Klausur Formale Systeme

Fakultät für Informatik

WS 2009/2010

Prof. Dr. Bernhard Beckert

18. Februar 2010

Name: Mustermann
Vorname: Peter
Matrikel-Nr.: 0000000

Klausur-ID: 0000

A1 (15)	A2 (10)	A3 (10)	A4 (6)	A5 (8)	A6 (11)	Σ (60)

Bewertungstabelle bitte frei lassen!

Zum Bestehen der Klausur sind 20 der erreichbaren 60 Punkte hinreichend.

Bonus: _____ **Gesamtpunkte:**

1 Zur Einstimmung ((4+2+3)+6 = 15 Punkte)

a. Bitte kreuzen Sie in den folgenden Tabellen die für die Formeln in der jeweiligen Logik zutreffende Eigenschaft an. Für korrekte Antworten erhalten Sie einen Punkt, für falsche Antworten wird ein Punkt abgezogen. Dabei werden jedoch nie weniger als 0 Punkte pro Tabelle vergeben.

PL1 (Prädikatenlogik 1. Stufe)	keine Formel der PL1	erfüllbar (aber nicht allgemeing.)	allgemeingültig (und erfüllbar)	unerfüllbar
$\forall x\,(p(x) \wedge \neg p(c))$				X
$\big((\exists x\, p(x)) \rightarrow p(c)\big) \leftrightarrow \forall x\,(p(x) \rightarrow p(c))$			X	
$\exists x \exists y\,(c \rightarrow p(x,y))$	X			
$p(c) \rightarrow \forall x\, p(x)$		X		

p ist ein Prädikatensymbol, c ist ein Konstantensymbol, die übrigen Bezeichner sind Variablen.

PL2 (Prädikatenlogik 2. Stufe)	erfüllbar (aber nicht allgemeing.)	allgemeingültig (und erfüllbar)	unerfüllbar
$\forall u \exists v \forall X\,\big(X(u) \leftrightarrow X(v)\big)$		X	
$\forall X \exists u\, X(u)$			X

u und v sind Individuenvariablen, X ist eine Prädikatvariable.

LTL (Lineare Temporale Logik)	erfüllbar (aber nicht allgemeing.)	allgemeingültig (und erfüllbar)	unerfüllbar
$(\mathbf{0}\ \mathbf{U}\ A) \leftrightarrow A$		X	
$(\Box\Diamond A) \leftrightarrow (\Diamond\Box A)$	X		
$(\Box\Box A) \wedge (\Diamond\neg A)$			X

A ist eine aussagenlogische Variable.

b. Bitte kreuzen Sie in der folgenden Tabelle das Zutreffende an. Für korrekte Antworten erhalten Sie einen Punkt, für falsche Antworten wird ein Punkt abgezogen. Dabei werden jedoch nie weniger als 0 Punkte für diese Teilaufgabe vergeben.

	Richtig	Falsch
Seien A und B beliebige aussagenlogische Formeln. Wenn A, B beide erfüllbar sind, dann ist $A \rightarrow B$ allgemeingültig.		X
Sei K ein beliebiger Kalkül für die Aussagenlogik. Wenn K vollständig ist, dann gilt alle aussagenl. Formeln ϕ: $\vdash_K \phi$ oder $\vdash_K \neg\phi$.		X
Sei (S, R, I) eine beliebige Kripke-Struktur und A eine aussagenlogische Variable. Wenn $\Box A \rightarrow A$ in (S, R, I) wahr ist, dann ist R reflexiv.		X
Jede prädikatenlogische Formel hat ein Herbrand-Modell.		X
Das Problem für LTL-Formeln zu entscheiden, ob sie erfüllbar sind, kann auf das Problem reduziert werden, für Büchi-Automaten zu entscheiden, ob die von ihnen akzeptierte Menge von ω-Wörtern leer ist.	X	
Jeder syntaktisch korrekte OCL-Ausdruck hat einen Typ.	X	

2 Formalisieren in Aussagenlogik ((2+1+1)+(3+3) = 10 Punkte)

a. Die *Enzyklopädie der Unsichtbaren Universität* hat einiges über Einhörner zu sagen. Stellen Sie dieses Wissen in Aussagenlogik dar, und zwar

i. in Form von Klauseln
ii. als Horn-Formeln (geschrieben als Implikation).

