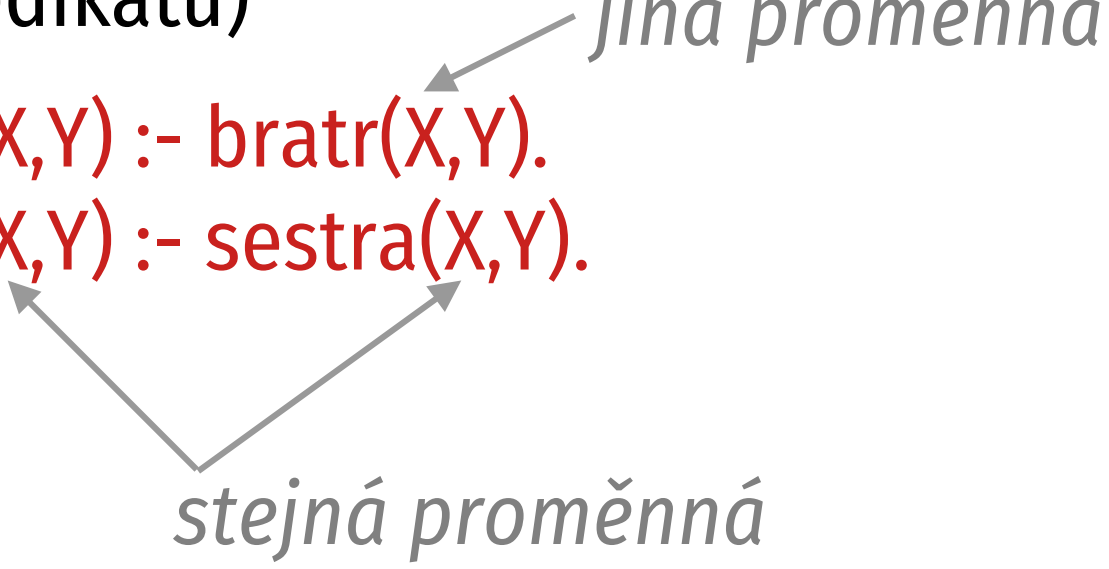
Proměnná

- začíná velkým písmenem
- **v pravidlech a faktech** – popisuje účastníky vztahu
- **v dotazech** – reprezentuje hledané objekty
- **volná proměnná** – o hodnotě zatím nic nevíme, interpret se ji snaží najít
- **vázaná proměnná** – z dřívějších kroků při řešení vyplývá její hodnota, je svázána s určitou hodnotou

Rozsah platnosti proměnné

- rozsahem platnosti proměnné je jen **jedna klauzule**
- stejnojmenná proměnná v sousední klauzuli nemá s touto nic společného (i když jsou součástí stejného predikátu)
- sourozenec(X,Y) :- bratr(X,Y).
  sourozenec(X,Y) :- sestra(X,Y).



Hodnota proměnné

- **hodnotu získává srovnáváním** při aplikaci klauzule
- „abychom toto mohli použít, proměnná *X* musí mít hodnotu *h*“
- jakmile byla přiřazena, hodnota se dále nemění
- změní se jen odvoláním použité klauzule a nalezením jiné (nebo stejné za jiných podmínek)

Příklad

```
rodic(jana,petr).
rodic(otto,eva).
dite(X,Y) :- rodic(Y,X).
```

```
?- rodic(X,petr).
```
najde X=jana

```
;
```
nenajde nic

```
?- dite(X,jana).
```
najde pravidlo, dosadí Y=jana,
hledá rodic(jana,X)
najde X=petr

```
;
```
nenajde nic

Anonymní proměnná

- zapisuje se jako podtržítko
- hodnota nás nezajímá, Prolog ji nezobrazuje ve výsledcích
- používá se v pravidlech
- příklad: predikát, zda X má dítě
  ma_dite(X) :- rodic(X,_).
- lze i několik anonymních, každá je nezávislá
  je_mezi(X) :- rodic(_,X), rodic(X,_).

