Programovací jazyk Prolog

doc. Ing. Miroslav Beneš, Ph.D.
katedra informatiky FEI VŠB-TUO
A-1007 / 597 324 213
http://www.cs.vsb.cz/benes
Miroslav.Benes@vsb.cz

Obsah

- Logický program
- Fakt, dotaz, pravidlo
- Term, substituce
- Logická proměnná
- Rekurzivní pravidla
- Seznamy

Úvod

- Hlavní myšlenka:
  Využití počítače k vyvozování důsledků na základě deklarativního popisu
- Postup**:**
  - reálný svět →
  - zamýšlená interpretace →
  - logický model →
  - program
- Výpočet - určení splnitelnosti či nesplnitelnosti **cíle**, případně včetně vhodných **substitucí**.

Logický program

- **Fakta**

```
vek(petr, 30).
vek(jana, 24).
```

- **Pravidla**

```
starsi(X, Y) :-
   vek(X, V1), vek(Y, V2), V1 > V2.
```

- **Dotazy**

```
?- starsi(petr, jana).
Yes
```

Co je to dotaz?

- Odpověď na dotaz vzhledem k programu = určení, zda je dotaz logickým důsledkem programu.
- Logické důsledky se odvozují aplikací dedukčních pravidel, např.
  P ⊢ P (identita)
  - Je-li nalezen fakt identický dotazu, dostaneme Yes.
  - Odpověď No znamená pouze to, že z programu nelze platnost dotazu vyvodit.

Předpoklad uzavřeného světa

```
zvire(pes).
zvire(kocka).
?- zvire(pes).
Yes
?- zvire(zirafa).
No
```

=> Předpokládáme platnost pouze toho, co je uvedeno v programu.

Logická proměnná

- Představuje nespecifikovaný objekt
- Jméno začíná velkým písmenem

```
?- vek(jana, X).
X = 24.
?- vek(pavla, X).
No
```

- Existuje X takové, že vek(jana, X) lze odvodit z programu? Pokud ano, jaká je hodnota X?

Kvantifikátory

- likes(X, beer).
  Pro všechna X platí likes(X, beer).

- ?- likes(X, beer).
  Existuje X takové, že likes(X, beer)?

Term

- Datová struktura definovaná rekurzivně:
  - Konstanty a proměnné jsou termy
  - Struktury jsou termy: *funktor(arg1, arg2, …)*
    - *funktor:* jméno začínající malým písmenem
    - *argument*: term
- Funktor je určen **jménem** a **aritou**
  - f(t1, t2, …, tn) …. f/*n*
- Příklad:
  - z/0 s/1 … z, s(z), s(s(z)), s(s(s(z)))

Substituce

- **Základní term** (ground term)
  - neobsahuje proměnné s(s(z))
- **Substituce**
  - Konečná množina dvojic ve tvaru $X_i=t_i$
  - Aplikace substituce θ na term A … Aθ
    f(X, a) {X=g(z), Y=b} = f(g(z), a)
- **Instance termu**
  - A je instancí B, existuje-li substituce θ taková, že A = Bθ
    f(g(z), a) je instancí termu f(X, a)

Konjunktivní dotazy

- ?- zvire(pes), zvire(kocka).
  Yes

Sdílení proměnných:

- ?- vek(X, V), vek(Y, V).
  Existují X a Y se stejným věkem?

Pravidla

- A :- B_1, B_2, …, B_n.
  A = **hlava** pravidla
  B_1, B_2, …, B_n = **tělo** pravidla

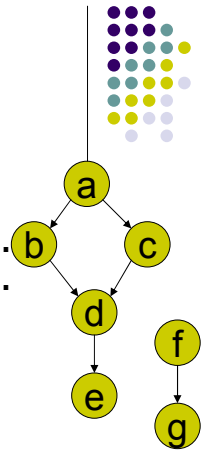
- ```
  syn(X, Y)  :- otec(Y, X), muz(X).
  deda(X, Y) :- otec(X, Z), otec(Z, Y).
  ```

- Proměnné jsou univerzálně kvantifikované.
- Platnost proměnných je celé pravidlo.
  ```

Rekurzivní pravidla

- **Definice grafu**

```
edge(a, b). edge(a, c). edge(b, d).
edge(c, d). edge(d, e). edge(f, g).
```



-
```
connected(N, N).
connected(N1, N2) :-
   edge(N1, L), connected(L, N2).
```


Rekurzivní pravidla

- **Přirozená čísla:**
  nat(z).
  nat(s(X)) :- nat(X).

- leq(z, X) :- nat(X).
  leq(s(X), s(Y)) :- leq(X, Y).

- add(z, X, X).
  add(s(X), Y, s(Z)) :- add(X, Y, Z).

Pracujeme s *relacemi* !

- `add(s(z), s(z), X).`
  `X=s(s(z))`
- `add(s(z), X, s(s(z))).`
  `X=s(z)`
- `add(X, Y, s(s(z))).`
  `X=z  Y=s(s(z)) ;`
  `X=s(z)  Y=s(z) ;`
  `X=s(s(z))  Y=z ;`
  `No`

Seznamy

- `list([]).`
  `list([X|Xs]) :- list(Xs).`

- `member(X,[X|Xs]).`
  `member(X,[Y|Ys]) :- member(X,Ys).`
  - ?- member(b, [a,b,c]).
  - ?- member(X, [a,b,c]).
  - ?- member(b, X)

Úkol

- Vytvořte predikát times/3 reprezentující operaci násobení
- Vytvořte predikáty even/1 a odd/1 pro ověření, zda je zadané číslo sudé, resp. liché.
- Vytvořte predikát prefix/2 pro zjištění, zda je jeden seznam prefixem druhého seznamu.