Prof. Dr. Günther Görz
Lehrstuhl Informatik 8 (Theoretische Informatik)
der Universität Erlangen-Nürnberg

27. März 2012

Bachelorprüfung
Grundlagen der Logik und Logikprogrammierung
— März 2012

Angaben zur Person (Bitte in Druckschrift ausfüllen!):

Name, Vorname: .....

Geburtsdatum: .....

Matrikelnummer: .....

Studienfach (Bitte ankreuzen):

☐ Informatik ☐ Wirtschafts-informatik ☐ Linguistische Informatik ☐ Computational Engineering ☐ Mathematik ☐ Techno-mathematik

☐ Sonstiges (Bitte angeben): .....

Nicht von der Kandidatin bzw. vom Kandidaten auszufüllen !!!

Punktzahlen:

Aufgabe	1	2	3	4	5	Σ
Max. Punkte	20	20	10	10	30	90
Erreichte Punkte						

Note: ☐ **Bonuspunkte:** ☐ **Endnote:** ☐

Organisatorische Hinweise

Die folgenden Hinweise bitte aufmerksam lesen und die Kenntnisnahme durch Unterschrift bestätigen!

- Bitte legen Sie Ihren Studentenausweis und einen Lichtbildausweis zur Personenkontrolle bereit!
- Die angegebene Punkteverteilung gilt unter Vorbehalt.
- Hilfsmittel (außer Schreibmaterial) sind **nicht** zugelassen.
- Fragen zu den Prüfungsaufgaben werden grundsätzlich **nicht** beantwortet!
- Wenn Sie die Prüfung aus gesundheitlichen Gründen abbrechen müssen, so muss Ihre Prüfungsunfähigkeit durch eine Untersuchung in der Universitätsklinik nachgewiesen werden. Melden Sie sich in jedem Fall bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.
- Die Bearbeitungszeit beträgt 90 Minuten.
- **Überprüfen Sie die Prüfungsaufgaben auf Vollständigkeit (12 Seiten inklusive Deckblatt) und einwandfreies Druckbild!** Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person einzutragen!
- Verwenden Sie für die Darstellung der Wahrheitswerte für wahr `t`, `w`, $\top$ oder 1 und für falsch `f`, $\bot$ oder 0.

  Ausnahmen werden explizit in der jeweiligen Aufgabenstellung gestattet!

Erklärung

Durch meine Unterschrift bestätige ich den Empfang der vollständigen Klausurunterlagen und die Kenntnisnahme der obigen Informationen.

Erlangen, 27. März 2012 .....
(Unterschrift)

Ich bin damit einverstanden, dass mein Prüfungsergebnis unter Angabe der Matrikelnummer anonymisiert veröffentlicht wird:

ja: ☐ nein: ☐

Erlangen, 27. März 2012 .....
(Unterschrift)

Aufgabe 1 (20 Punkte)

Beantworten Sie jeweils in wenigen vollständigen Sätzen, gegebenenfalls unter Zuhilfenahme von Formeln oder Beispielen, die nachfolgenden Fragen.

a) Welche drei Eigenschaften müssen für eine Relation gelten, damit sie *Äquivalenzrelation* genannt wird?

b) Wann nennt man eine Aussage *konsistent*?

c) Was ist ein *Literal*?

d) Welche Eigenschaft muss gelten, damit eine KNF als *Horn-Formel* bezeichnet wird?

e) Wann ist eine (quantorenlogische) Formel in *Pränexform*?

f) Was ist die *Closed World Assumption*?

Aufgabe 2 (20 Punkte)

Logik

a) Wodurch unterscheidet sich ein klassisch geführter Dialog von einem effektiv geführten?

b) Zeigen Sie die Allgemeingültigkeit von $(a \to c) \to ((b \to c) \to (a \vee (b \to c)))$ mithilfe eines effektiven Dialogs.

c) Ist die Aussage $(a \wedge \neg b) \wedge \Big(\big((c \longrightarrow a) \longrightarrow a \big) \longrightarrow c \Big)$ kontradiktorisch? Begründen Sie! Geben Sie, wenn möglich, ein Modell für die Aussage an!

d) Aussagenlogische Resolution: Zeigen Sie mit der *Resolutionsmethode*, dass

$$(b \wedge \neg a \wedge c) \vee (\neg b \wedge \neg d) \vee (c \wedge a \wedge b) \vee (b \wedge \neg c) \vee (d \wedge \neg b)$$

eine Tautologie ist.

e) Aussagenlogische Resolution: Zeigen Sie mit Hilfe der *Input-Resolution*, dass

$$(a \wedge b) \vee \neg a \vee (\neg b \wedge \neg c) \vee c$$

eine Tautologie ist.

