

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

## Übungen *Logikprogrammierung* – Blatt 2

Abgabe am Mittwoch, 19. April 2006, zu Beginn der Übung.

Eine weitere Erläuterung von Aufgabe 2 findet in der Übung am 12. April statt.

### Aufgabe 1 (1+3 Punkte)

In dieser Aufgabe sollen Sie Algorithmen auf Prolog-Listen schreiben. In Prolog werden Listen als Terme über der Signatur  $\Sigma = \Sigma_2 \cup \Sigma_0$  mit  $\Sigma_2 = \{.\}$  und  $\Sigma_0 = \{[]\}$  dargestellt. Die Liste mit den Elementen  $s(0)$  und  $s(s(0))$  wird durch den Term  $.(s(0), .(s(s(0)), []))$  repräsentiert. Eine alternative Schreibweise für  $.(t_1, t_2)$  ist  $[t_1 | t_2]$ . Die obige Liste kann also auch durch  $[s(0) | [s(s(0)) | []]]$  repräsentiert werden. Eine weitere Darstellungsmöglichkeit ist  $[s(0), s(s(0)) | []]$ , welches man auch als  $[1, 2]$  schreiben kann.

Die Länge einer Liste läßt sich z.B. folgendermaßen bestimmen:

```
length([], 0).  
length([X|XS], s(Y)) :- length(XS, Y).
```

- (a) Schreiben Sie ein 2-stelliges Prädikat `maximum` zum Berechnen des Maximums einer (nicht-leeren) Liste.

Zum Beispiel soll die Anfrage `?- maximum([s(0), 0, s(s(0))], M)` die Antwort `M = s(s(0))` liefern.

- (b) Schreiben Sie ein Prädikat `permutation`, welches alle Permutationen einer Liste berechnet. Die Aussage `permutation(L1, L2)` soll also genau dann wahr sein, wenn `L1` sich nur durch das Vertauschen von Elementen von `L2` unterscheidet.

Zum Beispiel sind `permutation([s(0), s(s(0))], [s(s(0)), s(0)])` und `permutation([s(0), s(s(0))], [s(0), s(s(0))])` wahr.

### Aufgabe 2 (6 Punkte)

In dieser Aufgabe sollen Sie Formeln mit Hilfe von Prolog-Programmen in Pränex-Normalform überführen. Verwenden Sie dafür die folgende Funktion  $\mathcal{D}$ , die Formeln über  $(\{f_0, f_1, \dots\}, \{p_0, p_1, \dots\}, \{X_0, X_1, \dots\})$  wie folgt auf Prolog-Terme abbildet:

$$\begin{aligned}\mathcal{D}(X_i) &= \text{var}(s^i(0)) \\ \mathcal{D}(f_i(t_1, \dots, t_n)) &= \text{function}(s^i(0), [\mathcal{D}(t_1), \dots, \mathcal{D}(t_n)])\end{aligned}$$

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

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

Z.B. wird die Formel  $(\neg\exists X_1 \forall X_0 p_0(f_1(X_0))) \wedge p_1(X_0, X_1)$  durch den folgenden Prolog-Term  $t$  dargestellt:

```
and(not(exists(s(0), forall(0, predicate(0, [function(s(0), [var(0)])])), predicate(s(0), [var(0), var(s(0))]))))
```

Schreiben Sie ein Prolog-Programm mit Klauseln zur Übersetzung von beliebigen Formeln in Pränex-Normalform. Das 2-stellige Prädikat **praenex** soll in seinem zweiten Argument eine zum ersten Argument erfüllbarkeitsäquivalente Formel in Pränex-Normalform berechnen.

Auf die Anfrage `?- praenex(t, Psi)` für den im obigen Beispiel verwendeten Term  $t$  soll man die Antwort `Psi = forall(0, exists(s(0), and(not(predicate(0, [function(s(0), [var(s(0))]))), predicate(s(0), [var(s(s(0))), var(s(s(s(0)))])))))` erhalten, d.h.,  $\forall X_0 \exists X_1 \neg p_0(f_1(X_1)) \wedge p_1(X_2, X_3)$ .

*Hinweis:* Ergänzen Sie das in der Datei `uebung2.pl` bereitgestellte Gerüst. Sie können diese Datei von der Webseite der Vorlesung herunterladen.

### Aufgabe 3 (1+3 Punkte)

In der Vorlesung wurde ein Verfahren zur Überführung von Formeln in Skolem-Normalform vorgestellt. Beweisen Sie die Erfüllbarkeitsäquivalenz der Transformation, indem Sie die beiden folgenden Aussagen beweisen:

- (a) Sei  $\varphi$  eine Formel mit den freien Variablen  $X_1, \dots, X_n$ . Dann ist  $\varphi$  genau dann erfüllbar, wenn

$$\exists X_1, \dots, X_n \varphi$$

erfüllbar ist.

- (b) Sei  $f$  ein neues  $n$ -stelliges Funktionssymbol, das nicht in  $\psi$  vorkommt. Dann ist

$$\forall X_1, \dots, X_n \exists Y \psi$$

genau dann erfüllbar, wenn

$$\forall X_1, \dots, X_n \psi[Y/f(X_1, \dots, X_n)]$$

erfüllbar ist.