Ex 3.4.1 Up to now: before performing resolution, we have to instantiate variables by ground terms. This instantiation does not only have to enable the next resolution step, but one has to guess the right instantiation which also allows all needed future resolution steps.

We need an inst. for \( X \) and \( Y \) such that

\[ p(X) \quad \text{and} \quad p(f(Y)) \] become equal.

(i.e., we have to unify \( p(X) \) and \( p(f(Y)) \))

But this instantiation should also allow future resolution steps (e.g., between \( q(...) \) and \( \neg q(...) \)).

Solution: do not instantiate variables by ground terms, but also allow instantiations by arbitrary terms. Only look for most general unifiers, i.e., only instantiate them in such a way that the next resolution step is possible.

In the example: Finally one uses \( \{ Y / a \} \) to derive \( \Pi \).

Def 3.4.2 (Unification)

A clause \( \Pi = \{ L_1, \ldots, L_n \} \) is unifiable iff there exists a substitution \( \sigma \) such that \( \sigma(L_1) = \ldots = \sigma(L_n) \) (i.e.,
$|\sigma(K)| = 1$. Such a subst. $\sigma$ is a unifier of $K$.

A unifier $\sigma$ is a most general unifier (mgu) iff

for any unifier $\sigma'$ there exists a subst. $\sigma$ such that

$\sigma'(X) = \sigma(\sigma(X))$ for all $X \in V$.

In the example: $K = \{ p(X), p(f(Y)) \}$

mgu $\sigma = \{ X / f(Y) \}$  $|\sigma(K)| = |\{ \sigma(p(X)), \sigma(p(f(Y))) \}|$ = 2 $p(f(Y)) \notin \Gamma = \gamma$

alternative unifier $\sigma' = \{ X / f(a), Y / a \}$

we have $\sigma' = \sigma \circ \gamma$

for $\gamma = \{ Y / a \}$

Observations:

- If a clause is unifiable, then it also has an mgu.
- The mgu is unique up to variable renaming.
  
  Ex: $\{ p(X), p(Y) \}$

  mgu $\sigma = \{ X / Y \}$ or $\sigma' = \{ Y / X \}$

- It is decidable whether a clause is unifiable
  and the mgu is computable.
- First unification algorithm by J. Robinson (1965).
  
  inventor of resolution

Ex 343
• Try to unify \{ q(f(X,Y)), q(g(X,Y)) \}
  \( \sigma = \emptyset \)

  **clash failure**

• Try to unify \{ q(X), q(g(a,Y)) \}
  \( \sigma = \emptyset \)

  **occur failure**

• Try to unify \{ p(f(z,g(a,Y)), g(z)), p(f(f(u,v),w), g(f(a,Y)) \}
  \( \sigma = \emptyset \)

  \( \sigma = \{ z/f(u,v) \} \)

  \{ p(f(f(u,v),g(a,Y)), h(f(u,v))), p(f(f(u,v),w), h(f(a,Y))) \}

  \( \sigma = \{ w/g(a,Y) \} \circ \{ z/f(u,v) \} = \{ w/g(a,Y), z/f(u,v) \} \)

  \{ p(f(f(u,v),g(a,Y)), h(f(u,v))), p(f(f(u,v),g(a,Y)), h(f(a,Y))) \}

  \( \sigma = \{ u/a \} \circ \{ w/g(a,Y), z/f(u,v) \} = \{ u/a, w/g(a,Y), z/f(a,Y) \} \)

  \{ p(f(f(a,v),g(a,Y)), h(f(a,v))), p(f(f(a,v),g(a,Y)), h(f(a,Y))) \}

  \( \sigma = \{ y/v \} \circ \ldots = \{ y/v, u/a, w/g(a,Y), z/f(a,Y) \} \)

  \{ p(\ldots), p(\ldots) \}
are the same now

Friday (May 8): 2 lectures (lecture instead of exerc. course)

Monday (May 11): ex. course instead of lecture

Thm 3.44 (Termination + Soundness of Unif. Alg.)
The unif. alg. terminates for every clause $K$ and it is sound, i.e., it returns an mgu for $K$ iff $K$ is unifiable.

