists y p(b,y)$
6[4]	$1p(X,a)$
7[5]	$0p(b,Y)$
	geschlossen mit $\sigma(X) = b$ und $\sigma(Y) = a$

Tabelle 5.1: Ein geschlossenes Tableau

1[]	$0\forall x\exists y p(x,y) \to \exists y\forall x p(x,y)$
2[1]	$0\exists y\forall x p(x,y)$
3[1]	$1\forall x\exists y p(x,y)$
4[2]	$0\forall x p(x,Y)$
5[3]	$1\exists y p(X,y)$
6[4]	$0p(f(Y),Y)$
7[5]	$1p(X,g(X))$
	$p(f(Y),Y)$ und $p(X,g(X))$ sind nicht unifizierbar
	es müßte $\sigma(X) = f(Y)$ und $\sigma(Y) = \sigma(g(f(Y))$ gelten

Tabelle 5.2: Ein offenes Tableau

Beispiele für ein geschlossenes und ein offenes Tableau findet man in den Tabellen 5.1 und 5.2. Um die Tableaux leichter nachvollziehbar zu machen, sind die Knoten mit $n[m]$ markiert. Die erste Zahl gibt eine fortlaufende Nummerierung aller Knoten. Die Zahl m in eckigen Klammern gibt an, welcher Knoten benutzt wurde, um den Knoten n zu erzeugen.

5.5.2 Korrektheit

Wir führen den Begriff des Modells auch für Tableaux ein.

Definition 5.53
Es seien $A \in For_\Sigma$, $M \subseteq For_\Sigma$, T ein Tableau für A über M und (D, I) eine Interpretation über $\overline{\Sigma}$, wobei $\overline{\Sigma} = \Sigma \cup \{f \mid f \text{ neues Funktionssymbol in } T\}$.
(D, I) heißt **Modell von T über M** gdw. gilt

- (D, I) ist Modell von M
- zu jeder Variablenbelegung β gibt es einen Pfad π in T mit $val_{D,I,\beta}(V) = W$ für alle V auf π.

Vorbemerkung zu den nachfolgenden Beweisen.
Wir wollen die folgende (übliche) **Schreibweise** verwenden. Mit $\mathcal{D} = (D, I)$, β eine zugehörige Variablenbelegung, B eine Formel, stehe

$\mathcal{D} \models B$ für: $\mathcal{D}$ ist Modell von B
$\mathcal{D} \models B[\beta]$ für: $val_{\mathcal{D},\beta}(B) = W$.

Entsprechend schreiben wir $\mathcal{D} \models V$, $\mathcal{D} \models V[\beta]$ für eine Vorzeichenformel V, ferner $\mathcal{D} \models N$, $\mathcal{D} \models N[\beta]$, $\mathcal{D} \models \pi$, $\mathcal{D} \models \pi[\beta]$ für eine Menge N von Formeln bzw. von Vorzeichenformeln, bzw. einen Pfad π (als Menge von Vorzeichenformeln). Dabei steht $\mathcal{D} \models M$ mit einer Formelmenge M als Abkürzung für $\mathcal{D} \models B$ für alle B aus M.

Lemma 5.54
M sei eine Formelmenge ohne freie Variablen.
Das Tableau T' über M gehe aus T über M durch Anwendung einer Tableauregel hervor.
Hat T ein Modell über M, dann auch T'.

Beweis:

Der Beweis gliedert sich in acht Fälle, je nachdem, welche Tableauregel von T nach T' führt.

[1] Kurzform, neues Tableau durch Substitution

1. β-*Fall:* Auf einem Pfad π von T liegt die β-Formel V, und T' entsteht, indem der Pfad π erweitert wird einmal zum Pfad π_1 durch Anhängen der Formel V_1 und zum anderen zum Pfad π_2 durch Anhängen der Formel V_2. Nach Voraussetzung hat T ein Modell $\mathcal{D}$, d. h. für jede Variablenbelegung β existiert ein Pfad π_0 von T mit $\mathcal{D} \models M$ und $\mathcal{D} \models \pi_0[\beta]$. Wir wollen zeigen, daß $\mathcal{D}$ auch Modell von T' ist. Betrachten wir dazu eine beliebige Variablenbelegung β. Ist der Pfad π_0 von T mit $\mathcal{D} \models \pi_0[\beta]$, der nach Voraussetzung existiert, verschieden von π, dann ist π_0 auch ein Pfad in T', und wir sind fertig. Ist $\pi_0 = \pi$, so liegt V auf π_0 und somit $\mathcal{D} \models V[\beta]$. Aus der Eigenschaft von β-Formeln folgt $\mathcal{D} \models V_1[\beta]$ oder $\mathcal{D} \models V_2[\beta]$ und somit $\mathcal{D} \models \pi_1[\beta]$ oder $\mathcal{D} \models \pi_2[\beta]$, und ohnehin ist $\mathcal{D} \models M$.

2. α-*Fall:* Analog. Bleibt dem Leser überlassen.

3. γ-*Fall:* Auf dem Pfad π von T kommt die γ-Formel V vor, und T' entsteht, indem π durch Hinzufügen der Formel $V_1(y)$, für eine auf π neue freie Variable y, zu dem Pfad π_1 verlängert wird. $\mathcal{D}$ sei ein Modell von T über M. Wir zeigen, daß $\mathcal{D}$ auch Modell von T' ist. Ist β eine Belegung, dann $\mathcal{D} \models \pi_0$ für ein π_0 in T. Wenn $\pi_0 \neq \pi$, ist π_0 auch ein Pfad von T', fertig. Wenn $\pi_0 = \pi$, hat man mit $\mathcal{D} \models V[\beta]$ auch $\mathcal{D} \models V_1(y)[\beta]$, also $\mathcal{D} \models \pi_1$.

4. δ-*Fall:* Auf dem Pfad π von T kommt die δ-Formel V vor mit den freien Variablen $x_1, \ldots, x_n$, und T' entsteht, indem π durch Anhängen der Formel $V_1(f(x_1, \ldots, x_n))$ für ein neues Funktionszeichen f zu π_1 verlängert wird. Nach Voraussetzung sei $\mathcal{D}$ Modell von T über M. Wir konstruieren eine Interpretation $\mathcal{D}'$, die sich von $\mathcal{D}$ nur darin unterscheidet, daß dem Funktionszeichen f eine Interpretation $f^{\mathcal{D}'}$ zugeordnet wird. Für $d_1, \ldots, d_n \in D$ und einer Variablenbelegung β mit $\beta(x_i) = d_i$ für $i = 1, \ldots, n$ gilt entweder
$$\mathcal{D} \models V[\beta],$$
in diesem Fall gibt es ein $d \in D$ mit
$$\mathcal{D} \models V_1[\beta_x^d],$$
oder $\mathcal{D} \models V[\beta]$ gilt nicht. Im letzten Fall wählen wir einen beliebigen Wert $d \in D$. Wir definieren dann $f^{\mathcal{D}'}(d_1, \ldots, d_n) = d$ für die verschiedenen $d_1, \ldots, d_n$ mit dem entsprechenden d.

   Damit ist die Definition der Interpretation $\mathcal{D}'$ abgeschlossen. Wir wollen zeigen, daß $\mathcal{D}'$ Modell von T' ist. Es sei β eine beliebige Belegung bzgl. $\mathcal{D}'$, β ist auch Belegung bzgl. $\mathcal{D}$, da sich der Grundbereich nicht geändert hat. Es gibt π_0 in T mit $\mathcal{D} \models \pi_0[\beta]$. Ist $\pi_0 \neq \pi$, so ist π_0 auch Pfad in T'. Ist $\pi_0 = \pi$, hat man $\mathcal{D} \models V[\beta]$, und nach Konstruktion von $\mathcal{D}'$ auch $\mathcal{D}' \models V_1[\beta]$. Da in den restlichen Formeln des Pfades π_1 und in M das Zeichen f nicht vorkommt, erhalten wir insgesamt $\mathcal{D}' \models \pi_1[\beta]$ und $\mathcal{D}' \models M$.

5. V-Regel: offensichtlich, da ein Modell von T über M stets Modell von M.

■

Lemma 5.55
Ist $\mathcal{D}$ Modell von T über M und ensteht T' aus T durch Schließen eines Pfades, dann ist $\mathcal{D}$ auch Modell von T'.

Beweis:

Gemäß Voraussetzung gibt es zu jeder Belegung β einen Pfad π in T mit $\mathcal{D} \models \pi[\beta]$. T' entstehe durch Anwenden der Substitution σ und Schließen eines Pfades gemäß einer der beiden Möglichkeiten in 5.48. Sei β eine Belegung. Wir definieren β' als die Modifikation von β gemäß σ, wie im letzten Teil des Beweises zu 5.54.

Nach dem Substitutionslemma gilt

$$\mathcal{D} \models C[\beta'] \text{ gdw. } \mathcal{D} \models \sigma(C)[\beta] \text{ für alle } C,$$

so daß aus $\mathcal{D} \models \pi[\beta']$ für den zu β' gehörigen Pfad π folgt: $\mathcal{D} \models \sigma(\pi)[\beta]$ (wo $\sigma(\pi) = \{\sigma(V) \mid V \text{ auf } \pi\}$).

■

Satz 5.56 (Korrektheitssatz des Tableaukalküls)
Sei $A \in For_\Sigma, M \subseteq For_\Sigma$ alle ohne freie Variable
Wenn es ein geschlossenes Tableau für A über M gibt, dann ist $M \models A$.

Beweis:

Wir bemerken zunächst, daß ein geschlossenes Tableau für A über M offensichtlich kein Modell haben kann.

Wir definieren für die Zwecke dieses Beweises das Anfangstableau T_0 für A über M als das aus einem Pfad bestehende Tableau, das genau die Formeln

- $0A$

- $1B$ für $B \in M$

enthält. Der bisher beschriebene Kalkül schreibt zwar vor, daß $0A$ in jedem Tableau einer Ableitung auf jedem Pfad vorkommen muß, aber darüber, wann eine Voraussetzung B aus M in ein Tableau eingebaut wird, gibt es keine Festlegung. In den kleinen Beispielen, die wir bisher betrachtet haben, haben wir es so eingerichtet, daß auch die Formeln aus M von Anfang an dabei sind. Offenbar können wir diese Normierung vornehmen ohne Einschränkung der Allgemeinheit.

Sei jetzt

$$T_0, \ldots T_k, T_{k+1}, \ldots T_n$$

eine Ableitung für A über M. T_n ist nicht erfüllbar, da es ein geschlossenes Tableau ist. Dann sind aber auch alle vorangegangen Tableaux unerfüllbar. Das können wir so einsehen. Angenommen T_{k+1} ist unerfüllbar. Wäre T_k erfüllbar, so müsste nach Lemma 5.54 auch T_{k+1} erfüllbar sein. Nach unserer Annahme kann das nicht sein, also ist zwangsläufig T_k unerfüllbar. Insbesondere ist T_0 nicht erfüllbar, was nach Definition von T_0 gleichbedeutend ist mit $M \models A$.

■

5.5.3 Hintikka-Mengen

Dieses Unterkapitel bereitet den Beweis des Vollständigkeitssatzes vor, der im folgenden Unterkapitel geführt wird. Ein wesentlicher Schritt in diesem Beweis ist die Konstruktion eines Modells für Formelmengen mit speziellen Eigenschaften, eben den Hintikka-Mengen. Die Konstruktion ist unabhängig vom Rest des Beweises und kann deshalb hier schon erledigt werden.

Definition 5.57 (Hintikka-Menge)
Eine Menge H von geschlossenen Vorzeichenformeln über einer Signatur Σ heißt eine **Hintikka-Menge**, wenn die folgenden Bedingungen erfüllt sind:

(H 1) Gilt für eine α-Formel V, $V \in H$,
dann auch $V_1 \in H$ und $V_2 \in H$.

(H 2) Gilt $V \in H$ für eine β-Formel V,
dann auch $V_1 \in H$ oder $V_2 \in H$.

(H 3) Gilt $V \in H$ für eine δ-Formel V,
dann gibt es einen variablenfreien Term t mit $V_1(t) \in H$.

(H 4) Gilt $V \in H$ für eine γ-Formel V,
dann gilt $V_1(t) \in H$ für jeden variablenfreien Term t.

(H 5) Für keine Formel A kommt sowohl $1A$, als auch $0A$ in H vor

Lemma 5.58 (Modell-Lemma)
Jede Hintikka-Menge H besitzt ein Modell.

Beweis:

Wir setzen

$$D = \{t : t \text{ ein Grundterm}\}$$

Falls die ursprüngliche Signatur kein Konstantensymbol enthält fügen wir an dieser Stelle ein beliebiges neues Konstantensymbol hinzu. Wir müssen sicherstellen, daß es mindestens einen Grundterm gibt, das Universum D also nicht leer ist.
I wird definiert durch
$I(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
$(t_1, \ldots, t_n) \in I(p) \quad \Leftrightarrow \quad 1p(t_1, \ldots, t_n) \in H$
$I(p) = W \quad \Leftrightarrow \quad 1p \in H$ (für 0-stelliges p).
Mit dieser Definition gilt für jeden Grundterm:

$$I(t) = t$$

Wir beweisen diese Behauptung durch Induktion über den Termaufbau. Für $t = c$, ein Konstantensymbol, gilt nach Definition

$$I(c) = c.$$

Sei jetzt $t = f(t_1, \ldots, t_n)$:

$$\begin{array}{rcll} I(t) & = & I(f)(t_1^{\mathcal{D}}, \ldots, t_n^{\mathcal{D}}) & \text{(Def. von } I(t)) \\ & = & I(f)(t_1, \ldots, [t_n) & \text{(Ind.Vor.)} \\ & = & f(t_1, \ldots, t_n) & \text{(Def. von } I(f)) \end{array}$$

Es bleibt, um den Beweis des Modell-Lemmas zu vervollständigen, noch nachzuweisen, daß für jede Formel $V \in H$ gilt

$$(D, I) \models V.$$

Dieser Nachweis wird wieder durch Induktion über den Aufbau von V geführt. (Man beachte, daß H nur geschlossene Formeln enthält.)

1. Fall: $V = 1p(t_1, \ldots, t_n)$

Falls $V \in H$, dann gilt $\mathcal{D} \models V$ nach Definition von $\mathcal{D}$.

2. Fall: $V = 0p(t_1, \ldots, t_n)$.

Wenn $V \in H$, dann gilt wegen (H 5) $1p(t_1, \ldots, t_n) \notin H$. Nach Definition von (D, I) also $(D, I) \not\models p(t_1, \ldots, t_n)$, d. h. $(D, I) \models \neg p(t_1, \ldots, t_n)$

Die weiteren Induktionsschritte sind jetzt einfache Konsequenzen aus (H 1) bis (H 4).

∎

5.5.4 Vollständigkeit

Satz 5.59 (Vollständigkeit von T)
Sei A eine Formel und M eine Menge von Formeln, alle ohne freie Variable. Gilt $M \models A$, dann gibt es ein geschlossenes Tableau für A über M.

Die Beweisidee ist Folgende: Wir geben *Tableaukonstruktionsvorschriften* an, nach denen ein geschlossenes Tableau für A über M konstruiert werden soll. Wichtig dabei ist, daß die Formeln für die Regelanwendungen *fair* ausgewählt werden d.h. nicht irgendwelche Möglichkeiten unberücksichtigt bleiben (vgl. oben 5.52). Wird auf diese Weise kein geschlossenes Tableau gefunden, so zeigen wir, daß dann $M \cup \{\neg A\}$ ein Modell haben muß, was im Widerspruch zu $M \models A$ steht.

Definition 5.60 (Tableaukonstruktion)

Es werden sukzessiv nicht nur Tableaux konstruiert, sondern parallel zu jedem Tableau eine Substitution für die freien Variablen in dem jeweiligen Tableau. Starttableau und -Subtitution ist das Paar $(\{0A\}, id)$. (T_{n+1}, σ_{n+1}) entsteht aus (T_n, σ_n), indem das Tableau T_n durch eine Tableauregel *fair* erweitert wird. Geschieht diese Erweiterung nicht durch Anwendung der γ-Regel, so ist $\sigma_{n+1} = \sigma_n$. Ansonsten entsteht σ_{n+1} aus σ_n, indem es um eine Substitution für die neue Variable aus der γ-Regel erweitert wird. Wodurch die Variable substituiert wird, ergibt sich aus einer (beliebigen aber festen) Aufzählung aller Grundterm $\{t_n : n \geq 1\}$. Bei der n-ten Anwendung der γ-Regel auf einem Pfad auf eine Vorzeichenformel V wird die neue Variable durch t_n substituiert. Wann Tableauerweiterungen *fair* sind, wird in der folgenden Definition festgelegt.

Definition 5.61 (Fairnessforderungen)

Die Konstruktion der (T_n, σ_n) ist fair, wenn:

1. nur Pfade π aus T_i verlängert werden, für die $\sigma_i(\pi)$ nicht geschlossen ist (geschlossene Pfade werden nicht erweitert),

2. für jede α-, β- oder δ-Formel V, die in einem Tableau T_j vorkommt, ein Index i mit $j < i$ existiert, so daß für jeden Pfad π in T_i, der V enthält, gilt: $\sigma_i(\pi)$ ist geschlossen oder V wurde zur Pfadverlängerung genutzt,

3. für jede γ-Formel V, die in einem Tableau T_j vorkommt, und jedem $m > 0$ ein Index i mit $j < i$ existiert, so daß für jeden Pfad π in T_i, der V enthält, gilt: $\sigma_i(\pi)$ ist geschlossen oder V wurde zur Pfadverlängerung m-mal genutzt (hier kommen die unterschiedlichen Substitutionen zum tragen, die neue Variable in T_i wird in σ_i durch t_m substituiert),

4. für alle $B \in M$ ein Index i existiert, so daß für jeden Pfad π in T_i gilt: $\sigma_i(\pi)$ ist geschlossen oder $1B$ liegt auf π.

Man beachte, daß Fairness eine sehr einschneidende Forderung ist, die im wesentlichen eine Art Breitensuche erzwingt.

Beweis von Satz 5.59
Die Aussage des Satzes kann erweitert werden zu:

> Sei A eine Formel und M eine Menge von Formeln.
> Gilt $M \models A$ und konstruiert man nach den Tableaukonstruktionsforschriften 5.60 Tableaux für A über M, dann wird nach endlich vielen Regelanwendungen ein geschlossenes Tableau $\sigma_i(T_i)$ gefunden.

Wir erinnern daran, daß wir uns auf geschlossene Formeln A beschränkt haben.

Wir wollen einen Widerspruchsbeweis führen, d.h. wir nehmen an, daß kein geschlossenes Tableau gefunden wird.

Man beachte, daß bei der Konstruktion der T_i die Substitutionen $\sigma(i)$ nicht tatsächlich ausgeführt werden. Sie werden nur "notiert", um auszuweisen, daß ein Pfad in einem T_i "jetzt geschlossen werden könnte". Dieser Pfad wird dann nicht mehr fortgesetzt. Es entsteht also jeweils T_{i+1} aus T_i durch Anhängen von Knoten an einem Pfad π, so daß $\sigma_i(\pi)$ nicht geschlossen ist. Durch Vereinigung der Tableaux T_i, $i = 0, 1, 2, ...$, entsteht ein Tableau T, und aus der Voraussetzung "kein $\sigma_i(T_i)$ ist geschlossen" folgt, daß T unendlich ist. Wir verwenden nun das aus der Graphentheorie bekannte

> *Königs Lemma:* In jedem unendlichen, endlich verzweigenden Baum existiert ein unendlicher Pfad.

Es sei π ein unendlicher Pfad in T. Wir zeigen: *Es gibt eine Hintikka-Menge H, die alle Vorzeichenformeln auf $\sigma(\pi)$ enthält.*

Man zeigt zuerst, daß $\sigma(\pi)$, aufgefaßt als Formelmenge, die Eigenschaften (H 1) – (H 5) erfüllt. (H 1) bis (H 4) folgen unmittelbar aus den Fairnessforderungen, während (H 5) aus der Tatsache folgt, daß der Pfad nicht geschlossen wird.

Betrachten wir (H 4) etwas genauer. Zunächst wissen wir, daß nach der Fairnessvorschrift jede γ-Formeln $V \in \sigma(\pi)$ unendlich oft zur Verlängerung des Pfades benutzt wurde und damit für jeden variablenfreien Term t auch $V_1(t)$ auf dem Pfad liegt (V_1 sei der Rumpf von V ohne Quantor).

Nach Lemma 5.58 hat H ein Modell. Da H $0A$ und alle $1B$, $B \in M$ enthält, ist dieses auch Modell von $M \cup \{\neg A\}$. Also $M \not\models A$ entgegen Annahme.

■

5.5.5 Der Satz von Herbrand

Dieser Unterabschnitt ist einem für die Historie des automatischen Beweisens wichtigen Resultat gewidmet, dem Satz von Herbrand, siehe Satz 5.64 auf Seite 204. Dieser Unterabschnitt ist für die folgenden Kapitel nicht erforderlich.

Wie in Definition 4.3 vereinbart bezeichnet $Term^0_\Sigma$ die Menge aller Grundterme über Σ.

Die im Beweis von Lemma 5.58 konstruierte Struktur $\mathcal{D}$ folgt einem bestimmten Muster, sie gehört zur Klasse der *Herbrand* Algebren, die wir jetzt einführen werden.

Definition 5.62 (Herbrand-Algebra)
Die Signatur Σ enthalte mindestens eine Konstante. Eine Interpretation (D, I) von Σ heißt *Herbrand-Interpretation* oder *Herbrand-Algebra* genau dann, wenn

1. $D = Term^0_\Sigma$
2. $I(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ für alle $n-$stelligen Funktionssymbole f, und beliebige Grundterme $t_1, \ldots, t_n$. (Im Falle $n = 0$ heißt das: $I(c) = c$.)

Zu einer Formelmenge $M \subseteq For_\Sigma$ ist ein *Herbrand-Modell* von M eine Herbrand-Algebra über Σ, welche Modell von M ist.

Die Voraussetzung, daß Σ mindestens eine Konstante enthalte, ist nicht sehr einschränkend: Ist das nicht der Fall, so nehme man eine Konstante hinzu. Diese Voraussetzung ist allerdings notwendig. Enthält Σ keine Konstante, dann ist $Term^0_\Sigma$ die leere Menge. Prädikatenlogische Strukturen müssen aber nach unserer Definition ein nicht-leeres Universum haben.

In einer Herbrand-Algebra wird also jeder Grundterm als er selbst interpretiert,

$$I(t) = t \text{ für Grundterme,}$$

insbesondere gilt das für Konstanten. Spielraum für *verschiedene* Herbrand-Algebren gibt es nur bei der Interpretation der Prädikatsymbole. Wir notieren noch das

Korollar 5.63
Ist $(Term^0_\Sigma, I)$ Herbrand-Algebra über Σ, $t \in Term_\Sigma$ mit $Var(t) = \{x_1, \ldots, x_n\}$, und β eine Belegung der Variablen mit Grundtermen, dann ist

$$val_{D,I,\beta}(t) = \{x_1/\beta(x_1), \ldots, x_n/\beta(x_n)\}(t).$$

Beweis

Übung

(Hieraus folgt nochmals, daß in einer Herbrand-Algebra gilt $I(t) = t$ für jeden Grundterm t.)

Satz 5.64 (2. Satz von Herbrand)
Sei ϕ eine quantorenfreie Formel ohne Gleichheit mit einer freien Variablen x . Dann gilt

$\exists x\phi$ ist allgemeingültig

gdw

es gibt eine natürliche Zahl n und Grundterme $t_1, \ldots, t_n$, sodaß
$\phi(t_1) \vee \ldots \vee \phi(t_n)$ allgemeingültig ist.

Beweis des 2. Satzes von Herbrand

Ist $\exists x\phi$ allgemeingültig, so gibt es nach dem Vollständigkeitsatz 5.59 ein geschlossenes Tableau T' (nach Anwednung einer unifizierenden Substitution σ). mit der Wurzelmarkierung $\forall x\neg\phi$. Falls in T' noch freie Variablen vorkommen ersetzt man sie alle durch eine Konstante c, die es in Σ nach Voraussetzung geben muß. So entsteht ein variablenfreies geschlossenes Tableau mit Wurzel $\forall x\neg\phi$. Jede Anwendung der Gammaregel auf die Wurzelformel setzt damit eine Formel der Form $\neg\phi(t_i/x)$ auf einen Tableauast. Sei n die Anzahl der Anwendungden dieser Regel. Ersetzen wird die Markierung des Wurzelknotens durch $\neg\phi(t_1/x) \wedge \ldots \wedge \neg\phi(t_{1n}/x)$ so läßt sich dasselbe Tableau T konstruieren. Anstelle der Anwenddung einer der n Gammaregeln kann man jetzt die Alpharegel mit der geänderten Markierung des Wurzelknotens benutzen. Es gilt als ein geschlossenes Tableau für $\neg\phi(t_1/x) \wedge \ldots \wedge \neg\phi(t_{1n}/x)$. Nach dem Korrekheitssatz für den Tableaukalkül ist $\phi(t_1/x) \vee \ldots \vee \phi(t_{1n}/x)$ allgemeingültig.

■

Die Aussage des 2.Satzes von Herbrand hat nicht mit dem Tableaukalkül zu tun. Dieser ist sozusagen nur ein Hilfmittel für den Beweis. Es gibt auch einen Beweis, der weitest gehend unabhängig ist von einem Kalkül. Um diesen Beweis zu präsentieren brauchen wir als Vorbereitung die nächste Definition und den 1.Satz von Herbrand.

Definition 5.65
Sei $A := \forall x_1 \ldots \forall x_n B$ mit quantoremfreiem B eine geschlossene Formel. Eine **Grundinstanz** von A ist eine Formel

$$\{x_1/t_1, \ldots, x_n/t_n\}(B)$$

mit: $t_1, \ldots, t_n \in Term^0_\Sigma$.
Ist M eine Menge geschlossener, universell quantifizierter Formeln, so sei

$$\text{Grundinstanzen}(M)$$

die Menge aller Grundinstanzen von Formeln in M.

Satz 5.66 (1. Satz von Herbrand)
Σ enthalte mindestens eine Konstante, und es sei M eine Menge geschlossener, universell quantifizierter Formeln. Ferner enthalte keine Formel in M das Gleichheitssymbol $\doteq$. Dann sind äquivalente Aussagen

1. M hat ein Modell
2. M hat ein Herbrand-Modell
3. Grundinstanzen(M) hat ein Modell
4. Grundinstanzen(M) hat ein Herbrand-Modell.

Beweis Die Implikationen 4 $\Rightarrow$ 3 und 2 $\Rightarrow$ 1 sind trivial; ebenso wegen der Allgemeingültigkeit von

$$\forall x_1 \ldots \forall x_n B \rightarrow \{x_1/t_1, \ldots, x_n/t_n\}(B)$$

die Implikationen 1 $\Rightarrow$ 3 und 2 $\Rightarrow$ 4. Wir brauchen nur noch zu zeigen, daß 3 $\Rightarrow$ 2.

Es sei (D, I) ein Modell von Grundinstanzen(M). Wir definieren eine Herbrand--Interpretation $(Term^0_\Sigma, J)$. Auf den Funktionssymbolen ist J schon vorgeschrieben. Wir setzen.

$J(P)$	$:=$	$I(P)$ für aussagenlogische Atome P
$J(p)$	$:=$	$\{(t_1, \ldots, t_n) \mid t_1, \ldots, t_n \in Term^0_\Sigma, val_{D,I}(p(t_1, \ldots, t_n)) = W\}$ für Prädikatsymbole p einer Stelligkeit $n \geq 1$.

Das heißt also, es gilt

$$val_{Term^0_\Sigma, J}(A) = val_{D,I}(A)$$

für jede geschlossene atomare Formel A, das keine Gleichung ist. Mittels trivialer Induktion haben wir dies dann auch für alle geschlossenen, quantorenfreien Formeln, die $\doteq$ nicht enthalten. Da (D, I) Modell von Grundinstanzen(M) ist, folgt

$$val_{Term^0_\Sigma, J}(\{x_1/t_1, \ldots, x_n/t_n\}(B)) = W$$

für alle Formeln $\forall x_1 \ldots \forall x_n B \in M$ und alle $(t_1, \ldots, t_n) \in (Term^0_\Sigma)^n$. Mit Hilfe des Substitutionslemmas schließen wir hieraus

$$val_{Term^0_\Sigma, J}(\forall x_1 \ldots \forall x_n(B)) = W$$

für jedes $\forall x_1 \ldots \forall x_n B \in M$, d. h. $(Term^0_\Sigma, J)$ ist Modell von M.

■

2.Beweis des 2. Satzes von Herbrand

$\exists x \phi$ ist allgemeingültig
$\Leftrightarrow$ $\neg \exists x \phi$ besitzt kein Modell
$\Leftrightarrow$ $\forall x \neg \phi$ besitzt kein Modell
$\Leftrightarrow$ $\{\neg \phi(t) \mid t$ Grundterm$\}$ besitzt kein Modell
$\Leftrightarrow$ es gibt ein n und $t_1, \ldots, t_n$ so daß
$\{\neg \phi(t_1), \ldots, \neg \phi(t_n)\}$ kein Modell besitzt
(Anwendung des Endlichkeitssatzes, Satz 5.84)
$\Leftrightarrow$ $\neg \phi(t_1) \wedge \ldots \wedge \neg \phi(t_n)$ besitzt kein Modell
$\Leftrightarrow$ $\phi(t_1) \vee \ldots \vee \phi(t_n)$ ist allgemeingültig

■

5.5.6 Übungsaufgaben

Übungsaufgabe 5.5.1
Zeigen Sie, daß die Formel

$$\forall x \forall y(\phi(x,y) \vee \psi(x,y)) \rightarrow \forall x \exists y \phi(x,y) \vee \exists x \forall y \psi(x,y)$$

eine Tautologie ist.
Diese simple Tautologie ist der Ausgangspunkt einer Methode, genannt Funktionseinführung (function introduction), zur Beschleunigung von Resolutionsbeweisen. Siehe etwa [BL92].

Übungsaufgabe 5.5.2
Die Formulierungen in den Sätzen 5.56 und 5.59 sind so allgemein gefasst, daß sie auch die Fälle abdecken, in denen die Prämissenmenge M oder das zu beweisende Theorem A freie Variable enthalten. In der Regel wird das nicht der Fall sein. In dieser Aufgabe wollen wir genau diesen seltenen Fall betrachten. Was geschieht, wenn man die Bildung des universellen Abschlusses weglässt? Geben Sie Beispiele, daß keine der beiden Implikationen in der folgenden Äquivalenzaussage richtig ist:

$M \models A$ gilt genau dann, wenn es ein geschlossenes Tableau für A über M gibt.

Übungsaufgabe 5.5.3
In der Definition 4.26 einer prädikatenlogischen Interpretation war verlangt worden, daß nur nichtleere Mengen als Trägermengen D auftreten können. Für die Zwecke dieser Übungsaufgabe wollen wir diese Einschränkung fallen lassen und auch $D = \emptyset$ zulassen. Wir nennen dies die *Nullsemantik*

1. Geben Sie ein Beispiel für eine Formel an, die in der üblichen Semantik allgemeingültig ist, aber in der Nullsemantik nicht.
2. Wenn man den in diesem Kapitel beschrieben Tableaukalkül für die Nullsemantik benutzt, welche Eigenschaft geht dabei verloren: die Vollständigkeit oder die Korrektheit?
3. Wie muß das Tableauverfahren abgeändert werden, damit es vollständig und korrekt bezüglich der Nullsemantik ist?

Übungsaufgabe 5.5.4
Man beweise im Tableaukalkül:

$$\forall t(a(next(t)) \leftrightarrow b(t) \wedge \neg reset(t)) \quad \vdash \quad \forall t(reset(t) \rightarrow \neg a(next(t)))$$

Übungsaufgabe 5.5.5

Zeigen Sie, daß eine Robbins Algebra $\mathcal{R}$, in der die Formel

$$\forall x(\neg\neg x = x)$$

gilt, schon eine Boolesche Algebra ist.

5.6 Sequenzenkalkül

Sequenzenkalküle wurden um 1935 von G. Gentzen eingeführt. Deswegen werden sie auch *Gentzenkalküle* genannt. Sie spielen eine zentrale Rolle in der mathematischen Logik, insbesondere in der Beweistheorie und haben etwa ab der 80-er Jahren vor allem im automatischen Beweisen an Bedeutung gewonnen. Für eine weitergehende Darstellung konsultiere man das Buch [Gal86].

5.6.1 Syntax und Semantik

Definition 5.67 (Sequenz)
Eine *Sequenz* wird notiert als eine Folge zweier endlicher Mengen aussagenlogischer Formeln getrennt durch das Symbol =>:

$$\Gamma => \Delta.$$

Γ wird Antezedent und Δ Sukzedent genannt. Sowohl links wie rechts vom Sequenzenpfeil => kann auch die leere Menge stehen. Wir schreiben dann

$$=> \Delta \text{ bzw. } \Gamma => \text{ bzw. } => .$$

Das etwas ungewöhnliche Wort „Sequenz" für eine Zeichenfolge der Form $\Gamma => \Delta$ hat sich fest eingebürgert. Die englischen Übersetzung von „Sequenz" heißt *sequent* und nicht etwa *sequence*.

Wir benutzen die folgenden abkürzenden Schreibweisen zur Einsparung der Vereinigungsoperation zwischen Mengen. Wir schreiben also für Formeln $A_1, \ldots, A_n, B$ und Formelmengen Γ, Δ:

$$\begin{array}{lll} A_1, \ldots, A_n & \text{für} & \{A_1, \ldots, A_n\} \\ \Gamma, B & \text{für} & \Gamma \cup \{B\} \\ \Gamma, \Delta & \text{für} & \Gamma \cup \Delta \end{array}$$

Ist Γ eine endliche Menge von Formeln, so schreiben wir $\bigwedge \Gamma$ für die Konjunktion der Formeln in Γ und $\bigvee \Gamma$ für die Disjunktion. Für $\Gamma = \emptyset$ setzen wir dabei $\bigwedge \emptyset = \mathbf{1}$ und $\bigvee \emptyset = \mathbf{0}$.

Definition 5.68 (Auswertung von Sequenzen)

Sei I eine Interpretation der Atome. Wir setzen die Auswertung val_I von Formeln einer Auswertung der Sequenzen fort durch.

$$val_I(\Gamma => \Delta) = val_I(\bigwedge \Gamma \to \bigvee \Delta)$$

Die Begriffe *Tautologie, Erfüllbarkeit* werden jetzt in naheliegender Weise auch für Sequenzen anstelle von einzelnen Formeln benutzt.

5.6.2 Der aussagenlogische Kalkül

Definition 5.69 (Axiome und Regeln des Sequenzenkalküls S0)

axiom

$$\frac{}{\Gamma, F => F, \Delta}$$

0-left

$$\frac{}{\Gamma, \mathbf{0} => \Delta}$$

1-right

$$\frac{}{\Gamma => \mathbf{1}, \Delta}$$

not-left

$$\frac{\Gamma, => F, \Delta}{\Gamma, \neg F => \Delta}$$

not-right

$$\frac{\Gamma, F => \Delta}{\Gamma => \neg F, \Delta}$$

impl-left

$$\frac{\Gamma => F, \Delta \quad \Gamma, G => \Delta}{\Gamma, F \rightarrow G => \Delta}$$

impl-right

$$\frac{\Gamma, F => G, \Delta}{\Gamma => F \rightarrow G, \Delta}$$

and-left

$$\frac{\Gamma, F, G => \Delta}{\Gamma, F \wedge G => \Delta}$$

and-right

$$\frac{\Gamma => F, \Delta \quad \Gamma => G, \Delta}{\Gamma => F \wedge G, \Delta}$$

or-left

$$\frac{\Gamma, F => \Delta \quad \Gamma, G => \Delta}{\Gamma, F \vee G => \Delta}$$

or-right

$$\frac{\Gamma => F, G, \Delta}{\Gamma => F \vee G, \Delta}$$

Definition 5.70 (Beweisbaum)
Ein Beweisbaum ist ein Baum, dessen Knoten mit Sequenzen markiert sind und für jeden Knoten n die folgende Einschränkung erfüllt:

1. Jeder Knoten hat keinen, einen oder zwei Nachfolgerknoten.
2. ist n_1 der einzige Nachfolgerknoten von n und sind $\Gamma => \Delta$ und $\Gamma_1 => \Delta_1$ die Markierungen von n und n_1, dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 => \Delta_1}{\Gamma => \Delta}$$

3. besitzt n die beiden Nachfolgerknoten n_1 und n_2 und sind $\Gamma => \Delta$, $\Gamma_1 => \Delta_1$ und $\Gamma_2 => \Delta_2$ die Sequenzen an den Knoten n, n_1 und n_2 dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 => \Delta_1 \quad \Gamma_2 => \Delta_2}{\Gamma => \Delta}$$

Wir nennen einen Beweisbaum *geschlossen* oder *vollständig* wenn er zusätzlich noch die folgende Bedingung erfüllt:

4. jeder Knoten n, der keinen Nachfolgerknoten hat, muß mit einem Axiom markiert sein.

Ein Beispiel eines Beweisbaumes ist in Abbildung 5.3 zu sehen. Die Definition 5.70 eines Beweisbaumes legt nicht fest in welcher Reihenfolge man ihn konstruiert. Es ist jedoch in dem meisten Fällen zweckmäßig mit dem Wurzelknoten zu beginnen. Bei diesem Vorgehen werden die Regeln aus Definition 5.69 *von unten nach oben* benutzt.

Definition 5.71 (Ableitbarkeit in S0)
Für $A \in For0_\Sigma$, $M \subseteq For0_\Sigma$ sagen wir, daß A *aus* M *ableitbar in* **S**0 ist, in Zeichen:

$$M \vdash_S A,$$

genau dann, wenn ein geschlossener Beweisbaum mit Wurzelmarkierung $M => A$ in **S**0 existiert.

Bemerkung 5.72
Für den seltenen, aber nicht auszuschließenden Fall, daß man mit einer unendlichen Menge M von Vorausetzungen arbeiten möchte, ist das bisherige Vorgehen nicht geeignet. Man müsste dazu erklären, was die Konjunktion einer unendlichen Formelmenge sein soll. In diesem Fall führt man eine neue Beweisregel ein:

use premiss

$$\frac{B, \Gamma => \Delta}{\Gamma => \Delta}$$

für $B \in M$

Satz 5.73 (Korrektheit und Vollständigkeit von S0)
Für jede endliche Menge aussagenlogischer Formeln M und jede aussagenlogische Formel A gilt:

$$M \models A \quad \Leftrightarrow \quad M \vdash_{\mathbf{S0}} A.$$

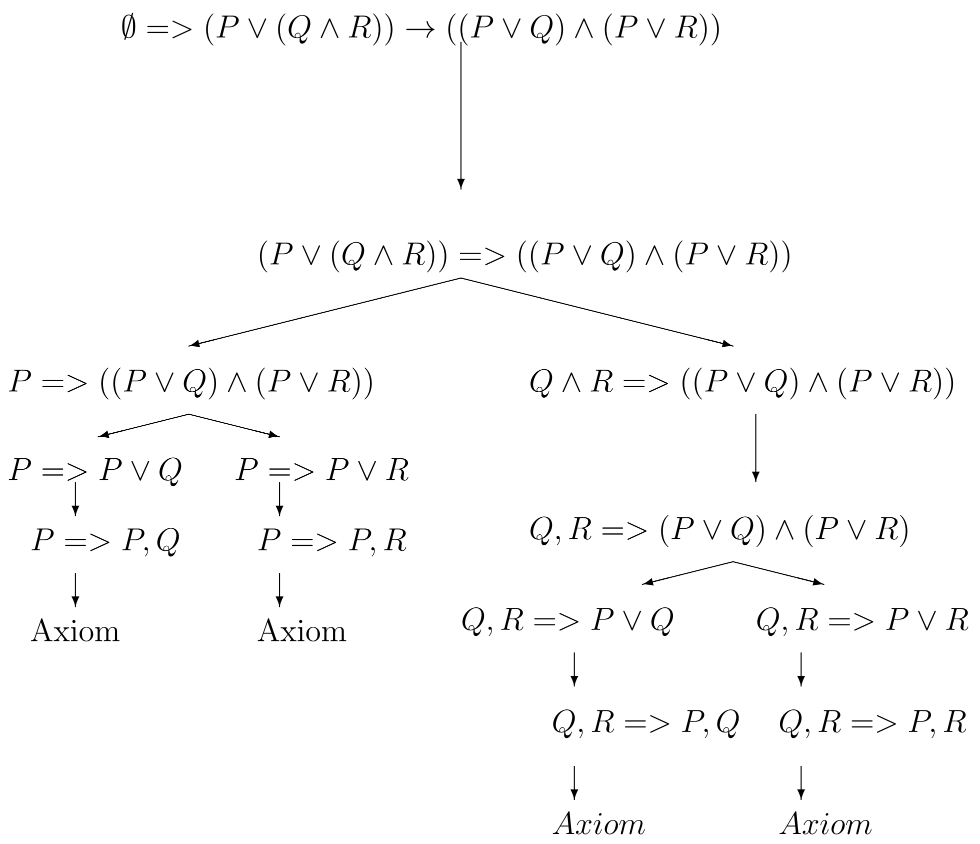


Abbildung 5.3: Beispiel eines Beweisbaums im Sequenzenkalkül

Beweis:

Korrektheit Wir gehen aus von einem geschlossenen Beweisbaums $\mathcal{T}$ mit Wurzelmarkierung $M \Rightarrow A$ und müssen zeigen, daß die Sequenz $M \Rightarrow A$ allgemeingültig ist. Dazu genügt es zu zeigen

1. jede axiomatische Sequenz $\Gamma, A \Rightarrow \Delta, A$ ist allgemeingültig und

2. für jede Beweisregel

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \quad \Gamma_2 \Rightarrow \Delta_2}{\Gamma \Rightarrow \Delta}$$

gilt wenn $\Gamma_1 => \Delta_1$ und $\Gamma_2 => \Delta_2$ allgemeingültig sind, dann ist auch $\Gamma => \Delta$ allgemeingültig.

3. analog zu (2) für Beweisregeln mit nur einer Prämisse.

Teil 1 ist unmittelbar einsichtig.
Zu**Teil 2** führen wir zwei Beispiele vor und vertrauen darauf, daß der Leser danach weiss, wie er die restlichen Fälle selbst nachrechnen kann.

or-left

$$\frac{\Gamma, H => \Delta \quad \Gamma, G => \Delta}{\Gamma, H \vee G => \Delta}$$

Wir habe als Voraussetzung, daß $\bigwedge \Gamma \wedge H \rightarrow \bigvee \Delta$ und $\bigwedge \Gamma \wedge G \rightarrow \bigvee \Delta$ allgemeingültig sind und sollen zeigen, daß das auch für $\bigwedge \Gamma \wedge (H \vee G) \rightarrow \bigvee \Delta$ zutrifft. Für jede Belegung I müssen wir also $val_I(\bigwedge \Gamma \wedge (H \vee G) \rightarrow \bigvee \Delta) = \mathbf{W}$ zeigen. Gilt $val_I(\bigwedge \Gamma \wedge (H \vee G)) = \mathbf{F}$, dann sind wir trivialerweise fertig. Gilt $val_I(\bigwedge \Gamma \wedge (H \vee G)) = \mathbf{W}$, dann gilt $val_I(\bigwedge \Gamma \wedge H) = \mathbf{W}$, oder $val_I(\bigwedge \Gamma \wedge G) = \mathbf{W}$. Indem wir von der ersten bzw der zweiten Voraussetzung Gebrauch machen folgt $val_I(\bigvee \Delta) = \mathbf{W}$ und wir sind fertig.

impl-right

$$\frac{\Gamma, H => G, \Delta}{\Gamma => H \rightarrow G, \Delta}$$

Als Voraussetzung haben wir die Allgemeingültigkeit von $\bigwedge \Gamma \wedge H \rightarrow G \vee \bigvee \Delta$ und sollen für jede Interpretation I zeigen $val_I(\bigwedge \Gamma \rightarrow (H \rightarrow G \vee \bigvee \Delta)) = \mathbf{W}$. Nehmen wir also $val_I(\bigwedge \Gamma) = \mathbf{W}$ an (der Fall $val_I(\bigwedge \Gamma) = \mathbf{F}$ führt wieder trivialerweise zum Ziel) und versuchen $val_I(H \rightarrow G \vee \bigvee \Delta)) = \mathbf{W}$ zu zeigen. Wir brauchen nur den Fall $val_I(\bigvee \Delta) = \mathbf{F}$ weiter zu verfolgen, im Fall $val_I(\bigvee \Delta) = \mathbf{W}$ sind wir wieder sofort fertig. Wir kommen zur letzten Fallunterscheidung. Gilt $val_I(H) = \mathbf{F}$, dann folgt trivialerweise $val_I(H \rightarrow G) = \mathbf{W}$ und wir haben es wieder geschafft. Es bleibt der Fall $val_I(H) = \mathbf{W}$. Dann ergibt sich aber zusammengenommen $val_I(\bigwedge \Gamma \wedge H) = \mathbf{W}$. Jetzt folgt mit der Voraussetzung $val_I(G \vee \bigvee \Delta) = \mathbf{W}$. Da wir aber im Laufe unserer Fallunterscheidungen uns im Fall $val_I(\bigvee \Delta) = \mathbf{F}$ befinden, muß $val_I(G) = \mathbf{W}$ gelten. Insgesamt haben wir damit $val_I(H \rightarrow G) = \mathbf{W}$ erhalten, wie gewünscht.
Teil 3 Analog zu Teil 2.

∎

Vollständigkeit Angenommen es gibt keinen geschlossenen Beweisbaum mit Wurzelmarkierung $\Gamma => \Delta$. Wir wollen zeigen, daß $\Gamma => \Delta$ nicht allgemeingültig ist. Sei $\mathcal{T}$ ein Beweisbaum mit Wurzelmarkierung $\Gamma => \Delta$ auf den keine weitere Regelanwendung mehr möglich ist. Nach Annahme kann $\mathcal{T}$ nicht geschlossen sein. Es gibt also einen Knoten n ohne Nachfolgerknoten, der mit $A_1, \ldots, A_n => B_1, \ldots, B_k$ markiert ist. Da keine weitere Regel auf dieses Blatt anwendbar ist, sind alle Formeln auf der Sequenz entweder Atome oder **0** oder **1** und es gilt $\{A_1, \ldots A_n\} \cap \{B_1, \ldots, B_k\} = \emptyset$. Aus dem selben Grund kann **0** nicht links und **1** nicht rechts auf der Sequenz auftreten. Die Sequenz $A_1, \ldots, A_n => B_1, \ldots, B_k$ ist nicht allgemeingültig, denn eine erfüllende Belegung I kann konstruiert werden, indem $I(A_i) = \mathbf{W}$ für alle Variablen links $(1 \leq i \leq n)$ und $I(B_j) = \mathbf{F}$ für alle Variablen rechts $(1 \leq j \leq k)$ gesetzt wird. Damit ist $val_I(A_1, \ldots, A_n => B_1, \ldots, B_k) = \mathbf{F}$.

Im Korrektheitsteil dieses Beweises haben wir gezeigt, daß für jede Regel

$$\frac{\Gamma_1 => \Delta_1 \quad \Gamma_2 => \Delta_2}{\Gamma => \Delta}$$

gilt: Wenn die beiden Prämissen (oder die eine, wenn es nur eine gibt) allgemeingültig sind, dann ist auch die Konklusion allgemeingültig. Umformuliert heißt das: Wenn $\Gamma => \Delta$ nicht allgemeingültig ist, dann ist $\Gamma_1 => \Delta_1$ oder $\Gamma_2 => \Delta_2$ nicht allgemeingültig. In dem Beweisbaum $\mathcal{T}$ haben wir schon einen Blattknoten gefunden, der mit einer nicht allgemeingültigen Sequenz markiert ist. Wiederholte Anwendung der genannten Eigenschaft der Beweisregeln führt zur Aussage, daß die Markierung des Wurzelknotens nicht allgemeingültig ist.

■

5.6.3 Der prädikatenlogische Kalkül

Es gibt verschiedene Tableaukalküle, in 5.5 wurde einer von ihnen vorgestellt. Es gibt auch verschiedene Sequenzenkalküle, vgl. die Bücher von [SA91] und [Gal86]. Wie im Falle der Aussagenlogik dargestellt, besteht eine enge Verwandtschaft zwischen Tableau- und bestimmten Sequenzenkalkülen, die in der Prädikatenlogik erhalten bleibt: Von Tableaux kann man zu sogenannten Blocktableaux übergehen, und von diesen zu Beweisbäumen für Sequenzen. Man erhält auf diese Weise einen zum Tableauxkalkül 5.5 gehörigen Sequenzenkalkül (der allerdings die Eigenschaft hat, daß eine pfadschließende Substitution auf den gesamten Beweisbaum anzuwenden ist, diesen also ändert.)

Wir betrachten gleich Formeln, die auch das $\doteq$-Zeichen enthalten können.

Die beiden folgenden Definitionen sind die prädikatenlogische Fortsetzung der aussagenlogischen Definitionen 5.67 und 5.68.

Definition 5.74 (Sequenz)
Eine *Sequenz* wird notiert als eine Folge zweier endlicher Mengen prädikatenlogischer Formeln getrennt durch das Symbol $=>$:

$$\Gamma => \Delta.$$

Γ wird Antezedent und Δ Sukzedent genannt. Sowohl links wie rechts vom Sequenzenpfeil $=>$ kann auch die leere Menge stehen.

Definition 5.75 (Auswertung von Sequenzen)
Sei $\mathcal{D}$ eine prädikatenlogische Struktur und β eine Variablenbelegung:

$$val_{\mathcal{D},\beta}(\Gamma => \Delta) = val_{\mathcal{D},\beta}(\bigwedge \Gamma \rightarrow \bigvee \Delta)$$

Es gelten die üblichen Vereinbarungen für leere Disjunktionen und Konjunktionen.

Definition 5.76 (Der Sequenzenkalkül S)
Die Regeln des Sequenzenkalküls sind in den Abbildungen 5.4, 5.5 und 5.6 zusammengestellt.

Die Definition eines Beweisbaum ist wörtlich dieselbe wie im aussagenlogischen Fall, Definition 5.70, mit einer kleinen Änderung in der Definition eines abgeschlossenen Beweisbaums.

Definition 5.77 (Beweisbaum)
Ein Ableitungsbaum ist ein Baum, so daß jeder innere Knoten ein oder zwei Nachfolger hat. Die Knoten des Baums sind mit Sequenzen markiert und für jeden Knoten n sind die folgende Einschränkung erfüllt:

1. ist n_1 der einzige Nachfolgerknoten von n und sind $\Gamma => \Delta$ und $\Gamma_1 => \Delta_1$ die Markierungen von n und n_1, dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 => \Delta_1}{\Gamma => \Delta}$$

2. besitzt n die beiden Nachfolgerknoten n_1 und n_2 und sind $\Gamma => \Delta$, $\Gamma_1 => \Delta_1$ und $\Gamma_2 => \Delta_2$ die Sequenzen an den Knoten n, n_1 und n_2 dann gibt es eine Sequenzenregel

$$\frac{\Gamma_1 => \Delta_1 \quad \Gamma_2 => \Delta_2}{\Gamma => \Delta}$$

axiom

$$\frac{}{\Gamma, F => F, \Delta}$$

0-left

$$\frac{}{\Gamma, \mathbf{0} => \Delta}$$

1-right

$$\frac{}{\Gamma => \mathbf{1}, \Delta}$$

not-left

$$\frac{\Gamma, => F, \Delta}{\Gamma, \neg F => \Delta}$$

not-right

$$\frac{\Gamma, F => \Delta}{\Gamma => \neg F, \Delta}$$

impl-left

$$\frac{\Gamma => F, \Delta \quad \Gamma, G => \Delta}{\Gamma, F \to G => \Delta}$$

impl-right

$$\frac{\Gamma, F => G, \Delta}{\Gamma => F \to G, \Delta}$$

and-left

$$\frac{\Gamma, F, G => \Delta}{\Gamma, F \wedge G => \Delta}$$

and-right

$$\frac{\Gamma => F, \Delta \quad \Gamma => G, \Delta}{\Gamma => F \wedge G, \Delta}$$

or-left

$$\frac{\Gamma, F => \Delta \quad \Gamma, G => \Delta}{\Gamma, F \vee G => \Delta}$$

or-right

$$\frac{\Gamma => F, G, \Delta}{\Gamma => F \vee G, \Delta}$$

Abbildung 5.4: Aussagenlogische Sequenzenregeln

all-left

$$\frac{\Gamma, \forall x F, F(x/X) => \Delta}{\Gamma, \forall x F => \Delta}$$

wobei X eine neue Variable ist.

all-right

$$\frac{\Gamma => F(x/f(x_1,\ldots,x_n)), \Delta}{\Gamma => \forall x F, \Delta}$$

wobei f ein neues Funktionssymbol ist und $x_1,\ldots,x_n$ alle freien Variablen in $\forall x F$.

ex-right

$$\frac{\Gamma => \exists x F, F(x/X), \Delta}{\Gamma, => \exists x F, \Delta}$$

wobei X eine neue Variable ist.

ex-left

$$\frac{\Gamma, F(x/f(x_1,\ldots,x_n)) => \Delta}{\Gamma, \exists x F => \Delta}$$

wobei f ein neues Funktionssymbol ist und $x_1,\ldots,x_n$ alle freien Variablen in $\exists x F$.

Abbildung 5.5: Prädikatenlogische Sequenzenregeln

identity

$$\frac{}{\Gamma => s \doteq s, \Delta}$$

symmetry-right

$$\frac{\Gamma => s \doteq t, \Delta}{\Gamma => t \doteq s, \Delta}$$

symmetry-left

$$\frac{\Gamma, s \doteq t => \Delta}{\Gamma, t \doteq s => \Delta}$$

eq-subst-right

$$\frac{\Gamma, s \doteq t => F(t), \Delta}{\Gamma, s \doteq t => F(s), \Delta}$$

eq-subst-left

$$\frac{\Gamma, F(t), s \doteq t => \Delta}{\Gamma, F(s), s \doteq t => \Delta}$$

Abbildung 5.6: Sequenzenregeln für die Gleichheit

Wir nennen einen Beweisbaum *geschlossen* oder *vollständig* wenn er zusätzlich noch die folgende Bedingung erfüllt:

3. es gibt eine Substitution σ, so daß für die Markierung A jedes Knoten n, der keinen Nachfolgerknoten hat, $\sigma(A)$ ein Axiom ist. Dazu zählt auch das Gleichheitsaxiom.

Man beachte, daß zunächst $A \equiv p(s) => p(t)$ kein Axiom zu sein braucht. Ist σ aber ein Unifikator von s und t dann wird $\sigma(A) \equiv p(\sigma(s)) => p(\sigma(t))$ zu einem Axiom.

Definition 5.78
Für endliche Formelmengen $\Delta, \Gamma \subseteq For_\Sigma$ sagen wir, daß die Sequenz $\Gamma => \Delta$ in **S** *ableitbar* ist, genau dann, wenn ein geschlossener Beweisbaum mit Wurzelmarkierung $\Gamma => \Delta$ existiert.
In Symbolen schreiben wir dafür auch $\Gamma \vdash_S \Delta$.

Satz 5.79 (Korrektheit des Sequenzenkalküls)
Für endliche Formelmengen $\Delta, \Gamma \subseteq For_\Sigma$ ohne freie Variablen gilt:

$$\text{wenn } \Gamma \vdash_S \Delta, \text{ dann } \models \bigwedge \Gamma \to \bigvee \Delta$$

Beweis: Sei also $\mathcal{T}$ ein geschlossener Beweisbaum für die Formel A über der Voraussetzungmenge M. $\mathcal{T}$ hat die Form $\sigma(\mathcal{T}_0)$ für eine geeignete Substitution σ.

Der Beweis wird durch strukturelle Induktion über den Beweisbaum geführt.

Axiome: Im einfachsten Fall besteht $\mathcal{T}$ nur aus einer einzigen Sequenz, die dann ein Axiom sein muß. Es können die folgenden Fälle auftreten:

axiom $$\frac{}{\Gamma', F => F, \Delta'}$$

identity $$\frac{}{\Gamma', => s = s, \Delta'}$$

Der weitere Beweis gliedert sich in Fallunterscheidungen nach der ersten Regelanwendung in $\mathcal{T}_0$ und verläuft nach dem folgenden Muster. Sei

$$\frac{\Gamma_1 => \Delta_1 \quad \Gamma_2 => \Delta_2}{\Gamma => \Delta}$$

die erste Regelanwendung auf $\mathcal{T}_0$. Nach Induktionsvoraussetzung wissen wir daß

$$\bigwedge \sigma(\Gamma_1) \to \bigvee \sigma(\Delta_1) \text{ und } \bigwedge \sigma(\Gamma_2) \to \bigvee \sigma(\Delta_2)$$

allgemeingültig sind. Daraus müssen wir auf die Allgemeingültigkeit von $\bigwedge \sigma(\Gamma) \to \bigvee \sigma(\Delta)$ schließen. Die aussagenlogischen Regeln werden wie im Beweis von Satz 5.73 abgehandelt. Wir führen einige der restlichen Fälle ausführlich vor.

all-right Regel

$$\frac{\Gamma \Rightarrow F(x/f(x_1,\ldots,x_n)), \Delta'}{\Gamma \Rightarrow \forall x F, \Delta'}$$

wobei f ein neues Funktionssymbol ist und $x_1,\ldots,x_n$ alle freien Variablen in $\forall x F$. Um den Beweisbaum $\mathcal{T}$ zu erhalten müssen wir noch die Substitution σ auf jede Sequenz anwenden. Als Induktionsvoraussetzung haben wir dann die Allgemeingültigkeit von

$$\bigwedge \sigma(\Gamma) \to F(\sigma(x/f(x_1,\ldots,x_n))) \vee \bigvee \sigma(\Delta')$$

zur Verfügung. Daraus sollen wir die Allgemeingültigkeit von

$$\bigwedge \sigma(\Gamma) \to \sigma(\forall x F) \vee \bigvee \sigma(\Delta')$$

herleiten.

all-left Regel

$$\frac{\Gamma, \forall x F, F(x/X) \Rightarrow \Delta'}{\Gamma, \forall x F \Rightarrow \Delta'}$$

wobei X eine neue Variable ist. Die restlichen Fälle bleiben dem Leser als Übungsaufgabe überlassen.

■

Bemerkung 5.80

Wollen wir beweisen, daß eine Formel A logisch folgt aus einer Menge M von Voraussetzungen, so hatten wir bisher nur die Möglichkeit betrachtet, die Sequenz $M \Rightarrow A$ herzuleiten. Für unendliche Voraussetzungenmengen M ist das keine Lösung. Für diesen Fall führen wir die folgende Regel ein:

$$(einfügen_M): \quad \frac{Cl_\forall B, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$
$$\text{mit } B \in M$$

Wenn M endlich ist (z. B. $M = \{B_1, \ldots, B_n\}$), kann man einen Beweisbaum erstellen, in dessen Wurzel die Sequenz $Cl_\forall B_1, \ldots, Cl_\forall B_n => A$ steht und auf die Einfügeregel verzichten.

Satz 5.81 (Vollständigkeit des Sequenzenkalküls)
Es seien $M \subseteq For_\Sigma, A \in For_\Sigma$. Dann gilt

$$M \models A \Rightarrow M \vdash_S A$$

Der Beweis benutzt die Vollständigkeit des Hilbert-Kalküls ohne Gleichheit, indem Beweise im Hilbert-Kalkül zu Beweisen im Sequenzenkalkül transformiert werden. Die Transformation geschieht über die Länge des Hilbert-Kalkülbeweises. Es sei $A_1, ..., A_n$ mit $A_n = A$ ein Beweis von A über M.

a) $A_n \in M$: Die Sequenz $=> A_n$ wird durch die $(einfügen)$-Regel sofort bewiesen, siehe obige Bemerkung 5.80.

b) A_n Axiom des Hilbert-Kalküls: Die Ableitungen dieser Axiome im Sequenzenkalkül bleiben dem Leser als Übungsaufgabe überlassen.

c) A_n ist durch die Modus-Ponens-Anwendung auf A_i und A_j mit $i < n$ und $j < n$ entstanden, wobei $A_j = A_i \to A_n$ ist. Dann erhält man $=> A_n$ im Sequenzenkalkül durch Anwendung der $(Schnitt)$-Regel (mit A_i).

$$\frac{A_i => A_n \qquad => A_i, A_n}{=> A_n}$$

$=> A_i, A_n$ erhält man aus einem Beweis für A_i.
$A_i => A_n$ erhält man durch Anwendung der $(Schnitt)$-Regel (mit $A_i \to A_n$).

$$\frac{A_i \to A_n, A_i => A_n \qquad A_i => A_i \to A_n, A_n}{A_i => A_n}$$

$A_i => A_i \to A_n, A_n$ läßt sich durch einen Beweis für $=> A_i \to A_n$ beweisen und
$A_i \to A_n, A_i => A_n$ durch die $(\to links)$-Regel.

d) A_n ist durch die Anwendung der Gen-Regel auf A_i mit $i < n$ entstanden. Dann beweist man $=> A_n$ im Sequenzenkalkül durch Anwendung der $(\forall rechts)$-Regel und die verbleibende offene Prämisse analog zu dem Beweis von A_i, der nach Induktionsvorraussetzung existiert.

Bemerkung 5.82
Die (*Schnitt*)-Regel ist nicht wirklich notwendig, man kann sie auch weglassen, der Kalkül bleibt vollständig (siehe z.B. [Gal86]).

Ohne Schnitt lassen sich dann Tableaux- und Sequenzenbeweise einfach ineinander überführen, wenn man die unterschiedliche Variablensubstitution (γ-Regel bzw. ($\forall right$) und ($\exists left$)) entsprechend berücksichtigt.

5.6.4 Übungsaufgaben

Übungsaufgabe 5.6.1
Beweisen Sie in S_0:

$$((A \land B) \to C) \to (A \to (B \to C))$$

Übungsaufgabe 5.6.2
Zeigen Sie die folgenden Äquivalenzen

A ist eine allgemeingültige Formel	gdw	$=> A$
	gdw	ist eine allgemeingültige Sequenz
$\neg A$ ist eine allgemeingültige Formel	gdw	$A =>$ ist eine allgemeingültige Sequenz
A ist eine erfüllbare Formel	gdw	$=> A$ ist erfüllbare Sequenz
$\neg A$ ist eine erfüllbare Formel	gdw	$A =>$ ist erfüllbare Sequenz

Übungsaufgabe 5.6.3
Die in den Abbildungen 5.4 und 5.5 vorgestellten Regeln beschreiben den Sequenzelkalkül, der unter dem Namen *System G* bekannt ist, siehe etwa [Gal86, Chapter 5]. in der mathematischen Logik und Beweistheorie ist eine Variante des Kalküls, das *System LK*, siehe [Gal86, Chapter 6] wesentlich bekannter. Es entsteht aus unserem System, indem die Regeln $and - left$ und $or - right$ jeweils durch die folgenden beiden Regeln ersetzt werden

LK-and-left-1

$$\frac{\Gamma, F => \Delta}{\Gamma, F \land G => \Delta}$$

LK-or-right-1

$$\frac{\Gamma => F, \Delta}{\Gamma => F \lor G, \Delta}$$

LK-and-left-2

$$\frac{\Gamma, G => \Delta}{\Gamma, F \land G => \Delta}$$

LK-or-right-2

$$\frac{\Gamma => G, \Delta}{\Gamma => F \lor G, \Delta}$$

Zum Kalkül *LK* gehört außerdem noch eine sogenannte *Schnittregel*, auf die wir hier allerdings nicht eingehen wollen. In dem berühmten *Schnitteliminationssatz* kann man beweisen, daß jeder Beweis in *LK* unter Benutzung der Schnittregeln in einen Beweis ohne Schnittregel transformiert werden kann. Da der Kalkül *LK* nicht die Mengenschreibweise benutzt, braucht man noch sogenannte strukturelle Regeln. Für unsere bescheidenen Zwecke brauchen wir nur die Verdoppelungsregel:

$$\frac{\Gamma, F, F => \Delta}{\Gamma, F => \Delta}$$

Leiten Sie in dem Kalkül *LK* die folgenden Sequenzen ab

1 $F \wedge G => F \vee G$
2 $F \wedge G => F \wedge G$
3 $F \wedge \neg F =>$

5.7 Weitere Anmerkungen zur Prädikatenlogik erster Ordnung

5.7.1 Andere Notationsformen (muss noch ausgeführt werden)

- Wechsel zwischen Präfix-, Infix-, Postfix-, Suffix-Notation
- Attributnotation von Argumentstellen („Objekte“)
- Anlehnungen an die natürliche Sprache. Aktions- und Situationslogiken, Conceptual Graphs

5.7.2 Metaresultate

Eine Konsequenz der Korrektheit und Vollständigkeit von **H** ist der

Satz 5.83 (Kompaktheit der PL1)
Für beliebige $M \subseteq For_\Sigma$, $A \in For_\Sigma$ gilt:

$$M \models A \Leftrightarrow E \models A \text{ für eine endliche Teilmenge } E \text{ von } M.$$

Als Spezialfall und unmittelbares Korollar ergibt sich hieraus der

Satz 5.84 (Endlichkeitssatz)
Eine Menge $M \subseteq For_\Sigma$ hat genau dann ein Modell, wenn jede endliche Teilmenge von M ein Modell hat.

Satz 5.85 (Nichtcharakterisierbarkeit der Endlichkeit)
Es sei Σ eine Signatur der PL1. Dann gibt es keine Formelmenge $M \subseteq For_\Sigma$, so daß für jede Interpretation (D, I) über Σ gilt:

$$(D, I) \text{ ist Modell von } M \Leftrightarrow D \text{ ist endlich.}$$

Beweis

Angenommen, es gäbe so ein M. Es seien $x_0, x_1, \ldots$ paarweise verschiedene Variable, und für jedes $n \in I\!N$, $n \geq 1$,

$$A_n := \exists x_0 \exists x_1 \ldots \exists x_n (\bigwedge_{i,j \leq n, i<j} \neg x_i \doteq x_j).$$

Offensichtlich gilt für jede Interpretation (D, I):

$$(D, I) \text{ ist Modell von } A_n \Leftrightarrow \#D > n.$$

Wir betrachten

$$M \cup \{A_n | n \in I\!N, n \geq 1\}.$$

Jede endliche Teilmenge E dieser Menge hat ein Modell, nämlich, wenn m maximal ist mit $A_m \in E$, ein beliebiges(D, I) mit $\#D = m + 1$. Nach dem Endlichkeitssatz müßte es also ein Modell von $M \cup \{A_n | n \in I\!N, n \geq 1\}$ geben. Das aber kann nicht sein, denn

- M hat nach Voraussetzung genau die (D, I) mit endlichem D
- $\{A_n | n \in I\!N, n \geq 1\}$ hat genau die (D, I) mit unendlichem D

zu Modellen.

■

5.7.3 Sorten

Bei der Axiomatisierung komplizierter Bereiche und Begriffe bedient man sich mit Erfolg eines zusätzlichen syntaktischen Hilfsmittels: Man verwendet *Sorten*, um die zu betrachtenden Gegenstandsbereiche von vornherein als *aufgeteilt* oder *typisiert* oder eben *sortiert* aufzufassen. Das ist schon von einfachen Beispielen aus der Mathematik her geläufig. Etwa unterscheidet man in der Geometrie Punkte, Geraden und Ebenen voneinander, in der Numerik und in den Programmiersprachen werden verschiedene Zahlentypen unterschieden: natürliche Zahlen, ganze Zahlen, rationale, reelle Zahlen usw.; in der linearen Algebra schließlich werden Skalare, Vektoren, Matrizen voneinander abgegrenzt. In allen Fällen soll dabei eine intendierte Klassifikation des Gegenstandsbereichs schon syntaktisch vorweggenommen werden.

Die Ausdrucksstärke der Logik wird durch die Hinzunahme von Sorten nicht erhöht. Jede Formel über einer sortierten Signatur läßt sich transformieren in eine erfüllbarkeitsäquivalente Formel ohne Sorten aber mit zusätzlichen Prädikatsymbolen. Doch kommt man bei größeren Anwendungen wohl kaum ohne explizite Sorten aus. Sie zu verwenden,

- entspricht den Prinzipien des Entwurfs von Informatiksystemen;
- macht größere Spezifikationen lesbarer (oder überhaupt erst lesbar);

- gestattet ein *modulares* Vorgehen, insbesondere beim gleichzeitigen Arbeiten mit verschiedenen Spezifikationen.

Beispiel 5.86 (Lineare Algebra)
Signatur:

- Die Menge der Sorten bestehe aus S (für Skalar) und V (für Vektor),
- die Funktion $+$ habe die Sortierung VVV (Vektoraddition),
- $\cdot$ habe die Sortierung SVV (Skalarmultiplikation),
- Konstanten $0^V, a_1, a_2, a_3$ der Sorte V und
- eine Konstante 0^S.

Die lineare Unabhängigkeit der Vektoren a_1, a_2, a_3 wird dann durch die Formel:

$$\forall x_1^S \forall x_2^S \forall x_3^S (x_1^S \cdot a_1 + x_2^S \cdot a_2 + x_3^S \cdot a_3 = 0^V \rightarrow (x_1^S = 0^S \land x_2^S = 0^S \land x_3^S = 0^S))$$

ausgedrückt.

Die Eigenschaft von a_1, a_2, a_3 , ein Erzeugendensystem zu sein, wird beschrieben durch:

$$\forall x^V \exists x_1^S \exists x_2^S \exists x_3^S (x^V = x_1^S \cdot a_1 + x_2^S \cdot a_2 + x_3^S \cdot a_3).$$

Man kann noch ein übriges tun, um die prädikatenlogische Notation der in der Mathematik üblichen anzunähern. Man vereinbart, daß zwei unmittelbar hintereinander geschriebene Terme, wovon der erste die Sorte S und der zweite die Sorte V besitzt, als Skalarmultiplikation gelesen werden sollen und kann den Punkt $\cdot$ weglassen. Außerdem ist die Kennzeichnung von Variablen der beiden Sorten durch Superskripte umständlich. Man vereinbart griechische Buchstaben für Skalare und lateinische Buchstaben für Vektoren zu verwenden. Dann liest sich die Basiseigenschaft von a_1, a_2, a_3 als:

$$\forall \lambda_1 \forall \lambda_2 \forall \lambda_3 (\lambda_1 a_1 + \lambda_2 a_2 + \lambda_3 a_3 = 0^V \rightarrow (\lambda_1 = 0^S \land \lambda_2 = 0^S \land \lambda_3 = 0^S))$$

$$\forall x \exists \lambda_1 \exists \lambda_2 \exists \lambda_3 (x = \lambda_1 a_1 + \lambda_2 a_2 + \lambda_3 a_3)$$

.

Das Beispiel zur linearen Algebra zeigt noch ein weiteres Phänomen einer benutzerfreundlichen Darstellung von Formeln auf. Üblicherweise würde man in den obigen Formeln keinen notationellen Unterschied machen zwischen der skalaren und der vektoriellen 0, denn schließlich läßt sich aus dem Zusammenhang erkennen, was gemeint ist. Man sagt, daß 0 ein *überladenes Symbol* ist und wir eine *überladene Notation* (*overloading*) verwenden. In den Beispielformeln kommt nur die Vektoraddition vor, käme auch die Addition in dem Skalarenkörper vor, so müssten wir dieselbe Diskussion auch für + führen.

Definition 5.87
Eine *ordnungssortierte Signatur* Σ ist ein Tupel

$$\Sigma = (S_\Sigma, \leq, F_\Sigma, P_\Sigma, \alpha_\Sigma),$$

wo $S_\Sigma, \leq, F_\Sigma, P_\Sigma$ paarweise disjunkte Mengen von Symbolen sind und ferner gilt:

1. S_Σ besteht aus endlich vielen *Sortensymbolen*;
2. $\leq$ ist eine partielle Ordnung auf der Menge S_Σ;
3. F_Σ, P_Σ sind wie früher die Mengen der Funktions- bzw. Prädikatsymbole;
4. α_Σ ist jetzt nicht mehr einfach die Stelligkeitsfunktion, sondern gibt zu jedem Funktions- bzw. Prädikatsymbol seine Sortierung an:
   $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \to S_\Sigma^*$
   wobei $\alpha_\Sigma(f) \in S_\Sigma^+$, wenn $f \in F_\Sigma$.

$\alpha_\Sigma(f) = Z_1 \ldots Z_n Z'$ bedeutet: f ist ein Funktionssymbol für Funktionen, welche n-Tupeln von Elementen der Sorten $Z_1, \ldots, Z_n$ (respektive) ein Element der Sorte Z' zuordnen.

$\alpha_\Sigma(p) = Z_1, \ldots, Z_n$ bedeutet: p ist gedacht zur Notation von Mengen von n-Tupeln von Elementen respective der Sorten $Z_1, \ldots, Z_n$. Im Falle $n = 0$ ist (wie früher) p ein aussagenlogisches Atom.

Man nimmt ferner an, daß auch die Menge der Variablen sortiert ist: Zu jedem $Z \in S_\Sigma$ gibt es unendlich viele Variable der Sorte Z, und für verschiedene Sorten sind diese Variablenmengen disjunkt.

Definition 5.88 (Sortierte Interpretationen)
Eine *Interpretation* über der sortierten Signatur $\Sigma = (S_\Sigma, F_\Sigma, P_\Sigma, \alpha_\Sigma)$ ist ein Paar (D, I), wo gilt:

$D = \{D_Z \mid Z \in S_\Sigma\}$ ist eine Familie nichtleerer (den Sorten $Z \in S_\Sigma$ entsprechender) Mengen, so daß für Z_1, Z_2 mit $Z_1 \leq Z_2$ gilt $D_{Z_1} \subseteq D_{Z_2}$.

I ordnet jedem Funktions- und Prädikatsymbol eine *sortengerechte* Bedeutung zu:

$I(f) : D_{Z_1} \times \cdots \times D_{Z_n} \rightarrow D_{Z'}$ wenn $\alpha(f) = Z_1 \cdots Z_n Z'$
$I(p) \subseteq D_{Z_1} \times \cdots \times D_{Z_n}$ wenn $\alpha(p) = Z_1 \cdots Z_n$.

Definition 5.89 (Sortierte Terme)
Die Sortierung wird auf Terme übertragen:

- Ist x eine Variable der Sorte Z, dann ist x Term von der Sorte Z
- Sind $t_1, \ldots, t_n$ Terme der Sorten respective $Z_1, \ldots, Z_n$ und f ein Funktionssymbol mit $\alpha_\Sigma(f) = Z_1 \cdots Z_n Z'$, dann ist $f(t_1, \ldots, t_n)$ ein Term der Sorte Z'.

Entsprechend geschieht die Definitionen sortengerechter atomarer Formeln. Die Definition allgemeiner Formeln einer sortierten Logik erfolt nun wie in Definition 4.5.

Es lassen sich nun die weiteren, bisher behandelten Begriffe der Prädikatenlogik übertragen, und die entsprechenden Resultate gelten. Insbesondere müssen Substitutionen sortengerecht sein, d. h.: x und $\sigma(x)$ sind von derselben Sorte.

Sortierte Logiken mit einer Ordnungsrelation, wie wir sie einführt haben, nennt man *ordnungssortierte Logiken*. Sie ermöglicht eine flexiblere Termbildung und Unifikation, als das bei der „strikten“ Sortierung möglich wäre.

Das Arbeiten mit sortierten und ordnungssortierten Logiken hat sich in vielfacher Weise gewinnbringend auf das automatische Beweisen ausgewirkt.

5.7.4 Übungsaufgaben

Übungsaufgabe 5.7.1
Sei Σ die leere Signatur, d.h. das einzige Prädikatszeichen in Formeln $A \in For_\Sigma$ ist das Gleichheitszeichen $\doteq$.
Zeigen Sie: Es gibt kein $A \in For_\Sigma$, welches genau die Interpretationen (D, I) mit unendlichem D zu Modellen hat.

Übungsaufgabe 5.7.2
Es sei Σ eine Signatur der PL1, so daß P_Σ das zweistellige Prädikatsymbol $\equiv$ enthält. Dann gibt es kein $G \subseteq For_\Sigma$, so daß $\doteq$ in keinem $A \in G$ auftritt und für alle Interpretationen (D, I) über Σ gilt:

$$(D, I) \text{ ist Modell von } G \quad \Leftrightarrow \quad I(\equiv) = \{(d,d) | d \in D\}$$
$$\text{(d. h. } I(\equiv) \text{ ist die Gleichheit auf } D\text{).}$$

Das heißt, wenn man die Gleichheit nicht als logisches Zeichen zur Verfügung hat, dann kann man sie auch nicht definieren, gleichgültig wie man Σ wählt.

5.8 Axiomatisierung von Datentypen

Ein wichtiger Bestandteil fast jeder Anwendung deduktiver Methoden ist der Umgang mit häufig wiederkehrenden Datenstrukturen. Das betrift sowohl mathematische Strukturen, wie z.B. die natürlichen, ganzen und die rellen Zahlen oder (endliche) Mengen, als auch programmiersprachliche Stukturen wie die ganzen Zahlen (*integers*) in Java, Listen, Zeichenketten (*strings*) oder Felder (*arrays*).

5.8.1 Natürliche Zahlen

Die Axiomatisierung der natürlichen Zahlen von Guiseppe Peano zählt zu einer der bekanntesten Axiomatisierungen überhaupt. Er hat sie 1898 veröffentlicht [Pea89], wobei er auf der früheren Veröffentlichung von Richard Dedekind [Ded87] aufbauen konnte. Unter den vielen Variationen unter denen man die Peanoschen Axiome antrifft, haben wir uns für die folgende entschieden.

Definition 5.90 (Peano Arithmetik)
Die Signatur Σ_{PA} enthält Konstantensymbole $0, 1$ für die natürliche Zahl *Null* und *Eins* und zweistellige Funktionszeichen für $+, *$ für Addition und Multiplikation.

Die Peano Arithmetik PA wird durch die folgenden Axiome gegeben:

1. $\forall x(x + 1 \not\doteq 0)$
2. $\forall x \forall y(x + 1 \doteq y + 1 \rightarrow x \doteq y)$
3. $\forall x(x + 0 \doteq x)$
4. $\forall x \forall y(x + (y + 1) \doteq (x + y) + 1)$
5. $\forall x(x * 0 \doteq 0)$
6. $\forall x \forall y(x * (y + 1) \doteq (x * y) + x)$
7. Für jede Σ_{PA}-Formel $\quad (\phi(0) \land \forall y(\phi(y) \rightarrow \phi(y + 1))) \rightarrow \forall x(\phi)$

Das 7.te Peano Axiom heißt das *Induktionsaxiom* und ist in Wirklichkeit ein *Axiomenschema* und steht für unendliche viele Instanzen. Die Formeln $\phi(0)$, $\phi(y)$, $\phi(s(y))$ entstehen, indem in ϕ die freien Vorkommen der Variable x

durch 0, y,$y+1$ ersetzt werden. Eventuell in der Gesamtformeln vorkommende freie Variable sind universell quantifiziert. Diese Quantoren wurden der einfacheren Lesbarkeit wegen weggelassen. Für die Formel $\phi = x + z \doteq z + x$ lautet z.B. die Instanziierung des Induktionsschemas voll ausgeschrieben:

$$\forall z((0{+}z \doteq z{+}0 \wedge \forall y(y{+}z \doteq z{+}y \rightarrow (y{+}1){+}z \doteq z{+}(y{+}1))) \rightarrow \forall x(x{+}z \doteq z{+}x))$$

Das Bestreben Peanos und seiner Zeitgenossen war einen mathematischen Gegenstand durch möglichst wenige Axiome möglichst vollständig zu erfassen. Im vorliegenden Fall sollten also alle uns bekannten Eigenschaften der natürlichen Zahlen sich aus den 6 Axiomen plus Induktionsschema herleiten lassen. Versuchen wir das doch einmal. Es sollte $1 \neq 0$ gelten. Die Axiome 1 und 3 bieten sich dafür an. Leider liefert die Instanziierung der Allquantoren mit 0 in Axiom 1 die Ungleichung $0 + 1 \neq 0$, und die mit 1 in Axiom 3 aber $1 + 0 \doteq 0$. Von der Kommunikativität der Addition steht aber nichts in den Axiomen. Man muß also zusätzlich $\forall x(x + 0 \doteq 0 + x)$ ableiten.

Lemma 5.91
Die folgenden Formeln sind aus PA ableitbar.

1. $\forall w(w + 0 \doteq 0 + w)$
2. $0 \neq 1$

Beweise Wir wollen fürs erste die nachfolgenden Beweise besonders ausführlich erklären.

ad 1 Hier ist Induktion nach w angesagt. Der Induktionsanfang ist $0 + 0 \doteq 0 + 0$ was nach Axiom 3 äquivalent ist zu der universellen Gleichheit $0 \doteq 0$.

Im Induktionsschritt können wir $w + 0 \doteq 0 + w$ annehmen und müssen $(w + 1) + 0 \doteq 0 + (w + 1)$ zeigen:

$$\begin{array}{rcll}
(w+1)+0 & \doteq & w+1 & \text{Axiom 3 von links nach rechts} \\
 & & & \text{mit } x \rightsquigarrow (w+1) \\
 & \doteq & (w+0)+1 & \text{Axiom 3 von rechts nach links} \\
 & & & \text{mit } x \rightsquigarrow w \\
 & \doteq & (0+w)+1 & \text{Induktionsvoraussetzung} \\
 & \doteq & 0+(w+1) & \text{Axiom 4 von rechts nach links} \\
 & & & \text{mit } x \rightsquigarrow 0, y \rightsquigarrow w
\end{array}$$

ad 2 Nach Axiom 3 gilt für die Instantiierung $x \rightsquigarrow 1$ die Gleichung $1 + 0 \doteq 1$. Mit Teil 1 des Lemmas folgt $0 + 1 \doteq 1$. Aus Axiom 1 folgt mit der Instanziierung $x \rightsquigarrow 0$ die Ungleichung $0 + 1 \neq 0$. Insgesamt also, wie gewünscht, $1 \neq 0$.

∎

Eine weitere fast selbstverständliche Eigenschaft der Addition ist die Assoziativität, die im nächsten Lemma bewiesen wird.

Lemma 5.92
Die Assoziativität der Addition

$$\forall u \forall v \forall w((u + v) + w \doteq u + (v + w))$$

ist aus PA ableitbar.

Beweis Wir führen Induktion nach der Variablen w. Im Induktionsanfang ist also $(u+v)+0 \doteq u+(v+0)$ zu zeigen. Wendet man auf der linken Seite Axiom 3 an mit x instantiiert zu $u+v$ und auf der rechten Seite ebenfalls Axiom 3 aber mit x instantiiert zu v so erhält man $u + v \doteq u + v$, wie gewünscht.

Im Induktionsschritt können wir $(u + v) + w \doteq u + (v + w)$ annehmen und müssen $(u + v) + (w + 1) \doteq u + (v + (w + 1))$ zeigen. Das geht so

$$\begin{array}{rcll}
(u+v)+(w+1) & \doteq & ((u+v)+w)+1 & \text{Axiom 4 von links nach rechts} \\
& & & \text{mit } x \rightsquigarrow (u+v), y \rightsquigarrow w \\
& \doteq & (u+(v+w))+1 & \text{Induktionsvoraussetzung} \\
& \doteq & (u+((v+w)+1) & \text{Axiom 4 von rechts nach links} \\
& & & \text{mit } x \rightsquigarrow u, y \rightsquigarrow v+w \\
& \doteq & (u+(v+(w+1)) & \text{Axiom 4 von rechts nach links} \\
& & & \text{mit } x \rightsquigarrow v, y \rightsquigarrow w
\end{array}$$

∎

Die gegebenen Beispielableitungen und die in den Übungsaufgaben folgenden dienen dazu dem Leser das Konzept der logischen Ableitung greifbar verständlich zu machen. Inbesondere die spezielle Beweistechnik der Induktion in den natürlichen Zahlen. Für praktische Zweckewird man versuchen vollautomatische oder interaktive Theorembeweiser einzusetzen. Das mathematische Ideal mit möglichst wenigen Axiomen auszukommen macht für Theorembeweiser keinen Sinn. Hier möchte man eine möglichst geschickte, nicht zu kleine Auswahl von Axiomen zur Verfügung haben. Für die Theorie der ganzen Zahlen wollte man z.B., auf jeden Fall die Assoziativität und Kommutativität unter den Axiomen haben. Die Existenz eines ökonomischen

Axiomensystems spielt dennoch ein Rolle, nämlich die folgende. Ein Theorembeweiser für die natürlichen Zahlen benutze das Axiomensystem Ax_{big}, das vieleicht aus mehreren Hundert Formeln besteht. Es ist absolut notwendig sicher zu stellen, daß Ax_{big} widerspruchsfrei ist. Ansonsten könnte man jede Formeln aus Ax_{big} ableiten und der Theorembeweiser wäre nutzlos. Jetzt bietet sich an einmal nachzuweisen, daß alle Formeln in Ax_{big} aus PA ableitbar sind. Die Widerspruchsfreiheit von Ax_{big} wird somit auf die von PA zurückgeführt. Das ist wesentlich übersichtlicher.

Ein viel wichtigere Frage ist, ob es nicht etwas Besseres gibt als eine Axiomatisierung in Logik erster Stufe. Algorithmen für die logische Ableitung sind schließlich äußerst komplex. Gibt es nicht einen einfacheren Algorithmus, der es erlaubt festzustellen, ob eine vorgegebene Formel in der Struktur $\mathcal{N} = (\mathbb{N}, +, *, 0, 1)$ der natürlichen Zahlen wahr ist oder falsch. Mit anderen Worten, ist dieses Problem entscheidbar? oder in rekursionstheoretischer Terminologie: Ist die Menge $Th(\mathcal{N})$ der in $\mathcal{N}$ wahren Formeln rekursiv? In [EFT07, Kapitel X] wird gezeigt, daß die Antwort nein ist. Die Situation ist sogar viel schlimmer, die Menge $Th(\mathcal{N})$ ist noch nicht einmal rekursiv aufzählbar.

Die Aussage, daß $Th(\mathcal{N})$ nicht axiomatisierbar ist, wird üblicherweise der 1.Gödelsche Unvollständigkeitssatz genannt. Er wurde zuerst in [Gö31] veröffentlicht. Der Beweis in [EFT07] ist aber wesentlich zugänglicher. Die Autoren zeigen, daß sich das notorische Halteproblem für `while` Programme auf die Entscheidbarkeit von $Th(\mathcal{N})$ reduzieren läßt. Der letzte Schritt, daß $Th(\mathcal{N})$ noch nicht einmal rekursiv aufzählbar ist, ist ebenfalls nicht übermäßig schwierig. Es gilt nämlich: Ist T eine vollständige, nicht rekursive Menge von Formeln der Prädikatenlogik erster Stufe, dann ist T nicht rekursiv aufzählbar. Dabei heißt T vollständig, wenn für jede Formel ϕ entweder $\phi \in T$ oder $\neg\phi \in T$. Der Gödelsche Beweisansatz hat aber den Vorteil, daß sich mit ihm auch die wesentlich allgemeinere Aussage des 2.Gödelsche Unvollständigkeitssatzes zeigen läßt.

Da $Th(\mathcal{N})$ nicht rekursiv aufzählbar ist, aber die Menge aller logischen Folgerungen aus dem Axiomensystem PA rekursiv auszählbar ist, folgt, daß es einen wahren Satz geben muß, der aus PA nicht abgeleitet werden kann. Die Axiomatisierung PA ist unvollständig.

Wir wollen einige Überlegungen zur Tragweite dieses Resultats anstellen. Gibt es ein Beispiel für einen wahren Satz, der aus PA nicht ableitbar ist? Der in [EFT07] gewählte Zugang ist zu abstrakt dazu. In [Gö31] wird tatsächlich ein Beispiel konstruiert, eine Kodierung einer Diagonalisierungsausage von der Form *ich bin nicht ableitbar*. In [PH77] wird gezeigt, daß eine gewisse wahre kombinatorische Eigenschaft aus der Familie der Ramsey Sätze aus PA nicht ableitbar ist. Nach heutiger Erkenntnis zeigt sich die Unvollständigkeit des Peanoschen Axiomensystems PA nur in sehr konstruierten Beispielen. Für konkrete Anwendungen, wie sie z.B. in der Programmverifikation auftreten, spielt sie keine Rolle. Hier fallen die praktischen Beschränkungen viel mehr ins Gewicht. Ein automatisches Beweissystem findet einen tatsächlich existierenden Beweis nicht immer, z.B. weil die eingestellte Suchstrategie in die Irre läuft, oder bei Induktionsbeweisen nicht die richtigte Induktionsbehauptung gefunden wird.

5.8.2 Übungsaufgaben

Übungsaufgabe 5.8.1
Zeigen Sie, daß die folgenden Formeln aus PA ableitbar sind

1. $\forall x(x + 1 \doteq 1 + x)$
2. $\forall x \forall y(x + y \doteq y + x)$

Neben den Axiomen aus Definition 5.90 können Sie natürlich auch die Formeln aus Lemma 5.92 benutzen.

Übungsaufgabe 5.8.2
Bisher haben wir die Ordnungsrelation auf den natürlichen Zahlen nicht betrachtet. Das wollen wir in dieser Übungsaufgabe nachholen. Dazu wird das Vokabular Σ_{PA} erweitert um das zweistellige Relationszeichen $\leq$. Der Deutlichkeit halber bezeichnen wir das neue Vokabular mit $\Sigma_{PA(\leq)}$. Zu den Axiomen PA kommt noch die Definition der neuen Relation hinzu

$$\forall x \forall y(x \leq y \leftrightarrow \exists z(x + z \doteq y))$$

Das erweiterte Axiomensystem soll mit $PA(\leq)$ bezeichnet werden. Zeigen Sie, daß die folgenden Formeln aus $PA(\leq)$ ableitbar sind.

1. $\forall x \forall y \forall z(x \leq y \land y \leq z \rightarrow x \leq z)$
2. $\forall x \forall y \forall u(x + u \doteq y + u \rightarrow x \doteq y)$
3. $\forall u \forall v(u + v \doteq 0 \rightarrow u \doteq 0 \land v \doteq 0)$
4. $\forall x \forall y(x \leq y \land y \leq x \rightarrow x \doteq y)$

Übungsaufgabe 5.8.3
Zeigen Sie, daß die folgenden Formeln aus PA ableitbar sind

1. $\forall x(0 * x \doteq 0)$
2. $\forall x(x * 1 \doteq x)$
3. $\forall x(1 * x \doteq x)$

Übungsaufgabe 5.8.4
Zeigen Sie, daß die folgenden Formeln aus PA ableitbar sind

1. $\forall x \forall y \forall z(x * (y + z) \doteq x * y + x * z)$
2. $\forall x \forall y \forall z((x * y) * z \doteq x * (y * z))$

Übungsaufgabe 5.8.5
Zeigen Sie, daß die folgenden Formeln aus PA ableitbar sind

1. $\forall x \forall y((x + 1) * y \doteq x * y + y)$
2. $\forall x \forall y(x * y \doteq y * x)$

5.9 Anwendung (Zusatzstoff)

5.9.1 Verifikation eines Schaltkreises

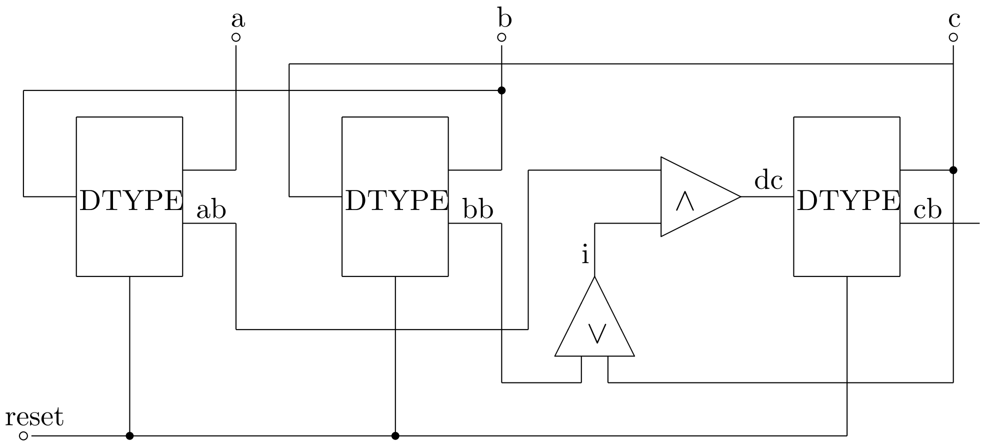


Abbildung 5.7: 3-Bit-Ring-Zähler

Abbildung 5.7 zeigt einen Schaltkreis für einen 3-Bit Ring Zähler. An den Ausgängen a, b, c sollen in aufeinanderfolgenden Taktzeitpunkten die Wertetripel 000, 001, 011, 111, 110, 100, 000 usw. auftreten, wenn kein reset erfolgt. Die abgebildete Schaltung ist aus der Arbeit [SGS+92] entnommen. Dort wird die Schaltung in der Hardwarebeschreibungssprache FUNNEL beschrieben und zur logischen Herleitung das auf ordnungssortierter Algebra basierende Beweissystem 2OBJ benutzt. Wir wollen in diesem Abschnitt vorführen, wie Darstellung und Inferenz direkt in der Prädikatenlogik aussehen.

