Formale Systeme

Prädikatenlogik: Syntax

Prof. Dr. Bernhard Beckert | WS 2010/2011

KIT – INSTITUT FÜR THEORETISCHE INFORMATIK

Einführendes Beispiel

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

Prädikatenlogische Formalisierung:

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

Logische Zeichen: $\forall x$, $\to$

Anwendungsabhängiges Vokabular:
Mensch(.), sterblich(.), Sokrates

Einführendes Beispiel

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

Prädikatenlogische Formalisierung:

$$\forall x(\text{Mensch}(x) \rightarrow \text{sterblich}(x))$$
$$\text{Mensch}(\text{Sokrates})$$
$$\text{sterblich}(\text{Sokrates})$$

Logische Zeichen: $\forall x$, $\rightarrow$

Anwendungsabhängiges Vokabular:
Mensch(.), sterblich(.), Sokrates

Einführendes Beispiel

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

Prädikatenlogische Formalisierung:

$$\forall x(\mathrm{Mensch}(x) \rightarrow \mathrm{sterblich}(x))$$
$$\mathrm{Mensch}(\mathrm{Sokrates})$$
$$\mathrm{sterblich}(\mathrm{Sokrates})$$

Logische Zeichen: $\forall x$, $\rightarrow$

Anwendungsabhängiges Vokabular:
Mensch(.), sterblich(.), Sokrates

Einführendes Beispiel

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

Prädikatenlogische Formalisierung:

$$\forall x(\mathrm{Mensch}(x) \rightarrow \mathrm{sterblich}(x))$$
$$\mathrm{Mensch}(\mathrm{Sokrates})$$
$$\mathrm{sterblich}(\mathrm{Sokrates})$$

Logische Zeichen: $\forall x$, $\rightarrow$

Anwendungsabhängiges Vokabular:
Mensch(.), sterblich(.), Sokrates

Einführendes Beispiel

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

Prädikatenlogische Formalisierung:

$$\forall x(\text{Mensch}(x) \rightarrow \text{sterblich}(x))$$
$$\text{Mensch}(\text{Sokrates})$$
$$\text{sterblich}(\text{Sokrates})$$

Logische Zeichen: $\forall x$, $\rightarrow$

Anwendungsabhängiges Vokabular:
Mensch(.), sterblich(.), Sokrates

Einführendes Beispiel 2

Die Java Card Platform Specification v2.2.1
(siehe http://java.sun.com/products/javacard/specs.html)
enthält u.a. die Klasse

public class Util
extends Object

mit der Methode `arrayCompare`:

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.

Method Detail `arrayCompare`

```
public static final byte arrayCompare (byte[] src,
                                       short srcOff,
                                       byte[] dest,
                                       short destOff,
                                       short length)
                               throws  ArrayIndexOutOfBoundsE,
                                       NullPointerException
```

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).

Method Detail `arrayCompare`

- 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) a `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.

Method Detail `arrayCompare`

- 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) a `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.

Method Detail `arrayCompare`

- 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) a `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.

Method Detail `arrayCompare`

- 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) a `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.

Formalisierung: Normales Verhalten

$$s \neq null \quad \wedge \quad sO \geq 0 \quad \wedge \quad sO + l \leq size(s) \quad \wedge$$
$$d \neq null \quad \wedge \quad dO \geq 0 \quad \wedge \quad dO + l \leq size(d) \quad \wedge \quad l \geq 0$$
$$\rightarrow$$

$$\begin{array}{ll}
\neg excThrown(E) & \wedge \\
(res = -1 \quad \vee \quad res = 0 \quad \vee \quad res = 1) & \wedge \\
(subSeq(s, sO, sO + l) = subSeq(d, dO, dO + l) \rightarrow res = 0) & \wedge \\
(\exists i{:}Int(1 \leq i \wedge i \leq l \wedge (at(s, sO + i) < at(d, dO + i) \quad \wedge & \\
\forall j{:}Int(1 \leq j \wedge j < i \rightarrow at(s, sO + j) = at(d, dO + j))) & \\
) \rightarrow res = -1) & \wedge \\
(\exists i{:}Int(1 \leq i \wedge i \leq l \wedge (at(s, sO + i) > at(d, dO + i) \quad \wedge & \\
\forall j{:}Int(1 \leq j \wedge j < i \rightarrow at(s, sO + j) = at(d, dO + j))) & \\
) \rightarrow res = 1) &
\end{array}$$

Logisches Vokabular in rot

s	für	*src*	*d*	für	*dest*
sO	für	*srcOff*	*dO*	für	*destOff*
l	für	*length*			
E	für	*java :: lang :: Exception*			
NPE	für	*java :: lang :: NullPointerException*			

Formalisierung: Ausnahmeverhalten

$$
\begin{aligned}
&\neg excThrown(E) && \vee \\
&excThrown(NPE) \wedge (s = null \vee d = null) && \vee \\
&excThrown(OBE) \wedge \\
&(sO < 0 \vee dO < 0 \vee l < 0 \vee sO + l > size(s) \vee dO + l > size(d))
\end{aligned}
$$

Prädikatenlogik erster Stufe

Definition: Logische Zeichen

Wie in der Aussagenlogik:

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

Neu:

$\forall$	Allquantor
$\exists$	Existenzquantor
v_i	Individuenvariablen, $i \in N$
$\doteq$	objektsprachliches Gleichheitssymbol
,	Komma

Prädikatenlogik erster Stufe

Definition: Signatur

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

- F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \to N$.

