3.4 Resolution in Predicate Logic

Montag, 15. Mai 2017  08:30

Goal: Resolution on arbitrary clauses (with variables).

Drawback up to now: one has to instantiate variables of the clauses by ground terms in the beginning. This instantiation must be chosen such that it enables all future needed resolution steps.

Ex. 34.1 Clause set
\[
\{ \{ p(x), \neg q(x) \}, \{ \neg p(f(y)) \}, \{ q(f(a)) \} \}
\]

To allow the next resolution step between \{ p(x), \neg q(x) \} and \{ \neg p(f(y)) \}, one has to unify \( p(x) \) and \( p(f(y)) \) (i.e., use an instantiation which makes them equal).

We can use the unifier \( [x/f(y)] \)

\[
\{ p(x), \neg q(x) \} \quad \{ \neg p(f(y)) \} \quad \{ q(f(a)) \}
\]

\[
\begin{array}{c}
\{ p(x), \neg q(x) \} \\
\{ \neg p(f(y)) \} \\
\{ q(f(a)) \} \\
\end{array}
\]

Advantage: If we instantiate \( x \) by a term \( f(y) \) with a variable \( y \), then we can decide later how to instantiate \( y \).

\[
[ y/a ]
\]

If we had required instantiations by ground terms, then we would have to instantiate \( x \) by \( f(a) \) in the first resolution step, because
Def 3.4.2 (Unification)  
A clause \( K = \{ L_1, \ldots, L_n \} \) is unifiable \textit{iff} there exists a substitution \( \sigma \) with \( \sigma(L_1) = \cdots = \sigma(L_n) \) (i.e., \(|\sigma(K)| = 1\)). Such a subst. is called a unifier of \( K \).

A unifier \( \sigma \) is most general unifier (mgu) of \( K \) \textit{iff} for every unifier \( \sigma' \) of \( K \) there exists a substitution \( \delta \) such that \( \sigma' = \delta \circ \sigma \).

Function Composition: first apply \( \delta \), then apply \( \sigma \).

\[ \sigma = \{ X / f( Y ) \} \]

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

\[ \sigma' = \delta \circ \sigma \]

for \( \delta = \{ Y / a \} \)

If a clause is unifiable, then it also has a mgu. The mgu is unique up to variable renaming.

\[ K = \{ p( X ) , p( f( Y ) ) \} \]

\[ \text{mgu} \ \sigma_1 = \{ X / Y \} \]
\[ \sigma_2 = \{ Y/X \} \quad \text{equal up to variable renaming} \]

First unification algorithm by J. Robinson (1965)

**Ex 343 Illustrate Unification Algorithm**

(a) \( \{ q(f(X,Y)), q(g(X,Y)) \} \) \( \uparrow \) \( \text{Clash Failure} \)

(b) \( \{ q(X), q(h(X)) \} \) \( \uparrow \) \( \rightarrow \) \( \text{Occur Failure} \)

\( X \) occurs in \( h(X) \)

(c) \( \{ r(f(f(X,Y), g(X)), h(f(U,V), W)), h(f(U,V)) \} \)

\( \uparrow \) \( \rightarrow \) \( \rightarrow \) \( \rightarrow \)

\( \sigma = \{ z/f(U,V) \} \)

\( \{ r(f(f(U,V), g(X)), h(f(U,V))), r(f(f(U,V), W), h(f(U,V))) \} \)

\( \rightarrow \) \( \rightarrow \) \( \rightarrow \)

\( \sigma = \{ \{ W/g(q,a,Y) \} \circ \{ z/f(U,V) \} \}

= \{ \{ W/g(q,a,Y), z/f(U,V) \} \}

\( \{ r(f(f(U,V), g(X)), h(f(U,V))), r(f(f(U,V), g(X)), h(f(a,Y))) \} \)

\( \uparrow \) \( \rightarrow \) \( \rightarrow \) \( \rightarrow \)

\( \sigma = \{ U/a \} \circ \{ \{ W/g(q,a,Y), z/f(U,V) \} \}

= \{ \{ U/a, W/g(a,Y), z/f(q,v) \} \} \)
\{ \neg \forall (f(a, V), g(a, Y)), h(f(a, V)) \uparrow, \\
\neg \forall (f(a, V), g(a, Y)), h(f(a, Y)) \uparrow \}
\sigma = \{ Y / V \} \circ \{ U / a, W / g(a, Y), Z / f(a, V) \}
= \{ Y N, U / a, W / g(a, V), Z / f(a, V) \}

This is the mgu.

