european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitiveness
INVESTMENTS IN EDUCATION DEVELOPMENT

1 Resolution in predicate logic

Exercise 1.1: Find all possible resolvents of the following pairs of clauses:

a) $C_1 = \{P(x)\}, C_2 = \{\neg P(f(x))\}$

b) $C_1 = \{P(x), P(y)\}, C_2 = \{\neg P(x), \neg P(y)\}$

c) $C_1 = \{P(x, y), P(y, z)\}, C_2 = \{\neg P(u, f(u))\}$

d) $C_1 = \{P(x, x), \neg R(x, f(x))\}, C_2 = \{R(x, y), Q(y, z)\}$

e) $C_1 = \{P(x, y), \neg P(x, x), Q(x, f(x), z)\}, C_2 = \{\neg Q(f(x), x, z), P(x, z)\}$

Solution 1.1: First, it is necessary to rename variables so that there are no common variable names in C_1, C_2. Then the set of literals for resolution is selected and unified. Its mgu is applied to C_1 and C_2 and then the resolution can be performed.

a) renaming in C_2: $\{x/y\}$
unification of the set: $\{P(x), P(f(y))\}$
mgu: $\{x/f(y)\}$
resolvent: $\square$

b) renaming in C_2: $\{x/x_1, y/y_1\}$
unification of the set: $\{P(x), P(y), P(x_1), P(y_1)\}$
mgu: $\{y/x, x_1/x, y_1/x\}$
resolvent: $\square$

Note: there are more solutions; they can be obtained when smaller subsets of literals are selected

c) two possible solutions:
no renaming
unification of the set: $\{P(x, y), P(u, f(u))\}$
mgu: $\{x/u, y/f(u)\}$
resolvent: $\{P(f(u), z)\}$

no renaming
unification of the set: $\{P(y, z), P(u, f(u))\}$
mgu: $\{y/u, z/f(u)\}$
resolvent: $\{P(x, u)\}$

Note: there are no more solutions; it is not possible to obtain $\square$ as a resolvent!

Other subtasks: analogically.

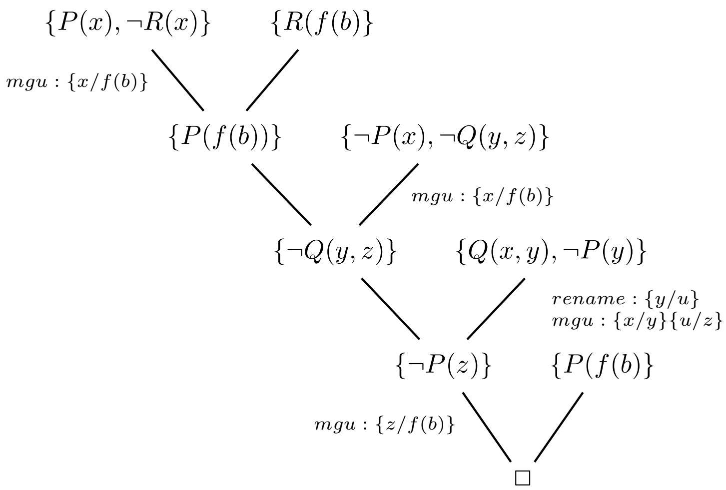
Exercise 1.2: Refute the following set of clauses

$$S = \{\{P(x), \neg Q(x, f(y)), \neg R(a)\}, \{R(x), \neg Q(x, y)\}, \{\neg P(x), \neg Q(y, z)\}, \{P(x), \neg R(x)\}, \{R(f(b))\}, \{Q(x, y), \neg P(y)\}\}$$

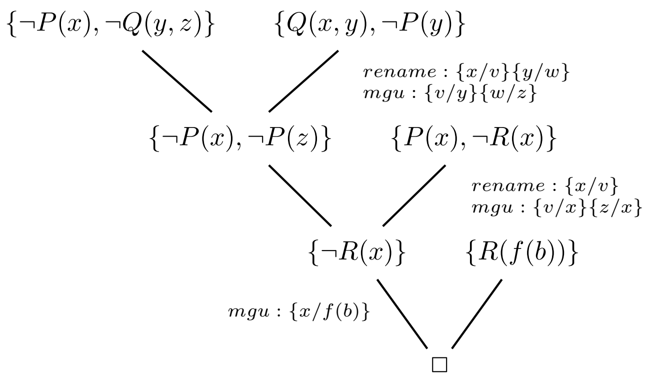
using linear resolution, LI resolution, LD resolution and SLD resolution.

