Karlsruher Institut für Technologie

Name: _____

Vorname: _____

Matrikel-Nr.: _____

Klausur Formale Systeme

Fakultät für Informatik

SS 2023

Prof. Dr. Bernhard Beckert

4. September 2023

Die Bearbeitungszeit beträgt 60 Minuten.

A1 (9)	A2 (12)	A3 (8)	A4 (7)	A5 (8)	A6 (9)	A7 (7)	Σ (60)

Bewertungstabelle bitte frei lassen!

Gesamtpunkte:

1 Zur Einstimmung ((2 + 2) + (1 + 1 + 1) + 2 = 9 Punkte)

a. Seien p, q einstellige Prädikatensymbole, r ein zweistelliges Prädikatensymbol und f ein einstelliges Funktionssymbol.

Geben Sie für folgende prädikatenlogische Formeln – **falls möglich** – jeweils zwei Interpretationen I über dem Universum $D = \{a, b\}$ an, und zwar jeweils

- eine Interpretation, in der die Formel **wahr** ist, und
- eine Interpretation, in der die Formel **falsch** ist.

In den Fällen, in denen eine Interpretation mit der gesuchten Eigenschaft **nicht existiert**, geben Sie dies an.

Hinweis: Es muss explizit angegeben werden, wenn eine passende Interpretation nicht existiert (schreiben Sie „existiert nicht“ neben den Kasten). Es genügt nicht, die Beschreibung der Interpretation leer zu lassen.

i. $\forall x \left(((\exists y\, p(y)) \land q(x)) \to \neg\forall z\, (p(z) \lor q(z))\right)$

Interpretation, in der die Formel wahr ist:

$I(p)(a) =$	$I(p)(b) =$	$I(q)(a) =$	$I(q)(b) =$

Interpretation, in der die Formel falsch ist:

$I(p)(a) =$	$I(p)(b) =$	$I(q)(a) =$	$I(q)(b) =$

ii. $(\exists x \forall y\, r(x, f(y))) \to (\exists x \forall y\, r(f(x), y))$

Interpretation, in der die Formel wahr ist:

$I(r)(a, a) =$	$I(r)(a, b) =$	$I(r)(b, a) =$	$I(r)(b, b) =$

$I(f)(a) =$	$I(f)(b) =$

Interpretation, in der die Formel falsch ist:

$I(r)(a, a) =$	$I(r)(a, b) =$	$I(r)(b, a) =$	$I(r)(b, b) =$

$I(f)(a) =$	$I(f)(b) =$

Fortsetzung 1 Zur Einstimmung

b. Geben Sie kurze Antworten zu folgenden Fragen bzw. Aufgaben:

i. Nennen Sie eine Konsequenz des Gödelschen Unvollständigkeitssatzes.

ii. Erläutern Sie den Unterschied zwischen $\models$ und $\vdash$.

iii. In der Vorlesung haben Sie Konfluenz als Eigenschaft von Termersetzungssystemen $(D, \to^*)$ kennengelernt. Konfluenz lässt sich aber auch auf Kalküle anwenden, indem D die Menge der Beweissituationen (z.B. Tableaus) und der Übergang $\to^*$ die Anwendung einer Kalkülregel ist. Begründen Sie warum die Konfluenz für die Beweissuche eine wichtige Eigenschaften darstellt.

c. Zeigen Sie die Unerfüllbarkeit folgender aussagenlogischer Klauselmenge über $\Sigma = \{P, \ldots, T\}$ mit dem **Markierungsalgorithmus**:

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

Geben Sie dabei an, in welcher Reihenfolge die Atome markiert werden.

Markierungsreihenfolge:

2 Abstrakte Datentypen und Prädikatenlogik
(4 + 3 + 3 + 2 = 12 Punkte)

Man kann abstrakte Datentypen (ADTs) in Prädikatenlogik formalisieren. Dazu sei eine Signatur $\Sigma = (P_\Sigma, F_\Sigma, \alpha_\Sigma)$ gegeben, und wir gehen davon aus, dass die Menge $K \subseteq F_\Sigma$ der Konstruktoren des ADTs eine Teilmenge der Funktionssymbole F_Σ ist. Zudem sei ein prädikatenlogisches Modell $M = (D, I)$ über der Signatur Σ gegeben.