Proof: The alg. terminates because the number of variables in the clause decreases in each loop iteration.

If the alg. returns a subst. $\sigma$, then $\sigma$ is a mgu of $K$ (since it checks $|\sigma(K)| = 1$ in step 2).

Thus: if $K$ is not unifiable

1) alg can't return a subst. $\sigma$
2) alg stops with failure. (since alg. terminates)

It remains to prove:

If $K$ is unifiable, then alg. returns a mgu $\sigma$.

Let $m \geq 0$ be the number of loop iterations of the alg. for the input $K$. For every $0 \leq i \leq m$, let $\sigma_i$ be the value of $\sigma$ after the $i$-th loop iteration.
We prove the following for all $0 \leq i \leq m$:

For every unifier $\sigma'$ of $K$, we have $\sigma' = \sigma' \circ \sigma_i$. (8)

This implies the soundness of the alg. if $K$ is unifiable:

• If the alg would stop with failure in the $(m+1)$-th loop iteration, then $\sigma_m(K)$ would not be unifiable.

But (8) implies: $\sigma' = \sigma' \circ \sigma_m$ and there exists a unifier $\sigma'$ of $K$.

$|\sigma'(K)| = 1$

$\exists |\sigma' \circ \sigma_m(K)| = 1$

$\exists \sigma'$ is a unifier of $\sigma_m(K)$. $\forall$

• So the alg. has to stop with success

$\forall |\sigma_m(K)| = 1$, i.e., $\sigma_m$ is a unifier.

Now (8) implies that for every unifier $\sigma'$ there exists a subst $\sigma'(\forall \sigma'_i = \sigma'')$ such that:

$\sigma' = \sigma' \circ \sigma_m$

$\forall \sigma_m$ is unif. of $K$.

Now we prove the following for all $0 \leq i \leq m$ by induction on $i$:

For every unifier $\sigma'$, we have $\sigma' = \sigma' \circ \sigma_i$. (8)

**Ind Base**: $i = 0$

$\sigma_0 = \emptyset$ $\Rightarrow \sigma' = \sigma' \circ \sigma_0$ holds for all substitutions $\sigma'$.

**Ind Step**: $i > 0$

Ind. Hypothesis: $\sigma' = \sigma' \circ \sigma_{i-1}$

To unify $\sigma_{i-1}(K)$, one has to replace a var. $X$ by a term $t$.
in Step 6. Thus: \( \sigma_i = \{ X/E \} \circ \sigma_{i-1} \).

We have:

\[
\sigma' \circ \sigma_i \\
= \sigma' \circ \{ X/E \} \circ \sigma_{i-1} \\
= \sigma' \circ \sigma_{i-1} \\
= \sigma' \quad \text{(by the ind. hyp.)}
\]

Reason for \( \sigma' = \sigma' \circ \{ X/E \} \):

For \( Y \neq X \):
\[
\sigma'(Y) = (\sigma' \circ \{ X/E \})(Y)
\]

For \( X \):
\[
(\sigma' \circ \{ X/E \})(X) = \sigma'(t) = \sigma'(X)
\]

Reason: \( \sigma' \) is also a unifier of \( \sigma_{i-1}(K) \)

\( (\text{since } |\sigma'(\sigma_{i-1}(K))| = |\sigma'(K)| = 1) \)

by ind. hyp.