Plnění cílů (1)

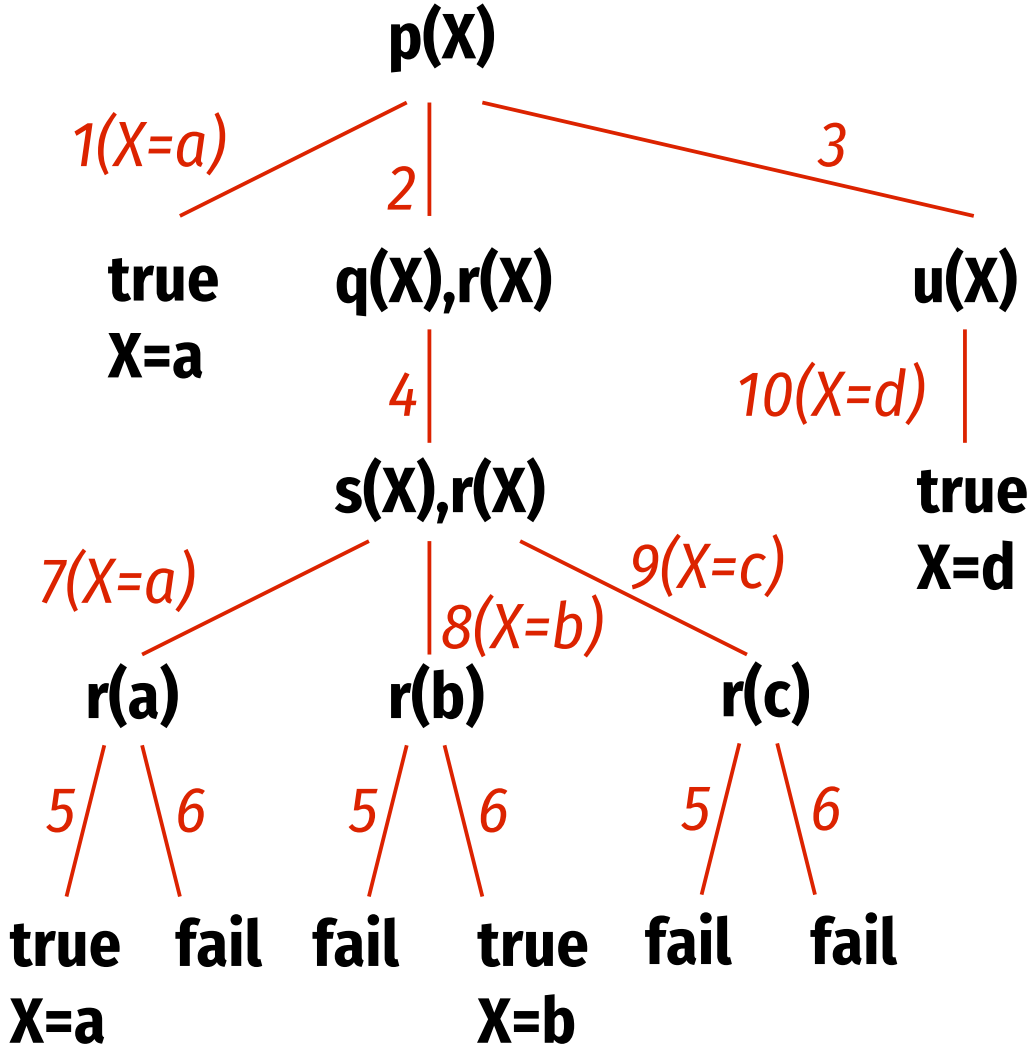
- program se prochází shora dolů a hledá se
  - fakt odpovídající prvnímu cíli nebo
  - pravidlo, jehož hlavička odpovídá prvnímu cíli
- **srovnávání (unifikace)** – odpovídá klauzule cíli?
  - stejné jméno a arita predikátu
  - konstanta vůči konstantě – musí být shodné
  - proměnná vůči konstantě – vytvoří se vazba
  - proměnná vůči proměnné – zůstane volná

