PROLOG
strategie vyhodnocení dotazu

Program v PROLOGu

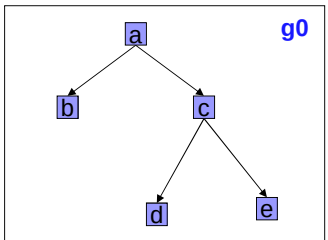
- Konečná **uspořádaná** množina faktů a pravidel (klauzulí), která se týkají téhož predikátu (t.j. predikátu uvedeného jako hlava pravidla), se nazývá **definice predikátu**.
- **Program** tvoří množina definic všech použitých predikátů.
- *Pro názornost budeme dále předpokládat, že **klauzule v programu jsou vzestupně očíslovány**, a to vždy od čísla* **1**.

- Jak popíšeme vztahy v grafu **g0**?

  Použijme označení

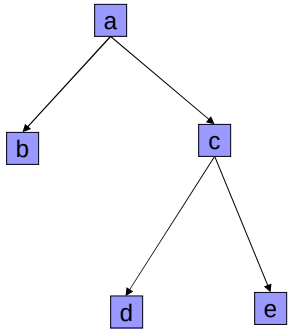
  hrana(Od,Do)

  cesta(Odkud,Kam)



Příklad „graf g0“, hledání cesty

```
hrana(a,b).  hrana(a,c).
hrana(c,d).  hrana(c,e).
c1(Y,Y).
c1(Z,K):-hrana(Z,M),c1(M,K).
   c2(Y,Y).
   c2(Z,K):-c2(Z,M),hrana(M,K).
c3(Z,K):-hrana(Z,M),c3(M,K).
c3(Y,Y).
   c4(Z,K):-c4(Z,M),hrana(M,K).
   c4(Y,Y).
```



Existuje cesta mezi uzly **b** a **d** nebo mezi **a** a **c**?

Které z řešení **c1**, **c2**, **c3** a **c4** je nejlepší?

Vyhodnocení dotazu

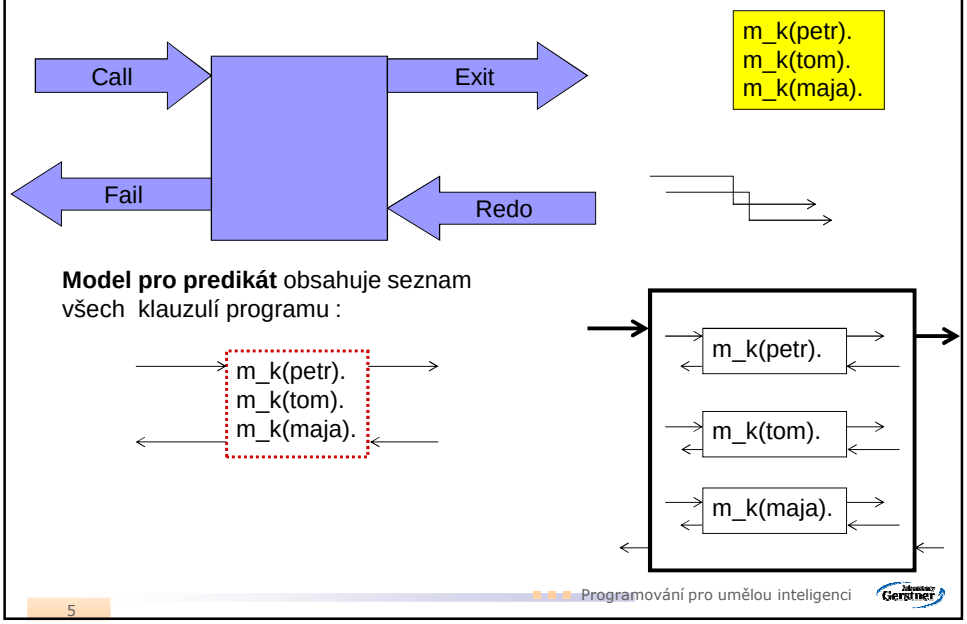
- Vyhodnocení dotazu **?-d1,…,dn.** programem **P** probíhá tak, že se program snaží postupně „zjednodušovat“ výchozí dotaz opakovaným využíváním svých klauzulí s cílem vytvořit **dotaz prázdný**.

  Jakmile se podaří dotaz redukovat na prázdný, vyhodnocení dotazu končí **úspěchem**: znamená to, že formule

  ∃ (d1 & … & dn) je logickým důsledkem programu **P**.