Beispielsweise kann man mit folgenden Konstruktoren den ADT *Binärbaum* formalisieren:

- *leaf*: eine Konstante (für Blätter)
- $tree(x, y)$: eine zweistellige Funktion (x und y sind der linke bzw. der rechte Teil-Baum)

Hinweis: Die Signatur kann neben Konstruktoren weitere Funktionen enthalten, wie bspw. *left*, *right*, die man auf Bäume anwenden kann. Das ist für diese Aufgabe aber nicht relevant.

a. Ein ADT heißt **generiert**, wenn

für alle $d \in D$ gilt: es gibt einen Grundterm t, der nur Konstruktoren enthält, mit $I(t) = d$

Man könnte meinen, dass Generiertheit in Prädikatenlogik formalisierbar ist. Das ist tatsächlich aber im allgemeinen nicht möglich. Versuchen könnte man es aber (vergeblich) für den Beispiel-ADT *Binärbaum* mit der Formel

$$\forall tr\,(tr \doteq leaf \vee \exists x \exists y (tr \doteq tree(x, y))) \ .$$

Begründen Sie kurz, warum diese Formel nicht formalisiert, dass der obige Beispiel-ADT generiert ist.

b. Ein ADT heißt **frei generiert**, wenn er generiert ist und zudem für alle Grundterme t_1, t_2, die nur Konstruktoren enthalten, gilt:

$$\text{wenn } t_1 \neq t_2 \text{ dann } I(t_1) \neq I(t_2)$$

Man kann in Prädikatenlogik formalisieren, dass ein gegebener ADT, von dem man weiß, dass er generiert ist, darüber hinaus auch *frei* generiert ist. Zeigen Sie für den Beispiel-ADT *Binärbaum*, dass das geht, indem Sie folgende prädikatenlogische Formel passend ergänzen:

$$\forall x \forall y \left(\neg(tree(x, y) \doteq leaf) \ \wedge \ \underline{\qquad\qquad\qquad\qquad\qquad\qquad} \right)$$

c. Ein wesentlicher Vorteil generierter ADTs ist, dass man eine korrekte und vollständige Induktionsregel angeben kann. Vervollständigen Sie in der Box die folgende Induktionsregel für den Beispiel-ADT *Binärbaum*.

Hinweis: $\phi(u)$ ist dabei eine beliebige prädikatenlogische Formel mit einer freien Variable u.

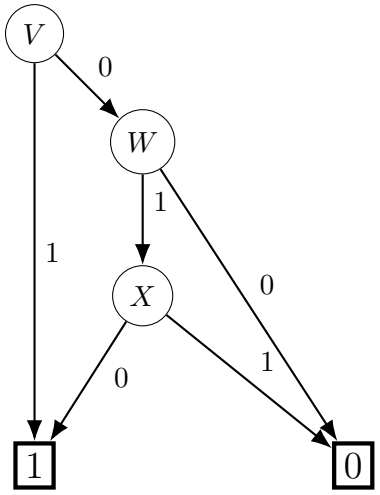
$$\frac{\phi(\mathit{leaf}) \qquad \boxed{}}{\forall u\, \phi(u)}$$

d. Wenn der ADT nicht generiert ist, ist die Regel aus Teilaufgabe c nicht korrekt. Begründen Sie kurz, warum das so ist.

3 Binary Decision Diagrams (3 + 3 + 2 = 8 Punkte)

Diese Aufgabe beschäftigt sich mit Binary Decision Diagrams (auch Shannongraphen genannt).

a. Unten sehen Sie den Shannongraphen G für die Formel $f_G \equiv V \vee (W \wedge \neg X)$. Zeichnen Sie einen Shannongraphen H für die Formel $f_H \equiv (\neg V \wedge W \wedge Y) \vee X$. Nutzen Sie dabei die folgende Ordnung der Variablen: $V < W < X < Y$.



b. Seien f_G und f_H die von Ihren Shannongraphen in a) beschriebenden Formeln. Geben Sie einen Shannongraphen für $f_G \to f_H$ an. Sie müssen Ihre Shannongraphen aus a) dabei wiederverwenden. Eine Reduktion des Graphen ist nicht notwendig; die Ordnung der Variablen aus a) muss **nicht** beachtet werden.

