Klausur Formale Systeme

Universität Karlsruhe
Fakultät für Informatik

WS 2005/2006

Prof. Dr. P. H. Schmitt

21. Februar 2006

Name: _____

Vorname: _____

Matrikel-Nr.: _____

Bitte geben Sie auf jedem benutzten Blatt rechts oben Ihren Namen und Ihre Matrikel-Nummer an!

A1 (12)	A2 (3)	A3 (3)	A4 (7)	A5 (6)	A6 (8)	A7 (7)	A8 (6)	A9 (8)	Σ (60)

Bewertungstabelle bitte frei lassen !!!

Zum Bestehen der Klausur benötigen Sie 20 der erreichbaren 60 Punkte.

Gesamtpunkte:

1 Zur Einstimmung (12 Punkte)

Kreuzen Sie in den folgenden Tabellen alles Zutreffende an.

Für jede falsche Antwort wird ein halber Punkt abgezogen!
(Dabei werden jedoch keinesfalls weniger als 0 Punkte für jede der drei Teilaufgaben vergeben.)

Hinweise:

- „PL1“ steht für „Prädikatenlogik erster Ordnung (mit Gleichheit $\doteq$)“; auf diese beziehen sich auch die Begriffe „erfüllbar“, „allgemeingültig“ und „unerfüllbar“.
- In Teilaufgabe a. kann eine Formel mehr als eine der genannten Eigenschaften haben. In Teilaufgabe b. und c. *genau* eine.
- c ist ein Funktionssymbol (mit der richtigen Stelligkeit).
- p, q, r, s sind Prädikatssymbole (mit der richtigen Stelligkeit).
- x, y sind Variablen.
- Es gelten die üblichen Klammereinsparungsregeln.

a.

	keine Formel der PL1	erfüllbar	allgemeingültig	unerfüllbar
$\exists x(p(x) \rightarrow \forall y(p(y)))$		×	×	
$\exists x(x \doteq x \wedge \forall x(\neg x \doteq c))$				×
$\mathbf{1 \doteq 0}$	×			
$\forall x(x \doteq p(x) \rightarrow x \doteq p(p(x)))$	×			
$(\neg(r \leftrightarrow s)) \leftrightarrow (\neg r \leftrightarrow s)$		×	×	

b.

	Richtig	Falsch
Sei F eine geschlossene PL1-Formel und σ eine Substitution. Dann gilt: $F \leftrightarrow \sigma(F)$ ist erfüllbar.	×	
Im Gegensatz zur Aussagenlogik gibt es in der PL1 Formeln, deren Allgemeingültigkeit mit dem Tableaukalkül nicht bewiesen werden kann.		×
Für beliebige PL1-Formeln F und G gilt: Wenn $\forall x(F \rightarrow G)$ nicht allgemeingültig ist, dann gibt es ein Modell M von $\forall xF$, so dass $M \models \forall xG$ nicht gilt.		×
Der Wahrheitswert einer PL1-Formel F kann sich durch Anwendung einer Substitution σ auf F verändern.	×	

c. Sind folgende LTL-Formeln allgemeingültig, d.h. gelten in allen omega-Strukturen?

LTL-Formel	Ja	Nein
$(A \rightarrow B) \rightarrow (\Diamond A \rightarrow \Diamond B)$		×
$\Box\Diamond A \rightarrow \Diamond\Diamond\Diamond A$	×	
$\Diamond\Diamond A \rightarrow \Diamond\Diamond\Diamond A$	×	

2 Konjunktive Normalform (3 Punkte)

Bringen Sie die folgende aussagenlogische Formel in konjunktive Normalform:

$$(((\neg P) \leftrightarrow Q)) \wedge ((P \vee Q) \rightarrow (P \wedge Q))$$

Lösung:

$$\begin{aligned}
&((((\neg P) \leftrightarrow Q)) \wedge ((P \vee Q) \rightarrow (P \wedge Q)) \\
\equiv&((\neg P \wedge Q) \vee (P \wedge \neg Q)) \wedge (\neg(P \vee Q) \vee (P \wedge Q)) \\
\equiv&((\neg P \wedge Q) \vee (P \wedge \neg Q)) \wedge (\neg P \wedge \neg Q) \vee (P \wedge Q)) \\
\equiv&((\neg P \vee P) \wedge (\neg P \vee \neg Q) \wedge (Q \vee P) \wedge (Q \vee \neg Q)) \\
&\wedge ((P \vee \neg P) \wedge (P \vee \neg Q) \wedge (Q \vee \neg P) \wedge (Q \vee \neg Q))
\end{aligned}$$