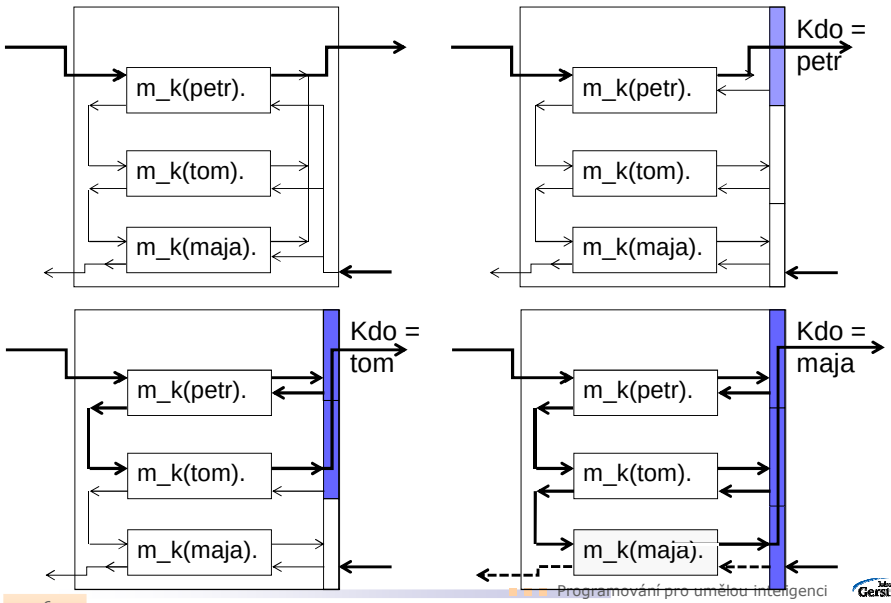
- Při postupném „zjednodušování“ výchozího dotazu je nutné postupovat systematicky. Postup definuje **strategie PROLOGu**

Krabičkový model výpočtu – predikát a jeho brány

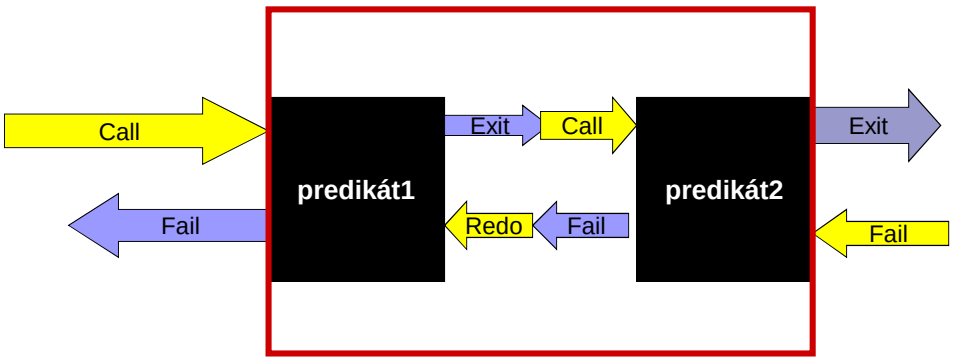


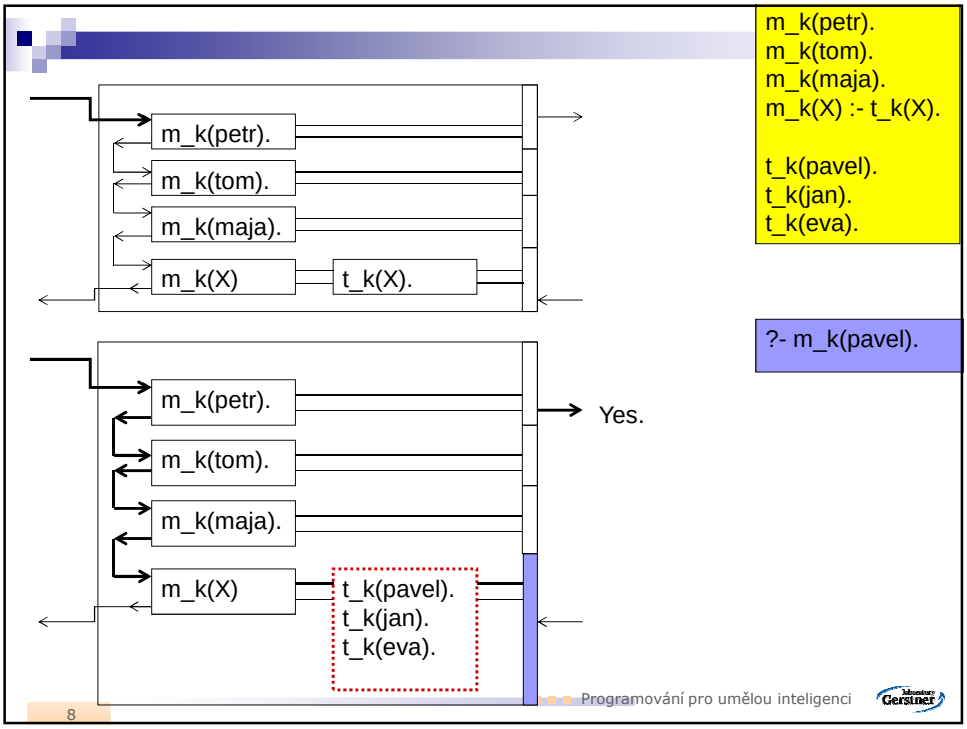
Model pro predikát obsahuje seznam všech klauzulí programu :

