

PROLOG

strategie vyhodnocení dotazu

1

Program v PROLOGu

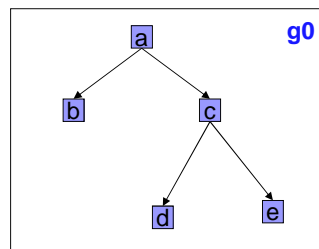
- Konečná **uspořádaná** množina faktů a pravidel (klauzulí), která se týkají téhož predikátu (t.j. predikátu uvedeného jako hlava pravidla), se nazývá **definice predikátu**.
- **Program** tvoří množina definic všech použitých predikátů.
- *Pro názornost budeme dále předpokládat, že **klauzule v programu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

- Jak popíšeme vztahy v grafu **g0**?

Použijme označení

hrana(Od,Do)

cesta(Odkud,Kam)



2

Programování pro umělou inteligenci



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).`

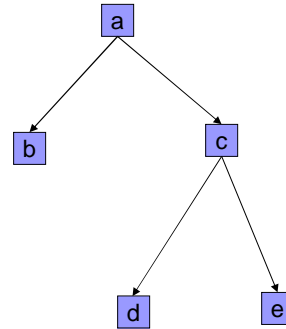
`c3(Y,Y).`

`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?



3

Programování pro umělou inteligenci



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*

$\exists (d1 \ \& \ \dots \ \& \ dn)$ je logickým důsledkem programu **P**.

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

4

Programování pro umělou inteligenci



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 ...

5

Programování pro umělou inteligenci



?-d1,...,dn.

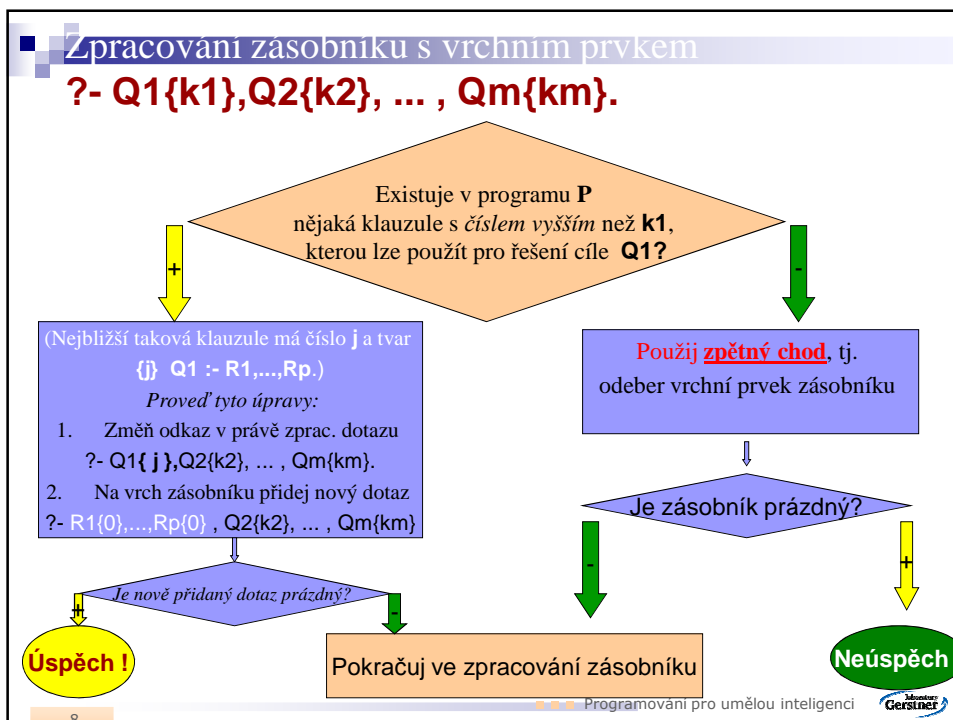
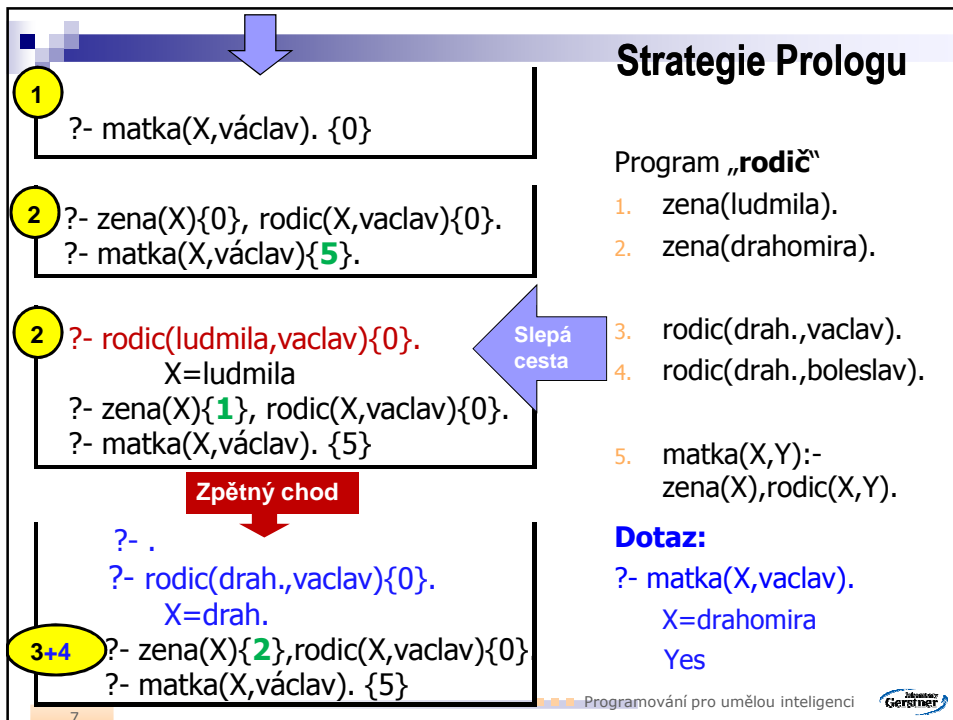
- Pro vyhodnocování dotazu je vytvářen **zásobník** - aktuální verze dotazu je přidána na jeho vrch a s tou se dále pracuje:
 - Aktuální dotaz a jeho jednotlivé podcíle se zpracovávají **zleva-doprava** (nejdřív se zcela „vyřeší“ čili „odstraní“ **d1**, ...). Výsledkem je upřesněná informace o hodnotě 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 **uspořádání**).
- *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}.

