Lecture 12

Resolution for predicate logic

Unification, resolution

Print version of the lecture in *Introduction to Logic for Computer Science*

presented by Prof Hongseok Yang

These lecture notes are very minor variants of the ones made by Prof James Worrell and Prof Christoph Haase for their 'Logic and Proof' course at Oxford.

A serious drawback of the ground resolution procedure is that it requires looking ahead to predict which ground instances of clauses will be needed in a proof. In this lecture, we introduce the predicate-logic version of resolution, which allows us to perform substitution "by need". This relies on the notion of unification, which we introduce next.

1 Unification

A *substitution* is a function θ from the set of σ-terms back to itself such that (writing function application on the right) $c\theta = c$ for each constant symbol c and $f(t_1, \ldots, t_k)\theta = f(t_1\theta, \ldots, t_k\theta)$ for each k-ary function symbol f. It is clear that the composition of two such substitutions (as functions) is also a substitution. We have previously considered substitutions of the form $[t/x]$ for a σ-term t and a variable x.

We write composition of substitutions diagrammatically, that is, $\theta \cdot \theta'$ denotes the substitution obtained by applying θ first and then θ'. (This convention matches the fact that for substitutions, we write function application on the right.) In particular, $[t_1/x_1] \cdots [t_k/x_x]$ denotes the substitution obtained by sequentially applying the substitutions $[t_1/x_1], \ldots, [t_k/x_k]$ left-to-right.

Given a set of literals $D = \{L_1, \ldots, L_k\}$ and a substitution θ, define $D\theta := \{L_1\theta, \ldots, L_k\theta\}$. We say that θ *unifies* D if $D\theta = \{L\}$ for some literal L. For example, the substitution $\theta = [f(a)/x][a/y]$ unifies $\{P(x), P(f(y))\}$, as does the substitution $\theta' = [f(y)/x]$. In this example, we regard θ' as a *more general unifier* because $\theta = \theta' \cdot [a/y]$, that is, θ factors through θ'.

We say that θ is a *most general unifier* of a set of literals D if θ is a unifier of D and any other unifier θ' factors through θ, i.e., we have $\theta' = \theta \cdot \theta''$ for some substitution θ''. Note that both the substitutions $[x/y]$ and $[y/x]$ are most general unifiers of $\{P(x), P(y)\}$. In fact, most-general unifiers are only unique up to renaming variables.

We will show that a set of literals either has no unifier or it has a most general unifier. Examples of sets of literals that cannot be unified are $\{P(f(x)), P(g(x))\}$ and $\{P(f(x)), P(x)\}$. The problem in the second case is that we cannot unify a variable x and term t if x occurs in t.

Theorem 1 (Unification Theorem)**.** *A unifiable set of literals D has a most general unifier.*

Proof. We claim that the following algorithm determines whether a set of literals has a unifier and, if so, outputs a most general unifier.

Unification Algorithm
Input: Set of literals D
Output: Either a most general unifier of D or "fail"

$\theta :=$ identity substitution
while θ is not a unifier of D **do**
begin
  pick two distinct literals in $D\theta$ and find the left-most positions at which they differ
  if one of the corresponding sub-terms is a variable x
    and the other term t does not contain x
  then $\theta := \theta \cdot [t/x]$ **else** output "fail" and halt
end
return θ

We argue termination as follows. In any iteration of the while loop that does not cause the program to halt, a variable x is replaced everywhere in $D\theta$ by a term t that does not contain x. Thus, the number of different variables occurring in $D\theta$ decreases by one in each iteration, and the loop must terminate.

The loop invariant is that for any unifier θ' of D, we have $\theta' = \theta \cdot \theta'$. Clearly, the invariant is established by the initial assignment of the identity substitution to θ. To see that the invariant is maintained by an iteration of the loop, suppose we find an occurrence of variable x in a literal in $D\theta$ such that a different term t occurs in the same position in another literal in $D\theta$. From the invariant, we know that θ' is a unifier of $D\theta$, and thus $t\theta' = x\theta'$. It immediately follows that $\theta' = [t/x] \cdot \theta'$. Thus, the loop invariant is maintained by the assignment $\theta := \theta \cdot [t/x]$.