$f \in F_\Sigma$ heißt Funktionssymbol,
$p \in P_\Sigma$ heißt Prädikatssymbol.
f ist n-stelliges Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;
p ist n-stelliges Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein nullstelliges Funktionssymbol heißt auch Konstantensymbol oder kurz Konstante,
ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Prädikatenlogik erster Stufe

Definition: Signatur

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

- F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \to N$.

$f \in F_\Sigma$ heißt Funktionssymbol,
$p \in P_\Sigma$ heißt Prädikatssymbol.
f ist n-stelliges Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;
p ist n-stelliges Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein nullstelliges Funktionssymbol heißt auch Konstantensymbol oder kurz Konstante,
ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Prädikatenlogik erster Stufe

Definition: Signatur

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

- F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \rightarrow N$.

$f \in F_\Sigma$ heißt Funktionssymbol,
$p \in P_\Sigma$ heißt Prädikatssymbol.
f ist n-stelliges Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;
p ist n-stelliges Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein nullstelliges Funktionssymbol heißt auch Konstantensymbol oder kurz Konstante,
ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Prädikatenlogik erster Stufe

Definition: Signatur

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

- F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \to N$.

$f \in F_\Sigma$ heißt Funktionssymbol,
$p \in P_\Sigma$ heißt Prädikatssymbol.
f ist n-stelliges Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;
p ist n-stelliges Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein nullstelliges Funktionssymbol heißt auch Konstantensymbol oder kurz Konstante,
ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Prädikatenlogik erster Stufe

Definition: Signatur

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

- F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \rightarrow N$.

$f \in F_\Sigma$ heißt Funktionssymbol,
$p \in P_\Sigma$ heißt Prädikatssymbol.
f ist n-stelliges Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;
p ist n-stelliges Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein nullstelliges Funktionssymbol heißt auch Konstantensymbol oder kurz Konstante,
ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Prädikatenlogik erster Stufe

Definition: Signatur

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

- F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \to N$.

$f \in F_\Sigma$ heißt Funktionssymbol,
$p \in P_\Sigma$ heißt Prädikatssymbol.
f ist n-stelliges Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;
p ist n-stelliges Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein nullstelliges Funktionssymbol heißt auch Konstantensymbol oder kurz Konstante,
ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Terme

KIT Karlsruhe Institute of Technology

Definition: 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 $f(t_1, \ldots, t_n) \in Term_\Sigma$

Ein Term heißt *Grundterm*, wenn er keine Variablen enthält.

Terme

Definition: 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 $f(t_1, \ldots, t_n) \in Term_\Sigma$

Ein Term heißt *Grundterm*, wenn er keine Variablen enthält.

Formeln

Definition: Atomare Formeln

At_Σ, die Menge der *atomaren Formeln über* Σ:

$$
\begin{aligned}
At_\Sigma \quad := \quad & \{s \doteq t \mid s, t \in Term_\Sigma\} \cup \\
& \{p(t_1, \ldots, t_n) \mid p \in P_\Sigma, t_i \in Term_\Sigma\}
\end{aligned}
$$

Formeln

Definition: 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:

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

Formeln

Definition: 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:

$$\neg A, (A \land B), (A \lor B), (A \rightarrow B), (A \leftrightarrow B), \forall x A, \exists x A$$

Gebundene und freie Variable

Definition

- Hat eine Formel A die Gestalt $\forall x B$ oder $\exists x B$, so heißt B der Wirkungsbereich des *Präfixes* $\forall x$ bzw. $\exists x$ von A.
- Ein Auftreten einer Variablen x in einer Formel A heißt gebunden, wenn es innerhalb des Wirkungsbereichs eines Präfixes $\forall x$ oder $\exists x$ einer Teilformel von A stattfindet.
- Ein Auftreten einer Variablen x in einer Formel A heißt frei, wenn es nicht gebunden ist und nicht unmittelbar rechts neben einem Quantor stattfindet.

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

gebundene Vorkommen
freie Vorkommen

Gebundene und freie Variable

Definition

- Hat eine Formel A die Gestalt $\forall xB$ oder $\exists xB$, so heißt B der Wirkungsbereich des *Präfixes* $\forall x$ bzw. $\exists x$ von A.
- Ein Auftreten einer Variablen x in einer Formel A heißt gebunden, wenn es innerhalb des Wirkungsbereichs eines Präfixes $\forall x$ oder $\exists x$ einer Teilformel von A stattfindet.
- Ein Auftreten einer Variablen x in einer Formel A heißt frei, wenn es nicht gebunden ist und nicht unmittelbar rechts neben einem Quantor stattfindet.

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

gebundene Vorkommen
freie Vorkommen

Notation

Definition

Es sei $A \in For_\Sigma$ und $t \in Term_\Sigma$.
$Bd(A) := \{x \mid x \in Var, x \text{ tritt gebunden in } A \text{ auf}\}$
$Frei(A) := \{x \mid x \in Var, x \text{ tritt frei in } A \text{ auf}\}$.
$Var(A) := Frei(A) \cup Bd(A)$
$Var(t) := \{x \mid x \in Var, x \text{ kommt in } t \text{ vor}\}$

Abschlussoperationen für Formeln

Definition

A heißt *geschlossen*, wenn $Frei(A) = \{\}$.
Ist $Frei(A) = \{x_1, \ldots, x_n\}$, 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 $Cl_\forall A = Cl_\exists A = A$.

