FLP páté cvičení

Martin Hyrš, Marek Milkovič, Lukáš Zobal

Faculty of Information Technology

Brno University of Technology

Funkcionální a logické programování, 2019/2020

Dynamické predikáty

Dynamické predikáty můžeme ”za běhu” vkládat a odstraňovat z databáze.

- `:- dynamic name/arity, name2/arity2.` – deklarujeme predikáty jako dynamické
- `assert(Term)` – vloží predikát (fakt nebo klauzuli) do databáze
- `retract(Term)` – unifikuje term a odstraní odpovídající predikát z databáze
- `retractall(Head)` – odstraní všechny predikáty s hlavičkou unifikovatelnou s `Head` z databáze
- `listing(Pred)` – zobrazí predikáty v databázi

`assert`(*+Term*)

Equivalent to `assertz/1`.
Deprecated: new code should use `assertz/1`.

Podmíněný ”příkaz” v Prologu

Můžeme specifikovat, kterou cestou se má vyhodnocování ubírat na základě podmínky – podobné if-then-else.

Sémantika

```
If -> Then ; _Else :- If, !, Then.
If -> _Then ; Else :- !, Else.
```

Poznámka: `!, If -> Then`

Poznámka2: `( If -> Then ; Else )`

```
signum(Num, Sig) :-
   Num==0 -> (
      Sig = 0
   ) ; (
      Num>0 -> Sig = 1 ; Sig = -1
   ).

signum2(0, 0).
signum2(N, -1) :- N<0.
signum2(N, 1) :- N>0.
```

Predikáty `bagof`, `setof`

Predikát `bagof/3` nalezne všechny unifikace dané proměnné/vzoru, které splní daný cíl: `bagof(Vzor, Cíl, Bag)`:

- Vzor – co chci unifikovat
- Cíl – cíl, pro který se unifikace hledají
- Bag – výsledný seznam všech navázání

`setof` navyše ještě seřadí výsledný seznam a odstraní duplicity.

Predikáty `bagof`, `setof`, `findall`

Příklad

```
street(bozetechova, brno, czechia).
street(holandska, brno, czechia).
street(vrsovicka, prague, czechia).
```

Příklad

```
?- bagof(St, street(St, brno, czechia), Sts).
Sts = [bozetechova, holandska].
?- bagof(St, street(St, City, Country), Sts).
City = brno
Country = czechia
Sts = [bozetechova, holandska]
...
?- bagof(St, City^street(St, City, Country), Sts).
Country = czechia
Sts = [bozetechova, holandska, vrsovicka]
```

1. úkol: Podmnožiny

Vytvořte predikát `subbags/2` pro výpočet množiny všech podmnožin[1]. Využijte predikát `append/3`.

Řešení

```
subbags( [], [[]]).
subbags([X|XS], P) :-


addOneToAll(_, [], []).
addOneToAll(E, [L|LS], [[E|L]|T]) :-
```

[1] přesněji multimnožinu všech podmultimnožin – nestarejte se o opakované výskyty

2. úkol: Roboti

Mějme v databázi dynamické predikáty `robot/2` a `dira/1`. Robotí svět je pouze jednorozměrný a diskrétní, jejich pozice je tedy dána celým číslem a mohou se pohnout doleva či doprava. Na jedné pozici může být maximálně jeden robůtek.
Predikáty mají tento význam:

- `robot(id, pos)` – robot `id` je na pozici `pos`
- `dira(pos)` – na pozici `pos` je díra

Půjdeme na to postupně. . .

Vytvořte predikáty:

- `obsazeno(pos)` – uspěje, pokud je na pozici díra či robot
- `vytvor(id, pos)` – vytvoří robůtka na pozici
- `vytvor(pos)` – vytvoří díru na pozici
- `odstran(pos)` – odstraní cokoliv je na pozici

Řešení

```
obsazeno(P) :-
vytvor(I, P) :-
vytvor(P) :-
odstran(P) :-
```

2b: Obsazené pozice

S pomocí `bagof` vytvořte predikát `obsazene_pozice(X)`, který vrátí seznam obsazených pozic, a predikát `obsazene_roboty(X)`, který vrátí seznam pozic obsazených roboty – ne dírami.

