Logic and Proof Hilary 2024

Resolution for Predicate Logic

James Worrell

1 Unification

A drawback of the ground resolution procedure is that it requires predicting which ground instances of clauses will be needed in a proof. In this lecture we introduce a version of resolution that allows us to perform substitution "by need". This relies on the notion of *unification*.

Substitutions. A *substitution* is a selfmap θ on the set of σ-terms 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. A substitution is thus determined by its action on variables. We denote by $[t/x]$ the substitution that maps the variable x to the term t and leaves all other variables unchanged. It is clear that the composition of two substitutions is a substitution. We write composition diagrammatically, that is, $\theta\theta'$ 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, $[t_1/x_1]\cdots[t_k/x_k]$ denotes the substitution obtained by sequentially applying the substitutions $[t_1/x_1], \ldots, [t_k/x_k]$ left-to-right.

Term Equations. A *term equation* is an ordered pair of terms $s \stackrel{?}{=} t$. A substitution θ is a *unifier* of a system of term equations $\{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$ if $s_i\theta = t_i\theta$ for all $i \in \{1, \ldots, n\}$. We further say that θ is a *most general unifier (mgu)* if any other unifier θ' factors through θ, i.e., we have $\theta' = \theta\theta''$ for some substitution θ''. For example, the substitution $\theta = [f(a)/x][a/y]$ unifies $x \stackrel{?}{=} f(y)$, as does the substitution $\theta' = [f(y)/x]$. Here θ' is an mgu and $\theta = \theta'[a/y]$, that is, θ factors through θ'. Note that both the substitutions $[x/y]$ and $[y/x]$ are both mgu's of the equation $x \stackrel{?}{=} y$. In fact, mgu's are only unique up to renaming variables. The term equation $f(x) \stackrel{?}{=} g(a)$, where f and g are different unary function symbols, has no unifier. Likewise the equation $x \stackrel{?}{=} f(x)$ has no unifier. A system S is *solved* if it is in the form $S = \{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$ where the x_i are distinct variables that do not appear in any term t_j. For such a solved form S the substitution $\theta_S := [t_1/x_1]\cdots[t_n/x_n]$ is well-defined and is an mgu; indeed, for any unifier θ of S we have $\theta = \theta_S\theta$.

Unifying Sets of Literals. The notion of an mgu can be lifted from terms to literals. For a literal L and substitution θ, we write $L\theta$ for the literal obtained by applying θ to each term appearing in L. Given a set of literals $D = \{L_1, \ldots, L_k\}$ we say that θ *unifies* D if $L_1\theta = \cdots = L_k\theta$. We say moreover that θ is a most general unifier if any other unifier factors through θ.

An mgu of a set of literals can be obtained by solving an appropriate set of term equations. Consider the set of literals $D := \{P(f(x), u), P(y, y), P(y, u)\}$. An mgu of D is an mgu of the system of equations $S := \{f(x) \stackrel{?}{=} y, y \stackrel{?}{=} y, u \stackrel{?}{=} y\}$. In the case at hand an mgu is $[f(x)/y][f(x)/u]$.

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.

1.1 Martelli and Montanari's Unification Algorithm.

We present an abstract form of the unification algorithm as a family of rewrite rules that can be applied non-deterministically to transform systems of equations into solved form or $\bot$, representing an unsatisfiable system. By convention we allow f and g in the rules **Decompose** and **Conflict** to be constant symbols (considered as nullary function symbols); e.g., an instance **Conflict** with $m = n = 0$ would be $\{a \stackrel{?}{=} b\} \Longrightarrow \bot$ for distinct constant symbols a and b.

- **Simplify:** $\{x \stackrel{?}{=} x\} \cup S \Longrightarrow S$ for any variable x
- **Swap:** $\{t \stackrel{?}{=} x\} \cup S \Longrightarrow \{x \stackrel{?}{=} t\} \cup S$ if t is not a variable
- **Decompose:** $\{f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n)\} \cup S \Longrightarrow \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup S$
- **Conflict:** $\{f(s_1, \ldots, s_m) \stackrel{?}{=} g(t_1, \ldots, t_n)\} \cup S \Longrightarrow \bot$ if $f \neq g$
- **Elim:** $\{x \stackrel{?}{=} t\} \cup S \Longrightarrow \{x \stackrel{?}{=} t\} \cup S[t/x]$ if x occurs in S and not in t
- **Occur:** $\{x \stackrel{?}{=} t\} \cup S \Longrightarrow \bot$ if x is a proper subterm of t.

The following proposition shows that the above rewriting system is terminating and that the order in which the rules are applied does not matter.

Proposition 1. Given a system S of term equations, there is no infinite sequence of rewrites $S = S_1 \Longrightarrow S_2 \Longrightarrow S_3 \Longrightarrow \cdots$. A maximal chain of rewrites starting from S either ends in $\bot$ or in a solved system T. In the first case we have that S has no unifier whereas in the latter case θ_T is a mgu of S.