Aufgabe 3 (10 Punkte)

Kreuz-Kringel-Kalkül

Gegeben sei folgender (Ihnen wohl bekannter) Kreuz-Kringel-Kalkül:

$$\begin{array}{rcl} & \Longrightarrow & \circ \\ & \Longrightarrow & + \\ a & \Longrightarrow & a \circ \\ a & \Longrightarrow & + \, a \, + \end{array}$$

a) Prüfen Sie, welche der folgenden Zeichenreihen ableitbar sind und welche nicht. Geben Sie, falls möglich, die einzelnen Ableitungsschritte an.

i) $\circ \circ \circ$

ii) $\circ \circ \circ +$

iii) $+ + + + \circ + \circ + \circ + \circ + \circ \circ$

b) Prüfen Sie, ob die folgenden Regeln im Kreuz-Kringel-Kalkül zulässig sind.

i) $a \Longrightarrow + \, a \circ +$

ii) $\circ \circ \Longrightarrow + +$

Aufgabe 4 (10 Punkte)

Unifikation

Sind die folgenden Ausdrücke unifizierbar? Falls ja, geben Sie jeweils einen Unifikator an (Großbuchstaben sind Variablen, Kleinbuchstaben sind Konstanten)!

a) $f(X) = f(a)$

b) $f(X) = f(a, b)$

c) $f(a(b(c))) = f(a(b(C)))$

d) $f(a(b), a(b)) = f(C, C)$

e) $a(b(X), [1, 2, 3]) = a(b([1, 2, 3]), X)$

f) $a(X, Y) = a(f(Y), f(X))$

g) $[[a, b], a, b] = [X|X]$

h) $[a(X), a(b)] = [Y, Y]$

i) $[a|[b|[a|[b]]]] = [X, X]$

j) $[a|[b|[a|[b]]]] = [X, Y, X, Y]$

Aufgabe 5 (30 Punkte)

Prolog

Schreiben Sie folgende Prolog-Programme. Verwenden Sie dazu keine schon vordefinierten Prolog-Prädikate (Ausnahme: arithmetische Prädikate), d.h. Prädikate wie `append/3`, `reverse/2`, `delete/3`, etc. dürfen von Ihnen nicht verwendet werden. Sie können gegebenenfalls in anderen Teilaufgaben schon entwickelte Programme wiederverwenden. Schreiben Sie

a) ein Prolog-Programm `sum/2`, welches die Summe einer Liste von Zahlen berechnet. Beispiel eines Aufrufs:

```
?- sum([8,12,7,15], X).
X = 42.
```

b) ein Prolog-Programm `prod/3`, welches das Skalarprodukt von zwei Listen von Zahlen berechnet. Sie können davon ausgehen, dass beide Listen gleich lang sind. Beispiel eines Aufrufs:

```
?- prod([4,3,7,5], [2,4,1,3], X).
X = 42.
```

c) ein Prolog-Programm `append/3`, welches zwei Listen aneinander hängt. Beispiel eines Aufrufs:

```
?- append([a,b,c], [d, e], X).
X = [a, b, c, d, e].
```

d) ein Prolog-Programm `reverse/2`, welches die Reihenfolge der Elemente einer Liste umkehrt. Beispiel eines Aufrufs:

```
?- reverse([1,2,3,4,5], X).
X = [5, 4, 3, 2, 1].
```

e) ein Prolog-Programm `isSubset/2`, welches testet, ob die erste Liste eine Teilmenge der zweiten ist. Sie können davon ausgehen, dass beide Mengen jedes Element nur einmal enthalten. Beispiel eines Aufrufs:

```
?- isSubset([e,a,d], [a,b,c,d,e,f]).
true .
?- isSubset([e,a,d,i], [a,b,c,d,e,f]).
false .
```

f) ein Prolog-Programm `findMin/2`, welches in einer Liste von Zahlen das kleinste Element findet. Sie können davon ausgehen, dass die Liste mindestens ein Element enthält. Beispiel eines Aufrufs:

```
?- findMin([3,7,2,5,2,8], X).
X = 2.
```

g) ein Prolog-Programm `removeFirst/3`, welches in einer Liste das erste Vorkommen eines bestimmten Elementes entfernt. Alle weiteren Vorkommen des Elements sollen davon unberührt bleiben. Beispiel eines Aufrufs:

```
?- removeFirst([9,3,6,1,5,4,6,8], 6, X).
X = [9, 3, 1, 5, 4, 6, 8].
```

h) ein Prolog-Programm `selectionSort/2`, welches mit Hilfe der Prädikate aus den vorherigen beiden Aufgaben ein Liste von Zahlen nach dem sogenannten SelectionSort-Verfahren aufsteigend sortiert: Dabei wird nach und nach das kleinste Element aus der übergebenen Liste entfernt und zur Ergebnisliste hinzugefügt. Beispiel eines Aufrufs:

```
?- selectionSort([9,3,6,1,5,4,6,8], X).
X = [1, 3, 4, 5, 6, 6, 8, 9].
```