Fortsetzung 3 Binary Decision Diagrams

c. Geben Sie eine allgemeine Konstruktionsvorschrift an um aus zwei beliebigen gegebenen Shannongraphen G, H einen Shannongraphen für $f_G \to f_H$ zu konstruieren. Beschreiben sie explizit die Transformation des Graphen und *nicht* die Transformation als Abfolge logischer Operationen.

4 Formalisieren in Prädikatenlogik (PL1)

(1 + 1 + 2 + 3 = 7 Punkte)

Gegeben sei die prädikatenlogische Signatur $\Sigma = (\{mother, author\}, \{person, program, trusts\}, \alpha)$. Sie enthält die einstelligen Prädikatensymbole $person(\cdot)$ und $program(\cdot)$, das zweistellige Prädikatensymbol $trusts(\cdot, \cdot)$, sowie die einstelligen Funktionssymbole $mother(\cdot)$ und $author(\cdot)$.

Zur Auswertung der Formeln werden nur solche Interpretationen (D, I) über Σ verwendet, in denen

- das Universum D eine Menge von Personen und Computerprogrammen ist,
- das Prädikat $person(x)$ genau dann wahr ist, wenn x eine Person ist,
- das Prädikat $program(x)$ genau dann wahr ist, wenn x ein Computerprogramm ist,
- das Prädikat $trusts(x, y)$ genau dann wahr ist, wenn x dem Element y vertraut,
- die Funktion $mother(x)$ eine Person x auf die Mutter von x abbildet und
- die Funktion $author(x)$ ein Computerprogramm x auf das Element abbildet, das x geschrieben hat.

Geben Sie jeweils eine Formel der Prädikatenlogik mit Gleichheit über Σ an, die folgende Sachverhalte darstellt:

a. Personen sind keine Computerprogramme.

b. Es gibt eine Person, die keiner Person außer sich selbst vertraut.

c. Es gibt Personen, die einer Person genau dann vertrauen, wenn ihre Mutter dieser Person vertraut.

d. Computerprogramme vertrauen grundsätzlich nur Computerprogrammen und Personen, die ein Computerprogramm geschrieben haben.

5 Tableaukalkül (8 Punkte)

Vervollständigen und schließen Sie den folgenden Tableaubeweis.

Notieren Sie

- bei jeder Erweiterung den Regeltyp $(\alpha, \beta, \gamma, \delta)$ und die Formel, auf die die Regel angewendet wird.
- bei jedem Abschluss, welche Formeln verwendet wurden.
- die schließende Substitution.

$$1\ \forall x\,(p(x) \to q(x)) \quad (1)$$
$$|$$
$$1\ \forall y\,\forall z\,(p(y) \land s(y,z) \to p(z)) \quad (2)$$
$$|$$
$$0\ s(a,b) \to \neg p(a) \lor q(b) \quad (3)$$

6 Spezifikation mit der Java Modeling Language

(4 + (3 + 2) = 9 Punkte)

a. Geben Sie in natürlicher Sprache wieder, was der folgende JML-Methodenvertrag für die Methode `m` aussagt.
Hinweis 1: Achten Sie auf die Klammerung!
Hinweis 2: Der Ausdruck `(\product int j; a <= j && j < b; f(j))` für gegebene Integer a und b steht für das Produkt $\prod_{j=a}^{b-1} f(j)$.

```
/*@ normal_behavior
  @  requires b >= 2;
  @  ensures (\exists int r; 0 <= r && r < x;
  @                x == (\product int i; 0 <= i && i < r; b) && \result == r)
  @    || ( !(\exists int r; 0 <= r && r < x;
  @                x == (\product int i; 0 <= i && i < r; b)) && \result == -1);
  @  assignable \nothing;
  @*/
static int m(int x, int b) { ... }
```

b. Gegeben seien die folgenden Klassen, die einen gerichteten Graphen implementieren. Die Knoten (`Node`) sowie die Kanten (`Edge`) werden dabei jeweils in einem Array gespeichert, wobei jede Kante eine Referenz auf ihren Startknoten (`start`) und ihren Endknoten (`end`) enthält. Um z.B. eine Straßenkarte darzustellen, sei die Objektinvariante des Graphen, dass es keine unereichbaren "Inseln" gibt, dass also jeder Knoten von jedem anderen Knoten aus erreichbar ist.