Das Schaltelement $DTYPE(d, reset, q, qb)$ hat vier Ein/Ausgänge, deren Abhängigkeiten durch die folgenden Formeln definiert ist:

Der Quantor $\forall t$ soll dabei über alle Zeitpunkte laufen und $next(t)$ bezeichnet den nächsten Zeitpunkt nach t. In dem 3-Bit Zähler kommen die folgenden Instanzen dieses Schaltelements vor:

$DTYPE(b, reset, a, ab)$
$DTYPE(c, reset, a, bb)$
$DTYPE(dc, reset, c, cb)$

was zur folgenden prädikatenlogischen Beschreibung führt:

$\forall t(a(next(t)) \leftrightarrow b(t) \land \neg reset(t))$
$\forall t(ab(t) \leftrightarrow \neg(a(t)))$

$\forall t(b(next(t)) \leftrightarrow c(t) \wedge \neg reset(t))$
$\forall t(bb(t) \leftrightarrow \neg(b(t)))$
$\forall t(c(next(t)) \leftrightarrow dc(t) \wedge \neg reset(t))$
$\forall t(cb(t) \leftrightarrow \neg(c(t)))$
$\forall t(i(t) \leftrightarrow (bb(t) \vee c(t)))$
$\forall t(dc(t) \leftrightarrow (ab(t) \wedge i(t)))$
$\forall t(q(next(t)) \leftrightarrow d(t) \wedge \neg reset(t))$
$\forall t(qb(t) \leftrightarrow \neg(q(t)))$

Die Eigenschaften der Schaltung, die man durch formale Ableitung aus den Axiomen verifzieren möchte, sind:

$\forall t(reset(t) \rightarrow \neg a(next(t)) \wedge \neg b(next(t)) \wedge \neg c(next(t)))$
$\forall t(\neg reset(t) \wedge \neg a(t) \wedge \neg b(t) \wedge \neg c(t) \rightarrow \neg a(next(t)) \wedge \neg b(next(t)) \wedge c(next(t)))$
$\forall t(\neg reset(t) \wedge \neg a(t) \wedge \neg b(t) \wedge c(t) \rightarrow \neg a(next(t)) \wedge b(next(t)) \wedge c(next(t)))$
$\forall t(\neg reset(t) \wedge \neg a(t) \wedge b(t) \wedge c(t) \rightarrow a(next(t)) \wedge b(next(t)) \wedge c(next(t)))$
$\forall t(\neg reset(t) \wedge a(t) \wedge b(t) \wedge c(t) \rightarrow a(next(t)) \wedge b(next(t)) \wedge \neg c(next(t)))$
$\forall t(\neg reset(t) \wedge a(t) \wedge b(t) \wedge \neg c(t) \rightarrow a(next(t)) \wedge \neg b(next(t)) \wedge \neg c(next(t)))$
$\forall t(\neg reset(t) \wedge a(t) \wedge \neg b(t) \wedge \neg c(t) \rightarrow \neg a(next(t)) \wedge \neg b(next(t)) \wedge \neg c(next(t)))$

5.9.2 Anwendung in der Mathematik

Mathematische Probleme eignen sich besonders gut als Testprobleme für automatische Beweiser: sie sind schon in einer abstrakten Formulierung gegeben und können leicht und direkt durch prädikatenlogische Formeln kodiert werden. Die in anderen Situationen erforderliche Formalisierung des Anwendungsbereichs wurde durch die jahrtausende lange Entwicklungsgeschichte der Mathematik bereits vorweggenommen. Deswegen war es von den Anfängen an sehr beliebt automatischen Beweiser mit mathematischen Problemen zu füttern. Am Anfang konnten die Programme nur Beweise, die schon lange bekannt und meist trivial waren, nachvollziehen. Allmählich konnten Maschinen auch mathematische Fragen beantworten, auf die noch kein Mensch bisher eine Antwort gefunden hatte. Die Entgegnung, es handele sich dabei um Fragen, für die sich kein Mathematiker je ernsthaft interessiert habe, war in den Anfangsjahren nicht ganz unbegründet. Das änderte sich aber spätestens am 10. Dezember 1996 als den Lesern der *New York Times* auf der Titelseite ein ungewöhnliche Schlagzeile in die Augen sprang. Die mathematische Vermutung, daß jede Robbins Algebra eine Boolesche Algebra ist wurde von einer Maschine bewiesen. Über 60 Jahre lang hatten sich auch namhafte Mathematiker daran die Zähne ausgebissen, jetzt war die Lösung in einer gemeinsamen Anstrengung der beiden automatischen Beweiser Otter und EQP an den Argonne National Laboratories gefunden worden, [McC96]. Wir wollen uns mit diesem Problem und seine Lösung etwas ausführlicher beschäftigen.

Definition 5.93

Eine Algebra mit einer einstelligen Funktion $\neg$ und einer zweistelligen Funktion $\vee$ heißt eine *Robbins Algebra*, wenn sie die folgenden Axiome erfüllt

R1 $x \vee y = y \vee x$

R2 $(x \vee y) \vee z = x \vee (y \vee z)$

R3 $\neg[\neg(x \vee y) \vee \neg(x \vee \neg y)] = x$

Die Beschäftigung mit diesem Axiomensystem war motiviert durch vorangegangene Arbeiten von E. Huntington. Er hatten in seinen Arbeiten [Hun33b] und [Hun33a] nachgewiesen, daß die Formeln **H1 - H3** eine Axiomatisierung der Klasse der Booleschen Algebren liefern

H1 $x \vee y = y \vee x$

H2 $(x \vee y) \vee z = x \vee (y \vee z)$

H3 $\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y) = x$

Der Versuch Axiom **H3** durch das doch sehr ähnlich aussehende **R3** zu ersetzen wird Herbert Robbins zugeschrieben. Einem weiteren Publikum wurde die Robbinsche Vermutung, daß die Axiome **R1** bis **R3** ein Axiomensystem für die Booleschen Algebren liefern, durch die Veröffentlichung in [HMT71][p. 245].

Einige erste Schritte zum Beweis dieser Vermutung sind einfach. Jede Boolesche Algebra erfüllt offensichtlich die Robbinschen Axiome. Die interessante Frage ist also: ist jede Robbins Algebra eine Boolesche Algebra?

Lemma 5.94

In jeder Robbins Algebra gilt

$$\forall x \exists z (\neg z = x)$$

Beweis: Zu einem gegebenen x wähle man $z = \neg(x \vee x) \vee \neg(x \vee \neg x)$. Die Behauptung folgt jetzt mit Axiom **R3**.

■

Die zunächst erzielten Resultate, damit meine ich die vor der Lösung erzielten, waren typischerweise von der Form

> wenn eine Robbins Algebra eine zusätzliche Bedingung B erfüllt, dann ist sie schon eine Boolesche Algebra.

Hier sind zwei Beispiel nach diesem Muster.

Lemma 5.95

ist in einer Robbins Algebra die Formel

$$\forall x, y \neg x = \neg y \rightarrow x = y$$

wahr, dann ist sie eine Boolesche Algebra.

Beweis: Ersetzt man x durch $\neg x$ in **R3**, so erhält man

$$\neg[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y)] = \neg x$$

Aus der Voraussetzung des Lemmas folgt daraus

$$[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y)] = x$$

Das ist aber genau das Huntingtonsche Axiom **H3**.

■

Satz 5.96

Jede endliche Robbins Algebra ist eine Boolesche Algebra.

Beweis: Nach Lemma 5.94 ist die Funktion $f(x) = \neg x$ in jeder Robbins Algebra surjektiv. Ist die Algebra endlich, dann ist F sogar injektiv. Also ist die Voraussetzung von Lemma 5.95 erfüllt.

■

Das nächste Lemma ist eine Spur komlizierter zu beweisen. Er wurde zuerst von dem Theorembeweiser Otter gefunden. Die kommentierten Otterbeweise sind in [Win90], für den menschlichen Leser kaum nachvollziehbar, enthalten.

Lemma 5.97

Falls es in einer Robbins Algebra $\mathcal{R}$ ein Element a gibt, so daß

$$\forall x (a \vee x = x)$$

gilt, dann ist $\mathcal{R}$ eine Boolesche Algebra.

Beweis: Wir erwähnen Anwendungen der Kommutativität und Assoziativität nicht.

Wir zeigen zunächst, daß in $\mathcal{R}$ gilt

$$\forall x (\neg(\neg x \vee \neg\neg x) = a) \tag{5.1}$$

Wir substituieren dazu in dem Robbinschen Axiom R3 a für x und x für y, das ergibt

$$\neg(\neg(a \vee x) \vee \neg(\neg x \vee a)) = a$$

Durch zweimalige Anwendung der Voraussetzung entsteht direkt 5.1. Nächstes Teilziel ist die Gleichung

$$\forall x \neg\neg(x \vee \neg\neg x) = \neg\neg x \tag{5.2}$$

Um das einzusehen substituieren wir in R3 wieder x für y, aber diesmal $\neg\neg x$ für x:

$$\neg[\neg(\neg\neg x \vee x) \vee \neg(\neg\neg x \vee \neg x)] = \neg\neg x$$

Auf der linken Seite können wir die zweite Disjunktion nach 5.1 durch a ersetzen und erhalten

$$\neg[\neg(\neg\neg x \vee x) \vee a] = \neg\neg x$$

woraus mit der vorausgesetzten Eigenschaft von a jetzt sofort 5.2 folgt. Das nächste Etappenziel ist

$$\forall x \neg\neg\neg x = \neg x \tag{5.3}$$

Wir beginnen wieder mir R3 und nehmen diesmal die Ersetzungen $x \Rightarrow \neg x$ und $y \Rightarrow \neg\neg x$ vor:

$$\neg[\neg(\neg x \vee \neg\neg x) \vee \neg(\neg x \vee \neg\neg\neg x)] = \neg x$$

Wegen 5.1 können wir die erste Disjunktion durch a ersetzen:

$$\neg[a \vee \neg(\neg x \vee \neg\neg\neg x)] = \neg x$$

Nach Eigenschaft von a folgt

$$\neg\neg(\neg x \vee \neg\neg\neg x) = \neg x$$

Benutzen wir 5.2 mit $\neg x$ substituiert für x, so ergibt sich 5.3. Wir behaupten, daß dann auch

$$\forall x \neg\neg = x \tag{5.4}$$

Ist nämlich ein beliebiges x gegeben, so besorge man sich nach Lemma 5.94 ein y mit $\neg y = x$. Damit ist die rechte Seite von 5.4 jetzt $\neg\neg\neg y$ nach 5.3 gleich $\neg y$ und damit auch gleich x. Das Gesamtergebnis folgt jetzt unmittelbar aus Übungsaufgabe 5.5.5.

■

Lemma 5.98

Wenn eine der beiden folgenden Bedingungen in einer Robbins Algebra erfüllt sind, dann ist sie schon eine Boolesche Algebra.

1. $\exists x(\neg(x \vee x) = \neg x)$
2. $\exists x, y(\neg(x \vee y) = \neg y)$

Das Lemma wurde ebenfalls in [Win90] mit Unterstützung durch das automatische Beweissytem *Otter*. Der Beweis der Robbinschen Vermutung gelang, in dem der Gleichungsbeweiser EQP die zweite Bedingung des Lemmas aus den Axiomen herleitete.

Weitere Teilergebnisse wurden berichtet in [Win92].

Eine umfangreiche Liste mathematischer Theoreme, die am Argonne National Laboratory mit Hilfe automatischer Beweiser gezeigt wurden findet man auf einer speziellen Webseite.

5.9.3 Semantische Technologien

Das Ziel Semantischer Technologien (engl. semantic technologies) ist es elektronische Informationen, wie sie z.B. im *world wide web* aber auch in firmeneigenen Netzen (*intranets*) präsent sind, genauer zu beschreiben. Eine genauere Beschreibung soll das Auffinden von Informationen, die Zusammenführung von Informationen aus unterschiedlichen Quellen erleichtern und die Erschließung impliziten Wissens ermöglichen. Bekannt geworden ist dieses Thema unter dem Stichwort *semantic web*. Da aber inzwischen die Skepsis wächst, ob die anvisierten Ziele im weltweiten Netz erreichbar sind und sich zudem auch andere Anwendungsfelder auftun, benutzen wir lieber die weniger eingeschränkte Bezeichnung *semantische Technologien*.

Ein zentraler Bestandteil semantischer Technologien ist die 2004 vom W3C (World Wide Web Consortium) standardisierte Wissensrepräsentationssprache OWL. Das Akronym OWL steht dabei, etwas verdreht für *web ontology language*. Genauer besehen gibt es drei Varianten von OWL: OWL Full, OWL DL und OWL Lite. Wir konzentrieren uns hier auf OWL DL. Die umfangreichere Sprache OWL Full ist erstens nicht entscheidbar und wirft noch ungeklärte Probleme in ihrer Semantik auf. OWL DL kann interpretiert werden als eine entscheidbare Teilmenge der Prädikatenlogik erster Stufe. Bevor wir den gesamten Sprachumfang von OWL DL betrachten, schauen wir uns den einfachsten, aber schon typischen Kern davon an. In der Forschung wird diese Sprache mit $\mathcal{ALC}$ bezeichnet. Die Grundbegriffe dabei sind

1. *Konzepte*, darunter kann man sich in erster Näherung Mengen oder einstellige Prädikate vorstellen,
2. *Rollen*, das sind im Wesentlichen binäre Relationen und
3. *Konstanten*, die Objekte, oder wie man in diesem Zusammenhang sagt, Ressourcen bezeichnen.

Die volle Version von OWL DL, in der Forschung mit $\mathcal{SHOIQ}(D)$ bezeichnet, wird später erklärt.

Definition 5.99 ($\mathcal{ALC}$-Ausdrücke)

Zu einem vorgegebenen Vokabular $V = \mathbf{C} \cup \mathbf{R} \cup \mathbf{N}$, von Konzepten $\mathbf{C}$, Rollen $\mathbf{R}$ und Konstanten $\mathbf{N}$ definieren wir die Menge der $\mathcal{ALC}$-Konzeptausdrücke

1. jedes Konzeptsymbol C aus $\mathbf{C}$ ist ein $\mathcal{ALC}$-Konzeptausdruck,
2. $\top$ und $\bot$ sind $\mathcal{ALC}$-Konzeptausdrücke,
3. sind $C_1, \ldots, C_k$ $\mathcal{ALC}$-Konzeptausdrücke, dann sind auch $C_1 \sqcap \ldots \sqcap C_k$, $C_1 \sqcup \ldots \sqcup C_k$ und $\neg C_1$ $\mathcal{ALC}$-Konzeptausdrücke,
4. ist C ein $\mathcal{ALC}$-Konzeptausdruck, R ein Rollensymbol aus $\mathbf{R}$, dann sind auch $\exists R.C$ und $\forall R.C$ $\mathcal{ALC}$-Konzeptausdrücke.

$\mathcal{ALC}$-Aussagen werden durch die beiden folgenden Regeln definiert.

5. ist C ein Konzeptausdruck und $c \in \mathbf{N}$ eine Konstante, dann ist $C(c)$ eine $\mathcal{ALC}$-Aussage,

6. ist R ein Rollensymbol und sind $c_1, c_2 \in \mathbf{N}$ Konstanten, dann ist $R(c_1, c_2)$ eine $\mathcal{ALC}$-Aussage,

Beispiel 5.100
Für dieses Beispiel benutzen wir Begriffe, die in einem juristischen Informationssystems relevant sein könnten.

1. $\mathbf{C} = \{U, E, AE\}$.
   Die Buchstaben stehen als Abkürzungen:

U	für	Urteil
E	für	Einspruch
AE	für	abgelehnten Einspruch

2. $\mathbf{R} = \{R, S\}$.
   Die Buchstaben stehen als Abkürzungen:

R	für	ist Revision von
S	für	ist Einspruch zu

3. $\mathbf{N} = \{Az5.100, Az2020, E0417\}$.

Beispiel eines $\mathcal{ALC}$-Konzeptausdrucks

$$C_1 = U \sqcap \exists R.U \sqcap \forall S.AE$$

Intuitiv steht das Konzept C_1 für alle Urteile, die Urteile zu einem Revisionsverfahren sind, zu denen alle Einsprüche abgelehnt wurden.
Der $\mathcal{ALC}$-Ausdruck $C_1(Az5.100)$ soll sagen, daß es sich bei der durch $Az5.100$ bezeichneten Ressource um ein Urteil handelt, das die an C_1 gestellten Filterbedingungen erfüllt.
Der $\mathcal{ALC}$-Ausdruck $R(Az5.100, Az2020)$ soll sagen, daß Urteil $Az5.100$ ein Revisionsurteil zu $Az202$ ist und $S(Az5.100, E0417)$ besagt, daß die durch $E0417$ bezeichnete Ressource ein Einspruch gegen das Urteil $Az5.100$ ist.

Man sieht, daß in $\mathcal{ALC}$ auf die Benutzung von Variablen verzichtet wird. In dem Ausdruck $\exists R.C$ wird nicht gesagt *es gibt eine Rolle*. Das würde auch keinen Sinn machen, da R für einen im Vokabular festgelegten Rollenbezeichner steht. Der Ausdruck $\exists R.C$ kann vielmehr in natürlicher Sprache umschrieben werden als *es gibt einen Rollenfüller für die Rolle R der ein Element des Konzeptes C ist.*

Wir geben eine präzise Semantikdefinition für $\mathcal{ALC}$-Konzepte und -Ausdrücke, indem wir sie in prädikatenlogische Formeln übersetzen. Wir übernehmen dabei das Vokabular V unverändert. Nur fassen wir jetzt die Symbole in $\mathbf{C}$ als einstellige, die in $\mathbf{R}$ als zweistellige Prädikatszeichen auf, während die Zeichen in $\mathbf{N}$ direkt als Konstantensymbole auftreten.

Bei der Transformation in Prädikatenlogik wird die unterdrückte Variable explizit. Jedem $\mathcal{ALC}$-Konzeptausdruck C ordnen wir eine prädikatenlogische Formel $C^0(x)$ zu, welche genau die freie Variable x enthält.

C ($\mathcal{ALC}$ – Konzeptausdruck)	$C^0(x)$ (prädikatenlogische Formel)
$C \in \mathbf{C}$	$C(x)$
$\top$	$\mathbf{1}$
$\bot$	$\mathbf{0}$
$C_1 \sqcap C_2$	$C_1(x) \wedge C_2(x)$
$C_1 \sqcup C_2$	$C_1(x) \vee C_2(x)$
$\neg C$	$\neg C(x)$
$\exists R.C$	$\exists y(R(x,y) \wedge C^0(y/x))$
$\forall R.C$	$\forall y(R(x,y) \to C^0(y/x))$
A ($\mathcal{ALC}$ – Aussage)	A^0 (prädikatenlogische Formel)
$C(c)$	$C^0(c)$
$R(c,d)$	$R(c,d)$

Im Unterschied zum Vorgehen in der Prädikatenlogik wird bei den meisten Konzeptlogiken angenommen, daß verschiedene Konstantensymbole auch verschiedene Objekte bezeichnen. Zu der Übersetzung eines $\mathcal{ALC}$-Konzeptausdrucks C muß man also noch die Ungleichungen $\neg(c_1 = c_2)$ für je zwei verschiedene Konstantensymbole $c_1, c_2 \in N$ hinzunehmen.

Definition 5.101 ($\mathcal{SHOIQ}$-Ausdrücke)

Zu einem vorgegebenen Vokabular $V = \mathbf{C} \cup \mathbf{R} \cup \mathbf{N}$, von Konzepten $\mathbf{C}$, Rollen $\mathbf{R}$ und Konstanten $\mathbf{N}$ definieren wir:

1. die Menge der $\mathcal{SHOIQ}$-Konzeptausdrücke
   (a) jedes Konzeptsymbol C aus $\mathbf{C}$ ist ein $\mathcal{SHOIQ}$-Konzeptausdruck,
   (b) $\top$ und $\bot$ sind $\mathcal{SHOIQ}$-Konzeptausdrücke,
   (c) sind $C_1, \ldots, C_k$ $\mathcal{SHOIQ}$-Konzeptausdrücke, dann sind auch $C_1 \sqcap \ldots \sqcap C_k$, $C_1 \sqcup \ldots \sqcup C_k$ und $\neg C_1$ $\mathcal{SHOIQ}$-Konzeptausdrücke,
   (d) sind $c_1, \ldots, c_n$ in $\mathbf{N}$, dann ist $\{c_1, \ldots, c_n\}$ ein $\mathcal{SHOIQ}$-Konzeptausdruck,

(e) ist C ein $\mathcal{SHOIQ}$-Konzeptausdruck, R ein $\mathcal{SHOIQ}$-Rollenausdruck, dann sind auch $\exists R.C$ und $\forall R.C$ $\mathcal{SHOIQ}$-Konzeptausdrücke.

(f) ist C ein $\mathcal{SHOIQ}$-Konzeptausdruck, $R \in \mathbf{R}$ ein einfaches Rollensymbol, dann sind auch $\leq nR.C$ und $\geq nR.C$ $\mathcal{SHOIQ}$-Konzeptausdrücke.

2. die Menge der $\mathcal{SHOIQ}$-Rollenausdrücke

   (a) jedes $R \in \mathbf{R}$ ist ein $\mathcal{SHOIQ}$-Rollenausdruck,

   (b) für jedes $R \in \mathbf{R}$ ist R^- ein $\mathcal{SHOIQ}$-Rollenausdruck,

3. die Menge der $\mathcal{SHOIQ}$-Aussagen

   (a) für $c_1, c_2 \in \mathbf{N}$ sind $c_1 = c_2$ und $c_1 \neq c_2$ $\mathcal{SHOIQ}$-Aussagen,

   (b) sind $c_1, c_2 \in \mathbf{N}$ und ist R ein $\mathcal{SHOIQ}$-Rollenausdruck, dann ist $R(c_1, c_2)$ eine $\mathcal{SHOIQ}$-Aussage,

   (c) sind C_1, C_2 $\mathcal{SHOIQ}$-Konzeptausdrücke, dann ist $C_1 \sqsubseteq C_2$ eine $\mathcal{SHOIQ}$-Aussage,

   (d) für $R_1, R_2 \in \mathbf{R}$ ist $R_1 \sqsubseteq R_2$ eine $\mathcal{SHOIQ}$-Aussage,

   (e) für $R \in \mathbf{R}$ ist $trans(R)$ eine $\mathcal{SHOIQ}$-Aussage.

   (f) für $R \in \mathbf{R}$ sind $func(R)$ und $invfunc(R)$ $\mathcal{SHOIQ}$-Aussagne.

Dabei heißt ein Rollensymbol R *einfach*, wenn es nicht durch die Aussagen $trans(R)$ als transitiv erklärt wurde und das auch auch für keine Unterrolle $R' \sqsubseteq R$ der Fall ist.

Die Semantik von $\mathcal{SHOIQ}$ geben wir wieder durch Übersetzung in die Prädikatenlogik, wobei in der folgenden Tabelle nur die neu hinzugekommenen Konstrukte aufgeführt sind.

C ($\mathcal{SHOIQ}$ – Konzeptausdruck)	$C^0(x)$ (prädikatenlogische Formel)
$\{c_1, \ldots, c_n\}$	$x = c_1 \vee \ldots \vee x = c_n$
$\leq nR.C$	$\exists^{\leq n} y(R(x,y) \wedge C^0(y/x))$
$\geq nR.C$	$\exists^{\geq n} y(R(x,y) \wedge C^0(y/x))$
R ($\mathcal{SHOIQ}$ – Rollenausdruck)	$R^0(x)$ (prädikatenlogische Formel)
$R \in \mathbf{R}$	$R(x,y)$
R^-	$R(y,x)$
A ($\mathcal{ALC}$ – Aussage)	A^0 (prädikatenlogische Formel)
$C_1 \sqsubseteq C_2$	$\forall x(C_1^0(x) \rightarrow C_2^0(x))$
$R_1 \sqsubseteq R_2$	$\forall x \forall y(R_1^0(x,y) \rightarrow R_2^0(x,y))$
$trans(R)$	$\forall x \forall y \forall z(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$
$func(R)$	$\forall x \forall y \forall z(R(x,y) \wedge R(x,z) \rightarrow y = z)$
$invfunc(R))$	$\forall x \forall y \forall z(R(y,x) \wedge R(z,x) \rightarrow y = z)$

Dabei wurde die folgenden Abkürzungen benutzt:

$$\begin{array}{lcl} \exists^{\leq n} x F(x) & \leftrightarrow & \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \wedge \forall x (F(x) \rightarrow \bigvee_{1 \leq i \leq n} x = x_i)) \\ \exists^{\geq n} x F & \leftrightarrow & \exists x_1 \ldots \exists x_n (\bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \wedge F(x_i) \end{array}$$

Das bisher gesagte gibt zwar den logischen Kern des mit dem Schlagwort *semantic web* beschriebenen Gebiets wieder. Es bleibt aber eine Menge von Details ungesagt, z.B. eine genaue Beschreibung des aktuellen Zustands der Standardisierungen des *World Wide Web Consortiums* W3C, eine genauere Darstellung des Zusammenhangs der hier skizzierten Logiken mit XML und RDF (Resource Description Framework), Algorithmen für Entscheidungsverfahren für $\mathcal{ALC}$ und $\mathcal{SHOIQ}$ und schließlich ein Überblick über die verfügbaren Implementierungen. Hinweise zu diesen Themen finden man z.B. in den beiden Lehrbüchern [AvH04] und [HKRS08].

Auf einen Aspekt allerdings soll hier doch noch kurz eingegangen werden. Woher das Vokabular einer prädikatenlogischen Sprache kommt ist für die Behandlung der logischen Eigenschaften absolut unerheblich. In der Praxis dagegen und für Unterstützungswerkzeuge ist das ein sehr entscheidender Punkt. Für die semantischen Technologien ist die Festlegung eines Vokabulars eine der großen Herausforderungen, schließlich sollten die Annotationen im gesamten *World Wide Web* gelesen und vielleicht auch verstanden werden können. Die Beispiele von OWL Ausdrücken, die wir bisher gezeigt haben sind in *abstrakter Syntax* geschrieben. Für den elektronischen Austausch wird RDF Syntax benutzt, die wiederum von der XML Syntax abgeleitet ist. Daher beginnt ein Dokument, das eine OWL Ontologie enthält mit einer Deklaration der benutzten *Namensräume* (engl. name spaces), wie in XML durch die Angabe von URI *Uniform Resource Identifier*. Die Grundidee dahinter ist, Vokabulare im Netz zu veröffentlichen, vielleicht sogar mit einigen Erklärungen zur intendierten Bedeutung, und darauf zu hoffen, daß eine einheitliche Benutzung über einen möglichst großen Personenkreis sich einstellt.

Wir betrachten hier das beliebte Beispielvokabular *FOAF* (Friend of a Friend). Das FOAF Vokabular stellt Ausdrucksmittel bereit, um Dinge zu beschreiben die Menschen auf ihre *home pages* stellen. Genauere Informationen finden man unter `http://xmlns.com/foaf/spec/`. In abstrakter Notation betrachten wir die Aussagen

$(Person \sqcap \exists name.\{\texttt{Dan Brickley}\} \sqcap \exists homepage.\{hp1\})(p1)$

und die Klassendefinition

$ProfileDokument = \top \sqcap \exists maker.\{p1\} \sqcap \exists primaryTopic.\{p1\}$

Dabei sind *Person*, *ProfileDokument* Konzeptsymbole, *name*, *homepage*, *maker*, *primaryTopic* Rollenbezeichner und $p1$, $hp1$ und `Dan Brickley` Namen. Hier sind wir etwas großzügig. Genauer ist `Dan Brickley` ein Literal im Datentyp Zeichenkette und $hp1$ eine URI. Wie die beiden Ausdrücke in RDF Notation aussehen ist in Abbildung 5.8 zu sehen.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<foaf:Person rdf:nodeID="p1">
 <foaf:name>Dan Brickley</foaf:name>
 <foaf:homepage
    rdf:resource="http://rdfweb.org/people/danbri/"/>
</foaf:Person>

<foaf:PersonalProfileDocument rdf:about="">
   <foaf:maker rdf:nodeID="p1"/>
   <foaf:primaryTopic rdf:nodeID="p1"/>
</foaf:PersonalProfileDocument>

</rdf:RDF>
```

Abbildung 5.8: RDF Dokument zum FOAF Vokabular

In Zeile 3 wird das eben genannte Vokabular importiert. Die anderen vorkommenden Rollen, *nodeId*, *resource* und *about* stammen aus den in den in Zeile 2 und 4 deklarierten Namensräumen.

Es bleibt abzuwarten, ob oder mit welchen Änderungen dieses Vorgehen erfolgreich sein wird. Hier ein kritisches Zitat aus [DJK04]

> Allerdings zeigt die Erfahrung, daß es organisatorisch und politisch nahezu aussichtslos ist eine Übereinkunft hinsichtlich eines global gültigen Vokabulars zu etablieren, auf das alle Kommunikationspartner zurückgreifen. Um die Anfordung der losen Kopplung aus der Service-orientierten Architektur (SOA) nicht aufgeben zu müssen, wird daher ein Konzept benötigt, mit dem es möglich ist, eigenständig lokal definierte Vokabulare gegeneinander abgleichen zu können.

5.9.4 Übungsaufgaben

Übungsaufgabe 5.9.1
In dem Lehrbuch [TZ71, Seite 20] sind, unter vielen anderen, die folgenden Gleichungen aus der elementaren Mengenlehre als Übungsaufgaben aufgeführt

1	$A \cap (B \setminus C)$	$=$	$(A \cap B) \setminus (A \cap C)$
2	$A \cap (B \setminus C)$	$=$	$(A \cap B) \setminus C$
3	$A \cup (B \setminus C)$	$=$	$(A \cup B) \setminus (C \setminus A)$
4	$A \cup (B \setminus C)$	$=$	$(A \cup B) \setminus ((A \cap B) \setminus C)$

Hierbei steht $A \setminus B$ für die mengentheoretische Differenz von A und B, d.h. $A \setminus B = \{a \in A \mid a \notin B\}$.

Formalisieren Sie die vier mengentheoretische Gleichungen als Formeln der Logik erster Stufe und beweisen Sie deren Gültigkeit mit einem automatischen Theorembeweiser, z.B. mit dem KeY-Beweiser.

Achtung! eine der Gleichungen ist nicht korrekt.

Übungsaufgabe 5.9.2
Die folgenden Informationen stammen aus [GA02].
Die Boolesche Algebra ist die Theorie der Mengen. Mengen kann man als einstellige Relationen auffassen. Gibt es auch eine Theorie der zweistelligen, der binären Relationen? Der erste der dazu einen Beitrag veröffentlichte war de Morgan [dM64]. Einen ersten Höhepunkte erreichte die Theorie durch das 649 Seiten starke Buch von E.Schröder [Sch95]. Alfred Tarski stellte dann in [Tar41] den Versuch einer Axiomatisierung vor. Wir stellen die Axiome in der Form vor, wie sie in [CT51] veröffentlicht wurden. Dazu betrachten wir eine beliebige Menge M und definieren für binäre Relationen a,b auf M die folgenden Operationen

$(a \vee b)(x, y)$	$\Leftrightarrow$	$a(x, y)$ oder $b(x, y)$
$(a \wedge b)(x, y)$	$\Leftrightarrow$	$a(x, y)$ und $b(x, y)$
$(\neg a)(x, y)$	$\Leftrightarrow$	nicht $a(x, y)$
$(a; b)(x, y)$	$\Leftrightarrow$	es gibt z mit $a(x, z)$ und $b(z, y)$
$a^{-1}(x, y)$	$\Leftrightarrow$	$a(y, x)$
$id(x, y)$	$\Leftrightarrow$	$x = y$

für alle x, y gilt $1(x, y)$
für alle x, y gilt $\neg 0(x, y)$

Tarskis Axiome sind

1. $((M \times M), \vee, \wedge, \neg, 0, 1)$ ist eine Boolesche Algbra, d.h. es gelten die folgenden Axiome

$$\begin{array}{lcll}
a \vee (b \vee c) & = & (a \vee b) \vee c & \textit{associativity} \\
a \wedge (b \wedge c) & = & (a \wedge b) \wedge c & \textit{associativity} \\
a \vee b & = & b \vee a & \textit{commutativity} \\
a \wedge b & = & b \wedge a & \textit{commutativity} \\
a \vee (a \wedge b) & = & a & \textit{absorption} \\
a \wedge (a \vee b) & = & a & \textit{absorption} \\
a \vee (b \wedge c) & = & (a \vee b) \wedge (a \vee c) & \textit{distributivity} \\
a \wedge (b \vee c) & = & (a \wedge b) \vee (a \wedge c) & \textit{distributivity} \\
a \vee \neg a & = & 1 & \textit{complements} \\
a \wedge \neg a & = & 0 & \textit{complements}
\end{array}$$

2. $(a; (b; c)) = (a; b); c$ (associativity of ;)

3. $a; (b \vee c) = (a; b) \vee (a; c)$ (distributivity of ; over $\vee$)
   $(a \vee b)^{-1} = a^{-1} \vee b^{-1}$ (distributivity of $^{-1}$ over $\vee$)

4. $a; id = a$ (identity law)

5. $(a^{-1})^{-1} = a$ (first involution law)

6. $(a; b)^{-1} = b^{-1}; a^{-1}$ (second involution law)

7. $(a^{-1}; \neg(a; b)) \subseteq \neg b$ (the cyclic axiom)

Man beachte, daß auch 7 sich, wie alle anderen Axiome, als Gleichung schreiben lässt: $(a^{-1}; \neg(a; b)) \vee \neg b = \neg b$

Formulieren Sie die Definitionen der Operatoren $\vee$, $\wedge$, $\neg$, $+$, ; und $^{-1}$ und die Gleichungen in einer Logik erster Stufe und beweisen Sie, daß die Gleichungen aus den Operationsdefinitionen folgen.

Übungsaufgabe 5.9.3
Beweisen Sie mit Hilfe eines automatischen Theorembeweisers die folgenden Aussagen.

Notation und Definitionen wie in Aufgabe 5.9.2 und außerdem

$$a + b \quad = \quad (a \wedge \neg b) \vee (b \wedge \neg a)$$

1. $a + (b + c) = (a + b) + c$

2. $a \vee b = a + b + (a \wedge b)$

3. $a + a = 0$

4. $(a \vee b) + (a \wedge b) = a + b$

5. $1 + a = \neg a$

Übungsaufgabe 5.9.4
Das ist eine etwas kompliziertere Aufgabe, die wieder mit der Notation und den Definitionen aus Aufgabe 5.9.2 arbeitet. Beweisen Sie, daß eine Relation a genau dann funktional ist, wenn $a^{-1}; a \subseteq id$ gilt
Zur Definition einer funktionalen Relation siehe Def. 1.2 auf Seite 1.

Übungsaufgabe 5.9.5
Die folgende Definition der ganzzahligen Divison ist der Java Language Description entnommen. Geändert wurden die Teile, die sich auf die Endlichkeit der ganzen Zahlen in Java beziehen, so daß die folgende Definition für die üblichen mathematischen ganzen Zahlen $\mathbb{Z}$ zu lesen ist. Außerdem wird vereinbart, daß die Relation $jdiv(n, 0) = q$ für beliebige Zahlen $n, q \in \mathbb{Z}$ wahr ist. Diese Art mit der Division durch 0 umzugehen ist unter dem Namen *underspecification* bekannt. Dabei wird angenommen, daß $div(n, 0)$ einen Wert in $\mathbb{Z}$ hat über den aber nichts bekannt ist.

> Integer division rounds toward 0. That is, the quotient produced for integer operands n and d is an integer value q whose magnitude is as large as possible while satisfying $|d \times q| \leq |n|$; moreover, q is positive when $|n| \geq |d|$ and n and d have the same sign, but q is negative when $|n| \geq |d|$ and n and d have opposite signs.

1. Finden Sie eine Formel ϕ der Prädikatenlogik erster Stufe mit freien Variablen n,d,q so daß $\phi \leftrightarrow q = jdiv(n, d)$ in den ganzen Zahlen gilt.

2. Zeigen Sie:
   $\forall x, y(y * jdiv(x, y) = x \vee y * jdiv(x, y) = x + 1)$

3. $\forall x, y(y * jdiv(x, y) = x \leftrightarrow \exists z(x = 2 * z))$ (x ist gerade)

4. $\forall x, y(y * jdiv(x, y) = x + 1 \leftrightarrow \exists z(x = 2 * z + 1))$ (x ist ungerade)

Übungsaufgabe 5.9.6
Es gibt Alternativen zu der ganzzahligen Division, wie sie in der vorangegangen Aufgabe 5.9.5 betrachtet wurde. In Phyton wird *Runden nach* $-\infty$ benutzt, d.h. $div(n, d)$ wir berechnet, in dem zu nächst die n/d als Dezimalzahl berechnet wird und dann nach $-\infty$ gerundet wird. Aus $5/2 = 2,5$ wird $div(5, 2) = 2$ und aus $-5/2 = -2,5$ wird $div(-5, 2) = -3$.

1. Geben Sie eine Formel ϕ der Prädikatenlogik, so daß $\forall n \forall d \forall q(div(n, d) = q \leftrightarrow \phi$ gilt.

2. Drücken Sie $jdiv$ aus Aufgabe 5.9.5 durch div aus.

Kapitel 6

Gleichheitslogik

6.1 Einleitung

Die *Gleichungslogik* ist der Spezialfall der Prädikatenlogik erster Ordnung, bei dem nur Gleichungen, $s \doteq t$, zwischen Termen betrachtet werden. Das ist im striktesten Sinne zu verstehen: es gibt keine Ungleichungen, keine Disjunktion oder Konjunktion von Gleichungen. Variablen können auftreten und sind implizit universell quantifiziert. Eine Gleichungsmenge E gilt also in einer Struktur $\mathcal{M}$, in Zeichen $\mathcal{M} \models E$, wenn für jede Gleichung $s \doteq t \in E$ gilt $\mathcal{M} \models \forall \bar{x}(s \doteq t)$, wobei $\bar{x}$ alle in t und s vorkommenden Variablen sind.

Die zentrale Aufgabenstellung lautet:

Gegeben eine endliche Menge E von Gleichungen und eine Gleichung $s \doteq t$.

Wann gilt $E \models s \doteq t$?

Trotz der eingeschränkten Ausdruckfähigkeit bleibt auch die Gleichungslogik im allgemeinen unentscheidbar, wie das Beispiel des Wortproblems in Halbgruppen zeigt. Für das Rechnen in gegebenen, abstrakt spezifizierten Datenstrukturen ist aber diese Entscheidbarkeit von großer Bedeutung, so daß dem Auffinden hinreichender Bedingungen für sie große Aufmerksamkeit gewidmet wird. Als weiterführende Literatur verweisen wir auf die Bücher [BH92], [HK89], [Bü97] und [vL90, Chapters 6].

Die Vorgehensweise um zu zeigen, daß eine Gleichung $s \doteq t$ aus eine Menge von Gleichungen E folgt, ist intuitiv und allgemein bekannt durch den gymnasialen Mathematikunterricht. Man ersetzt in $s \doteq t$ die linke Seite einer Gleichung durch ihre rechte Seite, oder umgekehrt, bis auf beiden Seiten des Gleichheitszeichens derselbe Term steht. Da in den Gleichungen Variablen vorkommen können, ist die Beschreibung dieser einfachen Operation dann doch nicht so einfach. Wir betrachten ein Beispiel, in dem die Menge E aus einer Gleichung besteht $E = \{(x + y) * z = x * z + y * z\}$. Gilt

$$E \models u * [(a + c) + 2) * c] \doteq u * ((a + c) * c + 2 * c]?$$

1. Wahl eines Teilterms der linken Seite der zu beweisenden Gleichung, in diesem Fall $(a + c) + 2) * c$.

2. Der Teilterm *passt* zur linken oder rechten Seite einer Gleichung in E. Im Beispiel zeigt für die Subsitution $\sigma(x) = (a+c)$, $\sigma(y) = 2$, $\sigma(z) = c$, daß $\sigma(LS) = (a + c) + 2) * c$ gilt.

3. Ersetzung des gewählten Teilterms durch die jeweils andere Seite der Gleichung aus E nach Anwendung der Substitution σ. $\sigma(RS)$ ist im Beispiel $(a + c) * c + 2 * c$

4. Nach der Ersetzung wird die linke Seite der zu beweisenden Gleichung $u * ((a + c) * c + 2 * c]$ und somit gleich der rechten Seite.

Wir halten diese Operation, die wir mit $\rightarrow^1_E$ wollen, in der folgenden Definition fest.

Definition 6.1
Sei E eine Menge von Gleichungen und seien $s.\ t$ Terme.

1. $s \rightarrow^1_E t \quad \Leftrightarrow$ es gibt eine Gleichung $l \doteq r \in E$
   und eine Substitution σ, so daß gilt:
   $\sigma(l)$ ist Unterterm von s
   t entsteht aus s, indem $\sigma(l)$ ersetzt wird durch $\sigma(r)$
2. $\rightarrow^+_E$ ist die transitive Hülle von $\rightarrow^1_E$
3. $\rightarrow_E$ ist die transitive reflexive Hülle von $\rightarrow^1_E$
4. $\leftrightarrow_E$ ist die transitive, reflexive, symmetrische Hülle von $\rightarrow^1_E$

Es gilt der folgende Korrektheits- und Vollständigkeitssatz:

Satz 6.2
Für jedes Gleichungssystem E und zwei beliebige Terme s, t gilt

$$E \models s \doteq t \quad \Leftrightarrow \quad s \leftrightarrow_E t$$

Beweis siehe [DJ90, Seite 262]. Das Resultat ist als *Satz von Birkhoff* bekannt. Die Orginalarbeit ist [Bir35]. Siehe dazu auch die Übungsaufgaben 6.4.1 bis 6.4.3 .

∎

Die automatische Berechnung der Relation $\leftrightarrow_E$ ist äußerst aufwendig. Es ist unmöglich vorherzusehen, welche Gleichungsanwendungen am Ende zum Ziel führen werden. Die Vermeidung nutzloser Schleifen ist ein weiteres Problem. Das führt zu dem folgenden neuen Ansatz.

Die Gleichungsmenge E wird als ein *Termersetzungssystem* betrachtet. Um $E \models s \doteq t$ zu zeigen werden die Gleichungen aus E nur als Ersetzungregeln der linken Seiten durch die rechte betrachtet. Auf diese Weise wird s zu s' und t zu t' umgeformt in der Hoffnung irgendwann einmal bei $s' = t'$ anzukommen. Noch besser wäre es, wenn s' und t' eindeutig durch s bzw. t bestimmt sind, wenn s' eine eindeutige Normalform von s ist.

Das Konzept einer eindeutigen Normalform und die schrittweise Normalisierung eines symbolischen Ausdrucks ist so elementar, daß sie in vielen Zusammenhängen in unterschiedlichen Ausprägungen eine Rolle spielt, nicht nur in der Gleichungslogik. Das übergreifenden Konzept für alle diese Ausprägungen ist das der *Reduktionssysteme*. Wir werden zunächst in Abschnitt 6.2 einige Resultate vorstellen, die sich schon auf der abstrakten Ebene allgemeiner Reduktionssysteme beweisen lassen und danach in Abschnitt 6.3 zur Gleichungslogik zurückkehren.

6.2 Reduktionssysteme

Einführung

Definition 6.3 (Reduktionssysteme)
Ein **Reduktionssystem** $(D, \succ)$ besteht aus einer nichtleeren Menge D und einer beliebigen, binären Relation $\succ$ auf D.

Definition 6.4 (Notation)
Ist $(D, \succ)$ ein Reduktionssystem, dann benutzen wir die folgenden Bezeichnungen:

$\rightarrow$	die reflexive, transitive Hülle von $\succ$
$\stackrel{+}{\rightarrow}$	die transitive Hülle von $\succ$
$\leftrightarrow$	die reflexive, transitive, symmetrische Hülle von $\succ$

Beliebige Reduktionssysteme sind viel zu allgemein, als daß ihr Studium irgend etwas Interessantes hervorbringen könnte. Es sind die folgenden Definitionen, die Anlaß zu einer nicht trivialen Theorie geben.

Definition 6.5

1. Ein Reduktionssystem $(D, \succ)$ heißt **konfluent**, wenn für jedes Tripel $s, s_1, s_2 \in D$ mit $s \rightarrow s_1$, $s \rightarrow s_2$ ein $t \in D$ existiert mit $s_1 \rightarrow t$ und $s_2 \rightarrow t$.

2. $(D, \succ)$ heißt **lokal konfluent**, wenn für alle $s, s_1, s_2 \in D$ mit $s \succ s_1$, $s \succ s_2$ ein $t \in D$ mit $s_1 \rightarrow t$ und $s_2 \rightarrow t$ existiert.

3. $(D, \succ)$ heißt **noethersch** (oder **wohlfundiert** oder **terminierend**), wenn es keine unendliche Folge $s_0 \succ s_1 \ldots \succ s_i \succ \ldots$ gibt.

4. Ein konfluentes und noethersches Reduktionssystem heißt **kanonisch**.

5. Ein Element $s \in D$ heißt **irreduzibel** (oder eine **Normalform**) in $(D, \succ)$, wenn kein $t \in D$ existiert mit $s \succ t$.

6. Sei $s \in D$. Ein Element $s_0 \in D$ heißt eine **Normalform für s** in $(D, \succ)$, wenn s_0 irreduzibel ist und $s \rightarrow s_0$ gilt.

Satz 6.6
Sei $(D, \succ)$ ein kanonisches Reduktionssystem. Dann gilt:

1. Zu jedem $s \in D$ gibt es eine eindeutige Normalform. Diese bezeichnen wir mit $irr(s)$.

2. Für $s, t \in D$ gilt
$$s \leftrightarrow t \text{ gdw } irr(s) = irr(t)$$

3. $(D, \succ)$ sei berechenbar im folgenden Sinne: Es gibt einen Algorithmus, der zu jedem $t \in D$ ein t' mit $t \succ t'$ liefert, wenn ein solches existiert und andernfalls ausgibt „t ist irreduzibel", Dann ist die Relation $\leftrightarrow$ entscheidbar.

Beweis:

Zu 1.: Zunächst erledigen wir die Eindeutigkeitsfrage. Angenommen es gäbe für $s \in D$ zwei Normalformen s_1, s_2. Das heißt es gilt $s \to s_1$ und $s \to s_2$. Wegen der Konfluenz von $(D, \succ)$ gibt es $t \in D$ mit $s_1 \to t$ und $s_2 \to t$. Das widerspricht der Irreduzibilität von s_1, s_2. Um uns von der Existenz einer Normalform zu überzeugen, betrachten wir $s \in D$, setzen $s_0 = s$ und wählen ein s_{i+1} mit $s_i \succ s_{i+1}$, solange s_i nicht irreduzibel ist. Da $(D, \succ)$ noethersch ist, wird nach endlich vielen Schritten ein irreduzibles s_i erreicht.

Zu 2.: Die Implikation von rechts nach links ist trivial. Gelte jetzt $s \leftrightarrow t$. Nach Definition des reflexiven, transitiven, symmetrischen Abschlusses gibt es eine Folge $s = s_0, s_1, \dots, s_n = t$, so daß für alle $0 \leq i < n$ entweder $s_i \succ s_{i+1}$ oder $s_{i+1} \succ s_i$ gilt. Der Nachweis von $irr(s) = irr(t)$ geschieht durch Induktion über n. Der Induktionsanfang $n = 0$, d.h. $s = t$ ist trivial. Sei also die Behauptung für Folgen der Länge $n - 1$ schon bewiesen. Also gilt $irr(s_1) = irr(t)$. Im Fall $s_0 \succ s_1$ gilt offensichtlich $irr(s_0) = irr(s_1)$, und wir sind fertig. Falls $s_1 \succ s_0$ gilt, folgt aus der Konfluenz, daß ebenfalls $irr(s_0) = irr(s_1)$ gelten muß.

Zu 3.: Zu gegebenem s, t wird wie folgt entschieden, ob $s \leftrightarrow t$. Beginnend mit $s, s_0 := s$, liefert der vorausgesetzte Algorithmus Elemente s_i mit $s_0 \succ s_1 \succ s_2 \succ \dots$, bis hierbei ein irreduzibles s_m erreicht ist. Da $(D, \succ)$ noethersch ist, tritt das auf jeden Fall ein und wird durch „s_m ist irreduzibel" mitgeteilt, ferner gilt $s_m = irr(s)$. Entsprechend erhält man $irr(t)$ aus t. Nach (2) ist $s \leftrightarrow t$ genau dann, wenn $irr(s) = irr(t)$.

■

Eine wesentliche Stütze für die Theorie der Reduktionssysteme ist das nachfolgende Lemma, welches sagt, daß für noethersche Systeme aus der lokalen

Konfluenz auch schon die Konfluenz folgt. Wir besprechen zunächst das in seinem Beweis verwendete Beweisprinzip der „noetherschen Induktion“.

Lemma 6.7 (Noethersche Induktion)
Für ein noethersches Reduktionssystem $(D, \succ)$ gilt das folgende Beweisprinzip der *Noetherschen Induktion:*

Es sei $X \subseteq D$, so daß für alle $a \in D$ gilt

$$\{b | a \succ b\} \subseteq X \Rightarrow a \in X.$$

Dann ist $X = D$.

Beweis:
Angenommen $X \neq D$. Sei $a_0 \in D \backslash X$. Nach Annahme über X gilt $\{b | a_0 \succ b\} \not\subseteq X$.
Es gibt also ein a_1 mit

$$a_0 \succ a_1, a_1 \notin X.$$

Nach Annahme über X gilt wieder $\{b | a_1 \succ b\} \not\subseteq X$ und es gibt ein a_2 mit

$$a_1 \succ a_2, a_2 \notin X.$$

Fährt man in dieser Weise fort, so erhält man eine unendliche Folge $(a_i)_{i \in \mathbb{N}}$ mit $a_i \succ a_{i+1}$ für alle i. Das ist ein Widerspruch, denn $(D, \succ)$ war als noethersch vorausgesetzt.

■

Hiermit zeigen wir das

Lemma 6.8
Wenn $(D, \succ)$ ein noethersches und lokal konfluentes Reduktionssystem ist, dann ist $(D, \succ)$ konfluent, d. h. kanonisch.

Beweis:
Wir verwenden noethersche Induktion bezüglich der Menge Confl:=

$$\{s | \text{Für alle } s_1, s_2 \text{ mit } s \rightarrow s_1, s \rightarrow s_2 \text{ existiert ein } t \text{ mit } s_1 \rightarrow t, s_2 \rightarrow t\}$$

Dazu müssen wir also zeigen, daß für alle s gilt:

$$\{s' | s \succ s'\} \subseteq \text{Confl} \Rightarrow s \in \text{Confl}$$

Es seien s, s_1, s_2 gegeben mit $s \rightarrow s_1$, $s \rightarrow s_2$. Im Falle $s = s_1$ oder $s = s_2$ ist man fertig. (Etwa: $s_1 = s \rightarrow s_2$). Sei also $s \neq s_1$, $s \neq s_2$. Also

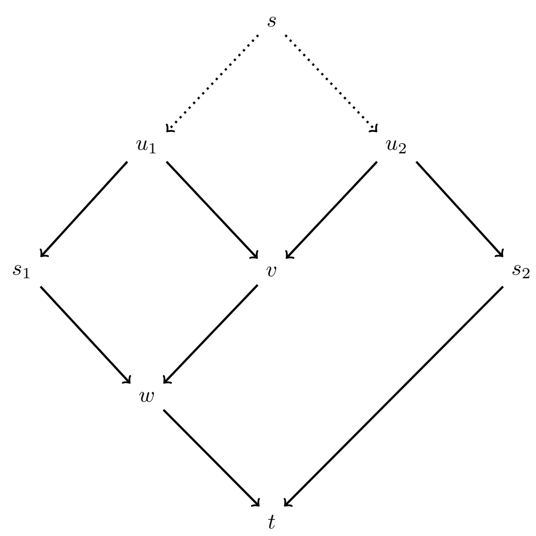


Abbildung 6.1: Von lokaler zu globaler Konfluenz

existieren u_1, u_2 mit $s \succ u_1 \to s_1$ und $s \succ u_2 \to s_2$. Wegen der lokalen Konfluenz von $(D, \succ)$ existiert ein v mit $u_1 \to v, u_2 \to v$. Nach Voraussetzung („Induktionsannahme“) liegt u_1 in Confl. Also gibt es ein w mit $s_1 \to w$ und $v \to w$. Entsprechend schließen wir aus der Induktionsannahme $u_2 \in$ Confl, daß ein Term t existiert mit $s_2 \to t$ und $w \to t$. Wir haben $s_1 \to t$ und $s_2 \to t$ und somit $s \in$ Confl, was zu beweisen war.

■

Beispiele

Polynomreduktion

Definition 6.9

- Ein **Potenzprodukt** in den Unbestimmten $X_1, \ldots, X_n$ über einem Körper K ist ein Ausdruck der Form
$$X_1^{e_1} * \ldots * X_n^{e_n}$$
mit natürlichen Zahlen e_j,
- Ein **Monom** in den Unbestimmten $X_1, \ldots, X_n$ über K ist ein Ausdruck der Form
$$c * pp,$$

wobei $c \neq 0$ ein Element aus K ist und pp ein Potenzprodukt.

- Ein **Polynom** in den Unbestimmten $X_1, \ldots, X_n$ über K ist ein Ausdruck der Form
$$m_1 + \ldots + m_k$$
mit Monomen m_i. Die Menge aller Polynome über K bildet mit den naheliegenden Produkt- und Summendefinition den Polynomring $K[X_1, \ldots, X_n]$.

Eine Ordnungsrelation $\prec$ auf der Menge aller Monome heißt **zulässig**, wenn für beliebige Monome m, m_1, m_2 gilt:

1. $1 \preceq m$ und
2. aus $m_1 \prec m_2$ folgt $m_1 * m \prec m_2 * m$

Die lexikographische Ordnung der Monome ist ein typisches Beispiel einer zulässigen Ordnungsrelation.

Beispiel 6.10 (Polynomreduktion)

Sei $B \subseteq K[X_1, \ldots, X_n]$. Die Reduktionsrelation $\succ_B$ auf $K[X_1, \ldots, X_n]$, die Polynomreduktion für B, wird wir folgt definiert.
$f \succ_B g$ gilt genau dann, wenn

- das größte Monom in f ist $m = c_1 * pp_1$ für $c_1 \in K$ und ein Potenzprodukt pp_1 und
- es gibt ein Polynom $h \in B$ mit größtem Monom $u = c_2 * pp_2$ mit $pp_1 = v * pp_2$ und
- $g = f - c_1 * c_2^{-1} * v * h$

Ein konkretes Beispiel für die Polynomreduktion

Sei $B = \{h_1 = xy^2 - x, h_2 = x - y^3\}$ $f = x^7y^2 + x^3y^2 - y + 1$.
Für $h = h_1$ haben wir in der Notation des Beispiels $c_1 = c_2 = 1$, $pp_1 = x^7y^2$, $pp_2 = xy^2$ und $v = x^6$.
Für $g = f - c_1 * c_2^{-1} * v * h = x^7 + x^3y^2 - y + 1$ gilt dann

$$f \succ_B g$$

Das Reduktionssystem $(K[X_1, \ldots, X_n], \succ_B)$ ist stets noethersch. Es muß nicht immer konfluent sein. Aber für jede Menge B gibt es eine Menge G, die dasselbe Ideal in dem Ring $K[X_1, \ldots, X_n]$ erzeugt wie B, so daß $\succ_G$ konfluent ist. G läßt sich aus B berechnen, z.B. mit dem Buchbergerschen Algorithmus.

Eine leicht zu lesende Einführung in dieses Gebiet findet man in [Buc85].

β-Reduktion

Beispiel 6.11 (β-Reduktion im λ-Kalkül)

Für zwei λ-Terme M, N ist die β-Reduktion $\succ_\beta$ definiert durch: $M \succ_\beta N$ genau dann, wenn

- ein Teiltermvorkommen $(\lambda x M_1)N_1$ in M gibt und
- N entsteht aus M, indem $(\lambda x M_1)N_1$ ersetzt wird $M_1[x \leftarrow N_1]$,
- wobei $M_1[x \leftarrow N_1]$ aus M_1 entsteht, indem jedes freie Vorkommen von x ersetzt wird durch N_1.

Die β-Reduktion auf der Menge aller λ -Terme ist konfluent, siehe z. B. [Bar84], Section 3.2. Die β-Reduktion ist nicht noethersch, so hat z.B. der Term $(\lambda x(xx))\lambda x(xx)$ keine Normalform.

Wortersetzung

Beispiel 6.12 (Semi-Thue-Systeme)
Sei R eine Menge von Paaren (r, s) von Wörtern über einem Alphabet Σ, d.h. $r, s \in \Sigma^*$. Die Relation $\succ_R$ auf der Menge Σ^* aller Wörter über Σ ist definiert durch:
$u \succ_R v$ gdw es gibt $(r, s) \in R$ und $x, y \in \Sigma^*$, so daß $u = xry$ und $v = xsy$.
(Σ, R) heißt ein **Semi-Thue-System** (string rewriting system).

Im Unterschied zu den jetzt zu besprechenden Termersetzungssystemen treten in Semi-Thue-Systemen keine Variablen auf und damit keine Substitutionen. Außerdem ist die interne Struktur von Wörtern wesentlich ärmer als die interne Struktur von Termen. Einen umfassenden Überblick über Semi-Thue-Systeme findet man in [Boo87].

6.3 Termersetzungssysteme

Wir kehren zurück zu unserem Anliegen, in einer Gleichungstheorie E die Ableitbarkeit einer Gleichung festzustellen.

Die allgemeinen Betrachtungen in 6.2 werden jetzt dazu spezialisiert, daß für eine vorgegebene endliche Menge E von Gleichungen

$$\succ \text{ die Relation } \overset{1}{\rightarrow}_E \ ist,$$

wie sie in Definition 6.1 eingeführt wurde.

Definition 6.13 (Termersetzungssysteme)
Ist E eine endliche Menge von Gleichungen über der Signatur Σ, dann nennen wir das Reduktionssystem $(\text{Term}_\Sigma, \overset{1}{\rightarrow}_E)$ ein *Termersetzungssystem*. Da dieses durch Σ und E eindeutig bestimmt ist, sprechen wir kürzer vom *Termersetzungssystem* (Σ, E).

Beispiel 6.14 (Boolesche Simplifikation E_{BS})
Ein einfaches Termersetzungssytem E_{BS} ist

$$\begin{array}{lcllcl} 0 \wedge x & = & 0 & 1 \wedge x & = & x \\ x \wedge 0 & = & 0 & x \wedge 1 & = & x \\ 0 \vee x & = & x & 1 \vee x & = & 1 \\ x \vee 0 & = & x & x \vee 1 & = & 1 \end{array}$$

in der Signatur $\Sigma = \{0, 1, \wedge, \vee\}$. Hierbei sind $\wedge$, $\vee$ (Boolesche) Funktionszeichen und nicht wie in anderem Zusammenhang aussagenlogische Operatoren.

Wir schreiben kurz E anstelle von E_{GBT}.
Für jeden variablenfreien Booleschen Term t gilt $t \rightarrow_E 0$ oder $t \rightarrow_E 1$.

Als ein weiteres nicht ganz so einfaches Beispiel betrachten wir die Axiome der Gruppentheorie aufgefasst als Termersetzungssystem:

Beispiel 6.15 (Gruppentheorie E_G)

$$\begin{array}{llcl} 1 & 0 + x & = & x \\ 2 & (x + y) + z & = & x + (y + z) \\ 3 & i(x) + x & = & 0 \end{array}$$

Ist das Termersetzungssystem E_G lokal konfluent?

Der Term $(i(x) + x) + z$ kann umgeschrieben werden einmal mit Gleichung 2 zu $i(x) + (x + z)$, mit Gleichung 3 aber auch zu $0 + z$ und weiter zu z mit

Gleichung 1. In einem konfluenten Termersetzungssystem müßte sich auch $i(x) + (x + z)$ zu z reduzieren lassen. Im vorliegenden System ist $i(x) + (x+z)$ eine Normalform und nicht weiter reduzierbar. E_G ist also nicht lokal konfluent.

Definition 6.16 (Kritische Paare)
Seien

$l_1 = r_1$, $l_2 = r_2$ zwei variablendisjunkte Gleichungen,
l'_2 ein Teilterm von l_1, der nicht nur aus einer Variablen besteht, und
μ ein allgemeinster Unifikator von l'_2 und l_2.

Dann heißt das Paar von Termen $\mu(r_1), \mu(l_1[r_2/l'_2])$ ein kritisches Paar der beiden Gleichung.
Dabei geht $(l_1[l'_2/r_2]$ aus l_1 hervor, indem ein Vorkommen des Teilterms l'_2 durch r_2 ersetzt wird.

Wir sagen, das kritische Paar $\mu(r_1), \mu(l_1[r_2/l'_2])$ entsteht aus der Überlagerung der Gleichung $l_1 = r_1$ mit der Gleichung $l_2 = r_2$.

Beispiel 6.17 (Kritisches Paare)
(1) Als erstes Beispiel eines kritischen Paares betrachten wir die beiden Gleichungen $(x + y) + z = x + (y + z)$ und $i(x) + x = 0$ aus Beispiel 6.15. Wir machen die beiden Gleichungen variablendisjunkt, indem wir die zweite Gleichung umbenennen zu $i(u) + u = 0$. Der allgemeinste Unifikator des nichttrivialen Teilterms $(x + y)$ der linken Seite der ersen Gleichung und $i(u) + u$ berechnet sich einfach zu $\mu(x) = i(u)$ und $\mu(y) = u$. Nach Definition erhalten wir das kritische Paar

$$(i(x) + (x + z)\ ,\ 0 + z)$$

(2) Das zweite Beispiel zeigt, daß ein kritisches Paar auch aus der Überlagerung einer Gleichung mit sich selbst entstehen kann. Das gilt, z.B. für die Gleichung $(x+y)+z = x+(y+z)$. Sei $(u+v)+w = u+(v+w)$ eine variablendisjunkte Kopie dieser Gleichung. Durch die Substitution $\mu(x) = (u + v)$, $\mu(y) = w$ wird $\mu((u+v)+w)$ ein Unterterm von $\mu((x+y)+z)$. Es entsteht das kritische Paar

$$((u + v) + (w + z)\ ,\ (u + (v + w)) + z)$$

Definition 6.18
Sei E die Menge der Gleichungen eines Termersetzungssystems.

1. Ein kritisches Paar von E ist ein kritisches Paar zweier Gleichungen $l_1 = r_1$, $l_2 = r_2$ aus E.

2. Ein kritisches Paar (l, r) heißt konfluent bezüglich E wenn es einen Term t gibt mit $l \rightarrow_E t$ und $r \rightarrow_E t$.

   Im Englischen nennt man konfluente Paare häufig *joinable pairs*.

Der folgende Satz ist von zentraler Bedeutung in der Theorie der Termersetzungssysteme. Der Nachweis der lokalen Konfluenz eines Termersetzungssystems wird damit entscheidbar.

Satz 6.19
Ein Termersetzungssystem E in dem alle kritischen Paare konfluent sind ist lokal konfluent.

Beweis Die Orginalarbeit ist [DP70].

Wir kommen zurück auf die beiden Beispiele 6.14 und 6.15.

Lemma 6.20
Das Termersetzungssystem E_{BS} ist kanonisch.

Beweis Da bei jedem Reduktionsschritt $t_1 \rightarrow^1_{E_{BS}} t_2$ der Term t_2 echt weniger Symbole enthält als t_1 ist E_{BS} terminierend. Die Teilterme aller linken Seiten aller Gleichungen in E_{BS} sind entweder eine Konstante, eine Variable oder der ganze Term. Man sieht schnell, daß überhaupt kein kritisches Paar existiert. Nach Satz 6.19 ist E_{BS} lokal konfluent und mit Lemma 6.8 dann auch konfluent.

∎

Im nachfolgenden Beispiel 6.22 kehren wir wieder zum Beispiel der Gruppentheorie zurück, von der wir schon festgestellt haben, daß sie nicht lokal konfluent ist. Das nächste Lemma zeigt, daß die Existenz eines nicht konfluenten kritischen Paares in einer Gleichungstheorie E noch nicht das Ende der Analyse sein muß.

Lemma 6.21
Sei (t, s) ein kritisches Paar eines Termersetzungssystem E. Dann stimmt die Gleichungstheorie von E überein mit der Gleichungstheorie von $E \cup \{t \doteq s\}$ und auch mit $E \cup \{s \doteq t\}$.

Das heißt, für jede Gleichung $t' \doteq s'$ gilt

$$E \models t' \doteq s' \quad \Leftrightarrow \quad E \cup \{t \doteq s\} \models t' \doteq s'$$

Beweis Zum Beweis genügt es zu zeigen, daß $E \models t \doteq s$ gilt. Nach Definition 6.16 gibt es variablendisjunkte Gleichungen $l_1 = r_1$, $l_2 = r_2$ in E, einen Teilterm l'_2 von l_1, der nicht nur aus einer Variablen besteht, und einen Unifikator μ von l'_2 und l_2, so daß $s = \mu(r_1)$ und $t = \mu(l_1[r_2/l'_2])$. Trivialerweise gilt $E \models \mu(l_1) \doteq \mu(l_1)$. Wegen $l_1 = r_1$ in E folgt $E \models \mu(l_1) \doteq \mu(r_1)$. Wegen $l_2 = r_2$ in E folgt $E \models \mu(l_1) \doteq \mu(l_1[r_2/l'_2])$. Wegen der Transitivität der Gleichheit folgt, wie gewünscht, $E \models \mu(r_1) \doteq \mu(l_1[r_2/l'_2])$, d.h. $E \models s \doteq t$.

■

Lemma 6.21 legt jetzt die folgende Vorgehensweise nahe: Wird ein nicht konfluentes kritisches Paar (t, s) für eine Termersetzungssystem E gefunden, so fährt man mit der Analyse von $E \cup \{t \doteq s\}$ oder $E \cup \{s \doteq t\}$ fort. Das ist immer noch dieselbe Gleichungstheorie aber das Paar (t, s) ist jetzt konfluent. Unglücklicherweise können durch die Hinzunahme von $t \doteq s$ neue nicht konfluente kritische Paare entstehen. Dann kann man die Prozedur wiederholen und hoffen, daß der ganz Prozess einmal aufhört. Dieses Verfahren ist unter dem Namen seiner Erfinder als *Knuth-Bendix Vervollständigung* bekannt. Wir demonstrieren die Knuth-Bendix Vervollständigung am Beispiel der Gruppentheorie, die eines der seltenen Beispiele ist, in dem der Prozess terminiert.

Beispiel 6.22 (Kritische Paare zur Gruppentheorie E_G)
Wir listen alle kritischen Paare des Reduktionssystems E_G. Wir vermerken welche Gleichung mit welcher anderen überlagert wird, geben die linke Seite nach Anwendung der unifizierenden Substitution an und in der dritten Spalte das kritische Paar:

1 in 2	$(0 + u) + z$	$(0 + (u + z)\ ,\ u + z)$
2 in 2	$((u + v) + w) + z$	$((u + v) + (w + z)\ ,\ (u + (v + w)) + z)$
3 in 2	$(i(u) + u) + z$	$(i(u) + (u + z)\ ,\ 0 + z)$

Um festzustellen welche dieser drei Paare konfluent sind reduzieren wir beide Seiten soweit wie möglich. Das führt zu den folgenden drei Paaren:

$$(u + z\ ,\ u + z)$$
$$(u + (v + (w + z))\ ,\ u + (v + (w + z)))$$
$$(i(u) + (u + z)\ ,\ z)$$

Man sieht, daß die ersten beiden Paare konfluent sind, das dritte aber nicht.

Sei E^1_G das Termersetzungssystem, das aus E_G entsteht durch Hinzunahme des neuen Axioms 4:

$$
\begin{array}{lll}
1 & 0+x & = & x \\
2 & (x+y)+z & = & x+(y+z) \\
3 & i(x)+x & = & 0 \\
4 & i(x)+(x+y) & = & y
\end{array}
$$

Die neuen kritischen Paare für E_G^1 sind

1 in 4	$i(0)+(0+u)$	$(i(0)+u\ ,\ u)$
2 in 4	$i(u+v)+((u+v)+w)$	$(i(u+v)+(u+(v+w))\ ,\ w)$
3 in 4	$i(i(u))+(i(u)+u)$	$(i(i(u))+0\ ,\ u)$
4 in 2	$(i(x)+(x+y))+w$	$(i(x)+((x+y)+w)\ ,\ y+w)$
4 in 4	$i(i(u))+(i(u)+(u+v))$	$(i(i(u))+v\ ,\ u+v)$

Reduktion der kritischen Paare ergibt:

$(i(0)+u\ ,\ u)$	$(i(0)+u\ ,\ u)$
$(i(u+v)+(u+(v+w))\ ,\ w)$	$(i(u+v)+(u+(v+w))\ ,\ w)$
$(i(i(u))+0\ ,\ u)$	$(i(i(u))+0\ ,\ u)$
$(i(x)+((x+y)+w)\ ,\ y+w)$	$(y+w\ ,\ y+w)$
$(i(i(u))+v\ ,\ u+v)$	$(i(i(u))+v\ ,\ u+v)$

Nur das vorletzte kritische Paar ist konfluent. Die restlichen Paare wollen wir wieder als neue Axiome hinzufügen. Dabei würde sowohl $i(i(x))+0=x$ als auch $i(i(x))+y=x+y$ hinzugefügt. Da die erste Reduktionsregel aus der zweiten folgt, wird nur die zweite hinzugenommen. So erhält man das Termersetzungssytem E_G^2.

$$
\begin{array}{lll}
1 & 0+x & = & x \\
2 & (x+y)+z & = & x+(y+z) \\
3 & i(x)+x & = & 0 \\
4 & i(x)+(x+y) & = & y \\
5 & i(0)+x & = & x \\
6 & i(x+y)+(x+(y+z)) & = & z \\
7 & i(i(x))+y & = & x+y
\end{array}
$$

Die neuen kritischen Paare für E_G^2 :

1 in 6	$i(0+u)+(0+(u+z))$	$(i(u)+(0+(u+z))\ ,\ z)$
1 in 6	$i(0+y)+(0+(y+z))$	$(i(0+y)+(y+z)\ ,\ z)$
3 in 5	$i(0)+0$	$(0\ ,\ 0)$
3 in 6	$i(i(y+z)+y)+(i(y+z)+(y+z))$	$(i(i(y+z)+y)+0\ ,\ z)$
3 in 7	$i(i(x))+i(x)$	$(x+i(x)\ ,\ 0)$
4 in 5	$i(0)+(0+v)$	$(0+v\ ,\ v)$
4 in 6	$i(i(u)+u)+(i(u)+(u+v))$	$(i(i(u)+u)+v\ ,\ v)$
4 in 7	$(i(i(x))+(i(x)+v)$	$(x+(i(x)+v)\ ,\ v)$
5 in 2	$(i(0)+x)+v$	$(i(0)+(x+v)\ ,\ x+v)$
5 in 4	$i(i(0))+(i(0)+x)$	$(i(i(0))+x\ ,\ x)$
5 in 6	$i(i(0)+y)+(i(0)+(y+z))$	$(i(i(0)+y)+(y+z)\ ,\ z)$
6 in 2	$(i(x+y)+(x+(y+z)))+w$	$(i(x+y)+((x+(y+z))+w)\ ,\ z+w)$
6 in 6	$i(i(u+v)+u)+(i(u+v)+(u+(v+w)))$	$(i(i(u+v)+u)+w\ ,\ v+w)$
7 in 2	$(i(i(x))+y)+z$	$(i(i(x))+(y+z)\ ,\ (x+y)+z)$
7 in 3	$i(i(x))+i(x)$	$(x+i(x)\ ,\ 0)$
7 in 4	$i(i(x))+(i(x)+u)$	$(x+(i(x)+u)\ ,\ u)$
7 in 6	ausgelassen	$(v+w\ ,\ v+w)$

Reduktion der kritischen Paare:

$(i(u)+(0+(u+z))\ ,\ z)$	$(z\ ,\ z)$
$(i(0+y)+(y+z)\ ,\ z)$	$(z\ ,\ z)$
$(0\ ,\ 0)$	$(0\ ,\ 0)$
$(i(i(y+z)+y)+0\ ,\ z)$	$(i(i(y+z)+y)+0\ ,\ z)$
$(x+i(x)\ ,\ 0)$	$(x+i(x)\ ,\ 0)$
$(0+v\ ,\ v)$	$(v\ ,\ v)$
$(i(i(u)+u)+v\ ,\ v)$	$(v\ ,\ v)$
$(x+(i(x)+v)\ ,\ v)$	$(x+(i(x)+v)\ ,\ v)$
$(i(0)+(x+v)\ ,\ x+v)$	$(x+v\ ,\ x+v)$
$(i(i(0))+x\ ,\ x)$	$(x\ ,\ x)$
$(i(i(0)+y)+(y+z)\ ,\ z)$	$(z\ ,\ z)$
$(i(x+y)+((x+(y+z))+w)\ ,\ z+w)$	$(z+w\ ,\ z+w)$
$(i(i(u+v)+u)+w\ ,\ v+w)$	$(i(i(u+v)+u)+w\ ,\ v+w)$
$(i(i(x))+(y+z)\ ,\ (x+y)+z)$	$(x+(y+z)\ ,\ x+(y+z)$
$(x+i(x)\ ,\ 0)$	$(x+i(x)\ ,\ 0)$
$(x+(i(x)+u)\ ,\ u)$	$(x+(i(x)+u)\ ,\ u)$
$(v+w\ ,\ v+w)$	$(v+w\ ,\ v+w)$

Die folgenden neuen Termersetzungsregel, resultierend aus den nicht konfluenten kritischen Paaren, stehen jetzt bereit um zu E_G^2 hinzugefügt zu werden:

$$\begin{array}{l} i(i(y+z)+y)+0=z \\ x+i(x)=0 \\ x+(i(x)+v)=v \\ i(i(u+v)+u)+w=v+w \end{array}$$

Unter den kritischen Paaren von E_G^3 kommt die Überlagerung $i(i(u+v)+u)+0$ von $i(i(y+z)+y)+0=z$ und $i(i(u+v)+u)+w=v+w$ vor, die zu dem nicht konfluenten kritischen Paar führt: $(v+0\ ,\ v)$. Nimmt man die neue Regel $(v+0=v)$ in E_G^4 dann sieht man, daß die Termersetzung $i(i(y+z)+y)+0=z$ überflüssig wird: sie kann aus $(v+0=v)$ und $i(i(u+v)+u)+w=v+w$ abgeleitet werden.

Fährt man in dem angefangen Prozeß weiter, was wir hier nicht mehr im Detail vorrechnen wollen: Konstruktion neuer nicht konfluenter Paare, Hinzunahme als neue Termersetzungsregeln, Weglassen von Regeln, die ableitbar geworden sind, so kommt man zu folgendem Endergebnis

$$\begin{array}{lcllcl}
\mathbf{E_{GK}} & & & & & \\
0+x & \to & x & (x+y)+z & \to & x+(y+z) \\
x+0 & \to & x & i(x)+(x+y) & \to & y \\
i(x)+x & \to & 0 & x+(i(x)+y) & \to & y \\
x+i(x) & \to & 0 & i(x+y) & \to & i(y)+i(x) \\
i(0) & \to & 0 & i(i(x)) & \to & x
\end{array}$$

Alle kritischen Paare des Termersetzungssystems E_{GK} sind konfluent. Nach Satz 6.19 ist E_{GK} also lokal konfluent.

Lemma 6.23
Das Termersetzungssystem E_{GK} ist kanonisch.

Beweis Wir nehmen die lokale Konfluenz als nachgewiesen an, was durch penibles Ausrechnen, wie es in Beispiel 6.22vorgemacht wurde, nachvollzogen werden kann. Es bleibt zu zeigen, daß E_{GK} terminierend ist. Dazu ordnen wir jedem Term t ein Gewicht $G(t)$ zu.

$$\begin{array}{lcll}
G(x) & = & 1 & x \text{ eine Variable} \\
G(0) & = & 1 & \\
G(x+y) & = & 2*G(x)+G(y) & \\
G(i(x)) & = & 5^{G(x)} &
\end{array}$$

Wir behaupten, daß für jede Termersetzungsregel $t_1 = t_2$ in E_{GK} gilt: $G(t_1) > G(t_2)$. Daraus folgt unmittelbar die Terminierung. Es gibt nur zwei Regel, für welche diese Ungleichung nicht offensichtlich ist:

$$\begin{array}{rcl} G(x+y)+z)) & = & 2*G(x+y)+G(z) \\ & = & 2*(2*G(x)+G(y))+G(z) \\ & = & 4*G(x)+2*G(y)+G(z) \\ & > & 2*G(x)+2*G(y)+G(z) \\ & = & 2*G(x)+G(y+z) \\ & = & G(x+(y+z)) \end{array}$$

Im nachfolgenden Argument machen wir Gebrauch von der Ungleichung $5^{a+b} > 5^a + 2*5^b$ für $a, b \geq 1$.

$$\begin{array}{rcl} G(i(x+y)) & = & 5^{G(x+y)} \\ & = & 5^{2*G(x)+G(y)} \\ & > & 5^{2*G(x)} + 2*5^{G(y)} \\ & > & 2*5^{G(y)} + 5^{G(x)} \\ & = & 2*G(i(y)) + G(i(x)) \\ & = & G(i(y)+i(x)) \end{array}$$

■

Als speziellen Fall von Satz 6.6 haben wir jetzt den

Satz 6.24
(Σ, E) sei ein kanonisches Termersetzungssystem.

1. Zu jedem Term t gibt es genau einen irreduziblen Term $irr(t)$ mit $t \rightarrow_E irr(t)$.
2. Für beliebige Terme s, t gilt:

$$E \models s \doteq t \Leftrightarrow irr(s) = irr(t).$$

3. Die Gültigkeit einer Gleichung in der Theorie von E ist entscheidbar.

Wenn unser vorgelegtes System (Σ, E) kanonisch ist, dann ist unser Problem also gelöst: Für eine gegebene Gleichung $s \doteq t$ läßt sich entscheiden, ob $E \models s \doteq t$. Leider ist das i.a. nicht der Fall; wir wissen ja, daß es unentscheidbare Gleichungstheorien gibt. Auch können wir i.a. einem Termersetzungssystem nicht ansehen, ob es noethersch ist bzw. ob es konfluent ist, beides ist i.a. unentscheidbar.

6.4 Übungsaufgaben

Übungsaufgabe 6.4.1
Zeigen Sie, daß für jede Menge E von Gleichungen und je zwei Terme s, t gilt

$$\text{Aus } s \leftrightarrow_E t \quad \text{folgt} \quad E \models s \doteq t$$

Das ist die einfache Richtung des Satzes 6.2.

Übungsaufgabe 6.4.2
Sei E eine Menge von Gleichungen in der Signatur Σ.
Zeigen Sie, daß die Relation $\leftrightarrow_E$ eine Kongruenzrelation auf der Menge $Term_\Sigma$ aller Σ-Terme ist, d.h.

1. $\leftrightarrow_E$ ist eine Äquivalenzrelation
2. Ist f ein n-stelliges Funktionszeichen in Σ und sind $s_1, \ldots s_n$, $t_1, \ldots t_n$ Σ-Terme mit $s_i \leftrightarrow_E t_i$ für alle $1 \leq i \leq n$ dann gilt auch $f(s_1, \ldots s_n) \leftrightarrow_E f(t_1, \ldots t_n)$.
3. Aus $s \leftrightarrow_E t$ folgt $\mu(s) \leftrightarrow_E \mu(t)$ für jede Substitution μ.
4. Sei μ_1, μ_2 Substitutionen, so daß für alle x gilt $\mu_1(x) \leftrightarrow_E \mu_2(x)$ dann gilt auch $\mu_1(s) \leftrightarrow_E \mu_2(s)$.

Übungsaufgabe 6.4.3
Diese Aufgabe ist dem Beweis der schwierigeren Richtung des Satzes von Birkhoff (Satz 6.2)gewidmet. Die zentrale Rolle spielt dabei die Σ-Struktur $\mathcal{F}_E = (F_E, I)$ zu einer Gleichungsmenge E. Wir beschreiben die Konstruktion von $\mathcal{F}_E$ ausführlich und stellen danach Fragen, die schrittweise zum Beweis führen werden. Das Universum F_E von $\mathcal{F}_E$ besteht aus den Kongruenzklassen modulo der Relation $\leftrightarrow_E$ auf der Menge aller Σ-Terme $Term_\Sigma$, siehe die vorangegangene Übungsaufgabe 6.4.2, d.h.

$$\begin{array}{rcl} F_E & = & \{[t]_E \mid t \in Term_\Sigma\} \\ {[t]_E} & = & \{s \in Term_\Sigma \mid t \leftrightarrow_E s\} \end{array}$$

Die Interpretation I für ein n-stelliges Funktionszeichen $f \in \Sigma$ wird definiert als

$$I(f)([t_1]_E, \ldots, [t_n]_E) = [f(t_1, \ldots, t_n)]_E$$

Die Unabhängigkeit dieser Definition von der Wahl der Repräsentanten der Äquivalenzklassen wird durch die Kongruenzeigenschaft von $\leftrightarrow_E$ sichergestellt, siehe Aufgabe 6.4.2(2).

Beweisen Sie die folgenden Aussagen:

1. Für jeden Term $t \in Term_\Sigma$ und Variablenbelegung β gilt

$$t^{\mathcal{F}_E,\beta} = [\sigma(t)]_E$$

   wobei die Substitution σ gegeben ist durch $\sigma(x) = t_{x,\beta}$, wobei $\beta(x) = [t_{x,\beta}]_E$. Nach Aufgabe 6.4.2(4) ist σ wohldefiniert.

2. $\mathcal{F}_E \models E$.

3. Gilt $E \models s \doteq t$ dann gilt auch $s \leftrightarrow_E t$.

Kapitel 7

Die Spezifikationssprache JML

In diesem Kapitel soll die Spezifikationssprache JML, Java Modeling Language, vorgestellt werden.

7.1 Historie und Motivation

JML ist eine Sprache zur formalen Spezifikation von Javaprogrammen. Sie wurde ins Leben gerufen von Gary Leavens, der bis heute eine zentrale Rolle in der JML *community* spielt. Die erste Veröffentlichung zu JML erschien 1999, [LBR99]. Die Sprache JML baut auf der heute nicht mehr benutzten Spezifikationssprache *LARCH* auf. Eine weitere Wurzel des Ansatzes ist das Konzept von Softwareverträgen, das im Englischen mit *design by contract* übersetzt wird, wie es z.B. in den Büchern [Mey91, Mey97] von Bertrand Meyer propagiert wird. Die logische Basis für das *design-by-contract* Paradigma liefert der nach Tony Hoare benannte *Hoare-Kalkül* [Hoa69, Hoa83, Hoa09].

Die umfangreichste Darstellung findet sich im Benutzerhandbuch [LPC+11]. Die Arbeit an JML fand und findet statt im Rahmen einer lose koordinierten internationalen Forschergemeinschaft. Eine Standardisierung gibt es nicht und ist für die nahe Zukunft nicht vorgesehen.

In der akademischen Forschung ist JML zur Zeit die am meisten benutzte formale Spezifikationssprache für Java, die durch eine Vielzahl von Werkzeugen auf den unterschiedlichsten Ebenen unterstützt wird, [BCC+05]. Aktuelle Informationen, elektronische Versionen der zitierten Papiere und Zugang zu vielen JML Werkzeugen findet man auf der Projektwebseite `http://www.cs.ucf.edu/~leavens/JML/`.

Das erste Ziel dieses Kapitel ist eine kurze Einführung in die logischen Grundlagen der formalen Softwareverifikation. Eine zweite Motivation besteht darin aufzuzeigen, wie die in den vorangegangene Kapiteln entwickelte Prädikatenlogik im praktischen Kontext eingesetzt werden kann. Wir werden sehen, daß JML eine syntaktische Variante dieser Logik für die Formulierung von Softwareverträgen benutzt.

7.2 Ein einführendes Beispiel

Wir beginnen mit einem einfachen Beispiel in Abbildung 7.1. Zur ersten Orientierung bemerken wir, daß JML Spezifikationen als spezielle Kommentare in Java Programme integriert werden. Der normale Java Compiler ignoriert diese Kommentare und JML Werkzeuge erkennen, daß in diesen Kommentare JML Eingaben stehen. Schon an diesem einfachen Beispiel können wir

Java + JML

```
public class PostInc{
    public PostInc rec;
    public int x,y;

    /*@  public invariant x>=0 && y>=0 &&
      @   rec.x>=0 && rec.y>=0;
      @*/

    /*@ public normal_behavior
      @ requires true;
      @ ensures rec.x == \old(rec.y) && rec.y == \old(rec.y)+1;
      @*/
    public void postinc() {
        rec.x = rec.y++;
    }
}
```

Java + JML

Abbildung 7.1: JML Annotationen in der Java Klasse `PostInc`

zwei Arten von Spezifikationen unterscheiden.

Die erste Kommentargruppe gibt eine *Invariante* an, die zweite ist ein Vertrag für die Methode, vor der sie unmittelbar steht, hier also für die Methode `postinc`.

Von einer Invarianten wird idealerweise erwartet, daß sie in jedem Zustand der Ausführung eines Programmes gilt. Wie man sich leicht vorstellen kann, ist das eine zu einschneidende Forderung, die außerdem noch den Mangel hat, daß nicht ganz klar ist, wie feinkörnig man verschiedene Programmzustände voneinander unterscheiden will. Verursacht die Anweisung `rec.y++` im Rumpf der Methode `postinc` einen einzigen Zustandsübergang oder gibt es einen oder mehrer Zwischenzustände? Letzteres würde die Java Language Specification tatsächlich nahelegen. Um diesen Schwierigkeiten aus dem Weg zu gehen, betrachten wir hier eine vereinfachte Invariantensemantik: eine Invariante gilt vor und nach jedem Methodenaufruf.

Nebenbei sei angemerkt, daß bei einer genaueren Analyse vor allem für komplexere Beispiele, das Konzept einer Invariante immer fragwürdiger wird. In der aktuellen Forschung zur Programmverifikation werden Invarianten gelegentlich schon durch andere Konzept, z.B. durch ein *ownership* Konzept, ersetzt.

Der zweite Spezifikationsbestandteil in Abb. 7.1, der Methodenvertrag für `postinc`, ist konzeptionell wesentlich unproblematischer. Wird die Methode in einem Zustand gestartet, in dem die Invariante gilt und die Vorbedingung, in JML durch das Schlüsselwort `requires` gekennzeichnet, dann gilt nach dem Ende der Methode die durch das Schlüsselwort `ensures` gekennzeichnete Nachbedingung und ebenfalls wieder die Invariante. Die englische Bezeichnung für Vor- und Nachbedingung ist *precondition* und *postcondition*. Die in Zeile 7 von Abb. 7.1 auftretende Deklaration `public normal_behavior` verlangt zusätzlich, daß die Methode normal terminiert, d.h. zunächst einmal, daß sie überhaupt terminiert, aber nicht durch das Werfen eines Ausnahmeobjekts (engl. *exception*).

Wenden wir uns nun den JML Ausdrücken selbst zu. Das Vokabular, aus dem JML Ausdrücke aufgebaut sind, stammt, mit wenigen Ausnahmen, aus dem Java Programm, zu dem die Ausdrücke gehören. In dem Ausdruck aus den Zeilen 5 und 6 von Abb. 7.1

```
x>=0 && y>=0 && rec.x>=0 && rec.y>=0;
```

treten die Attribute `x,y` und `rec` der Klasse `PostInc` auf. Die Relation `>=` und das Zahlenliteral `0` stammen aus dem Java Datentyp `int`. Schließlich ist `&&` das JML Äquivalent für die logische Konjunktion, die Boolesche Funktion *und*. Die entscheidende Beobachtung fehlt aber noch. Die Attribute `x,y` und `rec` sind nicht als statische Attribute deklariert. In Java Programmen können sie nur in der Form `t.x`, `t.y`, `t.rec` benutzt werden, wobei `t` ein Ausdruck vom Typ `PostInc` ist. Kommt trotzdem, etwa `x` vor, so ist das eine Abkkürzung für `this.x`. JML übernimmt diese Regelung. Voll ausgeschrieben liest sich die obige Invariante also als

```
this.x>=0 && this.y>=0 && this.rec.x>=0 && this.rec.y>=0;
```

In der Nachbedingung für die `postinc` Method, Zeile 11 in Abb.7.1, tritt die JML operation `\old` auf. Sie bewirkt, daß ihr Argument im Vorzustand des Methode ausgewertet wird. Sie kann nur in Nachbedingungen und nicht in Vorbedingungen oder Invarianten benutzt werden. Rein theoretisch könnte man auf die Operation `\old` verzichten und einen Vertrag wie folgt benutzen:

Java + JML

```
    @ requires oldrecy = rec.y;
```

```
    @ ensures rec.x == oldrecy && rec.y == oldrecy+1;
```
JAVA + JML

wobei `oldrecy` eine neue virtuelle Programmvariable ist. In den JML Werkzeugen wird eine solche oder ähnliche Auflösung des `\old` Konstrukts tatsächlich vorgenommen. Es ist aber unstrittig, daß die Benutzung von `\old` die Spezifikationsaufgabe wesentlich erleichtert.

Wir werfen noch einen Blick auf den Methodenrumpf von `postInc`, Zeile 14 in Abb.7.1. Was passiert, wenn die Methode in einem Zustand aufgerufen wird, in dem `self.rec == null` gilt? Ja, es wird eine *NullPointerException* ausgelöst. Somit würde die Methode ihren Vertrag nicht erfüllen, denn es wird normale Terminierung verlangt. Es ist schon alles in Ordnung. JML nimmt als Voreinstellung, als *default* an, daß alle vorkommenden Attribute und Parameter mit einem Objekttyp vom Nullobjekt verschieden sind.

JAVA + JML