**Theorem 3.44 (Termination and Soundness of UniFAlg)**
The unification alg. terminates for every clause \( \mathcal{K} \neq \emptyset \)
and it is sound, i.e., it computes a mgu for \( \mathcal{K} \)
iff \( \mathcal{K} \) is unifiable.

**Proof:** Alg. terminates because every iteration of the
loop (Steps 2-6) removes one variable from \( \sigma(\mathcal{K}) \).

If alg. terminates with success, then \( |\sigma(\mathcal{K})| = \nu \geq 0 \)
\( \sigma \) is a unifier of \( \mathcal{K} \).

Thus: if \( \mathcal{K} \) is not unifiable, alg. terminates with
clash or occur failure

It remains to show:
If \( \mathcal{K} \) is unifiable \( \exists \) alg. finds a unifier \( \sigma \) and
\( \sigma \) is mgu.

Let \( m \) be the number of loop iterations that the alg.
performs for clause \( \mathcal{K} \).
For all $0 \leq i \leq m$, let $\sigma_i$ be the value of $\sigma$ after the $i$-th loop iteration.

We show the following for all $0 \leq i \leq m$:

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

If (8) holds, then the alg. cannot stop with failure.

The reason is that then $\sigma_m(K)$ would not be unifiable.

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

(8)

$\sigma_m(K)$ is unifiable.

If (8) holds, then alg. returns a mgu:

We must have $|\sigma_m(K)| = 1$, i.e., $\sigma_m$ is a unifier of $K$.

By (8): for every unifier $\sigma'$, we have $\sigma' = \sigma' \circ \sigma_m$

$\sigma_m$ is mgu.

It remains to show the following for all $0 \leq i \leq m$:

for all unifiers $\sigma'$ of $K$, we have: $\sigma' = \sigma' \circ \sigma_i$. (8)

We prove (8) by induction on $i$:

Ind. Base: $i = 0$

$\sigma_0 = \sigma$. Thus: $\sigma' = \sigma' \circ \sigma$ holds for any $\sigma'$.

Ind. Step: $i > 0$

$\sigma_i = \{X/t\} \circ \sigma_{i-1}$
By the ind. hyp: $\sigma' = \sigma' \circ \sigma_{i-1}$.

Therefore, we have:

$$
\sigma' \circ \sigma_i = \sigma' \circ (X/t) \circ \sigma_{i-1}
$$

by def. of $\sigma_i$

$$
= \sigma' \circ \sigma_{i-1}
$$

since $\sigma' \circ (X/t) = \sigma'$

by the ind. hyp

Reason for $\sigma' \circ (X/t) = \sigma'$:

For $Y \neq X$, we clearly have $(\sigma' \circ (X/t))(Y) = \sigma'(Y)$.

For $X$:

$$(\sigma' \circ (X/t))(X) = \sigma'(t) = \sigma'(X)$$

Since $\sigma'$ is unifier of $\sigma_{i-1}(K)$ and every unifier of $\sigma_{i-1}(K)$ must make $X$ and $t$ equal,

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

Def 3.4.5 (Resolution in Pred. Logic)

Let $K_1$, $K_2$ be clauses. Then $R$ is a resolvent of $K_1$ and $K_2$ iff

- There exist variable renamings $\nu_1, \nu_2$ such that $\nu_1(K_1)$ and $\nu_2(K_2)$ have no
**Common Variables.**

- There are $L_1, ..., L_m \in \mathcal{P}(U_1)$ and $L'_1, ..., L'_n \in \mathcal{P}(U_2)$ such that
  
  \[
  \{ \overline{L_1}, ..., \overline{L_m}, L'_1, ..., L'_n \}
  \]
  
  is unsatisifiable with a mgu $\sigma$.

- $R = \sigma(L_1) \Delta \mathcal{P}(U_1) \setminus \{ L_1, ..., L_m \} \cup \mathcal{P}(U_2) \setminus \{ L'_1, ..., L'_n \}$

For a clause set $\mathcal{K}$, we define:

\[
\text{Res}^0(\mathcal{K}) = \mathcal{K}
\]

\[
\text{Res}^n(\mathcal{K}) = \text{Res}(\text{Res}^{n-1}(\mathcal{K})) 
\text{ for all } n \geq 0
\]

\[
\text{Res}(\mathcal{K}) = \bigcup_{n \geq 0} \text{Res}^n(\mathcal{K})
\]

For clauses without variables, this definition is equivalent to the definition of propositional resolution.