?- m_k(Kdo).



- **Krabičkový model výpočtu** – složený predikát





Zpracování log. programu (= vytváření derivačního grafu podle ***SLD resoluce***) obsahuje **2 zdroje nedeterminismu:**

- který podcíl rozvíjet? výběrové pravidlo – *„zleva-do-prava“*
- které pravidlo použít? *prohledávání derivačního grafu do hloubky*

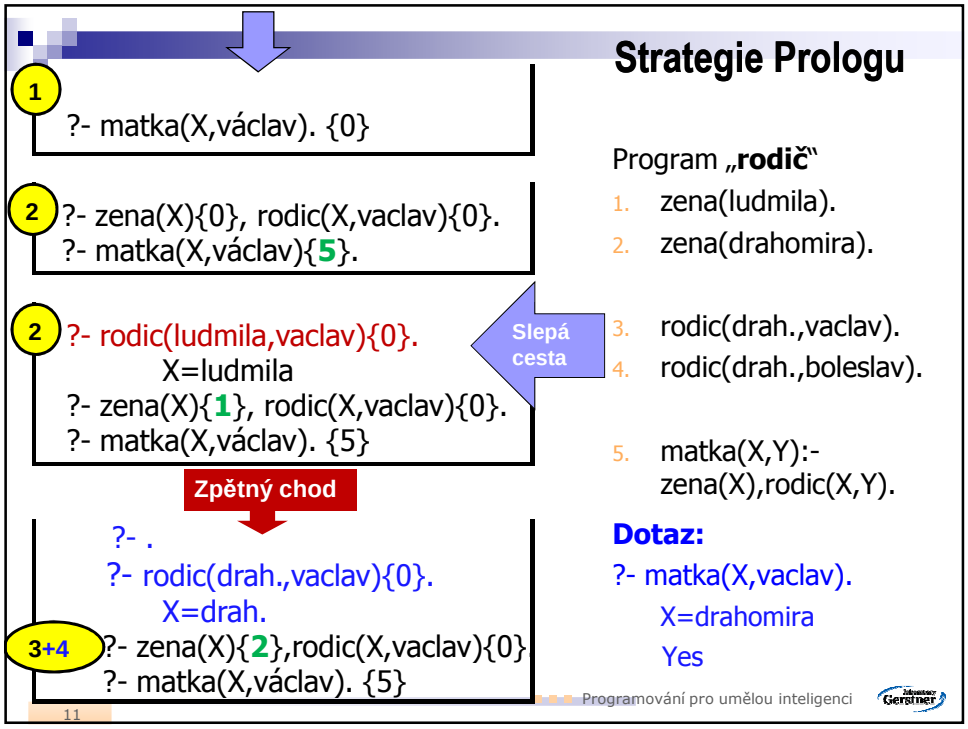
3 způsoby ukončení algoritmu při zpracování dotazu (odpovídají situaci, kdy není důvod/zdroje pokračovat):

