

Modellierung 9.1.08

Notiztitel

09.01.2008

2.5 Prolog

Variablen, Klauseln und Prädikate

Prolog-Programm:

- Fakten
 - Regeln
 - Anfragen
- } Horn-Klauseln, bilden Wissensbasis

Variablen: X, Y, Z, Alter (beginnen mit Grossbuchstaben)

Konstanten: abc, anton, 1234 (beginnen mit Kleinbuchst./Zahl)

Praedikate: person(), kind_von(,) (beginnen mit Kleinbuchst.)

Funktor Argumente

Fakten: person(klaus), kind_von(klaus,maria)

Stelligkeit: Anzahl Argumente eines Praedikates

Praedikate unterscheiden sich durch

- Funktor
- Stelligkeit

Beispiel: kind(paul). kind(maria,paul). /* nicht empfehlenswert */

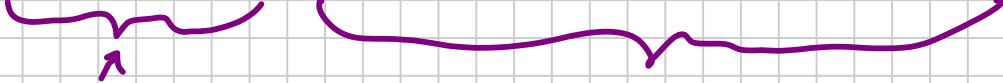
Anfragen:

?- hund(fido).

?- hund(X).

Regeln:

vater(X,Y):- elternteil(X,Y), mann(X).



Regelkopf Regelrumpf

$vater(x,y) \leftarrow elternteil(x,y), mann(x)$
 $\{vater(x,y), \neg elternteil(x,y), \neg mann(x)\}$

Rekursive Regeln:

Praedikat aus dem Regelkopf taucht im Rumpf auf

vorfahr(X,Y):-elternteil(X,Y).

vorfahr(X,Y):-elternteil(X,Z),vorfahr(Z,Y).

Reihenfolge der Regeln wichtig:

'Abbruchbedingung' bei der Rekursion

Beispiele

1) Regeln

$(A_1 \wedge \dots \wedge A_n \rightarrow B)$

$B :- A_1, \dots, A_n$

'deklarative' Lesart vs. 'prozedurale' Lesart

gluecklich(peter):- sonne, freizeit.
gluecklich(lisa):- gluecklich(peter),
in_der_naehelisa,peter).

2) Fakten

$(1 \rightarrow A)$

$A :- true$

A.

sonne:-true.
freizeit.
in_der_naehelisa,peter).

3) Ziele

$(G_1 \wedge \dots \wedge G_n \rightarrow 0)$

?- G_1, \dots, G_n .

?- gluecklich(lisa).

?-gl(lisa). gl(lisa):-gl(peter), idn(lisa,peter).

{~gl(lisa)} {gl(lisa), ~gl(peter), ~idn(lisa,peter)}

gl(peter):-sonne, freizeit.

{~gl(peter), ~idn(lisa,peter)} {gl(peter), ~sonne, ~freizeit}

sonne. freizeit.

{~idn(lisa,peter), ~sonne, ~freizeit} {sonne} {freizeit}

(nicht wie in Prolog!)

{~idn(lisa,peter), ~freizeit} {freizeit}

idn(lisa,peter).

{~idn(lisa,peter)} {idn(lisa,peter)}



Beobachtung:

Der Beweisgraph ist eine Kette (linear)

In jedem Resolutionsschritt wird eine Zielklausel mit einer Programmklausel resolviert.

Ergebnis: neue Zielklausel (nur negierte Literale)

"Freiheitsgrade":

- Auswahl des Literals in der Zielklausel
- Auswahl der Programmklausel

Prolog: SLD-Resolution

Selection Linear Definite Clause



Erweiterung:

- Programmklauseln mit Variablen
- Resolution mit Variablen-Substitution

?-gl(lisa). gl(X):-liebt(X,Y), gl(Y), idn(X,Y).

{~gl(lisa)} {gl(X), ~liebt(X,Y), ~gl(Y), ~idn(X,Y)}

Substitution: $\theta = \{X/lisa\}$

{~liebt(lisa,Y), ~gl(Y), ~idn(lisa,Y)}

Substitution: Ersetze alle Vorkommen der Variablen durch den Wert

summe(Z,0,Z). /* summe(X,Y,Z): X+Y=Z */ (Peano - Axiome)
summe(X,n(Y),n(Z)):- summe(X,Y,Z).

Beweise: summe(n(0), n(0), n(n(0))). (1+1=2)

{~summe(n(0),n(0),n(n(0)))} {summe(X,n(Y),n(Z)), ~summe(X, Y, Z)}

$\sigma = \{x/n(0), y/0, z/n(0)\}$

{~summe(n(0),0,n(0))} {summe(Z,0,Z)}

$\sigma = \{z/n(0)\}$
□

Antwortherzeugung:

?-antwort(S). antwort(S) :- summe(n(0), n(0), S).

antwort(n(n(0)))

{~antwort(S)} {antwort(S), ~summe(n(0), n(0), S)}

{~summe(n(0), n(0), S)} {summe(X,n(Y),n(Z),~summe(X,Y,Z))}

$\sigma = \{X/n(0), Y/0, S/n(Z)\}$

$S = n(n(0))$

{~summe(n(0),0,Z)} {summe(U,0,U)}

$\sigma = \{U/n(0), \underline{Z/n(0)}\}$

□

Erweiterungen:

Variablenumbenennungen ("standardizing apart")

Term-Vergleich (Unifikation) Beispiel: $f(X, g(X))$ $f(h(Z), g(h(Z)))$

"Negation as failure"

p : - a, not b, c

~~$\{p, \neg a, \neg \neg b, \neg c\}$~~

b nicht beweisbar not b = true