**Ex. 346**

\[
\begin{align*}
\{ p(f(X)), \neg q(2), p(2) \} & \quad \text{results in} \quad \{ \neg p(X), \neg g(X) \} \\
\overline{L_1} & \quad \text{and} \quad \overline{L_2} \\
\sigma_1 = \emptyset & \quad \text{and} \quad \sigma_2 = \{ X / U, U / X \}
\end{align*}
\]
Propositional resolution is sound and complete.

Now we want to prove soundness + completeness of resolution in pred. logic:

X is unsatisfiable iff ∅ ∈ Res*(X)

"∈": soundness
"→": completeness

For soundness, we need a similar resolution lemma as in prop. logic:

Lemma 3.47. (Resolution Lemma in Pred. Logic)
Let X be a set of clauses. If X₁, X₂ ∈ X and
R is resolvent of \( K_1 \) and \( K_2 \), then
\( K \) and \( K \cup \{ R \} \) are equivalent.

Proof: Similar as for prop. logic \( \Box \)

This suffices for soundness:
\[
\square \in \text{Res}^n(K) = \bigcup_{n \geq 0} \text{Res}^n(K) \quad \land \quad \text{there is an } n_0 \text{ with } \\
\square \in \text{Res}^{n_0}(K).
\]

By the resolution lemma: \( K \) is equivalent to \( \text{Res}(K) \)

\( \neg \square \text{ induction} \) for all \( n \geq 0 \)

Since \( \text{Res}^{n_0}(K) \) is unsatisfiable, \( K \) is unsat. as well.

Now we prove completeness.

Idea: We know that propositional resolution is complete. Let us re-use this observation by "lifting" propositional resolution proofs to resolution proofs in pred. logic.

Lemma 348 (Lifting Lemma)

Let \( K_1, K_2 \) be two clauses, let \( K_1', K_2' \) be ground instances of \( K_1 \) and \( K_2 \). 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 $R$. 

\[
\begin{array}{c}
K_1 \quad k_2 \\
\downarrow \text{gr. inst.} \quad \downarrow \text{gr. inst.} \\
K_1' \quad k_2' \\
\text{implies} \\
R' \\
\end{array}
\begin{array}{c}
K_1 \\
\downarrow \text{gr. inst.} \\
K_1' \\
\end{array}
\begin{array}{c}
k_2 \\
\end{array}
\]

Ex. 34.9.

\[
\{ p(f(X)), q(f(a)), r(g(f(a))) \} \quad \{ \neg p(X), r(g(X)) \}
\]

\[
\begin{array}{c}
\downarrow \text{gr. inst.} \quad \downarrow \text{gr. inst.} \\
\{ p(f(a)), q(f(a)) \} \\
\{ \neg p(f(a)), r(g(f(a))) \} \\
\end{array}
\begin{array}{c}
R' = \{ \neg q(f(a)), r(g(f(a))) \}
\end{array}
\]

**Proof of the lifting lemma 3.4.8:**

Let $\sigma_1$, $\sigma_2$ be variable renamings such that $\sigma_1(K_1)$ and $\sigma_2(K_2)$ are variable-disjoint. Then we can use the same ground subst. $\sigma$ to obtain

\[
K_1' = \sigma(\sigma_1(K_1)) \quad \text{and} \quad K_2' = \sigma(\sigma_2(K_2)).
\]

Since $R'$ is resolvent of $K_1'$ and $K_2'$, there is a literal with $\overline{L} \in K_1'$ and $\overline{L} \in K_2'$ such that

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

Let $L_1, \ldots, L_m \in \sigma_1(K_1)$ be all literals where

\[
\sigma(L_1) = \ldots = \sigma(L_m) = \overline{L}.
\]

Let $L_1', \ldots, L_n' \in \sigma_2(K_2)$ be all literals where

\[
\sigma(L_1') = \ldots = \sigma(L_n') = \overline{L}.
\]
So $\psi$ is a unifier of $\{\overline{L_1}, \ldots, \overline{L_m}, L_1', \ldots, L_n'\}$. Since this clause is unifiable, it also has a unifier $\psi'$.
Thus, there exists a subst. $\sigma$ with $\psi = \psi' \circ \sigma$.