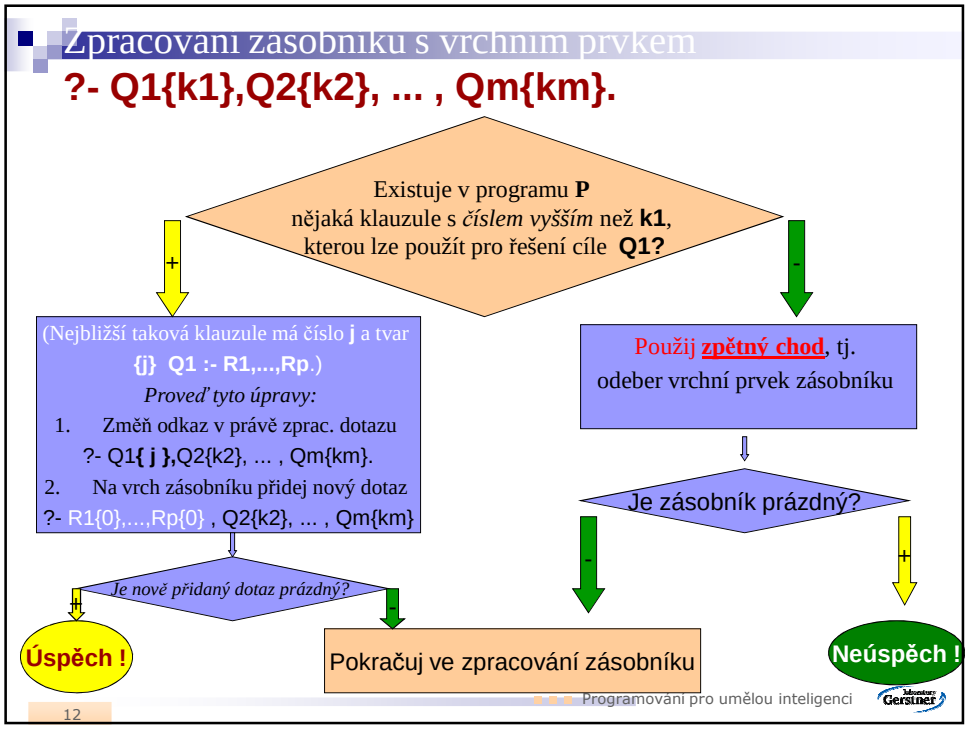
- dotaz byl redukován na prázdný, tj. výchozí dotaz byl splněn …**ÚSPĚCH (success)**
- všechny možnosti, jak řešit výchozí dotaz byly použity a selhaly … **KONEČNÉ SELHÁNÍ (finite failure)**,
- **neukončený výpočet** …

?-d1,…,dn.

- Pro vyhodnocování dotazu je vytvářen **zásobník** - aktuální verze dotazu je přidána na jeho vrch a s tou se dále pracuje:
  - Aktuální dotaz a jeho jednotlivé podcíle se zpracovávají **zleva-doprava** (nejdřív se zcela “vyřeší“ čili „odstraní“ **d1**, ...). Výsledkem je upřesněná informace o hodnotě proměnných z řešeného dotazu získaná pomocí **unifikace**.
  - Pro „zjednodušení“ podcíle **d1** (i dalších) se vždy postupně zkoušejí všechny klauzule z definice odpovídajícího predikátu (vychází se z jejich pevně daného **uspořádán**í).
- *Pro názornost budeme dále předpokládat, že každý atom dotazu je doplněn o informaci o **čísle poslední klauzule** dosud použité pro jeho řešení. Pro výchozí dotaz je to číslo* **0**:

?-d1{0},…,dn {0}.


Strategie Prologu
1
?- matka(X,václav). {0}
2
?- zena(X){0}, rodic(X,vaclav){0}.
?- matka(X,václav){5}.
2
?- rodic(ludmila,vaclav){0}.
X=ludmila
?- zena(X){1}, rodic(X,vaclav){0}.
?- matka(X,václav). {5}
Slepá cesta
Zpětný chod
?- .
?- rodic(drah.,vaclav){0}.
X=drah.
3+4
?- zena(X){2},rodic(X,vaclav){0}
?- matka(X,václav). {5}
Program „rodič"
1. zena(ludmila).
2. zena(drahomira).
3. rodic(drah.,vaclav).
4. rodic(drah.,boleslav).
5. matka(X,Y):- zena(X),rodic(X,Y).
Dotaz:
?- matka(X,vaclav).
X=drahomira
Yes
Programování pro umělou inteligenci
11


Zpracování zásobníku s vrchním prvkem
?- Q1{k1},Q2{k2}, ... , Qm{km}.
Existuje v programu P
nějaká klauzule s číslem vyšším než k1,
kterou lze použít pro řešení cíle Q1?
+
-
(Nejbližší taková klauzule má číslo j a tvar
{j} Q1 :- R1,...,Rp.)
Proveď tyto úpravy:
1. Změň odkaz v právě zprac. dotazu
?- Q1{ j },Q2{k2}, ... , Qm{km}.
2. Na vrch zásobníku přidej nový dotaz
?- R1{0},...,Rp{0} , Q2{k2}, ... , Qm{km}
Použij zpětný chod, tj.
odeber vrchní prvek zásobníku
Je zásobník prázdný?
Je nově přidaný dotaz prázdný?
Úspěch !
Pokračuj ve zpracování zásobníku
Neúspěch !
Programování pro umělou inteligenci
12