Plnění cílů (2)

- podaří-li se srovnat s faktem
  - pokud byl poslední cíl, ohlásí úspěch (a vytvořené vazby proměnných z dotazu)
  - jinak pokračuje dalším cílem, **znovu od začátku programu**, ale s vytvořenými vazbami
- podaří-li se srovnat s pravidlem, tělo nahradí 1. cíl
- jinak po dosažení konce programu odvolá poslední použitou klauzuli (a jí vytvořené vazby) a **pokračuje v hledání další vhodné klauzule** pro daný cíl

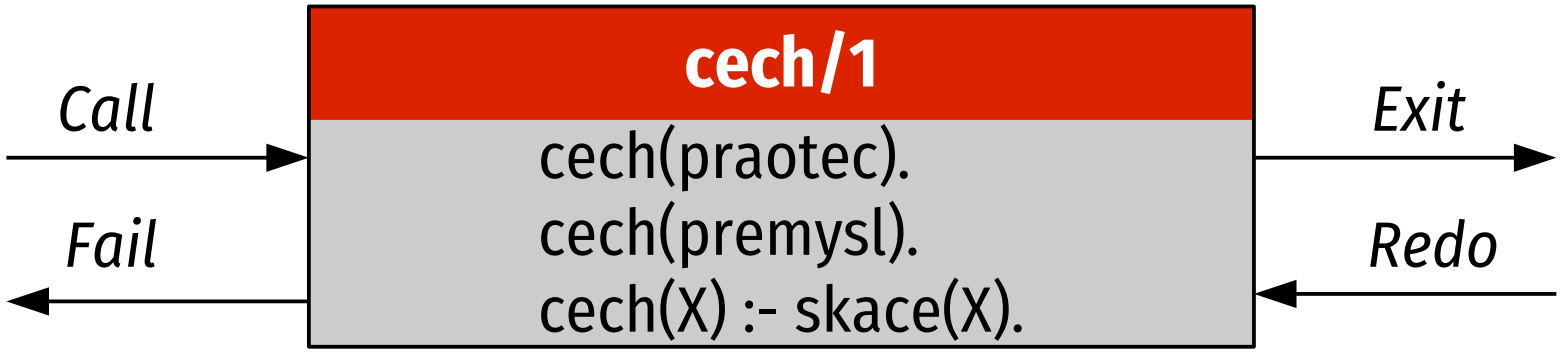
Derivační strom

```
p(a).                   /*1*/
p(X) :- q(X), r(X).     /*2*/
p(X) :- u(X).           /*3*/

q(X) :- s(X).           /*4*/

r(a).                   /*5*/
r(b).                   /*6*/

s(a).                   /*7*/
s(b).                   /*8*/
s(c).                   /*9*/

u(d).                   /*10*/
```

