Reasoning with First Order Logic (CP1)
SLD-Resolution: Examples

Ramón Béjar Torres

March 21, 2024

1 A Brief Reminder about SLD-Resolution

When solving queries of this kind:

$$P \models Q$$

but with first order logic programs (where $P \cup \{\neg Q\}$ will be in first order clausal form) we are going to use also SLD-Resolution, so we again will show all the possible answers to the query Q as successful branches (branches that end with the empty clause) of an SLD-Resolution Tree. Remember that ending with an empty clause means that we have reached a contradiction, so that $P \cup \{\neg Q\}$ is unsatisfiable and therefore Q is a logical consequence of P.

However, remember that resolution in first order logic needs to use unification between two complementary literals present in two clauses C_1 and C_2 to get a resolvent R that is a logical consequences of C_1 and C_2. This difference with propositional logic has the following consequences:

- A resolvent may have predicates with new terms: they are not present in any clause of the logic program (but they will be in the Herbrand Domain of the logic program).
- So, it is possible to get infinite different resolvents from the logic program, because the Herbrand Domain may be infinite when there are function symbols.
- A same clause may be reused in different resolution steps, but we have to rename its variables in every new resolution step we perform with the clause.

All this causes that there are SLD-Resolution Trees where some, or all, branches are infinite (so those branches do not end as either successful or failed).

In the SLD-Resolution Trees we show here, a goal clause G_i is connected with a resolvent R that is obtained resolving the leftmost literal L of G_i (that will be a negated literal) with the head literal H of another clause C_j (that will be a positive literal) of the program in the following way:

$$\begin{array}{c} G_i \\ \Big| \; j.\ \theta \\ R \end{array}$$

where θ denotes a most general unifier (mgu) of L and H. When the resolvent is the empty clause, R will be denoted as $\square$ and if G_i cannot be resolved with any clause of the program (its leftmost literal L does not unify with the head of any clause) then R will be denoted as $\blacksquare$.

2 CP1 Logic Programs - Examples

Example 1 Given the following logic program P:

1. $no\_barro\_zapatos(juan)$.
2. $no\_barro\_zapatos(pepe)$.
3. $no\_es\_asesino(X) \leftarrow no\_ha\_saltado(X)$.
4. $no\_ha\_saltado(X) \leftarrow no\_barro\_zapatos(X)$.

What consequences can we obtain from it ?

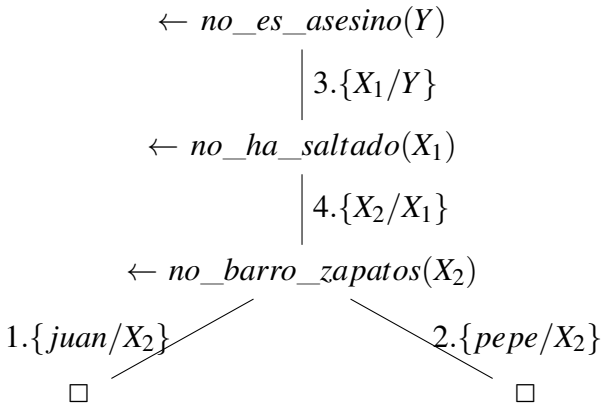

Figure 1: SLD-Resolution tree for the logic program of Example 1 with top goal $\leftarrow no\_es\_asesino(Y)$

If we want to prove the logical consequence:

$$P \models \exists Y no\_es\_asesino(Y)$$

We can do it by finding an SLD-Refutation for

$$\neg\exists Y no\_es\_asesino(Y) \equiv \forall Y \neg no\_es\_asesino(Y)$$

via P, or in logic programming terminology: an SLD-Refutation for the goal

$$\leftarrow no\_es\_asesino(Y)$$

via P.

The answers corresponding to the two successful branches of this tree are obtained with the composition of the unifiers used in the resolution steps of each branch:

1. Answer $no\_es\_asesino(juan)$. Obtained with the composition of unifiers on the first branch:
$$\{X_1/Y\} \cdot \{X_2/X_1\} \cdot \{juan/X_2\} = \{juan/Y\}$$

2. Answer $no\_es\_asesino(pepe)$. Obtained with the composition of unifiers on the second branch:
$$\{X_1/Y\} \cdot \{X_2/X_1\} \cdot \{pepe/X_2\} = \{pepe/Y\}$$

Example 2 Given the following logic program P:

$$1.\ ancestor(barcenas, cospedal).$$
$$2.\ ancestor(f(X), Y) \leftarrow ancestor(X, Y).$$

where the intended meaning of $f(X)$ is that it gives the father of X. What consequences can we obtain from it ?