Substitutionen

Karlsruhe Institute of Technology

Definition: Substitutionen

Eine *Substitution* ist eine Abbildung

$$\sigma : \mathit{Var} \rightarrow \mathit{Term}_\Sigma$$

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

Notation für Substitutionen

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 mit $\sigma(x) \neq x$ der Funktionswert $\sigma(x)$ ein Grundterm ist.
Mit id bezeichnen wir die identische Substitution auf Var, d.h. $id(x) = x$ für alle $x \in Var$.

Notation für Substitutionen

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 mit $\sigma(x) \neq x$ der Funktionswert $\sigma(x)$ ein Grundterm ist.
Mit *id* bezeichnen wir die identische Substitution auf *Var*, d.h. $id(x) = x$ für alle $x \in Var$.

Notation für Substitutionen

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 mit $\sigma(x) \neq x$ der Funktionswert $\sigma(x)$ ein Grundterm ist.

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

Notation für Substitutionen

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 mit $\sigma(x) \neq x$ der Funktionswert $\sigma(x)$ ein Grundterm ist.
Mit *id* bezeichnen wir die identische Substitution auf *Var*, d.h. $id(x) = x$ für alle $x \in Var$.

Anwendung von Substitutionen

Definition durch Beispiele

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
$$\mu(\exists y p(x,y)) = \exists y p(c,y).$$
3. Für $\sigma_1 = \{x/f(x,x)\}$ gilt
$$\sigma_1(\forall y p(x,y)) = \forall y p(f(x,x),y).$$
4. Für $\mu_1 = \{x/y\}$ gilt
$$\mu_1(\forall y p(x,y)) = \forall y p(y,y).$$

Anwendung von Substitutionen

Definition durch Beispiele

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

$$\mu(\exists y p(x,y)) = \exists y p(c,y).$$

3. Für $\sigma_1 = \{x/f(x,x)\}$ gilt

$$\sigma_1(\forall y p(x,y)) = \forall y p(f(x,x), y).$$

4. Für $\mu_1 = \{x/y\}$ gilt

$$\mu_1(\forall y p(x,y)) = \forall y p(y,y).$$

Anwendung von Substitutionen

Definition durch Beispiele

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

$$\mu(\exists y p(x,y)) = \exists y p(c,y).$$

3. Für $\sigma_1 = \{x/f(x,x)\}$ gilt

$$\sigma_1(\forall y p(x,y)) = \forall y p(f(x,x), y).$$

4. Für $\mu_1 = \{x/y\}$ gilt

$$\mu_1(\forall y p(x,y)) = \forall y p(y,y).$$

Anwendung von Substitutionen

Definition durch Beispiele

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
$$\mu(\exists y p(x,y)) = \exists y p(c,y).$$
3. Für $\sigma_1 = \{x/f(x,x)\}$ gilt
$$\sigma_1(\forall y p(x,y)) = \forall y p(f(x,x), y).$$
4. Für $\mu_1 = \{x/y\}$ gilt
$$\mu_1(\forall y p(x,y)) = \forall y p(y,y).$$

Kollisionsfreie Substitutionen

Definition: 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 Präfixes $\forall x$ oder $\exists x$, wo x eine Variable in $\sigma(z)$ ist.

$\mu_1 = \{x/y\}$ ist nicht kollisionsfrei für $\forall y p(x, y)$

Komposition von Substitutionen

Definition: 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 Anwendung der Substitution τ auf den Term $\sigma(x)$ verstanden werden muß.

Elementare Eigenschaften

Theorem

1. *Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, die Gleichung $\sigma(t) = \tau(t)$,
   dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t.*

Beweis

1. Strukturelle Induktion nach t.
   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \ldots, t_n)$. Dann gilt

$$\begin{aligned}\sigma(f(t_1, \ldots, t_n)) &= f(\sigma(t_1), \ldots, \sigma(t_n))\\ \tau(f(t_1, \ldots, t_n)) &= f(\tau(t_1), \ldots, \tau(t_n))\end{aligned}$$

Elementare Eigenschaften

Theorem

1. *Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, die Gleichung $\sigma(t) = \tau(t)$, dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t.*

Beweis

1. Strukturelle Induktion nach t.
   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \ldots, t_n)$. Dann gilt

$$\begin{array}{rcl} \sigma(f(t_1, \ldots, t_n)) & = & f(\sigma(t_1), \ldots, \sigma(t_n)) \\ \tau(f(t_1, \ldots, t_n)) & = & f(\tau(t_1), \ldots, \tau(t_n)). \end{array}$$

Elementare Eigenschaften

Theorem

1. *Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, die Gleichung* $\sigma(t) = \tau(t)$,
   dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t.

Beweis

1. Strukturelle Induktion nach t.
   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \ldots, t_n)$. Dann gilt

$$\begin{aligned} \sigma(f(t_1, \ldots, t_n)) &= f(\sigma(t_1), \ldots, \sigma(t_n)) \\ \tau(f(t_1, \ldots, t_n)) &= f(\tau(t_1), \ldots, \tau(t_n)). \end{aligned}$$

Elementare Eigenschaften

Theorem

1. *Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, die Gleichung*
   $\sigma(t) = \tau(t)$,
   dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t.

