

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

Übungen *Logikprogrammierung* – Blatt 4

Abgabe am Mittwoch, 3. Mai 2006, zu Beginn der Übung.

Aufgabe 1 (5 Punkte)

In dieser Aufgabe sollen quantorfrem Formeln in konjunktive Normalform (KNF) überführt werden. Verwenden Sie hierfür die aus Übung 2 bekannte Funktion \mathcal{D} , die quantorfrem Formeln über $(\{f_0, f_1, \dots\}, \{p_0, p_1, \dots\}, \{X_0, X_1, \dots\})$ wie folgt auf Prolog-Terme abbildet:

$$\begin{aligned} \mathcal{D}(X_i) &= \text{var}(\mathbf{s}^i(0)) \\ \mathcal{D}(f_i(t_1, \dots, t_n)) &= \text{function}(\mathbf{s}^i(0), [\mathcal{D}(t_1), \dots, \mathcal{D}(t_n)]) \\ \mathcal{D}(p_i(t_1, \dots, t_n)) &= \text{predicate}(\mathbf{s}^i(0), [\mathcal{D}(t_1), \dots, \mathcal{D}(t_n)]) \\ \mathcal{D}(\neg\varphi) &= \text{not}(\mathcal{D}(\varphi)) \\ \mathcal{D}(\varphi_1 \circ \varphi_2) &= \mathcal{D}(\cdot)(\mathcal{D}(\varphi_1), \mathcal{D}(\varphi_2)) \text{ mit } \cdot \in \{\wedge, \vee, \rightarrow\} \\ &\text{und } \mathcal{D}(\wedge) = \text{and}, \mathcal{D}(\vee) = \text{or} \text{ und } \mathcal{D}(\rightarrow) = \text{implies} \end{aligned}$$

Hierbei ist $\mathbf{s}^i(0)$ die Kurzschreibweise für $\overbrace{\mathbf{s}(\dots \mathbf{s}(0) \dots)}^{i\text{-mal}}$.

Z.B. wird die quantorfrem Formel $\neg(\neg p_0 \wedge (\neg p_1 \vee p_2))$ durch den folgenden Prolog-Term t dargestellt:

```
not(and(not(predicate(0, [])), or(not(predicate(s(0), [])),
predicate(s(s(0)), []))))
```

Schreiben Sie ein Prolog-Programm, das zu einer beliebigen quantorfrem Formeln eine äquivalente Formel in KNF berechnet. Das 2-stellige Prädikat **knf** soll in seinem zweiten Argument die zum ersten Argument äquivalente Formel in KNF berechnen.

Auf die Anfrage `?- knf(t, Psi)` für den im obigen Beispiel verwendeten Term t soll man die Antwort $\text{Psi} = \text{and}(\text{or}(\text{predicate}(0, []), \text{predicate}(\mathbf{s}(0), [])), \text{or}(\text{predicate}(0, []), \text{not}(\text{predicate}(\mathbf{s}(\mathbf{s}(0)), []))))$, d.h., $(p_0 \vee p_1) \wedge (p_0 \vee \neg p_2)$ erhalten.

Hinweis: Ergänzen Sie das in der Datei `uebung4.pl` bereitgestellte Gerüst. Sie können diese Datei von der Webseite der Vorlesung herunterladen.

Aufgabe 2 (8 Punkte)

In dieser Aufgabe sollen variablenfreie Klauselmengen mit Hilfe von Grundresolution auf Unerfüllbarkeit getestet werden. Eine Klauselmenge ist variablenfrei, wenn alle Literale in allen Klauseln Grundformeln sind. Stellen Sie Klauselmengen als Listen von Listen von Formeln dar. Verwenden Sie hierfür die in Aufgabe 1 beschriebene Darstellung von Formeln als Prolog-Terme.

Z.B. wird die Klauselmenge $[[p_0], [\neg p_0, p_1], [\neg p_1]]$ durch den folgenden Prolog-Term t dargestellt:

```
[[predicate(0, [])], [not(predicate(0, [])), predicate(s(0), [])],  
[not(predicate(s(0), []))]]
```

Schreiben Sie ein Prolog-Programm um variablenfreie Klauselmengen auf Unerfüllbarkeit zu testen. Hierbei sollen aber nur Resolutionsbeweise bis zu einer vorgegebenen Länge betrachtet werden. Das 2-stellige Prädikat `unsat` bekommt als erstes Argument eine Klauselmenge und als zweites die maximale Länge des Resolutionsbeweises in Resolutionsschritten.

Auf die Anfrage `?- unsat(t, s(s(0)))` für den im obigen Beispiel verwendeten Term t soll man die Antwort `Yes` erhalten, auf die Anfrage `?- unsat(t, s(0))` jedoch die Antwort `No`.

Hinweis: Ergänzen Sie das in der Datei `uebung4.pl` bereitgestellte Gerüst. Sie können diese Datei von der Webseite der Vorlesung herunterladen.

Aufgabe 3 (3 Punkte)

Betrachten Sie das folgende Logikprogramm

```
even(0).  
even(s(s(X)) :- even(X).
```

und die Anfrage

```
? - even(s(s(s(s(0))))).
```

Zeigen Sie mit Hilfe des Grundresolutionsalgorithmus, dass die der Anfrage entsprechende Formel φ aus den Formeln φ_1 und φ_2 folgt, die dem Logikprogramm entsprechen.