3 Davis-Putnam-Loveland Verfahren (3 Punkte)

Zeigen Sie mit dem Davis-Putnam-Loveland Verfahren die Unerfüllbarkeit der folgenden Klauselmenge:

$$\{\{P, Q, R\}, \{P, \neg R\}, \{\neg P, \neg Q\}, \{\neg P, Q, R\}, \{P, \neg Q\}, \{\neg R\}\}$$

Lösung:

1. Wahl der Einerklausel: $\{\neg R\}$
2. Reduktion: $\{\{P, Q\}, \{\neg P, \neg Q\}, \{\neg P, Q\}, \{P, \neg Q\}\}$
3. Wahl der Variable: P
4. Widerlege $S_P = \{\{P, Q\}, \{\neg P, \neg Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{P\}\}$
   (a) Reduktion: $\{\{\neg Q\}, \{Q\}\}$
   (b) Wahl der Einerklausel: $\{Q\}$
5. Widerlege $S_{\neg P} = \{\{P, Q\}, \{\neg P, \neg Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{\neg P\}\}$
   (a) Reduktion: $\{\{Q\}, \{\neg Q\}\}$
   (b) Wahl der Einerklausel: $\{Q\}$

4 Formalisieren in Prädikatenlogik (1+2+1+3 Punkte)

Gegeben sei folgende aus dem UML-Klassendiagramm in Abbildung 1 abgeleitete Signatur $\Sigma = (F_\Sigma, P_\Sigma, \alpha_\Sigma)$ der Prädikatenlogik erster Stufe:

- $F_\Sigma = \{kennzeichen\}$,
- $P_\Sigma = \{Person, Kind, Erwachsener, Fahrzeug, Fahrrad, Auto, Nummer, besitzt\}$,
- $\alpha_\Sigma(kennzeichen) = \alpha_\Sigma(Person) = \alpha_\Sigma(Kind) = \alpha_\Sigma(Erwachsener) = \alpha_\Sigma(Fahrzeug) = \alpha_\Sigma(Fahrrad) = \alpha_\Sigma(Auto) = \alpha_\Sigma(Nummer) = 1$
  und $\alpha_\Sigma(besitzt) = 2$

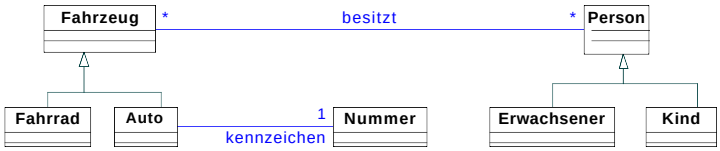


Abbildung 1: UML-Klassendiagramm

Die Bedeutung der Symbole ist wie folgt:

- Für jede Klasse K gibt es ein einstelliges Prädikatssymbol gleichen Namens, welches die Zugehörigkeit eines Objektes zu der Klasse K angibt, z.B. $Person, Kind$, usw.
- Für jeden Rollenbezeichner der Multiplizität 1 gibt es ein einstelliges Funktionssymbol gleichen Namens, hier also das einstellige Funktionssymbol *kennzeichen*, welches das Kennzeichen eines Autos liefert.
- Für jede benannte Assoziation gibt es ein zweistelliges Prädikatssymbol gleichen Namens, hier also *besitzt*. Die Formel $besitzt(x, y)$ bedeutet: x besitzt y.

Formalisieren Sie folgende Aussagen unter Verwendung der Signatur Σ so *exakt* wie möglich:

a. Kinder sind Personen, aber sie sind keine Erwachsenen.

Lösung:

$\forall x(Kind(x) \rightarrow (Person(x) \wedge \neg Erwachsener(x)))$

b. Über das Kennzeichen kann ein Auto eindeutig identifiziert werden.