Solution 1.2:

a) Linear resolution: the former resolvent is resolved with a clause from the refuted set or with some previous resolvent. The proof has a linear structure. Linear resolution is complete.



b) LI resolution (linear input r.) starts with a goal clause (= with no positive literal). The former resolvent is resolved with a clause from the refuted set. LI resolution is refined linear resolution and is not complete for general sets of clauses. However, it is complete for sets of Horn clauses (= with at most one positive literal).



c) LD resolution: a step towards implementation. It is defined only for Horn clauses and it is complete for them. Clauses are represented as ordered lists (usually referred as *ordered clauses* or *definite clauses*):

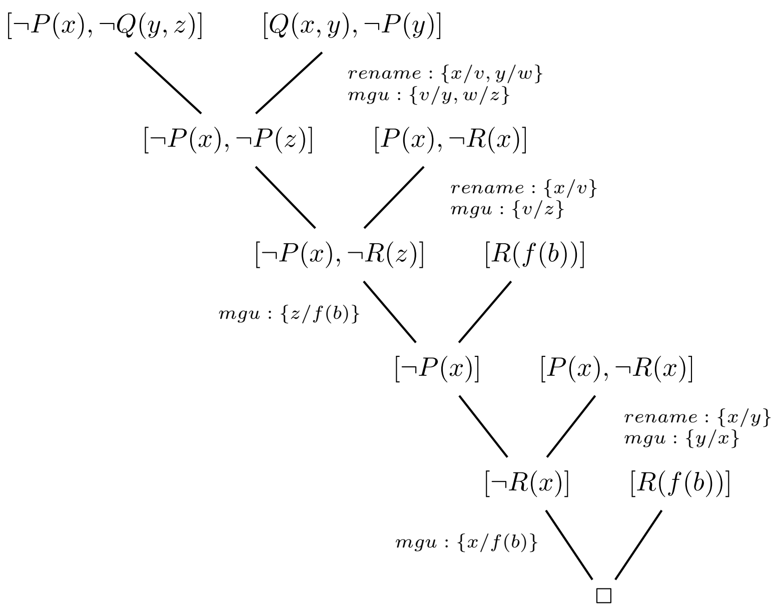
$$\begin{aligned} S' &= \{[P(x), \neg Q(x, f(y)), \neg R(a)], [R(x), \neg Q(x, y)], [\neg P(x), \neg Q(y, z)], \\ &\quad [P(x), \neg R(x)], [R(f(b))], [Q(x, y), \neg P(y)]\} \end{aligned}$$

Resolution rule is defined as follows: Let us have the ordered clauses

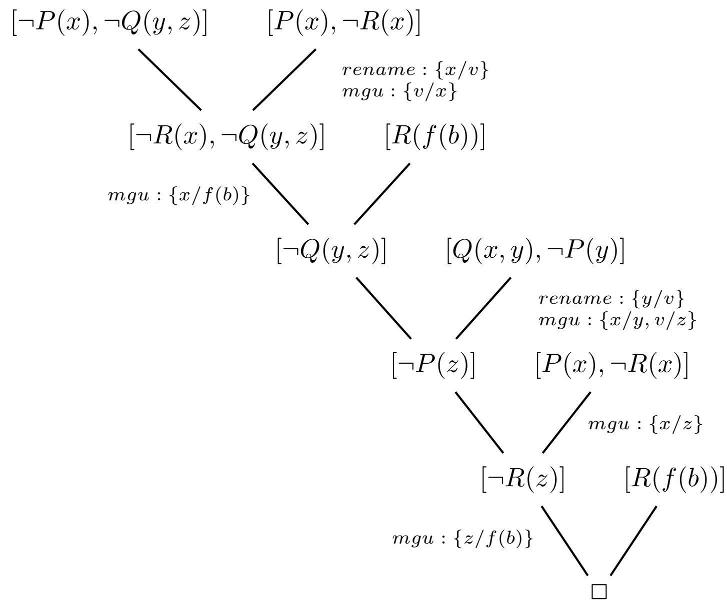
$$\begin{aligned} G &= [\neg A_1, \neg A_2, \ldots, \neg A_n] \text{ a} \\ H &= [B_0, \neg B_1, \neg B_2, \ldots, \neg B_m]. \end{aligned}$$

The resolvent of G and H for $\phi = mgu(B_0, A_i)$ is the following ordered clause:

$$[\neg A_1\phi, \neg A_2\phi, \ldots, \neg A_{i-1}\phi, \neg B_1\phi, \neg B_2\phi, \ldots, \neg B_m\phi, \neg A_{i+1}\phi, \ldots, \neg A_n\phi].$$



d) SLD resolution: next step towards implementation. It is a case of LD resolution. A rule (function) selects the literal for resolution. Prolog is an implementation of SLD resolution and here the selection rule always chooses the leftmost literal.



