Übungen zur Vorlesung Mathematische Logik

PD Dr. Fritz Hamm, Prof. Dr. P. Schroeder-Heister **Blatt 11**

Die Übungen basieren zu einem großen Teil auf einem Praktikum von Prof. R. Stärk.

DATALOG – PROLOG mit Variablen und Konstanten

Syntax von DATALOG:

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

Ein Term ist eine Variable oder eine Konstante. Ein Atom ist ein Ausdruck der Form

$$r(t_1, \ldots, t_n),$$

wobei r ein Relationssymbol ist und $t_1, \ldots, t_n$ Terme sind.

Bsp. eines DATALOG-Programms:

```
r(c0,c1).
r(c1,c2).
q(X,Y) :- r(X,Y).
q(X,Z) :- r(X,Y), r(Y,Z).
```

Die Bedeutung der letzten Klausel ist

$$\forall X \, \forall Y \, \forall Z \, [r(X,Y) \wedge r(Y,Z) \rightarrow q(X,Z)],$$

oder äquivalent dazu

$$\forall X \, \forall Z \, [\exists Y \, (r(X,Y) \wedge r(Y,Z)) \rightarrow q(X,Z)].$$

Wie funktioniert ein DATALOG Interpreter?

Wir nehmen an, daß Substitutionen Listen von Bindungen sind, also beispielsweise

$$\{t_1/X_1, \ldots, t_n/X_n\}.$$

Dabei sind $X_1, \ldots, X_n$ Variablen und $t_1, \ldots, t_n$ Terme.

relsymb(A)**:** falls A ein Atom ist, so gibt diese Funktion das Relationssymbol von A zurück, d.h. relsymb($r(t_1, \ldots, t_n)$) $= r$.

arity(A)**:** gibt die Stelligkeit des Relationssymbols des Atoms A zurück, d.h. arity(r) $= 0$ und arity($r(t_1, \ldots, t_n)$) $= n$.

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

apply(L, θ)**:** wendet die Substitution θ auf L an und erzeugt eine neue Kopie. L kann ein Atom oder eine Liste von Atomen sein; apply($[A_1, \ldots, A_n], \theta$) $= [A_1\theta, \ldots, A_n\theta]$ und apply(A, θ) $= A\theta$.

compose(θ_1, θ_2)**:** berechnet die Komposition der beiden Substitutionen θ_1 und θ_2, also compose(θ_1, θ_2) $= \theta_1\theta_2$.

rename(C)**:** gibt eine Umbenenung der Klausel C zurück und erzeugt eine neue Kopie. Wir nehmen an, daß es einen globalen Zähler i gibt, so daß rename(C) in der Klausel C alle Variablen mit dem Subskript i versieht und nachher i um 1 erhöht. Zum Beispiel, falls $i = 3$, dann ist

$$\text{rename}(q(X, Z) \text{ :- } r(X, Y), r(Y, Z)) = q(X_3, Z_3) \text{ :- } r(X_3, Y_3), r(Y_3, Z_3).$$

variable(t)**:** gibt 'true' zurück, falls der Term t eine Variable ist und 'false' sonst.

Die Funktion **match**(A, B, θ) gibt 'true' zurück, falls A und B unifizierbar sind und berechnet den allgemeinsten Unifikator in θ. Falls A und B nicht unifizierbar sind ist das Resultat 'false'.

```
function match(A, B: atom, var θ: substitution): boolean;
begin
  if (relsymb(A) ≠ relsymb(B)) or (arity(A) ≠ arity(B)) then return(false)
  else
    θ := {};
    for i := 1 to arity(A) do
      if variable(arg(A, i)) then
        θ := compose(θ, {arg(B, i)/arg(A, i)}); A := apply(A, θ); B := apply(B, θ)
      elsif variable(arg(B, i)) then
        θ := compose(θ, {arg(A, i)/arg(B, i)}); A := apply(A, θ); B := apply(B, θ)
      elsif arg(A, i) ≠ arg(B, i) then return(false) end
    end;
    return(true)
  end
end
```

