Universität Dortmund
Fachbereich Informatik
P. Fischer, R. Lübeck, A. Mester

Dienstag, 2.5.1995
Abgabe bis Montag, 8.5.1995 (12.00 Uhr)
Briefkasten 17 bzw. 18 in Pavillon 6

Übungen zur Stammvorlesung
Künstliche Intelligenz
Blatt 1

Aufgabe 1.1 (Kernaufgabe):

a) Gegeben sei ein mehrfachverzweigter Baum. Ein Beispiel findet sich in `~ki000/U1/ublevel.pl`:

```
nachf(a,b).  nachf(b,e).  nachf(d,h).  nachf(e,k).
nachf(a,c).  nachf(c,f).  nachf(e,i).  nachf(f,l).
nachf(b,d).  nachf(c,g).  nachf(e,j).
```

Wobei `nachf(a,b)` bedeutet: Nachfolger von a ist b.

Entwerfen Sie ein Prolog-Programm namens `blevelprint/1`, daß, aufgerufen mit der Wurzel des Baumes als einzigem Parameter (im Beispiel also: `blevelprint(a)`), die Ebenen des Baumes zeilenweise ausgibt. D.h. auf die Frage `?- blevelprint(a).` an das Prolog-System, sollte dieses mit

```
[a]
[b,c]
[f,g,d,e]
```
etc.
`yes` antworten.

Führen Sie die obige Anfrage durch. Welche Ausgabe erhalten Sie?

b) Den Baum aus Teilaufgabe a) erweitern wir durch zwei gerichtete Kanten zu einem gerichteten Wurzelgraphen (vgl. `~ki000/U1/uglevel.pl`): `nachf(d,c).` `nachf(l,a).`

Erweitern/ändern Sie Ihre Lösung zu Teilaufgabe a), sodaß nun die Ebenen des gerichteten Graphen auf die Frage `?- glevelprint(a).` hin ausgegeben werden.

Tips: Mit `write(X)` können Sie die aktuelle Instanziierung der Variablen X ausgeben; `nl` erzeugt in der Ausgabe eine neue Zeile; `append(H,V,E)` hängt die Liste H hinter die Liste V und liefert das Ergebnis E; `findall(T, Goal, L)` erzeugt eine Liste L der Terme T, für die Goal erfüllbar ist.

Aufgabe 1.2:

Gegeben sei der Stammbaum aus Aufgabe 0.1 (vgl. `~ki000/U0/stamm.pl`).
Entwerfen Sie möglichst unter Verwendung von Lösungsteilen aus Aufgabe 1.1 ein Prolog-Programm `nachfahren/0`, daß für jeden „Waisen“ (also jede Stammutter und jeden Stammvater) jeweils seinen Nachfahren-Stammbaum ausgibt (für jede Generation eine Zeile). Wenden Sie `?- nachfahren.` an. Wer sind die „Waisen“?
Hinweis: Sollten Sie die Aufgabe 1.1 nicht gelöst haben, so gehen Sie davon aus, daß Ihnen ein entsprechendes Prolog-Programm zur Verfügung steht (z.B. `glevelprint(Waise)`), daß die nötige Ausgabe liefert.
Tip: Neben der cut-Regel gibt es in Prolog auch noch den „local cut“.

Aufgabe 1.3:

Gegeben sei das „Missionare-Kannibalen-Problem“ aus der Vorlesung. Überprüfen Sie den im Skript auf Seite 21 angegebenen Zustandsgraphen. Erstellen Sie zu den beiden bereits angegebenen Operatoren die noch fehlenden acht und führen Sie eine Tiefensuche mit Ihren Operatoren durch, die als Ergebnis einen Pfad vom Start- zum Zielzustand liefert. Was ergibt Ihre Überprüfung?