The termination condition of the while loop is that θ is a unifier of D. In conjunction with the loop invariant, this implies that the final value of θ is a most general unifier of D. Finally, the invariant implies that if θ' is a unifier of D, then it is also a unifier of $D\theta$. But the algorithm only outputs "fail" if $D\theta$ has no unifier, in which case D has no unifier. $\square$

Example 2. Consider an execution of the unification algorithm on input $D = \{P(x,y), P(f(z),x)\}$. Scanning left-to-right, the leftmost discrepancy is underlined in $\{P(\underline{x},y), P(\underline{f}(z),x)\}$. Applying the substitution $[f(z)/x]$ to D yields the set $D' = \{P(f(z),\underline{y}), P(f(z),\underline{f}(z))\}$, where the underlined positions again indicate the left-most discrepancy. Applying the substitution $[f(z)/y]$ to D' yields the singleton set $\{P(f(z),f(z))\}$. Thus $[f(z)/x][f(z)/y]$ is a most general unifier of the set D.

2 Resolution

First-order resolution operates on a set of clauses, that is, a set of sets of literals. Given a formula $\forall x_1 \ldots \forall x_n F$ in Skolem form, we perform resolution on the clauses in the matrix F with the goal of deriving the empty clause. Although quantifiers do not explicitly appear in resolution proofs, we can see the variables in such a proof as being implicitly universally quantified. This is made more formal when we formulate the Resolution Lemma in the next section.

For any set of literals D, let $\overline{D}$ denote the set of complementary literals. For example, if $D = \{\neg P(x), R(x,y)\}$ then $\overline{D} = \{P(x), \neg R(x,y)\}$.

Definition 3 (Resolution)**.** Let C_1 and C_2 be clauses *with no variable in common.* We say that a clause R is a *resolvent* of C_1 and C_2 if there are sets of literals $D_1 \subseteq C_1$ and $D_2 \subseteq C_2$ such that $D_1 \cup \overline{D_2}$ has a most general unifier θ, and

$$R = (C_1\theta \setminus \{L\}) \cup (C_2\theta \setminus \{\overline{L}\}), \tag{1}$$

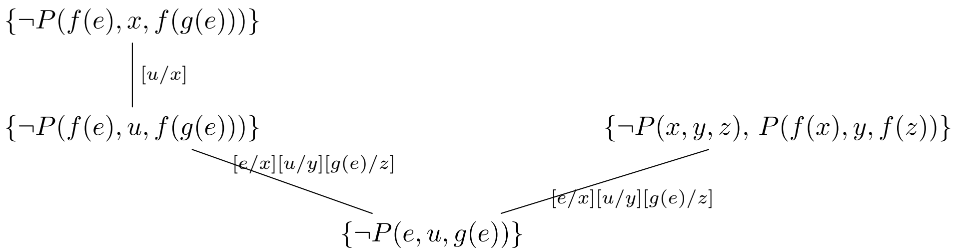

Figure 1: First-order resolution example

where $\{L\} = D_1\theta$ and $\{\overline{L}\} = D_2\theta$. More generally, if C_1 and C_2 are arbitrary clauses, we say that R is a resolvent of C_1 and C_2 if there are variable renamings θ_1 and θ_2 such that $C_1\theta_1$ and $C_2\theta_2$ have no variable in common, and R is a resolvent of $C_1\theta_1$ and $C_2\theta_2$ according to the definition above.

Example 4. Consider a signature with constant symbol e, unary function symbols f and g, and a ternary predicate symbol P. We compute a resolvent of the clauses $C_1 = \{\neg P(f(e), x, f(g(e)))\}$ and $C_2 = \{\neg P(x, y, z), P(f(x), y, f(z))\}$ as follows (see Figure 1). First apply the substitution $[u/x]$ to C_1, obtaining a clause C_1' that has no variable in common in C_2. Now unify complementary literals under the substitution $[e/x][u/y][g(e)/z]$, and obtain the resolvent $\{\neg P(e, u, g(e))\}$.

A *predicate-logic resolution derivation* of a clause C from a set of clauses F is a sequence of clauses $C_1, \ldots, C_m$, with $C_m = C$ such that each C_i is either a clause of F (possibly with the variables renamed) or follows by a resolution step from two preceding clauses C_j, C_k, with $j, k < i$. We write $\mathrm{Res}^*(F)$ for the set of clauses C such that there is a derivation of C from F.

Example 5. Consider the following sentences over a signature with ternary predicate symbol A, constant symbol e, and unary function symbol s. The idea is that A represents the ternary addition relation, e the zero element, and s the successor function.

$$
\begin{aligned}
F_1 &: \ \forall x\, A(e, x, x)\\
F_2 &: \ \forall x \forall y \forall z\, (\neg A(x, y, z) \vee A(s(x), y, s(z)))\\
F_3 &: \ \forall x \exists y\, A(s(s(e)), x, y)
\end{aligned}
$$

We use first-order resolution to show that $F_1 \wedge F_2 \models F_3$, that is, we show that $F_1 \wedge F_2 \wedge \neg F_3$ is unsatisfiable. We proceed in two steps.