Therefore, \( \sigma' \) must make \( X \) and \( t \) equal.

\[ \square \]

**Def 3.45** (Resolution in Predicate Logic)

Let \( K_1, K_2 \) be clauses. Then a clause \( R \) is resolvent of \( K_1 \) and \( K_2 \) iff:

- There exist variable renamings \( \varphi_1, \varphi_2 \) such that \( \varphi_1(K_1) \) and \( \varphi_2(K_2) \) have no common variables. \( \star \)
- There exist literals \( L_1, \ldots, L_m \in \varphi_1(K_1) \) and \( L_1', \ldots, L_n' \in \varphi_2(K_2) \) with

for universally quantified disjunction of its literals (i.e.: renaming of bound variables)
\[ L_1, \ldots, L_n \in \bigvee_2 (K_2) \text{ with } \nu, \mu \geq 1 \text{ such that} \]
\[ \{ \overline{L}_1, \ldots, \overline{L}_m, L'_1, \ldots, L'_n \} \text{ are unifiable with w.r.t. } \sigma \]
\[ R = 0 \left( (\nu_2(K_2) \setminus \{L_1, \ldots, L_m\}) \cup (\nu_1(K_1) \setminus \{L'_1, \ldots, L'_n\}) \right) \]

As before, we define the following for a clause set \( K \):
\[ \text{Res}(K) = K \cup R \mid R \text{ is resolvent of 2 clauses in } K \]
\[ \text{Res}^0(K) = K \]
\[ \text{Res}^u(K) = \text{Res}(\text{Res}^{u-1}(K)) \text{ for all } u \geq 0 \]
\[ \text{Res}^u(K) = \bigcup_{u \geq 0} \text{Res}^u(K) \]

Clearly, propositional resolution is a special case of this form of resolution.

[Example 346]

\[ \{ p(f(X)), \neg q(z), p(z) \} \]
\[ \{ \neg p(X), \forall g(X) \} \]
\[ \{ \neg p(U), \forall g(U) \} \]
\[ \{ \neg q(f(X)), \forall g(f(X)) \} \]
\[ \nu_2(K_2) \]

Variable renaming:
\[ \nu_1 = \emptyset \]
\[ \nu_2 = \{ X / U, U / X \} \text{ (must be injective)} \]
\[ L_1 = p(f(X)) \]
\[ L_2 = p(z) \]
\[ L_1' = \neg p(U) \]
Resolution in prel. logic is also sound and complete, i.e.: \( \neg \Phi \) is unsatisfiable iff \( \Box \in \text{Res}^*(\neg \Phi) \)

Soundness can be shown in a similar way as in prop. logic.

Lemma 3.4.7 (Resolution Lemma for Pred. Logic)

Let \( \Phi \) be a clause set. If \( \Phi_1, \Phi_2 \in \Phi \) and \( \Gamma \) is a resolvent of \( \Phi_1 \) and \( \Phi_2 \), then \( \Phi \) and \( \Phi \cup \{ \Gamma \} \) are equivalent.

Proof: similar to the propositional resolution lemma (Lemma 3.3.6)

This implies soundness of resolution:

1. \( \Box \in \text{Res}^*(\neg \Phi) \)
2. \( \Box \in \text{Res}^* (\neg \Phi) \) for some \( n \geq 0 \)
3. \( \text{Res}^* (\neg \Phi) \) is unsatisfiable
4. \( \Phi \) is unsatisfiable

(since the resolution lemma implies that \( \Phi \) and \( \text{Res}^* (\neg \Phi) \) are equivalent. Thus, by induction on \( n \) one can show that \( \Phi \) and \( \text{Res}^* (\neg \Phi) \) are equivalent.)

Now: Show completeness of resolution in pred. logic.

Goal: Use completeness of prop. resolution to show completeness of full resolution.
Solution: Lifting Lemma
shows how to "lift" a resolution proof from propositional
logic to predicate logic.

Lemma 348 (Lifting Lemma)
Let \( K_1, K_2 \) be clauses with ground instances \( K_1', K_2' \).
If \( R' \) is a (propositional) resolvent of \( K_1' \) and \( K_2' \)
then there exists a resolvent \( R \) of \( K_1 \) and \( K_2 \) such that
\( R' \) is a ground instance of \( R \).