Benutzen Sie jeweils die in Klammern angegebenen Variablennamen.

i. Wenn es ein Fabelwesen (F) ist, ist es unsterblich (U); wenn es aber kein Fabelwesen (F) ist, ist es auf jeden Fall kein Säugetier (S).

Klausel(n):
$\boxed{\neg F \vee U}$
$\boxed{F \vee \neg S}$

Horn-Formel(n):
$\boxed{F \rightarrow U}$
$\boxed{S \rightarrow F}$

ii. Wenn es unsterblich (U) und ein Fabelwesen (F) ist, hat es ein Horn (H).

Klausel(n):
$\boxed{\neg U \vee \neg F \vee H}$

Horn-Formel(n):
$\boxed{U \wedge F \rightarrow H}$

iii. Unsterbliche (U) Säugetiere (S) gibt es nicht.

Klausel(n):
$\boxed{\neg U \vee \neg S}$

Horn-Formel(n):
$\boxed{U \wedge S \rightarrow 0}$

b. Benutzen Sie den Markierungsalgorithmus, um zwei Gerüchte über Einhörner anhand der obigen Aussagen zu widerlegen. Geben Sie dabei die Reihenfolge der Literalmarkierungen an.

1. Es ist ein Säugetier (S).

Gerücht 1 als zusätzliche Hornfomel(n):
$\boxed{S}$

Literale in Markierungsreihenfolge:
1. S
2. F
3. U
4. 0

2. Es ist ein Fabelwesen (F) und hat kein Horn (H).

Gerücht 2 als zusätzliche Hornfomel(n):
$\boxed{F}$
$\boxed{H \rightarrow 0}$

Literale in Markierungsreihenfolge:
1. F
2. U
3. H
4. 0

3 Formalisieren in Prädikatenlogik (3+3+4 = 10 Punkte)

Gegeben sei:

- Die prädikatenlogische Signatur Σ mit den zweistelligen Funktionszeichen $+, *$ und der Konstanten 1.
- Die prädikatenlogische Interpretation $\mathcal{A} = (\mathbb{N}, I)$ mit:
  - Universum $\mathbb{N}$ (natürliche Zahlen ohne Null)
  - I interpretiert $+, *, 1$ wie in der Arithmetik üblich (Addition, Multiplikation, Zahl 1).
- Eine beliebige Variablenbelegung β

Geben Sie im folgenden jeweils eine Formel der Prädikatenlogik 1. Stufe mit Gleichheit an, die die geforderte Eigenschaft hat.

Hinweis: Benutzen sie in den Formeln ggf. die freien Variablen x und y.

1. $(\mathcal{A}, \beta) \models \mathit{Teilt}(x, y)$ gdw. $\beta(x)$ teilt $\beta(y)$ ohne Rest.

   Formel $\mathit{Teilt}(x, y)$:

$\exists z\,(y \doteq x * z)$

---

2. $(\mathcal{A}, \beta) \models \mathit{Prim}(x)$ gdw. $\beta(x)$ ist eine Primzahl.
   Hinweis: Sie dürfen $\mathit{Teilt}(x, y)$ verwenden.

   Formel $\mathit{Prim}(x)$:

Je nachdem, ob man die 1 bei der Definition der Primzahlen ausschließt oder nicht (beides richtig): $\forall z\,(\mathit{Teilt}(z, x) \rightarrow (z \doteq 1 \vee z \doteq x))$ oder $\neg(x \doteq 1) \wedge \forall z\,(\mathit{Teilt}(z, x) \rightarrow (z \doteq 1 \vee z \doteq x))$

---

3. $\mathcal{A} \models G$ gdw. Jede gerade Zahl ist die Summe zweier Primzahlen.[1]
   Hinweis: Sie dürfen $\mathit{Teilt}(x, y)$ und $\mathit{Prim}(x)$ verwenden.
   Hinweis: Beachten Sie, dass es keine Konstante 2 in der Signatur gibt.