Beweis

1. Strukturelle Induktion nach t.
   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \ldots, t_n)$. Dann gilt

$$\begin{array}{rcl} \sigma(f(t_1, \ldots, t_n)) & = & f(\sigma(t_1), \ldots, \sigma(t_n)) \\ \tau(f(t_1, \ldots, t_n)) & = & f(\tau(t_1), \ldots, \tau(t_n)). \end{array}$$

Elementare Eigenschaften

Theorem

1. *Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, die Gleichung* $\sigma(t) = \tau(t)$, *dann* $\sigma(s) = \tau(s)$ *für jeden Teilterm s von t.*

Beweis

1. Strukturelle Induktion nach t.
   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \ldots, t_n)$. Dann gilt auch

     $$f(\sigma(t_1), \ldots, \sigma(t_n)) = f(\tau(t_1), \ldots, \tau(t_n)).$$

     und es folgt $\sigma(t_i) = \tau(t_i)$ für $i = 1, \ldots, n$. Da jeder Teilterm s von t entweder mit t identisch oder Teilterm eines t_i ist, folgt 1. nach Induktionsvoraussetzung.

Elementare Eigenschaften

Theorem

1. *Gilt für $t \in Term_\Sigma$ und Substitutionen σ, τ, die Gleichung $\sigma(t) = \tau(t)$,*
   dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t.

Beweis

1. Strukturelle Induktion nach t.
   - Ist $t \in Var$, dann ist t selbst sein einziger Teilterm.
   - Sei $t = f(t_1, \ldots, t_n)$. Dann gilt auch

   $$f(\sigma(t_1), \ldots, \sigma(t_n)) = f(\tau(t_1), \ldots, \tau(t_n)).$$

   und es folgt $\sigma(t_i) = \tau(t_i)$ für $i = 1, \ldots, n$. Da jeder Teilterm s von t entweder mit t identisch oder Teilterm eines t_i ist, folgt 1. nach Induktionsvoraussetzung.

Elementare Eigenschaften

Theorem

1. *Gilt für* $t \in Term_\Sigma$ *und Substitutionen* σ, τ, *die Gleichung* $\sigma(t) = \tau(t)$,
   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.
2. Spezialfall von 1.

Variablenumbenennung

Theorem

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 $x = \tau(\sigma(x)) = \tau(\sigma(y)) = y$.

Variablenumbenennung

Theorem

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 $x = \tau(\sigma(x)) = \tau(\sigma(y)) = y$.

Variablenumbenennung

Theorem

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 $x = \tau(\sigma(x)) = \tau(\sigma(y)) = y$.

Unifikation

Definition

Es sei $T \subseteq Term_\Sigma$, $T \neq \{\}$, und σ eine Substitution über Σ.

σ unifiziert T,
oder: σ 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

$$\sigma(t) = \sigma(s).$$

Unifikation

Definition

Es sei $T \subseteq Term_\Sigma$, $T \neq \{\}$, und σ eine Substitution über Σ.

σ unifiziert T,
oder: σ 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

$$\sigma(t) = \sigma(s).$$

Unifikation

Definition

Es sei $T \subseteq Term_\Sigma$, $T \neq \{\}$, und σ eine Substitution über Σ.

σ unifiziert T,
oder: σ 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

$$\sigma(t) = \sigma(s).$$

Beispiel

$$\{f(g(a,x),g(y,b)),\quad f(z,g(v,w)),\quad f(g(x,a),g(v,b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a,a), w/b\}.$$

Beispiel

$$\{f(g(a, x), g(y, b)),\quad f(z, g(v, w)),\quad f(g(x, a), g(v, b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a, a), w/b\}.$$

Beispiel

$$\{f(g(a,x),g(y,b)),\quad f(z,g(v,w)),\quad f(g(x,a),g(v,b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a,a), w/b\}.$$

$$\{f(g(a,a),g(v,b)), f(g(a,a),g(v,b)), f(g(a,a),g(v,b))\}$$

Beispiel

$$\{f(g(a,x),g(y,b)),\quad f(z,g(v,w)),\quad f(g(x,a),g(v,b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a,a), w/b\}.$$

$$\left\{\begin{array}{l} f(g(a,a),g(v,b)), \\ f(g(a,a),g(v,b)), \\ f(g(a,a),g(v,b)) \end{array}\right\}$$

Elementare Fakten

1. Jeder Term ist mit sich selbst unifizierbar mittels *id*.
2. Zwei Terme der Gestalt
$$f(s_1, \ldots, s_n), g(t_1, \ldots, t_n)$$
(mit verschiedenem Kopf) sind nicht unifizierbar.
3. 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 $i = 1, \ldots, n$.
4. Sei $x \in Var$ und t ein Term. Dann sind
$$x \text{ und } t$$
genau dann unifizierbar, wenn x nicht in t vorkommt.

Elementare Fakten

1. Jeder Term ist mit sich selbst unifizierbar mittels *id*.
2. Zwei Terme der Gestalt

$$f(s_1, \ldots, s_n), g(t_1, \ldots, t_n)$$

   (mit verschiedenem Kopf) sind nicht unifizierbar.
3. 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 $i = 1, \ldots, n$.
4. Sei $x \in Var$ und t ein Term. Dann sind

$$x \text{ und } t$$

   genau dann unifizierbar, wenn x nicht in t vorkommt.