Rekurzivní pravidla

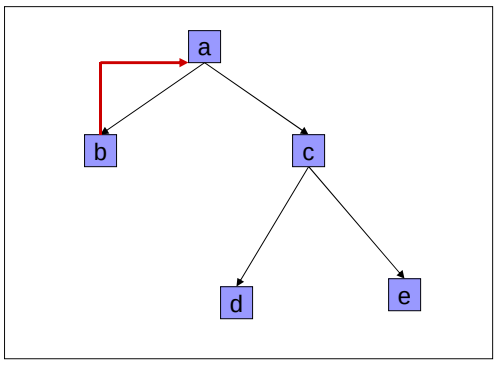
- Nerekurzivní definice:

  dědeček(U,Z):-muž(U), rodič(U,V), rodič(V,Z).
  pradědeček(A,B):-muž(A), rodič(A,C), rodič(C,D), rodič(D,B).
  rod_linie(P,PP):-rodič(P,PP).
  rod_linie(P,PP):-dědeček(P,PP).
  rod_linie(P,PP):-pradědeček(P,PP)...

- **Rekurzivní definice**:

  rod_linie(P,PP):-rodič(P,PP).
  rod_linie(P,PP):-rodič(P,P1),rod_linie(P1,PP).

g1



Jaké odpovědi dávají různé varianty programu?

Log.program g0 Dotaz	verze1	verze2	verze3	verze4
?-c_i(a,e).	Yes;no	Yes;abort	Yes;no	abort
?-c_i (b,e).	no	abort	no	abort

```
hrana(a,b).  hrana(a,c). hrana(c,d).   hrana(c,e).
c1(Y,Y).      c1(Z,K):-hrana(Z,M),c1(M,K).
  c2(Y,Y).   c2(Z,K):-c2(Z,M),hrana(M,K).
c3(Z,K):-hrana(Z,M),c3(M,K).  c3(Y,Y).
  c4(Z,K):-c4(Z,M),hrana(M,K). c4(Y,Y).
          ?- c1(a,e).
                    ?-c2(a,e).
                              ?-c3(a,e).
                                        ?-c4(a,e).
```


Jaké odpovědi dávají různé varianty programu?

```
hrana(a,b).  hrana(a,c). hrana(c,d).
hrana(c,e).  hrana(b,a).
c1(Y,Y).      c1(Z,K):-hrana(Z,M),c1(M,K).
  c2(Y,Y).   c2(Z,K):-c2(Z,M),hrana(M,K).
c3(Z,K):-hrana(Z,M),c3(M,K).  c3(Y,Y).
  c4(Z,K):-c4(Z,M),hrana(M,K). c4(Y,Y).
?- c1(a,e).
```

Po přidání hrany **h(b,a).**

Log.program g1 Dotaz	verze1	verze2	verze3	verze4
?-c_i (a,e).	abort	Yes;yes,...	abort	abort
?-c_i (b,e).	abort	Yes;yes,...	abort	abort

Jaké odpovědi dávají různé varianty programu?

Log.program g0 Dotaz	verze1	verze2	verze3	verze4
?-c_i (a,e).	Yes;no	Yes;abort	Yes;no	abort
?-c_i (b,e).	no	abort	no	abort

Po přidání hrany **h(b,a).**

Log.program g1 Dotaz	verze1	verze2	verze3	verze4
?-c_i (a,e).	abort	Yes;yes; …	abort	abort
?-c_i (b,e).	abort	Yes;yes; …	abort	abort

Způsoby ukončení logického programu

- Zpracování log. programu = vytváření derivačního grafu podle ***SLD resoluce*** – používá prohledávání do hloubky
  - cíl splněn – **Yes + substituce za proměnné**
  - cíl selhal – **No**
  - **neukončený výpočet** – např. zacyklení

Pravidla pro sestavení programu v Prologu

- **Postupujeme od jednoduššího ke složitějšímu:**
  - FAKTA
  - PRAVIDLA bez rekurze
  - PRAVIDLA s rekurzí - rekurzi na konec pravidla

- **Operace charakteristické pro Prolog:**
  - Srovnávání (matching)
  - Unifikace
  - Zpětný chod

Seznam a jak jej obratit

- s využitím predikátu **spoj(L1,L2, Vysledek)**

```
spoj([],S1,S1).
spoj([H|T],S1,[H|S]):-spoj(T,S1,S).
    o1([],[]).
    o1([X|L],S):-o1(L,L1),spoj(L1,[X],S).
```

- S využitím pomocné proměnné (akumulátor)

```
o2([],L,L).
o2([H|T],M,V):- o2(T,[H|M],V).
```

Srovnejte délku výpočtu obou řešení!