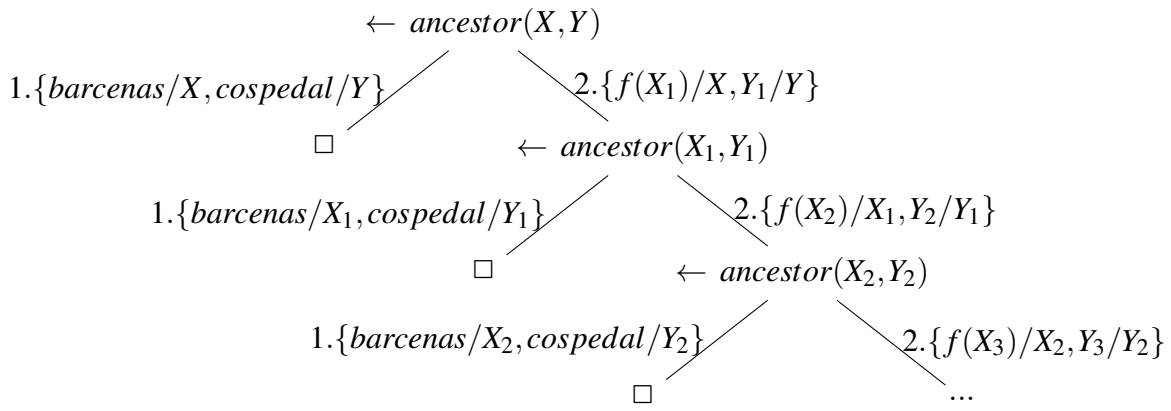

Figure 2: SLD-Resolution tree for the logic program of Example 2 with top goal $\leftarrow ancestor(X,Y)$. The branch that ends with the symbol ... indicates that is a never ending subtree, that can be extended forever, but the more we extend it, the more successful answers we will get.

This example gives an SLD-Resolution Tree with an infinite number of sucessful branches, because there are infinite answers to this query:

$$ancestor(barcenas, cospedal), ancestor(f(barcenas), cospedal), ancestor(f(f(barcenas)), cospedal), \ldots$$

These are the corresponding answers to some of the branches (obtained with the composition of the unifiers used in all the resolution steps of the branch):

1. Answer $ancestor(barcenas, cospedal)$. Obtained with the unique unifier used on the first branch:
$$\{barcenas/X, cospedal/Y\}$$

2. Answer $ancestor(f(barcenas), cospedal)$. Obtained with the composition of the unifiers used on the resolution steps of the second successful branch:
$$\{f(X_1)/X, Y_1/Y\} \cdot \{barcenas/X_1, cospedal/Y_1\} = \{f(barcenas)/X, cospedal/Y\}$$

3. Answer $ancestor(f(f(barcenas)),cospedal)$. Obtained with the composition of the unifiers used on the resolution steps of the third successful branch:

$$\{f(X_1)/X, Y_1/Y\} \cdot \{f(X_2)/X_1, Y_2/Y_1\} \cdot \{barcenas/X_2, cospedal/Y_2\} = \{f(f(barcenas))/X, cospedal/Y$$

So, we conclude that there is an infinite set of ancestors of cospedal related to his ancestor barcenas.

Example 3 Consider an investigation for finding what people corrupt other people, so that we start from some ground facts with the relation $co(X,Y)$ for some particular pairs of persons. Then, we also consider the information about friendship between some people $fr(X,Y)$ and a rule that says that: if a person Y is friend of X and a person Z corrupts Y, then Z also corrupts X. With all this information, the following logic program P can help us to discover the extent of the corruption.

1. $co(c,b)$.
2. $co(h,j)$.
3. $fr(b,a)$.
4. $fr(d,a)$.
5. $fr(j,d)$.
6. $co(Z,X) \leftarrow fr(Y,X), co(Z,Y)$.

To discover all the cases of corruption, the query to answer is: $\exists Z \exists X \; co(Z,X)$?

Answers:

1. $co(c,b)$. This answer is obtained in the first successful branch, resolving the goal with the first clause of the program.

2. $co(h,j)$. This answer is obtained in the second successful branch, resolving the goal with the second clause of the program.

