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

23. September 2011

Bachelorprüfung
Grundlagen der Logik und Logikprogrammierung
— September 2011

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	25	15	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 (13 Seiten inklusive Deckblatt) und einwandfreies Druckbild!** Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person einzutragen!
- Verwenden Sie für die Darstellung der Wahrheitswerte `t` bzw. $\top$ und `f` bzw. $\bot$.

  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, 23. September 2011 .....
(Unterschrift)

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

ja: ☐ nein: ☐

Erlangen, 23. September 2011 .....
(Unterschrift)

Aufgabe 1 (25 Punkte)

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

a) Was ist eine *Tautologie*?

b) Was ist eine *Kontradiktion*?

c) Was ist die *"hinreichende Bedingung"* und was die *"notwendige Bedingung"* in der Formel $A \to B$?

d) Welche Art von Dialogführung bezeichnet man als *Gewinnstrategie*?

e) Welche Form besitzen Aussagen in KNF?

f) Welche Form besitzen Aussagen in DNF?

g) Welche syntaktische Eigenschaft zeichnet Horn-Formeln aus?

h) Was ist *lineare Resolution*?

Aufgabe 2 (15 Punkte)

Logik

a) Erklären Sie die LDF-Regel! Welche Auswirkung hat ihre Anerkennung bzw. Ablehnung auf die Art des Dialogs?

b) Zeigen Sie die Gültigkeit von *duplex negatio affirmat* $\neg\neg A \to A$ mithilfe eines effektiven Dialogs unter Verwendung des *tertium non datur* als Hypothese.

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

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

$$(A \wedge B \wedge \neg C) \vee (\neg A \wedge B) \vee (\neg A \wedge \neg B) \vee (\neg B \wedge \neg C) \vee C$$

eine Tautologie ist.

Aufgabe 3 (10 Punkte)

Kreuz-Kringel-Kalkül

Gegeben sei folgendes (Ihnen wohl bekanntes) 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) $\circ + \circ +$

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

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

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

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

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

Aufgabe 4 (10 Punkte)

Unifikation

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

a) $f(a(b)) = f(a(B))$

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

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

d) $a(b(B)) = a(B)$

e) $X = [a, b, c]$

f) $[X, b, Z] = [a, Y, c]$

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

h) $[c(X), c(b)] = [Z, Z]$

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

j) $[a(b), a(X)] = [X, X]$

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, `mean/2`, welches den Durchschnitt einer Liste von Zahlen berechnet. Beispiel eines Aufrufs:

```
?- mean([1,2,3,4], X).
X = 2.5.
```

b) 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].
```

c) 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].
```

d) ein Prolog-Programm, `zip/3`, welches zwei Listen nach dem Reissverschlussprinzip vereinigt. Beispiel eines Aufrufs:

```
?- zip([1,2,3], [a,b,c], X).
X = [1, a, 2, b, 3, c] .
```

e) ein Prolog-Programm, `unzip/3`, welches eine Liste nach dem Reissverschlussprinzip in zwei Listen teilt. Beispiel eines Aufrufs:

```
?- unzip([1,a,2,b,3,c], L, R).
L = [1, 2, 3],
R = [a, b, c].
```

f) ein Prolog-Programm, `removeAll/3`, welches aus einer anderen Liste alle Elemente entfernt, die in einer zweiten Liste vorkommen. Beispiel eines Aufrufs:

```
?- removeAll([1,2,6,3,4,5], [2,3,6], X).
X = [1, 4, 5].
```

g) ein Prolog-Programm, `double/2`, welches alle Elemente einer Liste dupliziert. Beispiel eines Aufrufs:

```
?- double([1,2,3], X).
X = [1, 1, 2, 2, 3, 3].
```

h) ein Prolog-Programm, `flatten/2`, welches beliebig verschachtelte Listen in eine flache Liste umwandelt. Beispiel eines Aufrufs:

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