Michael Arndt
Fritz Hamm
Peter Schroeder-Heister

Sommersemester 2003
Übungsblatt 4
28.5.2003

Übungen zur Vorlesung *Logikprogrammierung*

Pure PROLOG

Syntax von PROLOG:

PROGRAM	→	CLAUSE PROGRAM
PROGRAM	→	ε
CLAUSE	→	ATOM.
CLAUSE	→	ATOM :- ATOMLIST
ATOMLIST	→	ATOM.
ATOMLIST	→	ATOM, ATOMLIST
GOAL	→	?- ATOMLIST
ATOM	→	RELSYMB
ATOM	→	RELSYMB (TERMLIST)
TERM	→	VARSYMB
TERM	→	CONSTSYMB
TERM	→	FUNSYMB (TERMLIST)
TERMLIST	→	TERM
TERMLIST	→	TERM, TERMLIST
RELSYMB	→	[a–z] ([a–zA–Z0–9_])*
VARSYMB	→	[A–Z] ([a–zA–Z0–9_])*
CONSTSYMB	→	[0–9]* | [a–z] ([a–zA–Z0–9_])*
FUNSYMB	→	[a–z] ([a–zA–Z0–9_])*

Man erhält volles PROLOG aus DATALOG, indem man den Begriff des Terms auf folgende Weise erweitert:

(1) Ein Variablensymbol ist ein Term.

(2) Ein Konstantensymbol ist ein Term.

(3) Falls f ein Funktionssymbol ist und $t_1, \ldots, t_n$ Terme sind, so ist $f(t_1, \ldots, t_n)$ ein Term.

PROLOG-Programme haben dieselbe Form wie DATALOG-Programme, d.h. ein Programm besteht aus einer endlichen Folge von Klauseln.

Beispiel eines PROLOG-Programms:

```
nat(0).
nat(s(X)) :- nat(X).
```

Die Bedeutung dieses Programms ist

$$\mathtt{nat(0)} \wedge \forall \mathtt{X}[\mathtt{nat(X)} \rightarrow \mathtt{nat(s(X))}].$$

Wie funktioniert der PROLOG-Interpreter?

Zusätzlich zu den Hilfsfunktionen des DATALOG-Interpreters brauchen wir folgende Funktionen.

mainsymb(t)**:** falls t der term $f(t_1, \ldots, t_n)$ ist, so ist mainsymb$(t) = f$ und falls t das Atom $r(t_1, \ldots, t_n)$ ist, so ist mainsymb$(t) = r$.

arity(t)**:** gibt die Stelligkeit des Terms oder Atoms t zurück, d. h. arity$(f(t_1, \ldots, t_n)) = n$ und arity$(r(t_1, \ldots, t_n)) = n$.

arg(t, i)**:** gibt das i-te Argument des Terms oder Atoms t zurück, falls ein solches existiert, also arg$(f(t_1, \ldots, t_n), i) = t_i$, falls $1 \leq i \leq n$, und arg$(r(t_1, \ldots, t_n), i) = t_i$.

Die Funktion **unify**(s, t, θ) gibt 'true' zurück, falls die beiden Terme oder Literale s und t unifizierbar sind und berechnet den allgemeinsten Unifikator in θ. Falls s und t nicht unifizierbar sind ist das Resultat 'false'.

```
function unify(s, t: atom ∪ term, var θ: substitution): boolean;
begin
  s := apply(s, θ);
  t := apply(t, θ);
  if variable(s) then θ := compose(θ, {t/s})
  elsif variable(t) then θ := compose(θ, {t/s})
  elsif (mainsymb(s) = mainsymb(t)) and (arity(s) = arity(t)) then begin
    for i := 1 to arity(s) do
      if not( unify(arg(s, i),arg(t, i),θ) ) then return(false) end
    end;
    return(true)
  else return(false)
end
```

Wie bei DATALOG versucht die Funktion **provable** ein Ziel mit einer Tiefensuche zu beweisen.

```
function provable(G: goal): boolean;
begin
  if goal = [] then return(true)
  else
    for i := 1 to MAXCLAUSES do
      C := rename(clause[i]);
      θ := ε;
      if unify(head(C), first(G), θ) then
        if provable(apply(concatenate(body(C), rest(G)), θ)) then
          return(true)
        end
      end
    end;
    return(false)
  end
end
```

Antworten des PROLOG-Interpreters

Das Konzept der Antworten ist bei PROLOG dasselbe wie bei DATALOG.

Aufgabe 1 Natürliche Zahlen (1 Punkt)

Gegeben ist folgendes PROLOG-Programm.

```
nat(z).
nat(s(X)) :- nat(X).
```

Schreiben Sie von Hand den Suchbaum zum Ziel `nat(s(s(z)))`. auf.

Aufgabe 2 Elementare Zahlentheorie (5 Punkte)

In dieser Aufgabe wird eine natürliche Zahl n dargestellt durch den PROLOG-Term

$$\underbrace{s(\cdots(s}_{n \text{ mal}}(z)\cdots).$$

Die Zahl 0 ist der Term z, die Zahl 1 der Term s(z), die Zahl 2 der Term s(s(z)), u. s. w.

(1) Schreiben Sie ein Prädikat `add(X,Y,Z)` mit der Bedeutung $\texttt{Z} = \texttt{X} + \texttt{Y}$.

(2) Schreiben Sie ein Prädikat `mul(X,Y,Z)` mit der Bedeutung $\texttt{Z} = \texttt{X} * \texttt{Y}$.

(3) Schreiben Sie die Prädikate `le(X,Y)` und `leq(X,Y)` mit der Bedeutung $\texttt{X} < \texttt{Y}$ und $\texttt{X} \leq \texttt{Y}$.

(4) Schreiben Sie die Prädikate `min(X,Y,Z)` und `max(X,Y,Z)` mit der Bedeutung $\texttt{Z} = \min(\texttt{X}, \texttt{Y})$ und $\texttt{Z} = \max(\texttt{X}, \texttt{Y})$.

(5) Wie lauten die Fragen, die man stellen muß um die Differenz $3 - 2$ und den Quotienten $4/2$ zu berechnen.

Hinweis: Schreiben Sie die folgende Operatordeklaration an den Anfang des Programms:

```
:- op(100,fy,s).
```

Dadurch wird das Funktionssymbol `s` als Prefix-Operator deklariert. Den Term `s(s(s(z)))` kann man nun als `s s s z` schreiben.

Aufgabe 3 (1 Punkt)

Gegeben ist folgendes PROLOG-Programm.

```
equal(X,X).
test :- equal(X,f(X)).
```

Welche Antworten liefert der Algorithmus bei den Zielen `?-test` und `?-equal(X,f(X))`? Welche Antworten liefert SWI-Prolog? Wie erklären Sie sich den Unterschied?