Lösung:

$\forall x \forall y(Auto(x) \wedge Auto(y) \wedge kennzeichen(x) \doteq kennzeichen(y) \rightarrow x \doteq y)$

c. Kinder können nur Fahrräder besitzen.

Lösung:

$\forall x \forall y(Kind(x) \wedge besitzt(x, y) \rightarrow Fahrrad(y))$

d. Jedes Kind, das ein Fahrrad besitzt, teilt sich ein Fahrrad mit mindestens einem weiteren Kind.

Lösung:

$\forall k_1((Kind(k_1) \wedge \exists f_1(Fahrrad(f_1) \wedge besitzt(k_1, f_1))) \rightarrow$
$(\exists k_2 \exists f_2(Kind(k_2) \wedge Fahrrad(f_2) \wedge \neg(k_1 \doteq k_2) \wedge besitzt(k_1, f_2) \wedge besitzt(k_2, f_2))))$

5 Semantik der Prädikatenlogik (6 Punkte)

Definition 1 (Endliche Folgerbarkeitsrelation $\models_{fin}$) *Sei M eine Menge prädikatenlogischer Formeln und F eine prädikatenlogische Formel. Wir sagen F ist eine endliche Folgerung aus M, in Zeichen $M \models_{fin} F$, genau dann wenn jedes endliche Modell von M auch ein Modell von F ist.*

Beweisen oder widerlegen Sie durch Angabe eines Gegenbeispiels folgende Aussagen:

a. $\models_{fin} \quad \Rightarrow \quad \models$

Lösung:

Nein. Gegenbeispiel: Sei lt ein zweistelliges Prädikatssymbol, $succ$ ein einstelliges Funktionssymbol und $zero$ eine Konstante, dann hat die Formel

$$\begin{aligned}\phi = \; & \forall x(\neg lt(x,x)) \wedge \\ & \forall x \forall y \forall z(lt(x,y) \wedge lt(y,z) \rightarrow lt(x,z)) \wedge \\ & \forall x(lt(x, succ(x)))\end{aligned}$$

nur unendliche Modelle. Damit ist $\{\phi\} \models_{fin} \mathbf{0}$ trivialerweise allgemeingültig, während $\{\phi\} \models \mathbf{0}$ unter anderem für $D = \mathbb{N}$ nicht gilt.

b. $\models \quad \Rightarrow \quad \models_{fin}$

Lösung:

Wenn jedes Modell D einer Formelmenge M auch ein Modell von F ist, so ist inbesondere auch jedes endliche Modell von M ein Modell von F.

6 Tableaukalkül (8 Punkte)

Vervollständigen Sie folgendes Tableau zu einem geschlossenen Tableau. Verwenden Sie ausschließlich die im Skript angegebenen Tableauregeln.

1 1 $\forall x(p(f(x)) \rightarrow q(x))$
2 1 $\forall x(\neg q(x) \vee r(a,x))$
3 0 $((\exists x \exists y(p(f(x)) \wedge p(f(y)))) \rightarrow (\exists x \exists y(r(a,x) \wedge q(y))))$

Lösung:

(1;–) **1** $\forall x(p(f(x)) \rightarrow q(x))$
(2;–) **1** $\forall x(\neg q(x) \vee r(a,x))$
(3;–) **0** $((\exists x \exists y(p(f(x)) \wedge p(f(y)))) \rightarrow (\exists x \exists y(r(a,x) \wedge q(y))))$
(4;3) **1** $\exists x \exists y(p(f(x)) \wedge p(f(y))$
(5;3) **0** $\exists x \exists y(r(a,x) \wedge q(y))$
(6;4) **1** $\exists y(p(f(c)) \wedge p(f(y)))$
(7;6) **1** $p(f(c)) \wedge p(f(d))$
(8;7) **1** $p(f(c))$
(9;7) **1** $p(f(d))$
(10;5) **0** $\exists y(r(a,X_1) \wedge q(y))$
(11;10) **0** $r(a,X_1) \wedge q(Y_1)$
(12;1) **1** $p(f(X_2)) \rightarrow q(X_2)$
(13;12) **0** $p(f(X_2))$ | (14;12) **1** $q(X_2)$
(15;11) **0** $r(a,X_1)$ | (16;11) **0** $q(Y_1)$
(17;2) **1** $\neg q(X_3) \vee r(a,X_3)$
(18;17) **1** $\neg q(X_3)$ | (19;17) **1** $r(a,X_3)$
(20;18) **0** $q(X_3)$

Nach Anwendung der Substitution $\{X_1/c,\ X_2/c,\ X_3/c,\ Y_1/c\}$ sind alle Äste dieses Tableaus geschlossen.