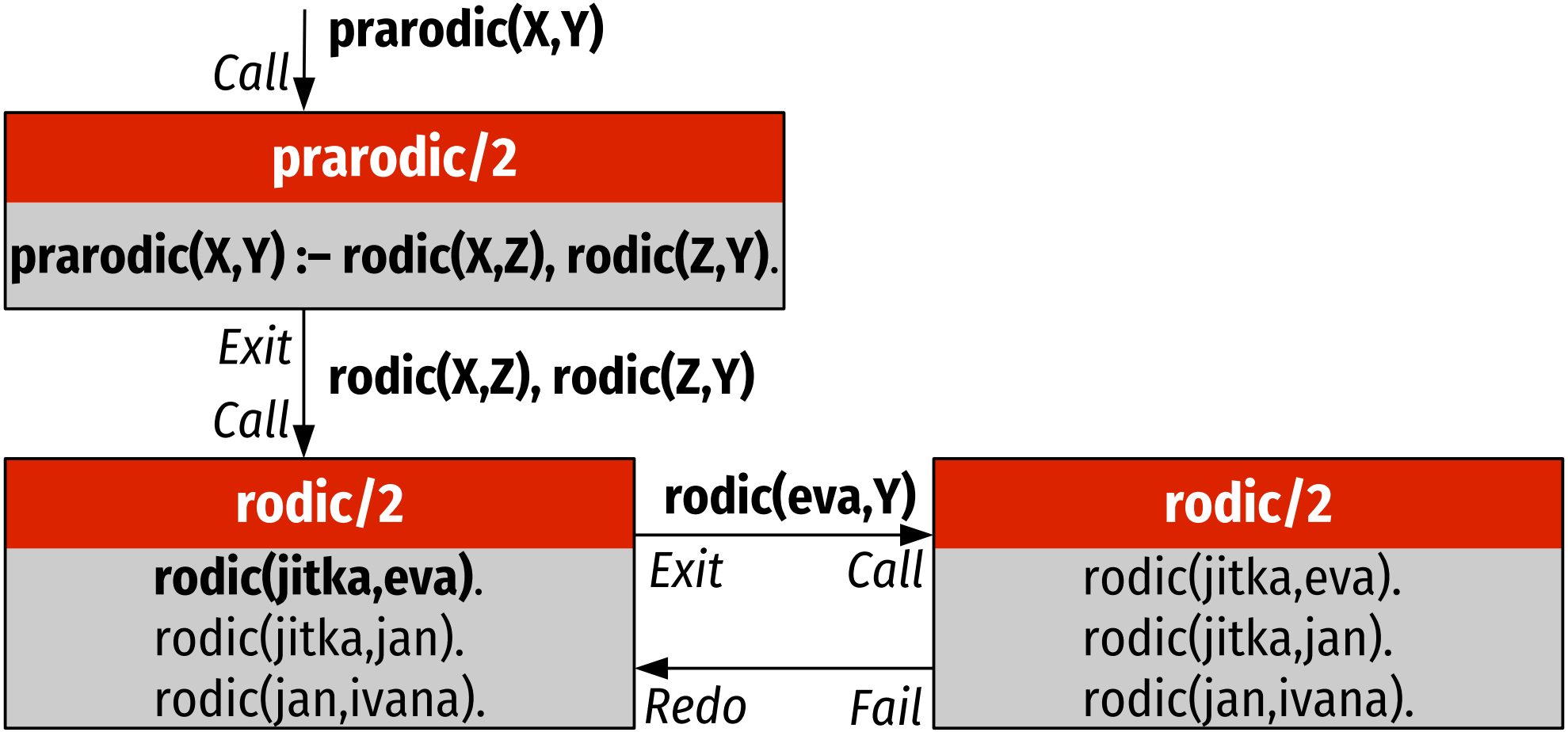


Blokový model výpočtu

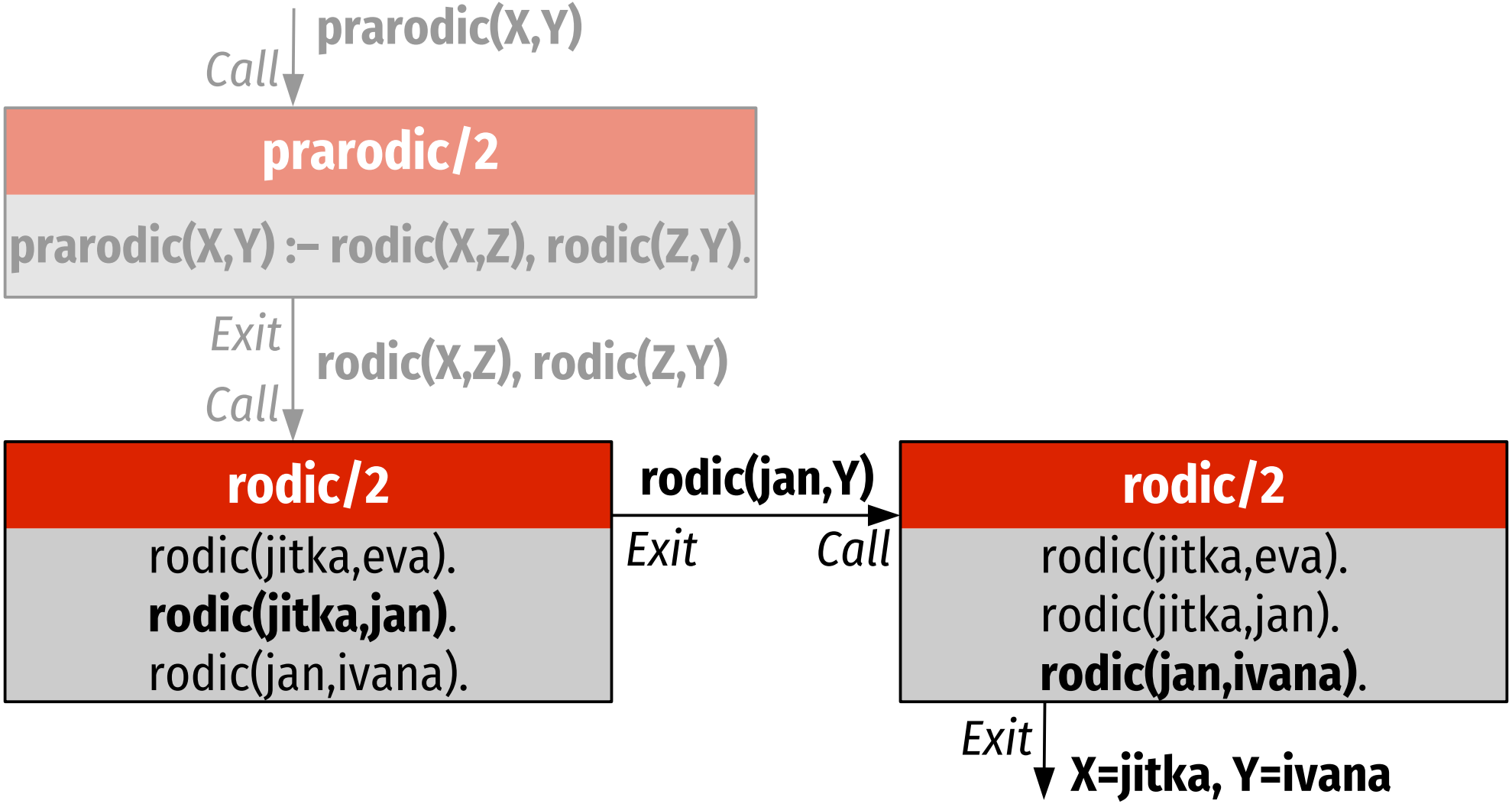
- volání procedury reprezentováno blokem se 4 porty
  - **Call** – první volání procedury
  - **Exit** – úspěch, podařilo se odvodit
  - **Redo** – hledá se další varianta (procedura byla úspěšná, ale selhalo pokračování)
  - **Fail** - neúspěch



Příklad (1)


prarodic(X,Y)
Call
prarodic/2
prarodic(X,Y) :– rodic(X,Z), rodic(Z,Y).
Exit
Call
rodic(X,Z), rodic(Z,Y)
rodic/2
rodic(jitka,eva).
rodic(jitka,jan).
rodic(jan,ivana).
rodic(eva,Y)
Exit
Call
Redo
Fail
rodic/2
rodic(jitka,eva).
rodic(jitka,jan).
rodic(jan,ivana).