Elementare Fakten

1. Jeder Term ist mit sich selbst unifizierbar mittels *id*.
2. Zwei Terme der Gestalt

   $$f(s_1, \dots, s_n), g(t_1, \dots, t_n)$$

   (mit verschiedenem Kopf) sind nicht unifizierbar.
3. Zwei Terme der Gestalt

   $$f(s_1, \dots, s_n), f(t_1, \dots, t_n)$$

   (mit demselben Kopf) sind genau dann unifizierbar, wenn es eine Substitution σ gibt mit $\sigma(s_i) = \sigma(t_i)$ für $i = 1, \dots, n$.
4. Sei $x \in Var$ und t ein Term. Dann sind

   $$x \text{ und } t$$

   genau dann unifizierbar, wenn x nicht in t vorkommt.

Elementare Fakten

1. Jeder Term ist mit sich selbst unifizierbar mittels *id*.
2. Zwei Terme der Gestalt

$$f(s_1, \ldots, s_n), g(t_1, \ldots, t_n)$$

   (mit verschiedenem Kopf) sind nicht unifizierbar.
3. 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 $i = 1, \ldots, n$.
4. Sei $x \in Var$ und t ein Term. Dann sind

$$x \text{ und } t$$

   genau dann unifizierbar, wenn x nicht in t vorkommt.

Beispiel

$$\{f(x, g(y)),\ \ f(g(a), g(z))\}$$

wird unifiziert durch

$$\sigma = \{x/g(a), z/y\} \quad \text{Ergebnis } f(g(a), g(y)),$$

aber auch durch

$$\tau = \{x/g(a), y/a, z/a\} \quad \text{Ergebnis } f(g(a), g(a)).$$

σ ist allgemeiner als τ – oder τ spezieller als σ –

$$\tau = \{y/a\} \circ \sigma.$$

Beispiel

$$\{f(x, g(y)),\ \ f(g(a), g(z))\}$$

wird unifiziert durch

$$\sigma = \{x/g(a), z/y\} \quad \text{Ergebnis } f(g(a), g(y)),$$

aber auch durch

$$\tau = \{x/g(a), y/a, z/a\} \quad \text{Ergebnis } f(g(a), g(a)).$$

σ ist allgemeiner als τ – oder τ spezieller als σ –

$$\tau = \{y/a\} \circ \sigma.$$

Allgemeinster Unifikator

Definition

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 $\sigma = \sigma' \circ \mu$.

Allgemeinster Unifikator

Definition

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 $\sigma = \sigma' \circ \mu$.

Eindeutigkeit des allgemeinsten Unifikators

Theorem

Es sei T eine unifizierbare, nichtleere Menge von Termen. Dann ist der allgemeinste Unifikator von T bis auf Variablenumbenennung eindeutig bestimmt,

d. h.:

Sind μ, μ' allgemeinste Unifikatoren von T mit

$$\mu(T) = \{t\} \text{ und } \mu'(T) = \{t'\},$$

dann gibt es eine Umbenennung π der Variablen von t mit

$$t' = \pi(t).$$

Eindeutigkeit des allgemeinsten Unifikators

Theorem

Es sei T eine unifizierbare, nichtleere Menge von Termen. Dann ist der allgemeinste Unifikator von T bis auf Variablenumbenennung eindeutig bestimmt,

d. h.:
Sind μ, μ' allgemeinste Unifikatoren von T mit

$$\mu(T) = \{t\} \text{ und } \mu'(T) = \{t'\},$$

dann gibt es eine Umbenennung π der Variablen von t mit

$$t' = \pi(t).$$

Eindeutigkeit des allgemeinsten Unifikators

Theorem

Es sei T eine unifizierbare, nichtleere Menge von Termen. Dann ist der allgemeinste Unifikator von T bis auf Variablenumbenennung eindeutig bestimmt,
d. h.:
Sind μ, μ' allgemeinste Unifikatoren von T mit

$$\mu(T) = \{t\} \text{ und } \mu'(T) = \{t'\},$$

dann gibt es eine Umbenennung π der Variablen von t mit

$$t' = \pi(t).$$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$

d.h. $t = \sigma'\sigma(t)$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$

d.h. $t = \sigma'\sigma(t)$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$

$$\text{d.h. } t = \sigma'\sigma(t)$$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$

d.h. $t = \sigma'\sigma(t)$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$

d.h. $t = \sigma'\sigma(t)$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$
$$\text{d.h. } t = \sigma'\sigma(t)$$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$
$$\text{d.h. } t = \sigma'\sigma(t)$$

Das kann nur sein, wenn für jede Variable $x \in Var(t)$ gilt

$$\sigma'\sigma(x) = x.$$

Beweis

Nach der Definition gibt es Substitutionen σ, σ' mit

- $\mu' = \sigma\mu$
- $\mu = \sigma'\mu'$

Daraus folgt

$$\mu(T) = t = \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t)$$
$$\text{d.h. } t = \sigma'\sigma(t)$$

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.

Unifikationsalgorithmus von Robinson

Definition

Zu $t \in Term_\Sigma$ und $i \in N$ 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.

Unifikationsalgorithmus von Robinson

Definition

Für $T \subseteq Term_\Sigma$ ist die *Differenz* von T, $D(T) \subseteq Term_\Sigma$, wie folgt definiert

1. $D(T) := T$ falls $\#T \leq 1$
2. Falls $\#T \geq 2$, sei j die kleinste Zahl, so daß sich zwei Terme aus T an der Position j unterscheiden.
   Setze $D(T) := \{t^{(j)} \mid t \in T\}$.

