RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHR- UND FORSCHUNGSGEBIET INFORMATIK II
RWTH Aachen · D-52056 Aachen · GERMANY
http://www-i2.informatik.rwth-aachen.de/lufgi2

LuFG Informatik II

Prof. Dr. Jürgen Giesl
Peter Schneider-Kamp
René Thiemann

Logikprogrammierung – Scheinklausur

Bearbeitungszeit: 90 Minuten

Vorname: ____

Name: ____

Matrikelnummer: ____

- Schreiben Sie auf jedes Blatt Vorname, Name und Matrikelnummer.
- Bitte beantworten Sie die Aufgaben in gut lesbarer Schrift auf den Aufgabenblättern (benutzen Sie auch die Rückseiten). Schreiben Sie *nicht* mit einem Bleistift oder einem roten Stift. Was nicht bewertet werden soll, kennzeichnen Sie bitte durch Durchstreichen.
- Werden Täuschungsversuche beobachtet, so wird die Klausur mit nicht bestanden gewertet.
- Entfernen Sie bitte nicht die Klammerung der Klausur.

Aufgabe	Maximale Punkte	Erreichte Punkte
1	6	
2	4	
3	6	
4	6	
5	8	
Σ	30	
Note		

Vorname	Name	Matr.-Nr.

Aufgabe 1 (6 Punkte)

Bitte kreuzen Sie für jede Aussage an, ob sie stimmt oder nicht. Jede richtige Antwort gibt einen Punkt. Jede falsche Antwort gibt einen halben Punkt Abzug. Unbeantwortete Aussagen geben keinen Punktabzug.

	Richtig	Falsch
Jede Formel lässt sich in eine äquivalente Formel in Skolem-Normalform umformen.		
Jede erfüllbare Formel besitzt ein Modell mit abzählbarem Träger.		
Input-Resolution ist vollständig, d.h. die leere Klausel $\square$ ist aus jeder unerfüllbaren Klauselmenge mit Input-Resolution herleitbar.		
SLD-Resolution ist vollständig, d.h. die leere Klausel $\square$ ist aus jeder unerfüllbaren Klauselmenge mit SLD-Resolution herleitbar.		
Es existieren nicht-primitiv rekursive Funktionen, die durch ein Logikprogramm berechnet werden können.		
Jedes Logikprogramm, das aus der Übersetzung einer primitiv rekursiven Funktion entsteht, erzeugt für jede Anfrage einen endlichen SLD-Baum.		

Vorname	Name	Matr.-Nr.

Aufgabe 2 (2 + 2 Punkte)

Gegeben sei die folgende Formel φ über der Signatur (Σ, Δ) mit $\Sigma = \Sigma_0 = \{\mathsf{a}\}$ und $\Delta = \Delta_1 = \{\mathsf{p}\}$:

$$\forall X \;(\;(\;\forall X\; \mathsf{p}(X)\;) \rightarrow \mathsf{p}(X)\;)$$

a) Überführen Sie φ zunächst in Pränex- und dann in Skolem-Normalform.

b) Geben sie alle Strukturen an, die ein Herbrand-Modell von φ sind.
Wieviele solche Strukturen gibt es?

Vorname	Name	Matr.-Nr.

Aufgabe 3 (1,5 + 2 + 2,5 Punkte)

Sei $\mathcal{P}$ ein Logikprogramm, A eine atomare Formel, $G = \{\neg A\}$ und σ eine Substitution. Beweisen Sie die folgenden Aussagen.

a) $D[\![\mathcal{P}, \sigma(G)]\!] \subseteq D[\![\mathcal{P}, G]\!]$

b) Falls $\sigma(A)$ eine *Grund*instanz von A ist und $(\sigma(G), \emptyset) \vdash_{\mathcal{P}}^{+} (\Box, \delta)$ gilt, dann ist $P[\![\mathcal{P}, \sigma(G)]\!] = \{\sigma(A)\}$.

c) Falls $\sigma(A)$ eine *Grund*instanz von A ist und $(\sigma(G), \emptyset) \vdash_{\mathcal{P}}^{+} (\Box, \delta)$ gilt, dann gilt auch $(G, \emptyset) \vdash_{\mathcal{P}}^{+} (\Box, \gamma)$, wobei $\sigma(A)$ eine Instanz von $\gamma(A)$ ist.
Tipp: Nutzen Sie a) und b).