$\varrho_1$ and $\varrho_2$ have the resolvent
$$R = \psi'\left((\varrho_1(\varrho_1) \setminus \{L_1, \ldots, L_m\}) \cup (\varrho_2(\varrho_2) \setminus \{L_1', \ldots, L_n'\})\right).$$
It remains to show that $R'$ is a ground instance of $R$.
$$R' = (\varrho_1' \setminus \{L_3\}) \cup (\varrho_2' \setminus \{L_3\})$$
$$= (\varrho_1(\varrho_1) \setminus \{L_3\}) \cup (\varrho_2(\varrho_2) \setminus \{L_3\})$$
$$= \psi\left((\varrho_1(\varrho_1) \setminus \{L_1, \ldots, L_m\}) \cup (\varrho_2(\varrho_2) \setminus \{L_1', \ldots, L_n'\})\right)$$
$$= \psi\left((\varrho_1' \setminus \{L_1, \ldots, L_m\}) \cup (\varrho_2'(\varrho_2) \setminus \{L_1', \ldots, L_n'\})\right)$$
$$= \psi (\psi')$$

\[\square\]

**Thm 3.4.10.** (Soundness + Completeness for Resolution in Red. Logic)

Let $\mathcal{X}$ be a set of clauses.
Then $\mathcal{X}$ is unsatisfiable iff $\Box \in \text{Res}^*(\mathcal{X})$.

**Proof:** $\Leftarrow$ (Soundness), see above.
$\Rightarrow$ (Completeness)
$\mathcal{X}$ unsatisfiable
\[\text{Herbrand} - \text{expansion } E(\mathcal{X}) \text{ is unsatisfiable (Thm 3.2.7)}\]
i.e., there is a finite set of ground instances of the clauses in \( \mathcal{K} \) that is unsatisfiable.

By completeness of propositional resolution (Thm. 3.3.7) one can derive \( \Box \) from these ground instances.

This, there is a sequence of ground clauses 
\[ K_1', ..., K_m' \] 
where for all \( 1 \leq i \leq m \) we have:

- \( K_i' \) is a ground instance of a clause \( K \in \mathcal{K} \) or
- \( K_i' \) is a resolvent of \( K_{i_1} \) and \( K_{i_2} \) for \( i_1, i_2 < i \).

With the lifting lemma 3.4.8 we now construct a sequence of clauses \( K_1', ..., K_m \) where \( K_i' \) is a ground instance of \( K_i \) (for all \( 1 \leq i \leq m \)) and all \( K_i \in \text{Res}^*(\mathcal{K}) \):

- If \( K_i' \) is a ground instance of some \( K \in \mathcal{K} \), then we choose \( K_i := K \).

- If \( K_i' \) is a resolvent of \( K_{i_1} \) and \( K_{i_2} \) and we have already constructed \( K_{i_1}, K_{i_2} \) such that 
  \( K_{i_1}' \) and \( K_{i_2}' \) are ground instances of \( K_{i_1}, K_{i_2} \),

  \[ K_i' \leftarrow \begin{array}{c}
    \text{lifting lemma} \\
    \text{we choose \( K_i \) to be the resolvent of} \\
    \text{\( K_{i_1}' \) and \( K_{i_2}' \) such that \( K_i' \) is a} \\
    \text{ground inst. of \( K_i \)}
  \end{array} \]
Since $\mathbf{K}_m' = \emptyset$ is a ground instance of $\mathbf{K}_m$, we have $\mathbf{K}_m = \emptyset$.

Thus: $\mathbf{K}_1, \ldots, \mathbf{K}_m = \emptyset$ is a resolution proof in

pred. logic. cographic shows $\emptyset \in \text{Res}^* \ (\forall)$. 

Now we can improve the algorithm to check

$\{ \varphi_1, \ldots, \varphi_n \} \vdash \varphi$.

1. Attempt Gilmore’s Alg.

2. Attempt Ground Resolution Alg.

3. Attempt Resolution Alg. also solves (a) 

Drawbacks:

(a) How to instantiate variables?

(b) How to check unsat.
in prop. logic?

Alg. is again a semi-decision procedure:

if $\{ \varphi_1, \ldots, \varphi_n \} \not\vdash \varphi$,

then the alg. could be non-terminating.

Problem: Alg. computes all possible resolution steps.

To improve efficiency, it would be desirable to

restrict the possible resolution steps.
without losing completeness.