Universität Bonn
Institut für Informatik III
Prof. Dr. R. Manthey
Dr. A. Behrend
Dipl.-Inform. C. Dorau

23.5.2008

Übungen
Deskriptive Programmierung
SS 2008

Blatt 3

Aufgabe 1 (Prolog-Tutorial)**.**

1. Nehmen Sie an, es existieren Fakten der Form `mother(X,Y)` und `father(X,Y)` mit der Bedeutung, dass X jeweils das entsprechende Elternteil von Y repräsentiert. Definieren Sie rekursiv die Relation `ancestor(X,Y)`, wobei X ein Vorfahre von Y ist. Definieren Sie dann die Relation `related(X,Y)` zur Modellierung allgemeiner Verwandtschaftsverhältnisse (also X und Y sind verwandt miteinander).

2. Definieren Sie die Relation `mod(X,Y,Result)`, die für die gegebenen positiven Integer-Werte X und Y als Ergebnis die Variable Result mit dem Wert von X mod Y instantiiert.

3. Überlegen Sie sich die Relationen `maximum(List,Max)` und `minimum(List,Min)` zur Bestimmung des jeweils größten und kleinsten Elements einer Liste.

4. Überlegen Sie sich zwei unterschiedliche Definitionen für die Relation `last(List,Last)` zum Finden des letzten Elements einer Liste `List` jeweils einmal mit und einmal ohne der Verwendung von `append/3`. Welche der beiden ist effizienter auswertbar und warum?

5. Definieren Sie eine Relation `delete(X,List1,List2)`, wobei List2 der Liste List1 entspricht aber das erste Auftreten von X entsprechend gelöscht wurde.

6. Nehmen Sie an, die interne Prolog-Datenbank enthält folgendes Fakt:

   ```
   days([sun,mon,tue,wed,thu,fri,sat]).
   ```

   Überlegen Sie sich eine Relation `week_beginning(Day,Week)`, welche gegeben einen Tag Day die Wochentagsliste beginnend mit diesem Tag Day in Week ausgibt.

7. Definieren Sie eine Relation `intersect(List1,List2,Res)`, die die Schnittmenge zweier Mengen gegeben als Listen ermittelt.

8. Sie haben in der ersten Übung den Auswertungsmechanismus von Prolog kennengelernt. Überlegen Sie sich ein Beipsiel, wo die Abarbeitungsreihenfolge von oben-nach-unten für das Durchsuchen der internen Datenbank und von links- nach-rechts für das Auswerten von Bedingungsteilen zu unendlichen Zyklen führen kann.

Aufgabe 2 (Haskell-Tutorial)**.** Sie haben in der Vorlesung kennengelernt, wie man potentiell unendliche Listen definiert. Diese Listen machen nur Sinn, wenn man jedes Element, das sie enthalten, nach endlicher Zeit auch in der Liste finden kann. Ein Element, das nie gefunden wird, braucht erst gar nicht in die Liste aufgenommen zu werden. Würde man beispielsweise die ganzen Zahlen in der Form $[0, 1, 2.., -1, -2..]$ angeben, so würde man beim Durchsehen der Liste "von links nach rechts" (und Haskell macht das so), die negativen Zahlen nie erreichen. Geben Sie Aufzählungsvarianten für die folgenden Mengen an, bei denen jedes Element nach endlicher Zeit gefunden wird:

a) Die Menge der ganzen Zahlen

b) Die Menge aller 2-Tupel von natürlichen Zahlen

c) Die Menge aller 3-Tupel von natürlichen Zahlen

d) Die Menge aller Tupel von natürlichen Zahlen

Können Sie dafür Ausdrücke in Haskell angeben?

Aufgabe 3 (Haskell-Tutorial*)**.** Stellen Sie sich vor, es stünden genau k verschiedene Münzarten mit den Beträgen $b_1, b_2, ..., b_k$ zur Verfügung mit $0 < b_1 < b_2 < ... < b_k$. Nun interessiert die Frage, auf wie viele verschiedene Arten ein Betrag a gewechselt werden kann:

1. Schreiben Sie eine Funktion `change (a)`, die für einen Geldbetrag `a` unter Verwendung von 5 Münzen zu 1, 2, 5, 10 und 50 Cent (also $b_1 = 1$, $b_2 = 2$, $b_3 = 5$, $b_4 = 10$ und $b_5 = 50$) berechnet, wie viele verschiedene Möglichkeiten es zum Wechseln gibt.