Vorname	Name	Matr.-Nr.

Aufgabe 4 (6 Punkte)

Gegeben sei das folgende Prolog-Programm $\mathcal{P}$. Das Prädikat `len(Xs,Y)` ist genau dann wahr, wenn `Xs` eine Liste von atomaren Formeln ist, von denen `Y` viele nicht beweisbar sind.

```
p(s(X),X).

len([], 0).
len([X|Xs], Y) :- not(X),!,len(Xs,Z),Y is Z+1.
len([_|Xs], Y) :- len(Xs,Y).
```

Geben Sie den SLD-Baum an, der bei der Anfrage "`?- len([p(X,0),p(0,X)],Y).`" entsteht. Sie können gleiche Teile mit "———— `"` ————" abkürzen.

Tipp: Überlegen Sie sich zunächst, durch welche zwei Klauseln das Prädikat `not/1` vordefiniert ist.

Vorname	Name	Matr.-Nr.

Aufgabe 5 (2 + 3 + 3 Punkte)

a) Implementieren Sie in Prolog ein Prädikat `fac`, so dass `fac(X,Y)` genau dann wahr ist, falls `Y` die Fakultät von `X` ist. Nutzen Sie die hierfür die eingebauten ganzen Zahlen. Die Fakultät ist wie folgt definiert:

$$fac(n) = \begin{cases} 1, & \text{falls } n \leq 0 \\ fac(n-1) \cdot n, & \text{sonst} \end{cases}$$

b) Eine *simple arithmetic expression (SAE)* ist eine Zahl oder ein Term der Form `mal(E1,E2)`, wobei die Argumente `E1` und `E2` wieder SAEs sind. Implementieren Sie ein Prädikat `eval(E,N)` in Prolog, welches genau dann beweisbar ist, wenn sich die SAE `E` zu der Zahl `N` auswerten lässt. Hierbei steht die SAE `mal(E1,E2)` für die Multiplikation von `E1` mit `E2`. Bei der Auswertung von `mal(E1,E2)` sollte `E2` nur ausgewertet werden, falls die Auswertung von `E1` nicht `0` ergibt.

Die Anfrage "`?- eval(mal(mal(5,0), mal(7,7)), N).`" sollte beweisbar sein und die Antwort `N=0` liefern. Hierbei sollte jedoch die SAE `mal(7,7)` nicht ausgewertet werden.

c) Wir erweitern die Klasse der SAEs um bedingte Ausdrücke, die die Form "`B ? E1 : E2`" haben. Hierbei ist `B` eine Bedingung, d.h. eine atomare Formel, und `E1` und `E2` sind SAEs.

Definieren Sie `?` und `:` als Operatoren, so dass bedingte Ausdrücke vom Prolog-System akzeptiert werden. Erweitern Sie dann `eval` für die Auswertung von bedingten Ausdrücken. Falls `B` beweisbar ist, so wertet der obige bedingte Ausdruck zum Resultat von `E1` aus. Falls man feststellt, dass `B` nicht beweisbar ist, dann ergibt sich das Resultat von `E2`. Auch hier sollten unnötige Auswertungen vermieden werden.

Die Anfrage "`?- eval((3 < 2) ? mal(3,3) : mal(4,4), N)`" sollte folglich die Antwort `N=16` liefern, ohne dabei `mal(3,3)` auszuwerten. In gleicher Weise sollte die Anfrage "`?- eval((2 < 3) ? mal(3,3) : mal(4,4), N)`" die Antwort `N=9` liefern, ohne dabei `mal(4,4)` auszuwerten.