```
1  public class PostInc{
2      public /*@ nullable @*/ PostInc rec;
3      public int x,y;
4
5      /*@ public invariant x>=0 && y>=0 &&
6        @   rec.x>=0 && rec.y>=0;
7        @*/
8
9      /*@ public normal_behavior
10       @ requires rec != null;
11       @ ensures rec.x == \old(rec.y) && rec.y == \old(rec.y)+1;
12       @*/
13     public void postinc() {
14         rec.x = rec.y++;
15     }
16 }
```
JAVA + JML

Abbildung 7.2: JML annotated JAVA class `PostInc`, version 2

Will man Attribute benutzen, die auch `null` als Wert annehmen können muß man das extra angeben durch den Modifikator `nullable`. Ein Beispiel dafür ist in Abb. 7.2 in Zeile 2 zu sehen. Jetzt muß man natürlich die Vorbedingung, Zeile 10, entsprechend verschärfen.

Die Benutzung von Sichtbarkeitsmodifikatoren wie `public` and `private` hat JML ebenfalls von JAVA übernommen. So darf, z.B. in einer als öffentlich, `public`, erklärten Annotation in der Klasse `A` kein privates Attribut einer anderen Klasse als `A` vorkommen. In einigen Fällen weicht JML von den JAVA Regeln ab und es gibt auch die Möglichkeit Sichtbarkeitsregelungen aus dem JAVA Code zu ignorieren. Wir gehen auf diesen Aspekt von JML nicht näher ein, siehe dazu [LPC+11, Section 2.4].

7.3 Schleifeninvarianten

Wir setzen die Erklärung von JML fort anhand des annotierten JAVA Programms in Abb. 7.3.

Die Methode `commonEntry` sucht in den durch die Parameter `l` und `r` gegebenen Grenzen einen Index, für den die beiden Felder `a1,a2` denselben Wert haben. Deswegen heißt die Klasse auch `SITA` für *search in two arrays*. Wir gehen das annotierte Programm Zeile für Zeile durch und setzen dabei die Erklärung von JML fort.

Die Vorbedingung für die Methode `commonEntry` in den Zeilen 5 und 6 in Abb. 7.3 setzt Einschänkungen an die Parameter `l` und `r`.

In Zeile 7 taucht eine neue Form von Spezifikationsklauseln auf, `assignable`. Hier wird spezifiziert, welche Werte die nachfolgende Methode höchstens ändern darf. In unserem Beispiel wird verlangt, daß `commonEntry` keine Änderungen bewirken darf. Methoden, die in diesem Sinne keine Seiteneffekt haben, nennt man *reine Methoden* (*pure methods*). Wir werden später, wenn wir auf detailiertere `assignable` Klauseln treffen, mehr zu diesem Thema zu sagen haben. Zwei Kommentare sollen aber schon hier gemacht werden.

Der erste Kommentar ist nur die Verbalisierung einer Selbstverständlichkeit. In der Methode eingeführte lokale Variablen, in unserem Beispiel die Variable `k`, spielen für die *assignable* Klausel keine Rolle. Der *Parser* würde ihr Auftreten auch zurückweisen. Betrachtet werden nur *von außen* beobachtbare Änderungen.

Es gibt zwei Möglichkeiten, beispielsweise die Aussage „*die Methode ändert den Wert von* `this.f` *nicht*“ zu interpretieren. Man kann erstens darunter verstehen, daß keine Zuweisung an `this.f` erfolgt. Auch `this.f` = `this.f` oder `this.f` = `this.f + 0` wären damit ausgeschlossen. Oder man erlaubt Zuweisungen an `this.f` besteht aber darauf, daß am Ende der Methode derselbe Wert wie zu Beginn wieder hergestellt ist. Für die Zwecke dieses Skripts halten wir uns an die zweite Version.

— Java + JML —

```
class SITA {
    public int[] a1, a2;

    /*@ public normal_behaviour
      @   requires 0 <= l && l < r &&
      @      r <= a1.length && r <= a2.length;
      @   assignable \nothing;
      @   ensures ( l <= \result && \result < r &&
      @                 a1[\result] ==  a2[\result])
      @           || \result == r;
      @   ensures (\forall int j; l <= j && j < \result;
      @                 a1[j] !=  a2[j] );
      @*/
    public int  commonEntry(int l, int r) {
        int k = l;

        /*@ loop_invariant
          @   l <= k && k <= r &&
          @   (\forall int i; l <= i && i < k; a1[i] != a2[i]);
          @
          @ assignable \nothing;
          @ decreases a1.length - k;
          @*/
        while(k < r) {
            if(a1[k] == a2[k]) {
                break;
            }
            k++;
        }
        return k;
    }
}
```

— Java + JML —

Abbildung 7.3: Search in two arrays

Die Nachbedingung für die Methode `commonEntry` findet sich in den Zeilen 8–12 in Abb. 7.3. Das zweifache Auftreten den Schlüsselwortes `ensures` wird dabei als logische Konjunktion der Einzeleinträge interpretiert. Die JML Variable `\result` steht für den Rückgabewert der Methode. Es wird verlangt, daß `\result` in dem Intervall $[l, r)$ liegt und `a1[\result] == a2[\result]` gilt. Alternativ darf `\result==r` gelten.

In den Zeilen 11 und 12 treffen wir auf das erste Beispiel einer quantifizierten Formel in JML. Die allgemeine Syntax ist

```
(\forall C x; B; R)      (\exists C x; B; R)
```

Wir nennen B die *Bereichseinschränkung* und R den Rumpf der quantifizierten Formel. In der prädikatenlogischen Notation aus Abschnitt 4.1 würde man diese Formeln in der Form

$$\forall C\ x(B \to R)$$
$$\exists C\ x(B \wedge R)$$

notieren.

JML*	Prädikatenlogik
`==`	$\doteq$
`&&`	$\wedge$
`||`	$\vee$
`!`	$\neg$
`==>`	$\to$
`<==>`	$\leftrightarrow$
`(\forall C x;e1;e2)`	$\forall x((x \neq null \wedge [e1]) \to [e2])$
`(\exists C x;e1;e2)`	$\exists x(x \neq null \wedge [e1] \wedge [e2])$

Dabei steht $[ei]$ für die prädikatenlogische Notation des JML Ausdrucks `ei`.

Tabelle 7.1: Vergleich der Notationen

Die Aufteilung in B und R ist aus logischer Sicht irrelevant, die Formeln `(\forall C x; B; R)` und `(\forall C x; true; B ==> R)` sind äquivalent. Viele Leute finden jedoch diese Aufteilung der quantifierten Formel hilfreich. Der erste Teil enthält Einschränkungen an den Bereich, über den die quantifizierte Variable läuft, wie in Zeile 12 in Abb. 7.3, der zweite Teil macht die eigentliche Aussage. Der Leser wird erraten haben, daß `==>` die JML Notation für die logische Implikation ist. Die Unterschiede zwischen der JML und der prädikatenlogischen Notation für die logischen Operatoren sind in Tabelle 7.1 zusammengefasst

Nach der Klärung der Syntax ist die Bedeuting des JML Ausdrucks in Zeile 11 und 12 von Abb. 7.3 lesbar: In dem Intervall [`l`,`\result`) gibt es keinen übereinstimmenden Eintrag in den Feldern `a1` und `a2`. Zusammen mit dem Teil der Nachbedingung in den Zeilen 9 und 10 folgt daraus, daß `\result` der kleinste Index mit übereinstimmenden Werten im Intervall [`l`,`r`) ist, bzw. `\result==r`, falls keine Index mit übereinstimmendem Wert in dem Intervall existiert.

Die Zeilen 17 bis 19 in Abb. 7.3 enthalten ein weiteres wichtiges Spezifikationskonzept: die Schleifeninvariante, auf Englisch *loop invariant*. Wie schon gewohnt bei JML bezieht sich die Annotation auf die Schleife **vor** der sie steht. Die Angabe einer Schleifeninvarianten SI verlangt, daß

1. (Anfangsfall)
   SI vor Eintritt in die Schleife erfüllt ist,

2. (Iterationsschritt)
   für jeden Programmzustand s_0, in dem SI und die Schleifenbedingung gilt, im Zustand s_1, der nach einmaligen Ausführen des Schleifenrumpfes beginnend mit s_0 erreicht wieder, wieder SI erfüllt ist.

3. (Anwendungsfall)
   nach Beendigung der Schleife reicht die Schleifeninvarianten SI zusammen mit den Bedingungen für die Schleifenterminierung aus für die Verifikation der Nachbedingung.

Sind diese drei Forderungen für den Programmkode in Abb. 7.3 erfüllt?

Anfangsfall Vor Beginn der Schleife gilt $k = l$. Setzt man diesen Wert in die Schleifeninvariante ein, so erhält man die Formel

```
l<=l && l<=r && (\forall int i; l<=i && i<l; a1[i] != a2[i]);
```

Der Allquantor läuft über den leeren Bereich, ist also trivialerweise wahr. Die Gleichung `l<=l` ist per Definition der kleiner-gleich Relation erfüllt und `l<=r` folgt aus der Vorbedingung `l<r`.

Iterationsschritt Das Programm befindet sich ein einem Zustand s_0. Intuitiv stellen wir uns darunter einen Zustand vor, der nach einer nicht bekannten Anzahl von Schleifendurchläufen erreicht wurde. Wir nehmen an, daß in s_0 die Schleifeninvariante und der Schleifbedingung wahr sind:

```
l <= k && k <= r &&
(\forall int i; l<=i && i<k; a1[i] != a2[i]) && l < k
```

Mit s_1 bezeichnen wir den Zustand der nach Ende des Schleifendurchlaufs erreicht wird. Um das Programm in Abb. 7.3 etwas interessanter zu machen, wurde eine `break` Anweisung in die Schleife eingebaut. Die Annahme, daß s_1 der Zustand nach Ende des Schleifendurchlaufs ist, umfaßt die Annahme, daß die `break` Anweisung nicht ausgeführt wurde. Die einzige Programmvariable, die während des Schleifendurchlaufs ihren Wert ändern kann, ist `k`, und zwar wird der Wert von `k` um genau 1 erhöht. Galt in s_0 `l<k`, so gilt in s_1 `l<=k`. In s_0 galt `(\forall int i; l <= i && i < k; a1[i] != a2[i])`. Diese Aussage gilt auch noch in s_1 wenn für den alten Wert von `k` im alten Zustand `a1[k] != a2[k]` galt. Das ist aber zutreffend, denn anderenfalls wäre die `break` Anweisung ausgeführt worden. Zusammengefasst haben wir gezeigt, daß in jedem Zustand, der durch wiederholte Aufführung des Schleifenrumpfes aus dem Zustand vor Schleifenbeginn erreicht werden kann, die Schleifeninvariante SI gilt.

Anwendungsfall Nach der Schleife ist in der Regel das Programm noch nicht zu Ende, und die Verifikationsaufgabe geht weiter. In dem Beispielprogramm in Abb. 7.3 ist noch die Anweisung `return k` auszuführen. Sei s_{se} der Programmzustand, der nach Beendigung der Schleife erreicht ist und s_e der Zustand nach Ausführung von `return k`. Somit ist s_e der Endzustand des Methodenaufrufs `commonEntry`. Wir müssen zwei Möglichkeiten unterscheiden:

1. Die Schleife terminiert, weil `k==r` erreicht wurde.
2. Es gilt `k<r` und die Schleife terminiert, weil `a1[k] == a2[k]` gilt.

Die folgende Argumentation berücksichtigt beide Fälle.

Wir wollen zeigen, daß im Zustand s_e die Nachbedingung aus den Zeilen 8–12 in Abb. 7.3 erfüllt ist. Da die Anweisung `return k` bereits ausgeführt ist, können wir darin `\result` durch `k` ersetzen und erhalten:

```
(1)   (l <= k && k < r && a1[k] ==  a2[k]) || k == r
(2)   (\forall int j; l <= j && j < k; a1[j] != a2[j])
```

Um diese Aussagen nachzuweisen, stehen uns zu Verfügung: die Invariante

```
(3)  l <= k && k <= r &&
(4)  (\forall int i; l <= i && i < k; a1[i] != a2[i])
```

und die Abbruchbedingungen

```
(4) k = r || a1[k] == a2[k]
```

Behauptung (2) wird unmittelbar durch (4) garantiert. Um (1) zu zeigen, nehmen wir an, daß die zweite Alternative `k == r` nicht gilt. Wir müssen jetzt argumentieren, daß der erste Disjunktionsteil von (1) gilt. Da wir aus (3) schon `k <= r` schon wissen, folgt aus der Annahme `k != r` schon `k < r`. Mit `k != r` folgt aus (4) noch `a1[k] ==  a2[k]` und damit die gewünschte Konklusion.

Die vorangegangenen Überlegungen setzten voraus, daß die Schleife in der Methode `commonEntry` terminiert. Die Terminierung ist eine zusätzliche Beweisverpflichtung. Aus der theoretischen Informatik wissen wir, daß die Terminierung von Programmen, selbst für einfache theoretische Programmiersprachen, ein unentscheidbares Problem ist. Aus der Praxis der Programmverifikation wissen wir, daß Terminierung von Schleifen schwierig ist. In JML gibt es deswegen eine Klausel `decreases`, in der der Programmierer angeben kann, warum er glaubt, daß die Schleife terminiert. Die verschiedenen JML Werkzeuge konnen dann überprüfen, ob diese Angaben stimmen. Die `decreases` Klausel besteht aus einem JML Ausdruck d vom Typ `int`, siehe Zeile 22 in Abb. 7.3. Diese Ausdruck d soll so gewählt sein, daß er stets einen Wert ≥ 0 hat und in jedem Schleifendurchlauf echt kleiner wird. Offensichtlich folgt aus diesen beiden Aussagen die Terminierung der Schleife.

Neben den Quantoren gibt es in JML noch einige andere Konstrukte die Variablen binden können. Es hat sich eingebürgert, solche Operatoren *verallgemeinerte Quantoren* (im Englischen *generalized quantifiers*) zu nennen.

Abb. 7.4 zeigt ein Beispiel eines Methodenvertrages, in dem der Summationsoperator vorkommt. Der geübte Leser sollte in der Lage sein, die Spezifikation für die Methode `sumAndMax` lesen zu können. Die Methoden hat keinen Rückgabewert, sie verändert die Attribute `sum` und `max` der Klasse `SumAndMax` und zwar so, daß nach Ende der Ausführung `sum` die Summe der Einträge in dem als Parameter übergebenen *array* `a` enthält und `max` den maximalen in `a` vorkommenden Wert.

Abb. 7.5 zeigt den Rumpf der Methode `SumAndMax` und die Schleifeninvariante.

Java + JML

```
class SumAndMax {
 int sum, max;
 /*@ public normal_behaviour
   @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
   @   assignable sum, max;
   @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= max);
   @   ensures (a.length > 0
   @           ==> (\exists int i; 0 <= i && i < a.length; max == a[i]));
   @   ensures sum == (\sum int i; 0 <= i && i < a.length; a[i]);
   @   ensures sum <= a.length * max;
   @*/
 void sumAndMax(int[] a) { ....}
}
```

Java + JML

Abbildung 7.4: Methodenkontrakt für `sumAndMax`

Abb. 7.6 zeigt die komplette Liste der verallgemeinerten Quantoren in JML. Die Syntax ist von derjenigen der Quantoren übernommen. Der Bereich über den die deklarierte Variable läuft wird durch das Bereichsprädikat `R` eingeschränkt; die Werte, die manipuliert werden sollen, werden durch den Ausdruck `t` repräsentiert. Die Bedeutung sollte intuitiv klar sein. Zur Auswertung etwa von `e = (\sum T x; R; t)` in einem gegebenen Programmzustand bestimmt man zunächst die Menge val_R der Objekte o in der Klasse T, so daß R wahr ist. Der Wert val_e von e ist dann die Summe $\Sigma\{val_{t,o} \mid o \in val_R\}$, wobei $val_{t,o}$ das Ergebnis der Auswertung von t ist, wenn die darin vorkommende Variable x mit o interpretiert wird. Für den Fall, daß $val_R = \emptyset$ ist, wird $val_e = 0$ vereinbart. Ist val_R unendlich, so ist val_e nicht definiert.

Hier einige Beispiele aus [LPC+11, Unterabschnitt 12.4.24.2]

```
(\sum     int i; 0 <= i && i < 5; i)   == 0 + 1 + 2 + 3 + 4
(\product int i; 0 < i  && i < 5; i)   == 1 * 2 * 3 * 4
(\max     int i; 0 <= i && i < 5; i)   == 4
(\min     int i; 0 <= i && i < 5; i-1) == -1
```

Java + JML

```
class SumAndMax {
  void sumAndMax(int[] a) {
    sum = 0; max = 0; int k = 0;
    /*@ loop_invariant
      @   0 <= k && k <= a.length
      @   && (\forall int i; 0 <= i && i < k; a[i] <= max)
      @   && (k == 0 ==> max == 0)
      @   && (k > 0 ==> (\exists int i; 0 <= i && i < k; max == a[i]))
      @   && sum == (\sum int i; 0<=i && i < k; a[i])
      @   && sum <= k * max;
      @  assignable sum, max;
      @  decreases a.length - k;
      @*/
    while(k < a.length) {
      if(max < a[k]) {
        max = a[k];
      }
      sum += a[k];
      k++;
    }
  }
}
```

Java + JML

Abbildung 7.5: Schleifeninvariante für `sumAndMax`

```
(\sum      T x; R ; t)
(\product  T x; R ; t)
(\max      T x; R ; t)
(\min      T x; R ; t)
```

Hierbei ist `R` ein JML Ausdruck vom Type `boolean`, in `t` ein JML Ausdruck von einem der in JML eingebauten numerischen Typen.

Abbildung 7.6: Verallgemeinerte Quantoren in JML

— Java + JML —

```
public class PostIncxx{
    public PostInc rec;
    public int x;

    /*@ public invariant x>=0 && rec.x>=0;
      @*/

    /*@ public normal_behavior
      @ requires true;
      @ ensures ???;
      @*/
    public void postinc() {
        rec.x = rec.x++;
    }
}
```

— Java + JML —

Abbildung 7.7: Variante `PostIncxx` der Klasse `PostInc`

7.4 Übungsaufgaben

Übungsaufgabe 7.4.1
Abb. 7.7 zeigt die Variante `PostIncxx` des Beispiels aus Abb. 7.1.

Geben Sie eine Nachbedingung an, welche die Java Semantik korrekt wiedergibt.

Übungsaufgabe 7.4.2
Wir wollen nicht ständig zwischen JML Syntax und der in Abschnitt 4.1 eingeführten prädikatenlogischen Notation hin und her wechseln. Eine gelegentliche Gegenüberstellung kann jedoch ganz instruktiv sein.
Schreiben Sie die Invariante aus Abb. 7.1 in prädikatenlogscher Notation. Überlegen Sie zuerst, was das Vokabular sein soll.

Übungsaufgabe 7.4.3
Schreiben Sie eine modifiziertes Java Programm, das keine `break` Anweisung benutzt, aber diesselbe Funktionalität hat wie die Methode `commonEntry` in Abb. 7.3. Passen Sie gegebenfalls die JML Annotationen an das neue Programm an.

Übungsaufgabe 7.4.4
Vervollständigen Sie die JML Annotation in dem folgenden Programm.

Hinweis: Die Methode `max` berechnet das Maximum der Werte `list[i]` für `0<=i && i<list.length`.

— Java + JML —

```java
public class Max0{
    public int[] list;

    /*@ public normal_behavior
      @ requires list.length > 0;
      @ ensures ??
      @ ensures ??
      @ assignable \nothing;
      @*/
    public int max() {
        int pos = 0;
        int m = list[0];

        /*@
          @ loop_invariant ??
          @ decreases ??
          @ assignable \nothing;
          @*/
        while (pos < list.length) {
            if (list[pos]>m) {
                m = list[pos];
            }
            pos++;
        }
        return m;
    }
}
```

— Java + JML —

Übungsaufgabe 7.4.5
Gegeben seien die beiden Java Klassen `Person` und `Firma`

— Java + JML —

```java
public class Person{
    public /*@ nullable @*/ Person vorgesetzer;
```

```
    public /*@ nullable @*/ Firma arbeitgeber;
    ...
}
```

— Java + JML —

— Java + JML —