   Formel G :

$\forall x\,\big(\mathit{Teilt}((1+1), x) \rightarrow \exists u \exists v(\mathit{Prim}(u) \wedge \mathit{Prim}(v) \wedge x \doteq u + v)\big)$ Hat man die 1 als Primzahl ausgeschlossen, dann gilt die Goldbachsche Vermutung nicht für 2. Auch wenn es nicht der obigen informellen Formulierung entspricht, wird darum auch als richtig anerkannt: $\forall x\,\big((\mathit{Teilt}((1+1), x) \wedge \neg(x \doteq (1+1))) \rightarrow \exists u \exists v(\mathit{Prim}(u) \wedge \mathit{Prim}(v) \wedge x \doteq u + v)\big)$

---

[1] Dies ist die „Goldbachsche Vermutung“.

4 Resolution (6 Punkte)

Seien

- p, r einstellige Prädikatensymbole,
- f ein einstelliges Funktionssymbol,
- c eine Konstante,
- x eine Variable.

Gegeben seien folgende vier Klauseln:

$\{p(x), r(x)\}$ $\{\neg p(f(c)), p(c)\}$ $\{p(c)\}$ $\{\neg p(x), p(f(x))\}$

Geben Sie genau die Klauseln an, die sich daraus in **einem** Resolutionsschritt ableiten lassen. Tragen Sie diese in die folgende Tabelle ein.

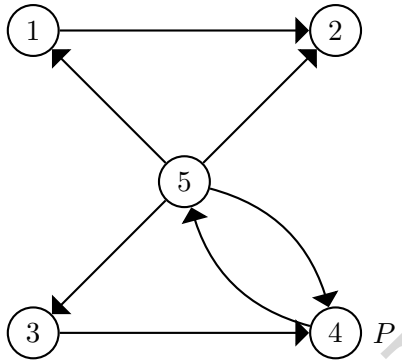
Wenn ein Paar von Klauseln keine Resolvente hat, markieren Sie dies in der Tablelle durch ein Kreuz (und nicht dadurch, dass Sie das Feld frei lassen).

In die grau hinterlegten Felder brauchen Sie nichts einzutragen.

	$\{p(x), r(x)\}$	$\{\neg p(f(c)), p(c)\}$	$\{p(c)\}$	$\{\neg p(x), p(f(x))\}$
$\{p(x), r(x)\}$	X			
$\{\neg p(f(c)), p(c)\}$	$\{p(c), r(f(c))\}$	X		
$\{p(c)\}$	X	X	X	
$\{\neg p(x), p(f(x))\}$	$\{p(f(x)), r(x)\}$	$\{p(f(c)), \neg p(f(c))\}$ $\{p(c), \neg p(c)\}$	$\{p(f(c))\}$	$\{\neg p(x), p(f(f(x)))\}$

5 Modallogik (4+4 = 8 Punkte)

Sei $\mathcal{K}$ die folgende Kripke-Struktur. Die atomare Aussage P ist dabei (wie eingezeichnet) nur in der Welt 4 wahr und in den anderen Welten falsch.



a. Geben Sie für jede der folgenden Formeln genau die Welten von $\mathcal{K}$ an, in welchen sie wahr ist:

i. $\Box P$ gilt in: 2, 3

ii. $\Diamond\neg P$ gilt in: 1, 4, 5

iii. $\Diamond\Diamond \mathbf{1}$ gilt in: 3, 4, 5

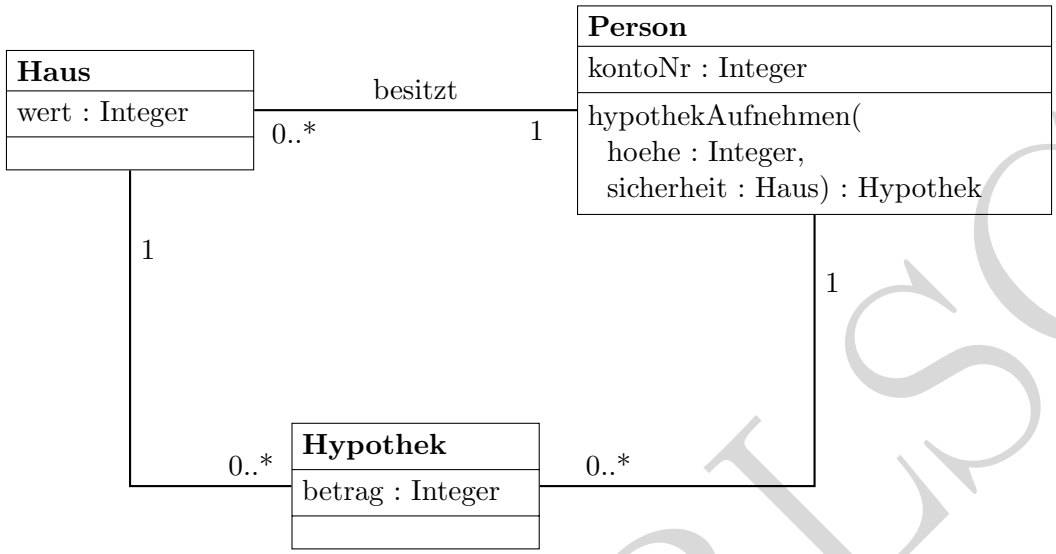
b. Geben Sie eine modallogische Formel an, die

