Einführung in das Programmieren Prolog
SS2006
Dr. Gunter Grieser

Übungsblatt zu Teil 2
(Einführung und Grundkonzepte)

Version 1.1

Aufgabe 2.1 (Schwierigkeitsgrad 1)

a. Erstellen Sie ein Prolog-Programm, das die nachfolgende umgangssprachliche Beschreibung formalisiert.

   Peter liebt Susi.
   Hans liebt Susi und Sabine.
   Sabine liebt Peter und haßt Hans.
   Susi liebt Peter und Felix.
   Susi haßt Sabine.
   Felix liebt sich selbst.

b. Stellen Sie die nachfolgenden Anfragen und vollziehen Sie den Prozeß der Lösungsfindung nach.

   Liebt Peter Susi?
   Liebt Susi Felix?
   Wen liebt Sabine?
   Wer liebt Sabine?
   Wer liebt jemanden, der ihn auch liebt?
   Wessen Liebe wird mit Haß vergolten?

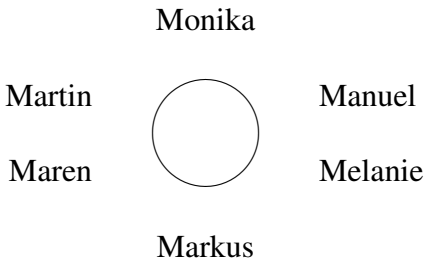
c. Tracen Sie die Anfragen aus b.

Aufgabe 2.2 (Schwierigkeitsgrad 1)

Überlegen Sie sich verschiedene Terme und unifizieren sie diese. Überprüfen Sie die Ergebnisse in Prolog.

Aufgabe 2.3 (Schwierigkeitsgrad 2)

An einem runden Tisch sitzen sechs Personen in der folgenden Anordnung.



a. Erstellen Sie ein Prolog-Programm unter Verwendung des Prädikats "sitzt direkt rechts neben", das die oben dargestellte Situation beschreibt.

b. Stellen Sie die nachfolgenden Anfragen und vollziehen Sie den Prozeß der Lösungsfindung nach.

   Wer sitzt rechts neben Melanie?

   Von wem ist Maren der linke Nachbar?

   Wer sind die Nachbarn von Monika?

   Wer sitzt Melanie gegenüber?

   Wer sitzt wem gegenüber?

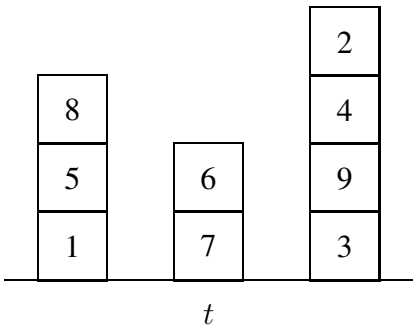
c. Erweitern Sie das in Teilaufgabe (a) erstellte Programm um Regeln für "links neben", "Nachbarn von" und "gegenüber".

d. Erweitern Sie (falls noch nicht geschehen) Ihre Programme aus den vorigen Teilaufgaben so, daß der Tisch beliebige Größe haben kann.

Aufgabe 2.4 (Schwierigkeitsgrad 2)

Auf einem Tisch t sind mit Nummern gekennzeichnete Blöcke gestapelt.

a. Erstellen Sie ein Prolog-Programm, in dem die dargestellte Situtation mit Hilfe eines Prädikats "auf" repräsentiert wird. Dabei bedeute "auf(X,Y)", daß Block X auf Y steht.



b. Definieren Sie ein Prädikat “über”, welches ausdrückt, daß ein Block über einem anderen Block bzw. über dem Tisch steht.

c. Stellen Sie die nachfolgenden Anfragen und vollziehen Sie den Prozeß der Lösungsfindung nach.

   Steht Block 8 über Block 1?

   Steht Block 2 über Block 7?

   Was steht über Block 3?

   Was steht unter Block 4?

Aufgabe 2.5 (Schwierigkeitsgrad 2)

Definieren Sie ein Prädikat “anzahl_blaetter(Term, N)”, das die Anzahl der Atome (Blätter) im Term zählt.

Hinweis: nehmen sie zunächst der Einfachheit halber an, daß alle Funktoren im Term 2stelig (oder 0stellig) sind.

Hinweis: Mittels `succ` können Sie den Nachfolger einer Zahl bestimmen.

Aufgabe 2.6 (Schwierigkeitsgrad 2)

Definieren Sie ein Prädikat “drucke_funktoren(Term)”, das alle im Term vorkommenden Funktoren (auch die nullstelligen!) ausdruckt.

Hinweis: nehmen sie zunächst der Einfachheit halber an, daß alle Funktoren im Term 2stelig (oder 0stellig) sind.

Aufgabe 2.7 (Schwierigkeitsgrad 2)

Definieren Sie ein Prädikat ”grund(Term)”, das testet, ob Term keine Variablen enthält, dh. ob Term ein Grund-Term ist.

Hinweis: nehmen sie zunächst der Einfachheit halber an, daß alle Funktoren im Term 2stelig (oder 0stellig) sind.

Aufgabe 2.8 (Schwierigkeitsgrad 2)

Definieren Sie ein Prädikat ”term_member(Term1, Term2)”, das testet, ob Term1 als Teilterm in Term2 vorkommt.

Hinweis: nehmen sie zunächst der Einfachheit halber an, daß alle Funktoren im Term 2stelig (oder 0stellig) sind.

Aufgabe 2.9 (Schwierigkeitsgrad 2)

Definieren Sie ein Prädikat “subterm_links(Term1, Term2)”, das aus Term1 alle Teilterme bis auf die am weitesten links stehenden löscht. d.h. aus $f(g(b,b),g(b,c),c)$ wird $f(g(b))$.

Kriegen Sie das auch mit den am weitesten rechts stehenden hin?

Aufgabe 2.10 (Schwierigkeitsgrad 2)

Betrachten Sie Ihr Programm aus Aufgabe 2.1.

Tracen Sie verschiedene Anfragen. Vorher überlegen Sie bitte auf dem Papier, wie die Lösung gefunden wird.

Aufgabe 2.11 (Schwierigkeitsgrad 3)

Definieren Sie ein Prädikat “spiegele(Term)”, das die Reihenfolge der Argumente im gesamten Term umdreht, d.h. aus $f(a,g(b,c),c)$ wird $f(c,g(c,b),a)$.

Hinweis: Eine Liste wird mit `reverse` umgedreht.

Aufgabe 2.12 (Schwierigkeitsgrad 3)

Schreiben Sie ein Programm, das Terme in ihrer Struktur als Bäume ausdruckt. D.h., $f(g(b,c),d)$ wird z.B. wie folgt ausgedruckt (Sie können hier natürlich noch viel schönere Ausgaben produzieren)

```
f
  g
    b
    c
  d
```

Hinweis: nehmen sie zunächst der Einfachheit halber an, daß alle Funktoren im Term 2stellig (oder 0stellig) sind.

Aufgabe 2.13 (Schwierigkeitsgrad 3)

Erzeugen Sie einen möglichst großen Term. Wie groß schaffen Sie es, bis der Speicher voll ist?

Aufgabe 2.14 (Schwierigkeitsgrad 3)

Wieviel Prozent ist Unifikation mit Occurs-Check langsamer als ohne? Erzeugen Sie dazu verschiedene große Terme und stoppen sie die Zeit.