Proof. We note that that the set $\mathbb{N}^3$ is well-ordered under the lexicographic order (i.e., there are no infinite decreasing chains). Say that a variable x is *solved* in a system S if it appears once in S with the single occurrence being in an equation of the form $x \stackrel{?}{=} t$. We define the *rank* of an equation system S to be the triple $(n_1, n_2, n_3) \in \mathbb{N}^3$, where n_1 is the number of variables in S that are not solved, n_2 is the total size of all terms occurring in S, and n_3 is the number of equations in S of the form $t \stackrel{?}{=} x$ with t not a variable. Then each rule that doesn't lead immediately to termination decreases the rank of a system. Specifically, **Elim** decreases n_1, while both **Decompose** and **Simplify** do not increase n_1 and decrease n_2, and **Swap** increases neither n_1 nor n_2 and decreases n_3. This proves termination.

On termination we either have $\bot$ or a solved system. It remains to observe that each rule preserves the set of unifiers of the system. We consider just the rule **Elim** by way of example. If θ is a solution of $\{x \stackrel{?}{=} t\}$ then $\theta = [t/x]\theta$. Hence θ is a solution $\{x \stackrel{?}{=} t\} \cup S$ if and only it is a solution of $\{x \stackrel{?}{=} t\} \cup S[t/x]$). □

From Proposition 1 we get:

Theorem 2 (Unification Theorem)**.** A unifiable set of literals D has a most general unifier.

1.2 Robinson's Unification Algorithm

We give a second variant of the unification algorithm, which usually attributed to J. Robinson. This version does not explicitly break terms down into subterms (as in the **Decompose** rule above). This makes the algorithm easier to think about in small examples, but makes the worst-case running time exponential (see the question sheet).

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

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

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 $\theta' = \theta\theta'$. 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 $\theta' = [t/x]\theta'$. Thus the loop invariant is maintained by the assignment $\theta := \theta[t/x]$.

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.

Example 3. 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 leftmost 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 sets of clauses, that is, sets 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 $\overline{D} = \{P(x), \neg R(x, y)\}$.

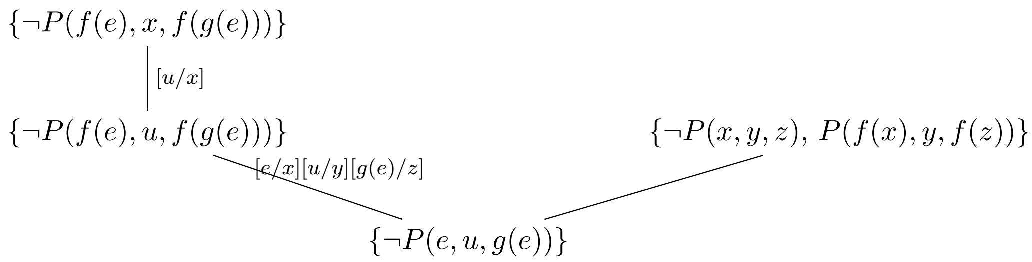

Figure 1: First-order resolution example

Definition 4 (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

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

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 5. 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 $[e/x][u/y][g(e)/z]$, obtaining the clause $\{\neg P(e, u, g(e))\}$.

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 6. 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, 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]

[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 $\forall x \forall y \forall z (A(e, x, x) \wedge ((\neg A(x, y, z) \vee A(s(x), y, s(z))) \wedge A(s(s(e)), x, y))$,

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 $[e/x_2][c/y_2][s(z_2)/z_3][z_4/z_3]$
5.	$\{A(e, y_5, y_5)\}$	clause of F_1
6.	$\square$	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 7 (Resolution Lemma)**.** Let $F = \forall x_1 \ldots \forall x_n G$ be a closed formula in Skolem form, with G quantifier-free. 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. $\square$

Corollary 8 (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. $\square$

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 9 (Lifting Lemma)**.** Let C_1 and C_2 be variable-disjoint 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 $\overline{L} \in G_2$. Since C_1 and C_2 have no variable in common we can write $G_1 = C_1\theta'$ and $G_2 = C_2\theta'$ for a single ground

which is logically equivalent to $F_1 \wedge F_2 \wedge G_3$.

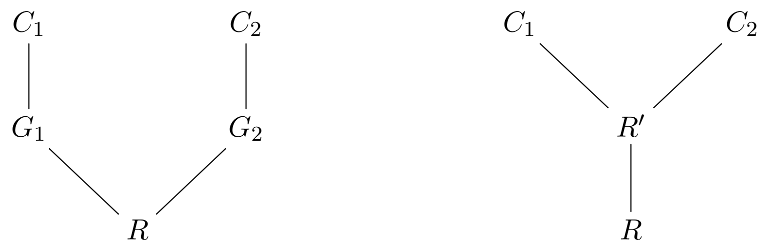

Figure 2: Ground resolution step on the left, and its predicate-logic lifting on the right.

substitution θ'. Let $D_1 \subseteq C_1$ be the set of literals mapped to L by θ' and let $D_2 \subseteq C_2$ be the set of literals mapped to $\overline{L}$ by θ'. Then θ' is a unifier of $D_1 \cup \overline{D_2}$. Writing θ for the most general unifier of $D_1 \cup \overline{D_2}$, 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.

Recall from the proof of the Unification Lemma that $\theta' = \theta\theta'$. By (2) we now have that

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

(The first equality uses the fact that $D_1\theta$ and $C_1\theta$ have disjoint images under θ' and likewise $D_2\theta$ and $C_2\theta$ have disjoint images under θ', which follows from $\theta' = \theta\theta'$.) We conclude that R is a ground instance of R' under the substitution θ'. □

Corollary 10 (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. □