Příklad (2)


prarodic(X,Y)
Call
prarodic/2
prarodic(X,Y) :– rodic(X,Z), rodic(Z,Y).
Exit
rodic(X,Z), rodic(Z,Y)
Call
rodic/2
rodic(jitka,eva).
rodic(jitka,jan).
rodic(jan,ivana).
rodic(jan,Y)
Exit
Call
rodic/2
rodic(jitka,eva).
rodic(jitka,jan).
rodic(jan,ivana).
Exit
X=jitka, Y=ivana

Backtracking

- postup řešení odpovídá průchodu derivačním stromem do hloubky
- zkusí první možnou variantu a pokusí se ji rozvinout až po splnění cíle
- pokud neuspěje nebo je požadováno další řešení, vrátí se a zkusí další variantu
- používán v umělé inteligenci k prohledávání stavových prostorů

Návrh predikátů

- otázka arity a významu argumentů
- jak vyjádřit, že adam je muž?
- adam_je_muz. – nic neumožňuje, nevhodné
- muz(adam). – krátké a elegantní; lze testovat, zda je muž; neumožňuje dotaz „Jakého pohlaví je adam?“
- pohlavi(adam,muz). – lze dotazy oběma směry, ale dost povídavé; lze získat z předchozího: pohlavi(X,muz) :- muz(X).

Zpracování pravdivosti

- logické konstanty – true, fail
- **konjunkce** – čárka (,)
  - pokud některá část selže, další se nevyhodnocují
- **disjunkce** – středník (;)
  - raz(X) :- dva(X); tri(X).
    je totéž co
    raz(X) :- dva(X).
    raz(X) :- tri(X).

Negace

- \+(X) splněna, pokud X nelze dokázat
- lze psát i jako not(X), ale \+ je tradiční
- rezoluční mechanismus zkouší odvodit X – pokud uspěje, vrátí se portem fail
- **nedá se použít pro volnou proměnnou** – vždy selže (najde hodnotu, pro kterou výrok platí, a zneguje)
- nehledá alternativy, redo vede rovnou k fail
- patří ke konci predikátů, kdy proměnné jsou vázány

Negace – důsledky

- např. nebude fungovat
  sestra(X, Y) :- \+(muz(X)), sourozenec(X, Y).
  lépe
  sestra(X, Y) :- sourozenec(X, Y), \+(muz(X)).
- \+(\+(X)) není totéž co X
  ?- muz(X).
  X=petr; X=otto
  ?- \+(\+(muz(X))).
  true

Rekurze

- stejný predikát v těle i hlavičce pravidla
- často používána, nahrazuje cykly
- predek(X,Y) :- rodic(X,Y).
  predek(X,Y) :- rodic(X,Z), predek(Z,Y).
- pozor na pořadí
  - prohození klauzulí by zhoršilo efektivitu
  - predek(X,Y) :- predek(X,Z), rodic(Z,Y).
    by vedlo k nekonečnému cyklu

Čísla v Prologu

- celá čísla
  - neomezená velikost
- racionální čísla
  - zlomek dvou celých čísel
  - reprezentována jako N rdiv M
  - kanonický tvar: M>0, N a M nesoudělná
- reálná (v plovoucí desetinné čárce)
  - zpravidla 64bitová

Výpočty

- predikát **is**:
  Číslo **is** *Výraz*
  vyhodnotí *Výraz* a vydá true, pokud je *Číslo* úspěšně unifikováno s výsledkem
- ***Výraz* nesmí obsahovat volné proměnné**
- obsah_kruhu(Polomer,Obsah) :-
  Obsah is 3.14 * Polomer * Polomer.
- místo funkce predikát s jedním argumentem navíc, který popisuje vzájemný vztah argumentů

Aritmetické operace

- +, –, * v obvyklých významech
- / typ výsledku závisí na typech operandů a vlastním výsledku – např. 4 / 2 vydá 2 (celé číslo)
- // celočíselné dělení ... truncate(X/Y)
- mod zbytek po celočíselném dělení
- rdiv racionální dělení

