IV.1 Grundkonzepte der logischen Programmierung

Dienstag, 20. Januar 2015 16:00

Idee:

Programm ≙
Wissensbasis

Ausführung d. Prog ≙
Stelle Fragen an
Prog.
↳ "Rechnen" =
"Beweisen"

Prolog =
Programming in
Log
aus
ist automat.
Beweisen ent-
standen.

Prolog-Prog. besteht aus
logischen Formeln
("Klauseln").
- Fakten
- Regeln

- Faktum:

Präd.Nat($Obj_1, \ldots, Obj_n$).
↑
Name
einer
Eigenschaft
(weiblich,
mutterVon, ...)

Eigenschaften/Relationen sind gerichtet.
Jedes Prädikat hat
eine Stelligkeit:
weiblich/1

verheiratet/2

Jede Aussage wie

weibl.ich(gerd)

ist wahr oder falsch.

- Kommentare:
  % bis Zeilenende

  oder

  /*
  ⋮
  */
- Prolog verwendet die "Closed World Assumption":
  Alles, was nicht aus Prog-Klauseln folgt, ist falsch.

Führe nun weitere Features v. Prolog ein

Variablen in Prog-Klauseln

- Prädikate u. Objekte beginnen mit Kleinbuchstaben
- Variablen beginnen mit Großbuchst. oder _ .
- Variablen in Prog-Klauseln sind allquantifiziert

(Bsp: alle Objekte sind Menschen)

```
?- mensch(X).
.
true
```

⇒ Prolog ist eine ungetypte Sprache

* Gleiche Variablen in einer Klausel müssen gleich instantiiert werden.

mag(X,Y).
"Jeder mag jeden."

mag(X,X).
"Jeder mag sich selbst."

Gleiche Var. in verschiedenen Klauseln haben nichts miteinander zu tun.

Variablen in Anfragen

-nötig, um das Prog. selbst Lösungen berechnen zu lassen.

* Var. in Anfragen sind existenzquantifiziert

Bsp: Gibt es ein X, das die Mutter von Susanne ist?
Was sind die Kinder von Renate?

Es liegt nicht fest, was Ein- u Ausgabeargumente sind, sondern dies hängt davon ab, an welchen Stellen Variablen in der Anfrage sind.

Anfrage kann mehrere Antworten haben.
";" zwingt Prolog zum Weitersuchen.
↑ steht für

"oder"

Kombination v. Fragen

Bsp: Ist gerd der Vater von susanne?

Prolog baut Beweisbaum auf:

?- ver(g, F), mV(F, sus)

| F = renate

?- mV(ren, sus)

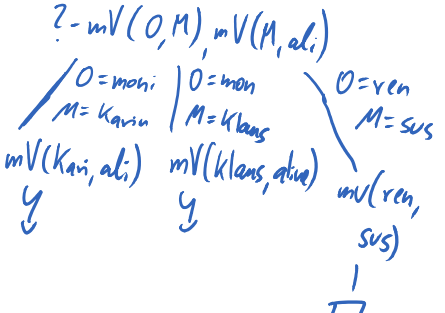
|

□

↑ leere Klausel, es muss nichts mehr bewiesen werden.

Dann gibt Prolog die Variablenbelegung aus, die beim Pfad vom ursprünglichen Beweisziel (a.d. Wurzel) zu □ verwendet wurde.

Zusammengesetzte Beweisziele werden von links nach rechts abgearbeitet.



Man hätte die Aussagen in der Anfrage vertauschen sollen, um den Suchraum einzuschränken.

Prolog baut Beweisbaum in Tiefensuche auf:

Suche auf:

- Setzt zurück bei Fehlschlag (Backtracking)
- Stoppt bei der ersten gefundenen Lösung
- Bei Eingabe von ";" wird der Baum weiter aufgebaut.

Regeln

p :- q, r .

Kopf | Rumpf

"falls"

bedeutet: p ist wahr, falls q und r wahr sind.

d.h.: Um p zu beweisen, reicht es, stattdessen q und r zu beweisen.

Bsp: V ist Vater von K, falls V mit Frau F verheiratet ist und F Mutter von K ist.

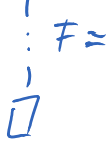
; ≙ oder

, ≙ und

?- vV(gerd, sus)

V = gerd
K = sus

?- verh(gerd, F), mV(F, sus)



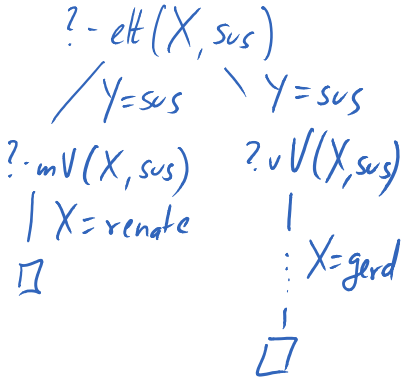
Prolog gibt die erfolgreiche Var-Belegung nur für die Variablen der ursprüngl. Anfrage aus.

Mehrere Regeln für ein Prädikat

Bisher: Regel für Prädikat, das gilt, wenn Bedg. 1 und Bedg. 2 erfüllt sind.

Jetzt: Präd, das gilt, wenn Bedg. 1 oder Bedg. 2 erfüllt sind.

Prolog
- arbeitet Prog-Klauseln von oben nach unten ab
- arbeitet zusammengesetzte Beweisziele v. links nach rechts ab



Auch möglich:

```
elternteil(X,Y) :-
    mV(X,Y) ; vV(X,Y).
```

Rekursive Regeln

Bsp: vorfahre

V ist vorfahre von X,
falls
V elternteil von X ist
oder
V ist elternteil v. Y und
Y ist vorfahre v. X

2. Klausel ist rekurs.v

Wer sind alines Vorfahren?