Beispiel

$$T = \{f(g(a,x), g(y,b)),\ \ f(z, g(v,w)),\ \ f(g(x,a), g(v,b))\}$$

$$D(T) = \{g(a,x), z, g(x,a)\}$$

Unifikationsalgorithmus von Robinson

Definition

Für $T \subseteq Term_\Sigma$ ist die *Differenz* von T, $D(T) \subseteq Term_\Sigma$, wie folgt definiert

1. $D(T) := T$ falls $\#T \leq 1$
2. Falls $\#T \geq 2$, sei j die kleinste Zahl, so daß sich zwei Terme aus T an der Position j unterscheiden.
   Setze $D(T) := \{t^{(j)} \mid t \in T\}$.

Beispiel

$T = \{f(g(a,x), g(y,b)),\ \ f(z, g(v,w)),\ \ f(g(x,a), g(v,b))\}$

$D(T) = \{g(a,x), z, g(x,a)\}$

Unifikationsalgorithmus von Robinson

Definition

Für $T \subseteq Term_\Sigma$ ist die *Differenz* von T, $D(T) \subseteq Term_\Sigma$, wie folgt definiert

1. $D(T) := T$ falls $\#T \leq 1$
2. Falls $\#T \geq 2$, sei j die kleinste Zahl, so daß sich zwei Terme aus T an der Position j unterscheiden.
   Setze $D(T) := \{t^{(j)} \mid t \in T\}$.

Beispiel

$T = \{f(g(a,x), g(y,b)),\ \ f(z, g(v,w)),\ \ f(g(x,a), g(v,b))\}$

$D(T) = \{g(a,x), z, g(x,a)\}$

Unifikationsalgorithmus von Robinson

Definition

Für $T \subseteq Term_\Sigma$ ist die *Differenz* von T, $D(T) \subseteq Term_\Sigma$, wie folgt definiert

1. $D(T) := T$ falls $\#T \leq 1$
2. Falls $\#T \geq 2$, sei j die kleinste Zahl, so daß sich zwei Terme aus T an der Position j unterscheiden.
   Setze $D(T) := \{t^{(j)} \mid t \in T\}$.

Beispiel

$T = \{f(g(a,x), g(y,b)),\ \ f(z, g(v,w)),\ \ f(g(x,a), g(v,b))\}$

$D(T) = \{g(a,x), z, g(x,a)\}$

Unifikationsalgorithmus von Robinson

Definition

Für $T \subseteq Term_\Sigma$ ist die *Differenz* von T, $D(T) \subseteq Term_\Sigma$, wie folgt definiert

1. $D(T) := T$ falls $\#T \leq 1$
2. Falls $\#T \geq 2$, sei j die kleinste Zahl, so daß sich zwei Terme aus T an der Position j unterscheiden.
   Setze $D(T) := \{t^{(j)} \mid t \in T\}$.

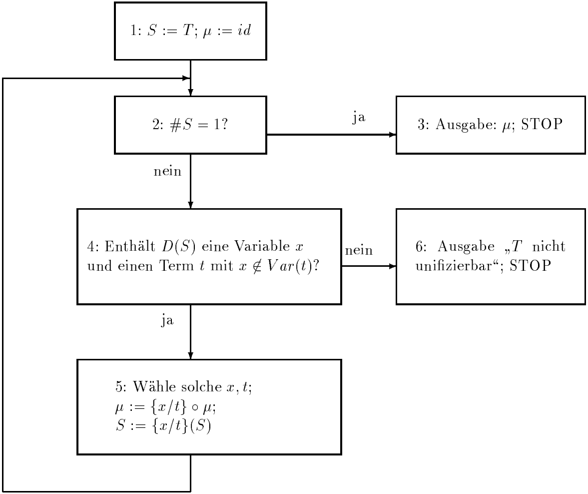
Beispiel

$$T = \{f(g(a,x),g(y,b)), \quad f(z,g(v,w)), \quad f(g(x,a),g(v,b))\}$$

$$D(T) = \{g(a,x), z, g(x,a)\}$$

Algorithmus von Robinson

Gegeben sei $T \subseteq Term_\Sigma$, T endlich und $\neq \emptyset$.



Unifikationstheorem

Theorem

1. *Der Algorithmus von* Robinson *terminiert für jedes endliche, nichtleere $T \subseteq Term_\Sigma$.*
2. *Wenn T unifizierbar ist, liefert er einen allgemeinsten Unifikator von T.*
3. *Wenn T nicht unifizierbar ist, liefert er die Ausgabe „T nicht unifizierbar“.*

Unifikationstheorem

Theorem

1. *Der Algorithmus von* ROBINSON *terminiert für jedes endliche, nichtleere* $T \subseteq Term_\Sigma$.
2. *Wenn* T *unifizierbar ist, liefert er einen allgemeinsten Unifikator von* T.
3. *Wenn* T *nicht unifizierbar ist, liefert er die Ausgabe „*T *nicht unifizierbar".*

Unifikationstheorem

Theorem

1. *Der Algorithmus von* Robinson *terminiert für jedes endliche, nichtleere* $T \subseteq Term_\Sigma$.
2. *Wenn* T *unifizierbar ist, liefert er einen allgemeinsten Unifikator von* T.
3. *Wenn* T *nicht unifizierbar ist, liefert er die Ausgabe „*T *nicht unifizierbar“.*

Unifikationstheorem

Theorem

1. *Der Algorithmus von* ROBINSON *terminiert für jedes endliche, nichtleere $T \subseteq Term_\Sigma$.*
2. *Wenn T unifizierbar ist, liefert er einen allgemeinsten Unifikator von T.*
3. *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 gibt es 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.