```
public class Firma{
   public boolean angestellter(Person person) {
      ...
   }
}
```

— Java + JML —

Schreiben Sie eine Invariante in der Klasse `Person`, die besagt, daß für jede Person, die einen Vorgesetzen hat, der Arbeitgeber dieser Person und der Arbeitgeber ihres Vorgesetzten übereinstimmen.

Kapitel 8

Modale Aussagenlogik

8.1 Einführung

Die folgende Darstellung orientiert sich u.a. an den Büchern [Pop94, HR00, FM99],

Im Unterschied zur klassischen Logik, in der nur die Wahrheit einer Aussage von Bedeutung ist, spielt in der modalen Logik die Art und Weise, der Modus, in der eine Aussage wahr ist eine ausschlaggebende Rolle. Beispielsweise kann eine Aussage

- notwendigerweise wahr, zufälligerweise wahr
- heute, gestern oder morgen wahr sein
- geglaubt werden, zum Wissen einer Person gehören,
- vor/nach einer Aktion, vor/nach Ausführung eines Programms wahr sein.

Wir beginnen mit zwei Einführungsbeispielen.

Das Puzzle von den drei Weisen Das folgende Puzzle ist in vielen Varianten weit verbreitet:

> Drei Weisen werden Hüte aufgesetzt, jedem genau einen. Die Hüte sind entweder weiß oder schwarz, und jedem ist bekannt, daß mindestens ein schwarzer Hut mit dabei ist. Jeder Beteiligte sieht, welche Hüte die anderen beiden aufsitzen haben und soll erschließen, welchen Hut er aufsitzen hat, natürlich ohne in einen Spiegel zu schauen, den Hut abzunehmen oder ähnliches. Nach einer Weile sagt der erste Weise: „Ich weiß nicht, welchen Hut ich auf habe.“ Nach einer weiteren Pause des Nachdenkens sagt der zweite: „Ich weiß auch nicht, welchen Hut ich auf habe.“ „Dann“, sagt der dritte, „weiß ich, daß ich einen schwarzen Hut auf habe.“

In Abbildung 8.1 ist zunächst die Menge aller möglichen Welten zu sehen. Das Tripel (b,w,b) z.B. steht dabei für die Welt, in der der erste Weise einen schwarzen Hut auf hat (“b” für black), der zweite Weise einen weißen und der dritte wieder einen schwarzen Hut auf hat. Die Welt (w,w,w) kommt nicht vor, da sie durch die Spielregeln ausgeschlossen ist. Aber die Abbildung zeigt noch mehr. Sie zeigt welche Welten jeder der drei Weisen für möglich hält. Das hängt natürlich ab, von der Welt in der er sich befindet. Stellen wir uns

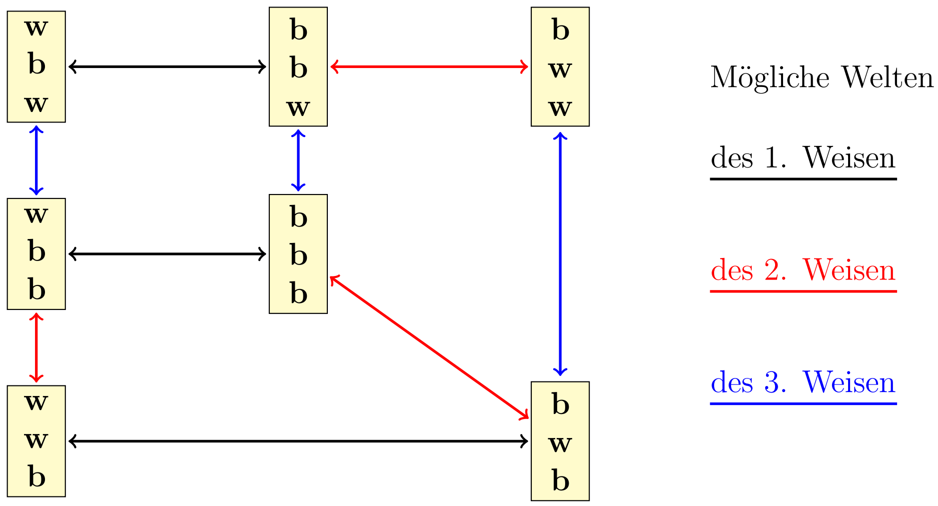


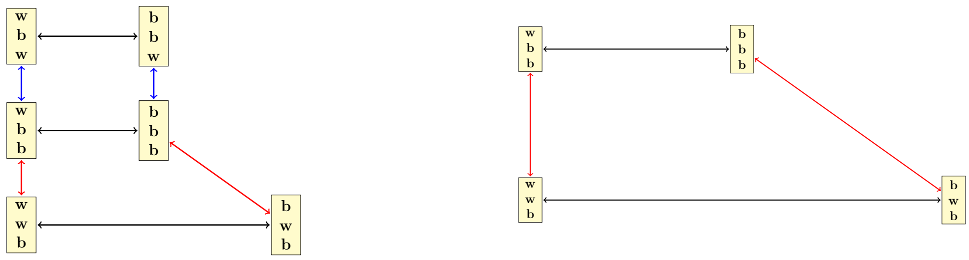
Abbildung 8.1: Mögliche Welten im Drei-Weisen-Beispiel

vor, der erste Weise befindet sich in der Welt (w,b,b). Er sieht zwei schwarze Hüte und hält zu Beginn die beiden Welten (w,b,b) und (b,b,b) für möglich. Das ist in Abbildung 8.1 durch Pfeile gekennzeichnet, wobei Pfeile, die in den eigenen Zustand zurückverweisen, der Übersichtlichkeit halber weggelassen wurden.

Nach dem Geständnis des ersten Weisen, er wisse nicht welche Farbe sein Hut hat, ist allen Beteiligten klar, daß die Welt (b,w,w) ausgeschlossen werden kann. Denn in dieser Welt sieht der erste Weise zwei weiße Hüte und weiss sofort, daß er den schwarzen auf haben muß. In der Abbildung 8.1 kann man diese Argumentation auch direkt ablesen. Aus Welt (b,w,w) führt kein schwarzer Pfeil, sie ist für den ersten Weise ohne Alternative. Die neue Situation ist in Abbildung 8.2(a) zu sehen. Nach der Äußerung des zweiten Weisen, wissen wir und alle Mitspieler, daß die Welt (w,b,w) nicht vorkommen kann. Der zweite Weise hätte sofort gewußt, daß er den schwarzen Hut auf hat. In Abbildung 8.2(a) gibt es aber noch eine zweite Welt, die für den zweiten Weisen ohne Alternative ist, nämlich (b,b,w). Denn hätte er in der Situation (b,?,w) einen weißen Hut auf, so hätte der erste Weise wissen müssen, daß er den einzigen schwarzen auf hat. Nach Elimination dieser beiden Welten ist die Situation in Abbildung 8.2(b) erreicht. Man sieht, daß dabei nur Welten vorkommen, in denen der dritte Weise einen schwarzen Hut auf hat.

Der Graph in Abbildung 8.1 kann als Modell für das Wissen der drei Akteure gedeutet werden. Die Aussage

in der Welt s weiß der i-te Weise die Aussage A



(a) Erster Schritt (b) Zweiter Schritt

Abbildung 8.2: Reduktion der möglichen Welten

wird dabei modelliert durch

in jeder für den i-ten Weisen von s ausgesehen möglichen Welt gilt A.

In modallogischer Notation wird diese Aussage geschrieben als

$$s \models \Box_i\ A$$

Vereinbaren wir, daß die Boolesche Variable B_i wahr ist in einer Welt, in der der i-te Weise einen schwarzen Hut auf hat und entsprechend für W_j, dann gilt in Abbildung 8.1 z.B.

$$\begin{array}{ll} (w,b,w) \models \Box_1 B_2 & (w,b,w) \models \Box_1 W_3 \\ \text{nicht } (w,b,w) \models \Box_1 B_1 & (b,w,w) \models \Box_1 B_1 \end{array}$$

Konfliktfreie Zugriffskontrolle Ein beliebter Algorithmus, der den konfliktfreien Zugriff mehrerer Prozesse auf eine kritische Ressource regelt, ist der sogenannte *bakery algorithm*. Der Name kommt von der in amerikanischen Bäckereien und *delicatessen shops* üblichen Praxis, daß der Kunde beim Eintreten eine Nummer zieht und dann wartet bis seine Nummer die kleinste unter den noch Wartenden ist. Dann ist er an der Reihe. Bei uns kennt man dieses Verfahren aus manchen Arztpraxen. Im Gegensatz zu dem namensgebenden Beispiel geht der Algorithmus nicht davon aus, daß es ein zentrales Gerät gibt, welches die Nummern verteilt. Das Problem, welches der *bakery* Algorithmus löst, ist im Englischen unter dem Namen *mutual exclusion* bekannt. Dieser Algorithmus ist überaus beliebt als Lehrbuchbeispiel und in einfache Fallstudien.

In der Beschreibung des Algorithmus gehen wir davon aus, daß jeder Prozess sich in einem der Zustände `idle, trying` oder `critical` befinded. Außerdem besitzt jeder Prozeß ein *ticket*, eine Nummer, die eine natürliche Zahl größer 0 ist. Prozesse, die noch nicht ihre Absicht bekundet haben, auf die kritische Ressource zuzugreifen, haben die Nummer 0. Wir können uns Prozesse

Customer
int ticket {idle, trying, critical} **phase**

Abbildung 8.3: Klassenspezifikation für Prozesse

vorstellen als Instanzen der Klasse *Customer*, siehe Abb.8.3. Der Algorithmus kann jetzt durch die folgenden Zustandsübergangsregeln beschrieben werden. Die Protokollregeln aus Tabelle 8.1 sehen aus, wie die Zustandsübergänge

try:	if phase = idle	then phase := trying ticket := max of all other tickets + 1
enter:	if phase = trying and ticket less than all other tickets	then phase := critical
leave	phase = critical	then phase := idle ticket := 0

Tabelle 8.1: Zustandsübergänge des *bakery* Protokolls

eines Automaten. Der einzig wichtige Unterschied besteht darin, daß Automaten in ihrer üblichen Form nur endliche viele Zustände haben dürfen. Das können wir erreichen, indem wir die Anzahl der beteiligten Prozesse und die maximale Nummer beschränken. Bei zwei Prozessen und maximaler Nummer 8 erhalten wir einen Automaten mit $576 = (3 * 8)^2$ Zuständen. Daß einige dieser Zustände nicht erreichbar sind, soll uns im Augenblick nicht interessieren. 576 Zustände sind für ein Lehrbuchbeispiel zu viel. Wir wollen daher zunächst die Nummern ganz ignorieren. Für zwei Prozesse bleiben dann noch 9 Zustände. Abbildung 8.4 zeigt diese Zustände einschließlich der

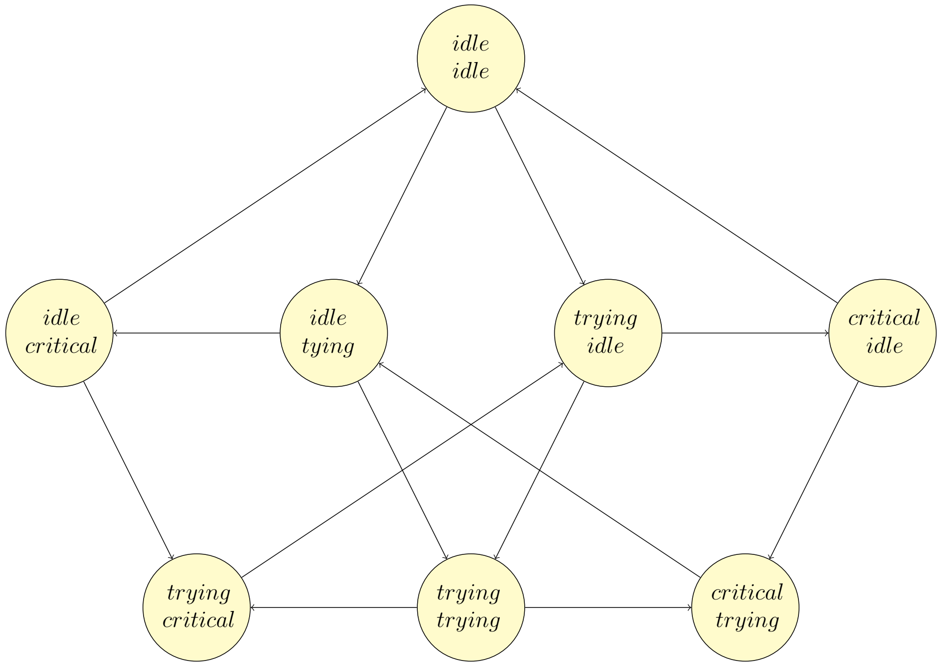


Abbildung 8.4: Zustandsübergänge im *bakery* Protokoll ohne Nummern

möglichen Übergänge. Der Zustand, daß beide Prozesse im kritischen Bereich sind, kommt nicht vor. Es ist die Aufgabe einer Protokollverifikation sicherzustellen, daß das gezeigt Modell mit einer Realisierung übereinstimmt. Eine Eigenschaft der Struktur in Abbildung 8.4, die für das Protokoll wichtig sein könnte, ist die Beobachtung, daß beide Prozess, wenn sie die Phase *trying* erreicht haben, immer in höchstens zwei Schritten in die kritische Phase kommen können, aber nicht müssen. Zur Formalisierung solche Eigenschaften ist ebenfalls die modale Logik geeignet. Die Booleschen Variablen $i.idle$, $i.trying$, $i.critical$ seien wahr in einem Zustand s, wenn in s der i-te Prozess in der angegebenen Phase ist. Dann ist die folgende Formel

$$1.trying \to (\Diamond 1.critical \vee \Diamond\Diamond 1.critical)$$

in jedem Zustand wahr. Der modale Operator $\Diamond$ funktioniert dabei so, daß eine Aussage $\Diamond A$ in einem Zustand s wahr ist, wenn es einen von s aus in einem Schritt erreichbaren Zustand t gibt, in dem A wahr ist.

8.2 Syntax und Semantik

Definition 8.1 (Syntax der modalen Aussagenlogik)
Sei Σ eine Menge aussagenlogischer Atome $\Sigma = \{P_0, P_1, \ldots\}$ dann ist die Menge $mFor0_\Sigma$ (bzw. kürzer $mFor0$), der *Formeln der modalen Aussagenlogik*, definiert durch

1. $\mathbf{1} \in mFor0_\Sigma$, $\mathbf{0} \in mFor0_\Sigma$
2. $\Sigma \subseteq mFor0_\Sigma$
3. Mit $A, B \in mFor0_\Sigma$ liegen ebenfalls in $mFor0_\Sigma$: $\neg A$, $(A \circ B)$ für $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
4. für $A \in mFor0_\Sigma$ gilt auch $\Box A \in mFor0_\Sigma$ und $\Diamond B \in mFor0_\Sigma$

Man liest $\Box A$ als *Box A* oder A ist notwendig. $\Diamond A$ wird gelesen als *Diamond* A oder A ist möglich.

Die modallogischen Operatoren lassen sich nicht mehr als Boolesche Funktionen deuten, eine *reine* Wahrheitswertsemantik ist daher nicht mehr möglich. Für die Angabe einer Semantik ist, wie durch die Beispiele des vorangegangenen Unterkapitels nahegelegt wird, eine Menge von Zuständen notwendig. Anstelle einer Interpretation der aussagenlogischen Variablen im Falle der klassischen Aussagenlogik, braucht man in der Modallogik für jeden Zustand eine Belegung der aussagenlogischen Variablen. Als letztes kommt noch hinzu, daß die Zustände nicht beziehungslos nebeneinander stehen, sondern durch eine Zugänglichkeitsrelation miteinander verbunden sind. All diese Angaben werden in dem Begriff einer*Kripke-Struktur* zusammengefasst.

Definition 8.2 (Kripke- Struktur)
Eine *Kripke-Struktur* $\mathcal{K} = (S, R, I)$ über Σ ist gegeben durch

- eine nichtleere Menge S von Zuständen
- eine Relation $R \subseteq S \times S$
- eine Interpretationsfunktion I: $(\Sigma \times S) \rightarrow \{W, F\}$

In der Modallogik nennt man Zustände gelegentlich auch *mögliche Welten*.

Zustände und Zugänglichkeitsrelation zusammen, ohne die Interpretation I, nennt man einen *Kripke-Rahmen*, $\mathcal{R} = (S, R)$.

Definition 8.3 (Auswertung der Formeln)
Sei $\mathcal{K} = (S, R, I)$, $s \in S$ ein Zustand, dann wird der Wahrheitswert einer Formel A im Zustand s mit $val_{(\mathcal{K},s)}(A)$ bezeichnet und wie folgt induktiv definiert:

$$
\begin{array}{rcl}
val_s(\mathbf{0}) & = & \mathbf{F} \\
val_s(\mathbf{1}) & = & \mathbf{W} \\
val_s(P) & = & I(P, s) \text{ für Atome } P \\
val_s(\neg A) & = & \begin{cases} \mathbf{F} & \text{falls } val_s(A) = \mathbf{W} \\ \mathbf{W} & \text{falls } val_s(A) = \mathbf{F} \end{cases} \\
val_s(A \circ B) & = & \begin{cases} \text{Wie in der Aussagenlogik} \\ \text{(wobei } \circ \text{ eine aussagenlogische Verknüpfung ist)} \end{cases} \\
val_s(\Box A) & = & \begin{cases} W & \text{falls für alle } s' \in S \text{ mit } sRs' \text{ gilt } val_{s'}(A) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases} \\
val_s(\Diamond A) & = & \begin{cases} W & \text{falls ein } s' \in S \text{ existiert mit } sRs' \\ & \text{und } val_{s'}(A) = \mathbf{W} \\ \mathbf{F} & \text{sonst} \end{cases}
\end{array}
$$

Wir sagen $\mathcal{K}$ ist ein *Modell* der Formel A, wenn $val_{(\mathcal{K},s)}(A) = W$ für alle $s \in S$.

Wenn $\mathcal{K}$ aus dem Kontext ersichtlich ist, schreiben wir $val_s(A)$ anstelle von $val_{(\mathcal{K},s)}(A)$, wie in der vorangegangenen Definition schon geschehen.

Wir benutzen gleichberechtigt die alternative Schreibweisen:

$$
\begin{array}{rcccl}
(\mathcal{K}, s) & \models & A & \Leftrightarrow & val_{(\mathcal{K},s)}(A) = \mathbf{W} \\
\multicolumn{5}{l}{\text{wenn } \mathcal{K} \text{ aus dem Kontext bekannt ist auch:}} \\
s & \models & A & \Leftrightarrow & val_s(A) = \mathbf{W} \\
\mathcal{K} & \models & A & \Leftrightarrow & \text{für alle } s \in S \text{ gilt } (\mathcal{K}, s) \models A \\
 & & & & \text{d.h. } \mathcal{K} \text{ ist Modell von } A.
\end{array}
$$

Bemerkung 8.4
Bei den Zuständen kommt es *nicht* etwa nur auf ihre Auswirkung auf val an, d.h. es kann durchaus in einer Kripke Struktur $\mathcal{K} = (S, R, I)$ zwei verschiedene Welten $s_1, s_2 \in S$ geben, so daß für alle Atome P gilt $val_{s_1}(P) = val_{s_2}(P)$.

Definition 8.5

A *allgemeingültig*	$\Leftrightarrow$	$val_{(\mathcal{K},s)}(A) = \mathrm{W}$ für alle $\mathcal{K} = (S, R, I)$ und alle $s \in S$
A *erfüllbar*	$\Leftrightarrow$	es gibt eine Kripke-Struktur $\mathcal{K} = (S, R, I)$ und ein $s \in S$ mit $val_{(\mathcal{K},s)}(A) = \mathrm{W}$.

Man überzeugt sich leicht, daß wie in der klassischen Aussagenlogik gilt: A allgemeingültig $\Leftrightarrow \neg A$ unerfüllbar.

Lemma 8.6 (Allgemeingültige modale Formeln)
Alle Instanzen der folgenden modallogischen Schemata sind allgemeingültig.

1. Jede aussagenlogisch Tautologie ist eine allgemeingültige modale Formel.
2. $\Box(A \to B) \to (\Box A \to \Box B)$
3. $(\Box A \land \Box B) \leftrightarrow \Box(A \land B)$
4. $(\Box A \lor \Box B) \to \Box(A \lor B)$
5. $\Box A \leftrightarrow (\neg\Diamond\neg A)$
6. $\Diamond A \leftrightarrow (\neg\Box\neg A)$
7. $(\Diamond A \lor \Diamond B) \leftrightarrow \Diamond(A \lor B)$
8. $\Diamond(A \land B) \to (\Diamond A \land \Diamond B)$
9. $(\Box(A \to B) \land \Box(B \to A)) \leftrightarrow \Box(A \leftrightarrow B)$
10. $\Box(A \to B) \to (\Diamond A \to \Diamond B)$

Beweise
zu 1: Diese Aussage ist offensichtlich richtig. Wir geben ein Beispiel:

Beispiel 8.7
Die modallogische Formel (A und B seien aussagenlogische Atome)
$((\underbrace{\Diamond A \land \neg\Box B}_{P_1}) \to \underbrace{\Box A}_{P_2}) \to ((\neg(\underbrace{\Diamond A \land \neg\Box B}_{P_1}) \to \underbrace{\Box A}_{P_2}) \to \underbrace{\Box A}_{P_2})$
ist eine Instanz des ausagenlogischen Schemas:
$(P_1 \to P_2) \to ((\neg P_1 \to P_2) \to P_2)$

zu 2: Sei $\mathcal{K} = (S, R, I)$ eine beliebige Kripke-Struktur und s ein beliebiger Zustand in S. Wir wollen $s \models \Box(A \to B) \to (\Box A \to \Box B)$ zeigen. Dazu genügt es zu zeigen, daß aus der Annahme $s \models \Box(A \to B)$ folgt $s \models \Box A \to \Box B$. Dazu genügt es wiederum aus der zusätzlichen Annahme $s \models \Box A$ auf $s \models \Box B$ zu schließen. Nach der Semantikdefinition besagen die beiden Voraussetzungen, daß für alle s' mit $R(s, s')$ gilt

$$\begin{array}{lll} s' & \models & A \to B \\ s' & \models & A \end{array}$$

Daraus folgt für alle s' mit $R(s, s')$ auch $s' \models B$, d.h. $s \models \Box B$, wie gewünscht.

zu 3: Die Definition dafür, daß die linke Seite der Äquivalenz in einem Zustand s eine Kripke-Struktur gilt lautet:

$$\begin{array}{ll} \text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models A & \text{und} \\ \text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models B & \end{array}$$

Die rechte Seite führt zu

$$\text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models A \wedge B$$

Die Äquivalenz ist jetzt klar ersichtlich.

zu 4: Aus

$$\begin{array}{ll} \text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models A & \text{oder} \\ \text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models B & \end{array}$$

folgt sicherlich

$$\text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models A \vee B$$

Die umgekehrte Implikation ist nicht allgemeingültig, siehe Abb.8.5(a).

zu 5: Nach Einsetzen der Semantikdefinition für beide Seiten erhalten wir

$$\text{für alle } s' \text{ mit } R(s, s') \text{ gilt } s' \models A$$

für die linke Seite und

$$\text{es gibt kein } s' \text{ mit } R(s, s'), \text{ so daß } s' \models \neg A \text{ gilt.}$$

für die rechte Seite. Offensichtlich sagen diese beiden Aussagen dasselbe.

zu 6: Folgt direkt aus 5. Damit meinen wir folgendes: Für $\neg A$ anstelle von A (schließlich ist A eine Schemavariable) lautet 5 $\Box\neg A \leftrightarrow \neg\Diamond\neg\neg A$. Setzt man das in der rechten Seite von 6 ein, so ergibt sich $\Diamond A \leftrightarrow \neg\neg\Diamond\neg\neg A$. Nach den offensichtlichen Vereinfachungen also die Tautologie $\Diamond A \leftrightarrow \Diamond A$.

Nach demselben Muster kann man aus 5 und 3 auf 7 schließen. Ebenso folgt 8 aus 5 und 4. Die Äquivalenz 9 schließlich ist eine direkte Konsequenz aus 3.

zu 10: Hier müssen wir noch einmal auf die Semantikdefinition zurückgreifen. Die linke Seite liefert

$$\text{für alle } s' \text{ mit } R(s,s') \text{ gilt } s' \models A \to B$$

Um daraus auf die rechte Seite zu schließen nehmen wir $s \models \Diamond A$ an, d.h.

$$\text{es gibt ein } s' \text{ mit } R(s,s') \text{ und } s' \models A$$

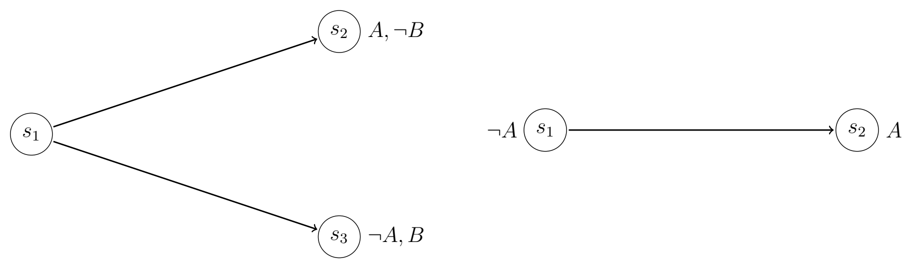
Zusammengenommen erhalten wir

$$\text{es gibt ein } s' \text{ mit } R(s,s') \text{ und } s' \models B.$$

Was die Definition für $s \models \Diamond B$ ist.

■

Die Punkte (5) und (6) in Lemma 8.6 zeigen außerdem, daß einer der beiden Operatoren $\Box$, $\Diamond$ entbehrlich ist. Es bilden also etwa $\{\mathbf{0}, \to, \Box\}$ eine Basis für die modale Aussagenlogik.



(a) Gegenbeispiel zu $\Box(A \vee B) \to (\Box A \vee \Box B)$

(b) Gegenbeispiel zu $\Box A \to A$

Abbildung 8.5: Gegenbeispiele

Definition 8.8
Sei A eine Formel und Γ eine Menge von Formeln der modalen Aussagenlogik.

1. A ist eine logische Folgerung aus Γ, in Symbolen $\Gamma \models A$, gdw für alle Kripke-Strukturen $\mathcal{K}$ und jede Welt s von $\mathcal{K}$ gilt: wenn $(\mathcal{K}, s) \models \Gamma$ dann auch $(\mathcal{K}, s) \models A$

2. A ist allgemeingültig wenn $\emptyset \models A$

Eine Variante wird als Definition 8.17 in Übungsaufgabe 8.5.2 betrachtet.

Das Zeichen $\models$ ist, wie schon in der Prädikatenlogik, überladen. Es bezeichnet die modallogische Folgerungsbeziehung und die Gültigkeit einer Formeln in einer Welt oder einer Kripke-Struktur. Da sich die verschiedenen Bedeutungen in der Typisierung des ersten Arguments unterscheiden, ist eine Verwechslung leicht auszuschließen.

8.3 Eigenschaften von Kripke-Rahmen

Die Formel $\Box A \rightarrow A$ ist nicht allgemeingültig, siehe Abb.8.5(b) für ein Gegenbeispiel. Die Formel ist dagegen wahr in allen Kripke-Strukturen $\mathcal{K} = (S, R, I)$ mit reflexivem Kripke-Rahmen (S, R). Wir nennen die gesamte Kripke-Struktur oder den ganzen Kripke-Rahmen reflexiv, wenn R eine reflexive Relation auf S ist.

Das ist nur die Spitze eines Eisbergs. Der Begriff der Allgemeingültigkeit kann auf die verschiedensten Teilklassen von Kripke-Strukturen relativiert werden.

Lemma 8.9 (Relative Allgemeingültigkeit modaler Formeln)
In den folgenden Formeln ist A eine aussagenlogische Variable.

1. In der Klasse aller reflexiven Kripke-Strukturen ist allgemeingültig: $\Box A \rightarrow A$

2. In der Klasse aller transitiven Kripke-Strukturen ist allgemeingültig: $\Box A \rightarrow \Box\Box A$

3. In der Klasse aller symmetrischen Kripke-Strukturen ist allgemeingültig: $A \rightarrow \Box\Diamond A$

4. In der Klasse aller dichten Kripke-Strukturen ist allgemeingültig: $\Box\Box A \rightarrow \Box A$
Dabei heißt eine Kripke-Struktur (S, R, I) dicht, wenn für alle $t_1, t_2 \in S$ mit $R(t_1, t_2)$ eine Welt $t_3 \in S$ existiert mit $R(t_1, t_3)$ und $R(t_3, t_2)$.

5. In der Klasse aller partiell funktionalen Kripke-Strukturen ist allgemeingültig:
   $\Diamond A \to \Box A$
   Dabei heißt eine Kripke-Struktur (S, R, I) partiell funktional, wenn für alle $s, t_1, t_2 \in S$ mit $R(s, t_1)$ und $R(s, t_2)$ folgt $t_1 = t_2$.

6. In der Klasse aller endlosen Kripke-Strukturen ist allgemeingültig:
   $\Box A \to \Diamond A$
   Dabei heißt eine Kripke-Struktur (S, R, I) endlos, wenn für jedes $s \in S$ ein t existiert mit $R(s, t)$.

Beweise
zu 1: Gilt $s \models \Box A$, dann gilt für alle s' mit $R(s, s')$ nach Definition $s' \models A$. Wegen der Reflexivität von R gilt insbesondere $R(s, s)$. Also $s \models A$.

zu 2: Gelte $s \models \Box A$, d.h. für alle s' mit $R(s, s')$ gilt $s' \models A$. Wir wollen $s \models \Box\Box A$ zeigen. Seien also beliebige s_1, s_2 gegeben mit $R(s, s_1)$ und $R(s_1, s_2)$, dann müssen wir $s_2 \models A$ zeigen. Wegen der Transitivität von R gilt aber unter diesen Voraussetzungen schon $R(s, s_2)$ und damit nach Voraussetzung $s_2 \models A$.

zu 3: Gelte $s \models A$. Wir wollen $s \models \Box\Diamond A$ zeigen. Dazu betrachten wir eine beliebige Welt t mit $R(s, t)$ und suchen ein s' mit $R(t, s')$ und $s' \models A$. Da R als symmetrisch vorausgesetzt ist, leistet $s' = s$ das Gewünschte.

zu 4: Gelte $t_1 \models \Box\Box A$. d.h. für alle t_3, t_2 mit $R(t_1, t_3)$ und $R(t_3, t_2)$ gilt $t_2 \models A$. Wir wollen zeigen, daß auch $t_1 \models \Box A$ gilt, d.h. für alle t_2 mit $R(t_1, t_2)$ ist $t_2 \models A$ zu zeigen. Wegen der Dichtheit gibt es ein t_3 mit $R(t_1, t_3)$ und $R(t_3, t_2)$, so daß aus der Voraussetzung tatsächlich $t_2 \models A$ folgt.

zu 5: Gelte $s \models \Diamond A$, d.h. es gibt t mit $R(s, t)$ und $t \models A$. Wir müssen $s \models \Box A$ nachweisen. Sei dazu t' eine beliebige Welt mit $R(s, t')$. Gilt $t' \models A$?. Wegen der partiellen Funktionalitätseigenschaft muß $t = t'$ gelten, und wir sind fertig.

zu 6: Gilt $s \models \Box A$, dann gilt für alle t mit $R(s, t)$ definitionsgemäß $t \models A$. Das schließt im Allgemeinen auch den Fall ein, daß kein t mit $R(s, t)$ existiert. Wegen der vorausgesetzten Endlosigkeit gibt es aber in der vorliegenden Situation mindestens ein solches t. Somit ist $s \models \Diamond A$ gezeigt.

■

Für die verschiedenen Klassen von Kripke-Strukturen und die dazugehörigen Logiken hat sich eine bizarre, wenig systematische Bezeichungskonvention etabliert:

die Klasse **K** aller Kripke-Strukturen

die Klasse **T** aller reflexiven Kripke-Strukturen

die Klasse **A4** aller transitiven Kripke-Strukturen

die Klasse **S4** aller reflexiven und transitiven Kripke-Strukturen

die Klasse **S5** aller transitiven und symmetrischen Kripke-Strukturen.

Die in Lemma 8.9 aufgelisteten Formeln sind so gewählt, daß sie die Klasse von Kripke-Strukturen, für die sie allgemeingültig sind, in einem gewissen Sinne charakterisieren. Die naive Vermutung, daß für eine Kripke-Struktur $\mathcal{K} = (S, R, I)$ mit $s \models \Box A \to \Box\Box A$ für alle $s \in S$, schon die Transitivität von (S, R) folgen würde, ist falsch. Ganz unabhängig von (S, R) wird $s \models \Box A \to \Box\Box A$ wahr, wenn wir nur $I(A, s) = W$ wählen für alle $s \in S$. Die folgende Definition beschreibt den richtigen Zusammenhang.

Definition 8.10
Sei **R** eine Klasse von Kripke-Rahmen, und A eine Formel der Modallogik.

A *charakterisiert* die Klasse **R** genau dann, wenn für alle Kripke-Rahmen (S, R) gilt

für alle I gilt $(S, R, I) \models A$
gdw
$(S, R) \in \mathbf{R}$

Lemma 8.11 (Charakterisierungen)
In den folgenden Formeln ist A eine aussagenlogische Variable.

1. Die Formel $\Box A \to A$ charakterisiert die Klasse der reflexiven Kripke-Rahmen.

2. Die Formel $\Box A \to \Box\Box A$ charakterisiert die Klasse der transitiven Kripke-Rahmen.

3. Die Formel $A \to \Box\Diamond A$ charakterisiert die Klasse der symmetrischen Kripke-Rahmen.

4. Die Formel $\Box\Box A \rightarrow \Box A$ charakterisiert die Klasse der dichten Kripke-Rahmen.

5. Die Formel $\Diamond A \rightarrow \Box A$ charakterisiert die Klasse der partiell funktionalen Kripke-Rahmen.

6. Die Formel $\Box A \rightarrow \Diamond A$ charakterisiert die Klasse der endlosen Kripke-Rahmen.

Beweise In allen Beweisen ist eine Implikationsrichtung schon durch Lemma 8.9 abgedeckt. Wenn z.B. (S, R) transitiv ist, dann gilt für alle I schon $(S, R, I) \models \Box A \rightarrow \Box\Box A$. Im Folgenden wird immer nur die verbleibende Richtung behandelt.

zu 1: Wir wollen zeigen, daß jeder Kripke-Rahmen (S, R), für den $(S, R, I) \models \Box A \rightarrow A$ für alle I gilt, reflexiv ist. Wir benutzen hier, wie in allen nachfolgenden Fällen, Beweis durch Widerspruch: Angenommen (S, R) ist nicht reflexiv, dann gibt es ein I, so daß $(S, R, I) \not\models \Box A \rightarrow A$. Wenn (S, R) nicht reflexiv ist, dann gibt es ein $s_0 \in S$ mit $\neg R(s_0, s_0)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s \neq s_0 \\ \mathbf{F} & \text{falls } s = s_0 \end{cases}$$

Nach dieser Definition gilt für $\mathcal{K} = (S, R, I)$ einerseits $(\mathcal{K}, s_0) \not\models A$ und andererseits $(\mathcal{K}, s_0) \models \Box A$. Also insgesamt $(\mathcal{K}, s_0) \not\models \Box A \rightarrow A$.

zu 2: Dieser Teil läuft, wie alle folgenden, nach demselben Muster ab, wie der erste. Wir fassen uns deswegen etwas kürzer.

Wenn ein Rahmen (S, R) nicht transitiv ist, dann gibt es $s_1, s_2, s_3 \in S$ mit $R(s_1, s_2)$ und $R(s_2, s_3)$ aber $\neg R(s_1, s_3)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } R(s_1, s) \\ \mathbf{F} & \text{falls } sonst \end{cases}$$

Nach dieser Definition folgt sofort $s_1 \models \Box A$. Außerdem gilt $s_3 \models \neg A$. Da s_3 in zwei R-Schritten von s_1 aus erreichbar ist, haben wir auch $s_1 \models \neg\Box\Box A$. Insgesamt $s_1 \models \neg(\Box A \rightarrow \Box\Box A)$.

zu 3: Wenn ein Rahmen (S, R) nicht symmetrisch ist, dann gibt es $s_1, s_2 \in S$ mit $R(s_1, s_2)$ und $\neg R(s_2, s_1)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s = s_1 \\ \mathbf{F} & \text{falls } sonst \end{cases}$$

Trivialerweise gilt $s_1 \models A$. Für s_2 gibt es entweder überhaupt kein s mit $R(s_2, s)$ oder für alle s mit $R(s_2, s)$ gilt $s \models \neg A$. Das heißt aber $s_2 \models \neg\Diamond A$ und somit auch $s_1 \models \neg\Box\Diamond A$. Zusammengenommen erhalten wir den Widerspruch $s_1 \models \neg(A \to \Box\Diamond A)$.

zu 4: Ist (S, R) kein dichter Kripke-Rahmen, dann gibt es t_1, t_2 mit $R(t_1, t_2)$, so daß kein t_3 existiert mit $R(t_1, t_3)$ und $R(t_3, t_2)$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s = t_1 \text{ und es gibt } s_1, s_2 \\ & \quad \text{mit } R(t_1, s_1) \text{ und } R(s_1, s_2) \\ \mathbf{F} & \text{falls } sonst \end{cases}$$

Unmittelbar aus der Definition von I folgt $t_1 \models \Box\Box A$. Andererseits gilt $t_2 \models \neg A$, also auch $t_1 \models \neg\Box A$.

zu 5: Ist (S, R) nicht partiell funktional, dann gibt es s_1, s_2, s_3 mit $R(s_1, s_2)$, $R(s_1, s_3)$ und $s_2 \neq s_3$. Wir definieren eine Interpretation I durch

$$I(A, s) = \begin{cases} \mathbf{W} & \text{falls } s = s_2 \\ \mathbf{F} & \text{falls } sonst \end{cases}$$

Somit gilt $s_1 \models \Diamond A$ und $s_1 \models \neg\Box A$. Insgesamt also $s_1 \not\models (\Diamond A \to \Box A)$

zu 6: Ist (S, R) nicht endlos, dann gibt es $s_1 \in S$ mit $\neg R(s_1, s)$ für alle $s \in S$. Für jede beliebige Formel C gilt dann offensichtlich $s_1 \models \Box C$ und $s_1 \models \neg\Diamond C$. Insbesondere $s_1 \models \neg(\Box A \to \Diamond A)$.

■

8.4 Entscheidbarkeit modaler Logiken

Die Erfüllbarkeit einer Formel der klassischen Aussagenlogik kann in endliche vielen Schritten entschieden werden. Schlimmstenfalls probiert man alle möglichen Interpretationen der vorkommenden aussagenlogischen Variablen aus. Um die Erfüllbarkeit einer Formel A der modalen Aussagenlogik nachzuweisen muß man eine Kripke-Struktur $\mathcal{K} = (S, R, I)$ angeben, so daß A in einer Welt $s \in S$ wahr ist. Ein einfaches Durchprobieren aller Möglichkeiten ist nicht mehr möglich, da die Anzahl der Welten in S beliebig groß sein kann. Wir wollen in diesem Unterabschnitt zeigen, daß die meisten modalen Aussagenlogiken dennoch entscheidbar sind. Der entscheidende Schritt besteht darin, eine obere Schranke für die Anzahl der möglichen Welten in einer erfüllenden Kripke-Struktur aus der Größe der Formel A herzuleiten.

Das zentrale Konzept auf dem Weg dorthin ist die Filtration.

Definition 8.12 (Filtration)
Sei $\mathcal{K} = (G, R, v)$ eine Kripke Struktur und Γ eine Menge von Formeln der modalen Aussagenlogik.
Wir benutzen hier und im Folgenden (G, R, v) anstelle von (S, R, I), aber das sollte dem Leser keine Schwierigkeiten bereiten.

Die Äquivalenzrelation $\sim_\Gamma$ auf G wird definiert durch:

$$g \sim_\Gamma h \text{ gdw } (\mathcal{K}, g) \models A \Leftrightarrow (\mathcal{K}, h) \models A \text{ für alle } A \in \Gamma$$

Für $g \in G$ bezeichnen wir mit $[g]$ die Äquivalenzklasse von g bezüglich $\sim_\Gamma$, i.e. $[g] = \{h \in G \mid g \sim_\Gamma h\}$. Die Kripke Struktur $\mathcal{K}_\Gamma = (G_\Gamma, R_\Gamma, v_\Gamma)$ ist wie folgt definiert

$$\begin{array}{lcll} G_\Gamma & = & \{[g] \mid g \in G\} & \\ {[g]}R_\Gamma[h] & \Leftrightarrow & \exists g_0 \in [g] \exists h_0 \in [h](g_0 R h_0) & \\ v_\Gamma([g], p) & = & v(g, p) & \text{für } p \in \Gamma \\ v_\Gamma([g], p) & = & \text{beliebig} & \text{sonst} \end{array}$$

$\mathcal{K}_\Gamma$ heißt eine *Filtration* von $\mathcal{K}$ bezüglich Γ.

Bemerkung 8.13
Man überzeugt sich leicht, daß diese Definition unabhängig von der Wahl der Repräsentanten der beteiligten Äquivalenzklassen ist.

Lemma 8.14

1. Sei Γ eine Menge von Formeln der modalen Aussagenlogik, abgeschlossen unter Teilformeln, dann gilt für alle $A \in \Gamma$ und alle $g \in G$

$$(\mathcal{K}, g) \models A \text{ gdw } (\mathcal{K}_\Gamma, [g]) \models A$$

2. Ist Γ endlich mit $\#\Gamma = n$, dann ist $\#G_\Gamma \leq 2^n$.

Bemerkung 8.15
Die Formelmenge Γ heißt *abgeschlossen unter Teilformeln* wenn für jedes $A \in \Gamma$ auch $B \in \Gamma$ gilt für jede Teilformel B von A.

Beweis:

(1) Der Beweis geschieht durch Induktion über den Formelaufbau von A. Ist A eine Aussagenvariable, dann folgt die Behauptung aus der Definition von v_Γ und der Äquivalenzrelation $\sim_\Gamma$. Wir betrachten nur noch den Induktionsschritt von B auf $\Box B$. Gelte $(\mathcal{K}_\Gamma, [g]) \models \Box B$. Dann gilt für alle $h \in G$ mit $[g]R_\Gamma[h]$ auch $(\mathcal{K}_\Gamma, [h]) \models B$. Nach Induktionsvoraussetzung folgt damit für alle $h \in G$ mit $[g]R_\Gamma[h]$ auch $(\mathcal{K}, h) \models B$ und damit auch für alle $h \in G$ mit gRh ebenfalls $(\mathcal{K}, h) \models B$, also auch $(\mathcal{K}, g) \models \Box B$. Gelte jetzt umgekehrt $(\mathcal{K}, g) \models \Box B$. Wir betrachten $g, h \in G$ mit $[g]R_\Gamma[h]$. Nach Definition gibt es $g_0, h_0 \in G$ mit $g_0 \sim_\Gamma g$, $h_0 \sim_\Gamma h$ und $g_0 R h_0$. Nach Definition von $\sim_\Gamma$ gilt auch $(\mathcal{K}, g_0) \models \Box B$ und damit auch $(\mathcal{K}, h_0) \models B$. Woraus mit der Definition von $\sim_\Gamma$ wieder $(\mathcal{K}, h) \models B$ folgt. Nach Induktionsvoraussetzung erhalten wir weiter $(\mathcal{K}_\Gamma, [h]) \models B$. Insgesamt ist damit für alle $[h] \in G_\Gamma$ mit $[g]R_\Gamma[h]$ die Gültigkeit von $(\mathcal{K}_\Gamma, [h]) \models B$ gezeigt und damit $(\mathcal{K}_\Gamma, [g]) \models \Box B$

(2) Enthält Γ n Formeln, so kann es höchstens 2^n Äquivalenzklassen von $\sim_\Gamma$ geben, da eine Äquivalenzklasse eindeutig bestimmt ist durch Angabe welche der Formeln in Γ wahr bez. falsch sein soll. Das Maximum 2^n wird tatsächlich erreicht, wenn Γ z.B. nur aus aussagenlogischen Variablen besteht.

∎

Satz 8.16
Die modale Aussagenlogik **K** ist entscheidbar, d.h. es gibt einen Algorithmus, der für jede Formel A entscheidet, ob A allgemeingültig in der Klasse **K** ist oder nicht.

Beweis: Sei Γ die endliche Menge aller Teilformeln von $\neg A$. Wenn $\neg A$ überhaupt erfüllbar ist, dann nach Lemma 8.14 auch in einer Kripke Struktur mit höchstens 2^n Welten, wobei n die Kardinalität von Γ ist. Alle Kripke Stukturen mit höchstens 2^n Welten lassen sich endlich aufzählen, und es ist für jede dieser Strukturen entscheidbar, ob $\neg A$ in ihr wahr ist oder nicht.

∎

8.5 Übungsaufgaben

Übungsaufgabe 8.5.1
Zeigen Sie die folgenden Behauptungen:

1. In der Klasse aller reflexiven Kripke-Strukturen sind allgemeingültig:
   $A \to \Diamond A$

2. In der Klasse aller transitiven Kripke-Strukturen sind allgemeingültig:
   $\Diamond\Diamond A \to \Diamond A$

3. In der Klasse aller symmetrischen Kripke-Strukturen sind allgemeingültig:
   $\Diamond\Box A \to A$.

4. In der Klasse aller transitiven und symmetrischen Kripke-Strukturen sind allgemeingültig:
   $\Diamond\Box A \to \Box A$

Übungsaufgabe 8.5.2
Die in Definition 8.8 definierte Folgerungsbeziehung wird genauer die *lokale* Folgerungsbeziehung genannt. Für die Zwecke dieser Übungsaufgabe schreiben wir deutlicher $\Gamma \models^L A$ anstelle von $\Gamma \models A$. Es gibt noch eine zweite Möglichkeit eine Folgerungsbeziehung, die *globale* Folgerungsbeziehung, zu definieren.

Definition 8.17
Sei A eine Formel und Γ eine Menge von Formeln der modalen Aussagenlogik.

A ist eine globale logische Folgerung aus Γ, in Symbolen $\Gamma \models^G A$, gdw
für alle Kripke-Strukturen $\mathcal{K}$ gilt:
wenn für jede Welt s von $\mathcal{K}$ gilt $(\mathcal{K}, s) \models \Gamma$
dann gilt für jede Welt s auch $(\mathcal{K}, s) \models A$

1. Zeigen Sie, daß $\emptyset \models^L A$ genau dann gilt wenn $\emptyset \models^G A$ gilt.

2. Zeigen Sie, daß aus $\Gamma \models^L A$ stets $\Gamma \models^G A$ folgt.

3. Finden Sie ein Beispiel, so daß $\Gamma \models^G A$ gilt aber nicht $\Gamma \models^L A$.

Übungsaufgabe 8.5.3
Sei $\mathcal{K} = (S, R, I)$ eine Kripke-Struktur und $T \subseteq S$ eine Teilmenge von S mit der folgenden Abschlußeigenschaft

$$\text{für alle } s_1, s_2 \text{ mit } s_1 \in T \text{ und } R(s_1, s_2) \text{ gilt } s_2 \in T$$

Wir definieren eine Teilstruktur $\mathcal{K}_T = (T, R_T, I_T)$ von $\mathcal{K}$ wobei R_T und I_T die Einschränkungen von R und I auf die Teilmenge T sind, d.h. für $s_1, s_2 \in T$ gilt:

$$\begin{array}{lll} R_T(s_1, s_2) & \text{gdw} & R(s_1, s_2) \\ I_T(P, s_1) & \text{gdw} & I(P, s_1) \end{array}$$

für jedes aussagenlogische Atom P.

Zeigen Sie für jedes $s \in T$ und für jede modallogische Formel A

$$(\mathcal{K}, s) \models A \text{ gdw } (\mathcal{K}_T, s) \models A$$

Wir geben zur Illustration ein Beispiel für eine Teilmenge $T \subseteq S$ mit der gewünschten Abschlußeigenschaft. Sei dazu $s_0 \in S$. Dann sei S_0 die Menge der von s_0 aus via R erreichbaren Welten:

$$\begin{array}{rcl} S_0 & = & \{s \in S \mid \text{es gibt } n \text{ und es gibt } s_i \text{ für } 0 \leq i \leq n \\ & & \qquad \text{mit } R(s_i, s_{i+1}) \text{ für } 0 \leq i < n \text{ und } s_n = s\} \end{array}$$

Übungsaufgabe 8.5.4
Ist A eine modallogische Formel so stehe $\Box^n A$ für die Formel $\underbrace{\Box \ldots \Box}_{n-\text{mal}} A$ und A^* für die Formelmenge $\{\Box^n A \mid 0 < n\}$
Zeigen Sie:

$$A \models^G B \text{ gdw } A^* \models^L B$$

Übungsaufgabe 8.5.5

Definition 8.18
Sei A eine modallogische Formel. Die *negationsduale* Formel B zu A wird erhalten, in dem jedes aussagenlogsiche Atom p in A durch $\neg p$ ersetzt wird.

Zeigen Sie: Ist A eine charakterisierende Formel für die Klasse $\mathbf{R}$ von Rahmen, dann ist auch die negationsduale Formel B eine charakterisierende Formel für $\mathbf{R}$.

Übungsaufgabe 8.5.6
Bestimmen Sie die negationsdualen Formeln zu

1. $\Box A \to A$
2. $\Box A \to \Box\Box A$

3. $A \to \Box\Diamond A$

4. $\Diamond A \to \Box A$

Übungsaufgabe 8.5.7
Wir nennen einen Kripke-Rahmen (S, R) *Euklidisch* wenn er die Formel $\forall x \forall y \forall z (R(x, y) \land R(x, z) \to R(y, z))$ erfüllt.
Beweisen Sie, daß die modallogische Formel $\Diamond p \to \Box\Diamond p$ die Klasse der Euklidschen Rahmen charakterisiert.

Übungsaufgabe 8.5.8
Wir nennen einen Kripke-Rahmen (S, R) *schwach konfluent* wenn er die Formel $\forall x \forall y (R(x, y) \to \exists z (R(x, z) \land R(y, z)))$ erfüllt.

Finden Sie eine modallogische Formel, welche die Klasse der schwach konfluenten Rahmen charakterisiert.
Offensichtlich ist jede reflexive, symmetrische Relation R auch schwach konfluent, die Umkehrung muß jedoch bei weitem nicht gelten.

Kapitel 9

Temporale Logik

Die temporale Logik LTL, die wir betrachten wollen, ist eine spezielle modale Logik. Es gibt sehr viele Varianten temporaler Logiken. Die größte Bedeutung haben aussagenlogische Temporallogiken. Wir wollen uns hier mit einem einfachen Beispiel begnügen, ebem mit der linearen temporalen Logik, LTL. Eine umfassendere Darstellung findet man in dem Skriptum P. Schmitt „Nichtklassische Logiken“.

Das gesamte Rahmenwerk aus dem vorangegangen Kapitel 8.1 wird wieder benutzt, insbesondere wird es in LTL wieder modale Operatoren geben, sogar mehr als bisher, und zur Erklärung der Semantik werden wieder Kripke-Strukturen $\mathcal{K} = (S, R, I)$ benutzt.

Im Allgemeinen werden für die Semantik von LTL sogenannte *Zeitstrukturen* verwendet.

Definition 9.1
Eine Zeitstruktur ist von der Form $\mathcal{T} = (S, R, I)$, wobei R eine strikte Quasiordnung (siehe Def.1.2) ist. Wir schreiben dann auch suggestiver $<$ anstelle von R und nennen die Elemente s der Menge S Zeitpunkte.

Mit Hilfe von Zeitstrukturen lässt sich auch ein kontinuierlicher Zeitverlauf modellieren. Als abstrakte Zeitpunkte kann man z.B. die Menge der reelen oder rationalen Zahlen benutzen. Zeitstrukturen müssen nicht notwendigerweise einen Anfangspunkt haben. Das ist hilfreich, wenn man nicht nur unbegrenzt viele zukünftige Zeitpunkte, sondern auch unbegrenzt viele Zeitpunkte in der Vergangenheit betrachten möchte.

Wir sagen Zeitpunkt t ist ein unmittelbarer Nachfolger des Zeitpunktes s, in Zeichen $s \prec t$, wenn $s < t$ gilt und kein Zeitpunkt s_0 existiert mit $s < s_0 < t$. In Zeitstrukturen muß nicht jeder Zeitpunkt einen unmittelbaren Nachfolger besitzen.

Omega-Strukturen sind der Spezialfall von Zeitstrukturen, in denen die Quasiordnung (S, R) gleich $(\mathbb{N}, <)$ ist. Details werden in der nachfolgenden Definition 9.3 gegeben.

9.1 Lineare Temporale Logik (LTL)

Definition 9.2 (LTL-Formeln)
Sei Σ eine Menge aussagenlogischer Atome. Die Menge $LTLFor$ (bzw. genauer $LTLFor_\Sigma$ falls erforderlich) der LTL-Formeln ist definiert durch

1. $\Sigma \subseteq LTLFor$

2. $\mathbf{1}, \mathbf{0} \in LTLFor$

3. Liegen A, B in $LTLFor$, dann auch alle aussagenlogischen Kombinationen von A und B.

4. für $A, B \in LTLFor$ gilt auch

   (a) $\Box A \in LTLFor$ und

   (b) $\Diamond B \in LTLFor$ und

   (c) $A \mathbf{U} B \in LTLFor$

   (d) $X\ A$

Die Symbole $\Box$, $\Diamond$, X und $\mathbf{U}$ heißen temporale Modaloperatoren. Die Notation für die temporalen Modaloperatoren ist leider nicht einheitlich. So finden man auch die folgenden Varianten

$G\ A$	für	$\Box A$
$F\ A$	für	$\Diamond A$
$A\ until\ B$	für	$A \mathbf{U} B$
$\circ A$	für	$X\ A$

Für die Semantik von LTL benutzten wir *omega-Strukturen*.

Definition 9.3
Eine *omega-Struktur* $\mathcal{R} = (\mathbb{N}, <, \xi)$ für eine aussagenlogische Signatur Σ besteht aus der geordneten Menge der natürlichen Zahlen

$$(\mathbb{N}, <)$$

interpretiert als Menge abstrakter Zeitpunkte und einer Funktion

$$\xi : \mathbb{N} \to 2^{\Sigma}$$

mit der Intention

$$p \in \xi(n) \Leftrightarrow \text{ in } \mathcal{R} \text{ ist } p \text{ zum Zeitpunkt } n \text{ wahr}$$

Für $\xi : \mathbb{N} \to 2^{\Sigma}$ und $n \in \mathbb{N}$ steht ξ_n für das bei n beginnende Endstück von ξ. In Zeichen

$$\xi_n(m) = \xi(n+m)$$

Inbesondere gilt $\xi_0 = \xi$.

Wir wisen darauf hin, daß wir hier im Unterschied zur früheren Praxis in Abschnitt 2.2 kleine Buchstaben p, q oder in alphabetischer Nähe liegende benutzen.

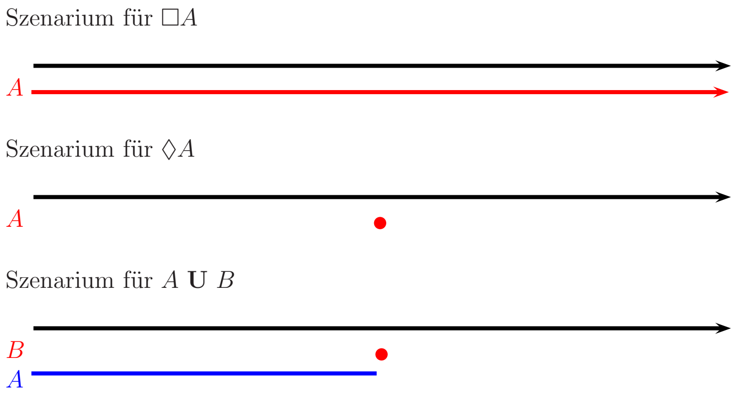


Abbildung 9.1:

Definition 9.4 (Semantik von LTL-Formeln)
Sei $\mathcal{R} = (\mathbb{N}, <, \xi)$ eine omega-Struktur und$A$ eine LTL Formel.

$\xi \models p$	gdw	$p \in \xi(0)$ (p ein AL Atom)
$\xi \models op(A, B)$		für aussagenlogische Kombinationen $op(A, B)$ von A und B wie üblich
$\xi \models \Box A$	gdw	für alle $n \in \mathbb{N}$ gilt $\xi_n \models A$
$\xi \models \Diamond A$	gdw	es gibt ein $n \in \mathbb{N}$ mit $\xi_n \models A$
$\xi \models A \mathbf{U} B$	gdw	es gibt $n \in \mathbb{N}$ mit $\xi_n \models B$ und für alle m mit $0 \leq m < n$ gilt $\xi_m \models A$
$\xi \models X\ A$	gdw	$\xi_1 \models A$

In Abbildung 9.1 ist der Versuch zu sehen, die Bedeutung der temporalen Operatoren graphisch darzustellen.

Beispiel 9.5

$\xi \models \Diamond X\ p$	gdw	$\xi_n \models X\ p$	für ein $n \in \mathbb{N}$
	gdw	$(\xi_n)_1 \models p$	für ein $n \in \mathbb{N}$
	gdw	$\xi_{n+1} \models p$	für ein $n \in \mathbb{N}$

$\xi \models X\ \Diamond p$	gdw	$\xi_1 \models \Diamond p$	
	gdw	$(\xi_1)_n \models p$	für ein $n \in \mathbb{N}$
	gdw	$\xi_{1+n} \models p$	für ein $n \in \mathbb{N}$

Die beiden Beispiele zeigen auch, daß $\Diamond X\ p \leftrightarrow X\ \Diamond p$ in allen omega-Strukturen wahr ist.

Das nächste Lemma zeigt, daß die beiden Operatoren $\Box$ und $\Diamond$ in der Definition 9.2 hätten eingespart werden können.

Lemma 9.6
Die folgenden Äquivalenzen gelten in allen Zeitstrukturen:

$$\begin{array}{l} \Diamond A \leftrightarrow \mathbf{1}\ \mathbf{U}\ A \\ \Box A \leftrightarrow \neg(\mathbf{1}\ \mathbf{U}\ \neg A) \end{array}$$

Beweis: Einfach.

∎

Dieses Lemma zeigt, daß man in LTL allein mit den Operatoren $\mathbf{U}$ und X auskommen kann. In der Literatur treten zusätzlich auch die Operatoren $\mathbf{U}_w$ und $\mathbf{V}$ auf, deren Semantik wir uns kurz anschauen wollen. $\mathbf{U}_w$ heißt der *schwache until-Operator* während $\mathbf{V}$ gelegentlich *release* Operator genannt wird.

Definition 9.7 (Zusätzliche Operatoren)

$$\begin{array}{lll} \xi \models A\ \mathbf{U}_w\ B & \text{gdw} & \text{für alle } n \in \mathbb{N} \text{ gilt } \xi_n \models (A \wedge \neg B) \text{ oder} \\ & & \text{es gibt } n \in \mathbb{N} \text{ mit } \xi_n \models B \text{ und} \\ & & \text{für alle } m \text{ mit } 0 \leq m < n \text{ gilt } \xi_m \models A \\ & & \\ \xi \models A\ \mathbf{V}\ B & \text{gdw} & \xi \models B \text{ und für alle } n \in \mathbb{N} \text{ gilt} \\ & & \text{falls } \xi_n \models \neg B \text{ dann gibt es ein } m \text{ mit} \\ & & 0 \leq m < n \text{ und } \xi_m \models A \end{array}$$

Siehe Abbildung 9.2 für eine Visualisierung des temporalen Operators $\mathbf{V}$.

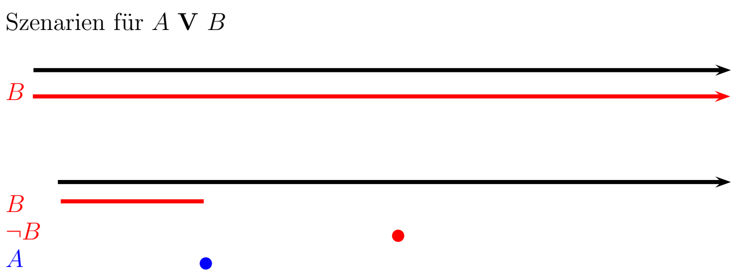


Abbildung 9.2:

Lemma 9.8
Für alle omega-Strukturen gilt:

1. $A\ \mathbf{U}\ B \leftrightarrow A\ \mathbf{U}_w\ B \wedge \Diamond B$

2. $A \ \mathbf{U}_w \ B \leftrightarrow A \ \mathbf{U} \ B \vee \Box(A \wedge \neg B)$

3. $A \ \mathbf{V} \ B \leftrightarrow \neg(\neg A \ \mathbf{U} \ \neg B)$

4. $A \ \mathbf{U} \ B \leftrightarrow \neg(\neg A \ \mathbf{V} \ \neg B)$

5. $A \ \mathbf{U} \ B \leftrightarrow (B \vee (A \wedge X \ (A \ \mathbf{U} \ B)))$

6. $A \ \mathbf{V} \ B \leftrightarrow (B \wedge A) \vee (B \wedge X \ (A \ \mathbf{V} \ B))$

7. $\neg(A \ \mathbf{V} \ B) \leftrightarrow \neg A \ \mathbf{U} \ \neg B$

8. $\neg(A \ \mathbf{U} \ B) \leftrightarrow \neg A \ \mathbf{V} \ \neg B$

Beweise:

zu 1 Einfach.

zu 2 Folgt direkt aus den Definitionen.

zu 3 Wir inspizieren die rechte Seite. Es gibt zwei Möglichkeiten, daß eine $\mathbf{U}$ -Aussage in einer omega-Struktur ξ fehlschlagen kann. Erstens, wenn zu keinem Zeitpunkt das zweite Argument wahr wird und zweitens, wenn es zwar einen Zeitpunkt gibt, zu dem das zweite Argument wahr ist, aber für jeden solchen Zeitpunkt t zwischen jetzt und t ist nicht immer das erste Argument wahr. Für die vorliegende Formel ergeben sich die beiden folgenden Möglichkeiten

1. für alle n gilt $\xi_n \models B$

2. für alle mit n mit $\xi_n \models \neg B$ gibt es ein m mit $0 \leq m < n$ und $\xi_m \models A$.

Somit gilt auf jeden Fall $\xi_0 = \xi \models B$. Im ersten Fall ist das klar. Im zweiten Fall würde $\xi_0 \models \neg B$ zu dem Widerspruch führen, daß ein m existieren soll mit $0 \leq m < 0$. Damit ist die erste Forderung von $\xi \models A \ \mathbf{V} \ B$ gezeigt. Der zweite Teil der Definition von $\xi \models A \ \mathbf{V} \ B$ folgt unmittelbar.

Der Beweis der umgekehrten Implikation ist leicht nachzuvollziehen.

zu 4 Aussagenlogische Umformulierung von (3).

zu 5 Wir betrachten ein ξ mit $\xi \models A$ **U** B. Wir unterscheiden die beiden Fälle $\xi \models B$ und $\xi \models \neg B$. Im ersten Fall kann man nichts weiter schließen, $\xi \models B$ reicht ja schon aus um $\xi \models A$ **U** B wahr zu machen. Im zweiten Fall muß auf jeden Fall $\xi \models A$ gelten und $\xi_1 \models A$ **U** B, was gleichbedeutend mit $\xi \models X\ (A$ **U** $B)$. Das erledigt die Implikatin von links nach rechts. Die inverse Implikation folgt ebenso einfach.

zu 6 Nehmen wir zunächst $\xi \models A$ **V** B an. Falls $\xi \models B \wedge X\ (A$ **V** $B)$ gilt dann gilt trivialerweise die recht Seite. Betrachten wir also den Fall $\xi \models \neg(B \wedge X\ (A$ **V** $B))$. Das heißt $\xi \models \neg B \vee \neg X\ (A$ **V** $B))$. Nach Annahme gilt $\xi \models A$ **V** B, woraus auf jeden Fall $\xi \models B$ folgt. Unsere Annahme reduziert sich damit auf $\xi \models \neg X\ (A$ **V** $B)$. Das kann aus zwei Gründen passieren

1. $\xi_1 \models \neg B$
2. $\xi_1 \models B$, es gibt $n > 1$ mit $\xi_n \models \neg B$ und es gibt kein $1 \leq m < n$ mit $\xi_m \models A$.

Im ersten Fall muß es wegen $\xi \models A$ **V** B ein j geben mit $0 \leq j < 1$ und $\xi_j \models A$. Für j bleibt also nur $j = 0$, d.h. $\xi \models A$. Im zweiten Fall folgt wieder aus $\xi \models A$ **V** B die Existenz eines j mit $0 \leq j < n$ mit $\xi_j \models A$. Wieder bleibt unter den Annahmen dieses Falles $j = 0$ als einzige Möglichkeit. Da ausserdem noch $\xi \models B$ aus $\xi \models A$ **V** B folgt haben wir $\xi \models (B \wedge A)$ gezeigt.

Gelte jetzt die rechte Seite $\xi \models (B \wedge A) \vee (B \wedge X\ (A$ **V** $B))$. Es ist einfach zu sehen, daß sowohl aus $\xi \models (B \wedge A)$ als auch aus $\xi \models (B \wedge X\ (A$ **V** $B))$ folgt $\xi \models A$ **V** B.

zu 7 Einfache Umformung von 3.

zu 8 Einfache Umformung von 4.

■

Lemma 9.9
Zu jeder LTL-Formel A, in der höchstens die temporalen Operatoren **V** und **U** vorkommen, gibt es eine äquivalente Formel A_{nnf} in Negationsnormform.

Dabei heißt in Fortsetzung der Terminologie aus der Aussagenlogik eine LTL-Formel eine *Negationsnormalform* wenn in ihr Negationszeichen nur vor Atomen auftreten.

Beweis Neben den aussagenlogischen Regeln braucht man nur (7) und (8) aus Lemma 9.8.

∎

Man beachte, daß der Operator **V** essentiell für die Existenz einer Negationsnomalform ist.

9.1.1 Übungsaufgaben

omega-Strukturen

$$\begin{tikzcd}[row sep=small]
\mathbb{N} & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & \ldots & \text{n} \arrow[d] & \text{n+1} \arrow[d] & \ldots \\
2^{\Sigma} & \xi(0) & \xi(1) & \xi(2) & \ldots & \xi(n) & \xi(n+1) & \ldots
\end{tikzcd}$$

omega-Wörter

$$\begin{tikzcd}[row sep=small]
\mathbb{N} & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & \ldots & \text{n} \arrow[d] & \text{n+1} \arrow[d] & \ldots \\
V & w(0) & w(1) & w(2) & \ldots & w(n) & w(n+1) & \ldots
\end{tikzcd}$$

Abbildung 9.3: Korrespondez zwischen omega-Srukturen und *omega*-Wörtern

Übungsaufgabe 9.1.1
Wir betrachten die folgende Definition eines neuen temporalen Operators $\mathbf{U}^+$:

$\xi \models A\ \mathbf{U}^+\ B$ gdw es gibt $n \in \mathbb{N}$ mit $0 < n$ und $\xi_n \models B$ und
für alle m mit $0 < m < n$ gilt $\xi_m \models A$

1. Drücken Sie $\mathbf{U}^+$ im Vokabular von LTL aus.
2. Definieren Sie $\mathbf{U}$ und X durch $\mathbf{U}^+$.

Übungsaufgabe 9.1.2
Sei p ein aussagenlogisches Atom. Geben Sie eine LTL-Formel A_{2p}, so daß für jedes ξ gilt

$$\xi \models A_{2p} \quad \text{gdw} \quad (p \in \xi(n) \Leftrightarrow n \text{ ist gerade})$$

Übungsaufgabe 9.1.3
Zeigen Sie, daß die folgenden LTL-Formeln in allen Omega-Strukturen wahr sind

1. $\neg(B \mathbf{U} C) \leftrightarrow \neg C \mathbf{U}_w (\neg C \wedge \neg B)$
2. $\neg(B \mathbf{U}_w C) \leftrightarrow \neg C \mathbf{U} (\neg C \wedge \neg B)$
3. $B \mathbf{U} C \leftrightarrow C \vee (B \wedge X (B \mathbf{U} C))$
4. $\Diamond B \leftrightarrow (B \vee X \Diamond B)$
5. $\Box B \leftrightarrow (B \wedge X \Box B)$

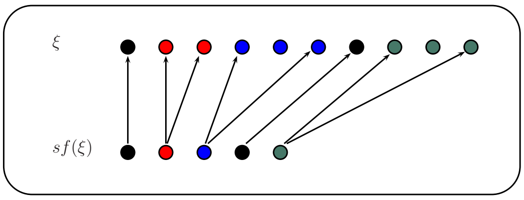


Abbildung 9.4: Die Abbildung an

Übungsaufgabe 9.1.4

Definition 9.10
Sei ξ eine omega-Struktur zur aussagenlogischen Signatur Σ. Ein *Stotterschritt* ist ein Paar von aufeinander folgenden Zeitpunkten, n, $n+1$, welche dieselben Atome aus Σ erfüllen, i.e. für alle $p \in \Sigma$ gilt $\xi(n) \models p \Leftrightarrow \xi(n+1) \models p$. Das ist nach Definition 9.3 gleichbedeutend mit $\xi(n) = \xi(n+1)$. Mit $sf(\xi)$ bezeichnen wir die omega-Struktur, die aus ξ entsteht, indem alle Stotterschritte entfernt werden. Wir müssen allerdings eine Ausnahme machen für den Fall, daß von einer Stelle an in ξ nur noch Stotterschritte auftreten. Diese bleiben dann erhalten.

In Formeln heißt das: Es gibt eine Funktion $an : \mathbb{N} \to 2^{\mathbb{N}}$, so daß für alle n gilt

1. für $n_1 < n_2$ gilt für alle $k_1 \in an(n_1)$, $k_2 \in an(n_2)$ stets $k_1 < k_2$
2. für alle n und alle $k_1, k_2 \in an(n)$ gilt $\xi(k_1) = \xi(k_2) = sf(\xi)(n)$,
3. für alle n für die $an(n)$ endlich ist gilt für alle $k_1 \in an(n)$, $k_2 \in an(n+1)$ stets $\xi(k_1) \neq \xi(k_2)$,

4. für alle n, so daß $an(n)$ unendlich ist gilt $an(n) = an(m)$ für alle $n \geq m$,
5. $\bigcup_{n\in\mathbb{N}} an(n) = \mathbb{N}$

Zwei omega-Strukturen ξ_1 und ξ_2 heißen *stotteräquivalent* falls $sf(\xi_1) = sf(\xi_2)$ gilt.

Zeigen Sie, daß für zwei stotteräquivalent omega-Strukturen ξ_1, ξ_2 und jede temporallogische Formel F, in der der Operator X nicht vorkommt, gilt:

$$\xi_1 \models F \quad \text{gdw} \quad \xi_2 \models F$$

Man sagt in diesem Fall, F ist eine stotterinvariante Formel.

Übungsaufgabe 9.1.5
Zeigen Sie, daß die Äquivalenzen 1 bis 3 aus Lemma 9.8 auch für alle Zeitstrukturen gelten.

Übungsaufgabe 9.1.6
In der Definition 9.4 von $\xi \models A$ auf Seite 310 wird in den rekursiven Schritten die omega-Struktur ξ geändert, sie wird z.B. ersetzt durch ξ_n. Eine Alternative dazu wäre die Definition von $(\xi, n) \models A$ mit der intuitiven Bedeutung, die Formel A ist wahr zum Zeitpunkt n in der omega-Struktur ξ. Wir haben uns für die Textversion entschieden, weil sie geringfügig weniger Schreibaufwand verursacht. Wie würde die rekursive Definition für die Alternative $(\xi, n) \models A$ aussehen?

Übungsaufgabe 9.1.7
Finden Sie eine LTL Formel F , die genau dann in einer omega-Struktur ξ wahr ist, wenn für jeden Zeitpunkt t_0, in dem p wahr ist, gilt:

1. Es gibt einen Zeitpunkt t_1 an dem q wahr ist und zwischen t_0 und t_1 ist q nicht wahr, und
2. es gibt einen Zeitpunkt t_1 an dem r wahr ist und zwischen t_0 und t_2 ist r nicht wahr, und
3. $t_1 < t_2$ ist.

Übungsaufgabe 9.1.8
Finden Sie eine LTL Formel F , die genau dann in einer omega-Struktur ξ wahr ist, wenn gilt

1. nach jedem Zeitpunkt t_0, in dem p gilt, kommt ein Zeitpunk t_1 mit $t_0 < t_1$, in dem q wahr ist und

2. gilt ein einem Zeitpunkt t_0 die Aussage p, dann gibt es bis zum nächsten Zeitpunkt t_1, $t_0 < t_1$ zu dem q wahr ist, genau einen Zeitpunkt, in dem r wahr ist.

Übungsaufgabe 9.1.9
Zeigen Sie, daß die folgenden Formeln Tautologien sind, genauer, daß sie in allen omega-Strukturen wahr sind.

1. $(A \mathbf{V} B) \mathbf{U} C \leftrightarrow C \ \vee \ (B \mathbf{U} (C \wedge (A \mathbf{V} B))) \ \vee \ (B \mathbf{U} (A \wedge B \wedge XC))$

2. $(\Box A) \mathbf{U} B \leftrightarrow B \vee (\Box A \wedge \Diamond B)$

3. $(C \vee \Box A) \mathbf{U} B \leftrightarrow B \vee (C \mathbf{U} B) \vee (C \mathbf{U} (\Box A \wedge \Diamond B))$

Übungsaufgabe 9.1.10
Jede LTL-Formel F, welche neben den aussagenlogischen Operatoren die temporalen Operatoren $\mathbf{U}$, $\mathbf{V}$, X , $\Box$ und $\Diamond$ benutzt, lässt sich leicht in eine äquivalente Negationsnormalform transformieren, dh. in eine Formel, in der Negationszeichen nur vor atomaren Formeln vorkommen. Man braucht nur wiederholt die Tautologien $\neg(A \mathbf{U} B) \leftrightarrow \neg A \mathbf{V} \neg B$, $\neg(A \mathbf{V} B) \leftrightarrow \neg A \mathbf{U} \neg B$, $\neg\Box A \leftrightarrow \Diamond\neg A$, $\neg\Diamond A \leftrightarrow \Box\neg A$ und $\neg X\ A \leftrightarrow X\neg A$ anzuwenden.
Frage: Kann es zu jeder Formel auch eine äquivalente Negationsnormalform geben, in der der Operator $\mathbf{V}$ nicht vorkommt?

Zeigen Sie, daß die folgende Formel in allen omega-Strukturen wahr ist.

$$A \mathbf{V} B \leftrightarrow \Box B \vee (B \mathbf{U} (A \wedge B))$$

Jetzt klar, daß die Antwort auf obige Frage *Ja* ist. Man transformiert F zuerst, wie gehabt, in eine Negationsnormalform. Die darin vorkommenden $\mathbf{V}$ Operatoren eliminiert man mit der gerade gezeigten Tautologie, wobei man beachtet, daß dabei keine neuen Negationszeichen eingeführt werden.

Übungsaufgabe 9.1.11
Zeigen Sie, daß die folgende Formel eine LTL-Tautologie ist

$$(A \mathbf{U} B) \mathbf{U} C \ \equiv \ C \vee (A \vee B) \mathbf{U} ((B \wedge X\ C) \vee (C \wedge (A \mathbf{U} B)))$$

für beliebige LTL-Formeln A; B, C.

Übungsaufgabe 9.1.12
Zur Formulierung dieser Übungsaufgabe brauchen wir zunächst zwei Definitionen.

Definition 9.11
Die Klasse $leLTL$ der *linkseinfachen* LTL-Formel ist induktive definiert durch:

1. jedes ausagenlogische Literal liegt in $leLTL$,
2. liegen F_1, F_2 in $leLTL$, dann auch $F_1 \wedge F_2$ und $F_1 \vee F_2$,
3. ist F in $leLTL$ und G eine beliebige LTL-Formel, dann ist auch $F\ \mathbf{U}\ G$ in $leLTL$.

Definition 9.12
Wir sagen, daß eine LTL Formel F die *Diskunktioneigenschaft* hat, wenn für jede omega-Struktur ξ für die $\xi \models \Box F$ gilt, auch gilt $\xi^1 \models \Box(F \vee a)$, wobei a ein aussagenlogisches Atom ist, das in F nicht vorkommt und die omega-Struktur ξ^1 definiert ist durch

$$\xi^1(n) = \begin{cases} \xi(2m) & \text{falls } n = 2m \\ \{a\} & \text{falls } n = 2m+1 \end{cases}$$

`muß noch vervollständig werden`

Übungsaufgabe 9.1.13
Zeigen Sie, daß die folgende Formel eine Tautologie ist

$$\Box(\neg r \to X\ r) \to \Diamond(r \wedge X\ X\ r)$$

Kapitel 10

Stärkere Logiken

10.1 Prädikatenlogik zweiter Ordnung

Bemerkung 10.1
Die Prädikatenlogik der zweiten Ordnung, kurz PL2, unterscheidet sich dadurch von der PL1, daß es nun auch Variable (über die man quantifizieren kann) für Funktionen und Prädikate gibt. In der vollen Ausbaustufe hätte man also zu jeder Stelligkeit von Prädikaten unendlich viele Prädikatenvariable, und entsprechend Funktionsvariable. Zusätzlich kann die Signatur fest zu interpretierende Symbole für Operatoren verschiedenen Typs enthalten: z. B. für Funktionale, die Funktionen auf Funktionen abbilden; für Relationen (bestimmter Stelligkeit) über Relationen (bestimmter Stelligkeiten) und/oder Funktionen (bestimmter Stelligkeiten), etc.. Wir werden das hier nicht weiter ausführen. Es zeigt sich nämlich, daß die kennzeichnenden Phänomene der zweiten Stufe bereits bei einem eher bescheidenen sprachlichen Rahmen auftreten (und trivialerweise bei Erweiterung erhalten bleiben).

Wir beschränken uns im folgenden auf Sprachen zweiter Ordnung, bei denen zusätzlich zur Ausstattung der PL1 noch Variable für einstellige und für zweistellige Prädikate (Relationen) vorhanden sind. (Insbesondere gibt es keine Funktionsvariablen und keine Signatursymbole „von höherem Typ".)

Definition 10.2
(Syntax der PL2 (in der hier betrachteten Ausbaustufe))

Sonderzeichen:
Wie in der PL1, also: $(,), \doteq, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$.

Variable:

$Var =$	$Ivar \cup Mvar \cup Rvar$ (paarweise disjunkt)
$Ivar$:	unendlich viele *Individuenvariable* $v_0, v_1, \ldots$ Notation: $x, y, z, \ldots$
$Mvar$:	unendlich viele *Mengenvariable* oder *einstellige Prädikatvariable* $M_0, M_1, \ldots$ Notation: $X, Y, Z, \ldots$
$Rvar$:	unendlich viele *Relationenvariable* oder zweistellige *Prädikatvariable* $R_0, R_1, \ldots$ Notation: $U, V, \ldots$

Signatur
$\Sigma = (F_\Sigma, P_\Sigma, \alpha_\Sigma)$ wie in der PL1

Terme
Term_Σ wie in der PL1

atomare Formeln:
$s \doteq t$ für Terme s, t

$p(t_1, \ldots, t_n)$ für $p \in P_\Sigma$, $\alpha_\Sigma(p) = n$, $t_1, \ldots, t_n \in \mathrm{Term}_\Sigma$
$X(t)$ für Mengenvariable X und Terme t
$U(s,t)$ für Relationenvariable U und Terme s, t

Formeln
For^2_Σ (kurz For_Σ oder For) enthält genau

- alle atomaren Formeln
- <u>true</u>, <u>false</u>
- mit $A, B \in For^2_\Sigma, x \in Ivar, X \in Mvar, U \in Rvar$
  auch: $\neg A, (A \wedge B), (A \vee B), (A \to B), (A \leftrightarrow B), \forall xA,$
  $\exists xA, \forall XA, \exists XA, \forall UA, \exists UA$

Die Begriffe „freie Variable“, „gebundene Variable“, „Substitution“, „kollisionsfreie Substitution“, „Präfix“, „Allabschluß“, „Existenzabschluß“ u.ä. werden entsprechend der PL1 gebildet.

Definition 10.3
(Semantik der PL2 (in der hier betrachteten Ausbaustufe))

Zu einer Interpretation (D, I) (wie in der PL1) sind jetzt neben Belegungen $\beta : Ivar \to D$ auch Belegungen $\gamma : Mvar \to P(D)$ und $\delta : Rvar \to P(D \times D)$ zu betrachten (P: Potenzmenge). Zu $D, I, \beta, \gamma, \delta$ ist $val_{D,I,\beta,\gamma,\delta}$ entsprechend dem Vorgehen in der PL1 definiert.

Auf Termen: $val_{D,I,\beta,\gamma,\delta}(t) = val_{D,I,\beta}(t)$ wie in der PL1 (hängt nicht ab von γ, δ).

Auf Formeln:

$$\left.\begin{array}{l} val_{D,I,\beta,\gamma,\delta}(p(t_1, \ldots, t_n)) \\ val_{D,I,\beta,\gamma,\delta}(s \doteq t) \end{array}\right\} \begin{array}{l} \text{Wie in der PL1} \\ \text{(Hängt nicht von } \gamma, \delta \text{ ab)} \end{array}$$

$$\begin{array}{lcl} val_{D,I,\beta,\gamma,\delta}(X(t)) = W & \Leftrightarrow & val_{D,I,\beta,\gamma,\delta}(t) \in \gamma(X) \\ val_{D,I,\beta,\gamma,\delta}(U(s,t)) = W & \Leftrightarrow & (val_{D,I,\beta,\gamma,\delta}(s), val_{D,I,\beta,\gamma,\delta}(t)) \in \delta(U) \end{array}$$

$$\left.\begin{array}{l} val_{D,I,\beta,\gamma,\delta} \text{ auf } \underline{\text{true}}, \underline{\text{false}}, \\ \neg A, (A \wedge B), (A \vee B), (A \to B), (A \leftrightarrow B), \forall xA, \exists xB \end{array}\right\} \text{Wie in der PL1}$$

$$val_{D,I,\beta,\gamma,\delta}(\forall XA) = W \quad \Leftrightarrow \quad \begin{array}{l} \text{Für jede Modifikation } \gamma^M_X \text{ von} \\ \gamma \text{ gilt } val_{D,I,\beta,\gamma^M_X,\delta}(A) = W \end{array}$$

$$val_{D,I,\beta,\gamma,\delta}(\forall U A) = W \quad \Leftrightarrow \quad \text{Für jede Modifikation } \delta_U^R \text{ von } \delta \text{ gilt } val_{D,I,\beta,\gamma,\delta_U^R}(A) = W$$

Existenzoperatoren: Entsprechend mit „Es gibt ...“.
(Modifikationen sind entsprechend der Begriffsbildung bei Belegungen von Individuenvariablen definiert.)

(D, I) heißt *Modell* von A $:\Leftrightarrow$ $val_{D,I,\beta,\gamma,\delta}(A) = W$ für alle β, γ, δ. (D, I) ist Modell einer Formelmenge M $:\Leftrightarrow$ (D, I) ist Modell jeder Formel in M.

$M \models A$ $:\Leftrightarrow$ Jedes Modell von M ist Modell von A
A *allgemeingültig* $:\Leftrightarrow$ $\models A$ (d. h. $\emptyset \models A$)
A *erfüllbar* $:\Leftrightarrow$ $\neg A$ ist nicht allgemeingültig.
(D.h.: Es gibt $D, I, \beta, \gamma, \delta$ mit $val_{D,I,\beta,\gamma,\delta}(A) = W$)

Bemerkung 10.4
Auf der zweiten Stufe kann es Formeln geben, in denen kein Signatursymbol und auch nicht $\doteq$ auftritt, z. B. $X(y)$. Ist eine solche Formel geschlossen, dann hängt ihr Wahrheitswert bei einer Interpretation (D, I) (nicht nur nicht von Belegungen, sondern auch) höchstens von D, nicht von I ab. Z.B. ist $\exists X \exists y X(y)$ allgemeingültig, $\forall X \exists y X(y)$ unerfüllbar, $\exists X(\exists y X(y) \wedge \exists y \neg X(y))$ gilt genau bei den (D, I) mit $\sharp D \geq 2$.

Satz 10.5
In der PL2 ist die Gleichheit durch eine Formel charakterisierbar.

Beweis
Mit $G := \forall X(X(x) \leftrightarrow X(y))$ gilt $\models x \doteq y \leftrightarrow G$.

Satz 10.6
Durch das Peano'sche Axiomensystem $P1, \ldots, P5$ in der PL2 sind die natürlichen Zahlen mit Addition und Multiplikation bis auf Isomorphie eindeutig charakterisiert.

Beweis
Das allgemeine Induktionsschema

$$\forall X((X(O) \wedge \forall y(X(y) \rightarrow X(S(y)))) \rightarrow \forall y X(y))$$

hat genau die durch O und S termerzeugten Interpretationen zu Modellen.

Übungsaufgabe 10.1.1
Über einer endlichen Signatur ist die Termerzeugtheit durch eine Formel der PL2 charakterisierbar. (Man orientiere sich an obigem Beispiel.)

Satz 10.7
In der PL2 ist die Endlichkeit einer Interpretation durch eine Formel charakterisierbar.

Beweis
Es gilt
(D, I) endlich
$\Leftrightarrow$ Jede injektive Funktion $F : D \to D$ ist auch surjektiv
$\Leftrightarrow$ Für jede Relation $R \subseteq D \times D$: Wenn R der Graph einer injektiven Funktion ist, dann ist diese auch surjektiv

Als charakterisierende Formel wählen wir

$$\begin{array}{lll} Fin & := & \forall U \quad ((\forall x \exists y U(x,y) \wedge \forall x \forall y \forall z (U(x,y) \wedge U(x,z) \to y \doteq z) \\ & & \qquad \wedge \forall x \forall y \forall z (U(x,z) \wedge U(y,z) \to x \doteq y)) \\ & & \qquad \to \forall y \exists x U(x,y)) \end{array}$$

Satz 10.8
Die PL2 ist nicht kompakt. D. h.:

- Aus $M \models A$ braucht i. a. nicht zu folgen:
  $E \models A$ für eine endliche Teilmenge E von M
- Es kann sein, daß jede endliche Teilmenge einer Formelmenge M ein Modell hat, diese selbst aber nicht.

Beweis
Im Beweis der Nichtcharakterisierbarkeit des Begriffs „endlich“ auf der ersten Stufe (Satz 5.85) wurde von der PL1 nur verwendet, daß sie kompakt ist. Wäre die PL2 kompakt, dann ließe sich dieser Beweis wörtlich übertragen, im Widerspruch zu Satz 10.7.

Satz 10.9
Für die PL2 kann es keinen korrekten und vollständigen Kalkül geben.

Beweis
Der Begriff der Ableitbarkeit aus einem Kalkül K ist stets kompakt, d. h.

$$M \vdash_K A \Rightarrow E \vdash_K A \text{ für eine endliche Teilmenge } E \text{ von } M.$$

Die Existenz eines korrekten und vollständigen Kalküls stünde also im Widerspruch zu Satz 10.8

10.1.1 Übungsaufgaben

Übungsaufgabe 10.1.2
Gegeben sei eine Signatur Σ mit einem zweistelligen Relationszeichen r, einem einstellige Relationszeichen v und Konstantenzeichen a und b. Für eine Σ-Struktur $\mathcal{G} = (G, R, c_a, c_e, V)$, dabei sei R die Interpretation von r, $(R = I(r))$ und $I(a) = c_a$, $I(b) = c_b$, und $I(v) = V$. definieren wir:

Definition 10.10

1. Ein *R-Pfad* ist eine Folge $a_1, \ldots, a_n$ $(n \geq 1)$ von Elementen von G, so daß für alle $1 \leq i < n$ gilt $R(a_i, a_{i+1})$ und $a_1 = c_a$ und $a_n = c_e$.

2. Wir nennen eine Teilmenge $V \subseteq G$ eine Verbindungsmenge, wenn es einen *R-Pfad* $a_1, \ldots, a_n$ gibt, so daß für alle $1 \leq i \leq n$ gilt $a_i \in V$.

3. Eine Verbindungsmenge heißt eine *minimale* Verbindungsmenge , wenn jede echte Teilmenge $V_0 \subset V$ keine Verbindungsmenge ist.

Geben Sie eine Σ-Formel ϕ_{VM} der Logik zweiter Stufe an, so daß $(G, R, c_a, c_e, V) \models \phi_{VM}$ genau dann gilt, wenn V eine minimale Verbindungsmenge ist.

10.2 Dynamische Logik

Bemerkung 10.11
(Modale Prädikatenlogik 1. Ordnung).
...

Bemerkung 10.12
(Dynamische Logik)

Die dynamische Logik (DL) ist eine Modallogik, jedoch mit einer Semantik, bei der für den Übergang zwischen Zuständen (Welten) nicht eine einzige, feste Relation gegeben ist, sondern gleichzeitig unendlich viele Relationen. Welche solche Relation jeweils zu wählen ist, läßt sich der auszuwertenden Formel entnehmen. Formeln können nämlich *Programme* einer bestimmten Programmiersprache enthalten, und zu jedem Programm gehört eine Übergangsrelation zwischen Zuständen. Dabei sind die Zustände in einer von der Programmiersprache (und der genaueren Aufbereitung der Semantik) abhängigen Weise eng verknüpft mit den Belegungen der Variablen. Man kann dann auf diese Weise auch reiche und komplizierte Programmiersprachen axiomatisch erfassen. Auf die Vielfalt dieser Möglichkeiten gehen wir hier nicht ein, sondern behandeln die Grundidee anhand einer besonders einfachen Programmiersprache, nämlich „elementarem PASCAL“, einer Sprache mit Zuweisung, Verzweigung und while-Schleife. Hier kann man als Zustände einfach die Variablenbelegungen selber wählen. Auch handelt es sich um eine deterministische Sprache; in allgemeineren Versionen ist die DL auch für Sprachen mit indeterministischen Programmkonstrukten betrachtet worden.

Die DL geht (in etwas anderer Version als hier dargestellt) zurück auf HAREL, sie wurde wesentlich ausgearbeitet in [Gol82].

Definition 10.13
(Syntax der DL)

Sonderzeichen: Wie in der PL1, zusätzlich $[,], <, >$, <u>skip</u>, <u>abort</u>, :=, ; , <u>if</u>, <u>then</u>, <u>else</u>, <u>while</u>, <u>do</u>

$$\left.\begin{array}{l} \textit{Variable}, Var \\ \textit{Signatur}, \Sigma \\ \textit{Terme}, \mathrm{Term}_\Sigma \\ \textit{atomare Formeln} \end{array}\right\} \text{wie in der PL1}$$

Boolesche Ausdrücke, $Bexp_\Sigma$, sind die quantorenfreien PL1-Formeln über Σ. Sie werden aus den atomaren Formeln und <u>true</u>, <u>false</u> durch die aussagenlogischen Operatoren gebildet.

Programme, $Prog_\Sigma$:

- <u>skip</u> und <u>abort</u> sind Programme
- Mit $x \in Var$ und $t \in \mathrm{Term}_\Sigma$ ist
  $(x := t)$ ein Programm (Zuweisung)
- Sind π, ρ Programme und ist ϵ ein Boolescher Ausdruck, dann sind auch Programme:
  $(\pi; \rho)$ (Hintereinanderausführung)
  <u>if</u> ϵ <u>then</u> π <u>else</u> ρ (Verzweigung)
  <u>while</u> ϵ <u>do</u> π (Schleife)

(Man beachte, daß wegen der Klammerung der Hintereinanderausführung eine schließende Klammer für Verzweigung bzw. Schleife unnötig ist.)

Formeln, For_Σ:

- Alle Booleschen Ausdrücke sind Formeln
- Mit $A, B \in For_\Sigma, x \in Var, \pi \in Prog_\Sigma$
  sind auch Formeln:
  $\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B)$,
  $\forall x A, \exists x A, [\pi]A, \langle\pi\rangle A$.

Definition 10.14
(Semantik der DL)
Zu einer Interpretation (D, I) über Σ und einem Zustand β, d.h. einer Belegung der Variablen, sind $val_{D,I,\beta}(t)$ für Terme t und $val_{D,I,\beta}(\epsilon)$ für Boolesche Ausdrücke ϵ definiert wie in der PL1.

Es sei $Sta := (Var \to D)$ die Menge der Zustände. Die *Semantik der Programme* wird festgelegt durch die Definition einer Relation $\rho(\pi) \subseteq Sta \times Sta$ für jedes $\pi \in Prog_\Sigma$. ($\rho(\ldots)$ hängt von (D, I) ab, aus Gründen der einfacheren Notation schreiben wir aber einfach $\rho(\pi)$ statt $\rho(\pi)_{D,I}$.)

$\rho(\underline{abort})$ ist die leere Relation

$\beta\rho(\underline{skip})\gamma$	$:\Leftrightarrow$	$\beta = \gamma$
$\beta\rho(x := t)\gamma$	$:\Leftrightarrow$	$\gamma = \beta_x^{val_{D,I,\beta}(t)}$
$\beta\rho(\pi_1; \pi_2)\gamma$	$:\Leftrightarrow$	Es gibt $\delta \in Sta$ mit$\beta\rho(\pi_1)\delta$ und $\delta\rho(\pi_2)\gamma$
$\beta\rho(\underline{\text{if}}\ \epsilon\ \underline{\text{then}}\ \pi_1\ \underline{\text{else}}\ \pi_2)\gamma$	$:\Leftrightarrow$	$(val_{D,I,\beta}(\epsilon) = W$ und $\beta\rho(\pi_1)\gamma)$ oder $(val_{D,I,\beta}(\epsilon) = F$ und $\beta\rho(\pi_2)\gamma)$
$\beta\rho(\ \underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi)\gamma$	$:\Leftrightarrow$	Es gibt $n \in I\!N$ und Zustände $\beta_0, \ldots, \beta_n$ mit: - $\beta_0 = \beta, \beta_n = \gamma$ - $\forall i$ mit $0 \leq i < n : val_{D,I,\beta_i}(\epsilon) = W$ und $\beta_i\rho(\pi)\beta_{i+1}$ - $val_{D,I,\beta_n}(\epsilon) = F$

Zu gegebenem π und β gibt es stets höchstens ein γ mit $\beta\rho(\pi)\gamma$: Die Relation $\rho(\pi)$ ist rechtseindeutig und hätte auch als partielle Funktion formuliert werden können. Das würde sich ändern, wenn man auch indeterministische Programme zuläßt.