7 Formalisieren in LTL (1+2+1+3 Punkte)

Im öffentlichen Busverkehrssystem von Nanocity verkehren 3 Linien zwischen 2 Endhaltestellen. Verwenden Sie folgende aussagenlogische Atome für die Formalisierung:

- H_iL_j dafür, dass Linie L_j an Haltestelle H_i steht (wobei $i \in \{1,2\}, j \in \{1,2,3\}$).

Zum Beispiel bedeutet H_1L_1, dass ein Bus der Linie L_1 an der Haltestelle H_1 hält.

Formalisieren Sie folgende Sachverhalte:

a. Zu keiner Zeit stehen zwei Busse von verschiedenen Linien an Haltestelle H_1.

Lösung:

$$\Box(\neg(H_1L_1 \wedge H_1L_2) \wedge \neg(H_1L_1 \wedge H_1L_3) \wedge \neg(H_1L_2 \wedge H_1L_3))$$

b. Jedesmal wenn ein Bus der Linie L_1 an der Haltestelle H_1 hält, wartet er solange bis ein Bus der gleichen Linie an Haltestelle H_2 hält.

Lösung:

$\Box(H_1L_1 \rightarrow (\mathbf{X}\ H_1L_1\ \mathrm{U_w}\ H_2L_1))$

c. An Haltestelle H_2 hält jede Linie immer wieder irgendwann an.

Lösung:

$$\Box(\Diamond(H_2L_1) \wedge \Diamond(H_2L_2) \wedge \Diamond(H_2L_3))$$

d. Linie L_1 muss solange an Haltestelle H_2 warten, bis direkt im Anschluss Linie L_2 oder Linie L_3 eintrifft.

Lösung:

$$\Box(H_2L_1 \rightarrow (H_2L_1\ \mathrm{U_w}\ (H_2L_2 \vee H_2L_3)))$$

8 LTL und Büchi-Automaten (5+1 Punkte)

Gegeben sei eine Signatur $\Sigma = \{p, q\}$ und das Alphabet $V = 2^{\Sigma} = \{a, b, c, d\}$ mit

$$a = \{\} \qquad b = \{p\} \qquad c = \{q\} \qquad d = \{p, q\}$$

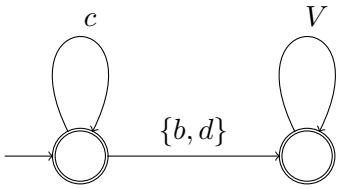
Gegeben sei folgende LTL-Formel ϕ:

$$\Box q \vee q \mathbf{U} p$$

a. Geben Sie einen Büchi-Automaten B_ϕ an, so dass gilt:

$$L^\omega(B_\phi) = \{\xi \in V^\omega \mid \xi \models \phi\}$$

Lösung:



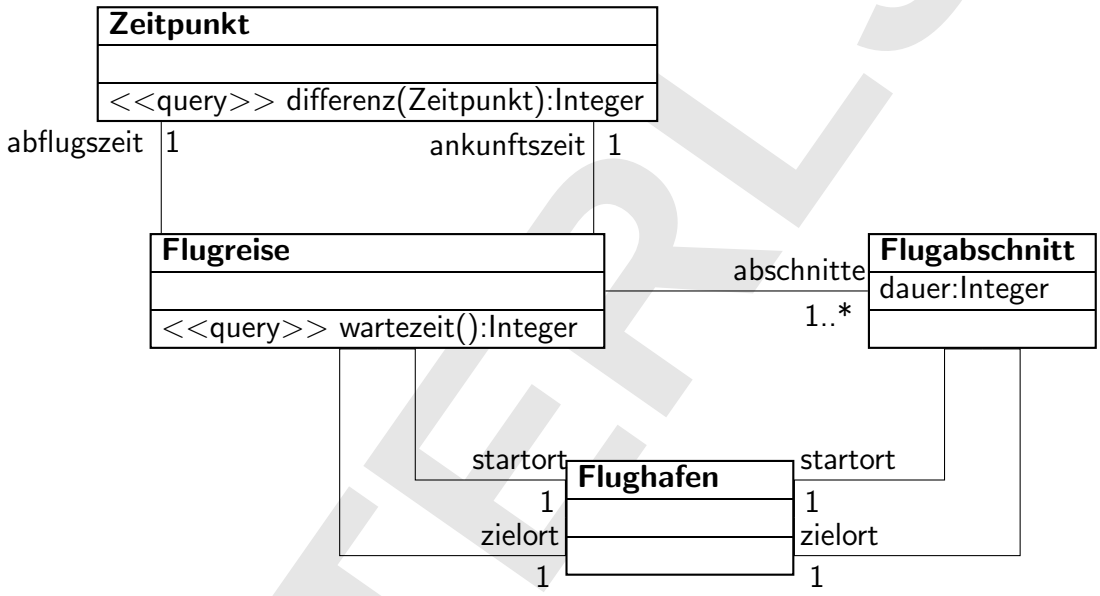
b. Geben Sie einen in der Vorlesung vorgestellten LTL-Operator $\circ$ an, so dass für alle Zeitstrukturen ξ gilt:

$$\xi \models p \circ q \quad \text{genau dann wenn} \quad \xi \models \phi$$

Lösung:

Der binäre Operator wurde nicht in der Vorlesung vorgestellt, deshalb bekommt jeder den Punkt!
$\circ$ entspricht fast dem Operator $\mathbf{V}$:

$$\xi \models p \mathbf{V} q \quad \text{genau dann wenn} \quad \xi \models \phi \wedge q$$



Die Query (Methode ohne Seiteneffekte) `differenz(z:Zeitpunkt):Integer` in der Klasse `Zeitpunkt` hat dabei folgende Funktionalität:
Ein Aufruf `a.differenz(b)` liefert die Zeit in Minuten, die zwischen den Zeitpunkten `a` und `b` liegt.
Das Attribut `dauer` in der Klasse `Flugabschnitt` gibt die Zeit in Minuten an, die ein Flug auf diesem Flugabschnitt dauert.

Abbildung zu Aufgabe 9

9 Object Constraint Language (1 + (2 + 2) + 3 Punkte)

Das links (auf der Rückseite von Blatt 9) dargestellte UML-Klassendiagramm sei gegeben.

a. Geben Sie einen OCL-Constraint an, der bestimmt, dass der Startort einer Flugreise nie gleich ist wie der Zielort.

Lösung:

```
context Flugreise
inv: self.startort <> self.zielort
```

b. Gegeben sei folgender OCL-Ausdruck:

```
reise.abschnitte->iterate(a; s:Set = Set{} |
     if (a.startort<>reise.startort)
        then s else s->including(a.startort) endif)
```

Hierbei bezeichne `reise` ein Objekt vom Typ `Flugreise`.
Der Ausdruck `Set{}` bezeichnet die leere Menge.

i. Geben Sie in natürlicher Sprache an, welche Menge dieser OCL-Ausdruck beschreibt.

Lösung:

die Menge, die genau den Startort von `reise` enthält (wenn der Reisestartort auch ein Startort eines Abschnitts ist; sonst ist die Menge leer).

ii. Geben Sie einen äquivalenten OCL-Ausdruck an, der *kein* `iterate` enthält.

Lösung:

```
reise.abschnitte->collect(a | a.startort)
                          ->select(s | s = reise.startort)->asSet
```

c. Die Operation wartezeit() berechnet die Zeit während einer Flugreise (in Minuten), die mit Warten verbracht wird (also genau die Differenz zwischen der Dauer einer Reise und der Summe der Dauer ihrer Flugabschnitte), egal wann die Operation aufgerufen wird. Geben Sie einen OCL-Constraint als Paar von Vor- und Nachbedingung an, der diese Eigenschaft ausdrückt.

Lösung:

```
context Flugreise::wartezeit():Integer
pre: true
post: result = self.ankunftszeit.differenz(self.abflugszeit)
       - self.abschnitte->collect(c|c.dauer)->sum()
```