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 gibt es 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.

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 gibt es 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.

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 gibt es 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.

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 gibt es 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.

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 gibt es 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.

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 gibt es 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.

Beweis

Unter einem *Schleifendurchlauf* in dem obigem Algorithmus verstehen wir einen *vollständigen* Durchlauf der Befehlsfolge 2–4–5.

Wir setzen

- $S_0 := T$, $\mu_0 := id$
- $S_{k+1} :=$ Wert von S nach dem $(k+1)$-ten Schleifendurchlauf
- $\mu_{k+1} := \mu$ nach dem $(k+1)$-ten Schleifendurchlauf
- x_k, t_k die im $(k+1)$-ten Durchlauf gewählten x, t.

Beweis

Unter einem *Schleifendurchlauf* in dem obigem Algorithmus verstehen wir einen *vollständigen* Durchlauf der Befehlsfolge 2–4–5.
Wir setzen

- $S_0 := T$, $\mu_0 := id$
- $S_{k+1} :=$ Wert von S nach dem $(k+1)$-ten Schleifendurchlauf
- $\mu_{k+1} := \mu$ nach dem $(k+1)$-ten Schleifendurchlauf
- x_k, t_k die im $(k+1)$-ten Durchlauf gewählten x, t.

Beweis

Im $(k+1)$-ten Schleifendurchlauf gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei kommt x_k nicht in t_k vor.
Nach Anwendung der Substitution $\{x_k/t_k\}$ gibt es in S_{k+1} kein Vorkommen der Variable x_k mehr.
Da t_k selbst ein Term in S_k war, werden durch die Subsitution auch keine neuen Variablen eingeführt.
Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k. Die Schleife terminiert nach endlichen vielen Durchläufen.

Beweis

Im $(k+1)$-ten Schleifendurchlauf gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei kommt x_k nicht in t_k vor.
Nach Anwendung der Substitution $\{x_k/t_k\}$ gibt es in S_{k+1} kein Vorkommen der Variable x_k mehr.
Da t_k selbst ein Term in S_k war, werden durch die Subsitution auch keine neuen Variablen eingeführt.
Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k. Die Schleife terminiert nach endlichen vielen Durchläufen.

Beweis

Im $(k+1)$-ten Schleifendurchlauf gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei kommt x_k nicht in t_k vor.
Nach Anwendung der Substitution $\{x_k/t_k\}$ gibt es in S_{k+1} kein Vorkommen der Variable x_k mehr.
Da t_k selbst ein Term in S_k war, werden durch die Subsitution auch keine neuen Variablen eingeführt.
Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k. Die Schleife terminiert nach endlichen vielen Durchläufen.

Beweis

Im $(k+1)$-ten Schleifendurchlauf gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei kommt x_k nicht in t_k vor.
Nach Anwendung der Substitution $\{x_k/t_k\}$ gibt es in S_{k+1} kein Vorkommen der Variable x_k mehr.
Da t_k selbst ein Term in S_k war, werden durch die Subsitution auch keine neuen Variablen eingeführt.
Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k. Die Schleife terminiert nach endlichen vielen Durchläufen.

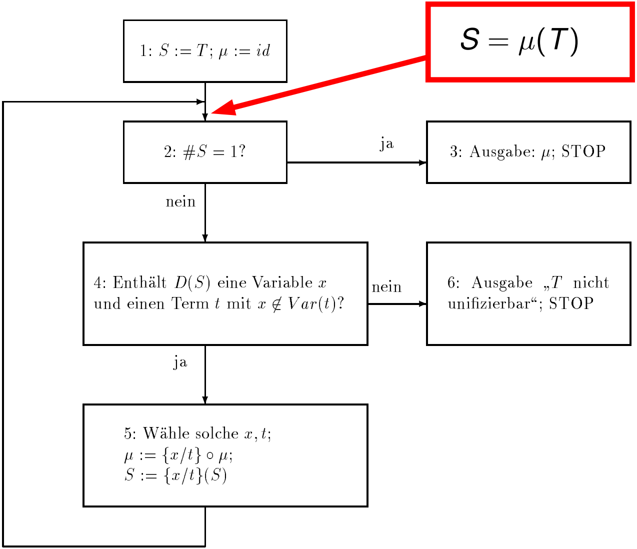
Beweis

Im $(k + 1)$-ten Schleifendurchlauf gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei kommt x_k nicht in t_k vor.
Nach Anwendung der Substitution $\{x_k/t_k\}$ gibt es in S_{k+1} kein Vorkommen der Variable x_k mehr.
Da t_k selbst ein Term in S_k war, werden durch die Subsitution auch keine neuen Variablen eingeführt.
Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k. Die Schleife terminiert nach endlichen vielen Durchläufen.

Algorithmus von Robinson



Beweis

Die Invariante besagt für alle $k \leq m$:

$$S_k = \mu_k(T)$$

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.