3. $co(c,a)$ This answer is obtained in the third successful branch, after composing the substitutions obtained as unifiers in every step:

$$\{Z_1/Z, X_1/X\} \cdot \{b/Y_1, a/X_1\} \cdot \{c/Z_1\} = \{c/Z, a/X\}$$

4. $co(h,a)$ This answer is obtained in the fourth successful branch, after composing the substitutions obtained as unifiers in every step:

$$\{Z_1/Z, X_1/X\} \cdot \{d/Y_1, a/X_1\} \cdot \{Z_3/Z_1, d/X_3\} \cdot \{j/Y_3\} \cdot \{h/Z_3\} = \{h/Z, a/X\}$$

5. $co(h,d)$ This answer is obtained in the last successful branch, after composing the substitutions obtained as unifiers in every step:

$$\{Z_1/Z, X_1/X\} \cdot \{j/Y_1, d/X_1\} \cdot \{h/Z_1\} = \{h/Z, d/X\}$$

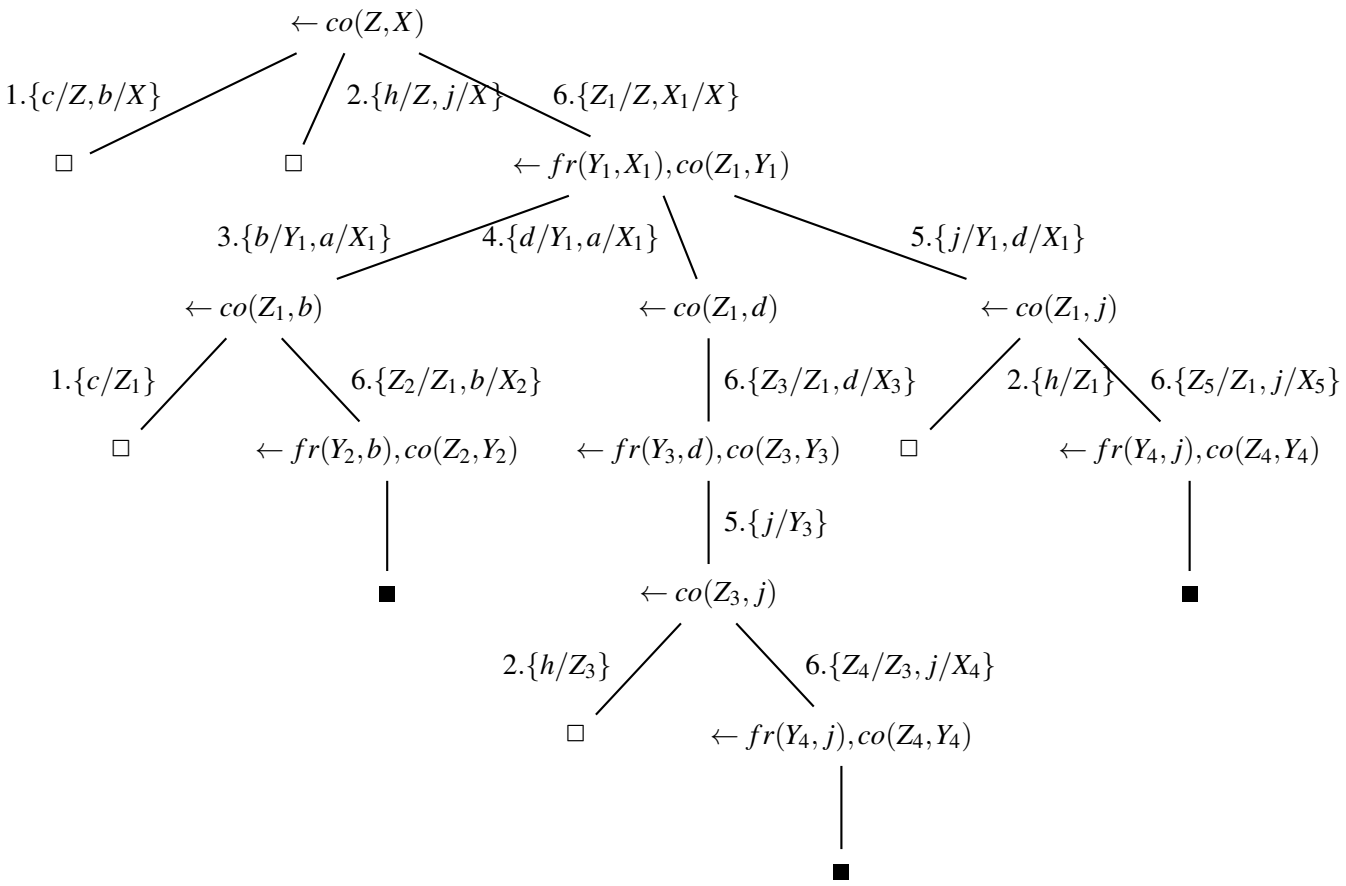

Figure 3: SLD-Resolution tree for the logic program of Example 3 with top goal $\leftarrow co(Z,X)$.