i. Nur in der Welt 3 gilt: $\Box P \wedge \Diamond\Diamond \mathbf{1}$

ii. Nur in der Welt 4 gilt: P

iii. Nur in den Welten 3 und 4 gilt: $(\Box P \wedge \Diamond\Diamond \mathbf{1}) \vee P$

UML-Diagramm zu Aufgabe 6



Übersicht über wichtige OCL-Operationen

Folgende Operationen sind auf alle Gesamtheiten (Mengen, Multimengen und Listen) anwendbar.

Operation	**Ergebnis** (bei Anwendung auf M)
`size()`	die Anzahl der Elemente in M.
`count(o)`	die Anzahl der Vorkommen von o in M.
`sum()`	die Summe der Elemente einer Menge von Zahlen.
`including(o)`	die Gesamtheit, die M erweitert um o entspricht.
`collect(v|exp)`	die Gesamtheit, die entsteht, wenn der Audruck exp für jedes Element in M ausgewertet wird.
`select(v|b)`	die Gesamtheit der Elemente v von M, die b erfüllen.
`intersection(N)`	der Durchschnitt von M und N.
`union(N)`	die Vereinigung von M und N.
`includes(o)`	wahr genau dann, wenn o ein Element in M ist.
`includesAll(N)`	wahr genau dann, wenn jedes Element n der Gesamtheit N auch ein Element in M ist.
`isEmpty()`	wahr genau dann, wenn M kein Element enthält.
`exists(v|b)`	wahr genau dann, wenn es ein Element v in M gibt, so dass dafür der boolesche Ausdruck b zu wahr auswertet.
`forAll(v|b)`	wahr genau dann, wenn für jedes Element v in M der boolesche Ausdruck b zu wahr auswertet.

Der folgende Operator ist auf Klassennamen anwendbar

`allInstances()`	der Ausdruck `Class.allInstances()` liefert alle Objekte in der Klasse `Class`.

6 OCL (4+(4+3) = 11 Punkte)

Auf der linken Seite ist ein UML-Klassendiagramm abgebildet. Es modelliert den Zusammenhang zwischen Personen, Häusern und Hypotheken: Eine Person kann Häuser besitzen und kann sie als Sicherheit benutzen, um Hypotheken (Kredite) darauf aufzunehmen.

Vervollständigen Sie die folgenden OCL-Constraints gemäß der angegebenen natürlichsprachlichen Bedeutung.

a. Verschiedene Personen haben verschiedene Kontonummern.

```
context Person
inv:
```

```
Person.allInstances()->
    forAll(p | p.kontoNr = self.kontoNr implies p = self)
```

b. Man kann nur dann eine Hypothek auf ein Haus aufnehmen, wenn die Summe der Beträge der bereits auf dieses Haus aufgenommenen Hypotheken plus die Höhe der neu beantragten Hypothek den Wert des Hauses nicht übersteigt.

```
context Person::hypothekAufnehmen(hoehe : Integer, sicherheit : Haus) : Hypothek
pre:
```

```
sicherheit.hypothek.betrag->sum() + hoehe <= sicherheit.wert
```

Wenn die o.g. Vorbedingung erfüllt ist, vergrößert die Ausführung von `hypothekAufnehmen(...)` die Menge der Hypotheken, die zu der Person gehören, die die neue Hypothek aufnimmt, genau um die Hypothek, die als Ergebnis der Methode zurückgegeben wird.

```
post:
```

```
self.hypothek = self.hypothek@pre->including(result)
```