CS206 Lecture 24

SLD-Resolution for Prolog

01005010,01005015,01D05003,01D05008
CSE Department.
IIT Bombay

Plan for Lecture 24

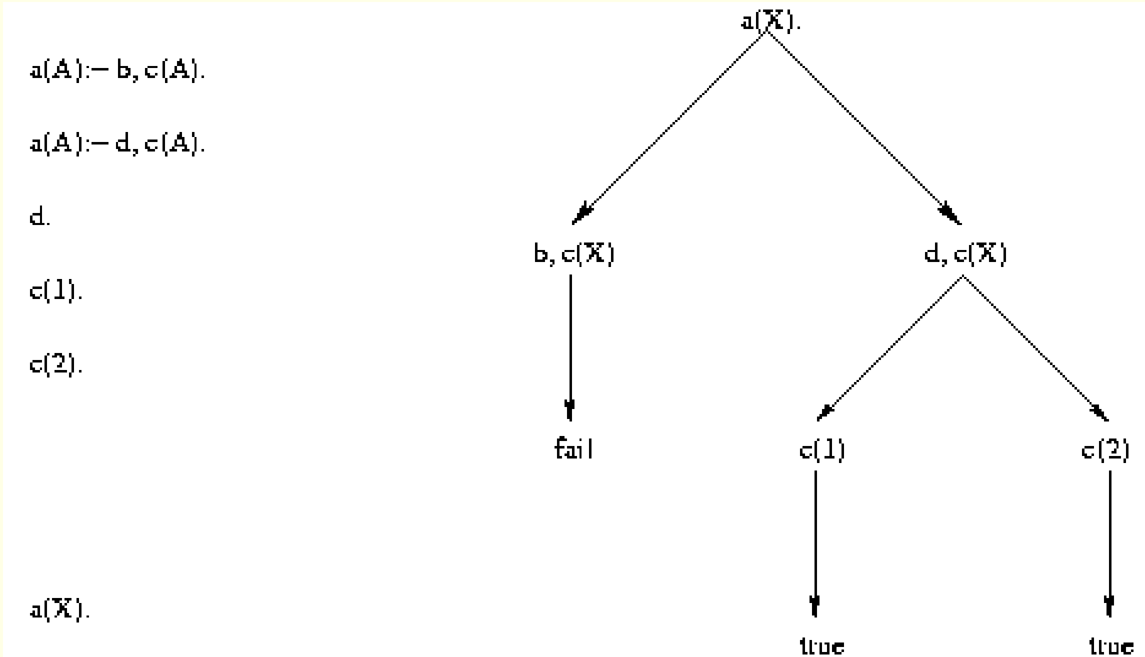
- SLD Trees
- Examples

SLD Trees

In prolog for each inquiry a so-called SLD tree from the rules and facts of a program is designed. The name SLD stands here for ' selected literally with definite clauses ' and has its origin in the work of Robert Kowalski for the linear resolution of horn clauses.

A SLD tree is suitable very well, in order to illustrate the expiration of a prolog program, because it represents the entire program with all its different processing. The root of this tree is our inquiry, which is knots the inquiries resulting in the program sequence and the sheets are with the key descriptors ' true ' or ' fail ' marks. The rules are processed from above downward, from left to the right.

Example 1



Example 1(cont.)

The SLD tree stretches itself in Example-1 over the inquiry 'a(X)' up. According to the first rule of our database is true a(X), if b apply and c(X). There is however no fact for the inquiry b, therefore is marked the first branch of the example tree with fail. On our request a(X) however also the second rule of our database can be used. Thereafter is true a(X) if D are true and c(X). There is a fact D, therefore D is true. For c(X) we find the fact c(1), whereby the variable X with 1 is substituted; c(X), the associated sheet of the tree is thus true is marked with true. There is however still another further rule c(2), for which c(X) is likewise true, the variable X obtains the value 2. Therefore a(1) and a(2) are possible answers on request a(X).