Step (i): separately Skolemise each formula.

Formula $\neg F_3$ is equivalent to

$$\exists y \forall z\, \neg A(s(s(e)), y, z).$$

Skolemising the above formula, we obtain the formula $G_3 := \forall z\, \neg A(s(s(e)), c, z)$, where c is a new constant symbol. Now $F_1 \wedge F_2 \wedge G_3$ is equisatisfiable with $F_1 \wedge F_2 \wedge \neg F_3$, and so it suffices to give a resolution refutation of $F_1 \wedge F_2 \wedge G_3$.[1] **Important:** When Skolemising more than one formula separately, different formulas may not introduce the same constant symbols. Always make sure to choose fresh constant symbols for every formula you are Skolemising.

[1]Formally, the notion of a resolution proof assumes a single Skolem-form formula. So strictly speaking, the proof below is a resolution refutation of the formula $\forall x \forall y \forall z (A(e, x, x) \wedge ((\neg A(x, y, z) \vee A(s(x), y, s(z))) \wedge \neg A(s(s(e)), c, z))$, which is logically equivalent to $F_1 \wedge F_2 \wedge G_3$.

Step (ii). derive the empty clause using resolution. The proof is as follows. Note that in order to always ensure that we resolve clauses with disjoint variables, we arrange it so that the variables in line k of the proof are subscripted with k. In particular, we add a variable renaming at the end of each unifying substitution so that the variables in the output formula have the right subscript for the next line of the proof.

1.	$\{\neg A(s(s(e)), c, z_1)\}$	clause of G_3
2.	$\{\neg A(x_2, y_2, z_2), A(s(x_2), y_2, s(z_2))\}$	clause of F_2
3.	$\{\neg A(s(e), c, z_3)\}$	1,2 Res. Sub $[s(e)/x_2][c/y_2][s(z_2)/z_1][z_3/z_2]$
4.	$\{\neg A(e, c, z_4)\}$	2,3 Res. Sub $[e/x_2][c/y_2][s(z_2)/z_3][z_4/z_2]$
5.	$\{A(e, y_5, y_5)\}$	clause of F_1
6.	$\Box$	4,5 Res. Sub $[c/y_5][c/z_4]$

Given a formula H with free variables $x_1, x_2, \ldots, x_n$, its *universal closure* $\forall^* H$ is the sentence $\forall x_1 \forall x_2 \ldots \forall x_n H$. The following lemma is key to the soundness of resolution.

Lemma 6 (Resolution Lemma). *Let $F = \forall x_1 \ldots \forall x_n G$ be a closed formula in Skolem form, with G quantifier-free and in CNF. Let R be a resolvent of two clauses in G. Then, $F \equiv \forall^*(G \cup \{R\})$.*

Proof. Clearly $\forall^*(G \cup \{R\}) \models F$. The non-trivial direction is to show that $F \models \forall^* R$. For this, since F is closed, it suffices to show that $F \models R$. (Check that you understand why this is so!)

To this end, suppose that R is a resolvent of clauses $C_1, C_2 \in G$, with $R = (C_1\theta \setminus \{L\}) \cup (C_2\theta' \setminus \{\overline{L}\})$ for some substitutions θ, θ' and complementary literals $L \in C_1\theta$ and $\overline{L} \in C_2\theta'$.

Let $\mathcal{A}$ be an assignment that satisfies $F = \forall^* G$. Since $C_1, C_2 \in G$, by the Translation Lemma $\mathcal{A} \models C_1\theta$ and $\mathcal{A} \models C_2\theta'$. Moreover, since $\mathcal{A}$ satisfies at most one of the complementary literals L and $\overline{L}$, it follows that $\mathcal{A}$ satisfies at least one of $C_1\theta \setminus \{L\}$ and $C_2\theta' \setminus \{\overline{L}\}$. We conclude that $\mathcal{A}$ satisfies R, as required. $\Box$

Corollary 7 (Soundness). *Let $F = \forall x_1 \ldots \forall x_n G$ be a closed formula in Skolem form. Let clause C be obtained from G by a resolution derivation. Then, $F \equiv \forall^*(G \cup C)$.*

Proof. Induction on the length of the resolution derivation, using the Resolution Lemma for the induction step. $\Box$

A Refutation Completeness

In this appendix we prove the refutation completeness of predicate-logic resolution proofs by showing that ground resolution proofs lift to predicate-logic resolution proofs. The proofs here are more technical and can be regarded as optional.

