

Softwaretechnik und Programmiersprachen I - Sommersemester 2011

Übungsblatt 7

Besprechung der Aufgaben in 25.12.02.55 am Mittwoch 14:30-16:00 Uhr oder Freitag 10:30-12:00 Uhr
Bei Fragen wenden Sie sich bitte an John Witulski: John.Witulski@uni-duesseldorf.de.

Aufgabe 7.1 (CNF für FOL)

Wandeln Sie die folgenden FirstOrderLogic-Terme in konjunktive Normalform um:

- $\forall \text{Mensch} \exists \text{Planet} \text{lebtauf}(\text{Mensch}, \text{Planet})$
- $\forall A (\exists B f(A,B) \vee \neg g(A)) \wedge (\neg \exists Z f(A,Z) \vee \neg \forall B f(A,B))$
- $\exists \text{Hund} \exists \text{Katze} \text{frisst}(\text{Hund}, \text{Katze}) \wedge \exists \text{Katze} \exists \text{Maus} \text{frisst}(\text{Katze}, \text{Maus}) \rightarrow \exists \text{Hund} \exists \text{Maus} \text{frisst}(\text{Hund}, \text{Maus})$

Aufgabe 7.2 (Intuition)

Versuchen Sie herauszufinden, was das folgende Prädikat macht:

```
s([], _, []).  
s([a(H,_) | T], a(H,X), [a(H,X) | T]).  
s([a(H,V) | T], a(I,L), [a(H,V) | R]) :-  
    H \= I,  
    s(T, a(I,L), R).
```

Betrachten Sie dabei folgenden Beispielaufruf:

```
?- s(T, a(i,7), L).
```

Aufgabe 7.3 (Listen 2)

- Schreiben Sie ein Programm `perm/2`, das alle Permutationen einer Liste erstellt.

```
?-perm([1, 2, 3], X).  
X=[1, 2, 3];  
X=[1, 3, 2];  
X=[2, 1, 3];  
X=[2, 3, 1];  
X=[3, 1, 2];  
X=[3, 2, 1]
```

- Schreiben Sie ein Programm, das den größten gemeinsamen Teiler identifiziert.

```
?-ggT(6, 9, X).  
X=3
```

Aufgabe 7.4 (Automaten)

Überlegen Sie sich, wie man einen Automaten wie in Abbildung 1 als Faktenbasis speichert. Schreiben Sie dazu ein Prädikat `accept/1`, das eine Liste übergeben bekommt und genau dann wahr ist, wenn der Automat das übergebene Wort akzeptiert. Zum Beispiel:

```
?-accept([d,a,b,a,b,b,b,c,d,c]).
```

yes

```
?-accept([d,a,b,a,e,d,c]).
```

no

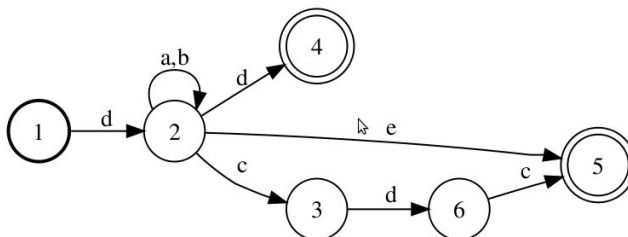


Abbildung 1: Automat M

Aufgabe 7.5 (Schafe 2)

Eine neue Anforderung hat eine Umstrukturierung der Schafdatenbank aus Aufgabe 1.4 zur Folge: Alle Schafe werden nun mittels Fakten `sheep/3` gespeichert, wobei das erste Argument den Namen des Schafs, das zweite das Geschlecht (female/male) und das dritte eine Liste von beliebigen Besonderheiten angibt. Das Prädikat `parent/2` bleibt von der Änderung unberührt. Die Beispieldatensätze aus Aufgabe 1.4 sehen jetzt wie folgt aus:

```
sheep(dolly,female,[]).
```

```
sheep(haba,female,[blue,mobile]).
```

```
sheep(doerte,female,[scrapy]).
```

```
sheep(friedrich,male,[]).
```

```
sheep(gunter,male,[black]).
```

- Passen Sie die Prädikate `sheep/1`, `mother/2`, `father/2` und `ancestor/2` so an, dass sie mit der Änderung funktionieren.
- Implementieren Sie ein Prädikat `related/2`, das angibt, ob zwei Schafe miteinander verwandt sind.
- Implementieren Sie ein Prädikat `search/3`, das ausgehend von einem Schaf ein verwandtes Schaf sucht, das eine gegebene Eigenschaft hat. Z.B. soll die Anfrage `search(haba,black,S)` ein schwarzes Schaf in der Familie von Schaf haba suchen
- Schreiben Sie ein Prädikat `relation/3`, das zu zwei Schafen die Art der Verwandtschaft ausgibt. Zum Beispiel ergibt folgende Antwort

```
?- relation(ernst, lena, X).  
X = [son,daughter,father]
```

dass ernst Sohn der Tochter des Vaters von lena ist. Als direkte Verwandtschaftsverhältnisse sollen ausschließlich mother, father, daughter und son dienen.