2 SLD-trees and resolution in Prolog

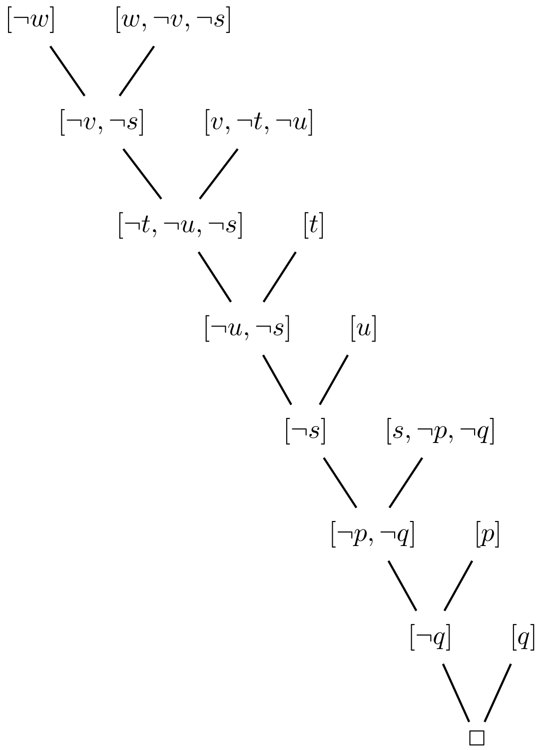
Exercise 2.1: Find a resolution refutation of the following program and goal in Prolog.

```
1. r :- p, q.        5. t.
2. s :- p, q.        6. q.
3. v :- t, u.        7. u.
4. w :- v, s.        8. p.

?- w.
```

Solution 2.1: Transformation into the set of ordered clauses:

$$S = \{[r, \neg p, \neg q], [s, \neg p, \neg q], [v, \neg t, \neg u], [w, \neg v, \neg s], [t], [q], [u], [p], [\neg w]\}$$

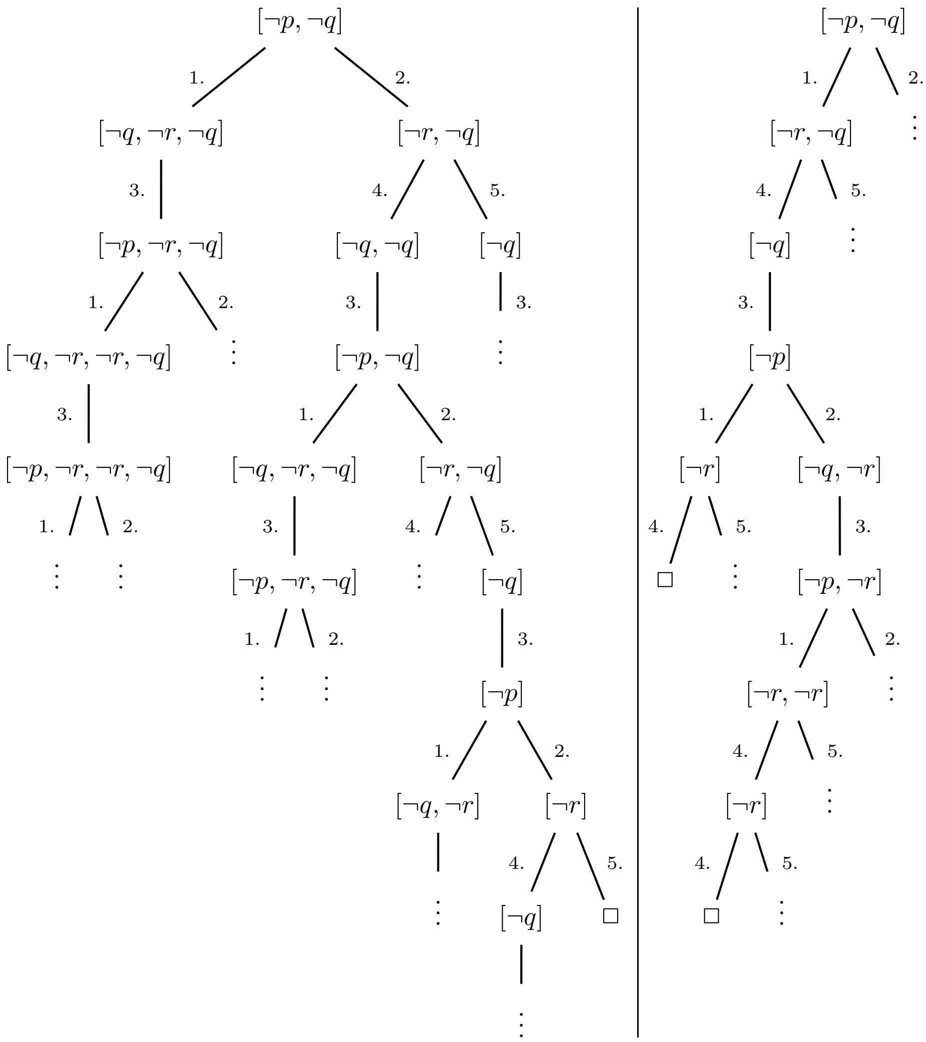
SLD resolution refutation:


[¬w]
[w, ¬v, ¬s]
[¬v, ¬s]
[v, ¬t, ¬u]
[¬t, ¬u, ¬s]
[t]
[¬u, ¬s]
[u]
[¬s]
[s, ¬p, ¬q]
[¬p, ¬q]
[p]
[¬q]
[q]
□

Exercise 2.2: Draw the SLD-tree for the following Prolog program (`Program 1`) and query. Find out, how the different order of the clauses (`Program 2`) affects the SLD-tree.

```
Program 1:                          Program 2:
 1. p :- q,r.      4. r :- q.        1. p :- r.        4. r.
 2. p :- r.        5. r.             2. p :- q,r.      5. r :- q.
 3. q :- p.                          3. q :- p.

 ?- p,q.
```

Solution 2.2:
SLD-trees (ordered clause notation):



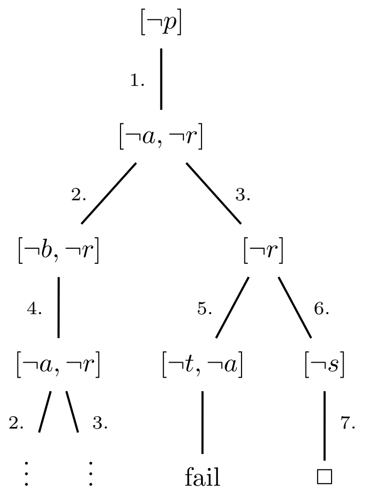
Exercise 2.3: Draw the SLD-trees for the following Prolog programs and goals:

```
Program 1:
 1. p :- a,r.  5. r :- t,a.
 2. a :- b.    6. r :- s.
 3. a.         7. s.
 4. b :- a.

 ?- p.
```

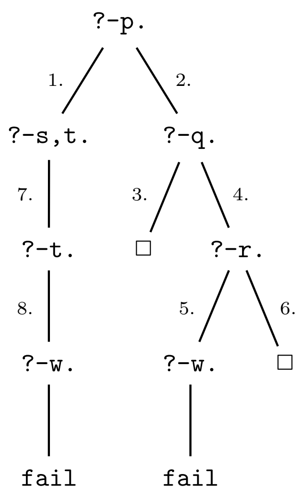
```
Program 2:
 1. p :- s,t.  5.  r :- w.
 2. p :- q.    6.  r.
 3. q.         7.  s.
 4. q :- r.    8.  t :- w.

 ?- p.
```

Solution 2.3:
SLD-trees: Program 1: ordered clause notation, Program 2: Prolog notation

Program 1:



