Fachhochschule Wedel
Prof. Dr. Ralf Möller

5.9.2002

Aufgaben zur zweiten Übergangsprüfung Grundlagen der Programmierung im SS 2002 (WI)
Zeit: 60 Minuten, erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den freien Stellen nach den jeweiligen Aufgaben ein (ggf. auf der jeweiligen Rückseite weiterschreiben). Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg !
Diese Klausur besteht einschließlich dieses Deckblattes aus 5 Seiten.

Aufgabe 1.

Gegeben sei folgende aussagenlogische Formel: $\neg(U \vee V) \rightarrow (U \wedge V)$.
Zeigen Sie, daß sich diese Formel in eine äquivalente Formel transformieren lässt, die gleichzeitig eine konjunktive und auch disjunktive Normalform hat. Hinweis: Vereinfachen Sie die Formel durch Transformationsregeln so weit wie möglich.

Aufgabe 2.

Es sei P ein zweistelliges Prädikatensymbol, Q ein einstelliges Prädikatensymbol, a eine Konstante, f ein einstelliges Funktionssymbol, und es seien x, y, z Variablen. Mit diesen syntaktischen Einheiten lassen sich Formeln bilden:

1. $(Q(a) \wedge P(a, x)) \rightarrow (Q(f(a)) \vee P(a, y))$
2. $P(a, y) \wedge \exists z\, P(z, a)$
3. $\forall z\, (Q(z) \vee \neg P(a, z))$

Welche der Formeln ist erfüllbar?

Eine Struktur A, die Symbolen Werte zuordnet, ist gegeben durch $A = (U_A, I_A)$.
Geben Sie eine Struktur an, die Modell für eine der erfüllbaren Formeln der obigen Auflistung ist. Hinweis: Sie müssen hierzu die Menge U_A und die Abbildungsfunktion I_A geeignet festlegen und zeigen, daß für Ihre Wahl von $A = (U_A, I_A)$ sich der Wahrheitswert „true" ergibt.
Bitte geben Sie an, für welche Formel Sie sich bei der Angabe eines Modells entscheiden.

Aufgabe 3.

Es seien Animal, Vegetarian, Sheep, Cow, und Mad_cow einstellige Prädikatensymbole. Die Symbole EATS und PART_OF seien zweistellige Prädikatensymbole, und es seien x, y, z Variablen. Mit diesen syntaktischen Einheiten lässt sich eine Menge von Formeln T bilden:

$$
\begin{aligned}
T := \{\ & \forall x\, (Sheep(x) \rightarrow \\
& \quad (Animal(x) \wedge Vegetarian(x)))\ , \\
& \forall x\, (Cow(x) \rightarrow \\
& \quad (Animal(x) \wedge Vegetarian(x)))\ , \\
& \forall x\, (Vegetarian(x) \rightarrow \\
& \quad (\neg\exists y\, (EATS(x,y) \wedge Animal(y)) \wedge \\
& \quad\ \forall y\, (EATS(x,y) \rightarrow (\neg\exists z\, (PART\_OF(y,z) \wedge Animal(z)))))\ \}
\end{aligned}
$$

Nehmen wir einmal an, diese Formeln stellen die Grundlage für die Spezifikation von Programmen für ein großes Softwareprojekt für das Landwirtschaftsministerium dar. Nehmen wir weiterhin an, vom Leiter des Landwirtschaftsministeriums wird die Aufgabe an Sie als Projektleiter herangetragen, ein Programm zu entwickeln, das ermittelt, welche Kühe (eines bestimmten Bestandes, der als Parameter eingeht) beim Verzehr gesundheitsgefährdend sind. Mit dem Programm soll der Bevölkerung verdeutlicht werden, was alles für die Ernährung getan wird. Bei einem Gespräch wird Ihnen mitgeteilt, wie diese Menge von Kühen definiert ist. Hierzu wird folgende Formel verwendet:

$$
\begin{aligned}
f := \forall x\, (Mad\_cow(x) \rightarrow & \\
& (Cow(x) \wedge \exists y\, (EATS\ (x, y) \wedge \exists z\, PART\_OF(y, z) \wedge Sheep(z))))
\end{aligned}
$$

Es stellt sich also bei der Projektakquisition heraus, daß das Programm die Menge M aller x berechnen soll, für die Mad_cow(x) gilt. Nach eingehender Beratung mit Ihren Mitarbeitern kommen Sie zu dem Schluß, daß Sie das Projekt, obwohl es lukrativ sein mag, aus Gewissensgründen nicht annehmen werden, da die Menge M immer leer sein wird.

Begründen Sie, warum dieses der Fall ist, indem Sie zeigen, daß die Formel $\exists x\ Mad\_cow(x)$ unerfüllbar bzgl. $T \cup \{f\}$ ist. Hinweis: Verwenden Sie folgende Argumentationstechnik: „Nehmen wir an, es gibt ein x, für das Mad_cow(x) gilt. Daraus ergeben sich gewisse Schlussfolgerungen usw. Am Ende ergibt sich aus der Folgerungskette ein Widerspruch. Also muß die Annahme falsch sein.“ Bitte verwenden Sie ggf. die Rückseite dieses Blattes.

Aufgabe 4.

Gegeben Sei die Funktion mod zur Berechnung des Restes der ganzzahligen Division von einer Zahl a durch eine Zahl b.

```
mod(a: N₀, b: N₁): N₀
  if a < b
    then a
    else mod(a-b, b)
  end if
```

1. Um welche Art der Rekursion handelt es sich bei dieser Definition der Funktion mod?

2. Terminiert mod für alle möglichen Eingabewerte? Bitte begründen Sie Ihre Antwort.

3. Wandeln Sie die Funktion in eine Funktion mod_iter um, die zu jeder Eingabe (a, b) die gleichen Werte wie mod berechnet, im Rumpf aber eine Schleife verwendet.