Logik für Informatiker (Diplom)

Prof. Dr. B. Nebel, Prof. Dr. W. Burgard — Universität Freiburg
Wintersemester 2007/2008 — Institut für Informatik

Übungsblatt 12

Abgabe: Dienstag, 29. Januar 2008

Aufgabe 12.1 (SLD-Bäume und Beweisbäume)

Betrachten Sie das folgende logische Programm:

```
edge(a,b).      target(d).
edge(a,c).      target(f).
edge(b,d).      reachable(a).
edge(c,d).      reachable(X) :- edge(Z,X), reachable(Z).
edge(e,f).      good(X) :- target(X), reachable(X).
edge(f,e).
```

Geben Sie den SLD-Baum zu diesem Programm und der Anfrage `?- good(X).` an. Vergleichen Sie Ihre Lösung mit der Ausgabe von `?- trace, good(X).` Geben Sie ferner einen der Beweisbäume für die Anfrage an.

Aufgabe 12.2 (Handlungsplanung)

Ein Roboter befindet sich im linken von zwei benachbarten Räumen. In demselben Raum befindet sich ein Paket, das der Roboter in den rechten Raum transportieren soll. Die Tür zwischen den beiden Räumen ist geschlossen, kann jedoch vom Roboter geöffnet werden, wenn er seine Greifarme frei hat, also gerade kein Paket trägt. Dem Roboter stehen die Aktionen `go_right`, `open_door`, `pick_up` und `drop` zur Verfügung, mit denen der den rechten Raum betreten kann, falls er sich im linken befindet und die Tür geöffnet ist; die Tür öffnen kann, wenn er sich im linken Raum befindet; das Paket aufheben kann, wenn er sich im gleichen Raum wie das Paket befindet; und das Paket im aktuellen Raum ablegen kann. Ein Zustand wird durch `state(rob_pos,pack_pos,door_status)` beschrieben, wobei für `rob_pos` die Werte `left` und `right`, für `pack_pos` die Werte `left`, `right` und `robot` sowie für `door_status` die Werte `open` und `closed` möglich sind. Kodieren Sie die vier Aktionen in Prolog und, analog zu `canget`, ein Prädikat `can_deliver/2`, das genau dann wahr ist, wenn von dem gegebenen Zustand ein Zielzustand erreichbar ist, in dem sich das Paket im rechten Raum befindet.

Verwenden Sie eine solche Kodierung von `can_deliver`, dass die Anfrage `?- can_deliver( state(left,left,closed), Plan ).` die Variable `Plan` an einen Plan bindet, der das Problem löst. Welche Ausgabe erhalten Sie?

Aufgabe 12.3 (Akkumulatoren)

Ein ganzzahlig markierter Binärbaum ist entweder ein Blatt mit einem ganzzahligen Wert oder eine Verzweigung mit einem ganzzahligen Wert, die zu einem linken und einem rechten Teilbaum führt (wobei die Teilbäume wieder ganzzahlig markierte Binärbäume sind):

```
bintree(leaf(X)) :-
    integer(X).
bintree(branch(Left,X,Right)) :-
    integer(X),
    bintree(Left),
    bintree(Right).
```

Eine Postorder-Traversierung eines Binärbaums kann deklarativ wie folgt in Prolog implementiert werden:

```
postorder_naive(leaf(X),[X]).
postorder_naive(branch(Left,X,Right),L) :-
    postorder_naive(Left,L1),
    postorder_naive(Right,L2),
    append(L1,L2,L3),
    append(L3,[X],L).
```

Finden Sie eine effizientere Prolog-Implementierung der Postorder-Traversierung, die Akkumulatoren verwendet, und vergleichen Sie die Effizienz der beiden Implementierungen durch Aufruf von

```
?- genbt(15,T), time(postorder_METHOD(T,L))
```

für `METHOD = naive` und `METHOD = accu`. Das Hilfsprädikat `genbt/2` ist in der Datei `blatt12.pl` definiert.

Das Programm aus Aufgabe 12.1, die naive Postorder-Traversierung und das Hilfsprädikat `genbt/2` finden Sie in der Datei `blatt12.pl` auf der Vorlesungswebsite. Die Abgabe der Programme sollte bis Dienstag, 29. Januar 2008, 16 Uhr, per Mail an `mattmuel@informatik.uni-freiburg.de` erfolgen.

Die Übungsblätter dürfen und sollten in Gruppen von zwei Studenten bearbeitet werden. Bitte schreiben Sie beide Namen auf Ihre Lösung.