$\beta\rho(\pi)\gamma$ besagt, daß π, beginnend im Zustand β, terminiert und zum Zustand γ führt. π, beginnend in β, terminiert genau dann, wenn es ein solches γ gibt. Etwa terminiert abort in keinem Zustand, ebenso while true do ρ (für beliebiges ρ). Das Programm while ϵ do skip terminiert genau bei Beginn in Zuständen β mit $val_{D,I,\beta}(\epsilon) = F$ (und dann nach 0 Schritten).

Formeln

Für Boolesche Ausdrücke ϵ ist nach Annahme $val_{D,I,\beta}(\epsilon)$ bereits definiert.

Durch strukturelle Induktion wird $val_{D,I,\beta}$ für Formeln

$$\neg A, (A \wedge B), (A \vee B), (A \to B), (A \leftrightarrow B), \forall x A, \exists x A$$

definiert entsprechend dem Vorgehen in der PL1.

$$val_{D,I,\beta}([\pi]A) = W :\Leftrightarrow \text{ Für jedes } \gamma \text{ mit } \beta\rho(\pi)\gamma \text{ gilt } val_{D,I,\gamma}(A) = W$$

$$val_{D,I,\beta}(\langle\pi\rangle A) = W :\Leftrightarrow \text{ Es gibt } \gamma \text{ mit } \beta\rho(\pi)\gamma \text{ und } val_{D,I,\gamma}(A) = W$$

Man bemerke, daß (in unserem Fall von deterministischen Programmen) gilt: $val_{D,I,\beta}([\pi]A) = W$ genau dann, wenn: Falls ein γ existiert mit $\beta\rho(\pi)\gamma$, dann ist für dieses (dann eindeutig bestimmte) γ $val_{D,I,\gamma}(A) = W$.

(D, I) heißt *Modell* von A, wenn $val_{D,I,\beta}(A) = W$ für alle Zustände β. Entsprechend sind wie in der PL1 erklärt: Modell einer Formelmenge M, allgemeingültig, erfüllbar, sowie die Folgerbarkeitsbeziehung $\models$.

Bemerkung 10.15
(Zur Semantik der DL)
$\langle \ldots \rangle$ ist entbehrlich, denn es gilt für beliebige D, I, β und beliebige π, A:

$$val_{D,I,\beta}(\langle\pi\rangle A) = val_{D,I,\beta}(\neg[\pi]\neg A),$$

d. h. $\langle\rangle$ läßt sich stets durch $\neg[]\neg$ ersetzen.

Die *partiellen Korrektheitsaussagen der HOARE-Logik*

> $A\{\pi\}B$, d.h.
> „Wenn in einem Zustand A gilt, und wenn π, ab diesem Zustand rechnend, terminiert, dann gilt im erreichten Zustand B"

lassen sich in der DL wiedergeben. Es ist nämlich $A\{\pi\}B$ gleichbedeutend mit

$$A \to [\pi]B.$$

Das ist die *partielle Korrektheit* von π bzgl. der *Vorbedingung* A und der *Nachbedingung* B. Entsprechend läßt sich die *totale Korrektheit* ausdrücken:

$$A \to \langle\pi\rangle B$$

besagt (d.h. wird zu W ausgewertet unter genau denjenigen D, I, β, für welche gilt): Falls A gilt (d.h. $val_{D,I,\beta}(A) = W$), terminiert π und im dann erreichten Zustand gilt B ($val_{D,I,\gamma}(B) = W$ für dieses γ).

Insbesondere besagt $[\pi]A$ (gleichbedeutend mit $\underline{\text{true}} \to [\pi]A$), daß immer, wenn π terminiert, hernach A gilt. Und $\langle\pi\rangle A$ besagt: π terminiert immer, und hernach gilt A.

Ferner: $A \to \langle\pi\rangle\ \underline{\text{true}}$ besagt: Wenn A gilt (d.h. rechnend ab Zuständen, in denen A zu W auswertet) terminiert π. $\langle\pi\rangle\ \underline{\text{true}}$ besagt, daß π terminiert (d.h. $val_{D,I,\beta}(\langle\pi\rangle\ \underline{\text{true}} = W \Leftrightarrow \pi$ in Interpretation (D, I) und rechnend ab β terminiert). $\neg\langle\pi\rangle\ \underline{\text{true}}$, d.h. $[\pi]\ \underline{\text{false}}$, besagt: π terminiert nicht.

Man beachte, daß die DL im Gegensatz zu den HOARE'schen Korrektheitsaussagen eine *volle* Logik ist, sie ist abgeschlossen gegenüber aussagenlogischen Operatoren, Quantoren und Programmeinfügungen.

Satz 10.16
Die natürlichen Zahlen mit Addition und Multiplikation lassen sich in der DL bis auf Isomorphie eindeutig charakterisieren.

Beweis
Die Formel

$$\forall x \langle (y := 0;\ \underline{\text{while}}\ \neg y \doteq x\ \underline{\text{do}}\ y := S(y)) \rangle\ \underline{\text{true}}$$

drückt die Termerzeugtheit durch Terme über O, S aus. Zusammen mit $Pe1, \ldots, Pe4$ charakterisiert sie also das Modell $(I\!N, I_0)$ bis auf Isomorphie eindeutig

Satz 10.17
Die DL ist nicht kompakt.

Beweis
Über einer Signatur mit zwei Konstanten a, b und einem einstelligen Funktionssymbol f bilden wir die Formeln $A_0, A_1, \ldots$:

$$\begin{aligned} A_0 &= \neg y \doteq b \\ A_{n+1} &= [y := f(y)]A_n. \end{aligned}$$

A_n sagt über die Schleife $\underline{\text{while}}\ \neg y \doteq b\ \underline{\text{do}}\ y := f(y)$, daß sie nicht nach genau n Schritten terminiert.

Ferner sei für $n = 0, 1, \ldots$:

$$B_n = [y := a]A_n.$$

B_n besagt: Nach Initialisierung von y mit a hat die Schleife $\underline{\text{while}}\ \neg y \doteq b\ \underline{\text{do}}$ $y := f(y)$ nicht die Eigenschaft, nach genau n Schritten zu terminieren (und damit von a aus durch genau n Anwendungen von f b zu erreichen).

Schließlich sei C die Formel

$$\langle y := a;\ \underline{\text{while}}\ \neg y \doteq b\ \underline{\text{do}}\ y := f(y) \rangle\ \underline{\text{true}}.$$

C sagt, daß die Schleife bei Initialisierung von y mit a terminiert.

Jede endliche Teilmenge E von $\{B_n \mid n \in I\!N\} \cup \{C\}$ hat ein Modell, nämlich $(I\!N, I)$ mit $I(f)(n) = n + 1$, $I(a) = 0$ und $I(b) = max\{B_n \mid n \in I\!N\} + 1$.

Aber $\{B_n \mid n \in I\!N\} \cup \{C\}$ hat kein Modell. Denn in einem solchen, (D, I),

- müßte man einerseits, beginnend mit $I(a)$, durch sukzessives Anwenden von $I(f)$ das Element $I(b)$ von D in endlich vielen Schritten erreichen (Modell von C),

- andererseits gäbe es kein n, so daß man von $I(a)$ aus durch n Anwendungen von $I(f)$ das Element $I(b)$ erreicht (Modell von $\{B_n \mid n \in \mathbb{N}\}$).

Korollar 10.18
Für die DL gibt es keinen korrekten und vollständigen Kalkül

Bemerkung 10.19
(Analyse der While-Schleife)

Wie das Beispiel im Beweis des Satzes 10.17 verdeutlicht, ist die „natürliche Semantik“ der While-Schleife der Grund für die Nichtkompaktheit der DL. Wir analysieren diese Semantik etwas genauer. Das eröffnet den Zugang zu einem „infinitären Kalkül“. Bei dieser (nicht mehr „konstruktiven“) Ausweitung unseres bisherigen Kalkülbegriffs bleibt dennoch ein Teil dessen erhalten, was man sich unter der Arbeit mit einem Kalkül vorstellt.

Definition 10.20
Zu $\epsilon \in Bexp_\Sigma, \pi \in Prog_\Sigma, A \in For_\Sigma$ definieren wir für alle $n \in \mathbb{N}$ Formeln $Loop_n(\epsilon, \pi, A)$ wie folgt.

$Loop_0(\epsilon, \pi, A) = \neg\epsilon \to A$
$Loop_{n+1}(\epsilon, \pi, A) = \epsilon \to [\pi]Loop_n(\epsilon, \pi, A)$

Korollar 10.21
Zu ϵ, π, A sowie einer Interpretatin (D, I) und einem Zustand β sei $\beta_0, \beta_1, \ldots$ die eindeutig bestimmte Folge von Zuständen mit $\beta_0 = \beta, \beta_i \rho(\pi) \beta_{i+1}$. Dann gilt für jedes n:
$val_{D,I,\beta}(Loop_n(\epsilon, \pi, A)) = W$
$\Leftrightarrow$ Wenn $val_{D,I,\beta_i}(\epsilon) = W$ für alle $i < n$ und $val_{D,I,\beta_n}(\epsilon) = F$, dann $val_{D,I,\beta_n}(A) = W$.

Beweis
Leichte Induktion nach n.

$Loop_n(\epsilon, \pi, A)$ besagt also: Wenn die Schleife <u>while</u> ϵ <u>do</u> π nach genau n Schritten terminiert, dann gilt hernach A.

Lemma 10.22
Für beliebige D, I, β gilt
$val_{D,I,\beta}([\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A) = W$
$\Leftrightarrow$ Für alle $n \in \mathbb{N} : val_{D,I,\beta}(Loop_n(\epsilon, \pi, A)) = W$

Beweis
Wir zeigen: $val_{D,I,\beta}([\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A) = W$ gdw.: Für alle n ist das (zu $val_{D,I,\beta}(Loop_n(\epsilon, \pi, A)) = W$ äquivalente) Kriterium aus Korollar 10.21 erfüllt.

Gelte $val_{D,I,\beta}([\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A) = W$, und sei $n \in I\!N$ gegeben. Wenn die Schleife nicht terminiert, d.h. $val_{D,I,\beta_i}(\epsilon) = W$ für alle i, dann $val_{D,I,\beta_n}(\epsilon) = W$, das Kriterium ist erfüllt. Wenn für ein m gilt $val_{D,I,\beta_m}(\epsilon) = F$ und $val_{D,I,\beta_i}(\epsilon) = W$ für alle $i < m$, dann (wegen $val_{D,I,\beta}([\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A) = W$) $val_{D,I,\beta_m}(A) = W$. Falls $n < m$: $val_{D,I,\beta_n}(\epsilon) = W$. Falls $n > m$: Nicht gilt für alle $i < n$, daß $val_{D,I,\beta_i}(\epsilon) = W$ (für $i = m$ nicht). Falls $n = m$: $val_{D,I,\beta_n}(A) = W$. In allen drei Fällen ist also das Kriterium für n erfüllt.

Gelte nun umgekehrt $val_{D,I,\beta}([\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A) = F$. Für das dann existierende m mit $val_{D,I,\beta_m}(\epsilon) = F$ und $val_{D,I,\beta_i}(\epsilon) = W$ für $i < m$ gilt $val_{D,I,\beta_m}(A) = F$. Für $n = m$ ist das Kriterium also nicht erfüllt.

Bemerkung 10.23
(Infinitäre Kalküle)

Aus Lemma 10.22 folgt

$$\{Loop_n(\epsilon, \pi, A) \mid n \in I\!N\} \models [\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A.$$

Es ist also sozusagen

$$\frac{\{Loop_n(\epsilon, \pi, A) \mid n \in I\!N\}}{[\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A}$$

für beliebiges ϵ, π, A eine „korrekte Regel“. Es ist keine Regel in unserem bisherigen Sinne, da sie unendlich viele Prämissen hätte. Wir bezeichnen sie als *infinitäre* Regel im Gegensatz zu den uns bekannten *finitären* Regeln. Eine infinitäre Regel kann nicht in der bekannten Weise zum Ableiten verwendet werden. Denn da eine Ableitung stets nur endlich lang ist, können niemals an einer bestimmten Stelle alle Prämissen der Regel produziert worden sein.

Also wird, abstrakter, zu einem „Kalkül“ K, der auch infinitäre Regeln enthalten kann, die Theoremmenge $Th_K(M)$ definiert als die kleinste Formelmenge, die M und die Axiome enthält, und die gegen die Regeln abgeschlossen ist. (Letzteres bedeutet: Wenn alle Prämissen einer Instanz der Regel in der Menge liegen, dann auch die Conclusio.) Man definiert $M \vdash_K A :\Leftrightarrow A \in Th_K(M)$.

Bemerkung 10.24
(Ein infinitärer Kalkül für die DL)

Auf der Suche nach einem korrekten und vollständigen *infinitären Kalkül* für die DL (d.h. einem solchen mit infinitären Regeln) muß man die obige infinitäre Regel noch etwas verallgemeinern, nämlich so, daß die Formeln $Loop_n(\epsilon, \pi, A)$ auch innerhalb bestimmter anderer Formeln als „Approximationen“ von $[\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A$ verwendet werden dürfen. Z.B. wäre

$$\frac{\{B \rightarrow Loop_n(\epsilon, \pi, A) \mid n \in I\!N\}}{B \rightarrow [\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A}$$

eine entsprechende Regel. Die Analyse des Sachverhalts liefert eine bestimmte Klasse zulässiger solcher Zusammensetzungen. Sie ist definiert durch eine Klasse **S** von *zulässigen Schemata* oder *zulässigen Formen*. Das sind syntaktische Gebilde, die eine *Metavariable* (einen Platzhalter) X enthalten, so daß jeweils durch Ersetzung von X durch eine Formel (an allen Stellen, und dann überall durch dieselbe Formel) eine Formel entsteht. Wir schreiben $\Phi(X)$ für ein solches Schema, und entsprechend $\Phi(A)$ für das Ersetzungsresultat von X durch A. Die Klasse **S** der *zulässigen Formen* ist dann definiert durch

- $X \in \mathbf{S}$
- mit $\Phi(X)$ enthält **S** auch $B \rightarrow \Phi(X)$, $\forall x \Phi(X)$, $[\pi]\Phi(X)$ (für Formeln B, Variable x, Programme π).

Anmerkung (**S** als Klasse *stetiger* Formen)
...

Hiermit definiert man nun die folgende infinitäre Regel, die *Omega-Iteration*:

$$0I\colon \frac{\{\Phi(Loop_n(\epsilon, \pi, A) \mid n \in I\!N\}}{\Phi([\underline{\text{while}}\ \epsilon\ \underline{\text{do}}\ \pi]A)} \qquad \text{für } \Phi \in \mathbf{S}.$$

Es gilt nun der folgende

Satz (Goldblatt 1982)
Für die DL gibt es einen korrekten und vollständigen infinitären Kalkül, der neben $0I$ nur finitäre Regeln enthält.

(Dabei haben wir die Axiome auch als Regeln, nämlich nullstellige Regeln, aufgefaßt.)

Es bleibt die Frage, wie man hiermit verfährt; denn „in normaler Weise ableiten“ kann man ja mit einer infinitären Regel nicht. Eine naheliegende Heuristik besteht darin, zu versuchen, die unendlich vielen Formeln $\Phi(Loop_n(\epsilon, \pi, A))$, $n \in I\!N$, durch Induktion über n zu beweisen. Zu diesem Zweck wird eine Zählerstruktur (Terme über O, S) in die Signatur fest eingebaut und die Formeln $\Phi(Loop_n(\epsilon, \pi, A))$ werden insgesamt durch eine Formel mit einer Zählervariablen wiedergegeben. Man kann ein zugehöriges Induktionsschema angeben und hat hiermit eine finitäre Approximation an eine Instanz von OI. Man erhält auf diese Weise einen korrekten finitären Kalkül. Dieser ist notwendigerweise unvollständig, jedoch durchaus leistungsfähig.

Kapitel 11

Automaten

11.1 Endliche Automaten

11.1.1 Deterministische endliche Automaten

Definition 11.1 (Endlicher Automat (finite-state-automaton))

Ein endlicher Automat ist gegeben durch

- eine endliche Menge S von Zuständen
- eine Menge VT von terminalen Zeichen
- einer Übergangsfunktion $\delta : S \times VT \to S$

Außerdem ist ein Element $s_0 \in S$ als Anfangszustand ausgezeichnet und eine nichtleere Teilmenge $S_1 \subseteq S$ als Menge von Endzuständen festgelegt.

Jeder endliche Automat EA akzeptiert eine Menge von Wörtern $L(EA)$ auf die folgende Weise:

Ein gegebenes Wort $w = a_1 \dots a_k$ denken wir uns auf das Eingabeband des EA geschrieben. Im Anfangszustand s_0 liest der Automat den ersten Buchstaben a_1 und geht in den Zustand $\delta(s_0, a_1) = s_i$ über. In diesem Zustand wird nur der zweite Buchstabe von w eingelesen und der nächste Zustand $\delta(s_i, a_2) = s_j$ erreicht. Das Wort gehört zu $L(EA)$, wird also von EA akzeptiert, wenn nach Einlesen des letzten Buchstabens a_k von w ein Endzustand aus S_1 erreicht wird.

Formal wird $L(EA)$ definiert, indem man die Übergangsfunktion δ zu einer Funktion

$$\delta : S \times VT^* \to S$$

fortsetzt.

Definition 11.2 (Fortsetzung von δ)

Der Funktionswert $\delta(s, w)$ wird induktiv über die Länge von w wie folgt definiert:

$$\begin{array}{lcl} \delta(s, \varepsilon) & = & s \\ \delta(s, aw_1) & = & \delta(s', w_1) \text{ wobei } \delta(s, a) = s' \end{array}$$

Definition 11.3 (L(EA))

$$L(EA) = \{w \in VT^* \mid \delta(s_0, w) \in S_1\}$$

Varianten:

Gelegentlich wird in Beispielen die Übergangsfunktion nicht für alle Paare (s, a) angegeben. Wird während der Abarbeitung eines Wortes w eine Situation (s, a) erreicht, für die $\delta(s, a)$ nicht definiert ist, so gilt w als nicht akzeptiert. Ein endlicher Automat, so daß $\delta(s, a)$ für alle $s \in S$ und $a \in VT$ definiert ist, heißt ein **vollständiger endlicher Automat**.

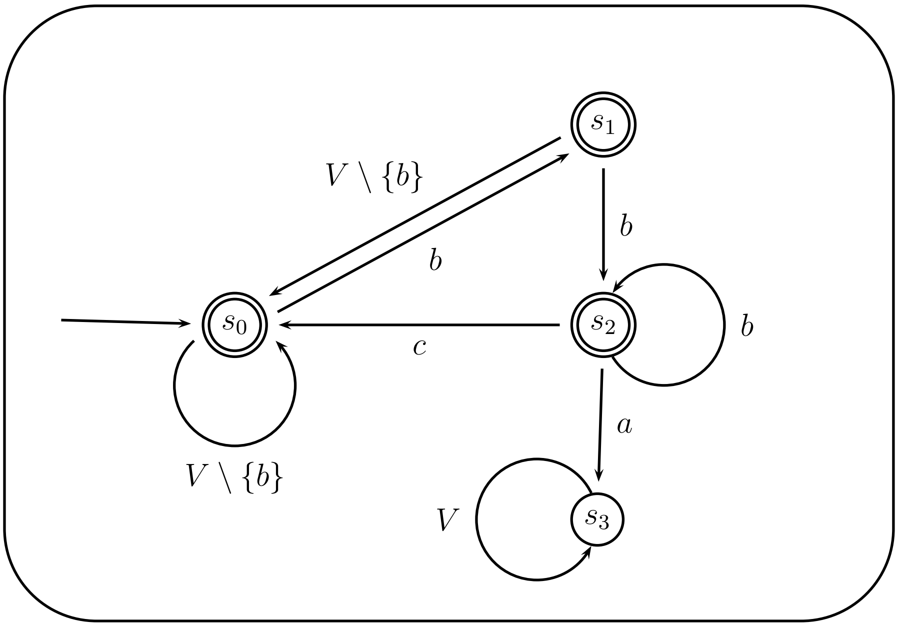


Abbildung 11.1: Der Automat N_{bba}

Beispiel 11.4
Der endliche Automat N_{bba} ist gegeben durch

S	$=$	$\{s_0, s_1, s_2, s_3\}$	alle Zustände
V	$=$	$\{a, b, c\}$	terminale Zeichen
S_1	$=$	$\{s_0, s_1, s_2\}$	Endzustände

Die Übergangsfunktion wird durch die Pfeile in Abbildung 11.1 in offensichtlicher Weise festgelegt.

$$L(N_{bba}) = \{w \in \{a, b\}^* \mid bba \text{ ist kein Teilwort von } w\}$$

Beispieldurchläufe: Das Wort *bbca* wird von N_{bba} akzeptiert, *cbbac* nicht.

11.1.2 Nichtdeterministische endliche Automaten

Definition 11.5 (Nichtdeterministische endliche Automaten)

Ein nichtdeterministischer endlicher Automat wird durch die folgenden Bestimmungsstücke gegeben:

- eine endliche Menge S von Zuständen,
- ein Alphabet V,
- ein Anfangszustand $s_0 \in S$,
- eine Übergangsfunktion $\delta : S \times V \to Pot(S)$,
- eine Menge $F \subseteq S$ von Finalzuständen,

Die Änderung gegenüber den deterministischen endlichen Automaten besteht also darin, daß die Übergangsfunktion δ als Werte Mengen von Zuständen annimmt:

Die Fortsetzung von δ zu einer Funktion

$$\delta : S \times V^* \to Pot(S)$$

wird jetzt wie folgt definiert:

$$\begin{array}{lcl} \delta(s, \varepsilon) & = & \{s\} \\ \delta(s, aw_1) & = & \{s' : \text{ es gibt } s_1 \in S \text{ mit } s_1 \in \delta(s, a) \\ & & \quad \text{und } s' \in \delta(s_1, w_1)\} \end{array}$$

Die von einem nichtdeterministischen endlichen Automaten NEA akzeptierte Sprache $L(NEA)$ wird jetzt durch

$$L(NEA) = \{w \in V^* : \delta(s_0, w) \cap F \neq \emptyset\}$$

definiert.

Varianten: In der angegebenen Definition kann ein nichtdeterministischer endlicher Automat im Zustand s, ohne einen Buchstaben zu lesen, nur nach s übergehen, $\delta(s, \varepsilon) = \{s\}$. Häufig begegnet man der Variante, die für $\delta(s, \epsilon)$ eine beliebige Teilmenge von S zuläßt. Übergänge von s zu $s_1 \in \delta(s, \varepsilon)$ nennt man dann spontane Übergänge.

Definition 11.6 (Endliche Automaten mit spontanen Übergängen)

Ein endlicher Automat mit spontanten Übergängen wird durch die folgenden Bestimmungsstücke gegeben:

- eine endliche Menge S von Zuständen,
- ein Alphabet V,
- ein Anfangszustand $s_0 \in S$,
- eine Menge $F \subseteq S$ von Finalzuständen,
- eine Übergangsfunktion $\delta : S \times (V \cup \{\varepsilon\}) \to Pot(S)$

Um die Fortsetzung der Transitionsfunktion δ für diese Variante endlicher Automaten zu beschreiben benötigt man die folgende Hilfsfunktion:

Definition 11.7
Sei $A = (S, V, s_0, \delta, F)$ ein endlicher Automat mit spontanen Übergängen, dann ist die Funktion

$$\varepsilon\text{-}cl : Pot(S) \to Pot(S)$$

für $I \subseteq S$ definiert als:

$\varepsilon\text{-}cl(I)$ ist die kleinste Teilmenge $J \subseteq S$ mit

1. $I \subseteq J$
2. für alle $s \in J$ gilt $\delta(s, \varepsilon) \subseteq J$.

Die Bezeichnung ε-*cl* soll an ε-*closure* erinnern.

Die Fortsetzung von

$$\delta : S \times (V \cup \{\varepsilon\}) \to Pot(S)$$

zu

$$\bar{\delta} : S \times V^* \to Pot(S)$$

kann jetzt definiert werden als:

$$\begin{array}{lcl} \bar{\delta}(s, \varepsilon) & = & \varepsilon\text{-}cl(\delta(s, \varepsilon)) \\ \bar{\delta}(s, w_1 a) & = & \varepsilon\text{-}cl(\{s' : \text{ es gibt } s_1 \in \bar{\delta}(s, w_1) \text{ so, daß } s' \in \delta(s_1, a)\}) \end{array}$$

Der folgende Satz ist von grundlegender Bedeutung für die Theorie endlicher Automaten und wird auch in allen einführenden Texten behandelt. Da wir später, auch auf Details des Satzes, zurückgreifen wollen, ist er auch hier noch einmal explizit aufgeführt.

Satz 11.8
Zu jedem nichtdeterministischen endlichen Automaten $\mathcal{A} = (S, V, s_0, \delta, F)$ gibt es einen deterministischen endlichen Automaten $\mathcal{B} = (Q, V, q_0, \Delta, G)$ mit

$$L(\mathcal{A}) = L(\mathcal{B})$$

Dabei kann $\mathcal{A}$ spontane Übergänge enthalten und muß auch nicht vollständig sein.

Beweis: Die neuen Zustände sind Teilmengen von alten Zuständen, genauer

$$\begin{array}{lcl} Q & = & \{X \subseteq S \mid \varepsilon\text{-}cl(X) = X\} \\ q_0 & = & \varepsilon\text{-}cl(\{s_0\}) \\ \Delta(X, a) & = & \varepsilon\text{-}cl(\bigcup_{s \in X} \delta(s, a)) \\ G & = & \{X \in Q \mid F \cap X \neq \emptyset\} \end{array}$$

Offensichtlich ist $\mathcal{B}$ deterministisch.
Man zeigt nun leicht, durch Induktion über die Länge des Wortes $w \in V^*$,

$$\bar{\delta}(s_0, w) = \Delta(q_0, w)$$

∎

Wir beweisen noch zwei Lemmata, die einerseits für sich genommen ein Licht auf das Konzept eines endlichen Automaten werfen, anderseits als Hilfsmittel in späteren Beweisen willkommen sind.

Lemma 11.9
Zu jedem nichtdeterministischen endlichen Automaten $\mathcal{A}$ gibt es einen nichtdeterministischen endlichen Automaten $\mathcal{B} = (Q_B, V, q_0^B, \delta_B, F_B)$ mit $L(\mathcal{A}) = L(\mathcal{B})$, so daß für alle $q \in Q_B$ gilt $q_0^B \notin \delta_B(q, x)$ für alle $x \in V$.
Mit anderen Worten, der Anfangszustand wird nur am Anfang betreten und ist dann nie wieder erreichbar. Das beinhaltet, daß es keinen Übergang von q_0^B zu sich selbst gibt.

Beweis: Wir beginnen mit $\mathcal{A} = (Q_A, V, q_0^A, \delta_A, F_A)$. Es sei q_0^B ein neuer Zustand und $Q_B = Q_A \cup \{q_0^B\}$. Unverändert bleibt $F_B = F_A$, falls $s_0^A \notin F_A$ und $F_B = F_A \cup \{q_0^B\}$ sonst. Die Übergangsfunktion bleibt zu nächst unverändert für $q \in Q_A$, $x \in V$, d.h. $\delta_B(q,x) = \delta_A(q,x)$ Neu hinzukommt $\delta_B(q_0^B, x) = \delta_A(q_0^A, x)$ Die Bedingung $q_0^B \notin \delta_B(q,x)$ ist für alle $q \in Q_B$ offensichtlich erfüllt. Außerdem gilt für jedes $w \in V^*$ die Gleichheit $\delta_B(q_0^B, w) = \delta_A(q_0^A, w)$. Besteht w nur aus einen Buchstaben, so ist das gerade die Definition von $\delta_B(q_0^B, x)$. Da aber im weiteren Verlauf der Abarbeitung des Wortes w der Anfangszustand q_0^B nie wieder betreten wird, kann keine Unterschied zwischen $\delta_B(q_0^B, w)$ und $\delta_A(q_0^A, w)$ auftreten. Das sichert, daß $w \in L(\mathcal{B})$ genau dann gilt, wenn $w \in L(\mathcal{A})$ gilt.

∎

Die Finalzustände eines endlichen Automaten spielen genau genommen eine zweifache Rolle. Zum einen zeigt ihr Erreichen an, daß ein Wort akzeptiert wird, zum anderen sind sie auch Zwischenzustände, nach denen die Aktion des Automaten weitergeht. Das folgende Lemma zeigt, daß man sich ohne Verlust an Ausdrucksstärke auf Automaten beschränken könnte, in denen Endzustände ihren Namen auch verdienen.

Lemma 11.10
Zu jedem nichtdeterministischen endlichen Automaten $\mathcal{A}$ mit $\varepsilon \notin L(\mathcal{A})$ gibt es einen nichtdeterministischen endlichen Automaten $\mathcal{B} = (Q_B, V, q_0^B, \delta_B, F_B)$ mit $L(\mathcal{A}) = L(\mathcal{B})$, so daß für alle $q \in F_B$ und alle $x \in V$ gilt $\delta(q,x) = \emptyset$.

Beweis: Wir beginnen mit $\mathcal{A} = (Q_A, V, q_0^A, \delta_A, F_A)$. Für jeden Finalzustand $q \in F_A$ wählen wir ein Symbol q^e, sodaß alle neuen q^e untereinander verschieden sind und auch nicht in Q_A vorkommen. Wir setzen $Q_B = Q_A \cup \{q^e \mid q \in F_A\}$. Endzustände in $\mathcal{B}$ sind nur die neuen Kopien der alten Endzustände, d.h. $F_B = \{q^e \mid q \in F_A\}$. Der Anfangszustand bleibt unverändert, $s_0^B = s_0^A$, und die Übergangsfunktion wird für $x \in V$ festgesetzt zu

$$\begin{array}{rcll} \delta_B(s,x) & = & \delta_A(s,x) \cup \{q^e \mid q \in \delta_A(s,x)\} & \text{falls } s \notin F_B \\ \delta_B(s,x) & = & \emptyset & \text{falls } s \in F_B \end{array}$$

Durch Induktion über die Länge des Wortes w zeigt man, daß für jedes $w \in V^*$ die Gleichung $\delta_B(s_0^B, w) = \delta_A(s_0^A, w) \cup \{q^e \mid q \in \delta_A(s_0^A, w)\}$ gilt. Da $F_B \cap \delta_B(s_0^B, w) = \emptyset$ genau dann gilt, wenn $F_A \cap \delta_A(s_0^A, w) = \emptyset$ gilt, erhalten wir, wie gewünscht, $L(\mathcal{A}) = L(\mathcal{B})$.

Hier folgt der induktive Beweis von

$$\delta_B(s_0^B, w) = \delta_A(s_0^A, w) \cup \{q^e \mid q \in \delta_A(s_0^A, w)\}$$

Für den Anfangsfall $w = \varepsilon$ ergibt sich nach Definition von δ

$$\delta_B(s_0^B, \varepsilon) = \{s_0^B\} = \{s_0^A\} = \delta_A(s_0^A, \varepsilon) \cup \{q_0^{A,e} \mid q_0^A \in F_A\}$$

Die Gleichung würde stimmen wenn der Anfangszustand von $\mathcal{A}$ keine Endzustand wäre. Das ist aber gerade durch die Voraussetzung $\varepsilon \notin L(\mathcal{A})$ garantiert.

Im Induktionsschritt nehmen wir die Gültigkeit der zu beweisenden Gleichung für das Wort w an und wollen sie für das Wort wa zeigen.

$$\begin{array}{rcl} s \in \delta_B(s_0^B, wa) & \Leftrightarrow & \exists s'(s' \in \delta_B(s_0^B, w) \wedge s \in \delta_B(s', a)) \\ & \Leftrightarrow & \exists s'(s' \in \delta_A(s_0^A, w) \cup \{q^e \mid q \in \delta_A(s_0^A, w)\} \\ & & \wedge s \in \delta_B(s', a)) \\ & \Leftrightarrow & \exists s'(s' \in \delta_A(s_0^A, w) \wedge s \in \delta_B(s', a)) \end{array}$$

Die erste Äquivalenz folgt aus der Definition von δ_B, die zweite nutzt die Induktionsvoraussetzung für $\delta_B(s_0^B, w)$ und die dritte Zeile rührt von der Beobachtung, daß $s' \in \{q^e \mid q \in \delta_A(s_0^A, w)\}$ nicht auftreten kann, da anderenfalls $\delta_B(s', a) = \emptyset$ wäre. Setzen wir die Definition von δ_B ein, erhalten wir:

$$s \in \delta_B(s_0^B, wa) \quad \Leftrightarrow \quad \exists s'(s' \in \delta_A(s_0^A, w) \wedge s \in \delta_A(s', a) \cup \{q^e \mid q \in \delta_A(s', a)\}$$

Woraus jetzt leicht folgt

$$s \in \delta_B(s_0^B, wa) \quad \Leftrightarrow \quad s \in \delta_A(s_0^A, wa) \cup \{q^e \mid q \in \delta_A(s_0^A, wa)\}$$

Wäre $\varepsilon \in L(\mathcal{A})$, so wäre s_0^A ein Endzustand in $\mathcal{A}$ gewesen. Nach der obigen Konstruktion ist s_0^A zwar auch noch der Anfangszustand von $\mathcal{B}$, aber kein Finalzustand mehr. Die in der Fomulierung des Lemmas gemachte Einschränkung an $\mathcal{A}$ ist also notwendig.

■

Definition 11.11 (Produktautomat)
Seien $\mathcal{A} = (S_A, V, s_0^A, \delta_A, F_A)$ und $\mathcal{B} = (S_B, V, s_0^B, \delta_B, F_B)$ zwei endliche Automaten über dem gemeinsamen Alphabet V, dann ist der Produktautomat $\mathcal{C} = (S_C, V, s_0^C, \delta_C, F_C) = \mathcal{A} \times \mathcal{B}$ gegeben durch:

$$\begin{array}{lcl} S_C & = & \{(s_1, s_2) \mid s_1 \in S_A, s_2 \in S_B\} \\ s_0^C & = & (s_0^A, s_0^B) \\ \delta_C((s_1, s_2), a) & = & \{(t_1, t_2) \mid t_1 \in \delta_A(s_1, a), t_2 \in \delta_B(s_2, a)\} \\ F_C & = & \{(s_1, s_2) \mid s_1 \in F_A, s_2 \in F_B\} \end{array}$$

Lemma 11.12
$L(\mathcal{A} \times \mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{B})$

Beweis Ist in jedem Einführungstext in die Theorie endlicher Automaten enthalten.

■

11.1.3 Reguläre Ausdrücke

Neben den Operationen, die für beliebige Mengen erklärt werden können, wie Vereinigung ($\cup$), Durchschnitt ($\cap$), relative Komplementbildung ($\setminus$) betrachtet wir hier Operationen die nur auf Mengen von Wörtern Sinn machen:

Definition 11.13 (Operationen mit Wortmengen)

Seien $L, L_1, L_2 \subseteq V^*$.

1. $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$,
2. $L^* = \{w_1 \dots w_n \mid n \geq 0, w_i \in L\}$

Manche Autoren verwenden für die Konkatenation von Wörtern w_1, w_2 und Sprachen L_1, L_2 die Notation $w_1 \cdot w_2$ bzw. $L_1 \cdot L_2$

Definition 11.14 (Reguläre Ausdrücke)

Die Menge Reg_V der regulären Ausdrücke über einem Alphabet V wird induktiv definiert durch:

1. $\emptyset \in Reg_V$,
2. $\varepsilon \in Reg_V$,
3. für jedes $a \in V$ ist $a \in Reg_V$,
4. für $t \in Reg_V$ gilt auch $(t)^* \in Reg_V$,
5. für $t_1, t_2 \in Reg_V$ gilt auch $(t_1 t_2) \in Reg_V$ und $(t_1 + t_2) \in Reg_V$.

Definition 11.15 (Semantik regulärer Ausdrücke)

Durch die folgende Vorschrift wird jedem regulären Ausdruck t über V eine Menge $S(t)$ von Wörtern in V^* zugeordnet.

1. $S(\emptyset) = \emptyset$,
2. $S(\varepsilon) = \{\varepsilon\}$,
3. $S(a) = \{a\}$,
4. $S((t)^*) = S(t)^*$,

5. $S((t_1 t_2)) = S(t_1)S(t_2)$ und $S((t_1 + t_2)) = S(t_1) \cup S(t_2)$.

Wir benutzen im folgenden stillschweigend die Assoziativität der Konkatenation und von $+$ um in regulären Ausdrücken Klammern einzusparen, also $(a + b + c)$ anstelle von $((a + b) + c)$.

Satz 11.16

1. Für jeden regulären Ausdruck t gibt es einen endlichen Automaten A mit $S(t) = L(A)$.

2. Zu jedem endlichen Automaten A gibt es einen regulären Ausdruck t mit $S(t) = L(A)$.

Beweis:
Für 1. siehe z.B. [Kfo88], Proposition 2, Seite 34 oder [HU79], Theorem 2.3, Seite 30.
Für 2. siehe z.B.[Kfo88], Proposition 12, Seite 38 oder [HU79], Theorem 2.4, Seite 33.

∎

Reguläre Ausdrücke sind in vielen Bereichen der Informatik ein alltägliches Hilfsmittel geworden. Als ein Beispiel dafür wollen wir etwas ausführlicher auf die Benutzung regulärer Ausdrücke im GNU Emacs Editor eingehen, siehe z.B. [Sta88] oder das GNU emacs manual `http://www.gnu.org/software/emacs/manual/emacs.html`. Während einer Editiersitzung kann man das Kommando `C-M-s` eingeben, worauf in der Kommandozeile `Regexp I-search:` erscheint. Hinter dem Doppelpunkt kann man jetzt einen regulären Ausdruck t in der emacs-Syntax (Abschnitt 13.5 in [Sta88]) eingeben. Der Kursor springt dann unmittelbar auf die erste, von der aktuellen Kursorposition aus gesehen, Zeichenkette im Textpuffer die in $S(t)$ liegt. Das Kommando `C-s` liefert das nächste Vorkommen einer Zeichenkette in $S(t)$. Abgesehen von der Nützlichkeit dieser Kommandos ist es auch instruktiv zu sehen, wie das theoretische Konzept regulärer Ausdrücke in praktisches umgesetzt wird.
Die erste Frage die zu beantworten ist lautet: Was ist das Alphabet V_{emacs}? Antwort V_{emacs} besteht aus den 256 Zeichen des erweiterten ASCII Zeichensatzes. Dazu gehören auch Zeichen, die man normalerweise nicht auf dem Bildschirm zu sehen bekommt, z.B. das ASCII Zeichen mit dem Oktalcode 012. Dieses Zeichen heißt LF , für line feed. Man sieht es nicht selbst auf dem Bildschirm, sondern nur seine Auswirkung, daß eben an dieser Stelle eine neue Zeile beginnt. Aber LF kann genau so in einem regulären Ausdruck benutzt werden, wie jedes andere Zeichen aus V_{emacs} auch. An dieser Stelle sollte man wissen, wie nicht-graphische Zeichen von emacs dargestellt

werden und wie man sie eingeben kann ohne ein eventuell damit verbundenes Editorkommando auszulösen. Das kann man in [Sta88] in den Abschnitten 2 und 4.1 nachlesen. Ein interessanteres Problem tritt auf, wenn wir nach Zeichenketten suchen wollen, die mit einem x beginnen auf welches eine gerade Anzahl von y's folgt. Wir würden gern den regulären Ausdruck $x(yy)^*$ eingeben. Die Klammern sind notwendig um den gewünschten regulären Ausdruck von dem ungewünschten $(xyy)^*$ oder $xy(y)^*$ zu unterscheiden. Andererseits sind die beiden Klammern (und) Zeichen in V_{emacs}. Gibt man nach dem Prompt `Regexp I-search:` $x(yy)^*$ ein so findet man alle Ausdrücke der Form $x(yy)\ldots)$ mit einer beliebigen Anzahl schliessender Klammern. Dieses Problem wird gelöst, wie an vielen anderen Stellen in der Informatik auch, indem man V_{emacs} unterteilt in Sonderzeichen und normale Zeichen. An dieser Stelle geht die Einfachheit des theoretischen Konzepts verloren unter vielfachen Ausnahme- und Sonderregeln, Kompatibilitätsrücksichten und historischen Gewohnheiten. In der emacs Syntax regulärer Ausdrücke sind zunächst einmal die folgenden neun Zeichen Sonderzeichen:

`$,^,.,*,+,?,[,],\`

Wir betrachten zunächst \. Es hat zwei Funktionen. Erstens dient es dazu, die Sonderzeichen zu normalen Zeichen zu machen, d.h. für den reguläre Ausdruck `\$` gilt $S($`\$`$) = \{$`$`$\}$ und konsequenterweise auch $S(\backslash\backslash) = \{\backslash\}$. Zweitens macht \ aus einigen normalen Zeichen Sonderzeichen. Hier ist eine Liste einiger auf diese Weise gebildeter Sonderzeichen:

`\|`, `\(`, `\)`, `'`, `` ` ``, `\b`, `\B`, `\<`, `\>`, `\w`, `\W`,`\scode`,`\Scode`
und `\z` für jede Ziffer `z`.

Wir wollen nur die wichtigsten dieser Sonderzeichen erklären. Eine vollständige Beschreibung findet man in [Sta88] in Abschnitt 13.5. Für reguläre emacs Ausdrücke t_1, t_2 gilt $S(t_1$`\|`$t_2) = S(t_1) \cup S(t_2)$, d.h. `\|` in regulären emacs Ausdrücken entspricht dem $+$ in den regulären Ausdrücken aus Definition 11.14. Die Sonderzeichen `\(` und `\)` dienen der Klammerung. Der oben erwähnte reguläre Ausdruck $x(yy)^*$ wird also in emacs Syntax geschrieben als x`\(`yy`\)*`. Hiermit ist auch gleich verraten, daß der Stern * in den regulären emacs Ausdrücken dieselbe Bedeutung hat wie in unseren regulären Ausdrücken. Da auch die Konkatenation von regulären emacs Ausdrücken mit der üblichen Semantik möglich ist, entspricht jedem regulären Ausdruck nach Definition 11.14 ein regulärer emacs Ausdruck mit derselben Bedeutung. Wir richten im folgenden unser Augenmerk auf die Frage, ob reguläre emacs Ausdrücke vielleicht ausdrucksmächtiger sind als unsere regulären Ausdrücke.

Das Sonderzeichen `.` besitzt als seine Interpretation $S(\texttt{.})$ die Menge aller Zeichen mit der einzigen Ausnahme des Zeichens LF. Da kann man in der Syntax aus Definition 11.14 auch hinschreiben als eine Summe mit 255 Summanden. Die Sonderzeichen `+` und `?` sind wie `*` postfix Operatoren, wobei `+` die übliche Bedeutung, $t\texttt{+} \equiv t(t)^*$, hat und $t\texttt{?} \equiv (\varepsilon + t)$. Die Sonderzeichen `^` und `$` suchen nach dem leeren Wort am Anfang bzw. am Ende einer Zeile. Genauer gilt für einen regulären emacs Ausdruck t $S(\texttt{\^{}}t) = \{w \in S(t) \mid w \text{ kommt am Anfang einer Zeile vor}\}$ und $S(t\texttt{\$}) = \{w \in S(t) \mid w \text{ kommt am Ende einer Zeile vor}\}$.
Die Sonderzeichen `\[` und `\]` erlauben eine bequeme Beschreibung endlicher Mengen von Zeichen. Es gilt $S(\texttt{[}a_1 \ldots a_k\texttt{]}) = \{a_1, \ldots, a_k\}$, jedenfalls dann, wenn die a_i keine Sonderzeichen enthalten. Allerdings gelten innerhalb der `[...]` Paare ganz andere Sonderzeichenregeln als bisher. Hier sind nämlich

`]`, `^`, `-`

die einzigen Sonderzeichen. Warum `]` ein Sonderzeichen ist, ist klar. Das Minuszeichen `-` wird benutzt um einen Bereich aufeinanderfolgender Zeichen zu beschreiben, z.B. $S(\texttt{[}a - z\texttt{]})$ = Menger aller Kleinbuchstaben. Schließlich ist $S(\texttt{[\^{}}a_1 \ldots a_k\texttt{]}) = \{z \in V_{emacs} \mid z \neq a_1 \text{ und } z \neq \ldots \text{ und } z \neq a_k\}$. Was macht man, wenn man eines der drei Sonderzeichen in einem `[...]` Paar verwenden möchte? Anstelle von `-` schreibt man `---`, `]` ist kein Sonderzeichen, wenn es unmittelbar nach `[` steht und `^` ist nur dann ein Sonderzeichen, wenn es unmittelbar nach `[` steht. Es gibt noch einige kleinere Subtilitäten, aber auf die brauchen wir hier nicht einzugehen.
Mancher Leser mag sich gefragt haben, was in dem oben erwähnten Prompt `Regexp I-search:` wohl das `I` bedeutet? Es steht für *inkrementell*. Die Suche nach der ersten Zeichenkette, die zu dem regulären Ausdruck gehört, erfolgt inkrementell. Tippt man als regulären Ausdruck a ein, so springt der Kursor zum ersten im Puffer vorkommenden a. Tippt man jetzt noch ein b dazu, so springt der Kursor zum ersten Vorkommen der Zeichenkette ab. Wie verhält sich die Suche bei Eingabe von $a\texttt{\textbackslash|}b$? Da waren sich die Implementierer nicht immer einig. In meiner emacs Version beginnt die Suche nach Eingabe von $a\texttt{\textbackslash|}b$ nochmal von vorn, so daß auch die Vorkommen von b gefunden werden, die vor dem ersten a im Text stehen.

11.1.4 Übungsaufgaben

Übungsaufgabe 11.1.1
Zeigen Sie, daß es zu jedem endlichen Automaten EA einen vollständigen endlichen Automaten EAc gibt mit $L(EA) = L(EAc)$. Kann EAc immer

gebildet werden, ohne die Menge der Zustände von EA zu vergrößern?

Übungsaufgabe 11.1.2
Zeigen Sie, daß es zu jedem nichtdeterministischen endlichen Automaten NEA mit spontanen Übergängen einen nichtdeterministischen endlichen Automaten NEA_0 ohne spontane Übergänge gibt mit $L(NEA) = L(NEA_0)$. ([Hopcroft, Ullman 79], p. 24.)

11.2 Omega Automaten

Manche in der Praxis vorkommenden Automaten besitzen keine terminalen Zustände, man möchte im Gegenteil sogar, daß sie nicht terminieren. Ein Beispiel dafür sind Sender- und Empfängerautomaten für Übertragungsprotokolle. Das führt auf der theoretischen Seite zur Einführung unendlich langer Wörter.

Definition 11.17 (Unendliche Wörter)

Sei V ein (weiterhin endliches) Alphabet.
Mit V^ω bezeichnen wir die Menge der unendlich langen Wörter mit Buchstaben aus V.
Für $n \in N$ bezeichnet $w(n)$ den n-ten Buchstaben in w und $w \downarrow (n)$ das endliche Anfangstück $w(0) \dots w(n)$ von w.
Wir nennen ein Wort $w \in V^\omega$ manchmal auch ein ω-Wort über V.

Wer Schwierigkeiten hat, sich eine unendlich lange Zeichenkette vorzustellen, der interpretiere ein Wort $w \in V^\omega$ als eine Funktion $w : I\!N \to V$, von den natürlichen Zahlen in das Alphabet. Die Präzisierung beantwortet auch die Frage des mit der Theorie der unendlichen Mengen vertrauten Lesers, welchen Grad der Unendlichkeit wir meinen: unendliche Wörter sind Folgen vom Ordnungstyp ω.
Man beachte außerdem, daß das leere Wort ε nicht in V^ω vorkommt.

Definition 11.18 (Operationen)

1. Ist $K \subseteq V^*$ eine Menge (endlicher) Wörter, so bezeichnet K^ω die Menge der unendlichen Wörter der Form

$$w_1 \dots w_i \dots \text{ mit } w_i \in K \text{ für alle } i$$

2. Die Verkettung zweier unendlich langer Wörter macht keinen Sinn, es ist jedoch möglich einem unendlich langen Wort ein endliches vorzuschalten. Ist $K \subseteq V^*$ und $J \subseteq V^\omega$, dann setzen wir:

$$KJ = \{w_1 w_2 \mid w_1 \in K, w_2 \in J\}$$

3. Ist $K \subseteq V^*$ eine Menge endlicher Wörter, dann ist

$$\vec{K} = \{w \in V^\omega \mid w \downarrow (n) \in K \text{ für unendlich viele } n\}$$

Manche Autoren benutzen $lim(K)$ anstelle von $\vec{K}$.

11.2.1 Büchi Automaten

Definition 11.19 (Büchi Automaten)

Sei $\mathcal{A} = (S, V, s_0, \delta, F)$ ein nicht deterministischer endlicher Automat. Für ein ω-Wort $w \in V^\omega$ nennen wir eine Folge $s_1, \ldots, s_n, \ldots$ eine *Berechnungsfolge* (Englisch *run*) für w, wenn für alle $0 \leq n$ gilt

$$s_{n+1} \in \delta(s_n, w(n))$$

Eine Berechnungsfolge $s_1, \ldots, s_n, \ldots$ heißt eine *akzeptierte Berechnungsfolge* wenn in ihr unendlich viele Finalzustände vorkommen.
Die von $\mathcal{A}$ akzeptierte ω-Sprache wird definiert durch

$$L^\omega(\mathcal{A}) = \{w \in V^\omega \mid \text{ es gibt eine akzeptierte Berechnungsfolge für } w\}$$

Wir nennen eine Menge L von ω-Wörtern ω**-regulär**, wenn es einen endlichen Automaten $\mathcal{A}$ gibt mit $L^\omega(\mathcal{A}) = L$.

Gelegentlich braucht man auch den Begriff einer Berechnungsfolge unabhängig davon, welches Wort damit assoziiert ist.
Eine Berechungsfolge für den Automaten $\mathcal{A} = (S, V, s_0, \delta, F)$ ist eine Folge $s_1, \ldots, s_n, \ldots$ von Zuständen, so daß für alle $0 \leq n$ ein $a \in V$ existiert mit $s_{n+1} \in \delta(s_n, a)$. Eine Berechnungsfolge heißt akzeptierend, wenn unendliche viele der s_i Endzustände sind.

Man beachte, daß eine Berechnungsfolge mit s_1 beginnt. Also ist s_1 der Zustand der nach Einlesen des ersten Buchstabens erreicht wird. Das ist eine technisches Detail, das später einige Vereinfachungen mit sich bringt.

Ein Büchi Automat $\mathcal{A}$ unterscheidet sich also in nichts von einem endlichen Automaten, nur die Definition, wann $\mathcal{A}$ ein ω-Wort akzeptiert ist neu.

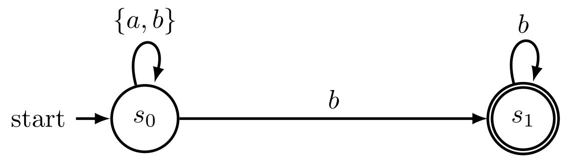


Abbildung 11.2: Der Büchi-Automat $\mathcal{N}_{afin}$

Satz 11.20 (Entscheidbarkeit)
Die Frage, ob für einen Büchi-Automaten $\mathcal{B}$ die Menge der akzeptierten Wörter nicht leer ist, d.h. $L^\omega(\mathcal{B}) \neq \emptyset$, ist entscheidbar.

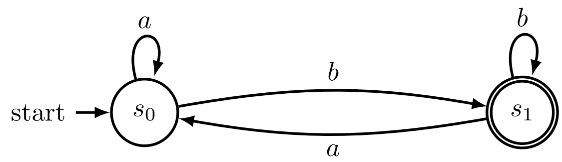


Abbildung 11.3: Büchi Automat für $a^*b(b+a^+b)^\omega$

Beweis: Um $L^\omega(\mathcal{B}) \neq \emptyset$ zu zeigen muß man nur einen erreichbaren Endzustand $q_f \in F$ finden, der auf einer Schleife liegt, d.h. q_f ist erreichbar von q_f ausgehend. Diese Frage kann sogar in linearer Zeit beantwortet werden, z.B. durch den Tarjan-Paige Algorithmus [Tar72].

∎

Das folgende Lemma fasst zusammen, was über den Zusammenhang zwischen $L(\mathcal{A})$ und $L^\omega(\mathcal{A})$ gesagt werden kann

Lemma 11.21
Sei $\mathcal{A} = (S, V, s_0, \delta, F)$ ein endlicher Automat und $K = L(\mathcal{A})$. Dann gilt

1. $L^\omega(\mathcal{A}) \subseteq \vec{K}$
2. Falls $\mathcal{A}$ deterministisch ist gilt sogar $L^\omega(\mathcal{A}) = \vec{K}$

Beweis:

zu 1: Nach Definition für $w \in L^\omega(\mathcal{A})$ gibt es eine Berechnungsfolge $\rho(w)$, so daß die Menge $F_w = \{n \in I\!N^+ \mid \rho(w)(n) \in F\}$ unendlich ist. Für alle $n \in F_w$ gilt $\rho(w)(n) \in F$ und daher $w \downarrow (n) \in K$. Also $w \in \vec{K}$.

zu 2: Sei jetzt umgekehrt $w \in \vec{K}$. Dann ist die Menge $R_w = \{n \in I\!N^+ \mid w \downarrow (n) \in K\}$ unendlich. Für jedes $n \in R_w$ gibt es also eine Berechungsfolge s_n von $\mathcal{A}$ für $w \downarrow (n)$. Da $\mathcal{A}$ deterministisch ist gibt es für jedes n nur eine Möglichkeit für s_n. Insbesondere ist s_n jeweils ein Anfangsstück von s_{n+1}. Somit erhalten wir im Limes eine unendliche Berechnungsfolge s für w, die unendlich oft einen Endzustand durchläuft.

∎

Für nicht-deterministische Automaten gilt Teil 2 von Lemma 11.21 nicht notwendigerweise. Wir betrachten dazu den Automaten N_{afin} in Abbildung 11.2. Es gilt $L^\omega(N_{afin}) = \{w \in \{a,b\}^\omega \mid$ in w kommt a nur endlich oft vor$\}$ und $L(N_{afin}) = \{w \in \{a,b\}^* \mid w$ endet auf b$\}$. Man sieht leicht, daß $L^\omega(N_{afin}) \neq Lim(L(N_{afin}))$.

Korollar 11.22
Eine Sprache $L \subseteq V^\omega$ wird von einem deterministischen Büchi Automaten akzeptiert, gdw es eine reguläre Sprache $K \subseteq V^*$ gibt mit $L = \vec{K}$.

Beweis:
Wird L von einem deterministischen Büchi Automaten akzeptiert, so kann L nach Lemma 11.21 auch in der angegebenen Weise dargestellt werden. Nehmen wir jetzt umgekehrt an, das $L = \vec{K}$ für eine reguläre Sprache $K \subseteq V^*$ gilt. Dann gibt es einen deterministischen endlichen Automaten $\mathcal{A}$ mit $K = L(\mathcal{A})$. Man überzeugt sich leicht, daß in diesem Fall $\vec{K} = L^\omega(\mathcal{A})$ gilt.

∎

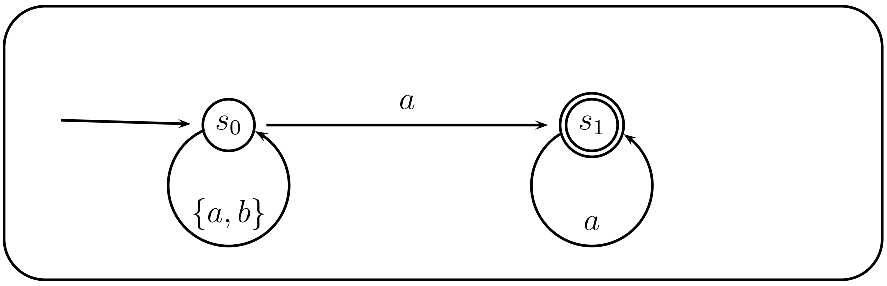


Abbildung 11.4: Der Büchi-Automat $\mathcal{N}_{bfin}$

Korollar 11.23
Es gibt Sprachen $L \subseteq V^\omega$, die von einem nicht deterministischen Büchi Automaten akzeptiert werden, aber von keinem deterministischen.

Beweis:
Sei $V = \{a, b\}$ und L die von dem Automaten $\mathcal{N}_{afin}$ in Abbildung 11.2 akzeptierte Sprache:

$$L = \{w \in V^\omega \mid w(n) = a \text{ nur für endlich viele } n\}$$

Wir zeigen, daß L nicht in der Form $L = \vec{K}$ für eine reguläre Sprache $K \subseteq V^*$ darstellbar ist. Wir nehmen an, das wäre doch der Fall und versuchen einen Widerspruch herzuleiten.
Es gibt ein $k_1 > 0$, so daß $b^{k_1} \in K$, da $b^\omega \in L$. Als nächstes muß es dann auch ein $k_2 > 0$ geben, so daß $b^{k_1}ab^{k_2} \in K$ liegt, weil $b^{k_1}ab^\omega \in L$ gilt. In dieser Weise fortfahrend gibt es $k_i > 0$, so daß $b^{k_1}ab^{k_2}a \ldots ab^{k_i} \in K$ gilt für alle i. Nach Annahme über die Darstellbarkeit von L folgt daraus auch $b^{k_1}ab^{k_2}a \ldots ab^{k_i}a \ldots \in L$ im Widerspruch zur Definition von L.

∎

11.2.2 Abschlusseigenschaften

Satz 11.24
Sind L_1, L_2 ω-reguläre Sprachen und ist K eine reguläre Sprache, dann ist auch

1. $L_1 \cup L_2$ ω-regulär,
2. K^ω ω-regulär, falls $\varepsilon \notin K$,
3. KL_1 ω-regulär,
4. $V^\omega \setminus L_1$ ω-regulär,
5. $L_1 \cap L_2$ ω-regulär.

Beweis:
zu 1. Seien zwei ω-reguläre Sprachen $L_1, L_2 \subseteq V^\omega$ gegeben und $\mathcal{A}_i = (Q_i, V, s_0^i, \delta_i, F_i)$ Büchi-Automaten mit $L_i = L_i^\omega(\mathcal{A}_i)$. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß $Q_1 \cap Q_2 = \emptyset$
Wir konstruieren einen Büchi-Automaten $\mathcal{A} = (Q, V, s_0, \delta, F)$ wie folgt, wobei s_0 ein neuer Zustand ist, der weder in Q_1 noch in Q_2 vorkommt.

$$
\begin{array}{lcll}
Q & = & Q_1 \cup Q_2 \cup \{s_0\} & \\
\delta(q,a) & = & \delta_i(q,a) & \text{falls } q \in Q_i \\
\delta(s_0,a) & = & \delta_1(s_0^1,a) \cup \delta_2(s_0^2,a) & \\
F & = & F_1 \cup F_2 &
\end{array}
$$

Man zeigt leicht, daß $L^\omega(\mathcal{A}) = L_1 \cup L_2$.
zu 2. Sei $\mathcal{A} = (Q_A, V, s_0^A, \delta_A, F_A)$ ein nichtdeterministischer endlicher Automat mit $L(\mathcal{A}) = K$. Wegen Lemma 11.9 können wir $s_0^A \notin \delta_A(q,x)$ für alle $q \in Q_A$ und $x \in V$ annehmen.

Wir definieren einen neuen Automaten $\mathcal{B} = (Q_B, V, s_0^B, \delta_B, F_B)$ durch:

$$
\begin{array}{lcll}
Q_B & = & Q_A & \\
s_0^B & = & s_0^A & \\
\delta_B(q,a) & = & \delta_A(q,a) \cup \{s_0^A\} & \text{falls } F_A \cap \delta_A(q,a) \neq \emptyset \\
\delta_B(q,a) & = & \delta_A(q,a) & \text{sonst} \\
F_B & = & \{s_0^B\} &
\end{array}
$$

Wir müssen uns überzeugen, daß $L^\omega(\mathcal{B}) = K^\omega$ gilt.

Betrachten wir dazu zunächst ein $w \in L^\omega(\mathcal{B})$. Sei $q_1 \dots q_i \dots$ eine akzeptierende Berechungsfolge für w. Nach Definition eines Büchi-Automaten muß der einzige Endzustand von $\mathcal{B}$ unendlich oft vorkommen. Seien $n_1, \dots n_k \dots$ die Positionen in aufsteigender Ordnung, sodaß $q_{n_k} = s_0^B$. Offensichtlich ist $n_1 = 0$. Mit w_k bezeichnen wir das Teilwort $w(n_k), w(n_k+1), \dots w(n_{k+1}-1)$ von w. Nach Einlesen eines Teilwortes w_k ist der Automat $\mathcal{B}$ im Zustand s_0^B. Das kann wegen der Definition von δ_B und der Voraussetzung an $\mathcal{A}$ nur der Fall sein wenn $F_A \cap \delta_A(s_0^A, w_k) \neq \emptyset$ gilt. Das heißt für alle k gilt $w_k \in L(\mathcal{A})$. Daraus folgt $w \in K^\omega$.

Sei jetzt umgekehrt w ein Wort aus K^ω. Also $w = w_1 \dots w_i \dots$ mit $w_i \in K$. Es gibt somit für jedes i akzeptierende Berechungsfolgen $q_{n_i}, q_{n_i+1}, \dots q_{n_{i+1}}$ für w_i in $L(\mathcal{A})$. Nach Definition einer endlichen akzeptierenden Berechungsfolge muß $q_{n_{i+1}} \in F_A$ gelten. Nach Definition von $\mathcal{B}$ ist auch $q_{n_i}, q_{n_i+1}, \dots q_{n_{i+1}-1}, s_0^B$ eine Berechnungsfolge für w_i in $\mathcal{B}$. Setzen wir diese Berechnungfolgen hintereinander erhalten wir eine Berechungsfolge für w, in der der Finalzustand q_0^B an jeder Position n_i, und damit unendlich oft, auftritt. Also $w \in L^\omega(\mathcal{B})$.

■

zu 3. Übungsaufgabe 11.2.1
zu 4. zurückgestellt auf später, Satz 11.30
zu 5. folgt unmittelbar aus 1. und 4.

■

Satz 11.25
Eine Teilmenge $L \subseteq V^\omega$ ist eine ω-reguläre Menge von ω-Wörtern, genau dann, wenn L eine endliche Vereinigung von Mengen der Form

$$JK^\omega$$

für reguläre Mengen $J, K \subseteq V^*$ ist, wobei $\varepsilon \notin K$.

Beweis:

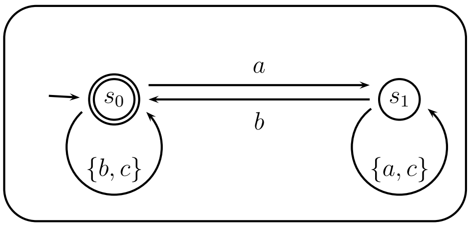
Sei $\mathcal{A} = (Q, V, s_0, \delta, F)$ ein Büchi-Automat mit $L^\omega(\mathcal{A}) = L$. Für $p, q \in Q$ sei

$$L_{p,q} = \{w \in V^* \mid q \in \delta(p, w)\}$$

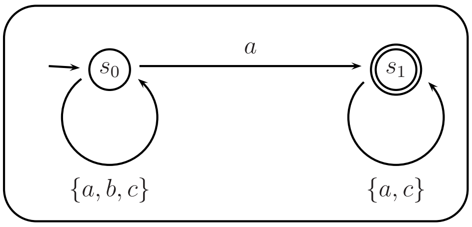
Offensichtlich ist jedes $L_{p,q} \subseteq V^*$ eine reguläre Menge. Man überzeugt sich leicht, daß

$$L = \bigcup_{p \in F} L_{s_0,p} L_{p,p}^\omega$$

gilt. Die umgekehrter Richtung der Behauptung folgt direkt aus Satz 11.24.



(a) $N_{baftera}$



(b) $co - N_{baftera}$

Abbildung 11.5: Beispiel zur Komplementbildung

■

In Abbildung 11.5 ist ein Paar komplementärer Büchi-Automaten zu sehen. Das Beispiel stammt aus [Tho90]. Es gilt $L^\omega(N_{baftera}) = \{w \in \{a, b, c\}^\omega \mid$ nach jedem a kommt irgendwann ein $b\}$. $L^\omega(co{-}N_{baftera})$ ist dazu offensichtlich die Komplementärmenge, d.h. $\{w \in \{a, b, c\}^\omega \mid$ es gibt ein a nach dem kein b mehr vorkommt$\}$. Daß es zu jedem Büchi-Automaten einen Komplementärautomaten gibt, d.h. die Abgeschlossenheit ω-regulärer Sprachen unter Komplementbildung in V^ω, muß noch bewiesen werden. Der Beweis bedarf einiger Vorbereitung. Eine zentrale Rolle spielt dabei die folgende Äquivalenzrelation.

Definition 11.26
Sei $\mathcal{A} = (Q, V, s_0, \delta, F)$ ein Büchi-Automat. Dann definieren wir für $p, q \in Q$ und $u, w \in V^*$

1. $L_{p,q} = \{w \in V^* \mid q \in \delta(p, w)\}$
   (wie im Beweis von Satz 11.25)

2. $L^F_{p,q} = \{w = a_0 \dots a_k \in V^* \mid$ es gibt eine Folge von Zuständen $q_0, \dots q_k$ mit
$$\begin{array}{lcll} q_0 & = & p & \\ q_k & = & q & \\ q_{i+1} & \in & \delta(q_i, a_i) & \text{für alle } 0 \le i < k \\ q_i & \in & F & \text{für ein } 0 \le i \le k \ \} \end{array}$$

3. $u \equiv_A w$ gdw
   für alle $p, q \in Q$ gilt
$$\begin{array}{lcl} u \in L_{p,q} & \Leftrightarrow & w \in L_{p,q} \\ u \in L^F_{p,q} & \Leftrightarrow & w \in L^F_{p,q} \end{array}$$

Offensichtlich gilt stets $L^F_{p,q} \subseteq L_{p,q}$ und falls $p \in F$ oder $q \in F$ auch $L^F_{p,q} = L_{p,q}$. Ebenso leicht sieht man, daß $\equiv_A$ eine Äquivalenzrelation ist.

Lemma 11.27
Die Äquivalenzrelation $\equiv_A$ hat folgende Eigenschaften:

1. für alle $p, q \in Q$ sind $L^F_{p,q}$ und $L_{p,q}$ reguläre Mengen,

2. für $u \in V^\omega$ gilt für die Äquivalenzklasse M_w

$$M_w = \bigcap_{(p,q)\in P} L_{p,q} \cap \bigcap_{(p,q)\notin P} \sim L_{p,q} \cap \bigcap_{(p,q)\in R} L^F_{p,q} \cap \bigcap_{(p,q)\notin R} \sim L^F_{p,q}$$

   wobei $P = \{(p,q) \in Q^2 \mid w \in L_{p,q}\}$ und $R = \{(p,q) \in Q^2 \mid w \in L^F_{p,q}\}$

3. $\equiv_A$ besitzt endlich viele Äquivalenzklassen,

4. jede Äquivalenzklasse von $\equiv_A$ ist eine reguläre Menge,

5. sind U, V Äquivalenzklassen von $\equiv_A$, dann folgt aus $UV^\omega \cap L^\omega(\mathcal{A}) \neq \emptyset$ schon $UV^\omega \subseteq L^\omega(\mathcal{A})$

Beweise:
zu 1 Für $L_{p,q}$ wurde das schon im Beweis von Satz 11.25 benutzt. Wegen $L^F_{p,q} = \bigcup_{f\in F} L_{p,f}L_{f,q}$ folgt auch die Regularität von $L^F_{p,q}$.
zu 2 folgt direkt aus der Definition von $\equiv_A$.
zu 3 unmittelbare Konsequenz aus 2.
zu 4 folgt aus 1, 2 und den Abschlußeigenschaften regulärer Mengen.
zu 5 Sei $w \in UV^\omega \cap L^\omega(\mathcal{A})$ gegeben und u ein weiteres Wort aus UV^ω. Es ist $u \in L^\omega(\mathcal{A})$ zu zeigen.
Wegen $w \in UV^\omega$ können wir w wie folgt zerlegen:

$$w = w_1 w_2 \ldots w_n \ldots$$

wobei $w_1 \in U$ und $w_i \in V$ für alle $i > 1$. Wegen $w \in L^\omega(\mathcal{A})$ gibt es eine Berechnungsfolge σ für w in $\mathcal{A}$ in der unendlich viele Zustände aus F auftreten. Wir interessieren uns für bestimmt Stützpunkte in der Folge σ; mit q_n bezeichen wir den Zustand in σ, der nach Einlesen des Teilwortes w_{n-1} und vor Beginn des Einlesens von w_n eingenommen wird. Insbesondere sei $q_0 = s_0$ der Anfangszustand. Sei schließlich $u \in UV^\omega$, genauer

$$u = u_1 u_2 \ldots u_n \ldots$$

mit $u_1 \in U$ und $u_i \in V$ für alle $i > 1$. Da U und V nach Voraussetzung $\equiv_A$ Äquivalenzklassen sind gilt für alle i

$$w_i \equiv_A u_i$$

Wir konstruieren eine Berechnungsfolge ρ für u wie folgt. Dabei wird ρ die Zustände q_n in derselben Reihenfolge durchlaufen, wie σ, aber welche Zustände auf dem Weg von q_n nach q_{n+1} auftreten ist von vornherein noch nicht festgelegt, ja es ist zunächst noch nicht einmal klar ob ein solcher Weg gefunden werden kann. Nach Definition der q_i gilt auf jeden Fall $w_n \in L_{q_{n-1},q_n}$. Wegen $w_n \equiv_A u_n$ gilt dann auch $u_n \in L_{q_{n-1},q_n}$. Man kann also den Weg von q_{n-1} nach q_n in ρ interpolieren. Kommt in σ ein Finalzustand zwischen q_{n-1} und q_n vor, so gilt $w_n \in L^F_{q_{n-1},q_n}$ und wegen $w_n \equiv_A u_n$ auch $u_n \in L^F_{q_{n-1},q_n}$. Das Teilstück von ρ zwischen q_{n-1} und q_n kann demnach auch so gewählt werden, daß darin ein Finalzustand vorkommt. Das zeigt, daß ρ eine akzeptierende Berechnungsfolge für u ist und damit $u \in L^\omega(\mathcal{A})$.

∎

Satz 11.28 (Satz von Ramsey)

Sei $P_1, \ldots, P_k$ eine endliche Partition aller zweielementigen Teilmenge der Menge $I\!N$ der natürlicher Zahlen. Dann gibt es eine unendliche Teilmenge $T \subseteq I\!N$ und ein i mit $1 \leq i \leq k$, so daß alle zweielementigen Teilmengen von Elementen aus T in P_i liegen.
In Formeln:
aus $P_1 \uplus \ldots \uplus P_k = I\!N^{[2]}$ folgt die Existenz einer unendlichen Menge $T \subseteq I\!N$, so daß $T^{[2]} \subseteq P_i$ für ein i.

Beweis: Siehe etwa [Mon76] p.448 oder [Fel78] p. 228 oder die Orginalarbeit [Ram29]. Wichtig bei der Ausgangssituation des Ramseyschen Satzes ist, daß für zwei natürliche Zahlen $n, m \in I\!N$ nicht $(n,m) \in P_i$ und $(m,n) \in P_j$ für $i \neq j$ auftreten kann. In der vorliegenden Formulierung wurde das dadurch ausgeschlossen, daß anstelle von geordneten Paaren ungeordnete Zweiermengen betrachtet wurden. Häufig wird das Problem auch dadurch gelöst, daß nur Paare (n,m) mit $n \leq m$ betrachtet werden.

Lemma 11.29
Zu jedem ω-Wort $w \in V^\omega$ gibt es eine $\equiv_A$-Äquivalenzklassen U und V mit $w \in UV^\omega$.

Beweis:
Seien $U_1, \ldots, U_n$ alle Äquivalenzklassen von $\equiv_A$ und bezeichne für $i < j$ $w_{i,j}$ das endliche Teilwort $w(i) \ldots w(j-1)$ von w. Dann wird durch

$$P_r = \{\{i,j\} \in I\!N^{[2]} \mid [w_{i,j}]_{\equiv_A} = U_r\}$$

eine Partition von $I\!N^{[2]}$ definiert. Nach dem Satz von Ramsey gibt es eine unendliche Teilmenge $T \subseteq I\!N$ und ein i mit $T^{[2]} \subseteq P_i$. Sei i_0 das kleinste

Element in T und $U = [w \downarrow (i_0)]_{\equiv_A}$, dann gilt offensichtlich

$$w \in UU_i^\omega$$

▪

Satz 11.30
Ist $L \subseteq V^\omega$ eine ω-reguläre Menge, dann auch $V^\omega \setminus L$

Beweis
Sei $\mathcal{A}$ ein Büchi Automat mit $L^\omega(\mathcal{A}) = L$.

$$V^\omega \setminus L = \bigcup_{(U,V) \in \mathcal{R}} UV^\omega$$

dabei gilt $(U, V) \in \mathcal{R}$ genau dann, wenn $UV^\omega \cap L = \emptyset$.

`argumentiere detailierter! Notationskonflikt für` V

▪

11.2.3 Varianten von Büchi Automaten

Wir beginnen mit der einfachsten Modifikation von Definition 11.19 für die wir nicht einmal einen neuen Namen erfinden wollen. Die modifizierten Automaten sind von der Form $\mathcal{C} = (S, V, S_0, \delta, F)$ mit einer Menge S_0 von Anfangszuständen anstelle eines einzigen Anfangszustands s_0.

Lemma 11.31
Zu jedem Büchi Automaten $\mathcal{C} = (S, V, S_0, \delta, F)$ mit einer Menge von Anfangszuständen gibt es einen Büchi Automaten $\mathcal{A}$ mit einem einzigen Anfangszustand und

$$L^\omega(\mathcal{C}) = L^\omega(\mathcal{A})$$

Beweis: Sei $S_0 = \{s_1, \ldots, s_k\}$. Wir setzen $\mathcal{C}_i = (S, V, s_i, \delta, F)$. Offensichtlich gilt $L^\omega(\mathcal{C}) = \bigcup_{i=1}^k L^\omega(\mathcal{C}_i)$. Die Existenz von $\mathcal{A}$ folgt jetzt aus Abgeschlossenheit ω-regulärer Mengen unter Vereinigung.

▪

Definition 11.32 (Erweiterte Büchi Automaten)
Ein erweiterter Büchi Automat

$$\mathcal{A} = (S, V, s_0, \delta, F_1, \ldots, F_n)$$

unterscheidet sich von einem (normalen) Büchi Automaten nur dadurch, daß er statt einer einzigen Menge F von Finalzuständen endliche viele solcher Mengen $F_1, \ldots, F_n$ enthält.

Ein Wort w wird akzeptiert, wenn es eine Berechungsfolge s für w gibt, die für jedes j, $1 \leq j \leq n$ unendlich viele Zustände aus F_j enthält. Die von $\mathcal{A}$ akzeptierte ω-Sprache kann dann definiert werden als:

$$\begin{array}{rcl} L^\omega(\mathcal{A}) & = & \{w \in V^\omega \mid \text{ es gibt eine Berechnungsfolge } s \text{ für } w, \\ & & \text{so daß für jedes } j, 1 \leq j \leq n, \\ & & \text{die Menge } \{i \mid s_i \in F_j\} \text{ unendlich ist.}\} \end{array}$$

Das Konzept eines erweiterten Büchi Automaten macht in vielen Fällen das Leben (für den theoretischen Informatiker) etwas einfacher, an Ausdrucksstärke wird gegenüber den einfachen Büchi Automaten nichts gewonnen:

Lemma 11.33
Zu jedem erweiterten Büchi Automaten $\mathcal{A}_e$ gibt es einen einfachen Büchi Automaten $\mathcal{A}$ mit

$$L^\omega(\mathcal{A}_e) = L^\omega(\mathcal{A})$$

Beweis: Dieselbe Konstruktion, die in dem Beweis von Lemma 12.2 benutzt werden wird, kann auch benutzt werde um zu einem erweiterten Automaten $\mathcal{A}_e = (S_e, V, s_0^e, \delta_e, F_1, F_2)$ einen äquivalenten einfachen Automaten $\mathcal{A}$ $= (S, V, s_0, \delta, F)$ zu konstruieren. Diese Konstruktion kann man dann n-mal wiederholen um auch den allgemeinen Fall abzudecken.

■

Definition 11.34 (Müller Automaten)
Ein Müller Automat $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$ ist ein endlicher Automat $\mathcal{M} = (S, V, s_0, \delta)$ ohne Angabe von Endzuständen, aber stattdessen mit einer Menge $\mathcal{F}$ nicht leerer Endzustandsmengen, d.h. für alle $F \in \mathcal{F}$ gilt $F \subseteq S$ und $F \neq \emptyset$.
Ist $\sigma = s_1, \ldots, s_n, \ldots$ eine Zustandsfolge, so bezeichnet $In(\sigma)$ die Menge der Zustände, die unendlich oft in σ vorkommen, also

$$In(\sigma) = \{s \in S \mid \text{es gibt unendlich viele } n \text{ mit } s_n = s\}$$

Die von einem Müller Automat $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$ akzeptierte ω-Sprache wird definiert durch:

$$L^\omega(\mathcal{M}) = \{w \in V^\omega \mid In(\sigma) \in \mathcal{F} \text{ für eine Berechnungsfolge } \sigma \text{ für } w\}$$

Lemma 11.35
Die Klasse der von nichtdeterministischen Büchi Automaten akzeptierten ω-Sprachen stimmt überein mit der von nichtdeterministischen Müller Automaten akzeptierten ω-Sprachen.

Beweis: Für die einfache Richtung, siehe Übungsaufgabe 11.2.9.

Für die umgekehrte Richtung gehen wir von einem Müller Automaten $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$ aus und suchen einen Büchi Automaten $\mathcal{A}$ mit $L^\omega(\mathcal{M}) = L^\omega(\mathcal{A})$. Wir werden den Automaten $\mathcal{A}$ nicht im einzelnen konstruieren, sondern seine Existenz aus den bisher bewiesenen Sätzen herleiten. Als erstes betrachten wir den Spezialfall, daß $\mathcal{F} = \{F_0\}$ ist mit $F_0 = \{a_1, \ldots, a_n\}$ für Zustände a_1 bis a_n. Seien $b_1, \ldots, b_m$ die anderen Zustände von $\mathcal{M}$. Nach Lemma 11.33 gibt es einen Büchi Automaten $\mathcal{A}_0$ mit

$$\begin{array}{rcl} L^\omega(\mathcal{A}_0) & = & \{w \in V^\omega \mid \text{ es gibt eine Berechnungsfolge } \sigma \text{ in } \mathcal{A}_0 \text{ für } w \\ & & \quad \text{mit } F_0 \subseteq In(\sigma)\} \end{array}$$

Außerdem können wir Büchi Automaten $\mathcal{A}_j$ konstruieren mit $L^\omega(\mathcal{A}_j) = \{w \in V^\omega \mid b_j$ kommt in w nur endlich oft vor$\}$. Für den *Durchschnittsautomaten* (nach Satz 11.24 (5)) $\mathcal{A}$ von $\mathcal{A}_0$,$\mathcal{A}_1$,...,$\mathcal{A}_m$ gilt dann $L^\omega(\mathcal{A}) = L^\omega(\mathcal{M})$.
Ist im allgemeinen Fall $\mathcal{F} = F_0, \ldots, F_k$. Dann liefert das bisherige Verfahren für $1 \leq i \leq k$ Büchi Automaten $\mathcal{B}_i$ mit $L^\omega(\mathcal{B}_j) = L^\omega((S, V, s_0, \delta, \{F_i\}))$. Für den *Vereinigungsautomaten* (nach Satz 11.24 (1)) $\mathcal{B}$ von $\mathcal{B}_1$,...,$\mathcal{B}_k$ gilt dann $L^\omega(\mathcal{B}) = L^\omega(\mathcal{M})$.

`Beweis noch mal kontrollieren.`

■

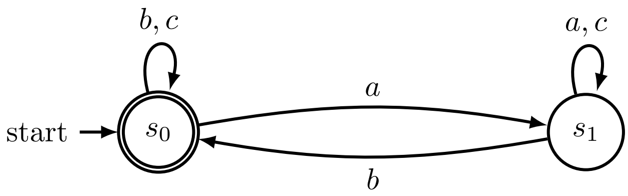
11.2.4 Übungsaufgaben

Übungsaufgabe 11.2.1

Ist L eine ω-reguläre Sprache und K eine reguläre Sprache, dann ist auch KL eine ω-reguläre Sprache.

Übungsaufgabe 11.2.2

Bestimmen Sie die von dem folgenden Büchi Automaten akzeptiert ω-Sprache in dem Alphabet $V = \{a, b, c\}$.



Übungsaufgabe 11.2.3

Der in Abbildung 11.2 gezeigte endliche Automat $\mathcal{A}$ ist nichtdeterministisch.

1. Berechnen Sie den äquivalenten deterministischen endlichen Automaten $\mathcal{A}_1$.

2. Bestimmen Sie $L^\omega(\mathcal{A}_1)$.

Übungsaufgabe 11.2.4

Sei $\mathcal{A}$ eine endlicher Automat, so daß für jeden Endzustand q_f und jeden Buchstaben x des Vokabulars $\delta_A(q_f, x) = \emptyset$ gilt. Nach Lemma 11.10 stellt diese Annahme keine Einschränkung der Allgemeinheit dar. Wir benutzen K als Abkürzung für $L(\mathcal{A})$.

Der Automat $\mathcal{B}$ sei wie folgt definiert:

$$
\begin{array}{lcll}
Q_B & = & Q_A & \\
s_0^B & = & s_0^A & \\
\delta_B(q, x) & = & \delta_A(q, x) & \text{falls } q \notin F_A, x \in V \\
\delta_B(q, x) & = & \delta_A(q, x) \cup \delta_A(s_0^A, x) & \text{falls } q \in F_A, x \in V \\
F_B & = & F_A &
\end{array}
$$

Zeigen Sie, $L^\omega(\mathcal{B}) = K^\omega$.

Übungsaufgabe 11.2.5

Zeigen Sie, daß in der vorangegangenen Übungsaufgabe, Nummer 11.2.4, die Voraussetzung an den Automaten $\mathcal{A}$ unverzichtbar sind.

Übungsaufgabe 11.2.6

Finden Sie endliche Automaten $\mathcal{A}_1$, $\mathcal{A}_1$, so daß für die akzeptierten Sprachen $K_i = L(\mathcal{A}_i)$ gilt

$$K_1 \cap K_2 = \emptyset \quad \text{aber} \quad \vec{K}_1 \cap \vec{K}_2 \neq \emptyset$$

Übungsaufgabe 11.2.7

Finden Sie einen deterministischen Büchi Automaten $\mathcal{A}$, so daß das Komplement der von $\mathcal{A}$ akzeptierten Wörter nicht von einem deterministischen Büchi Automaten akzeptiert wird.

Übungsaufgabe 11.2.8

In der Definition 11.19 eines Büchi Automaten wurde nicht geklärt, ob spontane Übergänge zugelassen sind oder nicht. Zeigen Sie: zu jedem endlichen Automaten $\mathcal{A}$ gibt es einen Automaten $\mathcal{B}$ ohne spontane Transitionen mit

$$L^\omega(\mathcal{A}) = L^\omega(\mathcal{B})$$

Übungsaufgabe 11.2.9

Finden Sie zu einem gegebenen (nicht deterministischen) Büchi-Automaten $\mathcal{A} = (S_A, V, s_0^A, \delta_A, F)$ einen Müller-Automaten $\mathcal{M} = (S_M, V, s_0^M, \delta_M, \mathcal{F})$ mit

$$L^\omega(\mathcal{A}) = L^\omega(\mathcal{M})$$

Übungsaufgabe 11.2.10

Zeigen Sie, daß es zu jedem Müller Automaten $\mathcal{M}$ über dem Alphabet V einen komplementären Müller Automaten $\mathcal{M}_c$ gibt, d.h.

$$L^\omega(\mathcal{M}_c) = V^\omega \setminus L^\omega(\mathcal{M})$$

Zusammen mit Lemma 11.35 erhalten wir damit einen alternativen Beweis zu Satz 11.30 für die Abgeschlossenheit ω-regulärer Mengen unter Komplementbildung .

Übungsaufgabe 11.2.11

Sei $\mathcal{A} = (S, V, s_0, \delta, F)$ ein endlicher Automat. Für ein Wort $w \in V^\omega$ definieren wir induktiv Mengen $Q_n(w)$ von Zuständen:

$$\begin{array}{lcl} Q_0(w) & = & \{s_0\} \\ Q_{n+1}(w) & = & \bigcup_{s \in Q_n(w)} \delta(s, w(n)) \end{array}$$

Schließlich sei

$$L^{alt}(A) = \{w \in V^\omega \mid \text{ für unendlich viele } n \text{ gilt } Q_n(w) \cap F \neq \emptyset\}$$

Frage: Gilt

$$L^{alt}(A) = L^\omega(A)$$

für alle A.

Übungsaufgabe 11.2.12

Das Alphabet V enthalte mindestens die beiden Buchstaben a und b. Die Menge L_{a3b} bestehe genau aus den Wörtern $w \in V^\omega$, so daß für jedes n mit $w(n) = a$ gilt $w(n+1) = b$ oder $w(n+2) = b$ oder $w(n+3) = b$.
Finden Sie einen Büchi Automaten $\mathcal{B}_{a3b}$ mit

$$L^\omega(\mathcal{B}_{a3b}) = L_{a3b}.$$

Übungsaufgabe 11.2.13

Das Alphabet V enthalte mindestens die beiden Buchstaben a und b. Finden Sie einen Büchi Automaten $\mathcal{B}_{a\infty->b\infty}$, so daß $w \in L^\omega(\mathcal{B}_{a\infty->b\infty})$ genau dann gilt, wenn w die Bedingung erfüllt:
Wenn a in w unendlich oft vorkommt, dann kommt auch b in w unendlich oft vor.
Diese Eigenschaft kann als Grundmuster für Fairnessbedingungen angesehen werden, wenn durch den Buchstaben a symbolisert werden soll, daß eine Aktion ausführbar ist und b symbolisiert, daß diese Aktion tatsächlich ausgeführt wird.

Übungsaufgabe 11.2.14

Wir stellen uns die symbolische Modellierung eines Systems vor, in dem eine Aktion A ausführbar sein kann oder nicht. Die Ausführbarkeit von A soll durch die Boolesche Variable a beschrieben werden. Weiterhin kann in dem betrachteten System die Aktion A tatsächlich ausgeführt werden oder nicht. Die Ausführung von A soll durch die Boolesche Variable b beschrieben werden. Es ist nicht ausgeschlossen, daß a und b gleichzeitig wahr sind. Zur Beschreibung dieses Systems in unserem bisherigen formalen Rahmen betrachten wir das Alphabet V mit den Buchstaben $w = \{\}$, $x = \{a\}$, $y = \{b\}$, $z = \{a, b\}$. Eine Ausführungsfolge des Systems wollen wir *schwach fair* (engl. weakly fair) nennen, wenn b unendlich of wahr wird, falls von einer Stelle an a immer wahr ist. Entsprechend nennen wir ein Wort $w \in V^\omega$ *schwach fair* wenn aus der Existenz einer Zahl n mit $w(m) \in \{x, z\}$ für alle $m \geq n$ folgt, daß es unendliche viele $k \geq n$ gibt mit $w(k) \in \{y, z\}$.
Finden Sie einen Büchi Automaten $\mathcal{B}_{wfair}$, so daß $L^\omega(\mathcal{B}_{wfair})$ genau die Menge der schwach fairen Wörter in V^ω ist.

Übungsaufgabe 11.2.15

Zu jedem Automatenalphabet V assoziieren wir die Signatur Σ_V, welche

für jeden Buchstaben $a \in V$ ein ein-stelliges Prädikatszeichen P_a enthält. Zusätzlich enthält Σ_V das zwei-stellige Relationszeichen $<$ und das ein-stellige Funktionszeichen $s()$.
Es gibt einen natürlichen Zusammenhang zwischen Wörtern $w \in V^\omega$ und der Struktur $\mathcal{M}_w$, die wie folgt definiert ist:

$$\begin{array}{lcl} \text{Universum} M_w & = & \mathbb{N} \text{ Menge der natürlichen Zahlen} \\ < & = & \text{übliche Ordnung auf } \mathbb{N} \\ s^{\mathcal{M}_w}(n) & = & n+1 \\ P_a^{\mathcal{M}_w}(n) \text{ ist wahr} & gdw & w(n) = a \end{array}$$

Beachten Sie, daß die ersten drei Teile der Definition von $\mathcal{M}_w$ nicht von w abhängen.
Wir fixieren ein Alphabet V, das mindestens a, b, c enthält. Finden Sie eine Formel F der Logik zweiter Stufe, so daß für alle $w \in V^\omega$ gilt $\mathcal{M}_w \models F$ genau dann, wenn $w \in \{ab, bbc\}^\omega$.

Übungsaufgabe 11.2.16

Mit der Notation aus Aufgabe 11.2.15:

1. Finden Sie eine Formel der Prädikatenlogik erster Stufe F_1, so daß für alle $w \in V^\omega$ gilt: $\mathcal{M}_w \models F_1$ gdw $w \in L_{a3b}$, wobei L_{a3b} in Aufgabe 11.2.12 definiert wurde.

2. Finden Sie eine Formel der Prädikatenlogik erster Stufe F_2, so daß für alle $w \in V^\omega$ gilt: $\mathcal{M}_w \models F_2$ gdw w liegt in der in in Aufgabe 11.2.13 definierten Sprache.

3. Dasselbe für die in Aufgabe 11.2.15 definierte Sprache. Die gesuchte Formel soll F_3 heißen.

Übungsaufgabe 11.2.17

Es ist naheliegend die in Definition 11.14 eingeführten regulären Ausdrücke zu ω-regulären Ausdrücken zu erweitern.

Definition 11.36 (ω-reguläre Ausdrücke)

Die Menge Reg_V der regulären Ausdrücke über einem Alphabet V wird induktiv definiert durch:

1. $\emptyset \in Reg_V$,

2. $\varepsilon \in Reg_V$,

3. für jedes $a \in V$ ist $a \in Reg_V$,

4. für $t \in Reg_V$ gilt auch $t^* \in Reg_V$ und $t^\omega \in Reg_V$,

5. für $t_1, t_2 \in Reg_V$ gilt auch $(t_1 t_2) \in Reg_V$ und $(t_1 + t_2) \in Reg_V$.

Definition 11.37 (Semantik ω-regulärer Ausdrücke)

Durch die folgende Vorschrift wird jedem ω-regulären Ausdruck t über V eine Menge $S(t)$ von Wörtern in $V^* \cup V^\omega$ zugeordnet.

1. $S(\emptyset) = \emptyset$,

2. $S(\varepsilon) = \{\varepsilon\}$,

3. $S(a) = \{a\}$,

4. $S(t^*) \;=\; \{w_1 \dots w_{n-1} w_n \mid \; w_i \in S(t) \cap V^* \text{ für } 1 \leq i < n, \;, \; w_n \in V^* \cup V^\omega, 0 \leq n \in \mathbb{N}\}$

5. $S(t^\omega) = \{w_1 \dots w_i \dots \mid S(t) \cap (V^* \setminus \{\varepsilon\}) \text{ für alle } 1 \leq i, i \in \mathbb{N}\}$,

6. $S((t_1 t_2)) = S(t_1) S(t_2)$ und $S((t_1 + t_2)) = S(t_1) \cup S(t_2)$.

Zeigen Sie, daß zu jedem Büchi Automaten $\mathcal{B}$ ein ω-regulärer Ausdruck $t_{\mathcal{B}}$ existiert mit

$$L^\omega(\mathcal{B}) = S(t_{\mathcal{B}})$$

Übungsaufgabe 11.2.18

Es ist naheliegend zu fragen, ob in der vorangegangen Übungsaufgabe 11.2.17 auch die umgekehrte Aussage gilt, d.h. ob es zu jedem ω-regulären Ausdruck t einen Automaten $\mathcal{B}$ gibt, der genau die Wörter in $S(t)$ akzeptiert. Das erste Problem besteht darin, daß $S(t)$ sowohl endliche als auch unendliche Wörter enthält, wir aber noch kein Automatenmodell kennengelernt haben, das endliche und unendliche Wörter gleichzeitig akzeptiert. Das könnte man wohl ändern, aber es bliebe noch ein steiniger Weg zu einer endgültigen Beantwortung der gestellten Frage. Wir schlagen hier einen anderen Lösungsweg vor. Wir definieren zwei Transformationen, fin und inf, auf der Menge der ω-regulären Ausdrücke durch wechselseitige Induktion:

$$
\begin{array}{lcllcl}
fin(\emptyset) & = & \emptyset & inf(\emptyset) & = & \emptyset \\
fin(\varepsilon) & = & \varepsilon & inf(\varepsilon) & = & \emptyset \\
fin(a) & = & a & inf(a) & = & \emptyset \\
fin(t^*) & = & fin(t)^* & inf(t^*) & = & fin(t)^* inf(t) \\
fin(t^\omega) & = & \emptyset & inf(t^\omega) & = & fin(t)^\omega \\
fin(t_1 t_2) & = & fin(t_1) fin(t_2) & inf(t_1 t_2) & = & fin(t_1) inf(t_2) \\
fin(t_1 + t_2) & = & fin(t_1) + fin(t_2) & inf(t_1 + t_2) & = & inf(t_1) + inf(t_2)
\end{array}
$$

Zeigen Sie

1. Für jeden ω-regulären Ausdruck t gilt

$$S(fin(t)) = S(t) \cap V^* \text{ und } S(inf(t)) = S(t) \cap V^\omega$$

2. Für jeden ω-regulären Ausdruck t gilt

$$S(t) = S(fin(t)) \cup S(inf(t))$$

3. Für jeden ω-regulären Ausdruck t gibt es einen endlichen Automaten $\mathcal{B}$, so daß $L(\mathcal{B}) = S(fin(t))$.

4. Für jeden ω-regulären Ausdruck t gibt es einen Büchi Automaten $\mathcal{B}$, so daß $L^\omega(\mathcal{B}) = S(inf(t))$.

Kapitel 12

Modellprüfung

12.1 Einführung