Konverzní funkce

- round(X), integer(X) zaokrouhlí na nejbližší celé
  floor(X) zaokrouhlí dolů
  ceiling(X), ceil(X) zaokrouhlí nahoru
  truncate(X) odřízne desetinnou část
- float(X) převede na číslo v plovoucí čárce
- rational(X) převede na racionální číslo (přesné)
  rationalize(X) podobné rational, ale zaokrouhluje
- float_fractional_part(X) přístup k částem reál. čísla
  float_integer_part(X)

Aritmetické funkce

- nejvýznamnější dostupné funkce:
- abs(X) absolutní hodnota
- max(X,Y), min(X,Y) maximum/minimum
- random(N) náhodné číslo od 0 do N–1
- log(X), log10(X) přirozený a dekadický logar.
- exp(X), X ** Y e^X, X^Y
- sin(X), cos(X),... goniometrické funkce

Porovnání čísel

- X =:= Y rovná se X a Y?
  není srovnání (unifikace)
  výrazy vyhodnotí, ale vyžaduje hodnoty
- X =\= Y liší se X od Y?
- X < Y
- X =< Y
- X > Y
- X >= Y

Porovnání vs srovnání

- **=:=** **porovnání**
  - jsou stejná čísla?
- **=** **srovnání**
  - jsou stejné struktury?
  - stejné predikáty, srovnatelné argumenty
- **\=** **nesrovnatelné**
  - negace srovnání

Obecné porovnání

- ==
  jsou obě strany identické?
  porovnává i atomy, nejen čísla
- \==
  jsou rozdílné?
- nesrovnává
  X == 2 je fail
  X == X je true (i pro volnou proměnnou)

Seznam

- uzavřen do hranatých závorek, položky oddělovány čárkami
- [raz, dva, tri] []
- uspořádaný – [a, b] se liší od [b, a]
- lze vnořovat
- nemusí být homogenní
- [muz(adam), [raz, dva], [3, [4], 5, 6]]
  seznam se třemi prvky

Přístup k částem seznamu

- seznam lze vyjádřit jako dvojici
  [*hlava* | *ocas*]
  hlava je první prvek seznamu,
  ocas seznam ostatních (vše kromě prvního)
- srovnání [X | Y] = [a, b, c, d] vytvoří vazby
  X = a
  Y = [b, c, d]
- pozor na rozdíl mezi [X, Y] (dvouprvkový seznam)
  a [X | Y] (libovolný neprázdný seznam)

Test seznamu

- seznam je prázdný nebo se skládá z hlavy a ocasu, který je seznamem:
- je_seznam([]).
  je_seznam([_|Ocas]) :- je_seznam(Ocas).
- existuje standardní predikát is_list(X), který testuje, zda je X seznam

Přidávání prvků

- Stary = [a, b, c],
  Novy = [x | Stary]
  vytvoří vazbu Novy = [x, a, b, c]
- lze i pro několik prvků současně:
  Novy = [x, y, z | Stary] vede k
  Novy = [x, y, z, a, b, c]
- podobně lze odebírat hodnoty ze začátku
  [A, B | Zbytek] = [1, 2, 3, 4, 5]
  A=1, B=2, Zbytek=[3, 4, 5]

Zpracování seznamu

- realizováno rekurzivně
- příklad: výpis prvků seznamu

```
vypis_seznam([]).
vypis_seznam([Hlava|Ocas]):-
    write(Hlava), nl, vypis_seznam(Ocas).
```

- výpis prázdného seznamu je prázdný
  jinak vypíše hlavu, odřádkuje a volá se rekurzivně pro ocas

Průchod seznamem obecně

- co s triviálním seznamem (prázdným, jednoprvkovým)
  - jak je pro něj definován výsledek?
  - poskytuje výchozí bod pro rekurzi
- netriviální seznam
  - zpracovat první prvek a zahrnout do výsledku
  - zavolat rekurzivně sám sebe na zbytek seznamu

vytvořeno s podporou
projektu ESF