i. Für die Formalisierung der Erreichbarkeit steht die Methode `reachable(Node a, Node b, int k)` zur Verfügung, die genau dann `true` zurückgeben soll, wenn der Knoten b von Knoten a aus in **genau** k Schritten erreichbar ist. Formulieren Sie diese Nachbedingung der Methode `reachable`.

ii. Formulieren Sie die oben gegebene Invariante.

Hinweis: Es ist erlaubt, in der `ensures`-Klausel die Methode `reachable` rekursiv zu benutzen, da sie mit dem Schlüsselwort `pure` markiert ist.

```
class Node { ... }

class Edge {
    Node start;
    Node end;
}

class DirectedGraph {
    Node[] nodes;
    Edge[] edges;


    /*@ invariant ____________________________________________________________

      @ ____________________________________________________________

      @ ____________________________________________________________

      @ ____________________________________________________________

      @*/

    /*@ normal_behavior
      @ requires k >= 0;
      @ requires (\exists int i; 0 <= i < nodes.length; nodes[i] == a);
      @ requires (\exists int i; 0 <= i < nodes.length; nodes[i] == b);

      @ ensures ____________________________________________________________

      @ ____________________________________________________________

      @ ____________________________________________________________

      @ ____________________________________________________________

      @ ____________________________________________________________

      @ ____________________________________________________________

      @*/
    static boolean /*@ pure @*/ reachable(Node a, Node b, int k) { ... };
}
```

7 Lineare Temporale Logik (LTL) und Büchiautomaten ((1,5 + 1,5 + 1,5) + 2,5 = 7 Punkte)

a. Im folgenden soll eine Auktion mit zwei Bietern A und B mittels LTL formalisiert werden. Die beiden Bieter können unter gewissen Bedingungen Gebote abgeben oder die Auktion beenden.

Gegeben ist folgende LTL-Signatur:

$$\Sigma = \{bidA, bidB, close\}$$

Die Variablen sollen ausdrücken, ob eine Aktion zu einem Zeitpunkt ausgeführt wird. Es gilt also: $bidA$ ist wahr gdw. A ein Gebot abgibt. $bidB$ ist wahr gdw. B ein Gebot abgibt. $close$ ist wahr gdw. die Auktion beendet wird.

Formalisieren Sie:

i. Keine zwei Aktionen können gleichzeitig ausgeführt werden.

---

ii. Die Auktion kann erst geschlossen werden, wenn mindestens zwei Zeitschritte lang kein Gebot abgegeben wurde.

---

iii. Bieter A darf keine zwei Gebote hintereinander abgeben, d.h. zwischen zwei Geboten von A muss immer ein Gebot von B liegen.

---

b. Geben Sie einen nicht-deterministischen Büchi-Automaten an, dessen akzeptierte Sprache den Modellen (ω-Wörtern) der LTL-Formel

$$\Box b \wedge \Box\Diamond a$$

über der Signatur $\Sigma = \{a, b\}$ entspricht.

Für das Vokabular $V = \mathbb{P}(\Sigma)$ (Potenzmenge von Σ) werden die folgenden, aus der Vorlesung bekannten, Abkürzungen definiert:

$$A = \{M \in V \mid a \in M\} \subset V \qquad B = \{M \in V \mid b \in M\} \subset V$$
$$\bar{A} = \{M \in V \mid a \notin M\} \subset V \qquad \bar{B} = \{M \in V \mid b \notin M\} \subset V$$
$$AB = A \cap B \qquad \bar{A}B = \bar{A} \cap B$$
$$A\bar{B} = A \cap \bar{B} \qquad \bar{A}\bar{B} = \bar{A} \cap \bar{B}$$