Um einen Einstieg in die Probleme und grundlegenden Konzepte temporaler Logiken zu finden, betrachten wir in diesem Abschnitt in aller Ausführlichkeit ein Beispiel aus dem Gebiet der Verifikation von Kommunikationsprotokollen. Bei der Ausarbeitung dieses Materials habe ich viel dem Buch [Hol04, Chapter 14] profitiert. Holzmann, und damit auch die folgende Darstellung, orientiert sich an dem ITU Standard SS7 (Signaling System 7) für öffentliche Telefonnetze (public-switched telephone network). Fairerweise muß man jedoch eingestehen, daß hier nur ein minimaler Teil von SS7 überhaupt zur Sprache kommt.

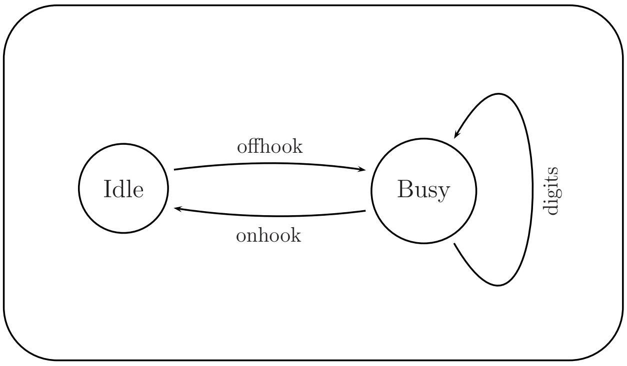


Abbildung 12.1: Einfaches Modell eines Telefonteilnehmers

In Abbildung 12.1 ist ein einfaches Modell eines Telefonteilnehmers (engl. subscriber) gezeigt. Die einzigen *Aktionen* in dem Modell sind *Hörer abnehmen* (offhook), *Hörer auflegen* (onhook) und *Wählen* (digits). Es wird nicht unterschieden ob der Teilnehmer selbst anruft oder angerufen wird. Wir beschränken uns in dem ganzen Szenarium auf den Teilnehmer, der das Gespräch initiiert. Nach dem der Benutzer den Hörer abgehoben und gewählt hat wird auch nicht weiter unterschieden, was passiert. Kommt eine Verbindung zustande? Ist die gerufene Nummer besetzt? Nimmt der Teilnehmer am anderen Ende der Leitung ab? Das alles ist in dem einen Zustand *Busy* zusammengefasst.

Ein zweiter Bestandteil des Kommunikationsmodels ist die lokale Vermittlungstelle. Ein Automat dafür ist in Abbildung 12.2 zu sehen. In dem Abschnitt 11.1 über Automaten hatten wir ein abstraktes Alphabet V betrachtet, das zur Markierung von Kanten diente. Der Kantenmarkierungen in Abbildung 12.2 sind dagegen um Einiges konkreter. Ausdrücke der Form $tpc?xxx$ besagen, daß die Vermittlungsanlage die Nachricht xxx über den

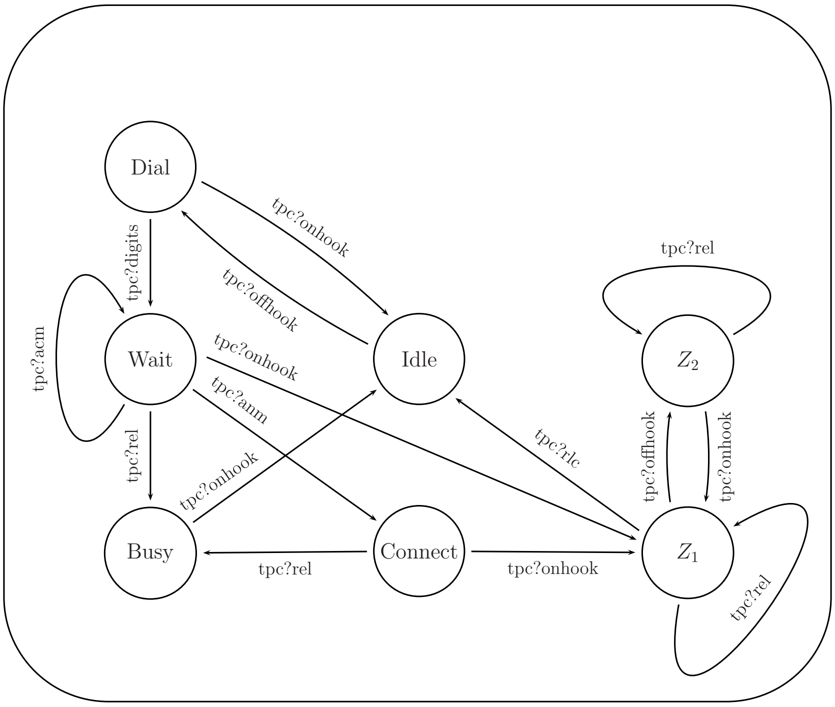


Abbildung 12.2: Einfacher Automat einer Telephonvermittlung

Kanal *tpc* empfangen hat. Folgendes Vokabular für Nachrichten steht zur Verfügung:

Nachricht	Bedeutung
offhook	Hörer wurde abgehoben
onhook	Hörer wurde aufgelegt
digits	Nummer wurde gewählt
iam	initial address message
acm	address complete message
rel	Einleitung des Verbindungsabbaus (release)
anm	End-zu-Endverbindung hergestellt
rlc	Quittierung des Verbindungsabbaus

Woher die empfangenen Nachrichten kommen kann man aus dem Automaten nicht erkennen. Das weiss die Anlage in Wirklichkeit auch nur, weil zu den Nachrichten noch Datenfelder gehören, in denen entsprechende Information geliefert wird. In unserem Szenarium können wir aber erklären woher die Nachrichten kommen. Die ersten drei `offhook`, `onhook` und `digits` kommem vom Teilnehmer. Wenn er den Hörer abhebt, geht bei der Vermittlungsanlage die Nachricht `tpc?offhook` ein. In dem Kommunikationsprozess sind neben den bisher geanntent Akteuren, dem Teilnehmer und der lokalen Vermittlungsstelle, weiterhin beteiligt, ein Zielteilnehmer, eine Zielvermittlungsstelle und auch Zwischenknoten des Verbindungsweges. Die Nachrichten `acm`, `anm`, `rel` und `rlc` werden von der Zielvermittlungsstelle an die lokale Vermittlung geschickt. Die *initial address message* wird, in unserem Szenarium. von der lokalen Vermittlung an die Zielvermittlungsstelle geschickt. In Abbildung 12.2 sind nur die Bedingungen für die Ausführung eines Zustandsübergangs eingetragen (engl.*guards*), die Aktionen, die zusätzlich zur Zustandänderung passieren sind der Übersichtlichkeit halber weggelassen worden.

Zwei typische Abläufe des Vermittlungsautomaten ist in Abbildung 12.3 zu sehen. 1.Beispiel: Zustand ist *idle*, Teilnehmer hebt Hörer ab, Zustand ist *dial*, Teilnehmer wählt, Zustand ist *wait*, der gewünschte Teilnehmer wird erreicht und nimmt seinerseits den Hörer ab, Zustand ist *connect*, der Zielteilnehmer legt den Hörer auf, Zustand ist *busy* , Teilnehmer legt den Hörer auf, Zustand ist *idle*. Der Beispielablauf 2 entsteht, wenn der das Gespräch initiierende Teilnehmer zuerst den Hörer auflegt.

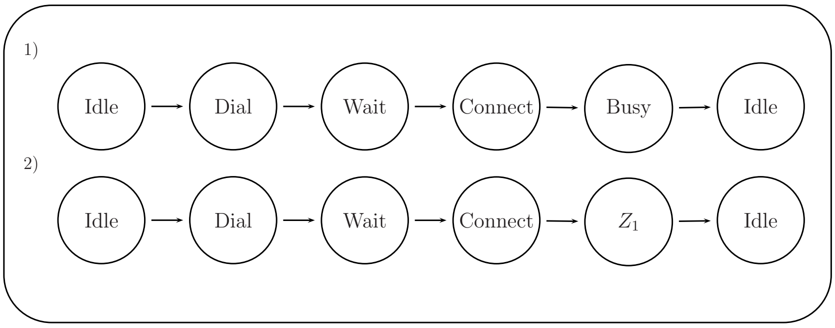


Abbildung 12.3: Beispielablauf des Automaten aus Abb.12.2

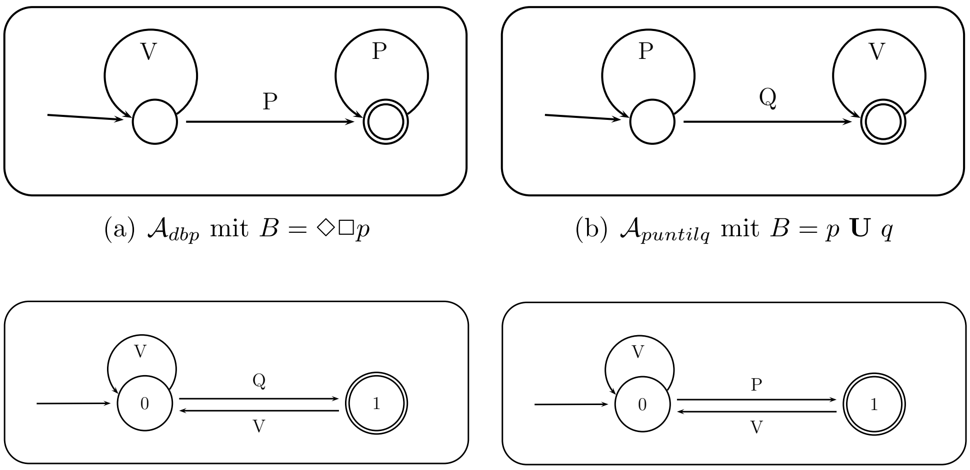
Eine wichtige Aufgabe der formalen Analyse von Kommunikationsprotokollen ist es Ausagen über aller möglichen Abläufe eines Systems zu machen. Zum Beispiel, daß zwischen zwei Auftreten des Zustand *connect* mindestens einmal der Zustand *idle* vorkommen muß. Wie sollen wir Abläufe formal beschreiben? Wir fassen dazu die Kreise in Abbildung 12.3 als abstrakte Zeitpunkte auf. Abtrakt deswegen, weil wir nicht messen wollen wieviel Zeit vergeht um von einem Zeitpunkt zum anderen zu kommen. Die zeitliche Ordnung in vorher und nachher und die Tatsache, das jeder Zeitpunkt einen unmittelbar folgenden hat sind hier ausschlaggebend. In Abbildung 12.3 ist nur ein endliches Anfangsstück eines möglichen Ablaufs gezeigt. Der Vermittlungsautomat hört ja nicht auf zu arbeiten. Deswegen wollen wir auch in unserer Formalisierung annehmen, daß die Folge der abstrakten Zeitpunkte keinen letzten Zeitpunkt aufweist. Mit anderen Worten haben wir uns gerade auf die Ordnungsstruktur der natürlichen Zahlen, $(\mathbb{N}, <)$, festgelegt. Für jeden der sieben Zustände des Automaten aus Abbidung 12.2 führen wir ein aussagenlogisches Atom ein und können dann eine Ablauf beschreiben als eine Kopie der natürlichen Zahlen, aufgefasst als Folge abstrakter Zeitpunkte, wobei zu jedem genau eines der aussagenlogischen Atome wahr ist. Es kann sinnvoll sein noch weitere aussagenlogische Atome einzuführen, z.B. in unserem Szenarium ein Atom OH, das in einem Zeitpunkt wahr ist, wenn zu diesem Zeitpunkt der Hörer des anrufenden Teilnehmers abgehoben ist. In dem Abschnitt werden wir auch zeigen, wie die temporale aussagenlogische Sprache LTL aus Abschnitt 9.1 zur Formulierung temporaler Eigenschaften benutzt werden kann. Insbesondere werden *omega-Strukturen* als Beschreibungsstruktur benutzt.

12.2 Büchi Automaten und LTL

Wir kommen nun zu einem wichtigen Punkt in der Entwicklung der temporalen Logik: Wir wollen einen Zusammenhang herstellen zwischen endlichen Automaten auf der einen Seite und der temporalen Logik LTL auf der anderen.

Wir erinnern zunächst daran, daß wir in Definition 11.19 die Menge $L^\omega(\mathcal{A})$ der von $\mathcal{A}$ akzeptierten unendlichen Wörter definiert hatten. Die in solchen Wörtern vorkommenden Buchstaben aus dem Alphabet V, die auch als Kantenmarkierungen des Automaten auftreten, hatten bisher keine besondere Bedeutung. Das soll sich jetzt ändern. Sei Σ die Menge aller aussagenlogischen Atome einer temporalen Logik, siehe Abbildung 9.3. Wir setzten jetzt das Automatenvokabular $V = 2^\Sigma$. Ein ω-Wort $\xi \in V^\omega$ ist jetzt gleichzeitig eine omega-Struktur. Jetzt können wir das nächste Ziel, realisiert durch Satz 12.3, formulieren. Zu jeder LTL-Formel B wollen wir einen Büchi-Automaten $\mathcal{A}_B$ finden mit

$$L^\omega(\mathcal{A}_B) = \{\xi \in V^\omega \mid \xi \models A\}$$



(a) $\mathcal{A}_{dbp}$ mit $B = \Diamond\Box p$ (b) $\mathcal{A}_{puntilq}$ mit $B = p\ \mathbf{U}\ q$

(c) $\mathcal{A}_{infq}$ mit $B = \Box\Diamond q$ (d) $\mathcal{A}_{infp}$ mit $B = \Box\Diamond p$

Abbildung 12.4: Automat für LTL-Formel B

Beispiel 12.1
Sei Σ ein Menge aussagenlogischer Atome mit $p, q \in \Sigma$, $V = 2^\Sigma$ das entprechende Automatenvokabular und $P = \{b \in V \mid p \in b\}$, $Q = \{b \in V \mid q \in b\}$.

Dann gilt für die Automaten aus der Abbildung 12.4:

$$\begin{array}{lcl} L^\omega(\mathcal{A}_{dbp}) & = & \{\xi \in V^\omega \mid \xi \models \Diamond\Box p\} \\ L^\omega(\mathcal{A}_{puntilq}) & = & \{\xi \in V^\omega \mid \xi \models p \; \mathbf{U} \; q\} \\ L^w(\mathcal{A}_{infp}) & = & \{\xi \in V^\omega \mid \xi \models \Box\Diamond p\} \\ L^w(\mathcal{A}_{infq}) & = & \{\xi \in V^\omega \mid \xi \models \Box\Diamond q\} \end{array}$$

Lemma 12.2
Seien $\mathcal{A}_1 = (S_1, V, s_1^0, \delta_1, F_1)$, $\mathcal{A}_2 = (S_2, V, s_2^0, \delta_2, F_2)$ Büchi-Automaten, C_1, C_2 LTL-Formeln mit $L^\omega(\mathcal{A}_1) = \{\xi \in V^\omega \mid \xi \models C_1\}$ und $L^\omega(\mathcal{A}_2) = \{\xi \in V^\omega \mid \xi \models C_2\}$

Dann gibt es einen Büchi-Automaten $\mathcal{C}$ mit

$$L^\omega(\mathcal{C}) = \{\xi \in V^\omega \mid \xi \models (C_1 \land C_2)\}$$

Wir benutzen gelegentlich die Notation $\mathcal{C} = \mathcal{A}_1 \cap \mathcal{A}_2$.

Beweis Nach Satz 11.24 ist die Richtigkeit des Lemmas schon klar. Der Beweis von Satz 11.24 war jedoch indirekt geführt worden. Wir wollen hier eine explizite Konstruktion von $\mathcal{C}$ aus $\mathcal{A}_1$ und $\mathcal{A}_2$ vorführen. Wir beginnen mit einem Beispiel. Ein erster Versuch könnte sein, für $\mathcal{C}$ das direkte Produkt von $\mathcal{A}$ und $\mathcal{B}$ einzusetzen. Das direkte Produkt von $\mathcal{A}_{infp}$ und $\mathcal{A}_{infq}$ ist in Abbildung 12.5 zu sehen.

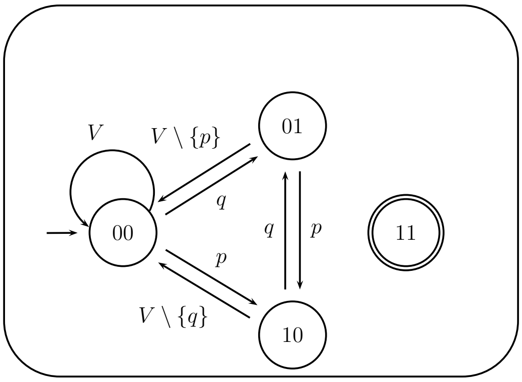


Abbildung 12.5: Erster Versuch eines Automaten für $(\Box\Diamond p) \land (\Box\Diamond q)$

Man sieht, daß dieser Automat überhaupt kein ω-Wort akzeptiert. In Abbildung 12.6 ist der korrekte Automat zu sehen. Wir haben die beiden unerreichbaren Zuständen dieses Automaten gleich ganz weggelassen.

Im allgemeinen kann $\mathcal{C} = (S, V, s^0, \delta, F)$ wie folgt konstruiert werden.

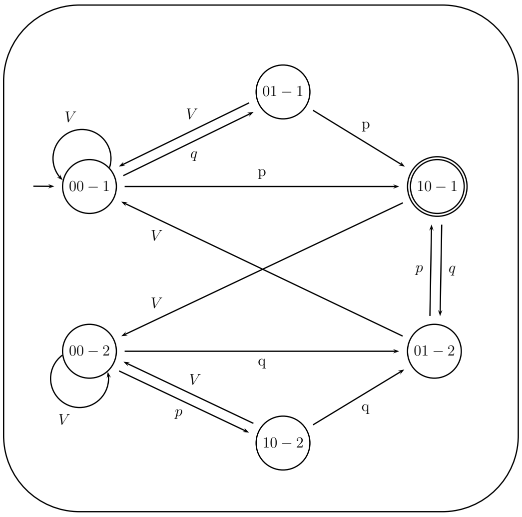


Abbildung 12.6: Ein Automaten für $(\Box\Diamond p) \wedge (\Box\Diamond q)$

$$\begin{array}{lcl}
S & = & S_1 \times S_2 \times \{1,2\} \\
s^0 & = & (s_1^0, s_2^0, 1) \\
F & = & F_1 \times S_2 \times \{1\} \\
\multicolumn{3}{l}{\text{für alle } s_1 \in S_1, s_2 \in S_2, i \in \{1,2\}} \\
(t_1, t_2, i) \in \delta^0((s_1, s_2, i), a) & \Leftrightarrow & t_1 \in \delta_1(s_1, a) \text{ und } t_2 \in \delta_2(s_2, a) \\
\multicolumn{3}{l}{\text{falls } s_1 \in F_1} \\
(t_1, t_2, 2) \in \delta^1((s_1, s_2, 1), a) & \Leftrightarrow & t_1 \in \delta_1(s_1, a) \text{ und } t_2 \in \delta_2(s_2, a) \\
\multicolumn{3}{l}{\text{falls } s_2 \in F_2} \\
(t_1, t_2, 1) \in \delta^2((s_1, s_2, 2), a) & \Leftrightarrow & t_1 \in \delta_1(s_1, a) \text{ und } t_2 \in \delta_2(s_2, a)
\end{array}$$

$$\delta((s_1, s_2, 1), a) = \delta^0((s_1, s_2, 1), a) \cup \delta^1((s_1, s_2, 1), a)$$
$$\delta((s_1, s_2, 2), a) = \delta^0((s_1, s_2, 2), a) \cup \delta^2((s_1, s_2, 2), a)$$

Die Idee dahinter ist, den Produktautomaten zweimal zu benutzen. Die Zustände von Teil 1 sind alle Tripel $(s_1, s_2, 1)$ mit $s_i \in S_i$, die Zustände von Teil 2 sind entsprechend die Tripel $(s_1, s_2, 2)$. Innerhalb einer der beiden Teile sind auf jeden Fall die Übergänge des Produktautomaten möglich, das wird durch δ^0 definiert. Ist s_1 ein Endzustand des ersten Automaten, so gibt es zusätzlich Transitionen von $(s_1, s_2, 1)$ in den 2. Teil von $\mathcal{C}$. Das beschreibt δ^1. Ist s_2 ein Endzustand des zweiten Automaten, dann gibt es außerdem Transitionen von $(s_1, s_2, 2)$ in den ersten Teil. Das wird durch δ^2 beschrieben.

Fortsetzung des Beweises von Lemma 12.2 Ist $r = r_1, r_2, \ldots, r_n \ldots$ eine Berechnungsfolge des Automaten $\mathcal{C}$, wobei $r_n = (s_n^1, s_n^2, j_n) \in S$, dann ist

$s^1 = s^1_1, s^1_2, \ldots, s^1_n \ldots$ eine Berechnungsfolge für $\mathcal{A}_1$ und $s^2 = s^2_1, s^2_2, \ldots, s^2_n \ldots$ eine Berechnungsfolge für $\mathcal{A}_2$. Ist r eine akzeptierende Berechnungsfolge, dann muß eine Endzustand $(s, t, 1)$ mit $s \in F_1$ unendlich oft als r_n vorkommen. Damit wäre auf jeden Fall s^1 eine akzeptierende Berechnungsfolge für $\mathcal{A}_1$. Wird aber ein Zustand $(s, t, 1)$ erreicht, dann ist der nächste Zustand von der Form $(s', t', 2)$. Von Zuständen mit der Endziffer 2 kommt man zu einem Zustand mit der Endziffer 1 nur wieder zurück, indem ein Zustand der Form $(s, t, 2)$ mit $t \in F_2$ durchlaufen wird. Daraus folgt, daß auch in s^2 ein Endzustand aus F_2 unendlich oft vorkommen muß.

∎

Satz 12.3
Zu jeder LTL-Formel B gibt es einen Büchi-Automaten $\mathcal{A}_B$ mit

$$L^\omega(\mathcal{A}_B) = \{\xi \in V^\omega \mid \xi \models B\}.$$

Beweis: Wir fixieren eine Menge P von aussagenlogischen Atomen und eine LTL-Formel $B \in LTLFor_P$. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß in der Formel B nur die temporalen Operatoren $\mathbf{U}$, $\mathbf{V}$ und X vorkommen, siehe Lemma 9.6. Wir können weiter annehmen, daß B in Negationsnormalform vorliegt, d.h. daß Negationszeichen $\neg$ nur vor Atomen vorkommt., siehe Lemma 9.9.

Wir geben explizit einen erweiterten Büchi Automaten $\mathcal{A}_B = (V, S, S_0, \delta, \mathcal{F})$ mit eine Menge von Anfangszuständen an, so daß

$$L^\omega(\mathcal{A}_B) = \{\xi \in V^\omega \mid \xi \models B\}$$

gilt. Der Rest des Beweises wird dann durch die Lemmata 11.31 und 11.33 erledigt.

Wir kommen zur Definition $\mathcal{A}_B$

Vokabular $V = 2^P$

Zustände Sei $subF(B)$ die Menge aller Teilformeln von B.

$$\begin{array}{lll} S & = & \{s \subseteq subF(B) \mid & \mathbf{0} \notin s \\ & & & \text{wenn } (C_1 \wedge C_2) \in s \text{ dann } C_1 \in s \text{ und } C_2 \in s \\ & & & \text{wenn } (C_1 \vee C_2) \in s \text{ dann } C_1 \in s \text{ oder } C_2 \in s \\ & & \} \end{array}$$

Anfangszustände $S_0 = \{s \in S \mid B \in s\}$

Übergangsfunktion Für $s, t \in S$ und $a \in 2^P$ gilt $t \in \delta(s, a)$ wenn alle folgenden Bedingungen erfüllt sind

1. Für alle $p \in P$ mit $p \in s$ gilt $p \in a$.
2. Für alle $p \in P$ mit $\neg p \in s$ gilt $p \notin a$.
3. Falls $X\ A \in s$ dann $A \in t$.
4. Falls $A\ \mathbf{U}\ B \in s$, dann gilt $B \in s$ oder $A \in s$ und $A\ \mathbf{U}\ B \in t$.
5. Falls $A\ \mathbf{V}\ B \in s$, dann $(B \in s$ und $A \in s)$ oder $B \in s$ und $A\ \mathbf{V}\ B \in t$.

Endzustandsmengen Seien $E_i = A_i\ \mathbf{U}\ B_i$ für $1 \leq i \leq k$ alle Formeln der Form $A\ \mathbf{U}\ B$, die in $subF(B)$ vorkommen. Dann ist $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$ mit

$$\mathcal{F}_i = \{s \in S \mid E_i \notin s \text{ oder } E_i \in s \text{ und } B_i \in s\}$$

Damit ist die Definition des Automaten $\mathcal{A}_B$ abgeschlossen und wir gehen nun daran zu beweisen, daß er die gewünschten Eigenschaften hat.

Behauptung 1 $L^\omega(\mathcal{A}_B) \subseteq \{\xi \in V^\omega \mid \xi \models B\}$
Sei $\xi \in L^\omega(\mathcal{A}_B)$ und $r = r_0, \ldots r_n, \ldots$ eine akzeptierende Berechnungsfolge des Automaten $\mathcal{A}_B$ für ξ (siehe Definition 11.19). Dann zeigen wir für jede Formel $C \in subF(B)$ und jedes $i \geq 0$

$$C \in r_i \Rightarrow \xi_i \models C$$

Da in jedem Fall $r_0 \in S_0$ gilt und B eine Element jedes Anfangszustands ist, folgt aus der Behauptung 1 insbesondere $\xi_0 = \xi \models B$.

Beweis von Behauptung 1 Durch strukturelle Induktion über die Komplexität der Formel C. Nach der Definition einer Berechnugsfolge gilt $r_{i+1} \in \delta(r_i, \xi(i))$, davon und von der obigen Definition der Transitionsrelation δ werden wir intensiven Gebrauch machen.

C ist Literal
Aus den Bedingungen 1 und 2 der Definition von δ folgt schon die Behauptung. Man muß sich an dieser Stelle erinnern, daß $p \in r_i$ gleichbedeutend mit $\xi(i) \models p$ und $p \notin r_i$ gleichbedeutend mit $\xi(i) \models \neg p$ ist.

$C = C_1 \wedge C_2$ **oder** $C = C_1 \vee C_2$
In diesem Fall greifen wir auf die Definition der Menge der Zustände S zurück. Wenn $(C_1 \wedge C_2) \in r_i$ ($(C_1 \vee C_2) \in r_i$) dann wurde verlangt daß $C_1 \in r_i$ und $C_2 \in r_i$ ($C_1 \in r_i$ oder $C_2 \in r_i$) gilt. Nach Induktionsvorausetzung folgt $\xi_i \models C_1$ und $\xi_i \models C_2$ ($\xi_i \models C_1$ oder $\xi_i \models C_2$) und damit auch $\xi_i \models C_1 \wedge C_2$ ($\xi_i \models C_1 \vee C_2$).

$C = C_1$ **U** C_2
Wir behaupten, daß es ein $i \leq j$ gibt mit $C_2 \in r_j$ und $C_1 \in r_{j'}$ für alle j' mit $i \leq j' < j$. Nach Definition von δ könnte $C_2 \in r_i$ gelten und wir hätten j gefunden für $j = i$. Anderenfalls gilt $C_1 \in r_i$ und $C_1 \mathbf{U} C_2 \in r_{i+1}$ und wir können dasselbe Argumente für $i+1$ anstelle von i wiederholen. Wenn wir in dieser Iteration auf ein erstes j stoßen mit $C_2 \in r_j$ dann haben wir es geschafft. Andererseits können wir noch nicht die Möglichkeit ausschließen, daß für alle $j \geq i$ gilt $C_2 \notin r_j$. Wir wir gesehen haben folgt daraus $C_1 \mathbf{U} C_2 \in r_j$ für alle $j \geq i$. An dieser Stelle erinnern wir uns daran, daß die akzeptierende Berechnung r unendlich oft die Endzustandsmenge $\mathcal{F}_k = \{s \in S \mid C_1 \mathbf{U} C_2 \notin s \text{ oder } C_1 \mathbf{U} C_2 \in s \text{ und } C_2 \in s\}$ trifft. (Wir haben angenommen, daß $C_1 \mathbf{U} C_2 = E_k$.) Insbesondere gibt es ein j mit $i \leq j$ mit $r_j \in \mathcal{F}_k$. Beide Annahmen $C_1 \mathbf{U} C_2 \in r_j$ und $C_2 \notin r_j$ zusammengenommen, stehen dazu im Widerspruch. Also muß es j mit $j \geq i$ und $C_2 \in r_j$ geben. Wir wählen ein kleinstes j mit dieser Eigenschaft. Nach den bisher gemachten Annahmen folgt jetzt $C_2 \in r_j$ und $C_1 \in r_{j'}$ für $i \leq j' < j$. Nach Induktionsvoraussetzung folgt $\xi_j \models C_2$ und $\xi_{j'} \models C_1$ für alle $i \leq j' < j$. Also insgesamt $\xi_i \models C_1 \mathbf{U} C_2$.

C_1 **V** C_2
Aus Teil 5 der Definition von δ folgt auf jeden Fall für alle $j \geq i$, wenn $C_1 \mathbf{V} C_2 \in r_j$, dann auch $C_2 \in r_j$. Somit haben wir auf jeden Fall schon einmal $C_2 \in r_i$ sicher. Aus der Induktionsvoraussetzung folgt damit $\xi_i \models C_2$. Gilt jetzt für ein $j+1 > i$ die Aussage $\xi_{j+1} \models \neg C_2$, dann folgt mit der Induktionsvoraussetzung $C_2 \notin r_{j+1}$ und nach der obigen Überlegung auch $C_1 \mathbf{V} C_2 \notin r_{j+1}$. Nach Teil 5 muß daher $C_1 \in r_j$ gelten, was nach Induktionsvoraussetzung wieder $\xi_j \models C_1$ liefert. Insgesamt ist damit $\xi \models C_1 \mathbf{V} C_2$ gezeigt.

Behauptung 2 $\{\xi \in V^\omega \mid \xi \models B\} \subseteq L^\omega(\mathcal{A}_B)$
d.h. zu jedem $\xi \in V^\omega$ mit $\xi \models B$ gibt es eine akzeptierende Berechnung $r = r_0, \ldots, r_n \ldots$ von $\mathcal{A}_B$ mit $r_{i+1} \in \delta(r_i, \xi(i))$ für alle $i \geq 0$.

Beweis von Behauptung 2 Wir setzen $r_i = \{C \in subF(B) \mid \xi_i \models C\}$. Man prüft leicht nach, daß für alle $i \geq 0$ gilt $r_i \in S$, $r_{i+1} \in \delta(r_i, \xi(i))$ und schließlich auch, daß $r = r_0, \ldots, r_n, \ldots$ eine akzeptierende Berechnungsfolge für $\mathcal{A}_B$.

■

Korollar 12.4
Erfüllbarkeit und Allgemeingültigkeit von LTL Formeln ist entscheidbar.

Beweis: Nach dem im vorangegangenen Satz beschriebenen Verfahren konstruiert man die Büchi Automaten $\mathcal{A}_B$ und $\mathcal{A}_{\neg B}$. Es gilt

B ist erfüllbar	$\Leftrightarrow$	$L^\omega(\mathcal{A}_B) \neq \emptyset$
B ist allgemeingültig	$\Leftrightarrow$	$L^\omega(\mathcal{A}_{\neg B}) = \emptyset$

Nach Satz 11.20 sind wir fertig.

■

12.2.1 Übungsaufgaben

Übungsaufgabe 12.2.1
Finden Sie einen Büchi-Automaten $\mathcal{A}_V$, so daß $L^\omega(\mathcal{A}_V)$ genau die omega-Strukturen sind, die $p\ \mathbf{V}\ q$ erfüllen.

Übungsaufgabe 12.2.2
Sei Σ eine aussagenlogische Signatur, mit $p, q, r \in \Sigma$. Finden Sie einen Büchi-Automaten $\mathcal{B}_{UU}$, der genau die omega-Strukturen (in der üblichen Weise als ω-Wörter repräsentiert) akeptiert, welche die LTL-Formel $p\ \mathbf{U}\ q\ \mathbf{U}\ r$ erfüllen.

Übungsaufgabe 12.2.3
In Abb. 12.5 wird ein Automat gezeigt, den genau die ω-Wörter akzeptiert, in denen sowohl p als auch q unendlich oft vorkommen. Der Automat diente als Beispiel für die allgemeine Konstruktionsmethode aus Lemma 12.2. Es gibt wesentlich einfachere Büchi Automaten für diese Sprache. Finden Sie einen mit 3 Zuständen.

Übungsaufgabe 12.2.4
Im Beweis von Lemma 12.2 haben wir schon ein Gegenbeispiel gesehen zur Gleichung $L^\omega(\mathcal{A}_1 \times \mathcal{A}_1) = L^\omega(\mathcal{A}_1) \cap L^\omega(\mathcal{A}_1)$.

1. Finden Sie ein weiteres möglichst einfaches Gegenbeispiel.

2. Zeigen Sie, daß die Inklusion $L^\omega(\mathcal{A}_1 \times \mathcal{A}_1) \subseteq L^\omega(\mathcal{A}_1) \cap L^\omega(\mathcal{A}_1)$ immer gilt.

3. Zeigen Sie, daß $L^\omega(\mathcal{A}_1 \times \mathcal{A}_1) = L^\omega(\mathcal{A}_1) \cap L^\omega(\mathcal{A}_1)$ gilt, wenn in $\mathcal{A}_1$ jeder Zustand ein Endzustand ist.

12.3 Intervallzeitlogik (Zusatzstoff)

Dieses Unterkapitel wurde im wesentlichen von Mattias Ulbrich zusammengestellt.

In Kapitel 9 wurde die lineare temporale Logik LTL vorgestellt. In Satz 12.3 wurde bewiesen, daß es für jede LTL-Formel über einer Signatur Σ einen akzeptierenden Büchi-Automaten gibt. Umgekehrt gibt es aber Büchi-Automaten über demelben Alphabet, für die es **keine** LTL-Formel gibt, für die sie ein akzeptierender Automat wären.

Abbildung 12.7: Büchi-Automaten ohne (a) bzw. mit (b) akzeptierter LTL-Formel

Als Beispiel sei hierfür der Automat $\mathcal{B}_{2p}$ aus Abb. 12.7a angeführt, der von eine omega-Struktur genau dann akzeptiert, wenn in (mindestens) jedem zweiten Zeitpunkt p zu wahr ausgewertet wird. Dies lässt sich nicht in LTL ausdrücken. Übungsaufgabe (ref!8.1.2) zeigt aber, dass es eine Formel gibt, falls man fordert, dass p **genau** in jedem zweiten Zeitpunkt zu wahr ausgewertet wird.

Es gibt eine verwandte Zeitlogik, deren Mächtigkeit genau der der Büchi-Automaten entspricht: Die **Intervallzeitlogik** (*Interval Temporal Logic ITL*) wurde von Ben Moszkowski im Rahmen seiner Dissertation eingeführt und in [Mos86] publizierte. Für jede Formel in ITL gibt es einen akzeptierenden Büchi-Automaten, und, umgekehrt, für jeden Büchi-Automaten gibt es eine ITL-Formel, für die er akzeptierender Automat ist.

12.3.1 Syntax und Semantik von ITL

Definition 12.5 (ITL-Formeln)
Sei Σ eine *endliche Menge* aussagenlogischer Atome. Die Menge Fml_{ITL} der ITL-Formeln über der Signatur Σ ist die kleinste Menge, für die gilt:

1. $\Sigma \subset Fml_{\mathrm{ITL}}$

2. Sind $f_1, f_2 \in Fml_{\mathrm{ITL}}$ ITL-Formlen, dann sind auch $f_1 \wedge f_2 \in Fml_{\mathrm{ITL}}$ und $\neg f_1 \in Fml_{\mathrm{ITL}}$ ITL-Formeln.

3. **skip** $\in Fml_{\mathrm{ITL}}$

4. Sind $f_1, f_2 \in Fml_{\mathrm{ITL}}$ ITL-Formlen, dann sind auch $f_1 ; f_2 \in Fml_{\mathrm{ITL}}$ und $f_1^* \in Fml_{\mathrm{ITL}}$ ITL-Formeln.

Die Formel $f_1 ; f_2$ wird als "f_1 Schnitt f_2" (oder "f_1 chop f_2") und f^* als "f Stern" (oder "f chop star") gelesen.

Anders als die LTL, in der Formeln nur über unendlichen omega-Strukturen $\xi : \mathbb{N} \to 2^{\Sigma}$ ausgewertet wurden, können in ITL auch *endliche* Zeitstrukturen $([0, k], \xi)$ zur Auswertung einer Formel herangezogen werden. Die Intervallschreibweise ist dabei für $a, b \in \mathbb{N}$ mit $a \leq b$ definiert durch $[a, b] := \{a, a+1, \ldots, b\}$ mit $\xi : [0, k] \to 2^{\Sigma}$. Ganz allgemein wird der Definitionsbereich D der Zeitstruktur (D, ξ) Intervall genannt und ist der endliche oder unendliche Abschnitt der diskreten Zeit, für den eine Formel ausgewertet wird.

In unserer Definition der Logik gilt für jedes Intervall D, dass

1. entweder es ein $k \in \mathbb{N}$ gibt mit $D = [0, k]$ oder

2. $D = \mathbb{N}$ gilt.

Für die nachstehenden Definitionnen ist es hilfreich, den Anfangspunkt einer Zeitstruktur variieren zu können. Ist M eine Zeitstruktur, dann ist M^d die beim Zeitpunkt d beginnende Teilstruktur von M. Weiterhin brauchen wir die Zeitstruktur $M^{d,e}$, die der Teilstruktur von M zwischen d und e entspricht.

Definition 12.6 (Teilzeitstrukturen)

1. Sei $M = (D, \xi)$ eine Zeitstruktur, $d \in \mathbb{N}$, falls $D = [0, k]$ verlangen wir $d \leq k$, falls $D = \mathbb{N}$ gibt es keine Einschränkung an d. Die Zeitstruktur M^d ist definiert durch
   $(\mathbb{N}, \xi)^d = (\mathbb{N}, \xi_d)$
   $([0, k], \xi)^d = ([0, k-d], \xi_d)$
   mit $\xi_d(n) := \xi(d+n)$

2. Sei $M = (D, \xi)$ eine Zeitstruktur, $0 \leq d \leq e$, falls $D = [0, k]$ verlangen wir $e \leq k$, falls $D = \mathbb{N}$ gibt es keine Einschränkung an e. Die Zeitstruktur $M^{d,e}$ ist definiert durch
   $M^{d,e} := ([0, e-d], \xi_d^e)$
   wobei $\xi_d^e : [0, e-d] \to 2^{\Sigma}$ die Einschränkung der Funktion $\xi_d : \mathbb{N} \to 2^{\Sigma}$ auf den Definitionsbereich $[0, e-d]$ ist,

Definition 12.7 (Semantik der ITL)
Sei $M = (D, \xi)$ eine Zeitstruktur, dann gilt für ITL-Formeln $f_1, f_2 \in Fml_{\mathrm{ITL}}$, $p \in \Sigma$.

$M \models p$	gdw.	$p \in \xi(0)$
$M \models f_1 \land f_2$	gdw.	$M \models f_1$ und $M \models f_2$
$M \models \neg f_1$	gdw.	$M \not\models f_1$
$M \models$ **skip**	gdw.	$D = [0, 1]$
$M \models f_1 \,;\, f_2$	gdw.	*(i)* $D = [0..k]$ und
		es gibt l, $0 \leq l < k$, mit $M^{0,l} \models f_1$ und $M^{l,k} \models f_2$
		oder (ii) $D = \mathbb{N}$ und
		es gibt l mit $M^{0,l} \models f_1$ und $M^{l} \models f_2$
		oder (iii) $D = \mathbb{N}$ und $M \models f_1$
$M \models f_1^*$	gdw.	*(i)* $D = [0..k]$ *endlich:*
		es gibt Indizes $0 = k_o < k_1 < \ldots < k_r = k$ mit $M^{k_i,k_{i+1}} \models f_1$
		(ii) $D = \mathbb{N}$ *unendlich:*
		Es gibt (unendl. viele) Indizes $k = k_o < k_1 < k_2 < \ldots$
		mit $M^{k_i,k_{i+1}} \models f_1$
		oder
		es gibt (endl. viele) Indizes $k = k_o < k_1 < ... < k_r$
		mit $M^{k_i,k_{i+1}} \models f_1$ and $M^{k_r} \models f_1$

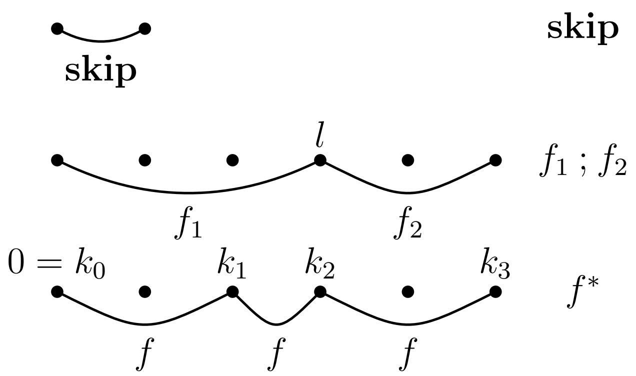


Abbildung 12.8: Graphische Veranschaulichung der Semantik von ITL

Mit Hilfe der temporalen Operatoren kann bei der ITL die Auswertung einer Zeitstruktur auf die Auswertung von Teilstrukturen reduziert werden. Durch das **skip**-Konstrukt können auch Bedingungen an die Länge der Teilsequenz gestellt werden.

Beispiel 12.8

$$
\begin{array}{lll}
M \models \textbf{skip} \,;\, \textbf{skip} & \text{gdw.} & \text{Es gibt } l \text{ mit } M^{0,l} \models \textbf{skip} \text{ und } M^l \models \textbf{skip} \\
& \text{gdw.} & |M^{0,l}| = 1 \text{ und } |M^l| = 1 \\
& \text{gdw.} & l - 0 = 1 \text{ und } D = [0,k] \text{ und } k - l = 1 \\
& \text{gdw.} & k = 2 \text{ gdw. } D = [0,2]
\end{array}
$$

Die (endl.) Länge eines Intervalls ist also axiomatisierbar, d.h. für jedes $k \in \mathbb{N}$ gibt es eine Formel, so dass genau die Zeitstrukturen der Länge k die Modelle der Formel sind.

Beispiel 12.9
Betrachten wir für das folgende Beispiel eine unendliche Struktur M_∞

$$
\begin{array}{lll}
M_\infty \models (p \wedge \textbf{skip})^* & \text{gdw.} & \text{Es gibt Indizes } k = k_o < k_1 < \ldots, \text{ so dass } M_\infty^{k_i,k_{i+1}} \models p \wedge \textbf{skip} \text{ (wobei das letzte Intervall auch unendlich sein könnte)} \\
& \text{gdw.} & \text{Es gibt Indizes } k = k_o < k_1 < \ldots, \text{ so dass } M_\infty^{k_i,k_{i+1}} \models p \text{ und } k_{i+1} - k_i = 1 \\
& \text{gdw.} & \text{Für alle } i \in \mathbb{N} \text{ gilt } M_\infty^{i,i+1} \models p \\
& \text{gdw.} & \text{Für alle } i \in \mathbb{N} \text{ gilt } p \in \xi(i)
\end{array}
$$

Ebenso zeigt man

$$
([0..k], \xi] \models (p \wedge \textbf{skip})^* \quad \text{gdw.} \quad \text{Für alle } i,\ 0 \leq i < k \text{ gilt } p \in \xi(i)
$$

Beispiel 12.10
Die Formel $\mathbf{1};\mathbf{0}$ kürzen wir mit **infinite** ab, dabei ist $\mathbf{1}$ die stets wahre aussagenlogische Konstante und $\mathbf{0}$ die stets falsche aussagenlogische Konstante. Es gilt:

$$
M \models \textbf{infinite} \ \Leftrightarrow\ M \text{ ist unendlich}
$$

Wäre M endlich so verlangt die Semantikdefinition, daß ein d existiert mit $M^{0,d} \models \mathbf{1}$ und $M^d \models \mathbf{0}$. Die letze Aussage kann nie gelten. Also muß M unendlich sein. In diesem Fall ist $M \models \mathbf{1};\mathbf{0}$ äquivalent zu $M \models \mathbf{1}$.

Bezeichnen wir mit **finite** die Formel $\neg$**infinite** dann haben wir offensichtlich

$$
M \models \textbf{finite} \ \Leftrightarrow\ M \text{ ist endlich}
$$

12.3.2 Ausdruckstärke von ITL

Es ist naheliegend ITL Formeln mit ω-regulären Ausdrücken, siehe Def.11.36, zu vergleichen: der ;-Operator entspricht kanonisch der Konkatenation von regulären Ausdrücken und der $*$-Operator der Logik entspricht der Wiederholungsverknüpfung. Wir werden in den beiden nächsten Sätzen zeigen, daß beide Formalismen die gleiche Ausdruckskraft haben.

Satz 12.11
Sei Σ eine Signatur für ITL, d.h. Σ ist die Menge der Atome, die in ITL auftreten können und $V = 2^\Sigma$ das Vokabular aus dem die Wörter gebildet werden, die durch ω-reguläre Ausdrücke beschriebenen werden. (Dieser Zusammenhang zwischen Σ und V wurde schon zu Beginn des Abschnitts 12.2 eingeführt.)

Zu jedem ω-regulären Ausdruck t über dem Vokabular V gibt es eine ITL-Formel $itl(t)$ in der Signatur Σ, so daß

$$S(t) = \{\xi \mid \xi \models itl(t)\}$$

$S(t)$ ist dabei die Auswertungsfunktion für ω-reguläre Ausdrücke, wie sie in Definition 11.37 gegeben wurde. Wörter aus $V^* \cup V^\omega$ werden als endliche bzw unendliche Zeitstukturen interpretiert, wie das schon zu Beginn des Abschnitts 12.2 für unendliche Wörter geschehen war.

Beweis Die Transformation itl wird in Tabelle 12.1 definiert:

	reg. Expr.	**ITL**
	t	$\mathrm{itl}(t)$
Vokabular	$M \in 2^\Sigma$	$\mathbf{skip} \wedge \bigwedge_{p \in M} p \wedge \bigwedge_{p \notin M} \neg p$
Konkatenation	$t_1 t_2$	$\mathrm{itl}(t_1)\ ;\ \mathrm{itl}(t_2)$
Unterscheidung	$t_1 + t_2$	$\mathrm{itl}(t_1) \vee \mathrm{itl}(t_2)$
Wiederholung	t^*	$\mathrm{itl}(t)^* \wedge \mathbf{finite}$
ω-Rekursion	t^ω	$\mathrm{itl}(t)^* \wedge \mathbf{infinite}$

Tabelle 12.1: Von reg. Ausdrücken zu ITL-Formeln

Der Beweis, dass diese Abbildung korrekt ist, erfolgt natürlich über strukturelle Induktion über reguläre Ausdrücke. Betrachten wir als ein Beispiel den Stern-Operator. Nach Induktionsvoraussetzung gilt für jede Zeitstruktur $M = (D, \xi)$, dass $\xi \in S(t)$ gdw. $\xi \models \mathrm{itl}(t)$. Wir wollen $S(t^*) = \{\xi \mid \xi \models itl(t^*)\}$ zeigen.

Es gilt:

$$\begin{array}{lcl} M \models \mathrm{itl}(t)^* \wedge \textbf{finite} & \Leftrightarrow & D = [0..k] \text{ endlich und es gibt Indizes} \\ & & 0 = k_o < k_1 < \ldots < k_r = k \text{ mit} \\ & & M^{k_i,k_{i+1}} \models \mathrm{itl}(t) \text{ für } 0 \leq i < r \\ & \overset{\text{Ind-vor.}}{\Longleftrightarrow} & D = [0..k] \text{ endlich und es gibt Indizes} \\ & & 0 = k_o < k_1 < \ldots < k_r = k \text{ mit} \\ & & \xi^{k_i,k_{i+1}} \in S(t) \text{ für } 0 \leq i < r \\ & \Leftrightarrow & \xi \in S(t^*) \end{array}$$

Die anderen Fälle analog.

■

Korollar 12.12

Sei Σ eine Signatur. Es gibt zu jeder LTL-Formel F eine äquivalente ITL-Formel F_{itl}, das soll heißen: für jede unendliche Zeitstruktur ξ in der Signatur Σ gilt

$$\xi \models F \text{ gdw } \xi \models F_{itl}$$

Beweis Folgt aus Satz 12.3 (von LTL zu Büchi Automat), Übungsaufgabe 11.2.17 (von Büchi Automat zu ω-regulärem Ausdruck) und dem vorangegangenen Satz 12.11 (von ω-regulärem Ausdruck zu ITL).

■

Satz 12.13

Seien Σ und V wie im vorangegangen Satz 12.11.

Zu jeder ITL-Formel F in der Signatur Σ gibt es einen regulären Ausdruck $reg(F)$ und einen ω-regulären Ausdruck $oreg(F)$ über dem Vokabular V, so daß

$$S(reg(F)) \cup S(oreg(F)) = \{\xi \mid \xi \models F\}$$

Beweis Zu jeder ITL Formel F werden die Ausdrücke $reg(F)$ und $oreg(F)$ in Tabelle 12.2 definiert.

In dieser Tabelle werde Durchschnitt $t_1 \cap t_2$ und Komplement $t \setminus V^\omega$ bzw. $t \setminus V^*$ benutzt. Das sind keine Operatoren aus der Basissyntax. Es ist aber bekannt, daß es reguläre, bzw. ω-reguläre Ausdrücke in der Basissyntax mit derselben Semantik gibt.

Es bleibt zu zeigen, daß reg und $oreg$ die geforderten Eigenschaften haben.

	ITL	endl. reg. Ausdr.	ω-reg. Ausdruck
	F	$reg(F)$	$oreg(F)$
AL-Atome	$p \in \Sigma$	PV^*	PV^ω
Negation	$\neg F$	$V^* \setminus reg(F)$	$V^\omega \setminus oreg(F)$
Konjunktion	$F \wedge G$	$reg(F) \cap reg(G)$	$oreg(F) \cap oreg(G)$
Schritt	**skip**	V	$\emptyset$
Schnitt	$F\,;G$	$reg(F)reg(G)$	$reg(F)oreg(G) + oreg(F)$
Stern	F^*	$reg(F)^*$	$reg(F)^*oreg(F) + reg(F)^\omega$

Tabelle 12.2: Von ITL-Formeln zu reg. Ausdrücken

Der Beweis geht hier über die Struktur der Formel. Betrachten wir wieder exemplarisch den Sternoperator. Es gilt:

1. ξ endlich: $\xi \models F^*$ $\iff$ $D = [0..k]$ und es gibt Indizes $0 = k_o < k_1 < \ldots < k_r = k$ mit $\xi^{k_i,k_{i+1}} \models F$ für $0 \leq i < r$

   $\overset{\text{Ind-vor.}}{\iff}$ $D = [0..k]$ endlich und es gibt Indizes $0 = k_o < k_1 < \ldots < k_r = k$ mit $\xi^{k_i,k_{i+1}} \in S(reg(F)) \cup S(oreg(F))$ für $0 \leq i < r$

   $\iff$ $\xi^{k_i,k_{i+1}} \in S(reg(F))$, da endliche Folgen

   $\iff$ $\xi \in V(reg(F)^*) = V(reg(F^*))$

2. ξ unendlich: $\xi \models F^*$ $\iff$ Es gibt $k = k_o < k_1 < \ldots$ mit $\xi^{k_i,k_{i+1}} \models f_1$
   oder
   es gibt (endl. viele) Indizes $k = k_o < \ldots < k_r$ mit $\xi^{k_i,k_{i+1}} \models F$ und $\xi^{k_r} \models F$

   $\overset{\text{Ind-vor.}}{\iff}$ Es gibt $k = k_o < k_1 < \ldots$ mit $\xi^{k_i,k_{i+1}} \in S(reg(F))$
   oder
   es gibt (endl. viele) Indizes $k = k_o < \ldots < k_r$ mit $\xi^{k_i,k_{i+1}} \in S(reg(F))$ und $\xi^{k_r} \in S(oreg(F))$

   $\iff$ $\xi \in S(reg(F)^\omega \cup reg(F)^*oreg(F)) = S(oreg(F^*))$

12.3.3 Konstruktive Transformation von LTL in ITL Noch in Arbeit

Nach Korollar 12.12 gibt es zu jeder LTL Formel F eine äquivalente ITL Formel $Itl(F)$. Allerdings ist das eine reine Existenzaussage und liefert kein brauchbares Verfahren $Itl(F)$ direkt aus F zu berechnen. Gibt es solches Verfahren? Dieser Frage wollen wir in diesem Unterabschnitt nachgehen.

Definition 12.14 (Linkseinfache LTL Formeln)

Die Menge der *linkseinfache LTL Formeln* ist induktiv definiert durch

1. für jedes Atom p, sind p, $\neg p$, $X\ \ p$ und $X\ \neg p$ linkseinfach.
2. sind F_1, F_2 linkseinfach, dann auch $F_1 \vee F_2$ und $F_1 \wedge F_2$.
3. ist F linkseinfach, G eine beliebige LTL-Formel, dann ist auch $F\ \mathbf{U}\ G$ linkseinfach.

Man beachte, daß per Definition jede linkseinfache Formel F eine Negationsnormalform ist, d.h. daß in F das Negationszeichen nur vor Atomen vorkommt.

Definition 12.15 (ITL-Normalform)

Eine LTL Formel F ist eine *ITL-Normalform*, wenn

1. F eine Negationsnormalform ist,
2. für jede in F vorkommende Teilformel der Form $F_1\ \mathbf{U}\ F_2$ die erste Komponente F_1 eine linkseinfache Formel ist und
3. in F die temporalen Operatoren $\mathbf{U}$ und $\Box$, vorkommen können.

Lemma 12.16

Zu jeder LTL-Formel F gibt es eine äquivalente LTL-Formel $nf(F)$ in ITL-Normalform.

Beweis Sei F eine beliebige LTL-Formel. Durch Anwendung der deMorganschen Regeln und der Tautologien $\neg(A\ \mathbf{U}\ B) \leftrightarrow \neg A\ \mathbf{V}\ \neg B$, $\neg(A\ \mathbf{V}\ B) \leftrightarrow \neg A\ \mathbf{U}\ \neg B$, $\neg\Diamond A \leftrightarrow \Box\neg A$, $\neg\Box A \leftrightarrow \Diamond\neg A$, $\neg X\ A \leftrightarrow X\ \neg A$ transformiere man F in eine Negationsnormalform F_1. Durch wiederholte Anwendung der Tautologie $A\ \mathbf{V}\ B \leftrightarrow \Box B \vee (B\ \mathbf{U}\ (A \wedge B))$ (siehe Übungsaufgabe 9.1.10) transformiere man F_1 in eine äquivalente Negationsnormalform F_2, in der $\mathbf{V}$

nicht mehr vorkommt. Als nächstes ersetzen wir jede Teilformel $A \mathbf{U} B$ von F_2 durch eine äquivalente ITL-Normalform. Das ist ein rekursives Verfahren. Ist $A \mathbf{U} B$ eine Teilformel von F_2, dann bringen wir A in eine aussagenlogische konjunktive Normalform. Wegen der leicht einzusehenden Tautologie $(A_1 \wedge A_2) \mathbf{U} B \leftrightarrow (A_1 \mathbf{U} B \wedge A_2 \mathbf{U} B)$ müssen wir nur noch den Fall betrachten $(A_1 \vee \ldots A_k) \mathbf{U} B$ wobei alle A_i entweder Literale sind oder von der Form $A_i = X \; A_{i0}$, $A_i = \Diamond A_{i0}$, $A_i = A_{i0} \mathbf{U} A_{i1}$ oder $A_i = \Box A_{i0}$. Als Induktionsvoraussetzung können wir annehmen, daß die Formeln A_{i0}, A_{i1} bereits linkseinfach sind. Einzig die Formeln der Form $A_i = \Box A_{i0}$ machen noch Schwierigkeiten. Durch wiederholte Anwendung der Tautologie $(C \vee \Box A) \mathbf{U} B \leftrightarrow B \vee (C \mathbf{U} B) \vee (C \mathbf{U} (\Box A \wedge \Diamond B))$ (siehe Übungsaufgabe 9.1.9) kann man $(A_1 \vee \ldots A_k) \mathbf{U} B$ in eine äquivalente linkseinfache LTL Formel transformieren.

∎

Definition 12.17

Für jede LTL Formel F definieren wir induktiv eine ITL-Formel $Itl(F)$ wie folgt

$$
\begin{array}{rcl}
Itl(A) & := & A \text{ falls } A \text{ ein Literal ist} \\
Itl(A \vee B) & := & Itl(A) \vee Itl(B) \\
Itl(A \wedge B) & := & Itl(A) \wedge Itl(B) \\
Itl(X \; A) & := & \mathbf{skip} \, ; Itl(A) \\
Itl(\Diamond A) & := & \mathbf{finite} \, ; Itl(A) \\
Itl(\Box A) & := & (Itl(A) \wedge \mathbf{skip})^* \\
Itl(A \mathbf{U} B) & := & (Itl(\Box A) \wedge \mathbf{finite}) \, ; Itl(B) \\
 & := & ((Itl(A) \wedge \mathbf{skip})^* \wedge \mathbf{finite}) \, ; Itl(B)
\end{array}
$$

wobei

$$
\begin{array}{rcl}
\mathbf{infinite} & := & \mathbf{1} \, ; \mathbf{0} \\
\mathbf{finite} & := & \neg \mathbf{infinite}
\end{array}
$$

Lemma 12.18

Ist A eine linkseinfache LTL Formel und M eine unendliche Zeitstruktur. Dann gilt für jedes $d \in \mathbb{N}$,mit $d > 0$:
Falls $M^{0,d} \models Itl(A)$, dann auch $M \models Itl(A)$.

Beweis Wir benutzen strukturelle Induktion nach der Definition 12.14 für linkseinfache Formeln. Im folgenden sei $M = (D, \xi)$ eine Zeitstruktur mit einem unendlichen Intervall D.

Ist $A \equiv p$ eine atomare Formel, so gilt

$$\begin{array}{rll} M \models Itl(p) & \text{gdw} & M \models p \\ & \text{gdw} & p \in \xi(0) \\ & \text{gdw} & p \in \xi_0(0) \\ & \text{gdw} & M^{0,d} \models p \\ & \text{gdw} & M^{0,d} \models Itl(p) \end{array}$$

Genauso leicht sieht man, daß für negative Literale gilt

$$M \models Itl(\neg p) \Leftrightarrow M^{0,d} \models Itl(\neg p)$$

Die Induktionsschritte für $F_1 \vee F_2$ und $F_1 \wedge F_2$ sind trivial.

Induktionschritt von F nach $\Diamond F$.

$$\begin{array}{rll} M^{0,d} \models Itl(\Diamond F) & \Leftrightarrow & M^{0,d} \models \mathbf{finite}\,;Itl(A) \\ & \Leftrightarrow & M^{0,k} \models \mathbf{finite} \text{ und } M^{k,d} \models Itl(A) \\ & & \text{für ein } k \text{ mit } 0 \leq k < d \\ & \Rightarrow & M^{0,k} \models \mathbf{finite} \text{ und } M^{k} \models Itl(A) \\ & & \text{Induktionsvoraussetzung} \\ & \Leftrightarrow & M \models \mathbf{finite}\,;Itl(A) \\ & \Leftrightarrow & M \models Itl(\Diamond F) \end{array}$$

Induktionschritt von F nach $X\ F$.

$$\begin{array}{rll} M^{0,d} \models Itl(X\ F) & \Leftrightarrow & M^{0,d} \models \mathbf{skip}\,;Itl(A) \\ & \Leftrightarrow & M^{0,k} \models \mathbf{skip} \text{ und } M^{k,d} \models Itl(A) \\ & & \text{für ein } k \text{ mit } 0 \leq k < d \\ & \Leftrightarrow & M^{1,d} \models Itl(A) \\ & \Rightarrow & M^{1} \models Itl(A) \\ & & \text{Induktionsvoraussetzung} \\ & \Leftrightarrow & M \models \mathbf{skip}\,;Itl(A) \\ & \Leftrightarrow & M \models Itl(X\ F) \end{array}$$

Induktionsschritt von F nach $F\ \mathbf{U}\ G$ für eine beliebige LTL-Formel G.

$$\begin{array}{rll} M^{0,d} \models Itl(F\ \mathbf{U}\ G) & \Leftrightarrow & M^{0,k} \models Itl(\Box F) \wedge \mathbf{finite} \text{ und } M^{k,d} \models Itl(G) \\ & & \text{für ein } k \text{ mit } 0 \leq k < d \end{array}$$

`showstopper`

Satz 12.19

Für jeder LTL-Formel F in NF-Normalform ?? gilt für jede (unendliche) omega Struktur

$$M \models F \text{ gdw } M \models Itl(F)$$

Beweis: Geschieht durch Induktion über den Formelaufbau vob F. Der Induktionsanfang, F ist ein Literal, ist einfach. Ebenso die Induktionsschritte für die aussagenlogischen Operatoren.

$$\begin{array}{lll}
M \models F_1 \mathbf{U} F_2 & \text{gdw} & M^d \models F_2 \text{ für ein } d \text{ und} \\
 & & M^k \models F_1 \text{ für alle } k, 0 \leq k < d \\
 & \text{gdw} & M^d \models Itl(F_2) \text{ für ein } d \text{ und} \\
 & & M^k \models Itl(F_1) \text{ für alle } k, 0 \leq k < d \\
 & & \text{nach Induktionsvoraussetzung}
\end{array}$$

12.4 Modellprüfung für LTL

12.4.1 Problemstellung

Generell wird eine Modellprüfungsaufgabe bestimmt durch die Angabe eines Modells $\mathcal{M}$ und einer Ausagen ϕ. Die Lösung der Aufgabe besteht in der Antwort, ob ϕ für $\mathcal{M}$ wahr ist oder nicht. Auf dieser allgemeinen Stufe lässt sich fast jedes Problem in eine Modellprüfungsaufgabe einkleiden. Wir konzentrieren uns hier auf den Spezialfalls, daß Modelle durch Büchi-Automaten gegeben werden und die Aussagen LTL-Formeln sind. Mehr noch, es muß der in Abschnitt 12.2, Seite 368 beschriebene Zusammenhang zwischen dem Automatenvokabular und den LTL-Atomen gelten, d.h. ist Σ die Menge aller aussagenlogischen Atome einer temporalen Logik, dann ist das Automatenvokabular für die Kantenmarkierungen $V = 2^{\Sigma}$. Es bleibt noch genauer zu erklären, was es heißen soll, daß ein Büchi-Automat $\mathcal{A}$ eine LTL Formel ϕ erfüllt. Damit meinen wir, daß für jedes von $\mathcal{A}$ akzeptierte ω-Wort ξ gilt $\xi \models \phi$. Durch die spezielle Wahl des Automatenvokabulars haben wir sichergestellt, daß ξ eine ω-Struktur beschreibt. Wir führen dafür die neue Notation ein

$$\mathcal{A} \models \phi \quad \text{gdw} \quad \text{für alle } \xi \in L^{\omega}(\mathcal{A}) \text{ gilt } \xi \models \phi$$

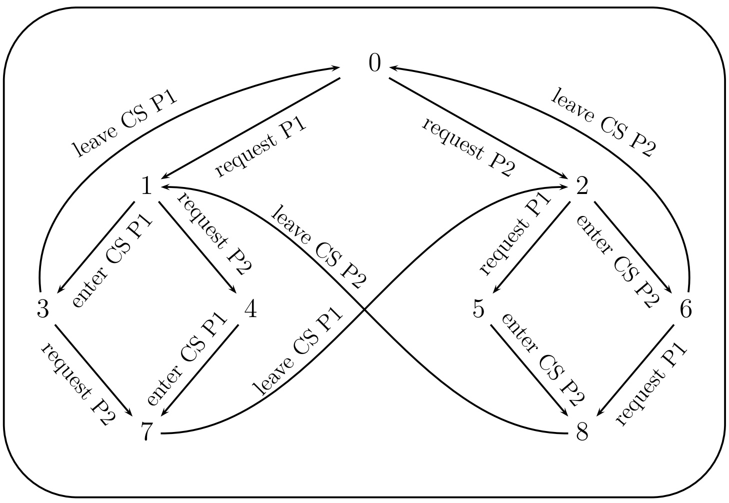


Abbildung 12.9: Ereignis-basiertes Automatenmodell

Als ein Beispiel für ein Modell betrachten wir Abbildung 12.9. Sie zeigt ein Modell zur Regelung des gemeinsamen Zugriffs mehrerer Prozesse, hier sind es zwei, auf eine gemeinsame Resource. Das damit gelöste Problem ist im Englischen unter der Bezeichnung *mutual exclusion problem* bekannt und dient oft als akademisches Beispiel. Der Automat in Abbildung 12.9 trägt als Kantenmarkierungen Ereignisse, was das intuitive Verständnis sicherlich erleichtert. Für die formale Analyse mit Hilfe von LTL müssen wir allerdings einen etwas anderen Blickwinkel einnehmen. Wir brauchen als erstes eine Menge Σ aussagenlogischer Atome. Im vorliegenden Fall entscheiden wir uns für die folgende Auswahl, wobei in unserem Beispiel $i \in \{1, 2\}$:

N_i	Prozeß i befindet sich in einer nichtkritischen Region
T_i	Prozeß i befindet sich in der Anmeldephase
C_i	Prozeß i befindet sich in einer kritischen Region

Die Kantenmarkierungen des Automaten aus Abbildung 12.9 müssen jetzt entsprechend angepasst werden, um den Zusammenhang zwischen dem aussagenlogischen Vokabular Σ und den Aktionen des Automaten herzustellen. Als Automatenvokabular V soll, wie schon gesagt, $V = 2^\Sigma$ benutzt werden. Der Zusammenhang zwischen den ereignisorientieren Labels und den Markierungen aus V erhält man dadurch, daß für eine Kante von s_1 nach s_2 die Markierung mit dem Ereignis e ersetzt wird mit der Teilmenge von Atomen, die in s_2 nach Ausführung der Aktion e wahr sind. Das ergibt die in Abbildung 12.10 gezeigten Kantenmarkierungen.

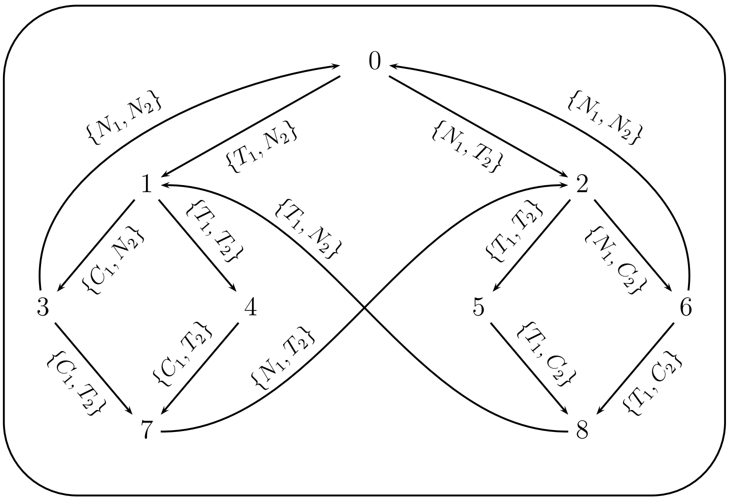


Abbildung 12.10: Aussagenbasiertes Automatenmodell

Wollte man das Beispiel mit dem vollständigen Automaten aus Abbildung 12.9 durchführen würde das Bild sehr schnell sehr undurchsichtig werden.

Wir beschränken uns daher auf den abgespeckten Automaten aus Abbildung 12.11. Das simplifiziert zwar die Aufgabenstellung bis zur Trivialität, aber der Grundgedanke der gesamten Vorgehensweise sollte so besser erkennbar sein.

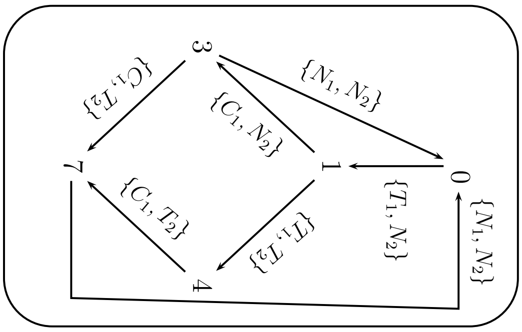


Abbildung 12.11: Reduziertes Automatenmodell $\mathcal{A}_{me}$

Nachdem das zu analysierende Automatenmodell feststeht, wenden wir uns den Eigenschaften zu, die wir gerne verifizieren möchten. Eine naheliegende Forderung wäre, daß jeder Prozeß, der sich für die Nutzung der exklusiven Resource anmeldet, schließlich auch den Zugang erhält. Das läßt sich als LTL-Formel über der Signatur Σ ausdrücken als

$$\mathcal{A}_{me} \models \Box(T_i \rightarrow \Diamond C_i) \quad ?$$

Wir erinnern noch einmal daran was $\mathcal{A}_{me} \models \Box(T_i \rightarrow \Diamond C_i)$ bedeutet. Für jede akzeptierte Berechnungfolge s gilt für die omega-Struktur ξ_s, die s zugeordnet ist $\xi_s \models \Box(T_i \rightarrow \Diamond C_i)$. In dem einfachen Beispielautomaten $\mathcal{A}_{me}$ ist die Menge der Finalzustände die Menge aller Zustände, also ist jede Folge von Zuständen, die den Kanten des Automaten folgt, eine akzeptierte Berechnungsfolge. Somit sind

$$\begin{array}{lcl} s_1 & = & 0\ 1\ 3\ 0\ 1\ \ldots \\ s_2 & = & 0\ 1\ 3\ 7\ 0\ \ldots \\ s_3 & = & 0\ 1\ 4\ 7\ 0\ \ldots \end{array}$$

Anfangsstücke akeptierender Berechungsfolgen. Bezeichnen wir mit ξ^i die omega-Struktur zu s_i dann sieht man leicht

$$\begin{array}{ll} \xi^1 \models T_1 & \xi^1_1 \models C_1 \\ \xi^2 \models T_1 & \xi^2_1 \models C_1 \\ \xi^2 \models T_1 & \xi^3_2 \models C_1 \end{array}$$

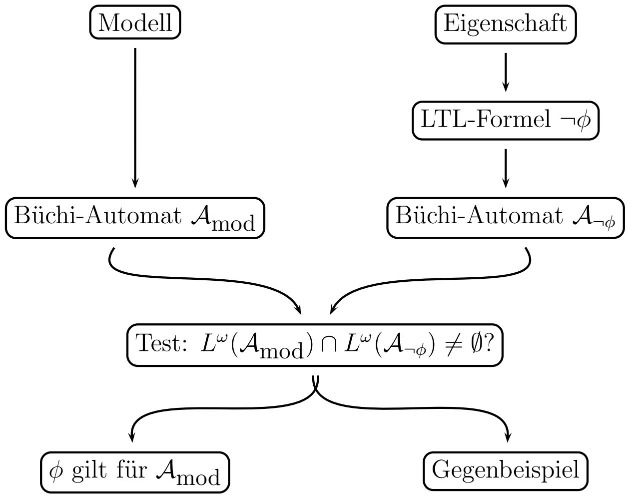


Abbildung 12.12: Übersicht über LTL Modellprüfung

12.4.2 Methode

Abbildung 12.12 zeigt schematisch den Ablauf eine Modellprüfung. Wir werden anhand des Beispiels aus Abschnitt 12.4.1 diesen Ablauf Schritt für Schritt nachvollziehen.

Wie in dem Übersichtsdiagramm in Abbildung 12.12 gezeigt, wird mit der Negation der zu verifizierenden Eigenschaft gearbeitet, d.h. in unserem Beispiel mit

$$\Diamond(T_i \wedge \Box\neg C_i)$$

Wir wollen uns hier auf die Aussage für den ersten Prozeß konzentrieren: $\Diamond(T_1 \wedge \Box\neg C_1)$.

Der nächste Schritt besteht darin, einen Büchi-Automaten $\mathcal{B}_{me}$ zu konstruieren, der im Sinne von Satz 12.3, die LTL-Formel $\Diamond(T_1 \wedge \Box\neg C_1)$ wiedergibt. Das ist nach allem, was wir bisher schon gesehen haben, einfach, siehe Abbildung 12.13.

Der Plan ist, zu zeigen, daß keine Berechnungsfolge des Automaten $\mathcal{A}_{me}$ eine omega-Struktur ergibt, die $\Diamond(T_1 \wedge \Box\neg C_1)$ erfüllt. Das läuft darauf hinaus, daß kein ω-Wort aus V^ω im Durchschnitt von $L^\omega(\mathcal{A}_{me})$ und $L^\omega(\mathcal{B}_{me})$ liegt. In dem vorliegenden Beispiel kann man den Produktautomat für diesen Test benutzen, weil $L^\omega(\mathcal{A}_{me} \times \mathcal{B}_{me}) = L^\omega(\mathcal{A}_{me}) \cap L^\omega(\mathcal{B}_{me})$ gilt. Im Allgemeinen muß der in Lemma 12.2 konstruierte Durchschnittsautomat $\mathcal{A}_{me} \cap \mathcal{B}_{me}$ benutzt werden. Im vorliegenden Fall ist insbesondere (s_1, t_1) ein Endzustand des Produktautomaten, wenn sowohl s_1 als auch t_1 in den Faktorautomaten

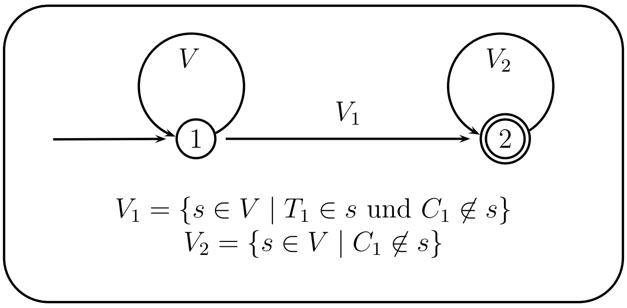


Abbildung 12.13: Büchi-Automat $\mathcal{B}_{me}$ zu $\Diamond(T_1 \wedge \Box\neg C_1)$

Endzustände sind. In dem Automaten $\mathcal{A}_{me}$ ist jeder Zustand ein Endzustand. Wir haben das nicht extra durch Doppelkreise angezeigt. Das erklärt den in Abbildung 12.14 gezeigten Produktautomaten.

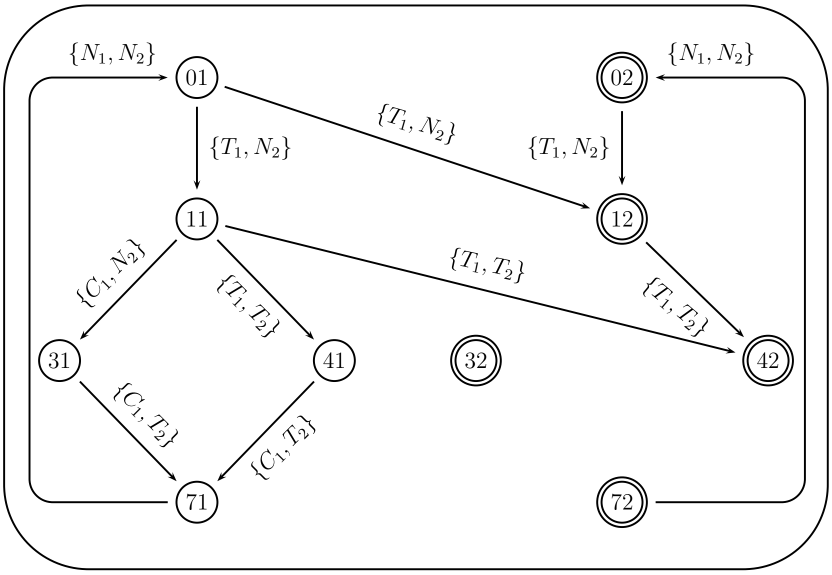


Abbildung 12.14: Produktautomat $\mathcal{A}_{me} \times \mathcal{B}_{me}$

Man sieht leicht, daß der Produktautomat keine Schleife enthält, die durch einen Endzustand geht, d.h. $L^\omega(\mathcal{A}_{me} \times \mathcal{B}_{me}) = \emptyset$

12.5 Aussagenlogische Modellprüfung (Zusatzstoff)

Dieses Kapitel stellt einen interessanten Zusammenhang her zwischen dem im Unterkapitel 12.4.1 vorgestellten Modellprüfungsproblem und aussagenlogischer Erfüllbarkeit, Definition 2.18. Es bildet die Grundlage für eine Alternative zur Modellprüfungsmethode aus Abschnitt 12.4.2, indem man Programme zur Lösung aussagenlogischer Erfüllbarkeitsprobleme, sogenannte SAT solver, einsetzen kann. Dieser Ansatz ist unter dem Namen *bounded model checking* bekannt geworden und wurde zuerst in der Arbeit [BCCZ99] vorgestellt.

Wir beginnen mit einer simplen notationellen Vorbereitung. Die Aussage, daß ein Büchi-Automat $\mathcal{A}$ eine LTL-Formel A erfüllt, $\mathcal{A} \models A$, bedeutet, daß für jede akzepierende Berechnungsfolge s von $\mathcal{A}$ die zugeordnete omega-Struktur ξ (dargestellt als ein Wort über einem geeigneten Alphabet) die Formel erfüllt, d.h. daß $\xi \models A$ gilt. Bisher hatten wir nur $\xi \models A$ definiert. Das war auch aus Gründen einer modularen Vorgehensweise nützlich so; man konnte die Theorie der Logik LTL entwicklen ohne sich festzulegen, woher die omega-Strukturen kommen. Beschäftigt man sich allerdings ausschließlich mit Modellprüfungsaufgaben, wie wir das in diesem Kapitel tun, so macht es mehr Sinn direkt zu definieren, wann eine akzepierende Berechnungsfolge $s = (s_i)_{i \geq 0}$ eine LTLT-Formel A erfüllt. Man beachte, daß $\xi(n)$ die Markierung der Kante von s_n nach s_{n+1} ist. Die Definition von $\xi \models A$, Def.9.4, läß sich jetzt trivialerweise umschreiben zu einer Definition von $s \models A$:

Definition 12.20 (Semantik von LTL für Berechungsfolgen)

$s \models p$	gdw	$p \in \xi(0)$ (p ein AL Atom)
$s \models op(A, B)$		für aussagenlogische Kombinationen $op(A, B)$ von A und B wie üblich
$s \models \Box A$	gdw	für alle $n \in \mathbb{N}$ gilt $s_n \models A$
$s \models \Diamond A$	gdw	es gibt ein $n \in \mathbb{N}$ mit $s_n \models A$
$s \models A \mathbf{U} B$	gdw	es gibt $n \in \mathbb{N}$ mit $s_n \models B$ und für alle m mit $0 \leq m < n$ gilt $s_m \models A$
$s \models X A$	gdw	$s_1 \models A$

wobei s_n die Restfolge $(s_m)_{m \geq n}$ ist.

Wir werden im weiteren Verlauf dieses Abschnitt die Gültigkeit einer LTL-Formel für eine Berechnungsfolge, d.h. eine Aussage der Form $s \models A$, durch aussagenlogische Formeln ausdrücken. Die erste Schwierigkeit dabei ist die

Unendlichkeit der Berechungsfolge s. Die Lösung kommt daher, daß wir beweisen können, daß es genügt, endliche, zyklischen Berechnungsfolgen zu betrachten.

Definition 12.21 (Zyklische Berechungsfolgen)

1. Eine endliche Berechnungsfolge $s_0, \ldots, s_k$ heißt *i-zyklisch*, falls $0 \leq i < k$ und $s_k = s_i$.

2. Eine endliche Berechnungsfolge $s_0, \ldots, s_k$ heißt *zyklisch*, falls sie *i-zyklisch* ist für ein i mit $0 \leq i < k$.

3. Eine i-zyklische Berechnungsfolge $s_0, \ldots, s_k$ heißt akzeptierend, wenn unter den Zuständen $s_i, \ldots, s_k$ mindestens ein Endzustand vorkommt.

4. Für eine i-zyklische Berechnungsfolge $s = s_0, \ldots, s_k$ und eine LTL Formel A schreiben wir $s \models A$ anstelle von $s_0, \ldots, s_{i-1}, (s_i, \ldots, s_{k-1})^\omega \models A$.

Eine i-zyklische Berechnungsfolge $s_0, \ldots, s_k$ enthält schon die gesamte Information über die unendliche, schließlich periodische Berechnungsfolge $s_0, \ldots, s_{i-1}, (s_i, \ldots, s_{k-1})^\omega$. Inbesondere ist $s_0, \ldots, s_{i-1}, (s_i, \ldots, s_{k-1})^\omega$ akzeptierend, wenn $s_0, \ldots, s_k$ akzeptierend ist.

Lemma 12.22

Sei $\mathcal{A}$ ein Büchi-Automat und A eine LTL-Formel.
Falls es eine akzeptierende Berechungsfolge t gibt mit $t \models A$, dann gibt es auch eine endliche zyklische akzeptierende Berechnungsfolge s_e mit $s_e \models A$.

Beweis Für einfache Formeln, etwa $A \equiv \Box p$ oder $A \equiv \Diamond p$, kann man leicht sehen, daß die Aussage des Lemmas richtig ist. Denn ist t eine akzeptierende Berechnungsfolge, dann muß ein Endzustand s_F unendlich oft vorkommen. Die endliche Teilfolge t_e von t bis zum zweiten Auftreten (also dem ersten wiederholten Auftreten) von s_F erfüllt auch $t_e \models \Box p$, wenn schon $t \models \Box p$ gegolten hat. Gilt $t \models \Diamond p$, dann sucht man das kleinste i mit $t_i \models p$. Liegt t_i vor dem ersten Auftreten von s_F, kann man wie gerade beschrieben verfahren, anderenfalls bricht man die Folge t mit dem ersten Vorkommen von s_F nach t_i ab.
Es ist allerdings schwierig zu sehen, wie man daraus ein allgemeines Verfahren erhalten kann oder ein induktives Argument darauf aufbauen kann. Deswegen benutzen wir den folgenden Trick.
Sei $\mathcal{B}_A$ der nach Satz 12.3 existierende Büchi-Automat, so daß $L^\omega(\mathcal{B}_A) = \{\xi \mid \xi \models A\}$. Sei weiterhin $\mathcal{C}$ der Automat $\mathcal{A} \cap \mathcal{B}_A$ aus Lemma 12.2 mit der Eigenschaft $L^\omega(\mathcal{C}) = L^\omega(\mathcal{A}) \cap L^\omega(\mathcal{B}_A)$. Nach Annahme des Lemmas gilt also

$L^\omega(\mathcal{C}) \neq \emptyset$. Es gibt also eine akzeptierende Berechungsfolge s' von $\mathcal{C}$, so daß für die omega-Struktur ξ', die s' zugeordnet ist, gilt $\xi' \models A$. Nach der Akzeptanzbedingung für Büchi-Automaten gibt es einen Endzustant s_F von $\mathcal{C}$, der unendlich oft in s' vorkommt. Sei s'_i das erste und s'_k das zweite Vorkommen von s_F in s' und s die Folge $s'_1, \ldots, s'_{i-1}(s'_i, \ldots, s'_{k-1})^\omega$. Dann liegt auch s in $L^\omega(\mathcal{C})$. Insbesondere gilt für die s zugeordnete omega-Struktur ξ wieder $\xi \models A$.
Jetzt müssen wir etwas genauer in die interne Struktur von $\mathcal{C}$ hineinschauen. Jeder Zustand s_j in der Folge s ist von der Form $s_j = (s^1_j, s^2_j, k_j)$ wobei s^1_j ein Zustand von $\mathcal{A}$ ist, s^2_j ein Zustand von $\mathcal{B}_A$ ist und $k_j \in \{1, 2\}$. Nach Konstruktion von $\mathcal{C}$ ist s^1_F ein Endzustand von $\mathcal{A}$ und damit die Projektion von s auf die erste Koordinate, $s^1 = s^1_1, \ldots, s^1_n, \ldots$ eine akzeptierende Berechnungsfolge von $\mathcal{A}$. Die Projektion s^2 von s auf die zweite Koordinate ist ebenfalls eine akzeptiende Berechungsfolge von $\mathcal{B}_A$. Als letztes bemerken wir, daß die omega-Struktur die s^1 und s^2 zugeordnet ist mit ξ übereinstimmt. Somit haben wir mit $s_e = s^1_1, \ldots s^1_k$ eine endliche zyklische Berechungsfolge gefunden, welche die Behauptung des Lemmas erfüllt.

■

Wir können jetzt mit der aussagenlogischen Kodierung beginnen.

Lemma 12.23

Zu einem gegebenen Büchi-Automaten $\mathcal{A}$ und eine LTL-Formel F gibt es für jedes $k \in \mathbb{N}$ eine aussagenlogische Formel M_k, so daß gilt

$$\begin{array}{lll} M_k \text{ ist erfüllbar} & \text{gdw} & \text{es gibt eine zyklische Berechnungsfolge } s \\ & & \text{der Länge } k \text{ mit } s \models F \end{array}$$

Falls erforderlich schreiben wir anstelle von M_k genauer $M_k(\mathcal{A}, F)$.

Beweis Wir geben explizit ein Konstruktionsverfahren für M_k an. Diese Konstruktion ist zwar aufwendig, aber leicht nachvollziehbar. Wir werden deswegen keinen zusätzlichen Beweis führen, daß die angeführte Konstruktion tatsächlich leistet, was sie soll. Wir werden zusätzlich die einzelnen Schritte anhand des Automaten $\mathcal{A}_{dbp}$ aus Abb.12.4(a) illustrieren.

Wir nehmen an, daß $\mathcal{A}$ n Zustände hat, auf die wir mit ihrer Nummer verweisen, also Zustand 1, 5 oder allgemein Zustand n. Als erste Vorbereitung brauchen wir eine Binärkodierung der Zustandsnummern. Die dafür nötigen Booleschen Variablen seien $c_1, \ldots, c_m$.

In dem Beispielautomaten $\mathcal{A}_{dbp}$ gibt es nur zwei Zustände, also reicht eine Boolesche Variable c aus. $I(c) = \mathbf{0}$ bezeichnet den Anfangszustand $I(c) = \mathbf{1}$ den eindeutigen Endzustand.

Außerdem kommen in der Aufgabenstellungen die aussagenlogischen Variablen aus der Formel A vor. Diese seien $\Sigma = \{p_1, \ldots, p_r\}$.

In unserem Beispiel sind p und q die einzigen aussagenlogischen Variablen.

Wir werden später noch einige weitere Hilfvariablen einführen, über die wir aber reden, wenn es soweit ist.

Da die Formel M_k über Folgen von Zuständen der Länge k reden soll, kommen alle Variablen in k Kopien vor, also c^i_j mit $1 \leq i \leq k$ und $1 \leq j \leq m$ und p^i_j mit $1 \leq i < k$ und $1 \leq j \leq r$.

In unserem Beispiel wollen wir mit $k = 3$ rechnen, also gibt es die aussagenlogischen Variablen c^1, c^2, c^3, p^1, p^2 und q^1, q^2.

Die Formel M_k wird in mehreren Teilen konstruiert

$$M_k \equiv \mathit{Init} \wedge \mathit{Trans} \wedge \bigvee_{1 \leq i < k} L_i$$

Ist I eine Interpretation der aussagenlogischen Variablen, dann fassen wir $I(c^i_1), \ldots, I(c^i_m)$ für jedes $1 \leq i \leq k$ als Binärkodierung der Zahl n_i auf. So erhalten wir eine Folge von Zuständen $\pi = n_1, \ldots, n_k$. Ebenso fassen wir $I(p^i_1), \ldots, I(p^i_r)$ als Kodierung einer Teilmenge b_i der Variablen $p_1, \ldots, p_r$ auf. Die b_i sind also Buchstaben aus dem Vokabular V der Kantenmarkierungen des Automaten $\mathcal{A}$. Insgesamt erhalten wir eine Wort $b_1, \ldots, b_{k-1}$ der Länge $k-1$. Die Formeln Init und Trans werden dafür sorgen, daß $n_1, \ldots, n_k$ eine Berechungsfolge von $\mathcal{A}$ ist mit der Folge $b_1, \ldots, b_{k-1}$ von Kantenmarkierungen.

Betrachten wir die Interpretation I_1, I_2 in unserem Beispiel

	c^1	c^2	c^3	p^1	p^2	q^1	q^2
I_1	0	1	1	1	1	1	0
I_1	0	1	1	1	0	0	1

I_1 und I_2 liefern die Zustandsfolge $\pi = 0, 1, 1$. I_1 liefert dazu die Buchstabenfolge $w_1 = \{p, q\}, \{p\}$ und I_2 die Folge $w_2 = \{p\}, \{q\}$. Man sieht, daß w_1 eine korrekte Kantenmarkierung für die Folge π ist, w_2 aber nicht.

Ist $d = d_1, \ldots, d_m$ die Binärkodenummer eines Zustands von $\mathcal{A}$ dann bezeichen wir mit $S^i_d = S^i_{d_1,\ldots,d_m}$ die aussagenlogische Formel $\bigwedge_{d_j=1} c^i_j \wedge \bigwedge_{d_j=0} \neg c^i_j$. Ist b ein Buchstabe des Kantenalphabets, also $b \subseteq \Sigma$, dann steht B^i_b für die Formel $\bigwedge_{p_j \in b} p^i_j \wedge \bigwedge_{p_j \notin b} \neg p^i_j$.

Ist $d_0 = d_1, \ldots, d_m$ die Binärkodenummer des Anfangszustands von $\mathcal{A}$ dann ist

$$\mathit{Init} \equiv S^1_{d_0}$$

Seien $(d_a^1, d_e^1, b^1), \ldots (d_a^K, d_e^K, b^K)$ alle Kanten des Automaten $\mathcal{A}$, aufgefasst als Übergang von dem Zustand mit der Binärkodierung d_a^j in den Zustand mit der Binärkodierung d_e^j und der Kantenmarkierung b^j für $1 \leq j \leq K$.

$$Trans \equiv \bigwedge_{1 \leq i < k} \bigvee_{1 \leq j \leq K} (S^i_{d_a^j} \wedge S^{i+1}_{d_e^j} \wedge B^i_{b^j})$$

Für unser Beispiel ergeben diese Definitionen $Init = \neg c^1$. Die Menge aller Kanten ist

$$\begin{array}{l}
(0, 0, \{\}), (0, 0, \{p\}), (0, 0, \{q\}), (0, 0, \{p, q\}) \\
(0, 1, \{p\}), (0, 1, \{p, q\}) \\
(1, 1, \{p\}), (1, 1, \{p, q\})
\end{array}$$

$$Trans = \begin{array}{l}
(\neg c^1 \wedge \neg c^2) \vee (\neg c^1 \wedge c^2 \wedge p^1) \vee (c^1 \wedge c^2 \wedge p^1) \\
\wedge \\
(\neg c^2 \wedge \neg c^3) \vee (\neg c^2 \wedge c^3 \wedge p^2) \vee (c^2 \wedge c^3 \wedge p^2)
\end{array}$$

Dabei haben wir schon einige Vereinfachungen vorgenommen und z.B. $(\neg c^1 \wedge \neg c^2 \wedge p^1 \wedge q^1) \vee (\neg c^1 \wedge \neg c^2 \wedge \neg p^1 \wedge q^1) \vee (\neg c^1 \wedge \neg c^2 \wedge p^1 \wedge \neg q^1) \vee (\neg c^1 \wedge \neg c^2 \wedge \neg p^1 \wedge \neg q^1)$ äquivalent ersetzt durch $(\neg c^1 \wedge \neg c^2)$. Als weitere Vereinfachung könnte man $(c^1 \wedge c^2 \wedge p^1)$ weglassen, weil das im Widerspruch zu $Init$ steht.

Die Formeln L_i werden dafür sorgen, daß die aus einer erfüllenden Interpretation I extrahierte endliche Berechnungsfolge π i-zyklisch und akzeptierend ist und, was am wichtigsten ist, die Formel F erfüllt:

$$L_i = Z_i \wedge Akz_i \wedge erfF_i$$

Die ersten beiden Bestandteile sind einfach

$$\begin{array}{lcl}
Z_i & \equiv & \wedge_{1 \leq j \leq m} c_j^k \leftrightarrow \wedge_{1 \leq j \leq m} c_j^i \\
Akz_i & \equiv & Fin_i \vee Fin_{i+1} \ldots \vee Fin_{k-1}
\end{array}$$

wobei $F_i \equiv S^i_{d_1^f} \vee \ldots \vee S^i_{d_R^f}$ wenn $d_1^f, \ldots, d_R^f$ die Binärkodes aller Finalzustände von $\mathcal{A}$ sind.

Als letztes bleibt $erfF_i$ zu erklären. Die Erfüllbarkeit dieser Formel soll sicherstellen, daß die durch die erfüllende Belegung I bestimmte i-zyklische Berechnungsfolge π die Formel F erfüllt. An dieser Stelle brauchen wir neue aussagenlogische Variable. Wir sind zwar nur an der Gültigkeit von F an der Position 1 von π interessiert, es wird sich aber durch die Definition der temporalen Operatoren nicht vermeiden lassen, auch die Gültigkeit an den anderen Positionen $1 \leq j \leq k$ zu betrachten. Außerdem hängt die Gültigkeit auch von dem zyklischen *Rücksprungspunkt* i ab. Also brauchen wir für jede

Teilformel C, für jedes i, $1 \leq i < k$ und jedes j, $1 \leq j \leq k$ eine neue aussagenlogische Variable, die wir mit $[C]_i^j$ bezeichnen. Die definierenden Formeln lassen sich dann wie in Abbildung 12.15 aufschreiben. Die Zeilen für $\Box C$ und $\Diamond C$ sind eigenlich überflüssig. Es hilft aber beim Verstehen, wenn man zuerst die Definition von $\Box C$ und $\Diamond C$ gesehen hat, bevor die komplizierteren Operatoren C_1 **U** C_2 und C_1 **V** C_2 drankommen. Es gibt keine allgemeine Regel für die Negation. Es wird vorausgesetzt, daß die Formel F in Negationsnormalform vorliegt und daß die in der Definition von $[C_1 \mathbf{V} C_2]_i^j$ auf der rechten Seite auftretende Formel $\neg C_2$ in Negationsnormalform transformiert wird. Zu den aus der Abbildung 12.15 sich ergebenden aussagenlogischen Äquivalenzformeln kommt schließlich noch hinzu

$$erfF_i \leftrightarrow [F]_i^1$$

$$\begin{array}{lcll}
[C]_i^j & \leftrightarrow & p_l^j \quad \text{falls } C = p_l \in \Sigma & \\
[C]_i^j & \leftrightarrow & \neg p_l^j \quad \text{falls } C = \neg p_l \text{ mit } p_l \in \Sigma & \\
[C_1 \wedge C_2]_i^j & \leftrightarrow & [C_1]_i^j \wedge [C_2]_i^j & \\
[C_1 \vee C_2]_i^j & \leftrightarrow & [C_1]_i^j \vee [C_2]_i^j & \\
[\Box C]_i^j & \leftrightarrow & \bigwedge_{j \leq l < k} [C]_i^l & \text{falls } j \leq i \\
[\Box C]_i^j & \leftrightarrow & \bigwedge_{j \leq l < k} [C]_i^l \wedge \bigwedge_{i \leq l < j} [C]_i^l & \text{falls } i < j \\
[\Diamond C]_i^j & \leftrightarrow & \bigvee_{j \leq l < k} [C]_i^l & \text{falls } j \leq i \\
[\Diamond C]_i^j & \leftrightarrow & \bigvee_{j \leq l < k} [C]_i^l \vee \bigvee_{i \leq l < j} [C]_i^l & \text{falls } i < j \\
[C_1 \mathbf{U} C_2]_i^j & \leftrightarrow & \bigvee_{j \leq l < k} ([C_2]_i^l \wedge \bigwedge_{j \leq n < l} [C_1]_i^n) & \text{falls } j \leq i \\
[C_1 \mathbf{U} C_2]_i^j & \leftrightarrow & \bigvee_{j \leq l < k} ([C_2]_i^l \wedge \bigwedge_{j \leq n < l} [C_1]_i^n) \vee & \\
 & & \bigvee_{i \leq l < j} ([C_2]_i^l \wedge \bigwedge_{j \leq n < k} [C_1]_i^n \wedge \bigwedge_{i \leq n < l} [C_1]_i^n) & \text{falls } i < j \\
[C_1 \mathbf{V} C_2]_i^j & \leftrightarrow & [C_2]_i^j \wedge \bigwedge_{i \leq l < k} ([\neg C_2]_i^l \rightarrow \bigvee_{i \leq n < l} [C_1]_i^n) & \text{falls } j \leq i \\
[C_1 \mathbf{V} C_2]_i^j & \leftrightarrow & [C_2]_i^j \wedge \bigwedge_{j \leq l < k} ([\neg C_2]_i^l \rightarrow \bigvee_{j \leq n < l} [C_1]_i^n) \wedge & \\
 & & \bigwedge_{i \leq l < j} ([\neg C_2]_i^l \rightarrow \bigvee_{i \leq n < l} [C_1]_i^n \vee \bigvee_{j \leq n < k} [C_1]_i^n) & \text{falls } i < j \\
[X\ C]_i^j & \leftrightarrow & [C]_i^{j+1} & \text{falls } j < (k-1) \\
[X\ C]_i^{k-1} & \leftrightarrow & [C]_i^i &
\end{array}$$

Abbildung 12.15: Zyklische Semantik für LTL Formeln

Wir wollen jetzt L_1 und L_2 für unser Beispiel vorrechnen. Wir beginnen mit $Z_1 \equiv c^1 \leftrightarrow c^3$ und $Z_2 \equiv c^2 \leftrightarrow c^3$. Da der einzige Finalzustand von $\mathcal{A}_{dbp}$ durch c gegeben ist erhalten wir $Akz_1 \equiv c^1 \vee c^2 \vee c^3$ und $Akz_2 \equiv c^2 \vee c^3$. Betrachten wir für unser Beispiel die Formel $F \equiv \Diamond\Box p$.

$$\begin{array}{lcl}
[F]_1^1 & \equiv & [\Box p]_1^1 \vee [\Box p]_1^2 \\
[\Box p]_1^1 & \equiv & [p]_1^1 \wedge [p]_1^2 \\
 & \equiv & p^1 \wedge p^2 \\
[\Box p]_1^2 & \equiv & [p]_1^2 \wedge [p]_1^1 \\
 & \equiv & p^2 \wedge p^1
\end{array}$$

Insgesamt also $[F]_1^1 \leftrightarrow p^1 \wedge p^2$

$$\begin{array}{lcl}
[F]_2^1 & \equiv & [\Box p]_2^1 \vee [\Box p]_2^2 \\
[\Box p]_2^1 & \equiv & [p]_2^1 \wedge [p]_2^2 \\
 & \equiv & p^1 \wedge p^2 \\
[\Box p]_2^2 & \equiv & [p]_2^2 \\
 & \equiv & p^2
\end{array}$$

Insgesamt also $[F]_2^1 \leftrightarrow p^2$.

Wir können jetzt die Berechnung der L_i für das Beispiel hinschreiben.

$$\begin{array}{lcl}
L_1 & \leftrightarrow & (c^1 \leftrightarrow c^3) \wedge (c^1 \vee c^2 \vee c^3) \wedge p^1 \wedge p^2 \\
 & \leftrightarrow & (c^1 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge p^1 \wedge p^2 \\
L_2 & \leftrightarrow & (c^2 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge p^2 \\
 & \leftrightarrow & c^2 \wedge c^3 \wedge p^2
\end{array}$$

Das Endergebnis der gesamten Konstruktion M_3 ist in Abbildung 12.16 zu sehen. Diese Formel läßt sich weiter vereinfachen, wenn man ausnutzt, daß $Init = \neg c^1$ gelten muß. Durch diese Vereinfachungen ergibt sich $c^2 \rightarrow p^1$, was dann zu weiteren Vereinfachungen benutzt wird. Schließlich erhält man $M_3 \equiv \neg c^1 \wedge c^2 \wedge c^3 \wedge p^1 \wedge p^2$. Offensichtlich ist diese Formel erfüllbar. Was nach ihrer Konstruktion besagt, daß es eine Berechnungsfolge in $\mathcal{A}_{dbp}$ gibt, für die $\Diamond\Box p$ wahr ist. Wir wissen sogar, daß für jede Berechnungsfolge von $\mathcal{A}_{dbp}$ diese Formel gilt.

Satz 12.24

Es gibt eine Berechnungsfolge eines Büchi-Automaten $\mathcal{A}$, welche die LTL-Formel F erfüllt genau dann, wenn es ein k gibt, so daß $M_k(\mathcal{A}, F)$ aussagenlogisch erfüllbar ist.

$$
\begin{array}{l}
\neg c^1 \wedge \\
(\neg c^1 \wedge \neg c^2) \vee (\neg c^1 \wedge c^2 \wedge p^1) \vee (c^1 \wedge c^2 \wedge p^1) \\
\wedge \\
(\neg c^2 \wedge \neg c^3) \vee (\neg c^2 \wedge c^3 \wedge p^2) \vee (c^2 \wedge c^3 \wedge p^2) \\
\wedge \quad (\\
((c^1 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge p^1 \wedge p^2) \\
\vee \\
(c^2 \wedge c^3 \wedge p^2) \\
\qquad)
\end{array}
$$

Abbildung 12.16: M_3 für den Beispielautmaten $\mathcal{A}_{dbp}$ und $F \equiv \Diamond\Box p$

Beweis Folgt aus den Lemmata 12.23 und 12.22.

∎

Das Verfahren der limitierten Modellprüfung (das ist unsere Übersetzung von *bounded model checking*) für einen Büchi Automaten $\mathcal{A}$ und eine LTL-Formel F funktioniert nun so, daß für ein zunächst kleines k die aussagenlogische Erfüllbarkeit der Menge $M_k(\mathcal{A}, F)$ analysiert wird. Dazu gibt es leistungsfähige Implementierungen von E ntscheidungsverfahren, sogenannte *SAT solver*. Wird eine Lösung gefunden, so haben wir das Ausgangsprobem gelöst. Wird dagegen festgestellt, daß $M_k(\mathcal{A}, F)$ unerfüllbar ist, kann man weiterfahren mit der Analyse der Erfüllbarkeit von $M_{k+1}(\mathcal{A}, F)$. In einer Situation, in der es keine Berechnungsfolge von $\mathcal{A}$ gibt, die F erfüllt, terminiert dieses Verfahren zunächst nicht. Man kann aber leicht sehen, daß aus dem Beweis von Lemma 12.22 eine obere Schranke für k abgeleitet werden kann. Die kleinste obere Schranke für die Lemma 12.22 gilt, nennt man die Vollständigkeitsschranke (*completeness threshhold*) des Problems. Ist für ein k das gleich oder grösser als die Vollständigkeitsschranke ist $M_k(\mathcal{A}, F)$ unerfüllbar, so wissen wir, daß es keine akzeptierte Berechnungsfolge von $\mathcal{A}$ gibt, die F erfüllt.

Das limitierte Modellprüfungsverfahren reduziert nicht die Komplexität des zu lösenden Problems, es beruht weiterhin auf einem NP-vollständigen Verfahren. Experimente haben jedoch gezeigt, daß die Stärken des automatentheoretischen Zugangs orthogonal liegen zu den Stärken des limitierten Modellprüfungsverfahrens, [BCC+03, CFF+01].

12.5.1 Übungsaufgaben

Übungsaufgabe 12.5.1

Berechnen Sie die Formeln L_1, L_2 und M_3 für das in der Konstruktion zu Lemma 12.23 benutze Beispiel für die Formel $F \equiv \Box\Diamond\neg p$.

12.6 Modellprüfung mit SPIN

`Diese Abschnitt ist noch in Bearbeitung.`

Das zentrale Konzept in den beiden vorangehenden Kapiteln 11 und 12 ist das Konzept eines Büchi-Automaten, siehe Definition 11.19. Für Beweise von allgemeinen Aussagen über Büchi-Automaten, z.B. Lemma 12.2 oder Satz 12.3, wurde jeweils auf diese Definition bezug genommen und dazu gibt es auch keine Alterantive. Dagegen wurden Beispiele für konkrete Automaten bisher ausschließlich in graphischer Form gegeben. Das ist natürlich nur für kleine Automaten möglich, für größere, wie sie in realistischen Anwendungen auftreten, ist das nicht praktikabel.