The lifting lemma states that one could instead perform
resolution on \( K_1 \), \( K_2 \) and obtain a resolvent \( R \) such that
\( R' \) is a ground instance of \( R \):
\[
\begin{align*}
\{ p(f(x)), \neg q(z), p(z) \} & \quad \{ \neg p(x), \neg q(g(x)) \} \\
\{ p(f(a)), \neg q(f(a)) \} & \quad \{ \neg p(f(a)), \neg q(g(f(a))) \} \\
\{ \neg q(f(a)), r(g(f(a))) \} & \quad \{ \neg p(f(a)), r(g(f(a))) \} \\
\{ \neg p(f(a)) \} & \quad \{ \neg p(f(a)) \} \\
\{ \neg p(f(a)), r(g(f(a))) \} & \quad \{ \neg p(f(a)), r(g(f(a))) \} \\
\{ \neg p(f(a)) \} & \quad \{ \neg p(f(a)) \}
\end{align*}
\]
Proof of the Lifting Lemma:

\[
\{ \forall y(f(x)), \forall y(g(f(x))) \} \vdash \{ \forall x \}.
\]

Let \( \nu_1, \nu_2 \) be var. renamings such that \( \nu_1(K_1) \) and \( \nu_2(K_2) \) are variable-disjoint.

Then \( K_1', K_2' \) are also ground instances of \( \nu_1(K_1) \) and \( \nu_2(K_2) \).

Since \( \nu_1(K_1) \) and \( \nu_2(K_2) \) are variable-disjoint, one can use the same ground subst. \( \sigma : \)

\[
\sigma(\nu_1(K_1)) = K_1' \quad \text{and} \quad \sigma(\nu_2(K_2)) = K_2' \quad \text{with} \quad K_1' \subseteq K_1 \quad \text{and} \quad K_2' \subseteq K_2.
\]

Since \( R' \) is resolvent of \( K_1' \) and \( K_2' \), there is a literal \( L \subseteq K_1 \) with \( \overline{L} \subseteq K_2' \) and

\[
R' = (K_1' \setminus L) \cup (K_2' \setminus \overline{L}).
\]

Let \( L_1, \ldots, L_m \) be all literals from \( \nu_1(K_1) \) that are mapped to \( L \) by \( \sigma \) (i.e., \( \sigma(L_1) = \ldots = \sigma(L_m) = L \)).

Let \( L_1', \ldots, L_n' \) be all literals from \( \nu_2(K_2) \) that are mapped to \( \overline{L} \) by \( \sigma \) (i.e., \( \sigma(L_1') = \ldots = \sigma(L_n') = \overline{L} \)).

We must have \( m, n \geq 1 \).

Since \( \sigma \) is a unifier of \( \{ L_1, \ldots, L_m, L_1', \ldots, L_n' \} \), there also exists a unifier \( \sigma' \).

Therefore, \( K_1 \) and \( K_2 \) have a resolvent.
\[ R = \varphi \left( (\forall_1(\neg L_1) \setminus \{L_1, \ldots, L_m\}) \cup (\forall_2(\neg L_2) \setminus \{L'_1, \ldots, L'_n\}) \right). \]

Thus:

\[ L_1 \quad L_2 \quad \quad L'_1 \quad L'_2 \]

\[ \neg L_1 \quad \neg L_2 \quad R \]

\[ \neg L'_1 \quad \neg L'_2 \quad R' \]

This still has to be shown.

Since \( \varphi \) is a unifier of \( \{L_1, \ldots, L_m, L'_1, \ldots, L'_n\} \) and \( \varphi' \) is a unifier, there exists a substitution \( \sigma \) such that \( \varphi = \varphi \circ \sigma \).

