Logic programming

- *logic program*: a finite set of Horn clauses
- inference (interpretation) based on SLD-resolution
- declarativness: the specification of a program is equal to the program

popel@fi.muni.cz

Prolog

- implementation of a logic programming language
- strategy: depth-first search in the SLD-tree
- historie: in 70th. – Colmerauer, Kowalski; D.H.D. Warren (WAM)

Syntax of Prolog I

Data structures

- terms (constants, variables, compound terms)
- constants:
  - `0, 123, -12, 1.0, 4.5E7, -0.12e+8`,
  - atoms (`'Bob Kowalski', [], s1, ==, 'beaver', atom`)
- variables (`N, _VYSLEDEK, Hodnota, A1, _12`),

  anonymous variable (`_`)
- compound terms: functor(name, arity), arguments

  ```
  point(X,Y,Z),
  tree(Value,tree(LV,LL,LR),tree(RV,RL,RR))
  ```

Syntax of Prolog II

Program

- an ordered set of program clauses (pravidla, fakta)
- variables local in a clause
- rule: head, body

  `date(D,M,Y):- day(D), month(M), year(R).`
- fact: a rule with an empty body (body = `true`)

  `date(14,'February',2001).`
- a goal: `?- date(29,'January',2001).`

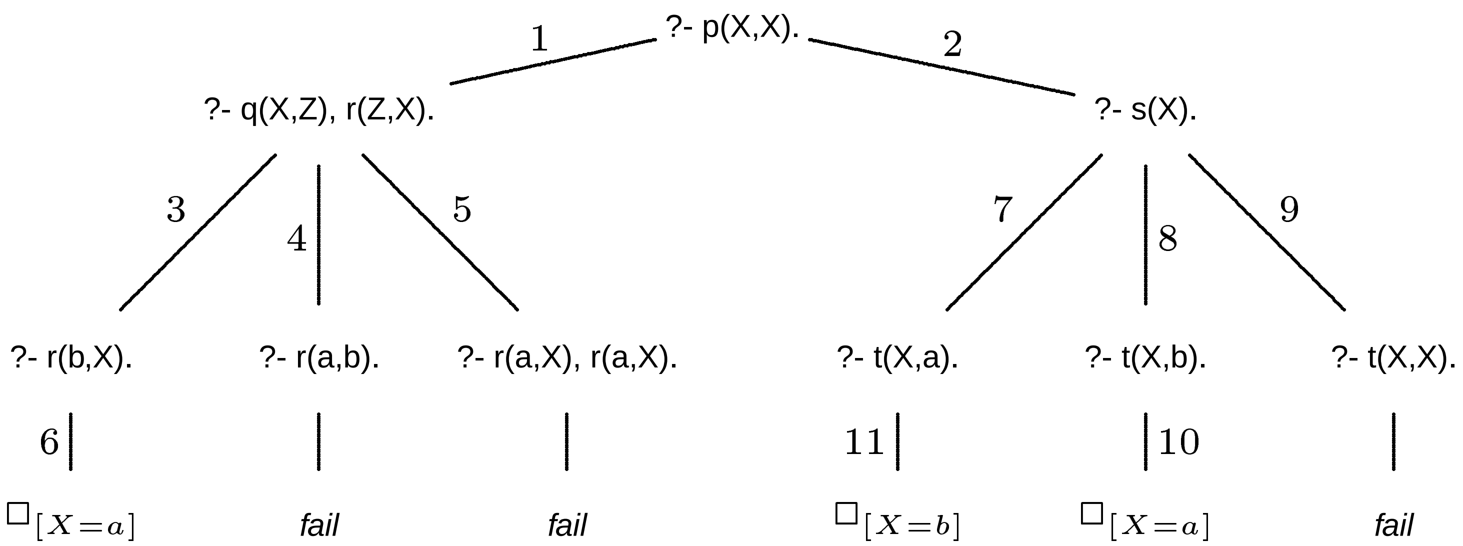
Explicit unification: = operator

Ex.: `X=Y, f(g(a,X))=f(Y)`

SLD-tree for a Prolog program

1. p(X,Y) :- q(X,Z), r(Z,Y).
2. p(X,X) :- s(X).
3. q(X,b).
4. q(b,a).
5. q(X,a) :- r(a,X).
6. r(b,a).
7. s(X) :- t(X,a).
8. s(X) :- t(X,b).
9. s(X) :- t(X,X).
10. t(a,b).
11. t(b,a).