Die Funtkion *provable* versucht, ein Ziel mit einer *Depth-First* Suche zu beweisen.

```
function provable(G: goal): boolean;
begin
  if goal = [] then return(true)
  else
    for i := 1 to MAXCLAUSES do
      C := rename(clause[i]);
      if match(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

Ein DATALOG-Interpreter versucht nicht nur zu beweisen, daß ein Ziel aus einem Programm folgt, sondern er berechnet dabei auch mögliche Substitutionen für Variablen in der Anfrage. Im Beispiel oben sind die möglichen Antworten auf die Frage `?- q(U,V)`:

(1) $\{$`c0/U`, `c1/V`$\}$.

(2) $\{$`c1/U`, `c2/V`$\}$.

(3) $\{$`c0/U`, `c2/V`$\}$.

Den DATALOG-Interpreter kann man leicht dahingehend modifizieren, daß er Antwortsubstitutionen berechnet. Angenommen, im Ziel `?-` $B_1, \ldots, B_m$ kommen die Variablen $X_1, \ldots, X_n$ vorkommen. Dann ändere man das Ziel zu `?-` $B_1, \ldots, B_m$, \$answer$(X_1, \ldots, X_n)$ ab. Die Funktion *provable* testet nun nicht mehr auf goal = [] sondern auf goal = [\$answer$(t_1, \ldots, t_n)$], und, wenn dies zutrifft, gibt sie die folgende Anwort zurück:

$$\{t_1/X_1, \ldots, t_n/X_n\}$$

Aufgabe 1 (2 Punkte)

Was geschieht, wenn in einem Programm die Klausel `p :- p` steht und man die Frage `?- p` stellt?

Aufgabe 2 (2 Punkte)

Wie kann man die folgende Aussage als DATALOG-Programm (um)schreiben? „Falls `p` wahr ist und `q` oder `r` wahr ist, so ist `s` wahr.“

Aufgabe 3 (4 Punkte)

Ein *vollständiger* Suchbaum umfaßt alle überhaupt durch das Programm erreichbaren Äste, d.h. er endet nicht mit dem ersten Ast, der im leeren Ziel endet. Schreiben Sie den vollständigen Suchbaum auf für das Ziel `?- p` und folgendes DATALOG-Programm.

```
p :- q, fail.
p :- q, s.
q :- t, u.
s :- t.
t.
t :- v.
u.
v.
```

Aufgabe 4 (4 Zusatzpunkte)

Geben Sie ein endliches DATALOG-Programm P und ein Ziel G, so daß der vollständige Suchbaum von G unendlich viele verschiedene Pfade hat, die in das leere Ziel enden.

Aufgabe 5 (4 Punkte)

Für welche der folgenden Paare von Atomen gibt die Funktion *match* den Wert *true* zurück? Berechnen Sie den allgemeinsten Unifikator θ für diese Paare.

(1) `p(a,X,X,b)` und `p(Y,Y,Z,Z)`

(2) `p(a,X,b,X)` und `p(Y,Y,Z,Z)`

(3) `r(X,V,X,V)` und `r(Y,Y,Z,Z)`

(4) `r(X,V,V,a)` und `r(Y,Y,Z,Z)`

Aufgabe 6 (4 Punkte)

Gegeben ist folgende Datenbank:

```
r(c0).
r(c1).
r(c2).
q(c1).
q(c2).
q(c3).
```

(1) Schreiben Sie ein 1-stelliges Prädikat `intersection(X)`, das wahr ist, falls sowohl `r(X)` als auch `q(X)` gilt.

(2) Schreiben Sie ein 1-stelliges Prädikat `union(X)`, das wahr ist, falls `r(X)` oder `q(X)` gilt.

Aufgabe 7 (8 Punkte)

Gegeben ist folgendes DATALOG-Programm:

```
r(c0,c1).
r(c1,c2).
q(X,Y) :- r(X,Y).
q(X,Z) :- r(X,Y), r(Y,Z).
```

Schreiben Sie von Hand die vollständigen Suchbäume für die folgenden Ziele auf:

(1) `?- q(c0,c2).`

(2) `?- q(U,V).`