Řešení

```
obsazene_pozice(X) :-
obsazene_roboty(X) :-
```

Vytvořte predikáty `doleva/1` a `doprava/1`, které pohnou robotem s daným `id`.

Řešení

```
inkrementuj(X,Y) :- Y is X+1.
dekrementuj(X,Y) :- Y is X-1.
doleva(I) :- pohni(I, dekrementuj).
doprava(I) :- pohni(I, inkrementuj).
pohni(I, Operace) :-
```

2d: Armageddon

Vytvořte predikát `armageddon/0`, který způsobí výbuch všech robotů – zůstanou na jejich místě díry.

Využijte predikát `forall(podmínka, akce).`

Řešení

```
armageddon :-
vybuch(P) :-
```

3. úkol: Odemykací gesta

Vytvořte predikát **`g_all/2`**, který bude postupně vracet odemykací gesta zadané délky pro matici o rozměrech 3x3. Předpokládejme, že je možné spojovat pouze body v osmiokolí a body se nesmí opakovat.

Postup

1. jedno spojení
2. gesta z dané pozice
3. všechna gesta

Vytvořte predikát **`g_allLength/1`**, který bude postupně vracet odemykací gesta všech délek.

3a: Odemykací gesta (jedno spojení)

Konvence – použijeme operátor `:` pro dvojici souřadnic.

Test pozice

```
g_size(3).

g_test(X:Y) :-
```

Spojení

```
g_move(X1:Y1, X2:Y2) :- X2 is X1 - 1, Y2 is Y1 - 1, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 - 1, Y2 is Y1 + 0, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 - 1, Y2 is Y1 + 1, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 + 0, Y2 is Y1 - 1, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 + 0, Y2 is Y1 + 1, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 + 1, Y2 is Y1 - 1, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 + 1, Y2 is Y1 + 0, g_test(X2:Y2).
g_move(X1:Y1, X2:Y2) :- X2 is X1 + 1, Y2 is Y1 + 1, g_test(X2:Y2).
```

3b: Odemykací gesta (gesta z dané pozice)

Forma: `g_one(X:Y, Len, L, Res)`

- **`X:Y`** – výchozí souřadnice,
- **`Len`** – požadovaná délka,
- **`L`** – navštívené souřadnice,
- **`Res`** – výsledné gesto

Př. použití: `g_one(2:2, 3, [], Res).`

Řešení

```
% testovani gesta (je L správně dlouhé?)
g_one(X:Y, Len, L, Res) :-

%dalsi gesta z pozice (posun na dalsi pozici Xn:Yn)
g_one(X:Y, Len, L, R) :-
```

3c: Všechna gesta dané délky

Všechna gesta

```
g_all(R, Len) :-
```

3d: Gesta všech délek

Vytvořte predikát **`g_allLength/1`**, který bude postupně vracet odemykací gesta všech možných délek. (Využijte predikát `g_all/2`.)

Všechny délky

```
g_allLength(R) :-

g_allLength(R, Len) :-
g_allLength(R, Len) :-
```

Hodnocena bude míra splnění zadání, kvalita řešení, čistota a kvalita kódu a vhodné užití komentářů.
Za inovativní přístup či obzvláště kvalitní řešení lze získat prémiové body navíc.

až 2 body navíc

- kvalitní řešení
- ? něco navíc

Odevzdaná verze musí splňovat zadání!
Funkčnost navíc aktivovat ručně (např. odkomentovat) a popsat v dokumentaci.

Vestavěné predikáty
není potřeba znovu implementovat kolo
(na cvičeních je nepoužíváme ze cvičných důvodů)

- Built-in predicates $\rightarrow$ Built-in list operations
- The swi-prolog library $\rightarrow$ Library(lists)

ne všechny knihovny fungují na referenčním stroji