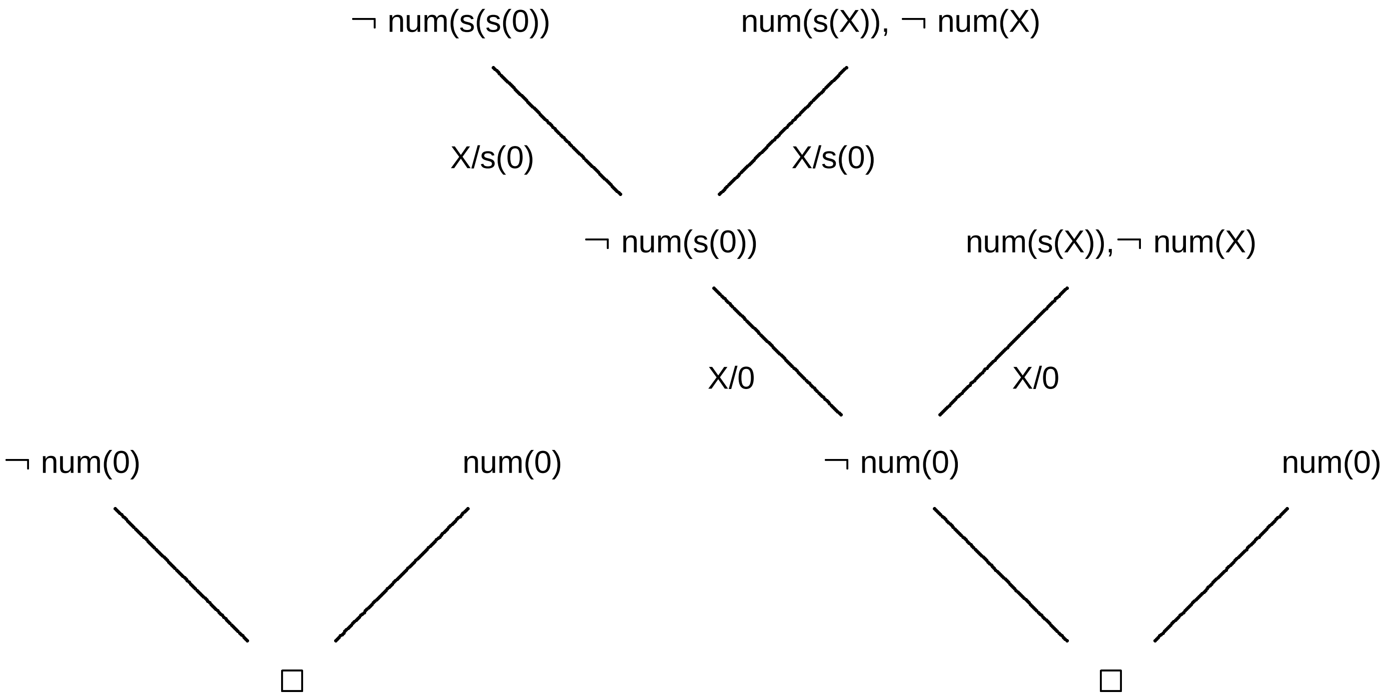
?- p(X,X).



SLD-resolution for a Prolog program

Ex.:

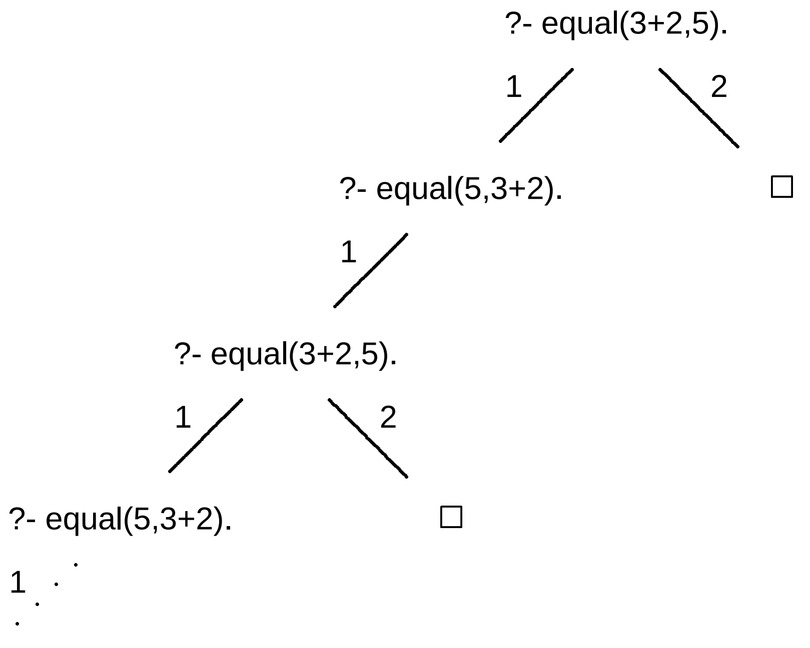
```
num(0).                       ?- num(0).
num(s(X)):- num(X).           ?- num(s(s(0))).
```