Lemma 8 (Lifting Lemma). *Let C_1 and C_2 be clauses with respective ground instances G_1 and G_2. Suppose that R is a propositional resolvent of G_1 and G_2. Then, C_1 and C_2 have a predicate-logic resolvent R' such that R is a ground instance of R'.*

Proof. The situation of the lemma is shown in Figure 2. We can write the ground resolvent R in the form $R = (G_1 \setminus \{L\}) \cup (G_2 \setminus \{\overline{L}\})$, for complementary literals $L \in G_1$ and $\overline{L} \in G_2$.

Let C_1' and C_2' be variable-disjoint renamings of C_1 and C_2, cf. Figure 2. Then, G_1 and G_2 are also ground instances of C_1' and C_2'. Thus we can write $G_1 = C_1'\theta'$ and $G_2 = C_2'\theta'$ for some ground substitution θ'. Let $D_1 \subseteq C_1'$ be the set of literals mapped to the literal L by θ' and let $D_2 \subseteq C_2'$ be the set of literals mapped to the

$$
\begin{tikzcd}
C_1 \arrow[d, dash] & & C_2 \arrow[d, dash] & & C_1 \arrow[d, dash] & & C_2 \arrow[d, dash] \\
G_1 \arrow[dr, dash] & & G_2 \arrow[dl, dash] & & C_1' \arrow[dr, dash] & & C_2' \arrow[dl, dash] \\
& R & & & & R' &
\end{tikzcd}
$$

Figure 2: Ground resolution step on the left, and its predicate-logic lifting on the right.

literal $\overline{L}$ by θ'. Then, θ' is a unifier of $D_1 \cup \overline{D_2}$. Writing θ for the most general unifier of $D_1 \cup D_2$, we have that

$$R' := (C_1'\theta \setminus D_1\theta) \cup (C_2'\theta \setminus D_2\theta) \tag{2}$$

is a predicate-logic resolvent of C_1 and C_2.

By the definition of most general unifier, there exists some θ'' such that $\theta' = \theta{\cdot}\theta''$. Thus we have

$$G_1 = C_1'\theta' = C_1'(\theta \cdot \theta'') = (C_1'\theta)\,\theta'' \;\text{ and }\; G_2 = C_2'\theta' = C_2'(\theta \cdot \theta'') = (C_2'\theta)\,\theta''.$$

Now from (2) we have that

$$
\begin{aligned}
R'\theta'' &= \Big((C_1'\theta \setminus D_1\theta) \cup (C_2'\theta \setminus D_2\theta)\Big)\,\theta'' \\
&= \Big((C_1'\theta)\,\theta'' \setminus (D_1\theta)\,\theta''\Big) \cup \Big((C_2'\theta)\,\theta'' \setminus (D_2\theta)\,\theta''\Big) \\
&= \Big(C_1'(\theta \cdot \theta'') \setminus D_1(\theta \cdot \theta'')\Big) \cup \Big(C_2'(\theta \cdot \theta'') \setminus D_2(\theta \cdot \theta'')\Big) \\
&= (G_1 \setminus \{L\}) \cup (G_2 \setminus \{\overline{L}\}).
\end{aligned}
$$

(Note that the last equality uses the fact that D_1 is precisely the set of literals in C_1' that map to L under θ' and similarly D_2 is precisely the set of literals in C_2' that map to $\overline{L}$ under θ'.) We conclude that R is a ground instance of R' under the substitution θ''.

$\square$

Corollary 9 (Completeness)**.** *Let F be a closed formula in Skolem form with its matrix F' in CNF. If F is unsatisfiable then there is a predicate-logic resolution proof of $\square$ from F'.*

Proof. Suppose F is unsatisfiable. By the completeness of ground resolution there is a proof $C_1', C_2', \ldots, C_n'$, where $C_n' = \square$ and each C_i' is either a ground instance of a clause in F' or is a resolvent of two clauses C_j', C_k' for $j, k < i$. We inductively define a corresponding predicate-logic resolution proof $C_1, C_2, \ldots, C_n$, such that C_i' is a ground instance of C_i. For each i, if C_i' is a ground instance of a clause $C \in F'$ then define $C_i = C$. On the other hand, suppose that C_i' is a resolvent of two ground clauses C_j', C_k', with $j, k < i$. By induction we have constructed clauses C_j and C_k such that C_j' is a ground instance of C_j and C_k' is a ground instance of C_k. By the Lifting Lemma we can find a clause C_i which is a resolvent of C_j and C_k such that C_i' is a ground instance of C_i. $\square$