```
/* Peterson's solution to mutual exclusion - 1981 */
bool turn, flag[2];
byte ncrit;
active [2] proctype user()
{
        assert(_pid == 0 || _pid == 1);
again:  flag[_pid] = 1;
        turn = _pid;
        (flag[1 - _pid] == 0 || turn == 1 - _pid);

        ncrit++;
        assert(ncrit == 1);     /* critical section */
        ncrit--;

        flag[_pid] = 0;
        goto again
}
```

Abbildung 12.17: *Mutual exclusion* für zwei Prozesse

Für dieses Problem gibt es eine Vielzahl von Lösungen. Wir wollen in diesem Abschnitt die Lösung, die G. Holtzmann für das SPIN System, [Hol04], benutzt vorstellen. Der Benutzer von SPIN muss die Automatenmodelle, die mit dem System analysiert werden sollen, in irgendeiner Form eingeben. Dazu dient die *Promela* Sprache (**Pro**cess **Me**ta **La**nguage). In Abbildung 12.6 ist ein einfaches Beispiel einer Prozeßbeschreibung in Promela zu sehen. Dieses Beispiel ist in den Testbeispielen enthalten, die mit dem SPIN System ausgeliefert werden. Es behandelt eine Lösung des konkurrierenden Zugriffs auf eine kritische Ressource in dem einfachen Fall, daß es nur zwei Prozesse gibt.

Wir beginnen mit einer Erklärung von Promela, die es zumindest erlaubt den Code in Abb. 12.6 zu verstehen. Als erstes ist zu bemerken, daß Promela keine Programmiersprache ist, sondern eine Beschreibungssprache für Systeme mit nebenläufiger Systeme (engl. concurrent systems). Promela Programme beschreiben Zustandsübergangssysteme durch die Definition von Prozesstypen, *process types*, anzeigt durch das Schlüsselwort `proctype`. In dem vorliegenden Codestück wird der Prozesstyp `users()` in den Zeilen 4 - 17 deklariert. Mit der Deklaration eines Typs ist noch keine Aktion verbunden. Das geschieht erst durch die Ausführung des `run` Befehls. Damit überhaupt Aktionen ins Rollen kommen gibt es einen Prozess `init`, der bei der Initialisierung gestartet wird. In der Deklaration von `init` könnten dann z.B., `run users(); run users();` stehen, womit zwei Prozesse vom Prozesstyp `users()` erzeugt würden. In dem vorliegenden Codebeispiel kommt keine `init` Deklaration vor. Stattdessen wird hier von einer Abkürzung Gebrauch gemacht, indem gleich bei der Prozeßdeklaration in Zeile 4 durch den Präfix `active [2]` zwei Prozesse des deklarierten Typs erzeugt werden. Die Prozesse werden in der Reihenfolge ihrer Aktivierung durchnumeriert. Auf die Nummer eines Prozesses, seine *process id*, kann mit der lokalen Variablen `_pid` zugegriffen werden. In dem betrachteten Code kommt das häufig vor, genauer in den Zeilen 6,7,8,9 und 15.
Neben der Prozesstypdeklaration von `users()` enthält der Code noch in den Zeilen 2 und 3 die Deklaration der globalen Variablen `turn`, `flag[2]`, `ncrit`. Dabei ist `flag` ein Feld (*array*) der Länge 2. Die Indizierung beginnt mit 0. Globale Variable können von jedem Prozess gelesen und geschrieben werden. Die Anweisungen im Rumpf der Prozesstypdeklaration von `users()` sind für einen Leser mit etwas Programmiererfahrung leicht zu verstehen, bis auf die Zeilen 6,9 und 12. Die `assert` Anweisungen in den Zeilen 6 und 9 werten den in der Klammer angebenen Booleschen Ausdruck aus. Falls er wahr ist, geht die Ausführung mit der nächsten Zeile weiter, anderenfalls wird ein Fehler berichtet und die Simulation oder Verifikation wird abgebrochen. Anstelle von Booleschen Ausdrücken konnen auch ganzzahlige Ausdrücke verwendet werden, die als wahr interpretiert werden wenn ihr Wert > 0 ist. Anders verhält es sich mit dem Kommando in Zeile 9. Eigentlich steht in dieser Zeile ein Boolescher Ausdruck. Es gehört zu den Eigenarten von Promela, daß jeder Ausdruck auch anstelle einer Anweisung stehen kann. Ergibt die Auswertung den Wahrheitswert *wahr*, dann geht die Ausführung mit der nächsten Anweisung weiter. Anderenfalls wird die Ausführung des Prozesses *blockiert*. Der Prozess bleibt blockiert, bis durch die Aktionen anderer Prozesse der Boo-

lesche Ausdruck wahr wird. Die Einteilung in ausführbare (engl. executable oder enabled) und blockierte (engl. blocked) Befehle ist eines der wichtigsten Mittel, mit denen man in Promela asynchrone Kommunikation beschreiben kann (Ein zweites Mittel sind Kanäle (engl channels), die in unserem Beispiel nicht vorkommen.)

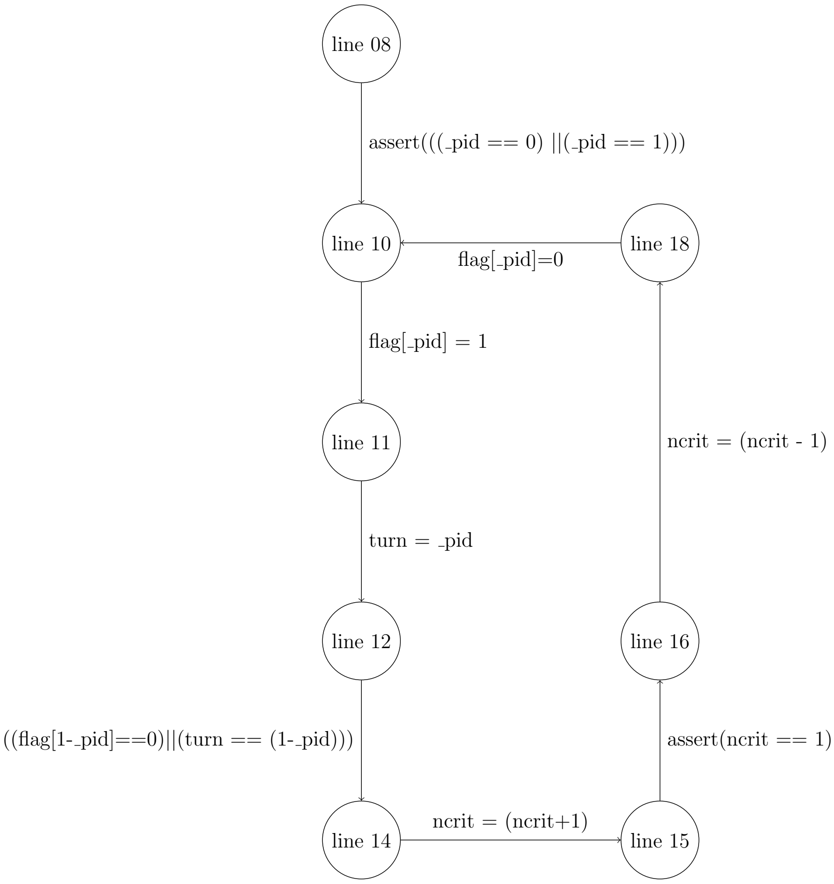
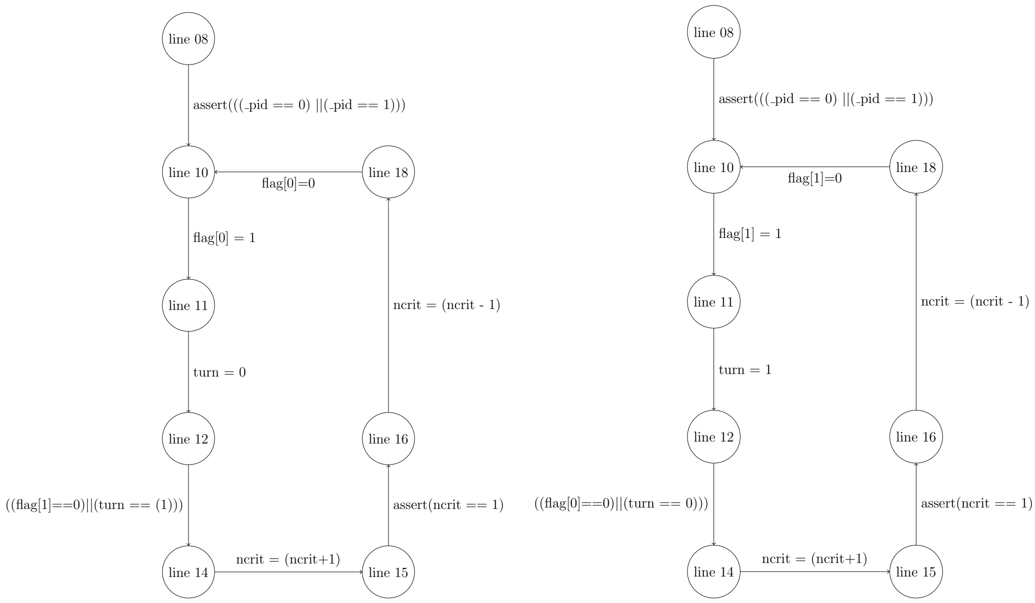


Abbildung 12.18: Generischer Automat zum *user* Prozess

SPIN bietet die Möglichkeit eine Prozesstypdeklaration zu visualisieren durch eine Art Automatenmodell. Die Visualisierung des `users()` Typs ist in Abb. 12.18 zu sehen. Man sieht, daß die Zustände durch die Nummern von Programmzeilen beschrieben werden. An den ausgehenden Kanten stehen die Anweisungen in der entsprechenden Zeile, die dann zum nächsten Zustand, d.h. zur dann erreichten Zeile führen. Diese Visualisierung vermittelt natürlich nur ein partielles Bild des Systems. Zum einen werden, der Übersichtlichkeit halber, die Werte der globalen Variablen nicht gezeigt, zum anderen fehlt die Darstellung des Zusammenwirkens der beiden Prozesse. In Abb. 12.19



(a) Prozess mit $pid = 0$ (b) Prozess mit $pid = 1$

Abbildung 12.19: Instanzen des generischen Automaten aus Abb. 12.18

sind, was man in SPIN nicht sieht, die Automaten für die beiden Prozessinstanzen vom Typ `users()` dargestellt, einmal für $\_pid = 0$ und einmal für $\_pid = 1$. Das Zusammenwirken der beiden Prozesse wird durch einen Automaten dargestellt, der ohne Berücksichtigung der globalen Variablen `ncrit` schon 512 Zustände umfassen würde. In Abb. 12.20 ist eine Approximation dieses Automaten zu sehen. Dabei wurde der durch *Zeile 08* gegeben Zustand weggelassen. Ein Zustand des approximativen Produktautomaten wird, untereinander geschrieben, gegeben durch die Programmzeilennummer von Prozess 0, Programmzeilennummer von Prozess 1 und den Wert der globalen Variablen `turn`. Man überlegt sich leicht, daß die Werte der beiden anderen globalen Variablen sich eindeutig aus den beiden Programmzeilen erschließen lassen. Für die Variable `turn` ist das nicht der Fall. Hier kommt z.B. sowohl $(10, 11, 0)$ als auch $(10, 11, 1)$ vor. Um die Abbildung 12.20 nicht durch zuviele Kanten unübersichtlich zu machen, wurden Kanten die vom unteren Ende des Graphen zurück zum oberen Ende führen durch korrespondierende Farbgebung kenntlich gemacht und die verbindende Kante unterdrückt. Es sei noch bemerkt, daß das SPIN System den Gesamtautomaten zu einer Promela Beschreibung nicht vorab berechnet. Er wird *on the fly* berechnet: immer wenn der Suchalgorithmus ausgehend von einem bestimmten Systemzustand die nächsten erreichbaren Zustände braucht, werden diese aus dem

Promela Code ermittelt.

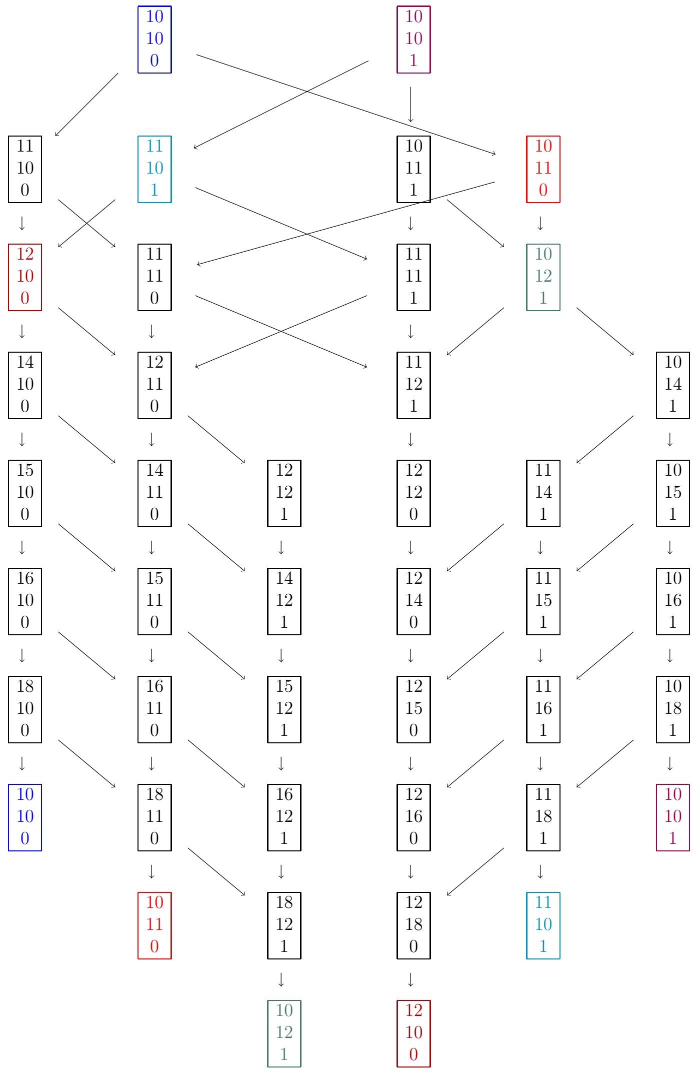


Abbildung 12.20: Ausschnitt des vollständigen Automaten zu Abb. 12.18

Kapitel 13

Lösungen

13.1 Lösungen zu Abschnitt 2.2 AL Syntax

Aufgabe 2.2.1 von Seite 20

(1) Die Menge der Terminalzeichen der Grammatik $\mathbf{G}_{AL}$ sei $\Sigma \cup \{\mathbf{1}, \mathbf{0}\} = \{P_0, \ldots P_n, \mathbf{1}, \mathbf{0}\}$, das einzige Nichtterminalzeichen sei F. Die Produktionsregeln von $\mathbf{G}_{AL}$ sind:

$$\begin{array}{lll} F & ::= & \mathbf{1} \mid \mathbf{0} \mid P_0 \mid \ldots \mid P_n \mid \\ & & (F \wedge F) \mid (F \vee F) \mid (F \rightarrow F) \mid (F \leftrightarrow F) \mid \\ & & \neg F \end{array}$$

(2) Die Menge der Terminalzeichen der Grammatik $\mathbf{G}_{AL+}$ sei $\{P, +\}$, die Nichtterminalzeichen $\{A, F\}$. Die Produktionsregeln von $\mathbf{G}_{AL+}$ sind:

$$\begin{array}{lll} A & ::= & P \mid A+ \\ F & ::= & \mathbf{1} \mid \mathbf{0} \mid A \mid \\ & & (F \wedge F) \mid (F \vee F) \mid (F \rightarrow F) \mid (F \leftrightarrow F) \mid \\ & & \neg F \end{array}$$

(3)

Aufgabe 2.2.2 von Seite 21

Der Beweis beider Behauptungen geschieht durch strukturelle Induktion nach dem Formelaufbau von A. Im einfachsten Fall ist $A = B$ und damit $A' = B'$. Aus der Voraussetzung $d(B') = d(B)$, bzw. $d(B') \leq d(B)$ folgt damit auch $d(A') = d(B') = d(A)$, bzw. $d(A') \leq d(A)$

Im Induktionsschritt sind ingesamt 6 Fälle zu unterscheiden:

1. $A = \neg A_0$ und B ist Teilformelvorkommen in A_0
2. $A = A_1 \wedge A_2$ und B ist Teilformelvorkommen in A_1
3. $A = A_1 \wedge A_2$ und B ist Teilformelvorkommen in A_2
4. $A = A_1 \vee A_2$ und B ist Teilformelvorkommen in A_1
5. $A = A_1 \vee A_2$ und B ist Teilformelvorkommen in A_2

Die Fälle (4), (5) sind vollkommen analog zu (2), (3) und (3) folgt wegen Symmetrie aus (2). Es genügt also die Fälle (1) und (2) zu betrachten.

Teil 1:
Fall (1) In diesem Fall gilt $A' = \neg A'_0$, wobei A'_0 aus A_0 hervorgeht durch Ersetzung des Vorkommens von B durch B'.

$$\begin{array}{rcll} d(A') & = & d(\neg A'_0) & \text{Form von } A' \\ & = & d(A'_0) + 1 & \text{Def. von } d \\ & = & d(A_0) + 1 & \text{Ind.voraussetzung} \\ & = & d(\neg A_0) & \text{Def. von } d \\ & = & d(A) & \text{Form von } A' \end{array}$$

Fall (2) In diesem Fall ist $A' = A'_1 \wedge A_2$, wobei A'_1 aus A_1 hervorgeht durch Ersetzung des Vorkommens von B durch B'.

$$\begin{array}{rcll} d(A') & = & d(A'_1 \wedge A_2) & \text{Form von } A' \\ & = & max\{d(A'_1), d(A_2)\} + 1 & \text{Def. von } d \\ & = & max\{d(A_1), d(A_2)\} + 1 & \text{Ind.voraussetzung} \\ & = & d(A_1 \wedge A_2) & \text{Def. von } d \\ & = & d(A) & \text{Form von } A' \end{array}$$

Teil 2:
Fall (1) In diesem Fall gilt $A' = \neg A'_0$, wobei A'_0 aus A_0 hervorgeht durch Ersetzung des Vorkommens von B durch B'.

$$\begin{array}{rcll} d(A') & = & d(\neg A'_0) & \text{Form von } A' \\ & = & d(A'_0) + 1 & \text{Def. von } d \\ & \leq & d(A_0) + 1 & \text{Ind.voraussetzung} \\ & = & d(\neg A_0) & \text{Def. von } d \\ & = & d(A) & \text{Form von } A' \end{array}$$

Fall (2) In diesem Fall ist $A' = A'_1 \wedge A_2$, wobei A'_1 aus A_1 hervorgeht durch Ersetzung des Vorkommens von B durch B'.

$$\begin{array}{rcll} d(A') & = & d(A'_1 \wedge A_2) & \text{Form von } A' \\ & = & max\{d(A'_1), d(A_2)\} + 1 & \text{Def. von } d \\ & \leq & max\{d(A_1), d(A_2)\} + 1 & \text{Ind.voraussetzung} \\ & = & d(A_1 \wedge A_2) & \text{Def. von } d \\ & = & d(A) & \text{Form von } A' \end{array}$$

Aufgabe 2.2.3 von Seite 21

Der Beweis der Gleichung $L(A) = Op(A) + 1$ geschieht durch strukturelle

Induktion über A. Ist A ein Atom so gilt $L(A) = 1$ und $Op(A) = 0$ und die Gleichung stimmt. Für $A = A_1 \wedge A_2$ oder $A = A_1 \vee A_2$ erhalten wir

$$\begin{array}{rcll}
L(A) & = & L(A_1) + L(A_2) & \text{Vorkommen in } A_1 \text{ und } A_1 \\
 & & & \text{sind disjunkt} \\
 & = & Op(A_1) + Op(A_2) + 2 & \text{Ind.Voraussetzung} \\
 & = & Op(A) + 1 & \text{wegen } Op(A) = Op(A_1) + Op(A_2) + 1
\end{array}$$

13.2 Lösungen zu Abschnitt 2.3 AL Semantik

Aufgabe 2.3.1 von Seite 36

zu 1

$A \wedge B$	wird ersetzt durch	$\neg(\neg A \vee \neg B)$
$A \rightarrow B$	wird ersetzt durch	$\neg A \vee B$
$A \leftrightarrow B$	wird ersetzt durch	$(A \rightarrow B) \wedge (B \rightarrow A)$

zu 2

$A \vee B$	wird ersetzt durch	$\neg A \rightarrow B)$

Damit ist Teil 2 auf Teil 1 zurückgeführt.
zu 3 Die funktionale Vollständigkeit der einelementigen Operatorenmenge $\{|\}$ ersieht man aus der folgenden Übersetzungtabelle:

$\neg A$	wird ersetzt durch	$A\|A$
$A \vee B$	wird ersetzt durch	$\neg A\|\neg B$
	also durch	$(A\|A)\|(B\|B)$

Der Rest folgt, da wir schon wissen, daß $\{\neg, \vee\}$ eine Basis ist.
zu 4 Die Übersetzungtabelle ist kurz:

$\neg A$	wird ersetzt durch	$1 \oplus A$

Da $\{\neg, \wedge\}$ eine Basis ist, ist auch $\{\mathbf{1}, \oplus, \wedge\}$ eine.

Aufgabe 2.3.2 von Seite 37

Reine Fleißarbeit.

Aufgabe 2.3.3 von Seite 37

Nach Vorausetzung wissen wir für jede Interpretation I

$$val_I(A) = val_I(A') \text{ und } val_I(B) = val_I B'$$

Für die Beweise machen wir von Lemma 2.22(4) Gebrauch.

(1)

$$\begin{array}{rll} val_I(\neg A) = \mathbf{W} & \text{gdw} & val_I(A) \neq \mathbf{W} \\ & \text{gdw} & val_I(A') \neq \mathbf{W} \\ & \text{gdw} & val_I(\neg A') = \mathbf{W} \end{array}$$

(2)

$$\begin{array}{rll} val_I(A \wedge B) = \mathbf{W} & \text{gdw} & val_I(A) = \mathbf{W} \text{ and } val_I(B) = \mathbf{W} \\ & \text{gdw} & val_I(A') = \mathbf{W} \text{ and } val_I(B') = \mathbf{W} \\ & \text{gdw} & val_I(A' \wedge B') = \mathbf{W} \end{array}$$

(3)

$$\begin{array}{lll} val_I(A \vee B) = \mathbf{W} & \text{gdw} & val_I(A) = \mathbf{W} \text{ or } val_I(B) = \mathbf{W} \\ & \text{gdw} & val_I(A') = \mathbf{W} \text{ or } val_I(B') = \mathbf{W} \\ & \text{gdw} & val_I(A' \vee B') = \mathbf{W} \end{array}$$

Aufgabe 2.3.4 von Seite 37

(1) A ist logisch äquivalent zu **1** falls für alle Interpretationen I gilt $val_I(A) = val_I(\mathbf{1}) = \mathbf{W}$.
Das stimmt überein mit der Definition der Allgemeingültigkeit von A.
(2) A ist logisch äquivalent zu **0** falls für alle Interpretationen I gilt $val_I(A) = val_I(\mathbf{0}) = \mathbf{F}$.
Das stimmt überein mit der Definition der Unfüllbarkeit von A.

Aufgabe 2.3.5 von Seite 37

Die wechselseitige Inkonsistenz zweier ausssagenlogischer Formeln A, B ist gleichbedeutend mit $\models \neg(A \wedge B$, was nach elementaren Umformungen gleichbedeutend ist mit $\models A \rightarrow \neg B$. Wendet man auf diese Implikation das Craigsche Interpolationslemma 2.26 an, so erhält man eine Formel C im gemeinsamen Vokabular von A und B mit $\begin{array}{l} \models A \rightarrow C \\ \models C \rightarrow \neg B \end{array}$

Da $\models C \rightarrow \neg B$ äquivalent zu $\models \neg(A \wedge B)$ sind wir fertig.

Aufgabe 2.3.6 von Seite 37

Da die C_i Interpolanten von $A \rightarrow B$ sind, sind definitionsgemäß auch die folgenden vier Formeln Tautologien:

$$\begin{array}{ccc} A \rightarrow C_1 & \text{und} & C_1 \rightarrow B \\ A \rightarrow C_2 & \text{und} & C_2 \rightarrow B \end{array}$$

Mit reiner Aussagenlogik folgt daß auch

$$A \rightarrow C_1 \wedge C_2 \text{ und } C_1 \vee C_2 \rightarrow B$$

Tautologien sind.
Da trivialerweise $(C_1 \wedge C_2) \rightarrow (C_1 \vee C_2)$ eine Tautologie ist, folgt

$$\begin{array}{ll} \text{aus } A \rightarrow C_1 \wedge C_2 & \text{auch } A \rightarrow C_1 \vee C_2 \\ \text{aus } C_1 \vee C_2 \rightarrow B & \text{auch } C_1 \wedge C_2 \rightarrow B \end{array}$$

Da offensichtlich in $C_1 \vee C_2$ und $C_1 \wedge C_2$ nur aussagenlogische Variable vorkommen, die in A und B vorkommen, sind $C_1 \vee C_2$ und $C_1 \wedge C_2$ Interpolanten von $A \to B$.

Aufgabe 2.3.7 von Seite 37

Offensichtlich kommt jede aussagenlogische Variable in D sowohl in A als auch in B vor.

Wir betrachten eine Interpretation I mit $val_I(A) = W$. Wir zielen darauf $val_I(D) = W$ zu zeigen. Seien, wie gesagt, $Q_1, \ldots, Q_k$ alle Variablen in B, die nicht in A vorkommen. Weiterhin seien $c_1, \ldots, c_k$ Konstanten aus $\{\mathbf{1}, \mathbf{0}\}$ und J die Belegung mit

$$J(Q) = \begin{cases} val_I(c_i) & \text{falls } Q = Q_i \text{ für } 1 \leq i \leq k \\ I(Q) & \text{sonst} \end{cases}$$

Es gilt weiterhin $val_J(A) = W$, da ja nur die Belegung von Aussagenvariablen geändert wurden, die nicht in A vorkommen. Da $A \to B$ eine Tautologie ist gilt auch $val_J(B) = W$. Das läßt sich äquivalent auch schreiben als $val_I(B[c_1, \ldots, c_k]) = W$. Wiederholt man dieses Argument für jedes k-Tupel $(c_1, \ldots, c_k) \in \{\mathbf{1}, \mathbf{0}\}^k$ dann erhält man.

$$val_I(\bigwedge_{(c_1,\ldots,c_n)\in\{\mathbf{1},\mathbf{0}\}^n} B[c_1, \ldots, c_n]) = val_I(D) = W$$

Damit ist schon einmal $val_I(A \to D) = W$ gezeigt für beliebiges I.

Gelte jetzt $val_I(D) = W$. Dann gilt für insbesondere für $c_i = \mathbf{1} \Leftrightarrow I(Q_i) = W$, $1 \leq i \leq k$, auch $val_I(B[c_1, \ldots, c_n]) = W$, denn $B[c_1, \ldots, c_n]$ ist ja konjunktiver Bestandteil von D. In diesem Fall gilt aber auch

$$val_I(B) = val_I(B[c_1, \ldots, c_n]) = W.$$

Damit ist auch die Allgemeingültigkeit von $D \to B$ gezeigt.

Aufgabe 2.3.8 von Seite 38

Seien A, B, C, D, U die in der Aufgabenstellung bezeichneten Formeln mit den dort vorausgesetzten Eigenschaften.

Wir zeigen zuerst, daß $C \to U$ eine Tautologie ist. Sei also I eine Interpretation mit $val_I(C) = W$. Nach Definition von C gibt es $c_i \in \{\mathbf{0}, \mathbf{1}\}$, $1 \leq i \leq n$ mit $val_I(A[c_1, \ldots, c_n]) = w$. Wir definieren, wie schon in vorangegangenen Beweisen,

$$J(P) = \begin{cases} c_i & \text{falls } P = P_i \text{ für } 1 \leq i \leq n \\ I(P) & \text{sonst} \end{cases}$$

Offensichtlich gilt $val_J(A) = W$. Wegen der Allgemeingültigkeit von $A \rightarrow U$ gilt auch $val_J(U) = W$. Da U eine Interpolante ist, kommen die aussagenlogischen Variablen $P_1, \ldots, P_n$ in U nicht vor, somit gilt auch $val_I(U) = W$.

Der Beweis des zweiten Teils is analog zum ersten. Wir führen ihn dennoch voll aus. Zu zeigen ist hier die Tautologieeigenschaft von $U \rightarrow D$. Sei also I eine beliebige Interpretation mit $val_I(U) = W$. Sei $c_1, \ldots, c_k$ ein beliebiges k-Tupel von Elementen aus $\{\mathbf{0}, \mathbf{1}\}$. Wir setzen:

$$J(Q) = \begin{cases} c_i & \text{falls } Q = Q_i \text{ für } 1 \leq i \leq k \\ I(Q) & \text{sonst} \end{cases}$$

Da die Q_i nicht in U vorkommen gilt auch $val_J(U) = W$. Aus $U \rightarrow B$ folgt also auch $val_J(B) = W$, was gleichbedeutend ist mit $val_I(B[c_1, \ldots, c_k]) = W$. Da die Wahl der c_i beliebig war haben wir $val_I(D) = W$ gezeigt. Da außerdem I beliebig war haben wir gezeigt, daß $U \rightarrow D$ allgemeingültig ist.

Aufgabe 2.3.9

Für jedes Land i gibt es drei AL-Variablen R_i, G_i, B_i. Nun muss für einen gegebenen Graphen formalisiert werden, dass

1. Jedes Land mindestens 1 Farbe hat,
2. jedes Land höchstens 1 Farbe hat und
3. benachbarte Länder nicht dieselbe Farbe haben.

Schreibweise. Im folgenden schreiben wir

$$\bigwedge_{i:b(i)} \phi(i)$$

als Abkürzung für eine endliche Konjunktion der aussagenlogischen Formeln $\phi(i)$, für die gilt, dass die Bedingung $b(i)$ ist erfüllt. Dabei ist i eine Variable auf der Metaebene, und b ist eine Formel der Metaebene. Abkürzungen wie $\bigvee_{i,j:b(i,j)} \phi(i,j)$ mit mehreren Variablen sind analog definiert.

Formalisierung.

$$\begin{aligned} F = & \bigwedge_{i:0\le i<L} (R_i \vee G_i \vee B_i) \\ \wedge & \bigwedge_{i:0\le i<L} \neg\big((R_i \wedge G_i) \vee (G_i \wedge B_i) \vee (B_i \wedge R_i)\big) \\ \wedge & \bigwedge_{i,j:0\le i,j<L, Na(i,j)} \big(\neg(R_i \wedge R_j) \wedge \neg(G_i \wedge G_j) \wedge \neg(B_i \wedge B_j)\big) \end{aligned}$$

Hinweis. Lässt man die zweite Bedingung weg, so erhält man eine bezüglich Erfüllbarkeit äquivalente Formel.

Ein konkretes Beispiel Wir geben die Formel explizit an, für $L = 5$ und die Nachbarschaftsbeziehung
$Na = \{(0,1),(0,2),(0,3),(0,4),(1,2),(2,3),(3,4),(4,1)\}$

$$\begin{aligned} F \;=\; & (R_0 \vee G_0 \vee B_0) \wedge (R_1 \vee G_1 \vee B_1) \wedge (R_2 \vee G_2 \vee B_2) \wedge \\ & (R_3 \vee G_3 \vee B_3) \wedge (R_4 \vee G_4 \vee B_4) \wedge \\ & \neg(R_0 \wedge G_0) \wedge \neg(R_0 \wedge B_0) \wedge \neg(G_0 \wedge B_0) \wedge \\ & \neg(R_1 \wedge G_1) \wedge \neg(R_1 \wedge B_1) \wedge \neg(G_1 \wedge B_1) \wedge \\ & \neg(R_2 \wedge G_2) \wedge \neg(R_2 \wedge B_2) \wedge \neg(G_2 \wedge B_2) \wedge \\ & \neg(R_3 \wedge G_3) \wedge \neg(R_3 \wedge B_3) \wedge \neg(G_3 \wedge B_3) \wedge \\ & \neg(R_4 \wedge G_4) \wedge \neg(R_4 \wedge B_4) \wedge \neg(G_4 \wedge B_4) \wedge \\ & \neg(R_0 \wedge R_1) \wedge \neg(G_0 \wedge G_1) \wedge \neg(B_0 \wedge B_1) \wedge \\ & \neg(R_0 \wedge R_2) \wedge \neg(G_0 \wedge G_2) \wedge \neg(B_0 \wedge B_2) \wedge \\ & \neg(R_0 \wedge R_3) \wedge \neg(G_0 \wedge G_3) \wedge \neg(B_0 \wedge B_3) \wedge \\ & \neg(R_0 \wedge R_4) \wedge \neg(G_0 \wedge G_4) \wedge \neg(B_0 \wedge B_4) \wedge \\ & \neg(R_1 \wedge R_2) \wedge \neg(G_1 \wedge G_2) \wedge \neg(B_1 \wedge B_2) \wedge \\ & \neg(R_2 \wedge R_3) \wedge \neg(G_2 \wedge G_3) \wedge \neg(B_2 \wedge B_3) \wedge \\ & \neg(R_3 \wedge R_4) \wedge \neg(G_3 \wedge G_4) \wedge \neg(B_3 \wedge B_4) \wedge \\ & \neg(R_4 \wedge R_1) \wedge \neg(G_4 \wedge G_1) \wedge \neg(B_4 \wedge B_1) \end{aligned}$$

Aufgabe 2.3.10

(1) Ist C eine tautologische Klausel in S, dann ist $S \setminus \{C\}$ erfüllbar, wegen der minimalen Inkonsistenz von S. Es gibt also I mit $I(D) = W$ für alle $D \in S$, $D \neq C$. Da C aber tautologisch ist gilt auch $I(C) = W$, was im Widerspruch zu angekommenen Unerfüllbarkeit von S steht.

(2) Da S unerfüllbar ist gibt es auf jeden Fall eine Klausel $D \in S$ mit $I(D) = F$. Wäre D eine isolierte Klausel, so wäre nach Aufgabe 2.4.17 $S \setminus \{D\}$

unerfüllbar, im Widerspruch zur minimalen Unerfüllbarkeit von S. Also gibt es sogar für jedes Literal L in D eine Klausel C, in der $\bar{L}$ vorkommt. Da für alle Literale L in D wegen $I(D) = F$ auch $I(L) = F$ gelten muß gilt $I(\bar{L}) = W$ und damit auch $I(C) = W$.

Antworten auf die Fragen zum Beweis von Lemma 2.26

1. Weil alle aussagenlogischen Variablen, die in A aber nicht in B vorkommen durch Konstanten **1** oder **0** ersetzt wurden.
2. Weil C eine Disjunktion von Formeln der Form $A[c_1, \ldots, c_n]$ ist.
3. Man muß $c_1 = 1$ falls $\mathbf{W} = I(P_1)$ und $c_1 = 0$ falls $\mathbf{F} = I(P_1)$ wählen. Vergleicht man die rekursiven Berechnungen von $val_I(A)$ und $val_I(A[c_1])$, so stellt man fest, daß an den Stellen, an denen im ersten Fall $I(P_1)$ gebraucht wird, im zweiten Fall $I(c_1)$ gebraucht wird. Wer ganz penibel sein will, kann einen Beweis über die Komplexität von A führen. Es ist aber verzeihlich im diesem Fall sich allein auf die Intuition und zwei, drei kleine Beispiele zu verlassen.
4. Das ist eine einfache Erweiterung der vorangegangenen Frage. Man muß $c_i = I(P_i)$ für alle $1 \leq i \leq n$ wählen.
5. Um $\models \mathbf{A} \rightarrow \mathbf{C}$ zu zeigen, muß man für jede Interpretation I mit $val_I(A) = \mathbf{W}$ zeigen, daß auch $val_I(A[c_1, \ldots, c_n]) = \mathbf{W}$ gilt für mindestens eine Wahl von $c_1, \ldots, c_n$.
6. Nein. Ein einfaches Gegenbeispiel wäre $A = P_1$, $c_1 = \mathbf{0}$ und $I(P_1) = \mathbf{W}$.
7. $I_1(P) = \begin{cases} \mathbf{W} & \text{falls } P = P_1 \text{ und } c_1 = \mathbf{1} \\ \mathbf{F} & \text{falls } P = P_1 \text{ und } c_1 = \mathbf{0} \\ I(P) & \text{sonst} \end{cases}$
8. Das ist jetzt einfach.
   $I_1(P) = \begin{cases} \mathbf{W} & \text{falls } P = P_i \text{ für ein } 1 \leq i \leq n \text{ und } c_i = \mathbf{1} \\ \mathbf{F} & \text{falls } P = P_i \text{ für ein } 1 \leq i \leq n \text{ und } c_i = \mathbf{0} \\ I(P) & \text{sonst} \end{cases}$
9. Es gilt $val_I(B) = val_J(B) = \mathbf{W}$.
10. Die Aufgabe ist für jede Interpretation I mit $val_I(C) = \mathbf{W}$ zu zeigen, daß auch $val_I(B) = \mathbf{W}$ gilt. Nach Definition von C als Disjunktion folgt aus $val_I(C) = \mathbf{W}$, daß $val_I(A[c_1, \ldots, c_n]) = \mathbf{W}$ für mindestens eine Wahl von $c_1, \ldots, c_n$ gelten muß. Wie in Antwort 8 finden wir eine

Interpretation J mit $val_J(A) = \mathbf{W}$. Da nach Voraussetzung $A \rightarrow B$ eine Tautologie ist, muß auch $val_J(B) = \mathbf{W}$ gelten. Da sich aber I und J nur für aussagenlogische Variablen P_i unterscheiden, die nicht in B vorkommen, kann man mit der Antwort 9 auf $val_I(B) = \mathbf{W}$ schließen.

13.3 Lösungen zu Abschnitt 2.4 AL Normalformen

Aufgabe 2.4.1

Wir beweisen die Terminierung der Schleife durch die Angabe einer *Variante*. Das ist die Standardvorgehensweise. Eine Schleifenvariante sv ist eine ganze Zahl, für die gilt

1. $vs \geq 0$ gilt vor jedem Schleifendurchlauf (auch vor dem ersten)
2. ist vs der Wert vor Beginn und vs' der Wert nach Ende eines Schleifendurchlaufs, dann gilt $vs > vs'$.

Können wir eine Variante finden, so terminiert die Schleife offensichtlich. Wir schlagen die folgende Variante vor

$$vs = \Sigma_{\neg B \in N(A)} 3^{d(B)}$$

Als Summe positive Zahl ist vs immer ≥ 0. Seien $N(A')$, vs' die Werte von $N(A)$ und vs nach dem Schleifendurchlauf. Um die Änderung von vs zu vs' zu analysieren, brauchen wir die Fallunterscheidung:
Fall $\mathbf{B = \neg B_0}$
$N(A) = N_{00}(A) \cup \{\neg\neg B_0, \neg B_0\} \cup N_{01}(A) \cup N_1(A)$ mit

- $N_{00}(A) = \{C \in N(A) \mid \neg B$ ist echtes Teilformelvorkommen von $C\}$
- $N_{01}(A) = \{C \in N(A) \mid C$ ist Teilformelvorkommen von $B_0\}$
- $N_2(A) = \{C \in N(A) \mid \neg B$ ist disjunkt zu $C\}$

Beim Übergang von A zu A' fallen die Teilvorkommen $\neg\neg B_0$, $\neg B_0$ weg, die Teilformeln in $N_{01}(A)$ und $N_2(A)$ werden durch die Ersetzung nicht verändert, die Formeln $C \in N_{00}(A)$ werden ersetzt durch C', wobei C' aus C hervorgeht, indem $\neg\neg B_0$ ersetzt wird durch B_0. Also

$$N(A') = \{C' \mid C \in N_{00}(A)\} \cup N_{01}(A) \cup N_1(A)$$

Da $d(B_0) < d(\neg\neg B_0)$ gilt erhalten wir nach Übungsaufgabe 2.2.2(2) für alle $C \in N_{00}(A)$ die Ungleichung $d(C') \leq d(C)$. Insgesamt führt das zu $vs' < vs$. Da auf jeden fall die echt positiven Summanden $3^{d(B)}$ und $3^{d(B_0)}$ aus der Summe wegfallen.
Fall $\mathbf{B = B_1 \wedge B_2}$
Wir nehmen eine ähnliche Aufteil vor, wie im vorangegangen Beweisteil. $N(A) = N_{00}(A) \cup \{\neg(B_1 \wedge B_2)\} \cup N_{01}(A) \cup N_1(A)$ mit

- $N_{00}(A) = \{C \in N(A) \mid \neg B$ ist echtes Teilformelvorkommen von $C\}$
- $N_{01}(A) = \{C \in N(A) \mid C$ ist Teilformelvorkommen von B_1 oder $B_2\}$
- $N_2(A) = \{C \in N(A) \mid \neg B$ ist disjunkt zu $C\}$

Somit

$$N(A') = \{C' \mid C \in N_{00}(A)\} \cup \{\neg B_1, \neg B_2\} \cup N_{01}(A) \cup N_1(A)$$

Da $d(\neg B) = d(\neg B_1 \vee \neg B_2)$ gilt wegen 2.2.2(1) für alle $C \in N_{00}$ die Gleichung $d(C) = d(C')$. Beim Übergang von vs nach vs' wird also lediglich der Summand $3^{d(B)}$ ersetzt durch $3^{d(B_1)} + 3^{d(B_2)}$. Mit Hilfe der Definition der Schachtelungstiefe d erhalten wir $3^{d(B_1)} + 3^{d(B_2)} \leq 2 * 3^{max\{d(B_1),d(B_2\}} < 3^{max\{d(B_1),d(B_2\}+1} = 3^{d(B)}$.
Man sieht hier, daß die Definition von vs mit 2 anstelle von 3 nicht funktionieren würde
Fall $\mathbf{B = B_1 \vee B_2}$ Vollkommen analog zum vorangegangen Fall. 2.2.2

Aufgabe 2.4.2

Wir betrachten in dieser Lösung, wie schon im ganzen Unterkapitel 2.4.1 nur die aussagenlogischen Operatoren in $Op_{nf} = \{\neg, \wedge, \vee\}$.

Wie in der Lösung zu Aufgabe 2.4.1 geben wir eine Schleifenvariante an, die diesmal ds heißen soll.

Als Vorbereitung darauf brauchen wir die k-Tiefe einer Formel, eine Variante der Schachtelungstiefe aus Definition 2.8.

Definition 13.1
Sei $A \in For0_\Sigma$ eine Formel in Negationsnormalform. die *k-Tiefe* von A, in Symbolen $d_k(A)$, zählt nur die Schachtelung von Konjunktionen.

$$\begin{array}{lcll} d_k(L) & = & 0 & \text{falls } L \text{ Literal} \\ d_k(A_1 \wedge A_2) & = & max\{d_k(A_1, d_k(A_2)\} + 1 & \\ d_k(A_1 \vee A_2) & = & max\{d_k(A_1, d_k(A_2)\} & \end{array}$$

Es gilt ist also $d_k(A) = 0$ genau dann wenn A eine Klausel, Disjuktion von Literalen, ist.

Wir schlagen die folgende Variante vor

$$ds = \Sigma_{F \in K(A)} 3^{d_k(F)}$$

Als Summe positiver Zahlen gilt stets $ds \geq 0$. Sei A die Formel vor einem beliebigen Schleifendurchlauf und A' die Formel danach. Ebenso seien ds und ds' die Werte der Variante vor und nach dem Schleifendurchlauf. Wir müssen $ds > ds'$ zeigen. Sei F eine minimale Formel in $K(A)$ und $F = B \vee (C \wedge D)$. (Der Fall $F = (C \wedge D) \vee B$ verläuft völlig analog.) A' geht also aus A hervor durch Ersetzung von F durch $(B \vee C) \wedge (B \vee D)$.

Nach Wahl von F können wir die Menge der Teilformelvorkommen $K(A)$ aufteilen in $K(A) = K_0 \cup \{F\} \cup K_1$, wobei K_0, die Teilformel enthält, die F echt enthalten, und K_1, die Teilformel, die disjunkt zu F liegen. Es gilt

$$K(A') \subseteq \{H' \mid H \in K_0(A)\} \cup \{(B \vee C), (B \vee D)\} \cup K_1(A)$$

wobei H' aus H entsteht, in dem F ersetzt wird durch $(B \vee C) \wedge (B \vee D)$.

Man prüft leicht nach, daß $d_k(F) = d_k((B \vee C) \wedge (B \vee D))$ gilt. Mit dem Analogon von Übungsaufgabe 2.2.2(1) für d_k anstelle von d, gilt somit für alle $H \in K_0(A)$ die Gleichheit $d_k(H) = d_k(H')$. Der einzige Unterschied zwischen ds und ds' besteht also darin, daß der Summand $3^{d_k(F)}$ wegfällt und schlimmstenfalls $3^{d_k(B \vee C)} + 3^{d_k(B \vee D)}$ hinzugefügt wird. Wegen

$$d_k(F) = max\{d_k(B), d_k(C \wedge D)\} = max\{d_k(B), d_k(C), d_k(D)\} + 1$$

gilt sowohl $d_k(B \vee C) < d_k(F)$ als auch $d_k(B \vee D) < d_k(F)$ und damit $3^{d_k(B \vee C)} + 3^{d_k(B \vee D)} < 3^{d_k(F)}$, wie gewünscht.

Aufgabe 2.4.3 von Seite 67

Zuerst transformieren wir $A \leftrightarrow (B \leftrightarrow C)$ in eine Formel, die nur aussagenlogischen Operatoren in $Op_{nf} = \{\neg, \wedge, \vee\}$ enthält und wenden dann die entsprechenden Distributivgesetze an.

$$\begin{array}{rcl}
A \leftrightarrow (B \leftrightarrow C) & \leftrightarrow & (A \wedge (B \leftrightarrow C)) \vee (\neg A \wedge \neg(B \leftrightarrow C)) \\
& \leftrightarrow & (A \wedge (\neg B \vee C) \wedge (B \vee \neg C)) \vee \\
& & (\neg A \wedge \neg((B \wedge C) \vee (\neg B \wedge \neg C)) \\
& \leftrightarrow & (A \wedge \neg B \wedge B) \vee (A \wedge \neg B \wedge \neg C) \vee \\
& & \quad (A \wedge C \wedge B) \vee (A \wedge C \wedge \neg C) \vee \\
& & (\neg A \wedge ((\neg B \vee \neg C) \wedge (B \vee C)) \\
& \leftrightarrow & (A \wedge \neg B \wedge \neg C) \vee (A \wedge C \wedge B) \vee \\
& & (\neg A \wedge \neg B \wedge B)(\neg A \wedge \neg B \wedge C) \\
& & \quad (\neg A \wedge \neg C \wedge B)(\neg A \wedge \neg C \wedge C) \\
& \leftrightarrow & (A \wedge \neg B \wedge \neg C) \vee (A \wedge C \wedge B) \vee \\
& & (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg C \wedge B)
\end{array}$$

Also ist

$$(A \land \neg B \land \neg C) \lor (A \land C \land B) \lor (\neg A \land \neg B \land C) \lor (\neg A \land \neg C \land B)$$

eine disjunktive Normalform von $A \leftrightarrow (B \leftrightarrow C)$.

$$\begin{array}{rcl}
A \leftrightarrow (B \leftrightarrow C) & \leftrightarrow & (\neg A \lor (B \leftrightarrow C)) \land (A \lor \neg(B \leftrightarrow C)) \\
& \leftrightarrow & (\neg A \lor (B \land C) \lor (\neg B \land \neg C)) \land \\
& & (A \lor \neg((\neg B \lor C) \land (B \lor \neg C)) \\
& \leftrightarrow & (\neg A \lor B \lor \neg B) \land (\neg A \lor B \lor \neg C) \land \\
& & \quad (\neg A \lor C \lor \neg B) \land (\neg A \lor C \lor \neg C) \land \\
& & (A \lor ((B \land \neg C) \lor (\neg B \lor C)) \\
& \leftrightarrow & (\neg A \lor B \lor \neg C) \land (\neg A \lor C \lor \neg B) \land \\
& & (A \lor B \lor \neg B) \land (A \lor B \lor C) \\
& & \quad (A \lor \neg C \lor \neg B) \land (A \lor \neg C \lor C) \\
& \leftrightarrow & (\neg A \lor B \lor \neg C) \land (\neg A \lor C \lor \neg B) \land \\
& & (A \lor B \lor C) \land (A \lor \neg C \lor \neg B)
\end{array}$$

Also ist

$$(\neg A \lor B \lor \neg C) \land (\neg A \lor \neg B \lor C) \land (A \lor B \lor C) \land (A \lor \neg B \lor \neg C)$$

eine konjunktive Normalform von $A \leftrightarrow (B \leftrightarrow C)$.

Aufgabe 2.4.4 von Seite 67

Wir geben die folgenden schrittweisen logisch äquivalenten Umformungen an die von A zu A' führen

$$\begin{array}{llllll}
1 & A = & (\neg P \land Q \land \neg R) & \lor & (P \land Q \land \neg R) & \lor \\
2 & & (P \land Q \land R) & \lor & & \\
3 & & (\neg P \land \neg Q \land R) & \lor & (P \land \neg Q \land R) & \\
4 & \leftrightarrow & (\neg P \land Q \land \neg R) & \lor & (P \land Q \land \neg R) & \lor \\
5 & & (P \land Q \land R) & \lor & (\neg P \land \neg Q \land R) & \lor \\
6 & & (\neg P \land \neg Q \land R) & \lor & (P \land \neg Q \land R) & \\
7 & \leftrightarrow & ((\neg P \lor P) & \land & (Q \land \neg R)) & \lor \\
8 & & ((P \land R) & \land & (Q \lor \neg Q)) & \lor \\
9 & & ((\neg Q \land R) & \land & (\neg P \lor P)) & \\
10 & \leftrightarrow & (\mathbf{1} & \land & (Q \land \neg R)) & \lor \\
11 & & ((P \land R) & \land & \mathbf{1}) & \lor \\
12 & & ((\neg Q \land R) & \land & \mathbf{1}) & \\
13 & \leftrightarrow & (Q \land \neg R)) & \lor & & \\
14 & & (P \land R) & \lor & & \\
15 & & (\neg Q \land R) & & & \\
16 & = & A' & & &
\end{array}$$

Bei der ersten äquivalenten Umformung wird nur zu Zeile 2, die schon vorhandene Disjunktion $(\neg P \wedge \neg Q \wedge R)$ ein zweites Mal hinzugefügt, so daß Zeile 5 entsteht. Zeile 1 wird unverändert in Zeile 4 übernommen, ebenso Zeile 3 in Zeile 6. Die nächste Umformungen wenden in den Zeilen 4 bis 6 jeweils das Distributivgesetz an, was zu den logisch äquivalenten Formeln in den Zeilen 7 bis 9 führt. Im nächsten Schritt, Zeile 10 bis 12, werden $(P \vee \neg P)$ und $(Q \vee \neg Q$ logisch äquivalent durch $\mathbf{1}$ ersetzt. Die Zeilen 13 bis 15 folgen aus 10 bis 12 jeweils durch Anwendung der Tautologie $\mathbf{1} \wedge F \leftrightarrow F$.

Die Äquivalenz von A zu A'' wird nach demselben Muster bewiesen. Wir geben nur noch die wichtigsten Zwischenschritte an.

$$
\begin{array}{rllclcl}
1 & A & = & (\neg P \wedge Q \wedge \neg R) & \vee & (P \wedge Q \wedge \neg R) & \vee \\
2 & & & (P \wedge Q \wedge R) & \vee & & \\
3 & & & (\neg P \wedge \neg Q \wedge R) & \vee & (P \wedge \neg Q \wedge R) & \\
4 & & \leftrightarrow & (\neg P \wedge Q \wedge \neg R) & \vee & (P \wedge Q \wedge \neg R) & \vee \\
5 & & & (P \wedge Q \wedge \neg R) & \vee & (P \wedge Q \wedge R) & \vee \\
6 & & & (\neg P \wedge \neg Q \wedge R) & \vee & (P \wedge \neg Q \wedge R) & \\
7 & & \leftrightarrow & ((\neg P \vee P) & \wedge & (Q \wedge \neg R)) & \vee \\
8 & & & ((P \wedge Q) & \wedge & (R \vee \neg R)) & \vee \\
9 & & & ((\neg Q \wedge R) & \wedge & (\neg P \vee P)) & \\
10 & & \leftrightarrow & (Q \wedge \neg R) & \vee & & \\
11 & & & (P \wedge Q) & \vee & & \\
12 & & & (\neg Q \wedge R) & & & \\
13 & & = & A'' & & &
\end{array}
$$

Aufgabe 2.4.5

Aus der vorangegangen Aufgabe liest man ab, daß die folgenden Implikationen Tautologien sind

$$
\begin{array}{ll}
(Q \wedge \neg R) \rightarrow A & (P \wedge R) \rightarrow A \\
(P \wedge Q) \rightarrow A & (\neg Q \wedge R) \rightarrow A
\end{array}
$$

Die linken Seiten dieser Implikationen kommen ja als Diskjunktsglieder in der zu A logisch äquivalenten Formel A' oder A'' vor. Dagegen ist keine der Implikationen

$$
\begin{array}{lll}
Q \rightarrow A & \neg R \rightarrow A & P \rightarrow A \\
R \rightarrow A & \neg Q \rightarrow A &
\end{array}
$$

eine Tautologie.

Aufgabe 2.4.6

Sei $A = D_1 \vee \ldots \vee D_k$ eine disjunktive Normalform von B. Damit sind $D_i \rightarrow B$

für jedes $\leq i \leq k$ Tautologien.
Angenommen es gibt für ein i eine echte Teilkonjunktion D'_i von D_i für die immer noch $D'_i \rightarrow B$ eine Tautologie ist. Im folgenden nehmen wir der Einfachheit halber $i = 1$ an. Da $D_1 \rightarrow D'_1$ allgemeingültig ist, ist es auch $A \rightarrow D'_1 \vee D_2 \vee \ldots \vee D_k$. Umgekehrt kann man von $D'_1 \rightarrow B$ und $D_j \rightarrow B$ für $2 \leq j \leq k$ schließen auf $D'_1 \vee D_2 \vee \ldots \vee D_k \rightarrow B$, was wegen der angenommen logischen Äquivalenz von A und B auch $D'_1 \vee D_2 \vee \ldots \vee D_k \rightarrow A$ zu Folge hat. Damit ist $D'_1 \vee D_2 \vee \ldots \vee D_k$ eine DNF von B mit echt weniger Symbolen als A im Widerspruch zur vorausgesetzten Minimalität von A.

Aufgabe 2.4.7

Sei $\bigvee_i \bigwedge_j A_{ij}$ eine disjunktive Normalform von A und $\bigwedge_r \bigvee_s B_{rs}$ eine konjunktive Normalform von B. Somit ist auch

$$\bigvee_i \bigwedge_j A_{ij} \rightarrow \bigwedge_r \bigvee_s B_{rs}$$

eine Tautologie. Für jedes i und r ist dann auch

$$\bigwedge_j A_{ij} \rightarrow \bigvee_s B_{rs}$$

eine Tautologie. Wir suchen zunächst Interpolanten C_{ir} für jede dieser Implikationen. Genauer gesagt, suchen wir C_{ir} mit

1. $\bigwedge_j A_{ij} \rightarrow C_{ir}$ und $C_{ir} \rightarrow \bigvee_s B_{rs}$ und
2. C_{ir} ist ein Literal, das sowohl in $\bigwedge_j A_{ij}$ als auch in $\bigvee_s B_{rs}$ vorkommt, oder C_{ir} ist gleich einer der beiden Konstanten $\mathbf{1}$ oder $\mathbf{0}$.

Falls $\bigvee_s B_{rs}$ eine Tautologie ist, dann erfüllt $C_{rs} = \mathbf{1}$ alle Forderungen. Falls $\neg \bigwedge_j A_{ij}$ eine Tautologie ist, dann tut es $C_{rs} = \mathbf{0}$. Ist weder $\neg \bigwedge_j A_{ij}$ noch $\bigvee_s B_{rs}$ eine Tautologie, dann behaupten wir, daß $\bigwedge_j A_{ij}$ und $\bigvee_s B_{rs}$ ein gemeinsames Literal haben müssen. Wäre das nicht der Fall, dann könnten wir eine Interpretation I finden, so daß für alle j $val_I(A_{ij}) = \mathbf{1}$ gilt und für alle s gilt $val_I(B_{rs}) = \mathbf{0}$. Man beachte, daß wir an dieser Stelle ausnutzen, daß nicht L und $\bar{L}$ beide in $\bigwedge_j A_{ij}$ vorkommen. Wie gleiche Eigenschaft wird auch für $\bigvee_s B_{rs}$ ausgenutzt. Für die Interpretation I gilt jedenfalls

$$val_I(\bigwedge_j A_{ij} \rightarrow \bigvee_s B_{rs}) = \mathbf{0}$$

im Widerspruch zur Tautologieeigenschaft von $\bigwedge_j A_{ij} \to \bigvee_s B_{rs}$. Damit ist gezeigt, daß es ein gemeinsames Literal von $\bigwedge_j A_{ij}$ und $\bigvee_s B_{rs}$ gibt und wir wählen C_{ir} als ein solches. Die Implikationen $\bigwedge_j A_{ij} \to C_{ir}$ und $C_{ir} \to \bigvee_s B_{rs}$ sind dann offensichtlich wahr.

Da für alle i und alle r die Formel $\bigwedge_j A_{ij} \to C_{ir}$ eine Tautologie ist, ist auch $\bigvee_i \bigwedge_j A_{ij} \to \bigvee_i C_{ir}$ eine Tautologie für jedes r, also auch $\bigvee_i \bigwedge_j A_{ij} \to \bigwedge_r \bigvee_i C_{ir}$. Die gleiche Argumentation können wir für die zweite Implikation führen: Da für alle i und r $C_{ir} \to \bigvee_s B_{rs}$ eine Tautologie ist, ist es auch $\bigvee_i C_{ir} \to \bigvee_s B_{rs}$ für alle r. Somit ist auch $\bigwedge_r \bigvee_i C_{ir} \to \bigwedge_r \bigvee_s B_{rs}$ eine Tautologie. Die Formel $C = \bigwedge_r \bigvee_i C_{ir}$ erfüllt somit alle Eigenschaften der interpolierenden Formel.

Aufgabe 2.4.8

$$\neg C_3 \Leftrightarrow (P_1 \leftrightarrow (P_2 \leftrightarrow (P_3 \leftrightarrow \neg P_1))) \Leftrightarrow$$
$$\Big(P_1 \wedge ((P_2 \wedge ((P_3 \wedge \neg P_1) \vee (\neg P_3 \wedge P_1)))$$
$$\vee(\neg P_2 \wedge ((P_3 \wedge P_1) \vee (\neg P_3 \wedge \neg P_1))))\Big)$$
$$\vee\Big(\neg P_1 \wedge ((\neg P_2 \wedge ((\neg P_3 \wedge P_1) \vee (P_3 \wedge \neg P_1)))$$
$$\vee(P_2 \wedge ((\neg P_3 \wedge \neg P_1) \vee (P_3 \wedge P_1))))\Big) \Leftrightarrow \ldots$$

Dagegen $(C_n)_{kknf}$:

$$\left.\begin{array}{l} a_1 \wedge \\ a_1 \leftrightarrow (P_1 \leftrightarrow a_2) \wedge \\ a_2 \leftrightarrow (P_2 \leftrightarrow a_3) \wedge \\ \ldots \\ a_n \leftrightarrow (P_n \leftrightarrow \neg P_1) \end{array}\right\} \quad \text{ergibt } 12 * n + 1 \text{ Klauseln}$$

Aufgabe 2.4.11

1. $sh(P_i, 0, 1)$
2. $sh(P_1, 0, sh(P_2, 0, 1))$
3. $sh(P_i, 1, 0)$
4. $sh(P_1, sh(P_2, 0, 1), 1)$

Aufgabe 2.4.14 Um

$$sh(sh(P_1, P_2, P_3), P_4, P_5) \equiv sh(P_1, sh(P_2, P_4, P_5), sh(P_3, P_4, P_5))$$

zu beweisen, kann man die rechte und linke Seite in disjunktive Normalform zerlegen und vergleichen:

$$\begin{array}{rcl} sh(sh(P_1,P_2,P_3),P_4,P_5) & \leftrightarrow & (sh(P_1,P_2,P_3)\wedge P_5)\vee(\neg sh(P_1,P_2,P_3)\wedge P_4) \\ & \leftrightarrow & (sh(P_1,P_2,P_3)\wedge P_5)\vee(sh(P_1,\neg P_2,\neg P_3)\wedge P_4) \\ & \leftrightarrow & (P_1\wedge P_3\wedge P_5)\vee(\neg P_1\wedge P_2\wedge P_5)\vee \\ & & (P_1\wedge\neg P_3\wedge P_4)\vee(\neg P_1\wedge\neg P_2\wedge P_4) \end{array}$$

$$\begin{array}{rcl} sh(P_1,sh(P_2,P_4,P_5),sh(P_3,P_4,P_5)) & \leftrightarrow & (P_1\wedge sh(P_3,P_4,P_5))\vee(\neg P_1\wedge sh(P_2,P_4,P_5)) \\ & \leftrightarrow & (P_1\wedge P_3\wedge P_5)\vee(P_1\wedge\neg P_3\wedge P_4)\vee \\ & & (\neg P_1\wedge P_2\wedge P_5)\vee(\neg P_1\wedge\neg P_2\wedge P_4) \end{array}$$

Es gibt allerdings auch eine schneller Möglichkeit indem man die Fallunterscheidung $P_1 = 0$ und $P_1 = 1$ betrachten. Im ersten Fall erhält man

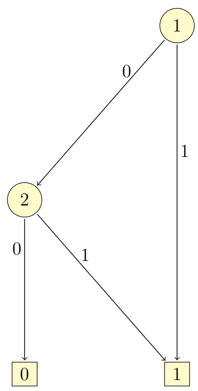
$$\begin{array}{lcl} sh(sh(0,P_2,P_3),P_4,P_5) & \equiv & sh(0,sh(P_2,P_4,P_5),sh(P_3,P_4,P_5)) \\ sh(P_2,,P_4,P_5) & \equiv & sh(P_2,P_4,P_5) \end{array}$$

im zweiten:

$$\begin{array}{lcl} sh(sh(1,P_2,P_3),P_4,P_5) & \equiv & sh(1,sh(P_2,P_4,P_5),sh(P_3,P_4,P_5)) \\ sh(P_3,P_4,P_5) & \equiv & sh(P_3,P_4,P_5) \end{array}$$

Aufgabe 2.4.15

Die Vermutung ist falsch. Der Shannongraph



repräsentiert die Formel $P_1 \vee P_2$. Die beiden Pfade von der Wurzel zum 1-Blatt liefern die Konjunktionen P_1 und $\neg P_1 \wedge P_2$. P_1 ist sicherlich ein Primimplikand von $P_1 \vee P_2$, aber $\neg P_1 \wedge P_2$ ist es nicht, da es den kürzeren Implikanden P_2 gibt.

Aufgabe 2.4.16

Wir führen die neuen aussagenlogischen Atome Q_1, Q_2, Q_3, Q_4 ein.

$$\underbrace{\underbrace{(\underbrace{(A_1 \wedge A_2)}_{Q_1} \vee \underbrace{(A_1 \wedge A_3)}_{Q_2}}_{Q_4} \vee \neg \underbrace{(A_2 \wedge A_3)}_{Q_3}}_{A}$$

1.Schritt	2.Schritt	3.Schritt
$Q_1 \leftrightarrow (A_1 \wedge A_2)$	$\neg Q_1 \vee (A_1 \wedge A_2)$	$\neg Q_1 \vee A_1$
		$\neg Q_1 \vee A_2$
	$Q_1 \vee \neg A_1 \vee \neg A_2$	$Q_1 \vee \neg A_1 \vee \neg A_2$
$Q_2 \leftrightarrow (A_1 \wedge A_3)$	$\neg Q_2 \vee (A_1 \wedge A_3)$	$\neg Q_2 \vee A_1$
		$\neg Q_2 \vee A_3$
	$Q_2 \vee \neg A_1 \vee \neg A_3$	$Q_2 \vee \neg A_1 \vee \neg A_3$
$Q_3 \leftrightarrow (A_2 \wedge A_3)$	$\neg Q_3 \vee (A_2 \wedge A_3)$	$\neg Q_3 \vee A_2$
		$\neg Q_3 \vee A_3$
	$Q_3 \vee \neg A_2 \vee \neg A_3$	$Q_3 \vee \neg A_2 \vee \neg A_3$
$Q_4 \leftrightarrow (Q_1 \vee Q_2)$	$\neg Q_4 \vee Q_1 \vee Q_2$	$\neg Q_4 \vee Q_1 \vee Q_2$
	$Q_4 \vee \neg(Q_1 \vee Q_2)$	$Q_4 \vee \neg Q_1$
		$Q_4 \vee \neg Q_2$
$A \leftrightarrow (Q_4 \vee \neg Q_3)$	$\neg A \vee Q_4 \vee \neg Q_3$	$\neg A \vee Q_4 \vee \neg Q_3$
	$A \vee \neg(Q_4 \vee \neg Q_3)$	$A \vee \neg Q_4$
		$A \vee Q_3$
A		

Aufgabe 2.4.17

Angenommen es gäbe eine erfüllende Interpretation I für $S \setminus \{C\}$. Sei L ein isoliertes Literal in C. Nach Annahme muß es mindestens ein solches geben. Wir setzen:

$$J(P) = \begin{cases} I(P) & \text{falls } P \text{ nicht im Literal } L \text{ vorkommt} \\ W & \text{falls } L = P \\ F & \text{falls } L = \neg P \end{cases}$$

Offensichtlich gilt $J(C) = W$. Aber gilt auch noch $J(D) = W$ für alle anderen Klauseln D in S? Sicher, da $\bar{L}$ in keiner Klausel in S vorkommt, kann schlimmstenfalls der Wechseln von $I(L) = F$ zu $J(L) = W$ erfolgen. Aus $I(D) = W$ folgt also auch $J(D) = W$.

Aufgabe 2.4.18 von Seite 71

Teil 1

Sei F eine Formel, die nur die Operatoren $\wedge, \oplus, 1$ benutzt. Mit Hilfe des

Distributivgesetzes aus Aufgabe 2.3.2 kann man F äquivalent umformen zu einer Formel der Form $\bigoplus_{1\leq i\leq k} F_i$, wobei die F_i Konjunktionen sind, $F_i = \bigwedge_{1\leq j\leq r_i} A_{ij}$ wobei A_{ij} ein aussagenlogisches Atom ist oder $A_{ij} = 1$. Indem man die Tautologien $1 \wedge A \leftrightarrow A$ und $A \wedge A \leftrightarrow A$ ausnutzt kann man die F_i weiter logisch äquivalent verfeinern, so daß entweder $F_i = 1$ oder $F_i = \bigwedge_{1\leq j\leq r_i} A_{ij}$, so daß alle A_{ij} Atome sind und für jedes i jedes Atom höchstens einmal in $\{A_{i,1},\ldots A_{i,r_i}\}$ vorkommt. Sind F_{i_1}, F_{i_2} bis auf die Reihenfolge gleich, kann man wegen der Tautologien $A \oplus A = 0$ und $A \oplus 0 = A$ beide aus der Summe weglassen ohne die logische Äquivalenz zu gefährden. Im Extremfall kann es passieren, daß durch Wiederholung dieser Operation die leere Summe übrig bleibt.

Teil 2

Die Formeln in Reed-Muller Normalform für $\Sigma = \{A_1,\ldots,A_n\}$ stehen in eindeutiger Korrespondenz zu den Mengen $\mathcal{A}$ von Teilmengen von Σ, $\mathcal{A} \subseteq \mathcal{P}(\Sigma)$ oder $\mathcal{A} \in \mathcal{P}(\mathcal{P}(\Sigma))$, wenn $\mathcal{P}$ den Potenzmengenoperator bezeichnet. Ist $\mathcal{A}$ gegeben so ist die zugeordente Normalform

$$\bigoplus_{a\in\mathcal{A}} \bigwedge_{A\in a} A$$

Ist umgekehrt $\bigoplus_{1\leq i\leq k} \bigwedge_{1\leq j\leq r_i} A_{ij}$ gegeben, so ist $\mathcal{A} = \{a_i \mid 1 \leq i \leq k\}$ mit $a_i = \{A_{ij} \mid 1 \leq j \leq r_i\}$.

Es gibt genau 2^{2^n} verschiedene Boolesche Funktionen mit n Argumenten und auch 2^{2^n} verschiedene Teilmengen von $\mathcal{P}(\Sigma)$. Somit kann es zu jeder Booleschen Funktion nur eine Reed-Muller Normalform geben.

Aufgabe 2.4.19 von Seite 72

Sei f eine n-stellige symmetrische Boolesche Funktion. Zur Konstruktion des sh-Graphen benutzen wir die aussagenlogischen Variablen $P_1,\ldots,P_n$ in dieser Reihenfolge, wobei wir die i-te Variable P_i der i-ten Argumentposition von f zu ordnen. Wir benutzen durchweg die in der Aufgabenstellung etablierte Notation für symmetrische Boolesche Funktionen. Wir können also annehmen, daß $f = f^n_X$ für eine geeignete Menge $X \subseteq \{0,\ldots n\}$.

Der Beweis wird aus einer genaueren Analyses des Verfahrens, zu einer gegebene Funktion einen zugehörigen sh-Graphen zu konstruieren, folgen, wie es im Beweis von Lemma 2.50 beschrieben und in Beispiel 2.51 vorgeführt wurde.

Die Konstruktion des sh-Graphen zu f^n_X, mit Wurzelmarkierung P_1, wird schrittweise reduzierte auf die Konstruktion von sh-Graphen mit Wurzelmarkierung P_j für eine Funktion f_Y^{n-j+1}. Zwei P_j markierte Knoten, für die Funktion $f_{Y_1}^{n-j+1}$ und $f_{Y_2}^{n-j+1}$ können identifiziert werden, wenn $Y_1 = Y_2$ gilt. Welche Y können überhaupt vorkommen? Die Funktionen $f_{Y_1}^{n-1}$, $f_{Y_2}^{n-1}$ entstehen, indem in f_X^n die Variable P_1 auf **W** bzw. **F** gesetzt wird. Somit erhalten wird $Y_1 = X - 1 = \{n - 1 \mid n \in X\}$ und $Y_2 = X$. Dasselbe trifft für alle weiteren f_Y^{n-j+1} zu. Es können also genau die Funktionen f_{X-k}^m für $1 \leq m, k \leq n$ vorkommen. Das sind aber höchstens n^2 viele.

Häufig kommt man sogar mit weniger Knoten aus, da die Funktionen f_X^m und $f_{X\cap\{0,1,\ldots m\}}^m$ übereinstimmen. Denn unter den m Argumenten kann auch höchstens m-mal **W** vorkommen.

∎

Aufgabe 2.4.20 von Seite 72

Am einfachsten ist es alle Pfade von der Wurzel zum Blattknoten **W** zu betrachten. Im Graphen G_{ex} gibt es davon nur zwei. Wir ordnen jedem Pfad eine Konjunktion von Literalen zu nach der folgenden Vorschrift: Tritt auf dem Pfad der Index i auf so so erscheint P_i in der Konjunktion falls der Pfad der **W** Kante folgt, und $\neg P_i$ falls der Pfad der **F** Kante folgt. Dem Pfad $1 - 3 - \mathbf{W}$ wird also die Konjunktion $P_1 \wedge P_3$ zu geordnet. Dem Pfad $1 - 2 - 3 - \mathbf{W}$ die Konjunktion $\neg P_1 \wedge P_2 \wedge P_3$. Aus der Definition von $f_{G_{ex}}$ ersieht man leicht, daß die Boolesche Funktion von $(P_1 \wedge P_3) \vee (\neg P_1 \wedge P_2 \wedge P_3)$ mit $f_{G_{ex}}$ übereinstimmt. Die angegebene Formel läßt sich vereinfachen zu $(P_1 \vee P_2) \wedge P_3$.

Aufgabe 2.4.21 von Seite 73

Seien $p_1, \ldots p_k$ alle maximalen Pfade von der Wurzel zum Blattknoten **W**. Jedem Booleschen Eingabetupel $\vec{b}$ wird wie in Definition 2.49 beschrieben ein Pfad zu geordnet. Sei n_i die Anzahl der Eingabetupel, die zu dem Pfad p_i führen. Kommte jede Variable x_j auf p_i vor ist $n_i = 1$. Kommt mindesten eine Variable x_j auf dem Pfad p_i nicht vor so ist $n_i > 0$ eine gerade Zahl.

Die Anzahl A der Eingabetupel $\vec{b}$ mit $f(\vec{b}) = \mathbf{W}$ berechnet sich zu

$$A = n_1 p_1 + \ldots + n_k p_k$$

Da die Summe A eine ungerade Zahl ein soll können nicht alle n_i gerade sein.

Offensichtlich ist die vorangegangene Argumentation unabhängig von der Variablenordnung.

Da die Anzahl aller Eingabetupel in $\{\mathbf{W}, \mathbf{F}\}^n$ gleich 2^n und damit gerade ist, muß auch die Anzahl der $\bar{b} \in \{\mathbf{W}, \mathbf{F}\}^n$ mit $f(\bar{b}) = \mathbf{F}$ ungerade sein. Eine Wiederholung des obigen Arguments zeigt, daß auch zum zweiten Wurzelknoten **F** ein Pfad der Länge n existieren muß.

In [LR13, Kapitel 55.2] wird diese Lösung Ron Rivest zugeschrieben, dem Miterfinder des RSA Verschlüsselungsverfahrens.

∎

13.4 Lösungen zu Kapitel 3 AL Erfüllbarkeitstests

Aufgabe 3.5.1 von Seite 89

Eine disjunktive Normalform $F = \bigvee_{i=1}^{i=n} F_i$ ist genau dann erfüllbar, wenn eine der Teilformeln F_i erfüllbar ist. Jedes F_i ist Konjunktion von Literalen, also erfüllbar wenn kein Paar komplementärer Literale in F_i vorkommt. Das läßt sich bei geeigneter Datenstruktur in $O(n)$ feststellen.

Aufgabe 3.5.2 von Seite 89

Wir vereinbaren für $A, S \in \ddot{A}qFor$ die Notation

$$\#_A(S) := \text{Anzahl der Vorkommen von } S \text{ in } A$$

Nach Satz 3.9 ist $A \in \ddot{A}qFor$ eine Tautologie gdw

1. $\#_A(P)$ gerade für alle $P \in \Sigma$ und
2. $\#_A(\mathbf{0})$ gerade.

Wir schließen jetzt

A ist unerfüllbar	gdw	$\neg A$ ist eine Tautologie
	gdw	$A \leftrightarrow \mathbf{0}$ ist eine Tautologie
	gdw	Für alle P ist $\#_{A\leftrightarrow\mathbf{0}}(P)$ gerade und $\#_{A\leftrightarrow\mathbf{0}}(\mathbf{0})$ gerade
	gdw	Für alle P ist $\#_A(P)$ gerade und $\#_A(\mathbf{0})$ ungerade

Aufgabe 3.5.3 von Seite 90

Man sieht leicht, daß die im Teil (a) des Korrektheitsbeweises von Satz 3.7 konstruiuerte Interpretation minimal.

Aufgabe 3.5.4 von Seite 90

Sei C beliebige definite Horn–Formel.
Die (triviale) Interpretation $I \equiv W$ erfüllt C, d.h. es gilt $val_I(C) = W$.

Beweis: Sei $D = L_1 \vee \cdots \vee L_k$ beliebige Klausel von C (da C definit ist, muß $k \geq 1$ also $C \neq \Box$ sein!).

Genau ein L_i ist positiv, sei z.B. $L_{i_0} = P$ für ein $P \in \Sigma$. Nach Def. von I und val ist $I(P) = W = val_I(P) = val_I(L_{i_0})$. Also ist auch $val_I(L_1 \vee \cdots \vee L_k) = val_I(D) = W$. Da D beliebig war, gilt $val_I(C) = W$.

Aufgabe 3.5.5 von Seite 90

Eine Möglichkeit, neben einer Reihe anderer, ist die folgende Wahl folgender Aussagenvariablen:

$$\begin{array}{l} Mo, Di, Mi, Do, Fr, Sa, So \\ GMo, GDi, GMi, GDo, GFr, GSa, GSo \\ LW \\ EW \end{array}$$

mit der intendierten Bedeutung

X	ist wahr, gdw	heute Xtag ist
GX	ist wahr, gdw	gestern Xtag war
LW	ist wahr, gdw	der Löwe die Wahrheit sagt.
EW	ist wahr, gdw	das Einhorn die Wahrheit sagt.

Die aussagenlogische Formulierung UL des Rätsels vom Löwen und dem Einhorn läßt sich damit aufschreiben:

$$\begin{array}{l} Mo \rightarrow \neg(Di \vee Mi \vee Do \vee Fr \vee Sa \vee So) \\ Di \rightarrow \neg(Mo \vee Mi \vee Do \vee Fr \vee Sa \vee So) \\ Mi \rightarrow \neg(Mo \vee Di \vee Do \vee Fr \vee Sa \vee So) \\ Do \rightarrow \neg(Mo \vee Di \vee Mi \vee Fr \vee Sa \vee So) \\ Fr \rightarrow \neg(Mo \vee Di \vee Mi \vee Do \vee Sa \vee So) \\ Sa \rightarrow \neg(Mo \vee Di \vee Mi \vee Do \vee Fr \vee So) \\ So \rightarrow \neg(Mo \vee Di \vee Mi \vee Do \vee Fr \vee Sa) \\ GMo \leftrightarrow Di \\ GDi \leftrightarrow Mi \\ GMi \leftrightarrow Do \\ GDo \leftrightarrow Fr \\ GFr \leftrightarrow Sa \\ GSa \leftrightarrow So \\ GSo \leftrightarrow Mo \\ \neg LW \leftrightarrow Mo \vee Di \vee Mi \\ \neg EW \leftrightarrow Do \vee Fr \vee Sa \\ LW \leftrightarrow GMo \vee GDi \vee GMi \\ EW \leftrightarrow GDo \vee GFr \vee GSa \end{array}$$

Diese Formelmenge könnte man jetzt in Klauselnormalform bringen und von einem SAT-solver lösen lassen. Die Lösung ist jedoch relativ leicht selbst zu erschließen.

Sei I ein erfüllende Belegung für UL. Wir bemerken zunächst, daß durch die Auflösung der gestrigen Tage gilt

$$\begin{array}{l} LW \leftrightarrow Di \vee Mi \vee Do \\ EW \leftrightarrow Fr \vee Sa \vee So \end{array}$$

Aus $val_I(Di) = \mathbf{W}$ würde sowohl $val_I(LW) = \mathbf{W}$ als auch $val_I(\neg LW) = \mathbf{W}$ folgen, was nicht sein kann. Also $val_I(Di) = \mathbf{F}$. Ebenso schließen wir auf $val_I(Mi) = \mathbf{F}$. Aus $val_I(Fr) = \mathbf{W}$ würde $val_I(EW) = \mathbf{W}$ als auch $val_I(\neg EW) = \mathbf{W}$ folgen, was ebenfalls nicht sein kann. Also $val_I(Fr) = \mathbf{F}$. Nach demselben Muster erhalten wir auch $val_I(Sa) = \mathbf{F}$. Andererseits muß $val_I(LW \vee \neg LW) = \mathbf{W}$ sein. Was nach den in LU enthaltenen Äquivalenzen gleichbedeutend ist mit $val_I(Mo \vee Di \vee Mi \vee Do) = \mathbf{W}$. Nachdem was wir über Di und Mi wissen heißt das $val_I(Mo \vee Do) = \mathbf{W}$. Dasselbe Argument für EW anstelle von LW liefert $val_I(So \vee Do) = \mathbf{W}$. Das läßt zunächst noch die folgenden Möglichkeiten offen:

1. $val_I(Mo) = \mathbf{W}$ und $val_I(So) = \mathbf{W}$
2. $val_I(Mo) = \mathbf{W}$ und $val_I(Do) = \mathbf{W}$
3. $val_I(Do) = \mathbf{W}$ und $val_I(So) = \mathbf{W}$
4. $val_I(Do) = \mathbf{W}$

Jetzt kommen die erste 7 Formeln in LU ins Spiel. Diese schließen die Lösungen 1 bis 3 aus. Die richtige Lösung ist somit „Donnerstag".

Man hätte vielleicht erwartet, das die auch folgende Hintergrundinformation wichtig sein könnte:

$$\begin{array}{lcl} Mo & \leftarrow & \neg(Di \vee Mi \vee Do \vee Fr \vee Sa \vee So) \\ Di & \leftarrow & \neg(Mo \vee Mi \vee Do \vee Fr \vee Sa \vee So) \\ Mi & \leftarrow & \neg(Mo \vee Di \vee Do \vee Fr \vee Sa \vee So) \\ Do & \leftarrow & \neg(Mo \vee Di \vee Mi \vee Fr \vee Sa \vee So) \\ Fr & \leftarrow & \neg(Mo \vee Di \vee Mi \vee Do \vee Sa \vee So) \\ Sa & \leftarrow & \neg(Mo \vee Di \vee Mi \vee Do \vee Fr \vee So) \\ So & \leftarrow & \neg(Mo \vee Di \vee Mi \vee Do \vee Fr \vee Sa) \end{array}$$

Die obige Lösung zeigt, daß das nicht der Fall ist.

Aufgabe 3.5.6 Wir benötigen die aussagenlogischen Variablen R_i, B_i, G_i für $0 \leq i < L$. Die Intention ist, daß R_i wahr ist, wenn das Land i rot gefärbt wird. Entsprechend für B_i und G_i.

$$\begin{aligned} FF \quad = \quad & \{\bigwedge_{0 \leq i < L}[\neg(G_i \wedge R_i) \wedge \neg(G_i \wedge B_i) \wedge \neg(B_i \wedge R_i)] \wedge \\ & \bigwedge_{0 \leq i < L}(G_i \vee B_i \vee R_i) \wedge \\ & \bigwedge_{i<j \& N(i,j)}[\neg(G_i \wedge G_j) \wedge \neg(R_i \wedge R_j) \wedge \neg(B_i \wedge B_j)]\} \end{aligned}$$

13.5 Lösungen zu Abschnitt 4.2 PL Syntax

Aufgabe 4.2.1 von Seite 109

1. Man sieht aus Definition 4.3, daß jeder Term mit einem Zeichen aus F_Σ beginnen muß. Außerdem überlegt man sich leicht anhand der Definitionen 4.4 und 4.5, daß jede Formel mit einem Zeichen aus P_Σ beginnen muß. Nach Definition 4.2 einer Signatur sind F_Σ und P_Σ disjunkt. Also kann eine Zeichenfolge nicht gleichzeitig ein Term und eine Formel sein.

2. Wie in der Lösung der ersten Teilaufgabe schon gemerkt, muß jede Formel mit einem Zeichen aus P_Σ beginnen. In einem Term kommen aber neben Variablen nur Zeichen aus F_Σ vor. Wegen der Disjunktheit von P_Σ mit F_Σ folgt die Behautpung unmittelber.

3. Die Lösung ist ein erstes Beispiel für das Beweisprinzip der *strukturellen Induktion*. Im vorliegenden Fall zeigen wir

   (1) die behauptete Aussage ist wahr, wenn s eine Variable ist.

   (2) ist $s = f(s_1, \ldots, s_n)$ und ist die Aussage für alle s_i wahr, dann ist sie auch für s richtig.

   Aus (1) und (2) folgt, daß die in Frage stehende Aussage für alle Terme s wahr ist. Man beachte, daß Teil (2) auch den Fall eines 0-stelligen Funktionszeichens abdeckt.

   zu (1) Für die beiden Teilterme t und u muß $t = u = s$ gelten und damit auch Alternative (a) in der Aufgabe,
   zu (2) Ist $t = s$, dann ist Alternative (a) wahr. Ist $t \neq s$ aber $u = s$, dann ist Alternative (b) wahr.

   Für den Rest des Arguments sind t und u echte Teilterme von s. Nach Definition von Teiltermen muß es also i_1, i_2 geben, so daß t Teilterm von s_{i_1} ist und u Teilterm von s_{i_2}. Gilt $i_1, \neq i_2$ dann sind t und u disjunkt und Alternative (c) trifft zu, Der verbleibende Fall, daß t und u beide Teilterme von s_i sind $i = i_1 = i_2$ folgt jetzt aus der Induktionsvoraussetzung.

4. `fehlt noch`

5. `fehlt noch`

6. `fehlt noch`

Aufgabe **4.2.2** von Seite 110
Wir benutzen die Bezeichnung $Tt(t)$ für die Menge aller Teilterme von t und $Tf(A)$ für die Menge aller Teilformeln von A.

$$
\begin{array}{lcl}
Tt(t) & = & \{t\} \\
 & & \text{falls } t \text{ eine Konstante oder eine Variable ist.} \\
Tt(f(t_1,\ldots,t_n)) & = & \{f(t_1,\ldots,t_n)\} \cup Tt(t_1) \cup \ldots \cup Tt(t_n) \\
Tf(A) & = & \{A\} \\
 & & \text{falls } A \text{ eine atomare Formel ist.} \\
Tf(\neg A) & = & \{\neg A\} \cup Tf(A) \\
Tf(A \circ B) & = & \{A \circ B\} \cup Tf(A) \cup Tf(B) \\
Tf(\forall x A) & = & \{\forall x A\} \cup Tf(A) \\
Tf(\exists x A) & = & \{\exists x A\} \cup Tf(A)
\end{array}
$$

Aufgabe **4.2.3** von Seite 110
`fehlt noch`
Aufgabe **4.2.4** von Seite 110
Wir definieren als Vorbereitung die Menge $V(t)$ der in einem Term t vorkommenden Variablen:

$$
\begin{array}{lcl}
V(x) & = & \{x\} \\
V(f(t_1,\ldots,t_n)) & = & \bigcup_{i=1}^{n} V(t_i)
\end{array}
$$

Jetzt folgen die rekursiven Definitionen der freien $Frei(A)$ und der gebundenen $Bd(A)$ Variablen in einer Formel A.

$$
\begin{array}{lcl}
Frei(s \doteq t) & = & V(s) \cup V(t) \\
Frei(p(t_1,\ldots,t_n)) & = & \bigcup_{i=1}^{n} V(t_i) \\
Frei(A_1 \circ A_2) & = & Frei(A_1) \cup Frei(A_2) \\
 & & \text{für jede aussagenlogische Verknüpfung } \circ \\
Frei(\forall x A_1) & = & Frei(A_1) \setminus \{x\} \\
Frei(\exists x A_1) & = & Frei(A_1) \setminus \{x\}
\end{array}
$$

$$
\begin{array}{lcl}
Bd(s \doteq t) & = & \emptyset \\
Bd(p(t_1,\ldots,t_n)) & = & \emptyset \\
Bd(A_1 \circ A_2) & = & Bd(A_1) \cup Bd(A_2) \\
 & & \text{für jede aussagenlogische Verknüpfung } \circ \\
Bd(\forall x A_1) & = & Bd(A_1) \cup \{x\} \\
Bd(\exists x A_1) & = & Bd(A_1) \cup \{x\}
\end{array}
$$

Aufgabe **4.2.5**
`fehlt noch`
Aufgabe **4.2.6**
`fehlt noch`
Aufgabe **4.2.7**
Die einzigen Variablen die in $kVg(t_1,t_2)$ vorkommen, sind die $nv(s_1,s_2)$. Wir

definieren die Substitutionen σ_i für $i = 1, 2$ durch $\sigma_i(nv(s_1, s_2)) = t_i$. Durch Induktion nach dem Formelaufbau von $kVg(t_1, t_2)$ zeigt man jetzt

$$\sigma_i(kVg(t_1, t_2)) = t_i$$

Besteht $kVg(t_1, t_2)$ nur aus einer Variablen, also $kVg(t_1, t_2) = nv(t_1, t_2)$, dann ist diese Behauptung offensichtlich richtig. Im anderen Fall gilt $kVg(t_1, t_2) = f(kVg(t_{11}, t_{21}), \ldots, kVg(t_{1k}, t_{2k}))$ und

$$\begin{array}{rcll}
\sigma_i(kVg(t_1, t_2)) & = & f(\sigma_i(kVg(t_{11}, t_{21})), \ldots, \sigma_i(kVg(t_{1k}, t_{2k}))) & \\
& = & f(t_{i1}, \ldots, t_{ik}, t_{2k}) & Ind.Vor. \\
& = & t_i &
\end{array}$$

Sei jetzt ein Term s gegeben und Substitutionen τ_1, τ_2 mit $\tau_1(s) = t_1$ $\tau_2(s) = t_2$. Wir müssen eine Substitution ρ finden mit $\rho(s) = kVg(t_1, t_2)$. Wir setzen für jede in s vorkommende Variable X

$$\rho(X) = kVg(\tau_1(X), \tau_2(X))$$

Wir zeigen $\rho(s) = kVg(\tau_1(s), \tau_2(s))$ durch strukturelle Induktion über s. Ist s nur eine Variable, stimmt die Definition von ρ mit der zu beweisenden Aussage überein. Es bleibt der Fall $s = f(s_1, \ldots, s_n)$ zu betrachten.

$$\begin{array}{rcll}
\rho(f(s_1, \ldots, s_n)) & = & f(\rho(s_1), \ldots, \rho(s_n)) & \\
& = & f(kVg(\tau_1(s_1), \tau_2(s_1)) \ldots, kVg(\tau_1(s_n), \tau_2(s_n))) & Ind.Vor. \\
& = & kVg(f(\tau_1(s_1), \ldots, \tau_1(s_n)), f(\tau_2(s_1), \ldots, \tau_2(s_n))) & \\
& = & kVg(\tau_1(f(s_1, \ldots, s_n)), \tau_2(f(s_1, \ldots, s_n))) & \\
& = & kVg(\tau_1(s), \tau_2(s)) &
\end{array}$$

■

Aufgabe 4.2.8
Für jede Variable x gilt nach Voraussetzung $\tau(\sigma(x)) = x$. Da σ nach Korrolar 4.17 eine Variablenumbenennung ist, gibt es für jede Variable z eine Variable y mit $z = \sigma(y)$. Somit gilt also $\tau(z) = \tau(\sigma(y)) = y$. Somit bildet τ Variable wieder auf Variable ab. Es bleibt die Injektivität zu zeigen. Wir nehmen also $\tau(z_1) = \tau(z_2)$ an. Wir wählen y_i mit $z_i = \sigma(y_i)$. Es gilt also $\tau(z_i) = \tau(\sigma(y_i)) = y_i$. Nach Annahme also $y_1 = y_2$ und damit auch

$$z_1 = \sigma(y_1) = \sigma(y_2) = z_2$$

■

13.6 Lösungen zu Abschnitt 4.3 PL Semantik

Aufgabe 4.3.1
`fehlt noch`
Aufgabe 4.3.2
A sei die Konjunktion der folgenden Formeln:

$$\forall x \neg r(x,x)$$
$$\forall x \forall y \forall z (r(x,y) \land r(y,z) \to r(x,z))$$
$$\forall x \forall y (r(x,y) \to \exists z (r(x,z) \land r(z,y))$$
$$\exists x \exists y r(x,y)$$

Aufgabe 4.3.3

$$val_{\mathcal{D},\beta}(\forall x B) = W$$
$\Leftrightarrow$ Für alle $d \in D$: $val_{\mathcal{D},\beta_x^d}(B) = W$

$\Leftrightarrow$ Für alle $d \in D$: $val_{\mathcal{D},\gamma_x^d}(B) = W$

(Nach Induktionsannahme. Denn es ist $Frei(B) \subseteq Frei(\forall x B) \cup \{x\}$; nach Voraussetzung über β, γ und wegen $\beta_x^d(x) = d = \gamma_x^d(x)$ also $\beta_x^d(y) = \gamma_x^d(y)$ für alle $y \in Frei(B)$ und alle $d \in D$)

$\Leftrightarrow$ $val_{\mathcal{D},\gamma}(\forall x B) = W$.

Aufgabe 4.3.4
`fehlt noch`
Aufgabe 4.3.5
Diese Formalisierungsaufgabe geht über den Rahmen, den wir bisher betrachtet haben hinaus, indem wir die natürlichen Zahlen $(\mathbb{N}, +, <)$ als Datenstruktur benutzen. Wir erklären weiter unten die Einzelheiten. Neben den auf $\mathbb{N}$ vorhanden vertrauten Relationen und Funktionen benutzen wir ein zweistelligen Prädikat $d(x,y)$ mit der intendierten Bedeutung, daß $d(i,j)$ wahr ist, wenn auf dem Feld mit den Koordinaten (i,j) eine Dame positioniert ist.2

Zunächst präsentieren wir die Löung für den Fall $n = 8$. Zur Abkürzung benutzen wir die Notation

$$\forall_8 x\ A \Leftrightarrow \forall x(1 \leq x \leq 8 \to A) \exists_8 x\ A \Leftrightarrow \exists x(1 \leq x \leq 8 \land A)$$

Die gesuchte Formel q_8 besteht aus der Konkjunktion der folgenden Teilformeln:

1 $\forall_8 x \forall_8 y(d(x,y) \rightarrow \forall_8 z(z \neq x \rightarrow \neg q(z,y)))$
2 $\forall_8 x \forall_8 y(d(x,y) \rightarrow \forall_8 z(z \neq y \rightarrow \neg q(x,z)))$
3 $\forall_8 x(d(x,x) \rightarrow \forall_8 z(z \neq x \rightarrow \neg d(z,z)))$
4 $\forall x(1 \leq x \leq 7 \wedge d(x+1,x) \rightarrow \forall z(1 \leq x \leq 7 \wedge z \neq x \rightarrow \neg d(z+1,z))$
5 $\forall x(1 \leq x \leq 6 \wedge d(x+2,x) \rightarrow \forall z(1 \leq x \leq 6 \wedge z \neq x \rightarrow \neg d(z+2,z))$
$\vdots$
9 $\forall x(1 \leq x \leq 2 \wedge d(x+6,x) \rightarrow \forall z(1 \leq x \leq 6 \wedge z \neq x \rightarrow \neg d(z+6,z))$
10 $\forall x(1 \leq x \leq 7 \wedge d(x,x+1) \rightarrow \forall z(1 \leq x \leq 7 \wedge z \neq x \rightarrow \neg d(z,z+1)))$
11 $\forall x(1 \leq x \leq 6 \wedge d(x,x+2) \rightarrow \forall z(1 \leq z \leq 6 \wedge z \neq x \rightarrow \neg d(z,z+2)))$
$\vdots$
15 $\forall x(1 \leq x \leq 2 \wedge d(x,x+6) \rightarrow \forall z(1 \leq z \leq 2 \wedge z \neq x \rightarrow \neg d(z,z+6)))$
16 $\forall x(1 \leq x \leq 8 \wedge d(x,9-x) \rightarrow \forall z(1 \leq z \leq 8 \wedge z \neq x \rightarrow \neg d(z,9-z)))$
17 $\forall x(1 \leq x \leq 7 \wedge d(x+1,9-x) \rightarrow$
$\forall z(1 \leq z \leq 7 \wedge z \neq x \rightarrow \neg d(z+1,9-z)))$
$\vdots$
22 $\forall x(1 \leq x \leq 2 \wedge d(x+6,9-x) \rightarrow$
$\forall z(1 \leq z \leq 2 \wedge z \neq x \rightarrow \neg d(z+6,9-z)))$
23 $\forall x(1 \leq x \leq 7 \wedge d(x,8-x) \rightarrow$
$\forall z(1 \leq z \leq 7 \wedge z \neq x \rightarrow \neg d(z,8-z)))$
$\vdots$
28 $\forall x(1 \leq x \leq 2 \wedge d(x,3-x) \rightarrow$
$\forall z(1 \leq z \leq 2 \wedge z \neq x \rightarrow \neg d(z,3-z)))$
29 $\forall_8 x \exists_8 y d(x,y)$

Die Formel 1 besagt, daß in einer Reihe, in der eine Dame positioniert ist, keine weitere stehen kann. Formel 2 sagt dasselbe für Spalten. Formel 2 sagt, wenn eine Dame auf der aufsteigenden Hauptdiagonalen von $(1,1)$ nach $(8,8)$ steht, keine weitere auf dieser Diagonalen stehen kann. Die Formeln 3 bis 9 sagen dasselbe aus für die immer kürzer werdenen aufsteigenden Diagonalen unterhalb der Hauptdiagonalen. Insbesondere ist Formel 9 äquivalent zu $d(1,7) \rightarrow \neg d(2,8)$. Die Formeln 10 bis 15 sagen aus, daß auf den immer kürzer werdenden aufsteigenden Diagonalen oberhalb der Hauptdiagonale höchstens eine Dame stehen kann. Formel 16 wendet sich der absteigenden Hauptdiagonalen von links oben, Feld $(1,8)$, nach rechts unten, Feld $(8,1)$ zu. Die nachfolgenden Formeln bis 28 decken die weiteren kürzeren absteigenden Diagonalen ab. Die letzte Formel 28 schließlich sagt, daß in jeder Reihe mindestens eine Dame stehen soll.

Man kann die Formeln 3 bis 9 zusammenfassen zu

$$\begin{array}{l}\forall k(0 \leq k \leq 6 \rightarrow \\ \quad \forall x(1 \leq x \leq (8-k) \wedge d(x+k, x) \rightarrow \\ \quad \forall z(1 \leq z \leq (8-k) \wedge z \neq x \rightarrow \neg d(z+k, z))))\end{array}$$

Ebenso die Formeln 10 bis 15:

$$\begin{array}{l}\forall k(0 \leq k \leq 6 \rightarrow \\ \quad \forall x(1 \leq x \leq (8-k) \wedge d(x, x+k) \rightarrow \\ \quad \forall z(1 \leq z \leq (8-k) \wedge z \neq x \rightarrow \neg d(z, z+k))))\end{array}$$

Ebenso lassen sich die Formeln welche die absteigenden Diagonalen betreffen zusammenfassen zu:

$$\begin{array}{l}\forall k(0 \leq k \leq 6 \rightarrow \\ \quad \forall x(1 \leq x \leq (8-k) \wedge d(x+k, 9-x) \rightarrow \\ \quad \forall z(1 \leq z \leq (8-k) \wedge z \neq x \rightarrow \neg d(z+k, 9-z)))) \\ \forall k(0 \leq k \leq 6 \rightarrow \\ \quad \forall x(1 \leq x \leq (8-k) \wedge d(x, 9-k-x) \rightarrow \\ \quad \forall z(1 \leq z \leq (8-k) \wedge z \neq x \rightarrow \neg d(z+k, 9-k-z))))\end{array}$$

Die Definition der Erfüllbarkeit ist in dem hier betroffenen Fall dahingehend abzuändern, daß eine Teil der gesuchten Interpretation festgelegt ist. Das Universum sind die natürlichen Zahlen $\mathbb{N}$ und die arithmetischen Funktionen $+$ und $-$ und die Relation $<$ sind im üblichen Sinne festlegt. Nur die Interpretation der zweistelligen Relation $d(x, y)$ ist noch frei. In diesem Sinne ist die Konjunktion der oben beschriebenen Formeln genau dann erfüllbar, wenn das Acht-Dame Problem eine Lösung hat.

Die Verallgemeinerung auf beliebiges n anstelle von 8 ist jetzt leicht zu bewerkstelligen:

1 $\forall x \forall y(1 \leq x \leq n \wedge 1 \leq y \leq n \wedge d(x, y) \rightarrow$
$\forall z(1 \leq z \leq n \wedge z \neq x \rightarrow \neg q(z, y)))$
2 $\forall x \forall y(1 \leq x \leq n \wedge 1 \leq y \leq n \wedge d(x, y) \rightarrow$
$\forall z(1 \leq z \leq n \wedge z \neq x \rightarrow \neg q(x, z)))$
3 $\forall k(0 \leq k \leq (n-2) \rightarrow$
$\forall x(1 \leq x \leq (n-k) \wedge d(x+k, x) \rightarrow$
$\forall z(1 \leq z \leq (n-k) \wedge z \neq x \rightarrow \neg d(z+k, z))))$
4 $\forall k(0 \leq k \leq (n-2) \rightarrow$
$\forall x(1 \leq x \leq (n-k) \wedge d(x, x+k) \rightarrow$
$\forall z(1 \leq z \leq (n-k) \wedge z \neq x \rightarrow \neg d(z, z+k))))$
5 $\forall k(0 \leq k \leq (n-2) \rightarrow$
$\forall x(1 \leq x \leq (n-k) \wedge d(x+k, n-1-x) \rightarrow$
$\forall z(1 \leq z \leq (n-k) \wedge z \neq x \rightarrow \neg d(z+k, n-1-z))))$
6 $\forall k(0 \leq k \leq (n-2) \rightarrow$
$\forall x(1 \leq x \leq (n-k) \wedge d(x, n-1-k-x) \rightarrow$
$\forall z(1 \leq z \leq (n-k) \wedge z \neq x \rightarrow \neg d(z+k, n-1-k-z))))$
7 $\forall x(1 \leq x \leq n \rightarrow \exists y(1 \leq y \leq n \wedge d(x, y)))$

Aufgabe 4.3.6
Aufgabe 4.3.7
Die Formel $p(x) \wedge \exists x \neg p(x)$ ist zwar erfüllbar, hat aber kein Modell.

Aufgabe 4.3.8
$A := p(x),\ \ B := \forall x P(x)$.

Aufgabe 4.3.9

1. $$p(x) \models \forall x p(x), \quad p(x) \not\models^{\circ} \forall x p(x)$$
   ist ein Gegenbeispiel für die umgekehrte Implikation.

2. ******

3. Ein Gegenbeispiel ist etwa $p(x) \models \forall x p(x)$, $\not\models p(x) \rightarrow \forall x p(x)$.

4. ******

Aufgabe 4.3.12

1. $\phi_3 \quad = \quad \exists x_1 \exists x_2 \exists x_3(\quad x_1 \not\doteq x_2 \wedge x_1 \not\doteq x_3 \wedge x_2 \not\doteq x_3 \wedge$
$\forall y(\quad y \doteq x_1 \vee y \doteq x_2 \vee y \doteq x_3))$
Dabei steht $s \not\doteq t$ abkürzend für $\neg(s \doteq t)$.

2. $\phi_n = \exists x_1 \ldots \exists x_n(\bigwedge_{1\leq i<j\leq n} x_i \not\doteq x_j \wedge \forall y(\bigvee_{1\leq i\leq n} y \doteq x_i))$

Aufgabe 4.3.13

zu 1 Wir versuchen mit möglichst wenig nicht-logischem Vokabular auszukommen und benutzen nur ein zwei-stelliges Funktionszeichen f.

$$Sp_{even} \quad = \quad \forall x(f(f(x) = x) \wedge x \neq f(x))$$

Gilt $\mathcal{M} \models Sp_{even}$ dann definieren wir für jedes Element m aus dem Universum von $\mathcal{M}$ die Menge $M_m = \{m, f^{\mathcal{M}}(m)\}$. Wegen $\mathcal{M} \models Sp_{even}$ hat jede der Mengen M_m genau zwei Elemente. Außerdem gilt für $m_1 \neq m_2$ entweder $M_{m_1} = M_{m_2}$ oder $M_{m_1} \cap M_{m_2} = \emptyset$. Gilt nämlich $m_2 \in M_{m_1}$ dann gilt entweder $m_2 = m_1$ oder $m_2 = f^{\mathcal{M}}(m_1)$. Im ersten Fall gilt natürlich auch $M_{m_1} = M_{m_2}$ und im zweiten Fall folgt zunächst $f^{\mathcal{M}}(m_2) = f^{\mathcal{M}}(f^{\mathcal{M}}(m_1)) = m_1$ und damit auch wieder $M_{m_1} = M_{m_2}$. Ähnlich verläuft die Argumentation im Fall $f^{\mathcal{M}}[m_2) \in M_{m_1}$. Bezeichnen wir mit M das Universum von $\mathcal{M}$ dann gilt offensichtlich $M = \bigcup_{m\in M} M_m$ und M muß eine gerade Anzahl von Elementen haben.
Für jede gerade Zahl $n = 2k$ kann man leicht ein Modell $\mathcal{M}^n$ konstruieren das Sp_{even} erfüllt. Man kann z.B. als Universum von $\mathcal{M}^n$ die Zahlen von 1 bis n wählen und die Interpretation von f wie folgt festsetzen:

$$f^{\mathcal{M}^n}(i) = \begin{cases} k+i & \text{falls } 1 \leq i \leq k \\ i-k & \text{falls } k+1 \leq i \leq n \end{cases}$$

zu 2 Wir verzichten hier auf eine minimale Signatur und legen mehr Wert auf Anschaulichkeit. Die Grundidee ist eine Formel Sp_{quad} aufzuschreiben, die sicherstellt, daß in jedem Modell $\mathcal{M}$ von Sp_{quad} eine (virtuelle) x-Achse und eine y-Achse existieren gegeben durch die einstelligen Prädikate $X()$ und $Y()$. Für jedes Element von $\mathcal{M}$ wird die x-Koordinate durch die einstellige Funktion $x()$ und entsprechend die y-Koordinate durch die einstellige Funktion $y()$ gegeben. Für jedes Koordinatenpaar (n, m) mit $n \in X^{\mathcal{M}}$ und $m \in Y^{\mathcal{M}}$ gibt es genau ein Element e in $\mathcal{M}$ mit $x^{\mathcal{M}}(e) = n$ und $y^{\mathcal{M}}(e) = m$. Wenn wir jetzt noch sicherstellen, daß X- und Y-Achse gleich lang sind, dann ist die Anzahl der Element von $\mathcal{M}$ eine Quadratzahl. Dazu benutzen wir die binäre Relation R, die eine eindeutige Korrespondenz zwischen X und Y beschreibt. In Formeln gegossen ergibt das:

$$\begin{array}{lll}
1 & \forall e(X(x(e)) \wedge Y(y(e))) & \wedge \\
2 & \forall e_1, e_2((x(e_1) = x(e_2) \wedge y(e_1) = y(e_2)) \rightarrow e_1 = e_2 & \wedge \\
3 & \forall n \forall m(X(n) \wedge Y(m) \rightarrow \exists e(x(e) = n \wedge y(e) = m)) & \wedge \\
4 & \forall u_1, u_2, w((R(u_1, w) \wedge R(u_2, w)) \rightarrow u_1 = u_2) & \wedge \\
5 & \forall u_1, u_2, w((R(w, u_1) \wedge R(w, u_2)) \rightarrow u_1 = u_2) & \wedge \\
6 & \forall u(X(u) \rightarrow \exists w(Y(w) \wedge R(u, w))) & \wedge \\
7 & \forall w(Y(u) \rightarrow \exists u(X(u) \wedge R(u, w))) &
\end{array}$$

zu 3 Wir übernehmen das Vokabular aus der vorangegangenen Teilaufgabe ohne das Relationszeichen R und bezeichnen mit ψ die Konjunktion der Formeln 1 bis 4. Zunächst setzen wir

$$\begin{array}{lll}
Sp'_{nonprime} & = & \psi \qquad \wedge \\
 & & \exists x \exists y(X(x) \wedge X(y) \wedge x \neq y) \quad \wedge \\
 & & \exists x \exists y(Y(x) \wedge Y(y) \wedge x \neq y)
\end{array}$$

Mit denselben Überlegungen wie unter (2) sieht man ein, daß jedes Modell $\mathcal{M}$ von $Sp'_{nonprime}$ genau aus $n * m$ Elementen besteht, wobei n die Anzahl der Element in $X^{\mathcal{M}}$ ist und m die Anzahl der Element in $Y^{\mathcal{M}}$. $Sp'_{nonprime}$ sichert zudem $n > 1$ und $m > 1$. Da 1 und 2 auch als Nichtprimzahlen zählen schließen wir ab mit der Definition:

$$Sp_{nonprime} = Sp'_{nonprime} \vee \phi_2 \vee \phi_1$$

Für die Anzahlformeln ϕ_n siehe die obige Lösung zu Aufgabe 4.3.12.
Als ein Beispiel für kompliziertere Spektren erwähnen wir die Menge aller Primzahlpotenzen, die genau die Menge der endlichen Kardinalitäten ist für die es endliche Körper gibt, siehe [Lan72, Chapter VII, Section 5].

Aufgabe 4.3.14

$A = p(x)$ und $B = \forall x p(x)$ leisten das Gewünschte. Denn: $p(x) \models \forall x p(x)$, aber $\not\models p(x) \rightarrow \forall x p(x)$

Aufgabe 4.3.15

Für $a = 1$ und $b = 1073741821 = \frac{1}{2}(max\_int - 5)$ gilt $b *_{jint} b = -2147483639$ und $(b + 1) *_{jint} (b + 1) = 4$, also $\mathcal{Z}_{jint} \models \phi[a, b]$, aber b ist keine ganzzahlige Quadratwurzel von 1.

Aufgabe 4.3.16

Teil 1:

$$
\begin{array}{ll}
\forall n \forall d(& (n \geq 0 \land jdiv(n,d) = div(n,d)) \\
& \lor \\
& (n < 0 \land jdiv(n,d) = -div(-n,d)) \\
\quad) &
\end{array}
$$

Teil 2:
Der Beweis benutzt strukturelle Induktion nach der Komplexität von s. Wir fixieren eine beliebige Interpretation (D, I) und eine Variablenbelegung β und schreiben im folgenden val anstelle von $val_{D,I,\beta}$.

Ist s eine Variable, dann ist $s_1 = s_2 = s$ und die Behauptung ist offensichtlich wahr.

Ist $s = f(s^1, \ldots, s^k)$ dann gilt $s_i = f(s_i^1, \ldots, s_i^k)$ für $i = 1$ und $i = 2$, wobei, wie der Leser vielleicht schon erraten hat, s_1^j aus s^j entsteht durch die Ersetzung von occ mit t_1. Analog entsteht s_2^j. Die Auswertung $val(s)$ liefert zunächst: $val(s) = I(f)(val(s^1), \ldots, val(s^k))$. Im Fall $(D, I, \beta) \models \phi$ sagt die Induktionshypothese $val(s^j) = val(s_1^j)$. Daraus folgt weiter $val(s)$ $= I(f)(val(s^1), \ldots, val(s^k)) = I(f)(val(s_1^1), \ldots, val(s_1^k)) = val(s_1)$. Analog erhält man im Fall $(D, I, \beta) \models \neg\phi$ das gewünschte Resultat $val(s) = val(s_2)$.

Es bleibt der Fall $s = \texttt{if } \phi' \texttt{ then } s_1' \texttt{ else } s_2'$. Der Spezialfall, daß s selbst das Vorkommen occ ist , das ersetzt werden soll, daß also $\phi \equiv \phi'$, $s_1 = s_1'$ und $s_2 = s_2'$ gilt, ist unmittelbar klar. Wir nehmen daher im folgenden an, daß occ in s_1 oder s_2 vorkommt. Wir betrachten zuerst den Fall $(D, I, \beta) \models \phi$. Diese teilt sich natürlicherweise in zwei Unterfälle: Falls $(D, I, \beta) \models \phi'$ dann $val(s)$ $= val(s_1')$ und die Induktionsvoraussetzung liefert $val(s_1') = val((s_1')_1)$. Falls $(D, I, \beta) \models \neg\phi'$ dann $val(s) = val(s_2')$ und die Induktionsvoraussetzung hilft uns auf $val(s_2') = val((s_2')_1)$ zu schließen. Zusammengenommen haben wir $val(s) = val(s_1)$ gezeigt.

Der Fall $(D, I, \beta) \models \neg\phi$ folgt analog.

Teil 3:
Wir benutzen wieder Induktion nach dem Formelaufbau von F. Die beste Strategie die in jedem Induktionsschritt anfallenden Äquivalenzen zu beweisen ist die Fallunterscheidung $(D, I, \beta) \models \phi$ oder $(D, I, \beta) \models \neg\phi$. Wir werden im folgenden Schreibarbeit sparen und $(\mathcal{D}, \beta)$ schreiben anstelle von (D, I, β) und val anstelle von $val_{D,I,\beta}$.
Für den einfachsten Fall $F \equiv r(s_1, \ldots, s_k)$ für ein Relationssymbol r ergibt sich die Beweisverpflichtung:

$$
\begin{array}{ll}
(\mathcal{D}, \beta) \models r(s_1, \ldots, s_k) \leftrightarrow r(s_{1,1}, \ldots, s_{k,1}) & \text{falls } (\mathcal{D}, \beta) \models \phi \\
(\mathcal{D}, \beta) \models r(s_1, \ldots, s_k) \leftrightarrow r(s_{1,2}, \ldots, s_{k,2}) & \text{falls } (\mathcal{D}, \beta) \models \neg\phi
\end{array}
$$

Aus Teil (2) des Beweises wissen wir:

$$\begin{array}{ll} val(s_i) = val(s_{i,1}) & \text{falls } (\mathcal{D}, \beta) \models \phi \\ val(s_i) = val(s_{i,2}) & \text{falls } (\mathcal{D}, \beta) \models \neg\phi \end{array}$$

und wir sind mit diesem Fall fertig.

Wenn wir aus der Induktionsvoraussetzung im Fall $(\mathcal{D}, \beta) \models \phi$ wissen, daß $(\mathcal{D}, \beta) \models F_i \leftrightarrow F_{i,1}$ dann folgt daraus natürlich auch: $(\mathcal{D}, \beta) \models (F_1 \wedge F_2) \leftrightarrow (F_{1,1} \wedge F_{2,1})$, $(\mathcal{D}, \beta) \models \neg F_1 \leftrightarrow \neg F_{1,1}$, und $(\mathcal{D}, \beta) \models (F_1 \vee F_2) \leftrightarrow (F_{1,1} \vee F_{2,1})$. Dasselbe Argument gilt auch im Fall $(\mathcal{D}, \beta) \models \neg\phi$, wobei der Index 1 durch Index 2 ersetzt werde muß. Zusammengenommen hat man damit gezeigt $(F_1 \wedge F_2) \leftrightarrow (\phi \wedge F_{1,1} \wedge F_{2,1}) \vee (\neg\phi \wedge F_{1,2} \wedge F_{2,2})$. Ebenso für die anderen aussagenlogischen Verknüpfungen.
Teil 4: Eine beliebige Formel F wird zuerst in eine äquivalente pränexe Normalform transformiert $Q_1 x_1 \dots Q_k x_k F_0$, mit $Q_i \in \{\forall, \exists\}$. Für die quantorenfreie Formel F_0 wiederholen wir für jedes Vorkommen *occ* eines bedingten Terms die Konstruktion aus Teil (3). und erhalten eine Formel F_1 ohne bedingte Terme mit

$$\forall x_1 \dots \forall x_k (F_0 \leftrightarrow F_1)$$

Daraus folgt auch

$$Q_1 x_1 \dots Q_k x_k F_0 \leftrightarrow Q_1 x_1 \dots Q_k x_k F_1$$

■

Aufgabe 4.3.17
Wir betrachten die Formel $F = (p(x) \wedge q(y)) \vee (\neg p(x) \wedge \neg q(y))$.
In der Struktur $\mathcal{M}$ mit zwei Elementen $M = \{a, b\}$ und $I^{\mathcal{M}}(p) = I^{\mathcal{M}}(q) = \{a\}$ gilt $\mathcal{M} \models \forall x \exists y F$ aber nicht $\mathcal{M} \models \exists y \forall x F$.

13.7 Lösungen zu Abschnitt 4.4 PL Normalformen

Aufgabe 4.4.6
Die Antwort ist nein. Die Gültigkeit der in Frage stehenden Aussage hängt sehr stark vom benutzten Vokabular ab. Betrachten wir z.B. das Vokabular Σ, das einzig und allein ein Konstantensymbol c entält. Das einzige Prädikatszeichen ist also die Gleichheit $\doteq$. Die Formel $\exists x(\neg x \doteq c)$ ist sicherlich erfüllbar. Es gibt aber keinen Term, der ein solches x bezeichnen könnte.

13.8 Lösungen zu Abschnitt 5.2 Hilbertkalkül

Aufgabe 5.2.1 TBD

Aufgabe 5.2.2 TBD

Aufgabe 5.2.3
Teil 1 Sei $\models$ die in Definition 2.17 eingeführte logische Folgerungsbeziehung. Wir zeigen durch strukturelle Induktion über den Aufbau eines Lambdaterms M vom Typ τ, daß $TFV(M) \models \tau$ gilt.
Hierbei wird τ als aussagenlogische Formel aufgefasst und $TFV(M)$ ist die Menge der Typen der Variablen in $FV(M)$.

1. Induktionsanfang: M ist die aussagenlogische Variable x vom Typ a. Die Behauptung reduziert sich in diesem Fall auf die triviale Aussage $a \models a$.

2. 1.Induktionsschritt: $M = \lambda x.M_2$
   Sei τ_1 der Typ von x und τ_2 der Typ von M_2. Dann ist $\tau_1 \to \tau_2$ der Typ von M.
   Sei weiterhin $\{x, x_1, \ldots, x_k\} = FV(M_1)$ und seien $\tau_1, a_1, \ldots, a_k$ die zugehörigen Typen in der angegebenen Reihenfolge. Nach Induktionsvoraussetzung gilt $\{\tau_1, a_1, \ldots, a_k\} \models \tau_2$. Mit dem Deduktionstheorem folgt daraus $\{a_1, \ldots, a_k\} \models \tau_1 \to \tau_2$, was zu zeigen war.

3. 2.Induktionsschritt: $M = M_2 N_2$
   Es sei $\tau_1 \to \tau_2$ der Typ von M_2. Dann muß der Typ von N_2 nach Definition τ_1 sein und M hat den Typ τ_2.
   Seien $FV(M_2) = \{x_1, \ldots, x_k\}$ und $FV(n_2) = \{y_1, \ldots, y_r\}$. Somit $VF(M) = FV(M_2) \cup FV(N_2)$. Es seien a_i der Typ von x_i und b_j der Typ von y_j.
   Nach Induktionsvorausetzung gilt
   $\{a_1, \ldots, a_k\} \models \tau_1 \to \tau_2$
   $\{b_1, \ldots, b_k\} \models \tau_1$
   Somit gilt umso mehr
   $\{a_1, \ldots, a_k, b_1, \ldots, b_k\} \models \tau_1 \to \tau_2$
   $\{a_1, \ldots, a_k, b_1, \ldots, b_k\} \models \tau_1$
   und damit auch
   $$\{a_1, \ldots, a_k, b_1, \ldots, b_k\} \models \tau_2$$

Teil 2 Für Ax1, das ist zur Erinnerung $a \rightarrow (b \rightarrow a)$, betrachten wir den Lambdaterm $M_1 = \lambda x.\lambda y.x$ wobei x vom Typ a und y vom Typ b ist. Der Typ vonM_1 ist dann, wie gewünscht, $a \rightarrow (b \rightarrow a)$.

Für $Ax2$, das ist $(a \rightarrow (b \rightarrow c)) \rightarrow ((a \rightarrow b) \rightarrow (a \rightarrow c))$, betrachten wir den Lambdaterm $M_2 = \lambda x.\lambda y.\lambda z.xz(yz)$ mit folgender Typisierung

x	hat Typ	$a \rightarrow (b \rightarrow c)$
y	hat Typ	$a \rightarrow b$
z	hat Typ	a

Dann gilt

1. Der Term yz ist typgerecht konstruiert und hat selbst den Typ b.
2. Der Term xz ist typgerecht konstruiert und hat selbst den Typ $b \rightarrow c$.
3. $.xz(yz)$ ist typgerecht konstruiert und hat selbst den Typ c.
4. Somit hat $\lambda z.xz(yz)$ den Typ $a \rightarrow c$.
5. weiter hat $\lambda y.\lambda z.xz(yz)$ den Typ $(a \rightarrow b) \rightarrow (a \rightarrow c)$ und
6. schließlich hat M_2 den Typ $(a \rightarrow (b \rightarrow c)) \rightarrow ((a \rightarrow b) \rightarrow (a \rightarrow c))$

Teil 3
$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$
Wir betrachten Variablen

Variablenname	Typ
x	$A \rightarrow B$
y	$B \rightarrow C$
z	A

Wir betrachten den Lambdaterm $N = \lambda x.\lambda y.\lambda z.y(xz)$.
Berechnung des Typs von N:

Term	Typ
z	A
xz	B
$y(xz)$	C
$\lambda z.y(xz)$	$A \rightarrow C$
$\lambda y.\lambda z.y(xz)$	$(B \rightarrow C) \rightarrow (A \rightarrow C)$
N	$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$

$(A \to B) \to C) \to (A \to (B \to C))$
Wir betrachten Variablen

Variablenname	Typ
x	$(A \to B) \to C$
y	A
z	B
u	A

Wir betrachten den Lambdaterm $M = \lambda x.\lambda y.\lambda z.(x(\lambda u.z))$.
Berechnung des Typs von M:

Term	Typ
z	B
$\lambda u.z$	$A \to B$
$x.\lambda u.z$	C
$\lambda z.(x.\lambda u.z)$	$B \to C$
$\lambda y.\lambda z.(x.\lambda u.z)$	$A \to (B \to C)$
M	$((A \to B) \to C) \to (A \to (B \to C))$

13.9 Lösungen zu Abschnitt 5.3 Resolution

Aufgabe 5.3.1

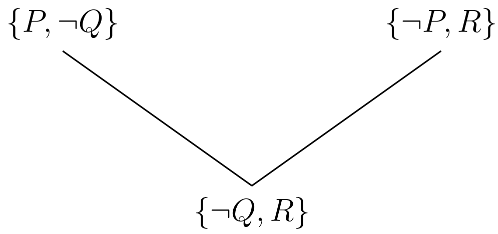
```
1 [] -p1h1 v -p2h1.
2 [] -p1h1 v -p3h1.
3 [] -p2h1 v -p3h1.
4 [] -p1h2 v -p2h2.
5 [] -p1h2 v -p3h2.
6 [] -p2h2 v -p3h2.
7 [] p1h1 v p1h2.
8 [] p2h1 v p2h2.
9 [] p3h1 v p3h2.
10 [7,2] p1h2 v -p3h1.
12 [7,5] p1h1 v -p3h2.
14 [8,3] p2h2 v -p3h1.
15 [8,1] p2h2 v -p1h1.
20 [9,6] p3h1 v -p2h2.
23 [10,4] -p3h1 v -p2h2.
49 [23,20] -p2h2.
50 [23,14] -p3h1.
51 [49,15] -p1h1.
54 [50,9] p3h2.
55 [51,12] -p3h2.
56 [55,54] .
----> UNIT CONFLICT .
```

Aufgabe 5.3.2 Die Formel $P \leftrightarrow Q$ ist sicherlich nicht unerfüllbar. Bringt man sie in Klauselnormalform entstehen die beiden Klauseln

$$\{\neg P, Q\} \quad \text{und} \quad \{P, \neg Q\}$$

Mit der vorgeschlagenen Regel könnte man daraus unmittelbar die leere Klausel ableiten. Also kann diese Regel nicht korrekt sein.

Aufgabe 5.3.3 Eine Krom-Formel ist eine Formel in KNF, wo jede Disjunktion höchstens zwei Literale enthält, d.h. eine Krom-Formel läßt sich als Klauselmenge schreiben, bei der jede Klausel maximal zwei Literale hat. Bei der Resolution entstehen aus solchen Klauseln immer nur wieder solche:



Insgesamt können höchstens $O(n^2)$ $[4n^2]$ Klauseln erzeugt werden. Spätestens nach $O(n^2)$ Schritten ist klar, ob $\Box$ erzeugbar ist oder nicht.

Aufgabe 5.3.4 Im Beweis des Vollständigkeitssatzes 5.14 von Seite 162 wird nur einmal davon Gebrauch gemacht, daß die Klauselmenge M_0 unter Resolventenbildung abgeschlossen ist. In der Notation des Beweises ist das die Resolution der beiden Formeln $D_1 \cup \{P_n\}$ und $C = C_1 \cup \{\neg P_n\}$, wobei alle Literale in $D_1 \cup C_1$ echt kleineren Index haben als n.

Aufgabe 5.3.5 Wir zeigen $val_I(C) = W$ für alle $C \in M_0$ durch Induktion über die Anzahl k der positiven Literale in C. Für den Induktionsanfang betrachten wir eine Klausel $C_0 \cup \{\neg P_n\}$ mit nur negativen Literalen, wobei alle Literale in C_0 einen kleineren Index als n haben. Falls $val_I(C_0) = W$ gilt so auch $val_I(C_0 \cup \{\neg P_n\}) = W$. Gilt $val_I(C_0) = F$ so erzwingt die Definition von I, daß $I(P_n) = F$ und damit ebenfalls $val_I(C_0 \cup \{\neg P_n\}) = W$ gilt.
Im Induktionsschritt betrachten wir eine Klausel $D \cup \{P_n\}$ mit $k > 0$ positiven Literalen und nehmen an, daß für alle Klauseln aus M_0 mit weniger als k vielen positiven Literalen $val_I(C) = W$ schon gezeigt ist. Gilt $I(P_n) = W$, so auch $val_I(D \cup \{P_n\}) = W$ und wir sind fertig. Gilt $I(P_n) = F$ dann gibt es nach Definition von I eine Klausel $C_0 \cup \{\neg P_n\}$ in M_0 mit nur negativen Literalen und $val_I(C_0) = F$. Da M_0 unter Resolventenbildung abgeschlossen ist, liegt auch die Resolvente $D \cup C_0$ von $D \cup \{P_n\}$ und $C_0 \cup \{\neg P_n\}$ in M_0. Da $D \cup C_0$ strikt weniger positive Literale als k enthält gilt nach Induktionsvoraussetzung $val_I(D \cup C_0) = W$. Wegen $val_I(C_0) = F$ folgt daraus $val_I(D) = W$ und wir sind fertig.

■

Aufgabe 5.3.6 Die Inspektion des Beweises zu Übungsaufgabe 5.3.5 zeigt, daß dort nur negative Resolutionsschritte erforderlich sind.

Aufgabe 5.3.7 Die Vollständigkeit kann nicht verloren gehen, da in der modifizierten Version mehr Resolventen gebildet werden können als vorher. Der Korrektheitsbeweis macht nicht Gebrauch von der Variablendisjunktheit. Der modifizierte Kalkül ist also weiterhin korrekt und vollständig. Er ist allerdings ineffizienter, da es mehr Möglichkeiten gibt eine Resolvente mehrfach auf verschiedenen Wegen herzuleiten.

Aufgabe 5.3.8 Wegen $Res'(M) \subseteq Res(M)$ ist der modifizierte Kalkül auf jeden Fall noch korrekt. Er ist aber nicht mehr vollständig. Die Formelmenge $\{\forall x(p(x, f(x))), \neg\exists x(p(a, x))\}$ ist sicherlich unerfüllbar. Nach Umwandlung in Klauselnormalform erhalten wir $\{\{p(x, f(x))\}, \{\neg p(a, x)\}\}$. Die beiden Einerklauseln $\{p(x, f(x))\}$, $\{\neg p(a, x)\}$ sind allerdings nicht resolvierbar, da die Unifikation von x mit $f(x)$ in der zweiten Argumentstelle von p nicht möglich ist. Dagegen liefert die Resolution von $\{p(x, f(x))\}$ mit $\{\neg p(a, y)\}$ sofort die leere Klausel.

13.10 Lösungen zu Abschnitt 5.4 AL Tableaux

Aufgabe 5.4.1
(1) Wir betrachten ein Tableau für die Formel $0(B \rightarrow A)$ mit $A = R \rightarrow (P \wedge Q)$ und $B = (P \wedge Q) \wedge R)$. Die auf den Wurzelknoten folgenden Knoten sind markiert mit $0(R \rightarrow (P \wedge Q))$ und $1((P \wedge Q) \wedge R)$. Jedes geschlossene Tableau (es gibt in diesem Fall nur eines) ist nicht regulär, R kommt zweimal vor.
(2) Die Lösung ist verblüffend einfach. Man modifiziert die Definition der Anwendung z.B. der α-Regel: Kommt eine α-Formel V auf einem Pfad π vor, so wird für jede der beiden Formeln V_i, die nicht bereits auf π vorkommt, der Pfad um einen mit V_i markierten Knoten verlängert. Entsprechend wir die β Regel modifiziert.

Aufgabe 5.4.2
(1)
$$\begin{array}{c|c} \multicolumn{2}{c}{1A \leftrightarrow B} \\ \hline 1A & 0A \\ 1B & 0B \end{array} \qquad \begin{array}{c|c} \multicolumn{2}{c}{0A \leftrightarrow B} \\ \hline 1A & 0A \\ 0B & 1B \end{array}$$

(2)
$$\begin{array}{c|c} \multicolumn{2}{c}{1sh(A, B, C)} \\ \hline 1A & 0A \\ 1C & 1B \end{array} \qquad \begin{array}{c|c} \multicolumn{2}{c}{0sh(A, B, C)} \\ \hline 1A & 0A \\ 0C & 0B \end{array}$$

Aufgabe 5.4.6
Sei T_0 ein Tableau und π_0 ein Pfad von T_0. Das Tableau T_1 entstehe aus T_0 durch Anwendung der β^+-Regel auf π_0. Es gibt also, z.B. eine Formel der Form $1(A \vee B)$ auf π_0 und T_1 entsteht in dem π_0 verlängert wird einnmal zu π_1, das ist π_0 verlängert um einen mit $1A$ markierten Knoten und zu anderen zu π_2, das ist π_0 verlängert um zwei Knoten, makiert mit $1B$ bzw. $0A$. Sei außerdem I eine erfüllende Interpretation für T_0, d.h. $I(T_0) = W$. Wir zeigen, daß dann auch $I(T_1) = W$ gilt. Es genügt den Fall $I(\pi_0) = W$ zu betrachten. Wir wissen dann jedenfalls $I(A \vee B) = W$. Gilt $I(A) = W$, dann auch $I(\pi_1) = W$ und damit auch $I(T_1) = W$. Andernfalls gilt $I(A) = F$ und damit nach Voraussetzung $I(B) = W$. Insgesamt also $I(\pi_2) = W$ und damit wieder $I(T_2) = W$.
Aufgabe 5.4.7
(1) Wir geben hier nur den Kern des Korrektheitsarguments wieder; der Rest läuft wie im Beweis von Satz 5.38. Sei I eine erfüllende Interpretation für einen Tableaupfad π und π_1, π_2 die durch die Anwendung der Schnittregel anstandenen beiden Fortsetzungspfade, dann gilt $I(\pi_1) = W$ oder $I(\pi_2) = W$. Das ist offensichtlich. Sei jetzt in derselben Ausgangssituation π_1 die durch eine Anwendung der β-Regel entstandene Fortsetzung von π. Liegen $1(F_1 \vee F_2)$

und $0F_1$ auf π, dann gilt $I((F_1 \vee F_2) = W$ und $I(F_1) = F$. Woraus $I(F_2) = W$ folgt und damit $I(\pi_1) = W$. Dieses Argument funktioniert natürlich auch für alle weiterern β-Formeln.

(2) Wir beschränken uns wieder auf den Kern des Arguments. Sei dazu π ein offener Pfad in einem erschöpften Tableau T. Wir definieren eine Interpretation I durch

$$I(P) = \begin{cases} W & \text{falls } 1P \in \pi \\ F & \text{falls } 0P \in \pi \\ \textit{beliebig} & \text{sonst} \end{cases}$$

Da der offene Pfad π nicht $1P$ und $0P$ gleichzeitig enthalten kann, ist die Definition von I möglich. Es bleibt durch Induktion über die Komplexität von $F \in \pi$ zu zeigen, daß $I(F) = W$ gilt. Der Rest des Arguments funktioniert wie im Beweis von Satz 5.38. Ist F eine atomare Vorzeichenformel, dann folgt die Behauptung direkt aus der Definition von I. Sei $1(F_1 \vee F_2) \in \pi$. Aus der Erschöpftheit von T folgt, daß die Schnittregel für $1(F_1 \vee F_2)$ angewendet wurde und $1F_1\pi$ oder $0F_1\pi$ gilt. Im ersten Fall folgt aus der Induktionsvoraussetzung $I(F_1) = W$ und damit auch $I(F_1 \vee F_2) = W$. Im zweiten Fall können wir wieder wegen der Erschöpftheit von T voraussetzen, daß die zugehörige β-Regel angewendet wurde und daher $1F_2 \in \pi$ gilt. Daraus folgt jetzt wie eben $I(F_1 \vee F_2) = W$.

13.11 Lösungen zu Abschnitt 5.5 PL Tableau

Aufgabe 5.5.1
TBD

Aufgabe 5.5.2

1. Es gibt ein geschlossenes Tableau für $p(x)$ über $\{p(a)\}$ aber $\{p(a)\} \not\models p(x)$
2. Es gilt $\{p(0), p(x) \rightarrow p(s(x))\} \models p(s(s(0)))$. Aber es gibt kein geschlossenes Tableau für $p(s(s(0)))$ über $\{p(0), p(x) \rightarrow p(s(x))\}$.

Aufgabe 5.5.3

1. $(\forall x p(x)) \rightarrow \exists x p(x)$ ist allgemeingültig in der üblichen Semantik, in der Interpretation mit leerer Trägermenge aber falsch.
2. Die Menge der erlaubten Interpretation ist für die Nullsemantik (um genau ein Element) größer. Also ist jede allgemeingültige Formel in der Nullsemantik auch allgemeingültig im bisherigen Sinne. Der unveränderte Tableaukalkül ist also weiterhin vollständig. Er ist aber für die Nullsemantik nicht mehr korrekt. Das Tableau mit der Wurzelmarkierung $0\forall x p(x) \rightarrow \exists x p(x)$ (siehe nächsten Aufzählungspunkt) kann geschlossen werden, aber $\forall x p(x) \rightarrow \exists x p(x)$ ist keine Tautologie in der Nullsemantik.
3. die γ-Regel von Seite 191 muß eingechränkt werden zu:
   γ^0-Regel Die γ-Regel, darf auf einem Pfad π nur angewendet werden, wenn
   (a) in den Formeln von π mindestens ein Konstantensymbol vorkommt
   oder
   (b) auf dem Pfad π einmal eine δ-Regel auf eine Formel ohne freie Variablen angewendet wurde.

Das folgende Tableau kann nach dieser Regel nicht geschlossen werden:

$$\begin{array}{ll} (1) & 0\forall x p(x) \rightarrow \exists x p(x) \\ (2) & 1\forall x p(x) \\ (3) & 0\exists x p(x) \\ (4) & 1p(X) \\ (5) & 0p(Y) \end{array}$$

Die Anwendungen der γ-Regeln in den Zeilen 2 und 3 erfüllt nicht die Einschränkungen der γ^0-Regel. Der ursprüngliche Kalkül war bezgl. der Nullsemantik vollständig aber nicht korrekt. Für den Kalkül mit der modifizierten γ-Regel bleibt nur die Vollständigkeit erneut zu beweisen. Wir betrachten dazu einen offenen Pfad π eines erschöpften Tableaus über $M \cup \{\neg A\}$. Wir beschränken uns auf den Fall, daß auf π kein Konstantenzeichen vorkommt und keine δ-Regel auf eine variablenfreie Formel angewendet wurde; anderenfalls gibt es keine Änderung gegenüber dem vorigen Beweis von Seite 201. Wir wollen zeigen, daß es eine Struktur $\mathcal{D}_0$ mit leerem Universum gibt, die alle Formeln, die auf π vorkommen erfüllt. Es mag zunächst überraschen, daß es mehr als eine Struktur $\mathcal{D}$ mit leerem Universum geben kann. Das liegt daran, daß wir in Definition 4.2 auch null-stellige Prädikatenzeichen p zugelassen haben. Für diese setzen wir:

$$\mathcal{D}_0 \models p \quad \Leftrightarrow \quad 1p \text{ kommt auf } \pi \text{ vor}$$

Da keine γ-Regel angewendet werden darf kommt auf π keine freie Variable X vor. Außderdem kann keine δ-Formel vorkommen. Anderenfalls würde durch die Anwendung der δ-Regel in Abwesenheit von freien Variablen ein Konstantensymbol eingeführt, was es aber auf π nicht geben soll. Hier gibt es noch den Spezialfall zu einer δ-Formel $\exists x F$ zu betrachten, in dem die Variable x in F nicht vorkommt. Die Anwendung der δ-Regel würde in diesem Fall nicht dazu führen, daß ein Konstantensymbol auf π vorkommt. Um diesen Fall abzudecken wurde der zweite Teil der Einschränkung für die Anwendung der γ-Regel eingeführt. Für den vorliegenden offenen Pfad π konnten wir daher annehmen, daß keine δ-Regel auf eine variablenfreie Formel angewendet wurde (andernfalls unterliegt die Anwendung der γ-Regel keiner Einschränkung und der Beweis von Seite 201 funktioniert wieder). Schließlich sind die einzigen auf π vorkommenden atomaren Formeln null-stellig, denn Konstantensymbole oder Variable, welche die Argumentpositionen besetzen könnten, stehen nicht zur Verfügung. Somit ist jede Formel auf π entweder eine atomare Formel, eine mit einem Allquantor beginnende Formel oder eine aussagenlogische Kombination von auf π vorkommenden Formeln. Was die atomaren Formeln F angeht, so wurde $\mathcal{D}_0$ so definiert, daß $\mathcal{D}_0 \models F$ gilt. Die mit einem universellen Quantor beginnenden Formeln sind in $\mathcal{D}_0$ wahr wegen des leeren Universums. Der induktive Beweis, daß für jede auf π vorkommende Formel F gilt $\mathcal{D}_0 \models F$ kann nun unverändert aus dem aussagenlogischen Beweis von Satz 5.38 von Seite 183 übernommen werden.

Aufgabe 5.5.4

	01[V] 1 $\forall t(a(next(t)) \leftrightarrow b(t) \wedge \neg reset(t))$		
	02[V] 0 $\forall t(reset(t) \rightarrow \neg a(next(t)))$		
	03[2] 0 $reset(c) \rightarrow \neg a(next(c))$		
	04[3] 1 $reset(c)$		
	05[3] 0 $\neg a(next(c))$		
	06[5] 1 $a(next(c))$		
	07[1] 1 $a(next(X)) \leftrightarrow b(X) \wedge \neg reset(X)$		
08[07]	1 $a(next(X))$	09[07]	0 $a(next(X))$
10[07]	1 $b(X) \wedge \neg reset(X)$	[06, 09] closed	
11[10]	1 $b(X)$		
12[10]	1 $\neg reset(X)$		
13[12]	0 $reset(X)$		
[04, 13] closed by $\sigma(X) = c$			

Aufgabe 5.5.5
Aus $\forall x(\neg\neg x = x)$ folgt $\forall x, y \neg x = \neg y \rightarrow x = y$ und damit ist nach Lemma 5.95 $\mathcal{R}$ eine Boolesche Algebra.

13.12 Lösungen zu Abschnitt 5.6 Sequenzenkalkül

Aufgabe 5.6.1
`to be done`

Aufgabe 5.6.2
`to be done`

Aufgabe 5.6.3
zu (1)

$$\begin{array}{cr}
F => F & \text{Axiom} \\ \hline
F => F \vee G & \text{LK-or-right-1} \\ \hline
F \wedge G => F \vee G & \text{LK-and-left-1}
\end{array}$$

zu (2)

$$\begin{array}{ccr}
F => F & G => G & \text{Axiome} \\ \hline
F \wedge G => F & F \wedge G => G & \text{LK-and-left-1/2} \\ \hline
\multicolumn{2}{c}{F \wedge G => F \wedge G} & \text{and-right}
\end{array}$$

zu (3)

$$\begin{array}{cr}
F => F & \text{Axiom} \\ \hline
F, \neg F => & \text{not-left} \\ \hline
F, F \wedge \neg F => & \text{LK-and-left-2} \\ \hline
F \wedge \neg F, F \wedge \neg F => & \text{LK-and-left-1} \\ \hline
F \wedge \neg F => & \text{Verdopplung}
\end{array}$$

13.13 Lösungen zum Kapitel 6 Gleichheitslogik

Aufgabe 6.4.1 Da $\leftrightarrow_E$ die reflexive, transitive, symmetrische Hülle von $\rightarrow_E$ ist genügt es folgendes zu beweisen:

1. $E \models u \doteq u$ für jeden Term u
2. Aus $E \models s \doteq t$ und $s \rightarrow_E s'$ folgt $E \models s' \doteq t$
3. Aus $E \models s \doteq t$ und $s' \rightarrow_E s$ folgt $E \models s' \doteq t$

(1.) ist wegen der Reflexivität der Glkeichheit offensichtlich.

(2.) Für jede Struktur $\mathcal{M}$ mit $\mathcal{M} \models E$ und jede Variablenbelegung β ist $(s')^{\mathcal{M},\beta} = t^{\mathcal{M},\beta}$, wobei $s^{\mathcal{M},\beta} = t^{\mathcal{M},\beta}$ vorausgesetzt ist. Es genügt also $s^{\mathcal{M},\beta} = (s')^{\mathcal{M},\beta}$ nachzuweisen. Nach Definition von $s \rightarrow_E s'$ gibt es eine Substitution σ und eine Gleichung $l \doteq r$ in E mit

1. $\sigma(l)$ ist ein Teilterm von s
2. s' entsteht aus s, indem $\sigma(l)$ durch $\sigma(r)$ ersetzt wird.

Die Gleichheit $s^{\mathcal{M},\beta} = (s')^{\mathcal{M},\beta}$ folgt somit, wenn $\sigma(l)^{\mathcal{M},\beta} = \sigma(r)^{\mathcal{M},\beta}$ gilt. Es gilt aber $\sigma(l)^{\mathcal{M},\beta} = l^{\mathcal{M},\beta'}$ für die Variablenbelegung β' definiert durch $\beta'(x) = \sigma(x)^{\mathcal{M},\beta}$, siehe Lemma 4.34. Ebenso gilt $\sigma(r)^{\mathcal{M},\beta} = r^{\mathcal{M},\beta'}$. Letztlich gilt $l^{\mathcal{M},\beta'} = r^{\mathcal{M},\beta'}$, da $\mathcal{M} \models \forall \bar{x}(l \doteq r)$ gilt.

Der Nachweis von (3.) ist analog zu (2) wegen der Symmetrie der Gleichheitrelation.

Aufgabe 6.4.2
(1): Da $\leftrightarrow_E$ die reflexive, transitive, symmetrische Hülle einer anderen Relation ist, ist $\leftrightarrow_E$ auf jeden Fall eine Äquivalenzrelation.
(2): Es bleibt die Kongruenzeigenschaft zu zeigen. Wir weisen diese für ein einstelliges Funktionzeichen f nach. Die Verallgemeinerung für n-stellige Funktionszeichen sollte daraus ersichtlich werden. Aus der Annahme $s_1 \leftrightarrow_E t_1$ müssen wir auf $f(s_1) \leftrightarrow_E f(t_1)$ schließen. Aus der Annahme $s_1 \leftrightarrow_E t_1$ folgt die Existenz von Termen $s_2, \ldots, s_n$ mit $s_n = t_1$ und für alle $1 \leq i < n$ gilt $s_i \rightarrow^1_E s_{i+1}$ oder $s_{i+1} \rightarrow^1_E s_i$. Gelingt es zu zeigen, daß für beliebige Terme u, v aus $u \rightarrow^1_E v$ auch $f(u) \rightarrow^1_E f(v)$ folgt, so sind wir fertig. Das ist aber einfach einzusehen. Aus $u \rightarrow^1_E v$ folgt die Existenz einer Gleichung $l \doteq r$ in

E und einer Substitution σ, so daß $\sigma(l)$ ein Teilterm von u ist und v durch Ersetzung von $\sigma(l)$ durch $\sigma(r)$ entsteht. Dann ist aber $\sigma(l)$ auch ein Teilterm von $f(u)$ und $f(v)$ entsteht aus $f(u)$ durch Ersetzung von $\sigma(l)$ durch $\sigma(r)$.
(3): Es genügt wieder zu zeigen, daß aus $s \rightarrow_E^1 t$ die Relation $\mu(s) \rightarrow_E^1 (t)$ folgt. Aus $s \rightarrow_E^1 t$ folgt die Existenz einer Gleichung $l \doteq r$ in E und einer Substitution σ mit

1. $\sigma(l)$ ist ein Teilterm von s
2. t entsteht durch Ersetzung von $\sigma(l)$ durch $\sigma(r)$.

Dann gilt aber auch

1. $\mu(\sigma(l))$ ist ein Teilterm von $\mu(s)$
2. $\mu(t)$ entsteht durch Ersetzung von $\mu(\sigma(l))$ durch $\mu(\sigma(r))$.

Das ist aber die Definition von $\mu(s) \rightarrow_E^1 (t)$.
(4): Der Beweis benutzt, wie kaum anders zu erwarten, Induktion nach dem Aufbau des Termes s. Der Anfangsfall, s ist eine Variable x, ist trivial. Für den Induktionsschritt $s = f(s_1, \ldots, s_n)$ berechnet man

$$\begin{array}{rcll} \mu_1(s) & = & \mu_1(f(s_1, \ldots, s_n)) & \\ & = & f(\mu_1(s_1), \ldots, \mu_1(s_n)) & \text{Eigenschaft von Substitutionen} \\ & = & f(\mu_2(s_1), \ldots, \mu_2(s_n)) & \text{Induktionsvoraussetzung} \\ & = & \mu_2(f(s_1, \ldots, s_n)) & \text{Eigenschaft von Substitutionen} \end{array}$$

Aufgabe 6.4.3
(1): Der Beweis wird durch strukturelle Induktion über den Aufbau des Termes t geführt.
Ist t eine Variable x, dann ergibt sich direkt

$$t^{\mathcal{F}_E,\beta} = \beta(x) = [t_{x,\beta}]_E$$

Ist $t_i^{\mathcal{F}_E,\beta} = [\sigma(t_i)]_E$ für alle $1 \leq i \leq n$ schon gezeigt, so ergibt sich für $t = f(t_1, \ldots, t_n)$:

$$\begin{array}{rcll} t^{\mathcal{F}_E,\beta} & = & I(f)(t_1^{\mathcal{F}_E,\beta}, \ldots, t_n^{\mathcal{F}_E,\beta}) & \text{allgemeine Semantikdefinition} \\ & = & I(f)([\sigma(t_1)]_E, \ldots, [\sigma(t_n)]_E) & \text{Induktionsvoraussetzung} \\ & = & [f(\sigma(t_1), \ldots, \sigma(t_n))]_E & \text{Def. on } \mathcal{F}_E \\ & = & [\sigma(f(t_1, \ldots, t_n))]_E & \text{Eigenschaft von Substitutionen} \end{array}$$

(2): Für jede Gleichung $l \doteq r$ in E und jede Variablenbelegung β ist $l^{\mathcal{F}_E,\beta} = r^{\mathcal{F}_E,\beta}$ zu zeigen. Nach Teil (1) dieser Übungsaufgabe genügt es dafür $\sigma(l) \leftrightarrow_E \sigma(r)$ zu zeigen; wobei σ, wie in Teil (1) berechnet wird. Aus $l \doteq r \in E$ folgt trivialerweise $l \leftrightarrow_E r$, woraus mit Übungsaufgabe 6.4.2(3) auf $\sigma(l) \leftrightarrow_E \sigma(r)$ geschlossen werden kann.

(3): Gelte $E \models s \doteq t$. Aus dem vorangegangenen Teil dieser Übungsaufgabe wissen wir $\mathcal{F}_E \models E$. Somit gilt für jede Variablenbelegung β auch $s^{\mathcal{F}_E,\beta} = t^{\mathcal{F}_E,\beta}$. Das gilt insbesndere für die spezielle Variablenbelegung $\beta_0(x) = [x]_E$. Nach Teil (2) dieser Übungsaufgabe erhalten wir $s^{\mathcal{F}_E,\beta_0} = [s]_E = [t]_E = t^{\mathcal{F}_E,\beta_0}$. Die Gleichheit der Äquivalenzklassen $[s]_E$ und $[t]_E$ ist aber gleichbedeutend mit $s \leftrightarrow_E t$.

13.14 Lösungen zu Kapitel 7 JML

Aufgabe 7.4.1 Reines Nachdenken hilft hier nichts. Die Substitution y für x in der Nachbedingung aus Abb. 7.1 führt zu

$$rec.x == \backslash old(rec.x)\ \&\&\ rec.x == \backslash old(rec.x) + 1;$$

was widersprüchlich ist.
Die Entwickler von Java haben sich für $rec.x == \backslash old(rec.x)$ entschieden.

Aufgabe 7.4.2 Die Signatur Σ umfasst mindestens die einstelligen Funktionen $x(), y()$ und $rec()$, die zweistellige Relation $\geq$ und die Konstanten 0, $this$. Dann sieht die Invariante so aus:

$$x(this) \geq 0 \wedge y(this) \geq 0 \wedge rec(x(this)) \geq 0 \wedge rec(y(this)) \geq 0$$

Aufgabe 7.4.3 Das folgende Programm leistet das Gewünschte:

— Java + JML —