Example: Incompleteness

1. `equal(X,Y):- equal(Y,X).`
2. `equal(3+2,5).`

```
?- equal(3+2,5).
```



List

- recursive data structure, ordered
- functor `./2`; prázdný seznam `[]`
- `.(Head,Tail)`, the notation used: `[Head|Tail]`, `Tail` is a list

```
.(a,[])                [a]            [a|[]]

.(a,.(b,.(c,[])))      [a,b,c]        [a,b|[c]]
                       [a|[b,c]]      [a,b,c|[]]
                       [a|[b,c|[]]]   [a|[b|[c|[]]]]
```

- can be represented as a tree

member/2 I

1) unification:

```
member(X,[X|_]).
member(X,[_|T]):- member(X,T).

?- member(a,[b,c,a]).
yes
?- member(a,[X,b,c]).
X=a
yes
```

member/2 II

2) identity:

```
member(X,[Y|_]):- X == Y.
member(X,[_|T]):- member(X,T).
?- member(a,[X,b,c]). % No
?- member(a,[a,b,a]),write(ok),nl,fail.
ok
ok
No
```

3) without a multiple occurence:

```
member(X,[Y|_]):- X == Y.
member(X,[Y|T]):- X \== Y, member(X,T).
?- member(a,[a,b,a]),write(ok),nl,fail.
ok
No
```

Example: Append two lists

```
append([],L,L).
append([H|T1],L2,[H|T]):- append(T1,L2,T).
------------------------------------------
?- append([a,b],[c,d],L).
L = [a, b, c, d]
Yes
?- append(X,[c,d],[a,b,c,d]).
X = [a, b]
Yes
?- append(X,Y,[a,b,c]).
X = []         Y = [a, b, c];
X = [a]        Y = [b, c];
X = [a, b]     Y = [c];
X = [a, b, c]  Y = [];
No
```

reverse/2

```
reverse([],[]).
reverse([H|T],L):- reverse(T,L1), append(L1,[H],L).
-------------------------------------------------------
?- reverse([a,b,c],L).
L = [c, b, a]
Yes
?- reverse([a,b,c],[c,b,a]).
Yes
?- reverse(L,[a,b,c]).
L = [c, b, a]
Yes
```

delete, permutation, prefix, postfix, sublist . . .

Sort

0) naive sort: generate and test

```
naive_sort(L,S):- perm(L,S), sorted(S).

sorted([]).
sorted([_]).
sorted([X,Y|T]):- X=<Y, sorted([Y|T]).
```

1) insertion sort

```
insert(X,[],[X]).
insert(X,[Y|T1],[Y|T2]):- X>Y, insert(X,T1,T2).
insert(X,[Y|T1],[X,Y|T1]):- X=<Y.

isort([],[]).
isort([X|L],S):- isort(L,Y), insert(X,Y,S).
```

Example: Quick sort

divide et concera

```
qsort([],[]).
qsort([H],[H]).
qsort([H|T],L):-
  divide(H,T,M,V),
  qsort(M,M1),
  qsort(V,V1),
  append(M1,[H|V1],L).

divide(_,[],[],[]).
divide(H,[K|T],[K|M],V):- K=<H, divide(H,T,M,V).
divide(H,[K|T],M,[K|V]):- K>H, divide(H,T,M,V).
```

Arithmetics

- + - * / mod ...
- is

```
?- A is 3*(4+2).
A=18
Yes
?- A is 3*(B+2).
Error
```

- < > >= =<

```
?- 3*4 > 2.
Yes
?- B =< 14.
Error
```

Example: symbolic derivation

```
d(x, 1).
d(N, 0)                  :- number(N).
d(-X, -A)                :- d(X,A).
d(X + Y, A + B)          :- d(X,A), d(Y,B).
d(X - Y, A - B)          :- d(X,A), d(Y,B).
d(X * Y, A*Y + B*X):- d(X,A), d(Y,B).
d(X / Y, (A*Y - X*B) / Y^2):- d(X,A), d(Y,B).
d(X ^ N, N*X^M * C):- number(N), M is N-1, d(X, C).
d(F^G, F^G*(B*log(F) + A*G/F)) :- d(G, B), d(F, A).
d(log(X), 1/X * Y) :- d(X,Y).
d(exp(X), exp(X)*Y):- d(X,Y).
d(sin(X), cos(X)*Y):- d(X,Y).
d(cos(X), -sin(X)*Y):- d(X,Y).
d(arctg(X), 1/(1+X^2)*Y):- d(X,Y).
```

Programming in Prolog

- load-consult a program:

```
consult('program.pl').
['program.pl'].
['program.pl',program2].
reconsult(program).
```

- printa the program:

```
listing.
```

- finish:

```
halt.
```

Prolog at FI

- SICStus Prolog (module add sicstus; sicstus)
- SWI Prolog `http://www.swi-prolog.org/` – free
- yap `http://www.dcc.fc.up.pt/ vsc/Yap/` – free

- MS-Windows: SWI, yap, sicstus