We now show that \( R' \) is indeed an instance of \( R \):

\[
R' = (\forall_1(\neg L_1) \setminus \{L_1\}) \cup (\forall_2(\neg L_2) \setminus \{L_2\})
\]

\[
= (\varphi(\forall_1(\neg L_1)) \setminus \{L_1\}) \cup (\varphi(\forall_2(\neg L_2)) \setminus \{L_2\})
\]

\[
= \varphi \left( (\forall_1(\neg L_1) \setminus \{L_1, \ldots, L_m\}) \cup (\forall_2(\neg L_2) \setminus \{L'_1, \ldots, L'_n\}) \right)
\]

(since \( L_1, \ldots, L_m \) are all literals from \( \forall_1(\neg L_1) \) that are mapped to \( \varphi \) by \( \varphi' \))

\[
= \varphi \left( \left( (\forall_1(\neg L_1) \setminus \{L_1, \ldots, L_m\}) \cup (\forall_2(\neg L_2) \setminus \{L'_1, \ldots, L'_n\}) \right) \right)
\]

\[
= \varphi \left( R \right).
\]

Thm 3.4.10 (Soundness + Completeness of Resolution in Pred. Logic)

Let \( \mathcal{H} \) be a finite clause set. Then
$K$ is unsatisfiable iff $\square \in \text{Res}^*(K)$.

Proof: $\Rightarrow$ (Soundness): direct consequence of the resolution lemma 3.4.7.

$\Rightarrow$ (Completeness):

$K$ is unsatisfiable

$\Rightarrow$ by Thm 3.27: Herbrand-expansion of $K$ is also unsatisfiable

The set of clauses containing all ground instances of clauses from $K$, i.e.,

$$\{ c(K) \mid K \in K, c(K) \text{ contains no variables} \}$$

$\Rightarrow$ by Thm 3.37 (completeness of resolution in propositional logic):

One can deduce $\square$ from the Herbrand-exp. of $K$, i.e.,

$\square \in \text{Res}^*(\{ c(K) \mid K \in K, c(K) \text{ has no variables} \})$.

There exists a sequence of ground clauses $K_1', \ldots, K_m'$ with $K_m' = \square$ and

for all $1 \leq i \leq m$:

- $K_i'$ is a ground instance of a clause from $K$ or
- $K_i'$ is resolvent of $K_j'$ and $K_k'$ for $j, k \leq i$}

Now we "lift" this resolution proof to predicate logic, i.e.,
we create a sequence \( K_1, \ldots, K_m \) where each

- \( K_i \) is an instance of \( K_i \) and
- \( K_i \in \text{Res}^g(\Sigma \Xi) \)

\[\text{Definition of } K_i: \]

- if \( K_i \) is a ground instance of some \( K \in \Sigma \Xi \), then
  choose \( K_i := K \)
- otherwise, \( K_i \) is a resolvent of \( K_j \) and \( K_k \) for \( j, k < i \).

We already defined \( K_j, K_k \) such that \( K_j \) and \( K_k \) are instances of \( K_j \) and \( K_k \) resp., and \( K_j, K_k \in \text{Res}^g(\Sigma \Xi) \).

\[
\begin{array}{c}
K_j \quad K_k \\
\downarrow \quad \downarrow \\
K_j' \quad K_k' \\
\downarrow \quad \downarrow \\
K_i \quad K_i' \\
\end{array}
\]

Lifting lemma states that there exists a resolvent of

\( K_j \) and \( K_k \) such that \( K_i' \) is an instance of this

resolvent. Choose \( K_i \) to be this resolvent.

\[\Rightarrow K_i \in \text{Res}^g(\Sigma \Xi)\]

Thus: \( K_m' \) is an instance of \( K_m \) and \( K_m \in \text{Res}^g(\Sigma \Xi) \)

\[\square\]

\[\Rightarrow K_m = \square \text{ and } \square \in \text{Res}^g(\Sigma \Xi).\]

Now we can improve our algorithm to check

entailment by using resolution in pred. logic.

Alg is a semi-decision procedure:
If \( \{y_1, \ldots, y_k\} \neq \emptyset \), then the alg. terminates and returns "true".

If \( \{y_1, \ldots, y_k\} \neq \emptyset \), then the alg. may not terminate. But if it terminates, then it returns "false".

Alg. is feasible in practice for small problems, but still too inefficient in general.

Problem: one has to generate all resolvents repeatedly (one also has to resolve clauses that were created by earlier resolution steps).

Goal: Restrict resolution (i.e., do not create all possible resolvents) without losing completeness.