

Prolog 1-1

- Prolog znamená programování v logice
- namísto otázky **jak** se má získat určitá hodnota se ptáme **co** platí mezi objekty
- vhodný jazyk pro úlohy kde nás zajímají vztahy mezi objekty
- namísto definic funkcí zapisujeme klauzule, které společně s vyhodnocovacím mechanismem umožňují odpověď uživateli na otázku zda nějaký vztah je splnitelný nebo ne

klauzule - 3 typy:

fakta - vyjadřují bezpodmínečně pravdivá tvrzení
term.

pravidla - vyjadřují tvrzení závislá na splnění určitých podmínek
term0 := term1, term2, ..., termN.

dotazy - vyvolávají výpočet za účelem zjistit, zda je dotaz splnitelný či nikoliv
?- term1, term2, ..., termN.

Během procesu vyhodnocení dochází v případě úspěchu k nastavení proměných na takové hodnoty, které umožní splnitelnost. Je-li více možností je Prolog schopen nalézt je všechny pomocí zpětného prohledávání (backtracking).

Rodinné vztahy pomocí Prologu:

fakta: `rodic(jarda,marie).`
`rodic(jarda,pepa).`
`rodic(marie,bozena).`
`rodic(marie,tonda).`
`muz(tonda).`
`zena(bozena).`

dotazy:

?- `rodic(jarda,pepa).` -> yes

?- `rodic(jarda,tonda).` -> no

?- `rodic(jarda,X)` -> `X=marie;` %středník vyvolá další řešení
`Y=pepa;`
`no`

Řetězec, který začíná velkým písmenem je považován za proměnou a Prolog se snaží nastavit jeho hodnotu, tak aby dotaz byl splnitelný. Anonymní proměnnou, která nás nezajímá značíme pomocí podtržítka `_`.

pravidla: (zkusíme nalézt jméno Jardovy babičky)

`babicka(Vnuk,X) :- rodic(Vnuk,Y), rodic(Y, X), zena(X).`

dotaz: ?- `babicka(jarda, X)` -> `X=bozena;`
`no`

- nejdříve se nalezne relace ve které Vnuk (Jarda) vystupuje jako syn nějakého rodiče Y. Dále pro osobu Y zkusíme nalézt relaci, ve které vystupuje jako dítě Xka. Pro X potom ještě ověříme existenci relace býti ženou.

Čárky mezi termy mají tedy význam logického AND.

Prolog 1-2

Datové objekty v Prologu

- jednoduché objekty - konstanty

- atomy např. jarda, neco

- čísla např. 10, -3

- proměnné např. X, Vnuk, _

- struktury např. vyuka(predmet(jui, cviceni), doba(streda), ucitel(dobsicek, miroslav))

Dotaz na všechna cvičení konkrétního učitele:

cvicici(Ucitel, Predm) :- vyuka(predmet(Predm, cviceni), _, Ucitel).

Deklarativní a procedurální sémantika programu

Mějme klazuli tvaru $P :- Q, R$, kde P, Q, R jsou libovolné tvary termů.

deklarativní pohled: P je pravdivé, jestliže Q i R jsou pravdivé; z platnosti Q a R plyne platnost P ; atd ..

procedurální pohled: aby se vyřešil problém P , je potřeba nejprve vyřešit problém Q a potom problém R ; aby se splnil cíl P , je potřeba nejprve splnit cíl Q a potom cíl R

Seznamové a číslicové operace

Seznam se v Prologu zapisuje jako $[bohous, jarda, pepa, milan]$. Vnitřní reprezentace je opět pomocí tečka dvojice $(bohous, (jarda, (pepa, (milan, [])))$). Prázdný seznam se značí pomocí $[]$, atom nil zde neexistuje.

Pomocí operátoru $|$ můžeme seznam rozdělit na prvních pár prvků a zbytek.

$[bohous, jarda | [pepa, milan]]$

Predikát member:

$member(X, [X | Zbytek])$.

$member(X, [_ | Zbytek]) :- member(X, Zbytek)$.

příklad:

$member(2, [1,2,3])$. -> yes

$member(X, [1,2,3])$. -> $X = 1$;

$X = 2$;

$X = 3$;

no

Mezi jednotlivými klauzulemi je logiké OR. Protože v podstatě provádíme definici vzorem musí umět Prolog důkladně srovnat hlavu pravidla a cíle. Dva termy se rovnají když jsou **identické** nebo **proměnné lze nastavit** na takové hodnoty aby se získaly identické termy -> **unifikace** – aktivní mechanismus jehož výsledkem je ano/ne.

Vložení na začátek seznamu:

$ins(X, L, [X | L])$.

příklad:

$ins(elf, [honza, pepa], L)$. -> $L = [elf, honza, pepa]$

Prolog 1-3

Ubrání prvku ze seznamu:

del1(X, [X|T], T).

příklad:

del1(1, [1,2,3], X) => X=[2,3]

delx(X, [X|T], T).

delx(X, [Z|L], [Z|LO]) :-

delx(X, L, LO).

příklad:

delx(1,[1,2,1,3],X). => X=[2,1,3];
X=[1,2,3];
no

Test na seznam:

list([]).

list([_|_]).

příklad:

list([1,2,3]). => yes
list(1). => no

Spojení dvou seznamů:

append([], L, L).

append([Head | Rest], L, [Head | Newrest]) :-

append(Rest, L, Newrest).

příklad:

append([1,2], [3,4], [1,2,3,4]). => yes
append([1,2], [3,4], X). => X=[1,2,3,4]
append([1,2], X, [1,2,3,4]). => X=[3,4]
append(X, [3,4], [1,2,3,4]). => X=[1,2]

Test na celé kladné číslo:

isint(0).

isint(N) :-

isint(N1),

N is N1 + 1.

příklad:

isint(5) -> true % při pokusu o další vyhodnocení nám to skončí v nekonečném hledání
isint(-5) -> % zkočíme v nekonečném hledání

Prolog 1-4

Součet čísel v seznamu:

```
sum([], 0).
sum([H | T], S) :-
    integer(H),
    sum(T, S1),
    S is S1 + H.
```

příklad:

```
sum([1,2,3], X). => X=6
```

Délka seznamu:

```
len([], 0).
len([_ | T], N) :-
    len(T, N1),
    N is N1 + 1.
```

příklad:

```
len([1,2,3], X). => X=3
```

Výběr nejmenšího prvku seznamu:

```
min_list([X], X).
min_list([H | T], H) :-
    min_list(T, X),
    H =< X.
min_list([H | T], X) :-
    min_list(T, X),
    H > X.
```

příklad:

```
min_list([3,4,1,2], X). => X=1
```

Jednosměrný faktoriál:

```
fact(0, 1).
fact(N, F) :-
    N > 0,
    N1 is N - 1,
    fact(N1, F1),
    F is F1 * N.
```

příklad:

```
fact(5, X). => X=120      % jedním směrem je to fajn
fact(X, 120). => ???     % druhým už moc ne
```

Prolog 1-5

Obousměrný faktoriál:

testfact(Akt,Poz,F) :-

fact(Akt,F),

Poz is Akt.

testfact(Akt,Poz,F) :-

fact(Akt,FS),

FS < F,

S1 is Akt + 1,

testfact(S1,Poz,F).

fact(0,1).

fact(N,F) :-

var(N),

testfact(1,N,F).

fact(N,F) :-

N > 0,

N1 is N - 1,

fact(N1,F1),

*F is F1 * N.*

příklad:

fact(5,X). => X=120

fact(X,120). => X=5