Exercise: Consider what happens if we try to solve the query, but substituting clause 6 of the logic program by this one:

$$co(Z,X) \leftarrow co(Z,Y), fr(Y,X).$$

In principle, we should be able to find the same answers, because this rule is logically equivalent to the original one. But depending on the way you perform search in the SLD-Resolution Tree, you may have problems to discover all the answers. Check it !

Example 4 Given the following logic program P:

1. $even(0)$.
2. $even(s(s(X))) \leftarrow even(X)$.

where the intended meaning of $s(X)$ is that it gives the successor number of X ($X+1$ if we interpret the domain as the set of numbers $\mathbb{N} \cup \{0\}$). What consequences can we obtain from it ?

Consider trying to answer if from P we can conclude that there is a number Z such that both Z and $s(s(s(Z)))$ are even numbers. That is, checking whether:

$$P \models \exists Z(even(Z) \wedge even(s(s(s(Z)))))$$

when we negate the query, we get the following goal:

$$\leftarrow even(Z), even(s(s(s(Z))))$$

It turns out that the above query is not a logical consequence of P, so that the SLD-Resolution Tree does not contain any successful branch. The problem is that it contains an infinite branch that never ends, and a number of infinite failed branches, so with SLD-Resolution we cannot certify that the query is not a logical consequence. Figure 4 shows its SLD-Resolution Tree. Compare this situation to the one we have in Example 2.

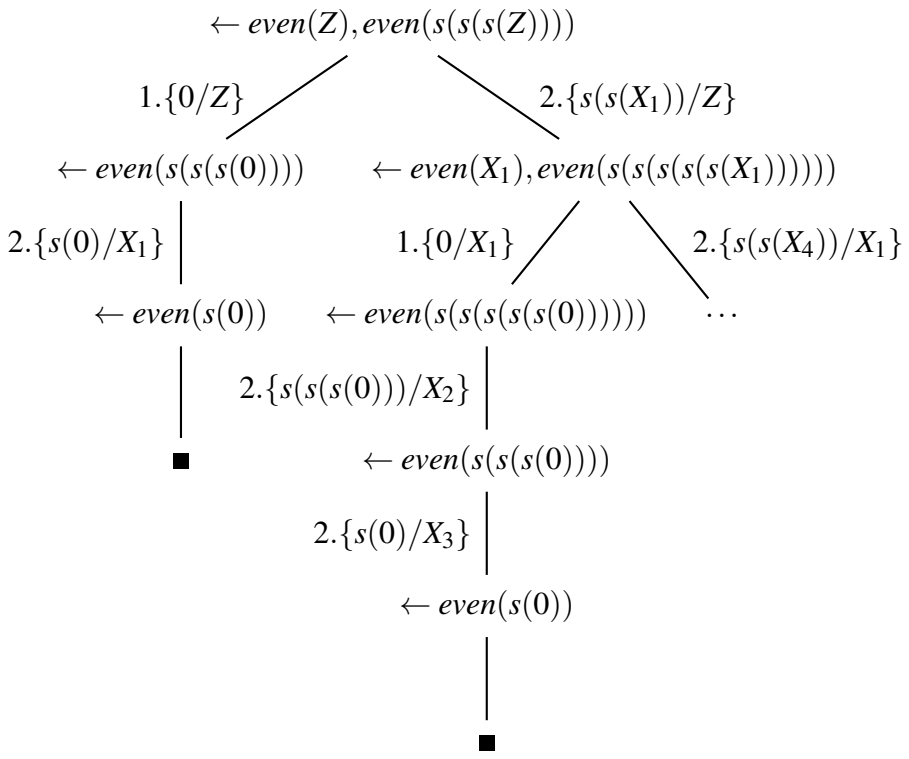

Figure 4: SLD-Resolution tree for the logic program of Example 4 with top goal $\leftarrow even(Z), even(s(s(s(Z))))$. The branch that ends with the symbol ... indicates that is a never ending subtree, that can be extended as much as we want, but the more we extend it, the more unsuccessful branches we will get.

What is going on with this tree is that (with depth-first search) we first try to prove the goal unifying $even(Z)$ with $even(0)$, so this implies to try to prove also $even(s(s(s(0))))$, but this last subgoal fails (it cannot be proved from P). Then, the next option would be to unify $even(Z)$ with $even(s(s(X)))$ and then unify $even(X)$ with $even(0)$, so this would imply to try to prove also $even(s(s(s(s(s(0))))))$ (is 5 an even number ?), but this also fails. This process can be repeated forever, extending in every step the different odd numbers that are tried to be proved as logical consequences of this program. But the only numbers X such that $even(X)$ is a logical consequence of P are the even numbers.