6

Programování pro umělou inteligenci





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:

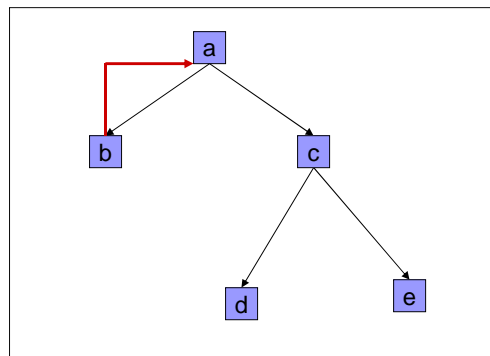
```
rod_linie(P,PP):-rodič(P,PP).  
rod_linie(P,PP):-rodič(P,P1),rod_linie(P1,PP).
```

9

Programování pro umělou inteligenci



g1



10

Programování pro umělou inteligenci



Jaké odpovědi dávají různé varianty programu?

Log.program g0	verze1	verze2	verze3	verze4
Dotaz				
?-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).

11

Programování pro umělou inteligenci



Jaké odpovědi dávají různé varianty programu?

hrana(a,b). hrana(a,c). hrana(c,d).

hrana(c,e). hrana(b,a).

Po přidání hrany
h(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).

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

12

Programování pro umělou inteligenci



Jaké odpovědi dávají různé varianty programu?

Log.program g ₀	verze1	verze2	verze3	verze4
Dotaz				
?-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 g ₁	verze1	verze2	verze3	verze4
Dotaz				
?-c _i (a,e).	abort	Yes;yes; ...	abort	abort
?-c _i (b,e).	abort	Yes;yes; ...	abort	abort

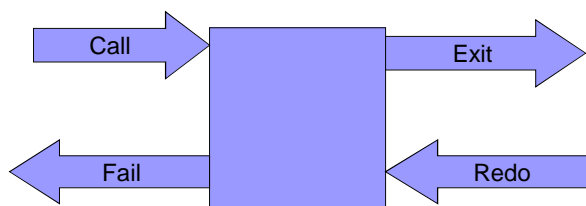
13

Programování pro umělou inteligenci



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
- **Krabičkový model výpočtu** – predikát a jeho **brány**

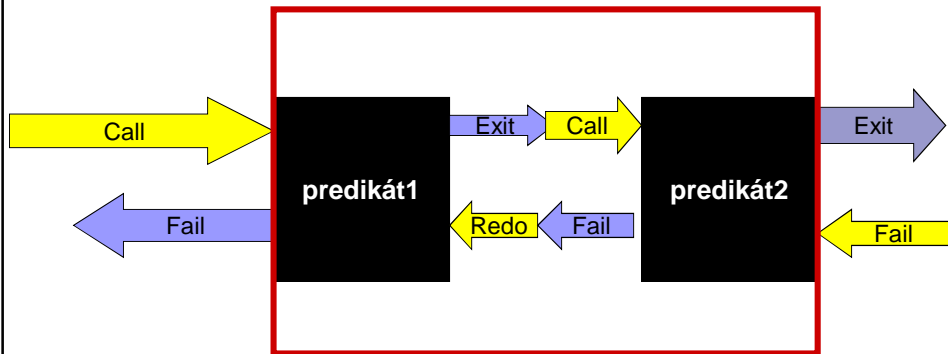


14

Programování pro umělou inteligenci



■ **Krabičkový model výpočtu – složený predikát**



15

Programování pro umělou inteligenci



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

16

Programování pro umělou inteligenci



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!