INFORMATIK

Universität des Saarlandes
FR Informatik

Harald Ganzinger
Andreas Podelski
Uwe Waldmann

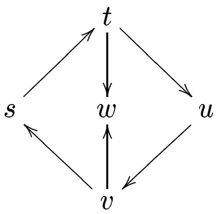
6. Juni 2001

Übungen zur Vorlesung „Programmiersprachen“
Übungsblatt 9

Programmieraufgabe 9.1: (*5 P.*)
Ein gerichteter Graph kann durch ein Prolog-Prädikat `e(V1,V2)` modelliert werden, das gerade dann erfüllt ist, wenn in dem Graphen eine Kante vom Knoten `V1` zum Knoten `V2` existiert; die Knoten sind dabei beliebige Prolog-Grundterme.

Beispiel:

```
e(s,t).
e(t,u).
e(u,v).
e(v,s).
e(t,w).
e(v,w).
```



Implementieren Sie ein Prädikat `reach(V1,V2,Path)`, das dann erfüllt ist, wenn ein zyklenfreier Pfad vom Knoten `V1` zum Knoten `V2` existiert und `Path` die Liste aller Knoten auf diesem Pfad ist (beginnend mit `V1` und endend mit `V2`). Pfade mit Kantenlänge 0 (d. h., Anfangsknoten = Endknoten) sind zulässig. Schreiben Sie das Programm so, daß es nacheinander *alle* möglichen Lösungen der Anfrage liefert, ganz gleich, wie weit die Anfrage instantiiert ist, und zwar auch dann, wenn der durch `e` gegebene Graph *nicht* zyklenfrei ist.

Programmieraufgabe 9.2: (*7 P.*)
Erweitern Sie Aufgabe 8.3 wie folgt um Elementklassen (wieder analog zu Unix-Shells):

- jede Liste $[x]$, wobei $x \notin \{*, ?, \text{'[' }\}$, matcht sich selbst.
- die Liste $[*]$ matcht jede (möglicherweise leere) Liste.
- die Liste $[?]$ matcht jede einelementige Liste.
- die Liste $[\text{'['}, x_1, \ldots, x_n, \text{']'}]$, wobei kein $x_i = \text{']'}$, matcht jede der Listen $[x_1]$, ..., $[x_n]$.
- falls l_1 l'_1 und l_2 l'_2 matcht, dann matcht die Konkatenation von l_1 und l_2 die Konkatenation von l'_1 und l'_2.

Beispiel: Die Liste `[a, '[', b, c, ']', d, *]` matcht die Liste `[a, c, d, e, f]`.

Implementieren Sie in Prolog ein Prädikat `matches(XL,YL)`, das genau dann erfüllt ist, wenn die Liste `XL` die Liste `YL` matcht.

Hinweis: Zeichenfolgen, die Sonderzeichen wie `[` enthalten, können in Prolog als Atome benutzt werden, sofern man sie in einfache Anführungszeichen setzt.

Aufgabe 9.3: (*4 P.*)
Unterscheiden sich die folgenden Prolog-Prädikate `a1`, ..., `a6` in ihrem Verhalten? Wenn ja, wie?

```
a1([],0).
a1([_|_],1).

a2([],0) :- !.
a2([_|_],1).

a3([],0).
a3(L,1) :- not(L = []).
```

```
a4([],0) :- !.
a4(_,1).

a5([],Y) :- Y = 0, !.
a5(_,1).

a6([],Y) :- !, Y = 0.
a6(_,1).
```

Aufgabe 9.4: (*4 P.*)
Geben Sie Implementierungen der Prädikate `r`, `s` und `t` an, so daß sich alle vier Prädikate `p1`, `p2`, `p3` und `p4` in ihrem Verhalten unterscheiden.

```
p1(X,1,Y) :- !, r(X,Z), s(Z,Y).
p1(X,_,Y) :- t(X,Y).

p2(X,1,Y) :- r(X,Z), !, s(Z,Y).
p2(X,_,Y) :- t(X,Y).

p3(X,1,Y) :- r(X,Z), s(Z,Y), !.
p3(X,_,Y) :- t(X,Y).

p4(X,1,Y) :- !, r(X,Z), s(Z,Y), !.
p4(X,_,Y) :- t(X,Y).
```

Schicken Sie die ausgetesteten Lösungen der Programmieraufgaben (inklusive einiger Testbeispiele mit Ergebnissen) bis zum 13. 6., 13:00 Uhr per Mail an Ihren Übungsgruppenleiter, Subject: `PS Blatt 9`. Bereiten Sie die übrigen Aufgaben so vor, daß Sie die Lösung in der nächsten darauffolgenden Übungsstunde vortragen können.