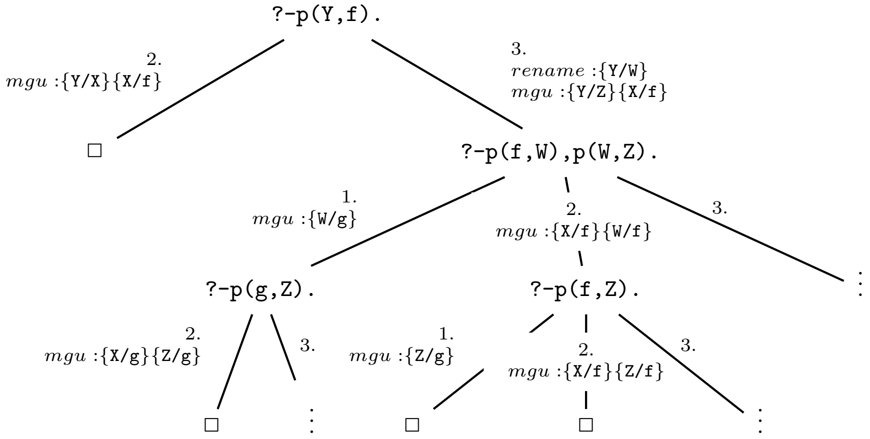
Program 2:



Exercise 2.4: Draw the SLD-tree for the following Prolog program and goal:

```
 1. p(f,g).     3. p(Z,X) :- p(X,Y), p(Y,Z).
 2. p(X,X).

 ?- p(Y,f).
```

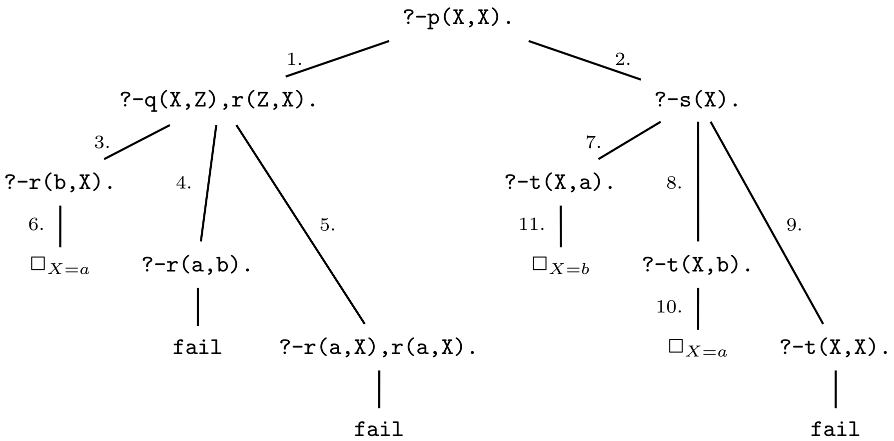
Solution 2.4: SLD-tree:



Exercise 2.5: Draw the SLD-tree for the following Prolog program and goal:

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

?- p(X,X).
```

Solution 2.5: SLD-tree: Prolog notation, renaming and mgus ommited (there are only final substitutions of the query variable)



Exercise 2.6: Find an SLD-resolution refutation of the goal `?- reverse([a,b,c],X).` assuming that the predicate `reverse/2` is defined as follows:

```
reverse(L1,L2) :- rev(L1,[],L2).
rev([H|T],A,L) :- rev(T,[H|A],L).
rev([],L,L).
```

Solution 2.6: SLD resolution refutation:

```
?-reverse([a,b,c],X).     reverse(L1,L2) :-rev(L1,[],L2).
                          mgu :{L1/[a,b,c]}{L2/X}
?-rev([a,b,c],[],X).      rev([H|T],A,L) :-rev(T,[H|A],L).
                          mgu : {H/a}{T/[b,c]}{A/[]}{L/X}
?-rev([b,c],[a],X).       rev([H|T],A,L) :-rev(T,[H|A],L).
                          mgu : {H/b}{T/[c]}{A/[a]}{L/X}
?-rev([c],[b,a],X).       rev([H|T],A,L) :-rev(T,[H|A],L).
                          mgu : {H/c}{T/[]}{A/[b,a]}{L/X}
?-rev([],[c,b,a],X).                          rev([],L,L).
                          mgu : {L/[c,b,a]}{X/[c,b,a]}
□ X=[c,b,a]
```

Resolution shows that the goal is a logical consequence of the program. In addition to it a result (reverted list) is also produced. It is the final substitution of the variable `X`, which is the answer of the Prolog system to the query. The query can be interpreted as follows: Is the formula ∃`X reverse([a,b,c],X)` a logical consequence of the program? If so, for what `X`?