Horn Clause

A set of atomic literals with at most one positive literal. Usually written

$$L < -L1, ..., Ln$$

$$(\forall x1)(\forall x2)...(\forall xn)(L1 \wedge L2 \wedge ... \wedge Ln- > L)$$

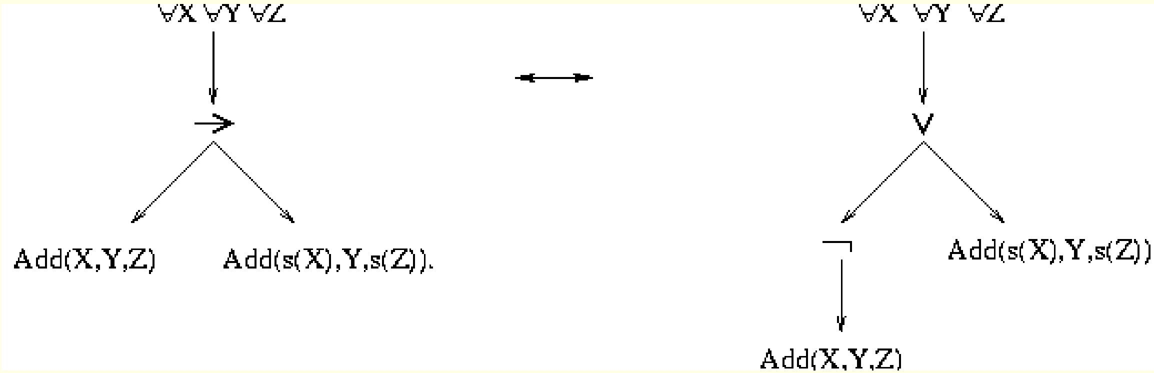
$$not(L1) \vee not(L2)...not(Ln) \vee L)(HornClause)$$

A definite clause is a Horn clause that has exactly one positive literal.

Examples

Prolog Rule (Horn Clause)

1. Add(0,X,X). (This is a unit clause with no negative literal)
2. Add(s(X),Y,s(Z)):-Add(X,Y,Z).



This clause has one negative literal.

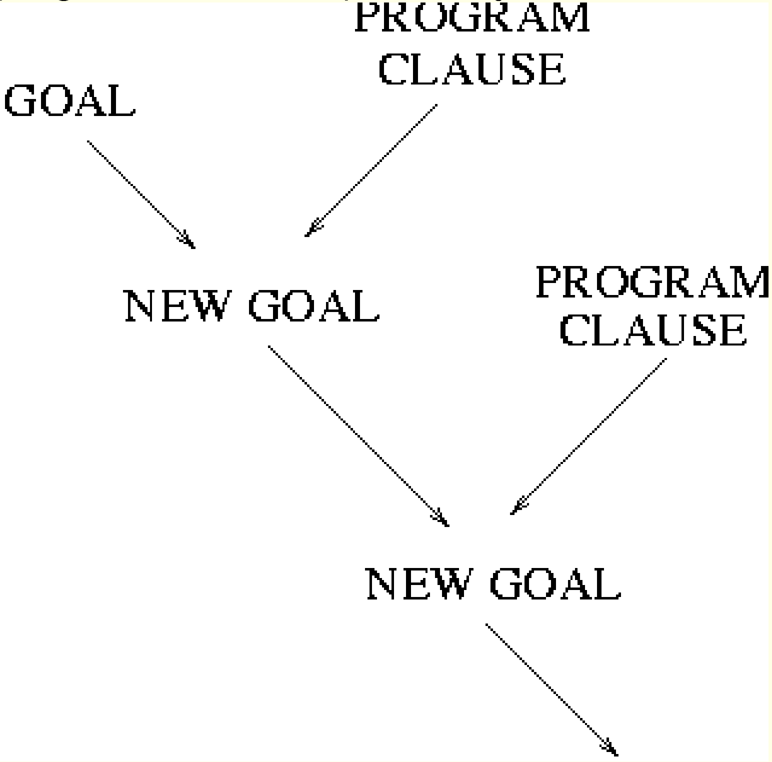
Prolog Goal

A Prolog Goal has single negative literal and no positive literal.

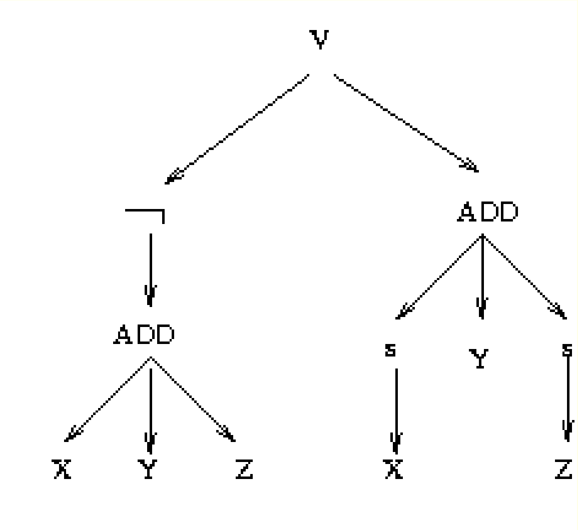
Prolog Goal :- add(X,Y,Z).