Beweis

Die Invariante besagt für alle $k \leq m$:

$$S_k = \mu_k(T)$$

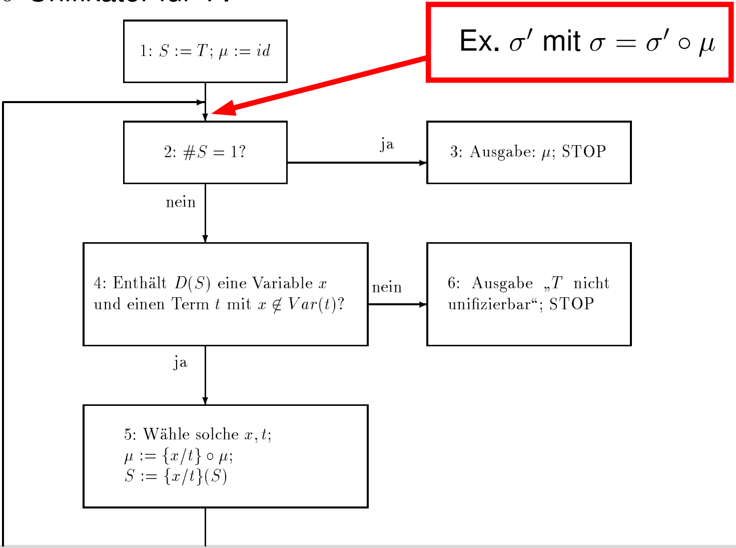
Hält das Programm mit Ausgabe einer Substitution μ an, dann hat μ den Wert μ_m, und es ist

$$\#\mu_m(T) = \#S_m = 1.$$

μ_m ist Unifikator von T.

Algorithmus von Robinson

σ Unifikator für T.



Beweis

Es sei σ ein Unifikator von T.

Behauptung: Für alle k gibt es 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.

Im $(k+1)$-ten Durchlauf wird 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 $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis

Es sei σ ein Unifikator von T.
Behauptung: Für alle k gibt es 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.
Im $(k+1)$-ten Durchlauf wird 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 $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis

Es sei σ ein Unifikator von T.
Behauptung: Für alle k gibt es 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.
Im $(k+1)$-ten Durchlauf wird 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 $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis

Es sei σ ein Unifikator von T.
Behauptung: Für alle k gibt es 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.
Im $(k+1)$-ten Durchlauf wird 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 $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis

Es sei σ ein Unifikator von T.
Behauptung: Für alle k gibt es 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.
Im $(k+1)$-ten Durchlauf wird 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 $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis

Es sei σ ein Unifikator von T.
Behauptung: Für alle k gibt es 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.
Im $(k + 1)$-ten Durchlauf wird 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 $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis (Forts.)

Wir setzen

$$\sigma_{k+1}(x) = \begin{cases} \sigma_k(x) & \text{falls } x \neq x_k \\ x_k & \text{falls } x = x_k. \end{cases}$$

Falls $x \neq x_k$ $\quad \sigma_{k+1}(\{x_k/t_k\}(x)) =$

$\sigma_{k+1}(x) =$

$\sigma_k(x)$,

Falls $x = x_k$ $\quad \sigma_{k+1}(\{x_k/t_k\}(x)) =$

$\sigma_{k+1}(\{x_k/t_k\}(x_k)) =$

$\sigma_{k+1}(t_k) =$

$\sigma_k(t_k) =$ $\qquad$ da $x_k \notin Var(t_k)$

$\sigma_k(x_k) = \sigma_k(x)$.

Somit $\quad \sigma_{k+1} \circ \{x_k/t_k\} = \sigma_k$.

Es folgt: $\sigma_{k+1} \circ \mu_{k+1} = \sigma_{k+1} \circ \{x_k/t_k\} \circ \mu_k$

$= \sigma_k \circ \mu_k = \sigma$. d. h. $(*)$.

Insbesondere gilt: $\quad \sigma = \sigma_m \circ \mu_m$.

Beweis (Forts.)

Wir setzen

$$\sigma_{k+1}(x) = \begin{cases} \sigma_k(x) & \text{falls } x \neq x_k \\ x_k & \text{falls } x = x_k. \end{cases}$$

$$\begin{array}{lll}
\text{Falls } x \neq x_k & \sigma_{k+1}(\{x_k/t_k\}(x)) = & \\
 & \sigma_{k+1}(x) = & \\
 & \sigma_k(x), & \\
\text{Falls } x = x_k & \sigma_{k+1}(\{x_k/t_k\}(x)) = & \\
 & \sigma_{k+1}(\{x_k/t_k\}(x_k)) = & \\
 & \sigma_{k+1}(t_k) = & \\
 & \sigma_k(t_k) = & \text{da } x_k \notin Var(t_k) \\
 & \sigma_k(x_k) = \sigma_k(x). &
\end{array}$$

Somit $\sigma_{k+1} \circ \{x_k/t_k\} = \sigma_k$.

Es folgt: $\sigma_{k+1} \circ \mu_{k+1} = \sigma_{k+1} \circ \{x_k/t_k\} \circ \mu_k$

$= \sigma_k \circ \mu_k = \sigma$. d. h. $(*)$.

Insbesondere gilt: $\sigma = \sigma_m \circ \mu_m$.

Beweis

σ_m unifiziert S_m (da σ T unifiziert).

Also muß $D(S_m)$ eine Variable x und einen Term t enthalten mit $x \notin Var(t)$

Die Antwort auf Test 2 muß also „Ja“ sein.

Beweis

σ_m unifiziert S_m (da σ T unifiziert).
Also muß $D(S_m)$ eine Variable x und einen Term t enthalten mit $x \notin Var(t)$
Die Antwort auf Test 2 muß also „Ja“ sein.

Beweis

σ_m unifiziert S_m (da σ T unifiziert).
Also muß $D(S_m)$ eine Variable x und einen Term t enthalten mit $x \notin Var(t)$
Die Antwort auf Test 2 muß also „Ja“ sein.