Einführung in das Programmieren Prolog
SS2006
Dr. Gunter Grieser

Übungsblatt zu Teil 8
(Fortgeschrittene Techniken)

Version 1.0

Aufgabe 8.1 (Schwierigkeitsgrad 2)

Verändern Sie den Metainterpreter:

- er soll auch oder behandeln können
- die Reihenfolge der Abarbeitung der Literale im Body soll von rechts nach links erfolgen
- die Abarbeitung der Literale im Body soll zufällig erfolgen

Für Fortgeschrittene:

- Die Abarbeitungsreihenfolge der *Regeln* soll von unten nach oben sein
- Die Abarbeitungsreihenfolge der *Regeln* soll zufällig sein
- Doppelte Berechnungen sollen vermieden werden
- Implementieren Sie auch $->$ (Achtung, auf korrekte Klammerung achten)

Aufgabe 8.2 (Schwierigkeitsgrad 3)

Eine Warteschlange ist eine Datenstruktur, bei der Elemente hinten angefügt und vorne weggenommen werden können. Wir wollen Warteschlangen mit Hilfe von Differenzlisten repräsentieren. Dabei wird die leere Warteschlange durch die Differenzliste Ws-Ws und eine beliebige Warteschlange durch die Differenzliste WKopf-WRest repräsentiert.

a. Definieren Sie ein Prädikat “anstellen(E,WSalt,WSneu)”, so daß WSneu die Warteschlange ist, die man aus WSalt durch Anhängen des Elementes E erhält.

b. Definieren Sie ein Prädikat “entfernen(E,WSalt,WSneu)”, so daß WSneu die Warteschlange ist, die man aus WSalt durch Entfernen des ersten Elementes E erhält, wenn WSalt nicht leer ist.

Aufgabe 8.3 (Schwierigkeitsgrad 3)

Definieren Sie ein Prädikat ”umdrehen_dl(Liste,UmgedrehteListe)”, das die Reihenfolge der Elemente einer Liste umdreht. Verwenden Sie zur Lösung des Problems Differenzlisten wobei explizite Aufrufe von *app* (*append*) vermieden werden sollen.

Aufgabe 8.4 (Schwierigkeitsgrad 3)

Aufgabe 3.8: Definieren Sie ein Prädikat ”sortiere(Liste,SortierteListe)”, das eine Liste von Elementen ”element(Name,Farbe)”, die *rot*, *weiß* und *blau* gefärbt sind so umordnet, daß alle roten Elemente zuerst, dann alle weißen, und schließlich alle blauen Elemente kommen. Diese Umordnung soll die ursprüngliche Reihenfolge der Elemente gleicher Farbe bewahren. Verwenden Sie zur Lösung des Problems Differenzlisten und vermeiden Sie explizite Aufrufe von *app* (*append*).

Aufgabe 8.5 (Schwierigkeitsgrad 3)

Erweitern Sie Ihr Programm aus 6.2, so daß die eingelesenen Prädikate so verändert werden, daß iterierte (oder zunächst beschränkte) Tiefensuche implementiert wird.

Aufgabe 8.6 (Schwierigkeitsgrad 3)

Erweitern Sie Ihr Programm aus 6.2, so daß die eingelesenen Prädikate so verändert werden, daß sie ein einem zusätzlichen Argument den Suchbaum mitprotokollieren.

Aufgabe 8.7 (Schwierigkeitsgrad 3)

Repräsentieren Sie einen zyklischen Graph mittels selbstbezüglicher Strukturen.

Schreiben Sie Prädikat, um sich in dem Graph entlang der Kanten zu bewegen, d.h. zum nächsten Knoten zu gelangen.

Schreiben Sie Prädikate, um einen Knoten hinzuzufügen oder zu löschen.

Aufgabe 8.8 (Schwierigkeitsgrad 3)

Repräsentieren Sie einen gerichteten Graphen, indem Sie sich auch noch Verknüpfungen zu den jeweiligen Vorgängern (d.h. entgegen der Kantenrichtung) speichern, um sich so auch effizient rückwärts durch den Graph bewegen zu können.

Schreiben Sie Prädikate zum Vor- und Rückwärtsbewegen, Einfügen und Löschen.