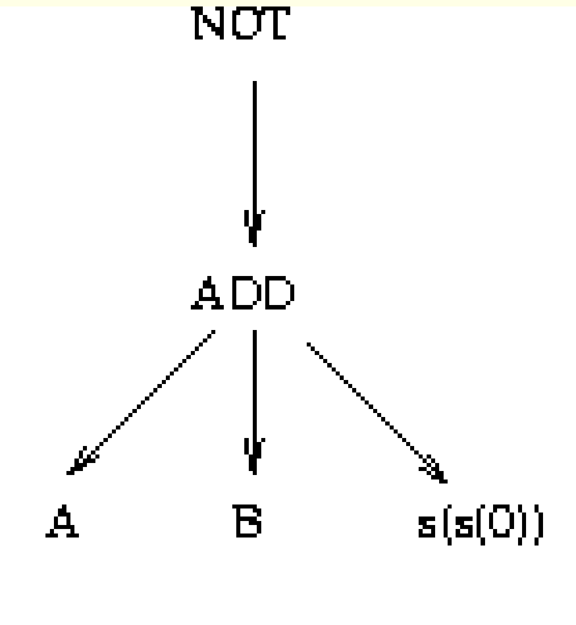
$$i.e. \exists x \exists y \exists z : -add(X, Y, Z).$$

Take the pre-nex normal form and Skolemize .Start by taking not of goal and resolve with the program clause from top to down, and left to right in order to get a new sub goal. This new sub goal can again be resolved with remaining program clauses to produce yet another sub goal and so on.



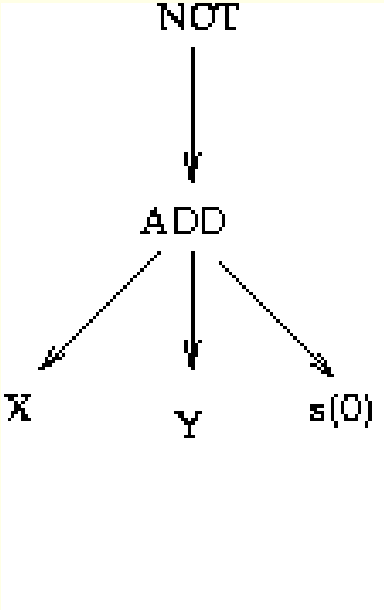
Clauses :

1. Add(0,X,X).
2. 

3. 

1. Resolving clause 3 with clause 1 we get A=0,B=s(s(0)).
2. Resolving clause 3 with clause 2 we get a new sub goal

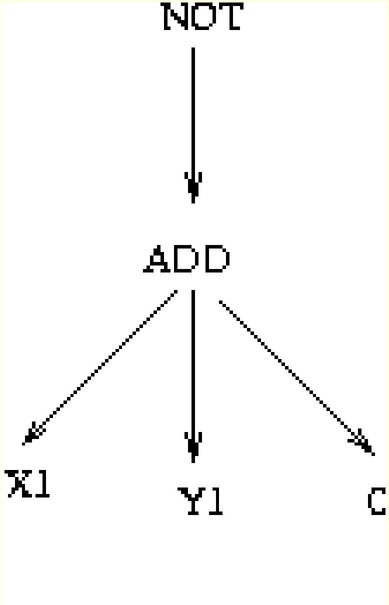
   not(Add(X,Y,Z)) and s(X)=A, Y=B and s(Z) = s(s(0)) (i.e. Z=s(0)).



3. Resolving this sub goal by clause 1 we get X=0, Y=s(0). Hence A=s(0), B=s(0).

4. Resolving this by clause 2 we get a new sub goal

not(Add(X1,Y1,Z1) and s(X1)=X, Y1=Y and s(Z1)=Z. (i.e. Z1=0)



Hence we get all the possible solutions.