Constraint Programmierung
Prof. Dr. Thom Frühwirth, Marc Meister

WS4 – Blatt 2
Besprechung am 9. Nov., 14 Uhr, O27/2201
Vorlesungsseite `http://www.informatik.uni-ulm.de/pm/veranstaltungen/ws2004/cp/`

Aufgabe 5. Consider the formula

$$\big(\forall X p(X,X)\big) \wedge \big(\forall X \forall Y \forall Z (p(X,Y) \wedge p(Y,Z)) \rightarrow p(X,Z)\big) \wedge \big(\forall X \forall Y p(X,Y) \vee p(Y,X)\big) \rightarrow \exists Y \forall X p(Y,X)$$

a) Show that every interpretation with a finite domain is a model.
b) Find an interpretation which is not a model.

Aufgabe 6.

a) Schreiben Sie ein Prolog-Prädikat `summe(X,Y,Z)`, das genau dann zutrifft, wenn zwei ganze Zahlen `X` und `Y` zwischen `1` und `10` die ganze Zahl `Z` als Summe haben.
Beispiel: `summe(5,6,11)` trifft zu.
b) Erweitern Sie, so daß bei gegeben ganzen Zahlen `X` und `Y` die Summe `Z` berechnet werden kann. Beispiel: `summe(5,6,Z)` berechnet `Z`.
c) Welche Antwort liefert das Ziel `summe(X,Y,10)`?

Aufgabe 7.
Schreiben Sie ein Prolog-Programm, das Ratschläge für die Planung von Speditionsaufträgen gibt. Das Programm benutzte eine Datenbasis mit Informationen über Entfernungen und Städte:

```
Hamburg,Bremen,80          Hamburg,Hannover,110
Hamburg,Berlin,230         Bremen,Hannover,100
Bremen,Dortmund,200        Hannover,Kassel,140
Hannover,Nuernberg,380     Dortmund,Kassel,130
Dortmund,Koeln,80          Kassel,Wuerzburg,180
Kassel,Frankfurt,180       Frankfurt,Wuerzburg,110
Nuernberg,Muenchen,160
```

a) Repräsentieren Sie diese Fakten in der Form `dist(city0,city1,km)`; testen Sie Ihre Datenbank mit Anfragen wie `dist(dortmund,bremen,Km)`, `dist(From,To,180)` und `dist(From,To,Km)`.
Tipp: Achten Sie auf Groß-/Kleinschreibung und auf Punkt am Ende.
b) Schreiben Sie ein Prolog-Prädikat `route(From,To)`, das genau dann zutrifft, wenn eine Verbindung zwischen `From` und `To` besteht.
Testen Sie mit `route(muenchen,berlin)`, `route(ulm,halle)` und schließlich mit `route(frankfurt,To)`.
c) Erweitern Sie `route/2` um ein Argument zum Prädikat `route(From,To,Route)`, das bzgl. `From` und `To` die gleiche Funktionalität hat wie `route/2`, gleichzeitig jedoch eine Liste `Route` der durchfahrenen Orte mitführt und so Zyklen vermeidet. Diesen Sachverhalt können Sie unter Verwendung von Negation durch Scheitern ausdrücken.
Testen Sie ausgiebig, mit und ohne Variablen, z.B. auch `route(dortmund,X,[kassel])`.
d) Schreiben Sie ein Proglog-Prädikat `shortestpath(From,To,Route,Km)` das genau dann gilt, wenn `From` und `To` auf der Route `Route` die kürzeste Entfernung `Km` haben.
Hinweis: Die kürzeste Entfernung ist gefunden, wenn es keine andere Route gibt mit einer noch kürzeren Entfernung.
Testen Sie ausgiebig, z.B. auch `shortestpath(kassel,bremen,[hannover],X)`.