```
class SITA0{ public int[] a1, a2; int i, j;
/*@ public normal_behaviour
  @  requires 0<=l && l<r &&  <=a1.length && r<=a2.length;
  @  assignable \nothing;
  @  ensures ( l <= \result && \result < r &&
  @  a1[\result] ==  a2[\result]) | \result == r ;
  @  ensures (\forall int j; l <= j && j < \result;
  @   a1[j] !=  a2[j] );
  @*/
 public int  commonEntry(int l, int r) { int k = l;
/*@ loop_invariant
  @  l <= k && k <= r
  @  && (\forall int i; l <= i && i < k; a1[i] != a2[i] );
  @  assignable \nothing;
  @  decreases a1.length - k;
  @*/
   while(k < r && a1[k] != a2[k]){k++;}
    return k;}}
```

— Java + JML —

Wie man sieht ist eine Änderung der JML Annotationen in diesem Fall nicht erforderlich.

Aufgabe 7.4.4 Hier ist eine mögliche Lösung:

Java + JML

```
public class Max0{
 public int[] list;
 /*@ public normal_behavior
   @ requires list.length > 0;
   @ ensures (\forall int j;0 <= j && j < list.length;
   @ list[j] <= \result );
   @ ensures (\exists int i;0 <= i && i < list.length;
   @ list[i] == \result);
   @ assignable \nothing;
   @*/
 public int  max(){
   int pos=0;
   int m=list[0];
  /*@
    @ loop_invariant pos >= 0 && pos <= list.length &&
    @ (\forall int j;0<= j && j < pos;list[j] <= m) &&
    @ (\exists int i;0<=i && i<list.length;list[i]==m);
    @ decreases list.length-pos;
    @ assignable \nothing;
    @*/
   while (pos < list.length) {
    if (list[pos]>m) {m = list[pos];}
       pos++;}
    return m;}
}
```

Java + JML

Aufgabe 7.4.5 Folgende Invariante leistet das Gewünschte:

Java + JML

```
public class Person{  ....
 /*@  public invariant
   @  vorgesetzer != null ==>
   @       arbeitgeber == vorgesetzter.arbeitgeber;
   @*/
```

Java + JML

13.15 Lösungen zu Abschnitt 8.1 Modale Logik

Aufgabe 8.5.1 Alle Aufgaben können nach dem gleichen Muster bewiesen werden, das wir anhand der transitiven Allgemeingültigkeit von $\Diamond\Diamond A \rightarrow \Diamond A$ vorführen wollen:

Nach Lemma 8.9 ist in transitiven Kripke-Strukturen $\Box A \rightarrow \Box\Box A$ allgemeingülig. Also ist auch $\Box\neg A \rightarrow \Box\Box\neg A$ allgemeingültig und damit auch die Kontraposition davon $\neg\Box\Box\neg A \rightarrow \neg\Box\neg A$. Indem man mehrfach die Tautologie $\neg\Box\neg C \leftrightarrow \Diamond C$ ausnutzt erhält man , wie gewünscht, $\Diamond\Diamond A \rightarrow \Diamond A$.

Aufgabe 8.5.2 Teile 1 und 2 sind offensichtlich.
Für Teil 3: $A \vdash^G \Box A$ aber nicht $A \vdash^L \Box A$.

Aufgabe 8.5.3 Intuitiv ist die Behauptung einleuchtend: für die Gültigkeit von A können nur die Zustände relevant sein, die von s aus via R erreichbar sind. Formal führen wir einen Beweis durch Induktion über den Formelaufbau von A. Ist $A = P$ ein Atom so gilt:

$$(\mathcal{K}_T, s) \models P \text{ gdw } I_T(P, s) = 1 \text{ gdw } I(P, s) = 1 \text{ gdw } (\mathcal{K}, s) \models P$$

Die Induktionsschritte für die aussagenlogischen Junktoren sind einfach, z.B. für die Konjunktion haben wir:

$$\begin{array}{llll}
(\mathcal{K}_T, s) \models A_1 \wedge A_2 & \text{gdw} & (\mathcal{K}_T, s) \models A_1 \text{ und } (\mathcal{K}_T, s) \models A_2 & (\text{Def.von } \models) \\
 & \text{gdw} & (\mathcal{K}, s) \models A_1 \text{ und } (\mathcal{K}, s) \models A_2 & (\text{Ind.Vor}) \\
 & \text{gdw} & (\mathcal{K}, s) \models A_1 \wedge A_2 & (\text{Def.von } \models)
\end{array}$$

Es bleibt ein interessanter Fall zu untersuchen:

$$\begin{array}{llll}
(\mathcal{K}_T, s) \models \Box A & \text{gdw} & \forall t \in T \text{ mit } R_T(s,t) \text{ gilt } (\mathcal{K}_T, t) \models A & (\text{Def.von } \models) \\
 & \text{gdw} & \forall t \in S \text{ mit } R_T(s,t) \text{ gilt } (\mathcal{K}_T, t) \models A & (\text{Def.von } T) \\
 & \text{gdw} & \forall t \in S \text{ mit } R(s,t) \text{ gilt } (\mathcal{K}_T, t) \models A & (\text{Def.von } R_T) \\
 & \text{gdw} & \forall t \in S \text{ mit } R(s,t) \text{ gilt } (\mathcal{K}, t) \models A & (\text{Ind.Vor.}) \\
 & \text{gdw} & (\mathcal{K}, s) \models \Box A & (\text{Def.von } \models)
\end{array}$$

Aufgabe 8.5.4 Die Richtung, wenn $A^* \vdash^L B$ dann $A \vdash^G B$, folgt wegen $A \in A^*$ aus Teil 2 von Aufgabe 8.5.2.
Sei jetzt $A \vdash^G B$ vorausgesetzt. Wir wollen $A^* \vdash^L B$ zeigen. Sei dazu $\mathcal{K}$ eine beliebige Kripke-Struktur und s_0 eine beliebige Welt von $\mathcal{K}$ mit $(\mathcal{K}, s_0) \models A^*$.

Unser Ziel ist $(\mathcal{K}, s_0) \models B$ herzuleiten. Sei S_0 die am Ende der Übungsaufgabe 8.5.3 definierte Teilmenge aller Welten von $\mathcal{K}$. Mit $\mathcal{K}_0$ bezeichnen wir die Kripke-Struktur $\mathcal{K}_{S_0}$ (siehe ebenfalls Übungsaufgabe 8.5.3). Nach Aufgabe 8.5.3 folgt auch $(\mathcal{K}_0, s_0) \models A^*$. Da jede Welt t in S_0 in endlich vielen R-Schritten von s_0 aus erreicht werden kann folgt daraus $(\mathcal{K}_0, t) \models A$ für alle $t \in S_0$. Wegen $A \vdash^G B$ folgt insbesondere $(\mathcal{K}_0, s_0) \models B$. Woraus wieder mit Aufgabe 8.5.3 folgt $(\mathcal{K}, s_0) \models B$.

∎

Aufgabe 8.5.5 Sei A eine modallogische Formel und B die dazu negationsduale Formel. Sei $\mathcal{K} = (S, R, I)$ eine Kripke Struktur und I_d die Interpretationsfunktion mit

$$I_d(p, s) = \begin{cases} \mathbf{F} & \text{falls } I(p, s) = \mathbf{W} \\ \mathbf{W} & \text{falls } I(p, s) = \mathbf{F} \end{cases}$$

Offensichtlich gilt für alle $s \in S$

$$((S, R, I), s) \models A \text{ gdw } ((S, R, I_d), s) \models B.$$

Wenn A eine charakterisierende Formel für $\mathbf{R}$ ist, dann gilt für alle $(S, R) \in \mathbf{R}$ und alle I $(S, R, I) \models A$. Dann gilt nach der eben gemachten Beobachtung für alle I_d auch $(S, R, I_d) \models B$. Da aber jede Interpretation sich in der Form I_d darstellen läßt, genauer gilt sogar $I = (I_d)_d$, haben wir somit für alle I $(S, R, I) \models B$.

Außerdem gilt: wenn für einen Kripke Rahmen (S, R) für alle Interpretationen I gilt $(S, R, I) \models A$, dann gilt $(S, R) \in \mathbf{R}$. Dann gilt diese Eigenschaft aber auch für B anstelle von A. Denn aus der Voraussetzung $(S, R, I) \models B$ für alle I folgt, wie eben mit denselben Argumenten wie oben, dass auch für alle I gilt $(S, R, I) \models A$.

Zusammengenommen sehen wir, daß auch B die Klasse $\mathbf{R}$ charakterisiert.

∎

Aufgabe 8.5.6

1. Die zu $\Box A \to A$ negationsduale Formel ist
   $A \to \Diamond A$.

2. Die zu $\Box A \to \Box\Box A$ negationsduale Formel ist
   $\Diamond\Diamond A \to \Diamond A$.

3. Die zu $A \to \Box\Diamond A$ negationsduale Formel ist $\Diamond\Box A \to A$.

4. Die zu $\Diamond A \to \Box A$ negationsduale Formel ist $\Diamond A \to \Box A$. (Selbstdualität)

Aufgabe 8.5.7 Es sind zwei Richtungen zu beweisen.
Teil 1 Sei $\mathcal{K} = (S, R, I)$ eine Kripke-Struktur mit Euklidischem Rahmen. Sei $s \in S$ mit $s \models \Diamond p$. Es gibt als $s_1 \in S$ mit $s_1 \models p$. Wir wollen $s \models \Box\Diamond p$ zeigen. Für jede Welt s_2 mit $R(s, s_2)$ müssen wir also $s_2 \models \Diamond p$ zeigen. Da (S, R) Euklidisch ist gilt $R(s_2, s_1)$ und $s_2 \models \Diamond p$ ist gezeigt.
Teil 2 Angenommen der Rahmen (S, R) ist nicht Euklidisch. Dann gibt es Welten $s, s_1, s_2 \in S$ mit $R(s, s_1)$, $R(s, s_2)$ und $\neg R(s_2, s_1)$. Wir definieren eine Interpretationsfunktion I durch:

$$I(p,t) = \begin{cases} \mathbf{W} & \text{falls } \neg R(s_2, t) \\ \mathbf{F} & \text{falls } \textit{sonst} \end{cases}$$

Wegen $\neg R(s_2, s_1)$ gilt also $s_1 \models p$ und damit $s \models \Diamond p$. Aus der Definition von I folgt außerdem $s_2 \models \neg\Diamond p$ und daher auch $s \models \neg\Box\Diamond p$. Somit gilt nicht $s \models \Diamond p \to \Box\Diamond p$.

Aufgabe 8.5.8 Wir zeigen, daß $\Diamond\Box p \to \Diamond p$ die Klasse der schwachkonfluenten Rahmen charakterisiert.
Teil 1 Sei $\mathcal{K} = (S, R, I)$ eine Kripke-Struktur mit schwach konfluentem Rahmen. Für $s \in S$ haben wir $s \models \Diamond\Box p \to \Diamond p$ zu zeigen. Nehmen wir also die rechte Seite der Implikation an, $s \models \Diamond\Box p$, und versuchen $s \models \Diamond p$ zu zeigen. Aus $s \models \Diamond\Box p$ folgt die Existenz einer Welt $s_1 \in S$ mit $R(s, s_1)$ und $s_1 \models \Box p$. Wegen der schwachen Konfluenzeigenschaft existiert eine Welt s_2 mit $R(s, s_2)$ und $R(s_1, s_2)$. Aus $s_1 \models \Box p$ folgt also auch $s_2 \models p$. Da auch $R(s, s_2)$ gilt haben wir $s \models \Diamond p$ gezeigt, wie gewünscht.
Teil 2 Angenommen der Rahmen (S, R) ist nicht schwach konfluent. Dann gibt es $s_1, s_2 \in S$ mit $R(s_1, s_2)$, so daß für alle $t \in S$ gilt $\neg R(s_1, t)$ oder $\neg R(s_2, t)$. Wir definieren eine Interpretationsfunktion I durch:

$$I(p,t) = \begin{cases} \mathbf{W} & \text{falls } R(s_2, t) \\ \mathbf{F} & \text{sonst} \end{cases}$$

Nach Definition von I gilt in der Kripke Struktur $\mathcal{K} = (S, R, I)$ auf jeden Fall $s_2 \models \Box p$ und auch $s_1 \models \Diamond\Box p$. Wäre auch $s_1 \models \Diamond p$ wahr, so würde es ein $t \in S$ geben mit $R(s_1, t)$ und $t \models p$. Was nach Definition von I zu $R(s_2, t)$ führt. Ein solches t sollte es aber nach Annahme nicht geben. Der Widerspruchsbeweis ist damit erfolgreich abgeschlossen.

13.16 Lösungen zu Abschnitt 9.1 LTL

Aufgabe 9.1.1

1. $A\ \mathbf{U}^+\ B \equiv X\ (A\ \mathbf{U}\ B)$
2. $A\ \mathbf{U}\ B \equiv B \vee (A \wedge (A\ \mathbf{U}^+\ B))$
   $X\ A \equiv \mathbf{0}\ \mathbf{U}^+\ A$

Aufgabe 9.1.2 $A_{2p} = p \wedge \neg X\ p \wedge \Box(p \leftrightarrow XX\ p)$
Aufgabe 9.1.3

1. $\neg(B\ \mathbf{U}\ C) \leftrightarrow \neg C\ \mathbf{U}_w\ (\neg C \wedge \neg B)$
   Beweis:
   Wir zeigen zunächst die Implikation $\rightarrow$.
   $\xi \models \neg(B\ \mathbf{U}\ C)$ kann aus zwei Gründen wahr sein
   1.Fall: Für alle n gilt $\xi_n \models \neg C$.
   Dann gilt aber offensichtlich auch $\xi \models \neg C\ \mathbf{U}_w\ (\neg C \wedge \neg B)$. Es würde sogar $\xi \models \neg C\ \mathbf{U}_w\ F$ für jedes beliebige F gelten
   2.Fall: Für jedes n mit $\xi_n \models C$ gibt es ein k, $0 \le k < n$ mit $\xi_k \models \neg B$. Wir wählen das kleinste n, so daß $\xi_n \models C$ gilt. Auch dazu gibt es ein k, $0 \le k < n$ mit $\xi_k \models \neg B$. Wegen der Minimalität von n gilt $\xi_k \models (\neg C \wedge \neg B)$ und für alle m, $0 \le m < k$ auch $\xi_m \models \neg C$. Also $\xi \models \neg C\ \mathbf{U}_w\ (\neg C \wedge \neg B)$

   Kommen wir zum Beweis der umgekehrten Implikation.
   $\xi \models \neg C\ \mathbf{U}_w\ (\neg C \wedge \neg B)$ kann aus zwei Gründen wahr sein.
   1.Fall: Für alle n gilt $\xi_n \models (\neg C \wedge B)$
   Dann gilt $\xi \models \neg(B\ \mathbf{U}\ C)$. Es würde sogar $\xi \models \neg(F\ \mathbf{U}\ C)$ für jedes F folgen.
   2.Fall: Es gibt n mit $\xi_n \models (\neg C \wedge \neg B)$ und für alle $0 \le k < n$ $\xi_k \models \neg C$. Betrachten wir ein m mit $\xi_m \models C$. Dann muß $n < m$ gelten. Weil für $k < m$ $\xi_k \models \neg B$ gilt haben wir insgesamt $\xi \models \neg(B\ \mathbf{U}\ C)$.

2. $\neg(B\ \mathbf{U}_w\ C) \leftrightarrow \neg C\ \mathbf{U}\ (\neg C \wedge \neg B)$
   Beweis für die Implikation $\rightarrow$:
   Wenn $\xi \models \neg(B\ \mathbf{U}_w\ C)$ gilt und für alle n gilt $\xi_n \models \neg C$, dann folgt für mindestens ein m auch $\xi_m \models \neg B$. Wenn aber für ein n gilt $\xi_n \models C$, dann gibt es wegen $\xi \models \neg(B\ \mathbf{U}_w\ C)$ ein m mit $0 \le m < n$ mit $\xi_m \models \neg B$. Auf jeden Fall gibt es also ein m mit $\xi_m \models \neg B$. Wir wählen das kleinste m mit dieser Eigenschaft, d.h. für alle k, $0 \le k < m$ gilt $\xi_k \models B$. Angenommen es gäbe ein j mit $0 \le j \le m$ mit $\xi_j \models C$. Wegen

$\xi \models \neg(B\ \mathbf{U}_w\ C)$ gibt es ein j_0, $0 \leq j_0 < j \leq m$ mit $\xi_{j_0} \models \neg B$. Das ist ein Widerspruch zur Wahl von m. Somit gilt für alle k, $0 \leq k \leq m$ $\xi_k \models \neg C$. Das beweist $\xi \models \neg C\ \mathbf{U}\ (\neg C \wedge \neg B)$.
Beweis für die Implikation $\leftarrow$:
Es gibt also ein n mit $\xi_n \models \neg B \wedge \neg C$ und für alle m, $0 \leq m < n$ gilt $\xi_m \models \neg C$. Eine Möglichkeit, daß $\xi \models B\ \mathbf{U}_w\ C$ wahr ist, wäre der Fall, daß für alle n' gilt $\xi_{n'} \models \neg C \wedge B$. Das kann wegen $\xi_n \models \neg B$ nicht sein. Die zweite Möglichkeit, daß $\xi \models B\ \mathbf{U}_w\ C$ wahr ist, liegt in der Existenz eines n' mit $\xi_{n'} \models C$ und für alle m', $0 \leq m' < n'$ gilt $\xi_{m'} \models B$. Nach Wahl von n muß $n < n'$ gelten. Dann verletzt aber $m' = n$ die Forderung $\xi_{m'} \models B$. Insgesamt haben wir also $\neg(B\ \mathbf{U}_w\ C)$ gezeigt.

3. $B\ \mathbf{U}\ C \leftrightarrow C \vee (B \wedge X\ (B\ \mathbf{U}\ C))$
   Beweis:
   $\xi \models B\ \mathbf{U}\ C$ gilt genau dann, wenn es ein n gibt mit $\xi_n \models C$ und für alle m, $0 \leq m < n$ gilt $\xi_m \models B$. Wir unterscheiden die Fälle $0 = n$ und $0 < n$. Im ersten Fall ist die zweite Bedingung trivialerweise erfüllt und es bleibt nur $\xi \models C$. Im zweiten Fall muß auf jeden Fall $\xi \models B$ gelten. Der Rest ist äquivalent zu $\xi_1 \models B\ \mathbf{U}\ C$, was gleichbedeutend ist mit $\xi \models X\ (B\ \mathbf{U}\ C)$.

4. $\Diamond B \leftrightarrow (B \vee X\ \Diamond B)$
   Beweis: `muß noch ergänzt werden`

5. $\Box B \leftrightarrow (B \wedge X\ \Box B)$
   Beweis: `muß noch ergänzt werden`

Aufgabe 9.1.4 Es genügt offensichtlich für jede omega-Struktur ξ zu zeigen, daß

$$\xi \models A \quad \text{gdw} \quad sf(\xi) \models A$$

gilt. Sei dazu $an : \mathbb{N} \to 2^{\mathbb{N}}$ die Hilfsfunktion aus der Definition von $sf(\xi)$. Wir zeigen allgemeiner, daß für jede LTL-Formel A ohne X , alle n und alle $k \in an(n)$ gilt

$$sf(\xi)_n \models A \quad \text{gdw} \quad \xi_k \models A$$

Die Behauptung folgt dann durch die Spezialisierung $n = k = 0$ wegen $0 \in an(0)$, $\xi_0 = \xi$ und $sf(\xi)_0 = sf(\xi)$.

Der Beweis erfolgt durch Induktion über den Aufbau der Formel C. Ist $C = p$ ein aussagenlogisches Atom, dann folgt die Behauptung aus $sf(\xi)(n) = \xi(k)$ für alle $k \in an(n)$. Die aussagenlogischen Fälle sind trivial. Es bleibt der Fall $C = A\ \mathbf{U}\ B$ zu betrachten. Wir fixieren n und $k \in an(n)$.

1.Teil $\quad sf(\xi)_n \models A \Rightarrow \xi_k \models A \mathbf{U} B$

Es gibt also ein $t \geq n$ mit $sf(\xi)_t \models B$ und für alle t' mit $n \leq t' < t$ gilt $sf(\xi)_{t'} \models A$.

Fall 1 $n = t$**:** Per Induktionsvoraussetzung folgt $\xi_k \models B$ und damit auch schon $\xi_k \models A \mathbf{U} B$.

Fall 2 $n < t$ **und** $an(n) \neq an(t)$**:** Sei m das kleinste Element aus $an(t)$. Wegen $n < t$ gilt jetzt $k < m$. Nach Induktionsvoraussetzung gilt weiter $\xi_m \models B$. Für jedes m' mit $k \leq m' < m$ gibt es ein t', $n \leq t' < t$ mit $m' \in an(t')$. Wieder folgt mit Hilfe der Induktionsvoraussetzung aus $sf(\xi)_{t'} \models A$ auch $\xi_{m'} \models A$. Insgesamt haben wir jetzt $\xi_k \models A \mathbf{U} B$ gezeigt.

Fall 3 $n < t$ **und** $an(n) = an(t)$**:** In diesem Fall gilt $k \in an(t)$ und somit folgt, wieder nach Induktionsvoraussetzung, aus $sf(\xi)_t \models B$ auch $\xi_k \models B$ und daher unmittelbar $\xi_k \models A \mathbf{U} B$.

2.Teil $\quad \xi_k \models A \mathbf{U} B \Rightarrow sf(\xi)_n \models C$

Es gibt also ein m, $k \leq m$ mit $\xi_m \models B$ und für alle m', $k \leq m' < m$ gilt $\xi_{m'} \models A$.

Fall 1 $m \in an(n)$**:** Dann gilt nach Induktionsvoraussetzung auch $sf(\xi)_n \models B$ und damit sofort $sf(\xi)_n \models A \mathbf{U} B$.

Fall 1 $m \notin an(n)$**:** Dann gibt es t, $n < t$ mit $m \in an(t)$ was nach IV wieder $sf(\xi)_t \models B$ liefert. Für t' mit $n \leq t' < t$ gibt es wenigstens ein $m' \in an(t')$ mit $k \leq m' < m$. Nach IV folgt aus $\xi_{m'} \models A$ auch $sf(\xi)_{t'} \models A$. Ingesamt ergibt sich wieder $sf(\xi)_n \models A \mathbf{U} B$.

∎

Aufgabe 9.1.5 `muß noch ergänzt werden`
Aufgabe 9.1.6 Sei ξ eine omega-Struktur, $n \in \mathbb{N}$. A, B seien LTL Formeln.

$(\xi, n) \models p$	gdw	$p \in \xi(n)$ (p ein AL Atom)
$(\xi, n) \models op(A, B)$		für aussagenlogische Kombinationen $op(A, B)$ von A und B wie üblich
$(\xi, n) \models \Box A$	gdw	für alle $m \in \mathbb{N}$ mit $m \geq n$ gilt $(\xi, m) \models A$
$(\xi, n) \models \Diamond A$	gdw	es gibt ein $m \in \mathbb{N}$ mit $m \geq n$ und $(\xi, m) \models A$
$(\xi, n) \models A\ \mathbf{U}\ B$	gdw	es gibt $m \in \mathbb{N}$ mit $m \geq n$ und $(\xi, m) \models B$ und für alle k mit $n \leq k < m$ gilt $(\xi, k) \models A$
$(\xi, n) \models X\ A$	gdw	$(\xi, n+1) \models A$

■

Aufgabe 9.1.7 $F = \Box(p \rightarrow \Diamond r \wedge (\neg r\ \mathbf{U}\ q))$
Aufgabe 9.1.8 $F = \Box(p \rightarrow (\neg r \wedge \neg q)\ \mathbf{U}\ (r \wedge \neg q \wedge X\ (\neg r \wedge \neg q)\ \mathbf{U}\ q)$
Aufgabe 9.1.9

1. Zum Beweis der Richtung von links nach rechts nehmen wir für eine beliebige omega-Struktur ξ an, daß $\xi \models (A\ \mathbf{V}\ B)\ \mathbf{U}\ C$ gilt. Nach Definition der Semantik des $\mathbf{U}$ -Operators gibt es ein $n \in \mathbb{N}$ mit $\xi_n \models C$ und für alle m, $0 \leq m < n$ gilt $\xi_m \models (A\ \mathbf{V}\ B)$. Daraus folgt wegen der Semantik des $\mathbf{V}$ -Operators insbesondere für alle m, $0 \leq m < n$ gilt $\xi_m \models B$. Falls $n = 0$ gilt dann haben wir $\xi \models C$. Von jetzt an setzen wir $n < 0$ voraus und unterscheiden drei Fälle:

   $\xi_n \models B$ und $\xi_{n-1} \models A$ In diesem Fall gilt wie man leicht nachprüft $\xi \models B\ \mathbf{U}\ (A \wedge XC \wedge B)$

   $\xi_n \models B$ und $\xi_{n-1} \models \neg A$ In diesem Fall gilt $\xi \models B\ \mathbf{U}\ (C \wedge (A\ \mathbf{V}\ B))$. Als einzig nicht-triviale Teilbehauptung bleibt $\xi_n \models A\ \mathbf{V}\ B$ zu zeigen. Da $\xi_n \models B$ in diesem Fall gilt, müssen wir nur noch zeigen, daß aus $\xi_k \models \neg B$ für $n < k$ die Existenz von r mit $n \leq r < k$ folgt mit $\xi_r \models A$. Da wir $\xi_{n-1} \models A\ \mathbf{V}\ B$ wissen, gibt es auf jedenfalls ein r mit $n - 1 \leq r < k$ mit $\xi_r \models A$. Aus der Fallannahme folgt aber, daß $r \neq (n-1)$ ist, und wir sind fertig.

   $\xi_n \models \neg B$ Da $\xi_{n-1} \models A\ \mathbf{V}\ B$ gilt folgt $\xi_{n-1} \models A$ und damit gilt $\xi \models B\ \mathbf{U}\ (A \wedge B \wedge XC)$.

   Wenden wir uns jetzt der umgekehrten Implikation zu. Gilt $\xi \models C$ dann gilt offensichtlich auch $\xi \models (A\ \mathbf{V}\ B)\ \mathbf{U}\ C$ Es bleiben die folgenden beiden Behauptungen zu zeigen:

   $(B\ \mathbf{U}\ (C \wedge (A\ \mathbf{V}\ B))) \rightarrow (A\ \mathbf{V}\ B)\ \mathbf{U}\ C$
   Aus $\xi \models B\ \mathbf{U}\ (C \wedge (A\ \mathbf{V}\ B))$ folgt die Existenz einer Zahl $n \in \mathbb{N}$ mit $\xi_n \models (C \wedge (A\ \mathbf{V}\ B))$ und für alle $0 \leq m < n$ gilt $\xi_m \models B$. Daraus

folgt, auch für den Fall $n = 0$, $\xi \models B$. Um $\xi \models (A \mathbf{V} B) \mathbf{U} C$ zu zeigen brauchen wir noch für alle $0 \le m < n$ $\xi_m \models A \mathbf{V} B$. Sei dazu für ein k mit $m < k$ $\xi_k \models \neg B$. Aus dem bisher Gesagten folgt $n < k$. Weil aber $\xi_n \models A \mathbf{V} B$ gilt, muss es ein r geben mit $n \le r < k$ mit $\xi_r \models A$. Weil auch $m \le r < k$ gilt haben wir damit $\xi_m \models A \mathbf{V} B$.

$(B \mathbf{U} (A \wedge B \wedge XC)) \rightarrow (A \mathbf{V} B) \mathbf{U} C$
Aus $\xi \models B \mathbf{U} (A \wedge B \wedge XC)$ folgt die Existenz einer Zahl $n \in \mathbb{N}$ mit $\xi_n \models A \wedge B \wedge XC$ und für alle $0 \le m < n$ gilt $\xi_m \models B$. Um $\xi \models (A \mathbf{V} B) \mathbf{U} C$ zu beweisen, brauchen wir eine Zahl n' mit $\xi_{n'} \models C$ und für alle $0 \le m < n'$ $\xi_m \models A \mathbf{V} B$. In dem Beweis im vorangegangen Abschnitt hatten wir dasselbe Problem, dort aber sillschweigend $n' = n$ gesetzt. Hier setzen wir $n' = n+1$. Damit haben wir schon einmal sicher $\xi_{n'} \models C$. Wir fixieren ein m mit $0 \le m \le n$ und müssen $\xi_m \models A \mathbf{V} B$ zeigen. Auf jeden Fall haben wir schon einmal $\xi_m \models B$. Angenommen es gibt k mit $m < k$ mit $\xi_k \models \neg B$. Wir müssen ein r finden mit $m \le r < k$ und $\xi_r \models A$. Nach den bisher akkumulierten Annahmen muß $n < k$ gelten. Wegen $\xi_n \models A$ und $m \le n$ können wir $r = n$ wählen.

∎

2. $\xi \models (\Box A) \mathbf{U} B$ gdw
   es gibt $n \ge 0$ mit $\xi_n \models B$ und für alle $i, 0 \le i < n$ gilt $\xi_i \models \Box A$
   gdw
   $\xi \models B$ oder es gibt $n > 0$ mit $\xi_n \models B$ und
   für alle $i, 0 \le i < n$ gilt $\xi_i \models \Box A$
   gdw
   $\xi \models B$ oder es gibt $n > 0$ mit $\xi_n \models B$ und $\xi_0 \models \Box A$
   gdw
   $\xi \models B$ oder $\xi \models \Diamond B$ und $\xi \models \Box A$
   gdw
   $\xi \models B \vee (\Diamond B \wedge \Box A)$

3. Für die Implikation $lhs \rightarrow rhs$ von rechts nach links nehmen wir $\xi \models (C \vee \Box A) \mathbf{U} B$ an. Es gibt also ein n mit $\xi_n \models B$ und für alle m, $0 \le m < n$ gilt $\xi_m \models (C \vee \Box A)$. Gilt $n = 0$ dann $\xi \models B$ und damit $\xi \models rhs$. Sei von jetzt an $n < 0$. Falls für alle m, $0 \le m < n$ gilt $\xi_m \models C$, dann $\xi \models C \mathbf{U} B$ und damit wieder $\xi \models rhs$. Es bleibt der Fall zu betrachten, in dem ein m existiert mit $0 \le m < n$ und $\xi_m \models \neg C$. Wir betrachten das kleinste m mit dieser Eigenschaft, d.h. für k, $0 \le k < m$ gilt $\xi_k \models C$. Es muß nach den bisherigen Festlegungen $\xi_m \models \Box A$ gelten.

Es gilt ebenfalls $\xi_m \models \Diamond B$. Also insgesamt $\xi \models (C \mathbf{U} (\Box A \wedge \Diamond B))$ und damit $\xi \models rhs$.
Kommen wir zum Beweis der umgekehrten Implikation und betrachten eine omega-Struktur mit $\xi \models rhs$. Aus $\xi \models B$ oder $\xi \models C \mathbf{U} B$ folgt unmittelbar $\xi \models (C \vee \Box A) \mathbf{U} B$. Bleibt also $\xi \models C \mathbf{U} (\Box A \wedge \Diamond B)$ zu untersuchen. Es gibt also ein k mit $\xi_k \models \Box A \wedge \Diamond B$ und für alle m, $0 \leq m < k$ gilt $\xi_m \models C$. Wegen $\xi_k \models \Diamond B$ gibt es ein n, $k \leq n$ mit $\xi_n \models B$. Für alle m, $0 \leq m < n$ gilt $\xi_m \models (C \vee \Box A)$, den für $0 \leq m < k$ gilt $\xi_m \models C$ und aus $\xi_k \models \Box A$ folgt für alle $k \leq m$ auch $\xi_m \models \Box A$. Somit $\xi \models lhs$.

Aufgabe 9.1.10 Wir konzentrieren uns als erstes auf die Implikation von links nach rechts.
Falls $\xi \models A \mathbf{V} B$ folgt zunächst $\xi \models B$. Wir unterscheiden den Fall $\xi \models \Box B$, womit *linke Seite impliziert rechte Seite* gilt und den Fall, daß es ein n gilt mit $\xi_n \models \neg B$. Sei n_0, das kleinste n mit dieser Eigenschaft. Wegen $\xi \models A \mathbf{V} B$, gibt es ein m, $0 \leq m < n_0$ mit $\xi_m \models A$. Wegen der Wahl von n_0 folgt daraus $\xi \models B \mathbf{U} (A \wedge B)$
Nun zur Implikation von rechts nach links.
Gilt $\xi \models \Box B$, dann offensichtlich auch $\xi \models A \mathbf{V} B$. Gilt $B \mathbf{U} (A \wedge B)$ dann auch $\xi \models B$ und es gibt n_0 mit $\xi_{n_0} \models A \wedge B$ und für alle m, $0 \leq m < n_0$ auch $\xi_m \models B$. Um $\xi \models A \mathbf{V} B$ zu zeigen, nehmen wir an, daß für ein n gilt $\xi_n \models \neg B$. Nach Wahl von n_0 gilt jedenfalls $n_0 < n$. Somit gibt es ein k mit $0 \leq k < n$ und $\xi_k \models A$ nämlich $k = n_0$.

■

Aufgabe 9.1.11
Implikation $\rightarrow$
Aus $\xi \models (A \mathbf{U} B) \mathbf{U} C$ folgt die Existenz einer Zahl d mit $\xi_d \models C$ und für alle k, $0 \leq k < d$, gilt $\xi_k \models A \mathbf{U} B$. Fall $d = 0$ dann gilt $\xi \models C$ und die rechte Seite der Implikation ist wahr für ξ. Im folgenden können wir also $0 < d$ annehmen. Wir stellen zunächst fest, daß für alle k, $0 \leq k < d$ $\xi_k \models (A \vee B)$ gilt. Wir treffen die folgende Fallunterscheidung
$\mathbf{\xi_{d-1} \models B}$
Daraus folgt $\xi \models (A \vee B) \mathbf{U} (B \wedge X\ C)$ und die rechte Seite der Implikation ist wahr für ξ.
$\mathbf{\xi_{d-1} \not\models B}$
Wie oben festgestellt gilt $\xi_{d-1} \models A \mathbf{U} B$. Es gibt also ein n mit $d - 1 \leq n$ mit $\xi_n \models B$ und für alle k, $d - 1 \leq k < n$ gilt $\xi_k \models A$. Nach der Fallannahme muß sogar $d \leq n$ gelten. Das zeigt $\xi_d \models C \wedge (A \mathbf{U} B)$ und die rechte Seite der Implikation ist wahr für ξ.

Implikation $\leftarrow$
Ist die rechte Seite der behaupteten Äquivalenz wahr in ξ und gilt $\xi \models C$, dann ist natürlich auch die linke Seite wahr. Anderenfall gilt $\xi \models (A \vee B)$ **U** $((B \wedge X\ C) \vee (C \wedge (A$ **U** $B)))$. Es gibt also ein d mit $\xi_d \models ((B \wedge X\ C) \vee (C \wedge (A$ **U** $B))$ und für alle k, $0 \leq k < d$ gilt $\xi \models A \vee B$. Das führt zu der offensichtlichen Fallunterscheidung
$\xi_\mathbf{d} \models \mathbf{B} \wedge \mathbf{X}\ \mathbf{C}$
Hier gilt $\xi_{d+1} \models C$ und für alle k, $0 \leq k \leq d$ $\xi_k \models A \vee B$. Daraus folgt für alle k, $0 \leq k \leq d$ $\xi_k \models A$ **U** B und wir sind fertig.
$\xi_\mathbf{d} \models \mathbf{C} \wedge (\mathbf{A}\ \mathbf{U}\ \mathbf{B})$
Falls für alle k, $0 \leq k \leq d$ $\xi_k \models B$ gilt dann gilt für alle diese k trivialerweise $\xi_k \models A$ **U** B. Anderenfalls sei k_0 der größte Index mit $k_0 < d - 1$ und $\xi_{k_0} \models B$. Für alle k mit $0 \leq k < k_0$ gilt dann $\xi_k \models A$ **U** B. Nach Wahl von k_0 gilt für alle k mit $k_0 < k < d$ $\xi_k \models A$. Zusammen mit $\xi_d \models A$ **U** B folgt daraus für alle k mit $k_0 \leq k < d$ $\xi_k \models A$ **U** B. Zusammengenommen haben wir gezeigt $\xi \models (A$ **U** $B)$ **U** C.
Aufgabe 9.1.12 `to be done`
Aufgabe 9.1.13 Gelte $\xi \models \Box(\neg r \rightarrow X\ r)$. Wir zeigen $\xi \models \Diamond(r \wedge X\ X\ r)$.
Fall 1: $\xi \models r$
Fall 1.1: $\xi_2 \models r$
Dann gilt $\xi \models r \wedge X\ X\ r$ und damit $\xi \models \Diamond(r \wedge X\ X\ r)$ wie gewünscht.
Fall 1.2: $\xi_2 \models \neg r$
Dann gilt nach Voraussetzung $\xi_3 \models r$.
Fall 1.2.1: $\xi_1 \models r$
Dann gilt $\xi_1 \models r \wedge X\ X\ r$ und damit $\xi \models \Diamond(r \wedge X\ X\ r)$ wie gewünscht.
Fall 1.2.1: $\xi_1 \models \neg r$
Nach Voraussetzung folgt $\xi_2 \models r$. Das steht im Widerspruch zur Fallannahme 1.2. Fall 1.2.1 kann nicht eintreten.
Fall 2: $\xi \models \neg r$
Nach Voraussetzung gilt $\xi_1 \models r$
Fall 2.1: $\xi_3 \models r$
Es gilt also $\xi_1 \models r \wedge X\ X\ r$ und damit auch $\xi \models \Diamond(r \wedge X\ X\ r)$.
Fall 2.2: $\xi_3 \models \neg r$
Nach Voraussetzung gilt $\xi_4 \models r$
Fall 2.2.1: $\xi_2 \models r$
Dann gilt $\xi_2 \models r \wedge X\ X\ r$ und damit $\xi \models \Diamond(r \wedge X\ X\ r)$.
Fall 2.2.2: $\xi_2 \models \neg r$
Nach Voraussetzung folgt $\xi_3 \models r$
Das ist ein Widerspruch zu Fallannahme 2.3. Also kann Fall 2.2.2 nicht eintreten.

13.17 Lösungen zu Abschnitt 11.1 Endliche Automaten

Aufgabe 11.1.1 `muß noch ergänzt werden`
Aufgabe 11.1.2 Sei $\mathcal{A} = (Q, V, q_0, \delta_A, F_A)$ ein endlicher Automat mit spontanen Übergängen.
Wir definieren den endlichen Automaten $\mathcal{B} = (Q, V, q_0, \delta_B, F_B)$ mit

$$\begin{array}{lcll} \delta_B(s,a) & = & \delta_A(s,a) \cup \bigcup_{s' \in Sp_a} \delta_A(s',a) & \\ F_B & = & F_A & \text{falls } F_A \cap \varepsilon - cl(q_0) = \emptyset \\ F_B & = & F_A \cup \{q_0\} & \text{falls } F_A \cap \varepsilon - cl((q_0) \neq \emptyset \end{array}$$

wobei $Sp_a = \varepsilon - cl(\{s\})$.
Wir wollen $L(\mathcal{A}) = L(\mathcal{B})$ zeigen.

Für das leere Wort gilt $\varepsilon \in L(\mathcal{A})$ genau dann wenn $\varepsilon - cl(q_0) \cap F \neq \emptyset$. Das ist genau dann der Fall wenn $q_0 \in F_B$. Was wiederum äquivalent ist zu $\varepsilon \in L(\mathcal{B})$.

Es bleibt für jedes nichtleere Wort $w \in V^*$ zu zeigen, daß $\bar{\delta_A}(s,w) = \bar{\delta_B}(s,w)$ gilt. `muß noch ergänzt werden`
Weiterhin muß gezeigt werden, daß für jedes nichtleere Wort w gilt $\bar{\delta_A}(s,w) \cap F_A \neq \emptyset$ genau dann, wenn $\bar{\delta_B}(s,w) \cap F_B \neq \emptyset$. `muß noch ergänzt werden`

13.18 Lösungen zu Abschnitt 11.2 Büchi Automaten

Aufgabe 11.2.1 `muß noch ergänzt werden`
Aufgabe 11.2.2 `muß noch ergänzt werden`
Aufgabe 11.2.3 Für den nicht-deterministische Automat $\mathcal{N}_{afin}$ galt $L^\omega(\mathcal{N}_{afin}) =$

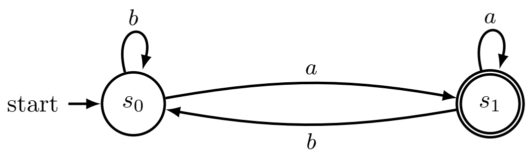


Abbildung 13.1: Deterministischer Automat zu $\mathcal{A}_{afin}$

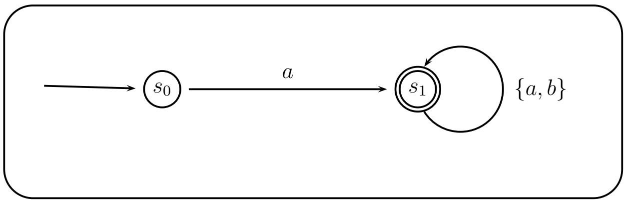
$\{a, b\}^* a^\omega$. Für den deterministische Automat $\mathcal{N}_{Detafin}$, siehe Abbildung 13.1, besteht $L^\omega(\mathcal{N}_{Detafin})$ aus alle Wörtern $w \in \{a, b\}^\omega$ in denen sowohl a als auch b unendlich oft vorkommen.

Aufgabe 11.2.4 Wir zeigen zuerst $K^\omega \subseteq L^\omega(\mathcal{B})$ und danach $L^\omega(\mathcal{B}) \subseteq K^\omega$.

Sei $w \in K^\omega$, also $w = w_1 \ldots w_i \ldots$ mit $w_i \in K$. Für jedes i gibt es also eine Berechungsfolge s_i des Automaten $\mathcal{A}$ für w_i, die mit s_0^A beginnt und in einem Zustand $q_i^f \in F_A$ endet. Da in $\mathcal{B}$ jeder Übergang aus $\mathcal{A}$ weiterhin möglich ist, sind alle s_i auch Berechungsfolgen in $\mathcal{B}$. Sei q_i^1 der Zustand in s_i der nach Einlesen des ersten Buchstabens a_i^1 von w_i angenommen wird. Nach Definition von $\mathcal{B}$ ist ein direkter Übergang von q_i^f nach q_i^1 möglich (präziser: $q_i^1 \in \delta(q_i^f, a_i^1)$). Die Berechnungsfolgen s_i lassen sich also zu einer unendlichen Berechungsfolge für $\mathcal{B}$ zusammensetzen, in der unendliche viele Endzustände (die q_i^f) vorkommen, d.h. $w \in L^\omega(\mathcal{B})$.

Für die umgekehrte Richtung fixieren wir ein $w \in L^\omega(\mathcal{B})$. Es gibt also eine Berechungsfolge mit unendlich vielen Endzuständen $q_1 \ldots q_i \ldots$. Mit w_i bezeichnen wir das Wort, das beginnend mit dem Nachfolgezustand q_{i-1}^1 von q_{i-1} bis zu q_i eingelesen wird. Das Wort w_1 soll natürlich mit dem ersten Buchstaben von w beginnen. Offensichtlich gilt $w = w_1 \ldots w_i \ldots$. Es bleibt jetzt nur noch die Frage zu beantworten, ob alle w_i in K liegen. Nach der Voraussetzung über den Automaten $\mathcal{A}$ ist der einzige Nachfolgezustands eines Enzustandes in $\mathcal{B}$ der Anfangszustand. Es muß also für alle i gelten $q_i^1 = s_0^B = s_0^A$. Damit ist $w_i \in L(\mathcal{A}) = K$.

Aufgabe 11.2.5 Wir betrachten den folgenden Automaten, der die Voraussetzungen aus der Übungsaufgabe 11.2.4 verletzt.



Die Konstruktionsvorschrift aus Übungsaufgabe 11.2.4 liefert in diesem Fall $\mathcal{A} = \mathcal{B}$. Aber es gilt $ab^\omega \in L^\omega(\mathcal{A})$ und $ab^\omega \neq K^\omega$.

Aufgabe 11.2.6 Der Automat $\mathcal{A}_{endeb}$ akzeptiert die Menge $K_1 = L(\mathcal{A}_{endeb}) =$

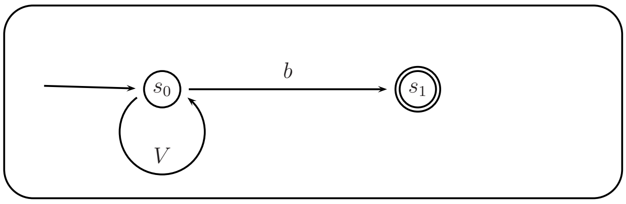


Abbildung 13.2: Der Automat $\mathcal{A}_{endeb}$

$\{w \in V^* \mid w \text{ endet auf } b\}$ Genau so kann man einen Automaten $\mathcal{A}_{endea}$ konstruieren mit $K_2 = L(\mathcal{A}_{endea}) = \{w \in V^* \mid w \text{ endet auf } a\}$. Offensichtlich gilt $K_1 \cap K_2 = \emptyset$. Aber ein $w \in V^\omega$, in dem sowohl a als auch b unendlich oft vorkommt, liegt in $\vec{K}_1 \cap \vec{K}_2$.

Aufgabe 11.2.7 Der Automat $\mathcal{A}_{endeb}$ aus Abbildung 13.2 leistet das Gewünschte. Angenommen, das Komplement der von $\mathcal{A}_{endeb}$ akzeptierten omega-Wörter könnte durch einen deterministischen Büchi Automaten akzeptiert werden. Nach Korollar 11.22 heißt das, daß eine Menge $K \subseteq V^*$ endlicher Wörter existiert mit $\vec{K} = V^\omega \setminus L^\omega(\mathcal{A}_{endeb})$ oder genauer $\vec{K} = \{w \in V^\omega \mid w \text{enthält nur endlich viele } b\}$. Wir zeigen, daß so etwas nicht möglich ist. Da $ba^\omega \in V^\omega \setminus L^\omega(\mathcal{A}_{endeb})$ liegt gibt es nach Definition von $\vec{K}$ ein k_1 mit $ba^{k_1} \in K$. das omega-Wort $ba^{k_1}ba^\omega$ liegt wieder in $V^\omega \setminus L^\omega(\mathcal{A}_{endeb})$ und nach Definition von $\vec{K}$ gibt es k_2 mit $ba^{k_1}ba^{k_2} \in K$. So fortfahrend finden wir für jedes $i \in \mathbb{N}$ Zahlen k_i mit $w_i = ba^{k_1}ba^{k_2} \ldots ba^{k_2} \in K$. Dann liegt aber auch der *Limes* w aller w_i in $\vec{K}$. Das ist aber ein Widerspruch, da w unendlich viele b enthält.

Aufgabe 11.2.8 Sei $\mathcal{A}$ ein endlicher Automat, der spontane Übergänge enthalten kann. Man überzeuge sich zunächst davon, daß die Aussage des Zerlegungssatzes 11.25 richtig ist, unabhängig davon, ob spontane Übergänge vorkommen oder nicht. Wir wissen damit

$$L^\omega(\mathcal{A}) = \bigcup_{i=1}^{n} J_i K_i^\omega$$

für reguläre Mengen J_i, K_i endlicher Wörter. Aus der Theorie endlicher Automaten für endliche Wörter wissen wir, daß es Automaten $\mathcal{A}_i$, $\mathcal{B}_i$ ohne spontane Übergänge gibt mit $J_i = L(\mathcal{A}_i)$ und $K_i = L(\mathcal{B}_i)$. Nach Satz 11.24 wissen wir, daß es einen Büchi Automaten $\mathcal{C}$ gibt mit $L^\omega = \bigcup_{i=1}^n J_i K_i^\omega$. Wir müssen uns nur noch davon überzeugen, daß in den im Beweis von Satz 11.24 benutzen Automatenkonstruktionen keine spontanen Übergänge benutzt wurden. Es zeigt sich, daß das tatsächlich der Fall ist.

Aufgabe 11.2.9 Sei $\mathcal{A} = (S_A, V, s_0^A, \delta_A, F)$ ein Büchi Automat. Wir betrachten den Müller Automaten $\mathcal{M} = (S_A, V, s_0^A, \delta_A, \mathcal{F})$, der alle Bestimmungsstücke von $\mathcal{A}$ übernimmt bis auf die Menge der Endzustände. Diese soll durch $\mathcal{F} = \{Q \subseteq S_A \mid Q \cap F \neq \emptyset\}$ festgelegt werden.
Man sieht leicht daß $L^\omega(\mathcal{A}) = L^\omega(\mathcal{M})$ gilt.

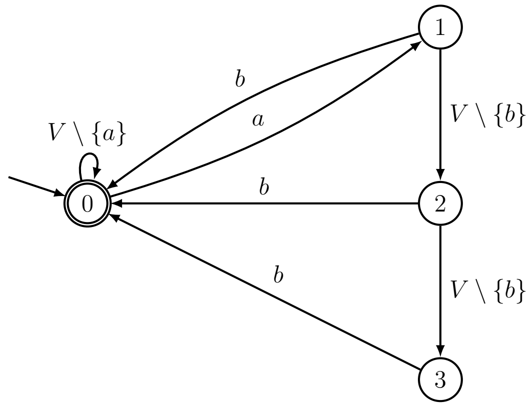
Aufgabe 11.2.10 Sei $\mathcal{M} = (S, V, s_0, \delta, \mathcal{F})$, dann ist $\mathcal{M}_c = (S, V, s_0, \delta, \mathcal{F}_c)$, wobei $\mathcal{F}_c = \{F \subseteq S \mid F \neq \emptyset \text{ und } F \notin \mathcal{F}\}$.

■

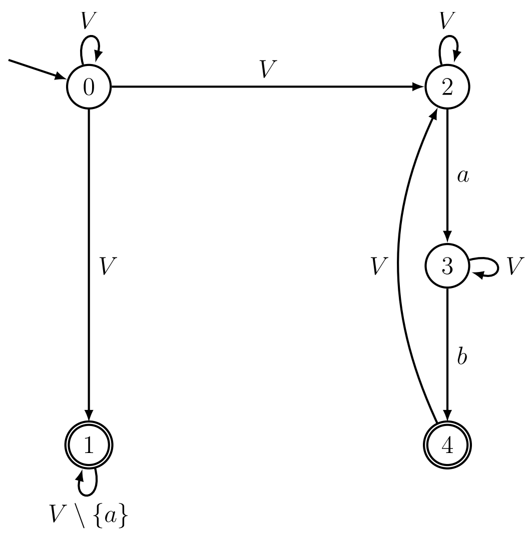
Aufgabe 11.2.11 Gilt $w \in L^\omega(A)$ dann gibt es nach Definition 11.19 eine Berechnungsfolge $s_0, \ldots, s_n, \ldots$ für w, in der für unendlich viele n gilt $s_n \in F$. Die Definition einer Berechnungsfolge stellt sicher, daß sie mit dem Startzustand anfängt und für alle n gilt $s_{n+1} \in \delta(s_n, w(n))$. Daraus folgt unmittelbar $s_n \in Q_n(w)$ für alle n und damit $w \in L^{alt}(A)$.
Gelte jetzt umgekehrt $w \in L^{alt}(A)$. Wir wählen für jedes n einen beliebigen Zustand $s_n \in Q_n(w)$. Wenn $Q_n(w) \cap F \neq \emptyset$ dann wählen wir $s_n \in Q_n(w) \cap F$. Man beachte außerdem, daß $Q_n(w)$ nie die leere Menge sein kann. Denn ist $Q_m(w)$ leer, dann sind auch alle $Q_{m'}(w)$ für $m' \geq m$ leer und die Bedingung $Q_{m'}(w) \cap F \neq \emptyset$ nicht mehr für unendliche viele m' erfüllt. Nach Definition der $Q_n(w)$ ist $s_0, \ldots, s_n, \ldots$ eine Berechnungsfolge für w mit unedlich vielen Endzuständen, also $w \in L^\omega$.

■

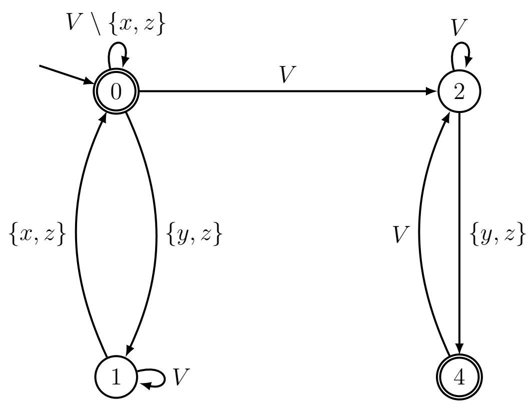
Aufgabe 11.2.12 Der Automat $\mathcal{B}_{a3b}$:



Aufgabe 11.2.13 Man macht sich zunächst klar, daß genau die Wörter $w \in V^\omega$ akzeptiert werden sollen, in denen entweder nur endliche viele a vorkommen oder in denen sowohl a als auch b unendlich oft vorkommen. Das führt zu $\mathcal{B}_{a\infty->b\infty}$:



Aufgabe 11.2.14 Man macht sich zunächst klar, daß $\mathcal{B}_{wfair}$ genau die Wörter w akzeptieren soll, in denen (1) nach jedem Vorkommen von x oder z einer der Buchstaben y oder z vorkommt oder (2) die Menge $\{k \mid w(k) \in \{y, z\}\}$ unendlich ist.



Aufgabe 11.2.15

$$
\begin{array}{lll}
F & = & \exists X(\\
& & [\ (X(0) \wedge P_a(0) \wedge P_b(s(0)) \wedge \neg X(s(0)) \wedge X(s^2(0))) \\
& & \quad \vee \\
& & \quad (X(0) \wedge P_b(0) \wedge P_b(s(0)) \wedge \neg X(s(0)) \wedge \neg X(s^2(x)) \wedge P_c(s^2(0)) \wedge X(s^3(x)))] \\
& & \wedge \\
& & \forall x(X(x) \rightarrow \\
& & (P_a(x) \wedge P_b(s(x)) \wedge \neg X(s(x)) \wedge X(s^2(x))) \\
& & \vee \\
& & (P_b(x) \wedge P_b(s(x)) \wedge \neg X(s(x)) \wedge P_c(s^2(x)) \wedge \neg X(s^2(x)) \wedge X(s^3(x)))) \\
& &)
\end{array}
$$

Aufgabe 11.2.16

1. $F_1 = \forall x(P_a(x) \rightarrow (P_b(s(x)) \vee P_b(s(s(x))) \vee P_b(s(s(s(x))))))$
2. $F_2 = \forall x \exists y(x < y \wedge P_a(y)) \rightarrow \forall x \exists y(x < y \wedge P_b(y))$
3. $F_3 = \exists x \forall y(x < y \rightarrow P_x \vee P_z) \rightarrow \forall x \exists y(x < y \wedge P_y \vee P_z)$

Aufgabe 11.2.17
Nach Satz 11.25 gilt

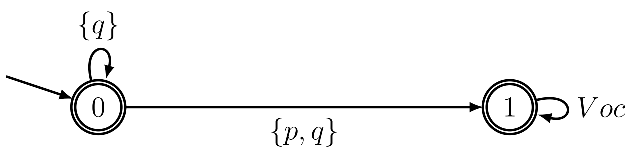
$$L^\omega(\mathcal{B}) = \bigcup_{i=1}^{k} J_i K_i^\omega$$

für reguläre Mengen $J_i, K_i \subseteq V^*$. Es gibt, wie aus der Theorie der endlichen Automaten bekannt ist, reguläre Ausdrücke t_i, s_i mit $V(t_i) = J_i$, $V(s_i) = K_i$, siehe z.B. 11.16. Somit leistet $t = \Sigma_{i=1}^k t_i s_i^\omega$ das gewünschte.
Aufgabe 11.2.17
Teil 1 ist einfach. Teil 2 folgt direkt aus 1.
Teil 3: Man sieht leicht, daß in $fin(t)$ der $^\omega$ Operator nicht vorkommt, also

$fin(t)$ ein (normaler) regulärer Ausdruck ist.
Teil 4 läßt sich durch strukturelle Induktion über t beweisen. Dazu braucht man Teil 3 dieser Aufgabe und die Abschlußeigenschaften von Büchi Automaten.

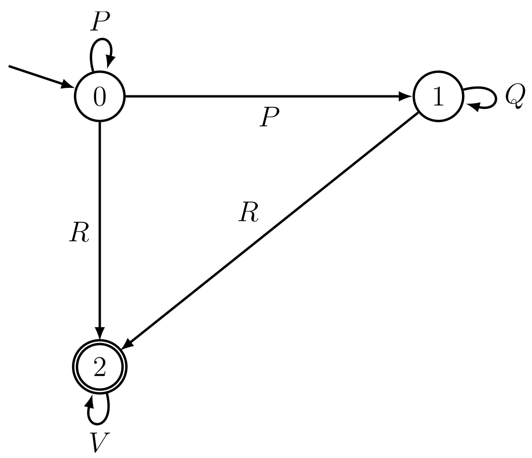
13.19 Lösungen zu Abschnitt 12.2 Büchi Automaten und LTL

Aufgabe 12.2.1 Das Vokabular Voc des Automaten $\mathcal{A}_V$ besteht aus allen Teilmengen der aussagenlogischen Atome $\{p, q\}$, also $Voc = \{\{\}, \{p\}, \{q\}, \{p, q\}\}$. Der gesuchte Automat ist:

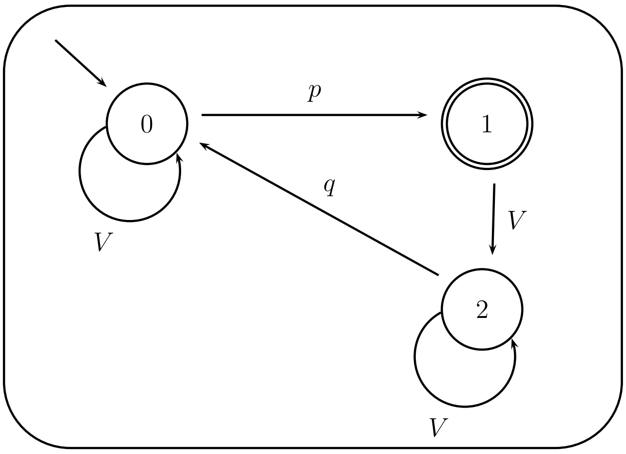


Aufgabe 12.2.2 Das Vokabular V für $\mathcal{B}_{UU}$ ist wie immer $V = 2^{\Sigma}$. Wir definieren die Teilmengen von V:

$$
\begin{array}{lcl}
P & = & \{a \in V \mid p \in a\} \\
Q & = & \{a \in V \mid q \in a\} \\
R & = & \{a \in V \mid r \in a\}
\end{array}
$$

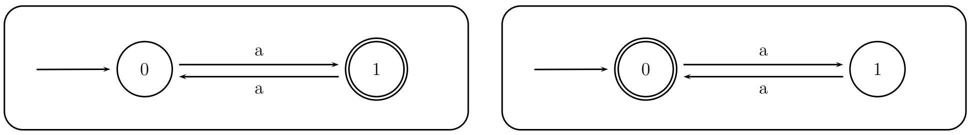


Aufgabe 12.2.3



Aufgabe 12.2.4

zu 1 Wir betrachten die Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$ aus Abbildung 13.3



(a) $\mathcal{A}_1$ (b) $\mathcal{A}_2$

Abbildung 13.3: Automaten für Gegenbeispiel

über dem Kantenalphabet $V = \{a\}$. Es gilt offensichtlich $L^\omega(\mathcal{A}_1) = L^\omega(\mathcal{A}_2) = \{a^\omega\}$. In dem Produktautomaten $\mathcal{A}_1 \times \mathcal{A}_2$ ist keine Endzustand vom Anfangszustand aus erreichbar, also $L^\omega(\mathcal{A}_1 \times \mathcal{A}_2) = \emptyset$.

zu 2 Ist $((s_{1i}, s_{2i}))_{i\geq 0}$ eine akzeptierende Berechnungsfolge für $\mathcal{A}_1 \times \mathcal{A}_2$, dann gibt es einen Endzustand $t_f = (t_{1f}, t_{2f})$ dieses Automaten, so daß $t_f = (s_{1i}, s_{2i})$ für unendlich viele i gilt. Da nach Definition der Endzustände in Produktautomaten t_{1f} und t_{2f} Endzustände von $\mathcal{A}_1$ beziehungsweise $\mathcal{A}_2$ sein müssen, sind $(s_{1i})_{i\geq 0}$ und $(s_{2i})_{i\geq 0}$ akzeptierende Berechnungsfolgen von $\mathcal{A}_1$ beziehungsweise $\mathcal{A}_2$.

zu 3 Wir benutzen die Notation aus Teil 2. Wenn $(s_{1i})_{i\geq 0}$ und $(s_{2i})_{i\geq 0}$ akzeptierende Berechnungsfolgen für $\mathcal{A}_1$ beziehungsweise $\mathcal{A}_2$ sind, dann gibt es einen Endzustand t_{2f}, so daß die Menge $Inf = \{i \mid s_{2i} = t_{2f}\}$ unendlich ist. Nach den Voraussetzungen für diese Aufgabe ist für jedes $i \in Inf$ $(s_{1i}, s_{2i}) = (s_{1i}, t_{2f}$ ein Endzustand des Produktautomaten und somit die Folge $((s_{1i}, s_{2i}))_{i\geq 0}$ eine akzeptierende Berechungsfolge des Produktautomaten.

∎

13.20 Lösungen zu Abschnitt 5.7 PL Anmerkungen

Aufgabe 5.7.1 Wir überlegen uns zunächst, daß es für jedes n eine Σ-Formel A_n gibt, so daß für alle (D, I) mit $(D, I) \models A_n$ gilt A_n hat mindestens n Elemente. Diese Aussage kann man eigentlich direkt hinschreiben:

$$A_n = \exists x_0 \ldots \exists x_{n-1}(\bigwedge_{0 \leq i < j < n} \neg x_i \doteq x_j)$$

Da Σ leer ist, hat die Interpretationsfunktion I einen leeren Definitionsbereich und wir lassen I für den Rest dieser Lösungsbeschreibung weg. Gilt $D \models A_n$ für alle n, dann muß D unendlich sein. Die Behauptung der Übungsaufgabe ist nun, daß es nicht möglich ist, die unendlich vielen Formeln A_n durch eine äquivalent zu ersetzen. Angenommen, es wäre doch möglich und $D \models A$ gilt genau dann, wenn D unendlich ist. Dann gilt $\{A_n \mid 0 < n\} \vdash A$. Nach dem Endlichkeitssatz gibt es schon ein m mit $\{A_n \mid 0 < n < m\} \vdash A$. Sei D_m eine Menge mit genau m Elementen. Dann gilt für alle $0 < n < m$ sicherlich $D \models A_n$. Also gilt auch $D \models A$. Damit hätten wir aber ein endliches Modell für A gefunden im Widerspruch zur Annahme.

Aufgabe 5.7.2 TBD

13.21 Lösungen zu Abschnitt 5.8 Axiomatisierung von Datentypen

Aufgabe 5.8.1 Beide Beweis benutzen Induktion,

zu 1 Im Induktionsanfang ist $0 + 1 \doteq 1 + 0$ zu zeigen. Da folgt aus Teil 1 von Lemma 5.92.

Im Induktionsschritt muß von $x + 1 \doteq 1 + x$ geschlossen werden auf $(x + 1) + 1 \doteq 1 + (x + 1)$.

$$\begin{array}{rcll} (x+1)+1 & \doteq & (1+x)+1 & \text{Induktionsvoraussetzung} \\ & \doteq & 1+(x+1) & \text{Lemma 5.92.2} \end{array}$$

zu 2 Induktion läuft über die Variable x. Der Induktionsanfang $0 + y) \doteq y + 0$ folgt aus Lemma 5.91 Teil 1.

Im Induktionsschritt müssen wir von $x+y \doteq y+x$ auf $(x+1)+y \doteq y+(x+1)$ schließen

$$\begin{array}{rcll} (x+1)+y & \doteq & x+(1+y) & \text{5.92} \\ & \doteq & x+(y+1) & \text{Teil 1 dieses Lemmas} \\ & \doteq & (x+y)+1 & \text{5.92} \\ & \doteq & (y+x)+1 & \text{Induktionsvoraussetzung} \\ & \doteq & y+(x+1) & \text{5.92} \end{array}$$

■

Aufgabe 5.8.2

zu 1 Auf $x \leq y$ und $y \leq z$ folgt nach der Definition der Ordnung $x+u \doteq y$ und $y+w \doteq z$. Die erste Gleichung in die zweite eingesetzt ergibt, nach der Anwendung der Assoziativität der Addition, $x+(u+w) \doteq z$ und damit $\exists v(x+v \doteq z)$.

zu 2 Hier bietet sich Induktion nach u an. Für $u = 0$ folgt $x \doteq y$ aus Axiom 3. Im Induktionsschritt gehen wir von der Hypothese $x+u \doteq y+u \rightarrow x \doteq y$ aus und müssen $x+(u+1) \doteq y+(u+1) \rightarrow x \doteq y$ zeigen. Die linke Seite der Implikation liefert mit Axiom 4 zunächst $(x+u)+1 \doteq (y+u)+1$. Mit Axiom 2 ergibt das $x+u \doteq y+u$. Mit der Induktionshypothese erhalten wir, wie gewünscht, $x \doteq y$.

zu 3 Die Aussage läßt sich durch Induktion über v zeigen. Im Induktionsanfang, $v \doteq 0$, ist $u+0 \doteq 0 \rightarrow u \doteq 0 \wedge v \doteq 0)$ zu zeigen. Wegen Axiom 3 folgt aus der linken Seite der Implikation $u+0 \doteq 0+0$. Woraus mit Axiom 2 folgt $u \doteq 0$. Im Induktionsschritt ist $u+(v+1) \doteq 0 \rightarrow u \doteq 0 \wedge v \doteq 0)$ zu zeigen. Die linke Seite kann geschrieben werden als $(u+v)+1 \doteq 0$, was nach Axiom 1 nicht der Fall sein kann. Somit ist die Implikation trivialerweise wahr. Auf die Induktionshypothese wußte wir hier garnicht zurückgreifen.

zu 4 In der folgenden Argumentation erwähnen wir Anwendungen der Assoziativität (Lemma 5.92) und Kommutativität (Aufgabe 5.8.1) nicht mehr. Aus $x \leq y$ und $y \leq x$ folgt nach Definition der Ordnung $x+u \doteq y$ und $y+w \doteq x$, was zusammengenommen $x+(u+w) \doteq x$ liefert. Wegen Axiom 3 können wir auch $x+(u+w) \doteq x+0$ schreiben. Aus Teil 2 dieser Aufgabe folgt $u+w \doteq 0$. Aus Teil 3 dieser Aufgabe schlieplich $u \doteq w \doteq 0$ Also $x \doteq y$.

■

Aufgabe 5.8.3 **zu 1** Induktion nach der Variablen x.
Im Induktionsanfang is $x * 0 \doteq 0$ zu zeigen, was direkt aus Axiom 5 folgt.

Im Induktionsschritt müssen wir von der Induktionshypothese $0 * x \doteq 0$ schließen auf $0 * (x + 1) \doteq 0$. Das geht so

$$\begin{array}{rcll} 0 * (x+1) & \doteq & 0 * x + 0 & \text{Axiom 6} \\ & \doteq & 0 + 0 & \text{Induktionshypothese} \\ & \doteq & 0 & \text{Axiom 3} \end{array}$$

zu 2 Hierzu brauchen wir, zur Abwechslung, mal keine Induktion.

$$\begin{array}{rcll} x * 1 & \doteq & x * (0+1) & \text{Lemma 5.91(1)} \\ & \doteq & x * 0 + x & \text{Axiom 6} \\ & \doteq & 0 + x & \text{Teil 1 dieser Aufgabe} \\ & \doteq & x & \text{Lemma 5.91(1)} \end{array}$$

zu 3 Hier brauchen wir wieder das Induktionsschema.
Im Anfangsfall ist $1 * 0 \doteq 0$ zu zeigen, was aus Axiom 5 folgt.
Gelte jetzt $1 * x \doteq x$ und $1 * (x + 1) \doteq x + 1$ ist zu zeigen. Das geht einfach:

$$\begin{array}{rcll} 1 * (x+1) & \doteq & 1 * x + 1 & \text{Axiom 6} \\ & \doteq & x + 1 & \text{Induktionshypothese} \end{array}$$

■

Aufgabe 5.8.4 **zu 1** Induktion nach z
Der Induktionsanfang, $\forall x \forall y (x * (y + 0) \doteq x * y + x * 0)$, folgt aus den Axiomen 3 und 5.
Im Induktionsschritt ist $\forall x (x * (y + (z + 1)) \doteq x * y + x * (z + 1))$ zu zeigen.

$$\begin{array}{rcll} (x * (y + (z+1)) & \doteq & (x * ((y + z) + 1)) & \text{Assoz. von } + \text{, Lemma 5.92} \\ & \doteq & x * (y + z) + x & \text{Axiom 6} \\ & \doteq & (x * y + x * z) + x & \text{Ind.Hyp} \\ & \doteq & x * y + (x * z + x) & \text{Assoz. von } + \\ & \doteq & x * y + x * (z + 1) & \text{Axiom 6} \end{array}$$

zu 2 Induktion nach z
Der Induktionsanfang, $\forall x \forall y ((x * y) * 0 \doteq x * (y * 0))$, folgt durch dreimalige Anwendung von Axiom Axiom 5.
Im Induktionsschritt ist $\forall x \forall y ((x * y) * (z + 1) \doteq x * (y * (z + 1)))$ zu zeigen.

$$\begin{array}{rcll} (x * y) * (z+1) & \doteq & (x * y) * z + x * y & \text{Axiom 6} \\ & \doteq & x * (y * z) + x * y & \text{Ind.Hyp} \\ & \doteq & x * (y * z + y) & \text{Teil 1 des Lemmas} \\ & \doteq & x * (y * (z + 1))) & \text{Axiom 6} \end{array}$$

■

Aufgabe 5.8.5
zu 1 Induktion nach y
Der Induktionsanfang $\forall x((x+1)*0 \doteq x*0+0)$ folgt aus den Axiomen 3 und 5.
Im Induktionsschritt ist $(x+1)*(y+1) \doteq x*(y+1)+(y+1)$ zu zeigen.

$$\begin{array}{rcll} (x+1)*(y+1) & \doteq & (x+1)*y+(x+1) & \text{Axiom 6} \\ & \doteq & (x*y+y)+(x+1) & \text{IndHyp} \\ & \doteq & (x*y+x)+(y+1) & \text{Assoz. u. Komm. von } + \\ & & & \text{Lemma 5.92 und} \\ & & & \text{Aufgabe 5.8.1(2)} \\ & \doteq & x*(y+1)+(y+1) & \text{Axiom 6} \end{array}$$

zu 2 Induktion nach y
Der Induktionsanfang, $\forall x(x*0 \doteq 0*x)$, folgt aus Axiom 5, $x*0 \doteq 0$ und Aufgabe 5.8.3(1), $0*x \doteq 0$.
Im Induktionsschritt ist $\forall x(x*(y+1) \doteq (y+1)*x)$ zu zeigen.

$$\begin{array}{rcll} x*(y+1) & \doteq & x*y+x & \text{Axiom 6} \\ & \doteq & y*x+x & \text{IndHyp} \\ & \doteq & (y+1)*x) & \text{Teil 1 des Lemmas} \\ & & & \text{mit } y \rightsquigarrow x, x \rightsquigarrow y \end{array}$$

■

13.22 Lösungen zu Abschnitt 5.9 PL Anwendungen

Aufgabe 5.9.1 Die Formalisierungen findet man in den .key-Dateien SET001.key bis SET004.key. Die vierte Gleichung $A \cup (B \setminus C) = (A \cup B) \setminus ((A \cap B) \setminus C)$ ist nicht korrekt. Man betrachte dazu Mengen A,B,C, so daß ein Element a existiert mit $a \notin A$, $a \in B$ und $a \in C$. Dann ist a eine Element der rechten Seite, aber nicht der linken Seite.

Aufgabe 5.9.2 Siehe die zugehörigen .key-Dateien.

Aufgabe 5.9.3 Siehe die zugehörigen .key-Dateien.

Aufgabe 5.9.4 Siehe die zugehörigen .key-Datei.

Aufgabe 5.9.5
zu (1): Zu gegebenen Zahlen n (Zähler, engl. numerator), d (Nenner, engl. denominator) ist q das Ergebniss der ganzzahligen Division von n durch d. In Formeln:

$$
\begin{array}{rcll}
jdiv(n,d) = q & \leftrightarrow & & d = 0 \quad \vee \\
& & & (n \geq 0 \\
& & \wedge & d > 0 \\
& & \wedge & q \geq 0 \\
& & \wedge & n \geq d * q \\
& & \wedge & d * (q+1) > n) \quad \vee \\
& & & (n \leq 0 \\
& & \wedge & d < 0 \\
& & \wedge & q \geq 0 \\
& & \wedge & n \leq d * q \\
& & \wedge & d * (q+1) < n) \quad \vee \\
& & & (n \geq 0 \\
& & \wedge & d < 0 \\
& & \wedge & q \leq 0 \\
& & \wedge & n \geq d * q \\
& & \wedge & d * (q-1) > n) \quad \vee \\
& & & (n \leq 0 \\
& & \wedge & d > 0 \\
& & \wedge & q \leq 0 \\
& & \wedge & n \leq d * q \\
& & \wedge & d * (q-1) < n)
\end{array}
$$

zu (2):

Aufgabe 5.9.6
Teil 1

$$
\begin{array}{rcl}
div(n,d) = q & \leftrightarrow & d = 0 \ \vee \\
& & (q * d \leq n \ \wedge \\
& & ((d \geq 0) \wedge (q * d \geq 1 + n - d) \vee \\
& & (d < 0) \wedge (q * d \geq 1 + n_0 + d_0)))
\end{array}
$$

Teil 2
$jdiv(n,d) = \texttt{if}\ (n \geq 0)\ \texttt{then}\ div(n,d)\ \texttt{else}\ -div(-n,d)$
Siehe Aufgabe 4.3.16 zur Notation.

13.23 Lösungen zu Abschnitt 10.1 PL2

Aufgabe 10.1.2

13.24 Lösungen zu Abschnitt 12.4 Modelprüfung

Aufgabe

13.25 Lösungen zu Abschnitt 12.5 Bounded Model Checking

Aufgabe 12.5.1

$$
\begin{array}{lcl}
[\Box\Diamond\neg p]_1^1 & \equiv & [\Diamond\neg p]_1^1 \wedge [\Diamond\neg p]_1^2 \\
[\Diamond\neg p]_1^1 & \equiv & [\neg p]_1^1 \vee [\neg p]_1^2 \\
 & \equiv & \neg p^1 \vee \neg p^2 \\
[\Diamond\neg p]_1^2 & \equiv & [\neg p]_1^2 \vee [\neg p]_1^1 \\
 & \equiv & \neg p^2 \vee \neg p^1 \\
[\Box\Diamond\neg p]_2^1 & \equiv & [\Diamond\neg p]_2^2 \wedge [\Diamond\neg p]_2^1 \\
[\Diamond\neg p]_2^2 & \equiv & [\neg p]_2^2 \\
 & \equiv & \neg p^2 \\
[\Diamond\neg p]_2^1 & \equiv & [\neg p]_2^1 \vee [\neg p]_2^2 \\
 & \equiv & \neg p^1 \vee \neg p^2
\end{array}
$$

Insgesamt also $[\Box\Diamond\neg p]_1^1 \leftrightarrow \neg p^1 \vee \neg p^2$ and $[\Box\Diamond\neg p]_2^1 \leftrightarrow \neg p^2$. Die restlichen Formeln können wir unverändert aus dem Beispiel in der Konstruktion zu Lemma 12.23 übernehmen.

$$
\begin{array}{l}
\neg c^1 \wedge \\
(\neg c^1 \wedge \neg c^2) \vee (\neg c^1 \wedge c^2 \wedge p^1) \vee (c^1 \wedge c^2 \wedge p^1) \\
\wedge \\
(\neg c^2 \wedge \neg c^3) \vee (\neg c^2 \wedge c^3 \wedge p^2) \vee (c^2 \wedge c^3 \wedge p^2) \\
\wedge \quad (\\
((c^1 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge (\neg p^1 \vee \neg p^2) \\
\vee \\
((c^2 \leftrightarrow c^3) \wedge (c^2 \vee c^3) \wedge \neg p^2) \\
\qquad)
\end{array}
$$

Das läßt sich weiter äquivalent umformen zu

$$
\begin{array}{l}
\neg c^1 \wedge \\
\neg c^2 \vee p^1 \\
\wedge \\
(\neg c^2 \wedge \neg c^3) \vee (\neg c^2 \wedge c^3 \wedge p^2) \vee (c^2 \wedge c^3 \wedge p^2) \\
\wedge \quad (\\
(c^2 \wedge \neg c^3 \wedge \neg p^2) \\
\vee \\
(c^2 \wedge c^3 \wedge \neg p^2) \\
\qquad)
\end{array}
$$

Jetzt kann man sehen. daß diese Formel nicht erfüllbar ist. Das stimmt auch überein mit der automatentheoretischen Semantik, denn der Automat $\mathcal{A}_{dbp}$ akzeptiert keine Berechungsfolge, die $\Box\Diamond\neg p$ erfüllt.

Literaturverzeichnis

[AvH04] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer.* Cooperative Information Systems. MIT Press, April 2004.

[B9̈8] Egon Börger. *Berechenbarkeit, Komplexität, Logik.* Vieweg Verlagsgesellschaft, 1998.

[Bar77] Jon Barwise, editor. *Handbook of Mathematical Logic.* North-Holland Publ.Co., 1977.

[Bar84] H. Barendregt. *The Lambda Calculus*, volume 103 of *Studies in Logic.* North-Holland, Amsterdam, 1984.

[BCC$^+$03] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers,*, 58, 2003.

[BCC$^+$05] Lilian Burdy, Yoonsik Cheon, David R. Cok, Michael D. Ernst, Joseph R. Kiniry, Gary T. Leavens, K. Rustan M. Leino, and Erik Poll. An overview of jml tools and applications. *STTT*, 7(3):212–232, 2005.

[BCCZ99] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *Proc. of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, LNCS. Springer-Verlag, 1999.

[BH92] K. H. Bläsius and H.-J. Bürckert (Hrsg.). *Deduktionssysteme. Automatisierung des logischen Denkens. 2.Auflage.* R. Oldenbourg Verlag, 1992.

[Bir35] Garrett Birkhoff. On the structure of abstract algebras. *Proc.Cambrigde Phil.Soc.*, 31:433–454, 1935.

[BJL88] Blair, Robert C. Jeroslow, and Lowe. Some results and experiments in p rogramming techniques for propositional logic. *Computers and Operations Research*, 13(13):633–645, 1988.

[BL92] Matthias Baaz and Alexander Leitsch. Complexity of resolution proofs and function introduction. *Annals of Pure and Applied Logic*, 57:181–215, 1992.

[Boo87] R. Book. Thue systems as rewriting systems. *J. of Symbolic Computation*, 1 & 2:39–68, 1987.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

[Buc85] Bruno Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Recent Trends in Multidimensional Systems Theory*. Reidel Publ.Co., 1985.

[Bü97] Reinhard Bündgen. *Termersetzungssysteme*. Vieweg Verlag, 1997.

[Car58] Lewis Carrol. *Symbolic Logic*. Dover Publications NY, 1958.

[CFF+01] F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M.Y. Vardi. Benefits of bounded model checking at an industrial setting. In *Proc. 12th Intl. Conference on Computer Aided Verification (CAV'01)*, LNCS, pages 436–453. Springer, 2001.

[Cha03] Patrice Chalin. Improving JML: For a safer and more effective language. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *Proc. Formal Methods Europe, Pisa, Italy*, volume 2805 of *LNCS*, pages 440–461. Springer-Verlag, 2003.

[Chu56] Alonzo Church. *Introduction to Mathematical Logic*, volume 1. Princeton University Press, 1956.

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, London, 1973.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *STOC*, pages 151–158. ACM, 1971.

[CT51] L.H. Chin and A. Tarski. Distributive and modular laws in the arithmetic of relation algebras. *University of California Publications in Mathematics, New Series*, 1:341–384, 1951.

[dB80] J. de Bakker. *Mathematical Theory of Program Correctness*. International Series in Computer Science. Prentice Hall, 1980.

[Ded87] Richard Dedekind. *Was sind und was sollen die Zahlen*. Friedrich Vieweg, Braunschweig, 1887. online version at `http://echo.mpiwg-berlin.mpg.de/MPIWG:01MGQHHN`.

[DGHP99] Marcello D´Agostino, Dov .M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In van Leeuwen [vL90], pages 243–320.

[DJK04] Wolfgang Dostal, Mario Jeckle, and Werner Kriechbaum. Semantik und web services: Vokabulare und ontologien. *Java Spektrum*, 3:51 – 54, 2004.

[dM64] A. de Morgan. On the syllogism, no.iv, and on the logic of relations. *Transactions of the Cambridge Philosophical Society*, 10:331–358, 1864.

[DP70] D.E.Knuth and P.B.Bendix. Simple word problems in universal algebra. In J.Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.

[D'S10] Vijay D'Silva. Propositional interpolation and abstract interpretation. In Andrew D. Gordon, editor, *ESOP*, volume 6012 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2010.

[Ede92] Elmar Eder. *Relative Complexities of First-Order Calculi*. Artificial Intelligence. Vieweg Verlag, 1992.

[EFT92] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Einführung in die Mathematische Logik. 3., vollständig überarbeitete und erweiterte Auflage*. BI Wissenschaftsverlag, 1992.

[EFT07] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Einführung in die mathematische Logik (5. Aufl.)*. Spektrum Akademischer Verlag, 2007.

[EMC$^+$99] Hartmut Ehrig, Bernd Mahr, Felix Cornelis, Martin Große-Rhode, and Philip Zeitz. *Mathematisch-strukturelle Grundlagen der Informatik*. Springer -Lehrbuch. Springer, 1999.

[Fel78] Walter Felscher. *Naive Mengen und abstrakte Zahlen I.* B.I.Wissenschaftsverlag, 1978.

[Fit90] Melvin C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.

[FM99] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*, volume 277 of *Synthese Library*. Kluwer Academic Publishers, 1999.

[GA02] Steven Givant and Hajnal Andréka. Groups and algebras of binary relations. *The Bulletin of Symbolic Logic*, 8(1):38–64, 2002.

[Gal86] Jean H. Gallier. *Logic for Computer Science: Foundations of Automated Theorem Proving*. Harper and Row, New York, 1986.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39:176–210, 1935.

[Gen69] Gerhard Gentzen. Investigation into logical deduction. In E.Szabo, editor, *The Collected Papers of Gerhzard Gentzen*, pages 68–131. North-Holland, 1969.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman and Co, 1979.

[Gol82] Robert Goldblatt. *Axiomatising the logic of computer programming*, volume 130 of *LNCS*. Springer-Verlag, 1982.

[Gö31] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik und Physik*, 38:173—-198, 1931.

[Har09] John Harrison. *Handbook of Pratical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[HK89] D. Hofbauer and R. D. Kutsche. *Grundlagen des maschinellen Beweisens*. Vieweg, 1989.

[HKRS08] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web - Grundlagen*. eXamen.press. Springer-Verlag Berlin Heidelberg, 2008.

[HMT71] L. Henkin, T. Monk, and A. Tarski. *Cylindrical Algebras, Part I.* North Holland Publ. Co, 1971.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[Hoa83] C. A. R. Hoare. An axiomatic basis for computer programming (reprint). *Commun. ACM*, 26(1):53–56, 1983.

[Hoa09] C. A. R. Hoare. Viewpoint - retrospective: an axiomatic basis for computer programming. *Commun. ACM*, 52(10):30–32, 2009.

[Hol04] G.J. Holzmann. *The Spin Model Checker, Primer and Reference Manual.* Addison-Wesley, Reading, Massachusetts, 2004.

[Hoo88] John N. Hooker. A quantitative approach to logical inference. *Decision Support Systems*, 4:45–69, 1988.

[HR00] Michael R. Huth and Mark D. Ryan. *Logic in Computer Science. Modelling and reasoning about systems.* Cambridge University Press, 2000.

[HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ–rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13:211–221, 1994.

[HU79] J. E. Hopcroft and J. D. Ullmann. *Introduction to Automata Theory.* Addison-Weseley, 1979.

[Hun33a] E. Huntington. Boolean algebra. a correction. *Transactions of the AMS*, 35:557–558, 1933.

[Hun33b] E. Huntington. New sets of independent postulates for algebra of logic. *Transactions of the AMS*, 35:274–304, 1933.

[HW90] B. Heinemann and K. Weihrauch. *Logik für Informatiker.* B. G. Teubner Verlag, 1990.

[JS74] N.D. Jones and A.L. Selman. Turing machines and the spectra of first-order formulas with equality. *J. Symbolic Logic*, 39:139–150, 1974.

[Kfo88] Robert A. Molland Michael A. Arbiband A. J. Kfoury. *An Introduction to Formal Language Theory.* Texts and Monographs in Computer Science. Springer-Verlag, 1988.

[Lan72] Serge Lang. *Algebra.* Addison-Wesley, fifth edition, 1972.

[LBR99] Gary T. Leavens, Albert L. Baker, and Clyde Ruby. *Behavioral Specification for Businesses and Systems*, chapter Chapter 12: JML: A Notation for Detailed Design, pages 175–188. Kluwer Academic Publishers, 1999.

[LBR03] Gary T. Leavens, Albert L. Baker, and Clyde Ruby. Preliminary design of JML: A behavioral interface specification language for Java. Technical Report 98-06y, Iowa State University, Department of Computer Science, 2003. Revised June 2004.

[Lie06] Hans Liebig. *Logischer Entwurf digitaler Systeme (4. Aufl.).* Springer, 2006.

[Lov78] Donald W. Loveland. *Automated Theorem Proving. A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science.* North-Holland, 1978.

[LPC+11] Gary T. Leavens, Erik Poll, Curtis Clifton, Yoonsik Cheon, Clyde Ruby, David Cok, Peter Müller, Joseph Kiniry, Patrice Chalin, and Daniel M. Zimmermanand Werner Dietl. *JML Reference Manual*, February 2011.

[LR13] Richard J. Lipton and Kenneth W. Regan. *People, Problems, and Proofs - Essays from Gödel's Lost Letter: 2010.* Springer, 2013.

[McC96] William McCune. Robbins algebras are boolean. *Association for Automated Reasoning Newsletter*, 35:1–3, 1996.

[McM03] K.L. McMillan. Interpolation and sat-based model checking. In F. Somenzi and W. Hunt, editors, *Computer Aided Verification (CAV03)*, 2003.

[Men87] E. Mendelson. *Introduction to Mathematical Logic.* Wardsworth, Pacific Grove, Calif., 1987.

[Mey91] Bertrand Meyer. *Eiffel: The Language.* Prentice-Hall, 1991.

[Mey97] Bertrand Meyer. *Object-Oriented Software Construction, 2nd Edition.* Prentice-Hall, 1997.

[Min90] G. Mints. Gentzen-type systems and resolution rules, part 1: Propositional logic. In *Proc. COLOG-88, Tallin*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer, 1990.

[Mon76] J. Donald Monk. *Mathematical Logic*. Springer-Verlag, 1976.

[Mos86] Ben Moszkowski. *Executable Temporal Logic Programs*. Cambride University Press, Cambridge England, 1986.

[Pea89] Guiseppe (Ioseph) Peano. *Arithmetices principia: nova methodo exposita*. Augustae Taurinorum, 1889. online version at http://www.archive.org/stream/arithmeticespri00peangoog#page/n6/mode/2up.

[PH77] Jeff Paris and Leo Harrington. A mathematical incompleteness in peano arithmetik. In Barwise [Bar77], chapter D8, pages 1133–1142.

[Pie02] Benjamin C . Pierce. *Types and Programming Languages*. The MIT Press, 2002.

[Pop94] Sally Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[Ram29] F. P. Ramsey. On a problem of formal logic. *Proc. of London Math.Soc.*, 30:264–286, 1929.

[SA91] Volker Sperschneider and G. Antoniou. *Logic, a foundation for computer science*. Addison-Wesley, 1991.

[Sch95] E. Schröder. *Vorlesungen über die Algebra der Logik, vol.III, Algebra und Logik der Relative*. B.G. Teubner, 1895. Nachdruck: Chelsea Publishing Company, New York, 1966.

[Sch92] Peter H. Schmitt. *Theorie der logischen Programmierung*. Springer Verlag, 1992.

[Sch00] Uwe Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, 2000.

[SGS+92] V. Stavridou, J. A. Goguen, A. Stevens, S. M. Eker, S. N. Aboneftis, and K. M. Hobley. FUNNEL and 2OBJ: Towards an integrated hardware design envirinment. In Stavridou et al. [SMB92], pages 197–223.

[Sha38] C. E. Shannon. A symbolic analysis of relay and switching circuits. *AIEE Transactions*, 67:713–723, 1938.

[SMB92] V. Stavridou, T. F. Melham, and R. T. Boute, editors. *Theorem provers in Circuit Design, Proceed. of Internat. Conf. Nijmegen.* IFIP Transactions A-10. IFIP, 1992.

[Smu68] Raymond Smullyan. *First-Order Logic.* Springer, 1968.

[Smu95] Raymond M. Smullyan. *First-Order Logic.* Dover Publications, New York, second corrected edition, 1995. First published in 1968 by Springer.

[Sta88] Richard Stallman. GNU Emacs Manual, sixth edition, emacs version 18. Technical report, Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, 1988.

[Tar41] Alfred Tarski. On the calculus of relations. *J. of Symbolic Logic*, 6:73–89, 1941.

[Tar72] R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1:146–160, 1972.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In van Leeuwen [vL90], pages 135–192.

[Tse68] G.S. Tseitin. On the complexity of derivations in propositional calculus. In A.O.Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, 1968.

[Tse83] G.S. Tseitin. On the complexity of derivations in propositional calculus (reprint). In J.Siekmann and G.Wrightson, editors, *Automation of Reasoning: Classical Papers on Computational Logic.* Springer, 1983.

[TZ71] G. Takeuti and W.M. Zaring. *Introduction to Axiomatic Set Theory.* Undergraduate Texts in Mathematics. Springer Verlag, 1971.

[vL90] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science. Vol. B : Formal Models and Semantics.* Elsevier, Amsterdam, 1990.

[Win90] S. Winker. Robbins algebra: Conditions that make a near-boolean algebra boolean. *J. Automated Reasoning*, 6(4):465–489, 1990.

[Win92] S. Winker. Absorption and idempotency criteria for a problem in near-boolean algebras. *J. Algebra*, 153(2):414–423, 1992.