

Prinzip: Programm = Wissensbasis

Ausführung: Prog. versucht versucht, Anfragen mit Hilfe der Wissensbasis zu beantworten.

Bsp: Verwandtschaftl. Zusammenhänge
(→ Expertensysteme)

Prolog = Programming in Logic

Wissensbasis/Prog besteht besteht aus logischen Formeln (sogenannte Klauseln):

- Fakten (Aussagen über Objekte)
- Regeln (erlauben es, von bekannten Fakten auf neue Fakten zu schließen)

Fakten:

- Name der Eigenschaft/Relation/ (Objekte, die diese Eigenschaft haben)
Prädikate
- Relationen haben Stelligkeit und sind gerichtet
- Prädikate/Relationen und Objekte beginnen mit Kleinbuchstaben.

Kommentare:

- % bis Zeilenende
- /*
 :
 */

Prolog-Ausführung

- Programm laden:
 consult dateiname.pl
- Stelle Anfragen,
 Prolog versucht, sie zu
 beweisen.
- Prolog verwendet
 Closed World Assumption
 d.h.: alles, was nicht aus
 den Formeln des Programms
 folgt, ist falsch.

?- verheiratet(monika, werner).
false

- Namen der Relationen
 und der Objekte beginnen
 mit Kleinbuchstaben.

Variablen in Programm- Klauseln

Variablen in Programm- Klauseln

- Variablen: Strings, die mit Großbuchstaben oder _ beginnen.
- Variablen im Programm sind allquantifiziert, d.h., sie stehen für alle möglichen Belegungen.

?-mensch (5).

true

- Mehrfache Vorkommen der gleichen Variable in der gleichen Klausel müssen gleich instanziiert werden.

mag (X, Y). ← Jeder mag jeden

mag (X, X). ← Jeder mag sich selbst.

Gleiche Variablen in verschiedenen Klauseln haben nichts miteinander zu tun.

Variablen in Anfragen

- Variablen in Anfragen sind existenzquantifiziert.
"Existiert ein X , so dass $\text{mutterVon}(X, \text{susanne})$ wahr ist?"
d.h. "Wer ist die Mutter von susanne?"
- Es ist nicht festgelegt, welches Argument eines Prädikats Ein- bzw. Ausgabe ist, sondern dies hängt von der Anfrage ab.
- Prolog durchsucht die Wissensbasis von oben nach unten und gibt die erste gefundene Antwort aus.
Wenn man danach ";" eingibt, sucht Prolog nach der nächsten Lösung etc.
(; entspricht "oder")

Kombination von Anfragen

- Man kann mehrere Aussagen "gleichzeitig" beweisen lassen, indem man sie mit ";"

Konjunktiv

(, entspricht "und")

Bsp: Ist gerd der Vater
von susanne?

Prolog beweist solche Anfragen,
indem es einen Beweisbaum
konstruiert:

?-verk(gerd, F), mV(F, sus).

| F=renate

?-mV(renate, sus).

|
□

← leere Klausel,
d.h., es muss nichts
mehr bewiesen
werden

Aus dem erfolgreichen Beweispfad
(von ursprünglicher Anfrage
zu □) lässt sich die
Antwortsubstitution

F = rena

ablesen.

• Zusammengesetzte Formeln
werden von links nach rechts
bearbeitet.

?- $mV(O, M)$, $mV(M, aLine)$.

$O = \text{monika}$
 $M = \text{Karin}$

$O = \text{moni}$
 $M = \text{Klaus}$

$O = \text{ren}$
 $M = \text{sus}$

?- $mV(\text{Karin}, aLine)$.

?- $mV(\text{Klaus}, aLine)$.

?- $mV(\text{sus}, aLine)$

↓

↓

↓
□

Wenn ein Beweisversuch fehlschlägt, dann setzt Prolog zum nächst-
oberen Beweisziel zurück (Backtracking) und versucht dieses
mit der nächsten möglichen Programmklausel zu lösen.

Im Bsp hätte man
 $mV(O, M)$ und $mV(M, aLine)$
vertauschen sollen, um die
Lösung schneller zu finden.

Regeln

• Neben Fakten kann ein
Prog. auch Regeln ent-
halten. Diese dienen dazu,
neues Wissen herzuleiten.

• :- bedeutet "falls"

Kopf
 $P :- q, r.$
Rumpf

bedeutet: P ist wahr,

falls q und r wahr sind.

bzw. wenn q und r wahr sind,
dann ist auch p wahr

?-vaterVon(gerd, sus).

| V = gerd
| K = sus

?-verl(gerd, F), mV(F, sus).

| F = reate

?-mV(reate, sus)

|
□

Die Antwortsubstitution ist
nur der Teil der Variablenbe-
legung, der die Variablen
in der ursprünglichen Anfrage
betrifft.

Mehrere Regeln für ein Prädikat

?-elternteil(X, susanne).

/ Y = sus

\ Y = sus

?-mV(X, sus)

?-vaterVon(X, sus)

| X = reate

|

| X = gerd

□

□

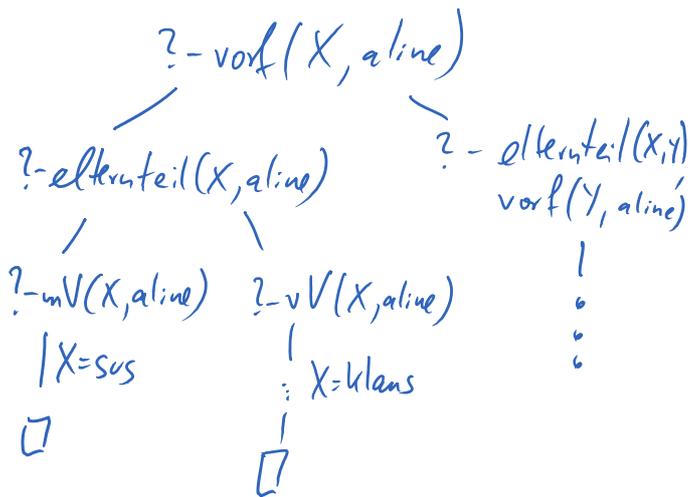
Rekursive Regeln

Prädikat aus dem Kopf der Regel tritt auch im Rumpf der Regel auf.

- V ist vorfahre von X , falls V ein elternteil von X ist.
- V ist vorfahre von X , falls V elternteil von einem Y ist und Y ist vorfahre von X



führt zu rekursiver Regel



Keine Kontrollstrukturen,
aber:

- Prog-Klauseln werden

von oben nach unten
abgearbeitet.

- Folgen von Beweiszielen
(z.B. im Rumpf einer Regel)
werden von links nach rechts
abgearbeitet.