Resolution Principle Theorem Proving

The Resolution Principle along with paramodulation provides a simple basis for mechanized inference and question answering

Introduction: A Simple Example

The Resolution Principle was proposed by J. Alan Robinson in 1965 as a basis for mechanical theorem proving, and has dominated mechanized deduction in AI since then. It is a descendant of Herbrand's proof procedure (1930) and Prawitz' improvement of it (1960). In the late 70's Robert Kowalski in Britain and Alain Colmerauer in France developed the logic programming language Prolog, which uses resolution-based inference as its fundamental computational operation. This language (and logic programming more generally) became very influential in AI, and further enhanced the significance of resolution.

Proving a conclusion ϕ from a set of premises Δ using the resolution principle involves two stages. First, we add the denial of ϕ to the premises, i.e., we form $\Delta, \neg\phi$ (strictly, we should say $\Delta \cup \{\neg\phi\}$), and we convert this to *clause form*. In clause form, we have a set of disjunctions called *clauses*, where the disjuncts in each clause are *literals*, i.e., either *atoms* (atomic formulas of form $\pi(\tau_1, ..., \tau_n)$ or $\sigma = \tau$) or negated atoms; there are no explicit quantifiers, but free variables are regarded as universally quantified. Then we repeatedly apply resolution (and, possibly, factoring and paramodulation) till we derive the empty clause $\Box$. This contradiction establishes the desired conclusion, ϕ.

For example, suppose our premises are

1. $(\forall$x $((\text{Dog(x)} \Rightarrow \text{Barks(x)}))$ (All dogs bark)

and

2. Dog(Snoopy),

and we wish to prove that "Something barks", i.e.,

$(\exists$x Barks(x)).

We form the denial of this desired conclusion and add it to our premises:

3. $\neg(\exists$x Barks(x)).

We now convert these formulas to clause form. For formula (1), this is

1'. $\neg$Dog(x) $\vee$ Barks(x) (Any x either isn't a dog, or barks)

Note that universal quantification of x is implicit, and we've converted the conditional to a disjunction. You can see intuitively that this says the same as (1). The two literals in this clause happen to be atoms. Formula (2) is unchanged in clause form – it has just one literal, and such clauses are called *unit clauses*. For formula (3), we first "pass the negation through the quantifier", changing $\exists$ to $\forall$ in doing so:

($\forall$x $\neg$Barks(x)).

Note that in general $\neg(\exists\nu\ \phi)$ is equivalent to $(\forall\nu\ \neg\phi)$; i.e., if nothing exists which is ϕ, then surely everything is *not* ϕ (and *vice versa*). In fact it's not hard to prove that the entailment relation "$\models$" holds in both directions. (Similarly $\neg(\forall\nu\ \phi)$ is equivalent to $(\exists\nu\ \neg\phi)$.) Now we just drop the universal quantifier, getting the unit clause

3'. $\neg$Barks(x).

This completes the conversion to clause form. In the second stage, we repeatedly resolve pairs of clauses till we get the empty clause. First, we resolve (1') and (2), by *unifying* the literal $\neg$Dog(x) with the literal Dog(Snoopy), (i.e., letting x = Snoopy), and "cancelling" the resolved literals. The conclusion is the disjunction of remaining literals, in this case just

4. Barks(Snoopy) r[1'a,2] (resolve literal "a" of clause 1' with clause 2)

Notice that since we substituted Snoopy for x, this is now a special case of the original literal Barks(x) in (1'). Next we resolve (4) and (3'), again using unifier x = Snoopy; when we cancel the unified literals, we are left with no literals, i.e., the empty clause:

5. $\square$

This completes the resolution proof.

We now need to spell out the details of how we convert to clause form starting with arbitrary formulas of FOL, how we unify arbitrary literals, and how we form the *resolvent* after unifying literals of two clauses. Eventually we'll also talk about factoring (needed for completeness if we use the simplest form of resolution), and paramodulation (needed if we want to handle equality, i.e., full FOL).

Conversion to Clause form

Starting with any closed formula, we perform the following 6 transformations. (All but Skolemization – step 4 – are based on logical equivalences, i.e., the original formula entails, and is entailed by, its replacement.)

1. **Eliminate $\Rightarrow$ and $\Leftrightarrow$:**

   Replace $(\phi \Leftrightarrow \psi)$ by $((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi))$
   $(\phi \Rightarrow \psi)$ by $(\neg\phi \vee \psi)$

2. **Move $\neg$ inward** (so that we are left only with negated atomic wffs):

Replace $\neg(\phi \vee \psi)$ by $(\neg\phi \wedge \neg\psi)$
$\neg(\phi \wedge \psi)$ by $(\neg\phi \vee \neg\psi)$
$\neg\neg\phi$ by ϕ
$\neg(\forall x\phi)$ by $(\exists x\neg\phi)$
$\neg(\exists x\phi)$ by $(\forall x\neg\phi)$

Actually, steps 1 and 2 can be neatly combined with a single recursive algorithm $\mathcal{T}$:
$\mathcal{T}(\phi \Leftrightarrow \psi) = (\mathcal{T}(\phi \Rightarrow \psi) \wedge \mathcal{T}(\psi \Rightarrow \phi))$
$\mathcal{T}(\phi \Rightarrow \psi) = (\mathcal{T}(\neg\phi) \vee \mathcal{T}(\psi))$
$\mathcal{T}(\neg(\phi \vee \psi)) = (\mathcal{T}(\neg\phi) \wedge \mathcal{T}(\neg\psi))$
$\mathcal{T}(\neg(\phi \wedge \psi)) = (\mathcal{T}(\neg\phi) \vee \mathcal{T}(\neg\psi))$
$\mathcal{T}(\neg(\forall x\phi)) = (\exists x\mathcal{T}(\neg\phi))$
$\mathcal{T}(\neg(\exists x\phi)) = (\forall x\mathcal{T}(\neg\phi))$

We also need to add additional rules for the remaining possibilities, where $\mathcal{T}$ just moves inward without making any change, e.g., $\mathcal{T}(\phi \vee \psi) = (\mathcal{T}(\phi) \vee \mathcal{T}(\psi))$, $\mathcal{T}(\forall x\phi) = (\forall x\mathcal{T}(\phi))$, etc. Also when $\mathcal{T}$ reaches a literal, it just returns that literal without change:
$\mathcal{T}(\neg\pi(\tau_1, ..., \tau_n)) = \neg\pi(\tau_1, ..., \tau_n)$
$\mathcal{T}(\pi(\tau_1, ..., \tau_n)) = \pi(\tau_1, ..., \tau_n)$.

3. **Standardize apart**:
Rename any duplicate variables so that each quantifier binds a unique variable.

4. **Eliminate existential quantifiers by Skolemization**, making sure that dependencies of existential variables on universal variables are reflected in the Skolem functions' argument structure.

   For an existential not lying within the scope of any universal quantifier, we just drop the $\exists\nu$ from $\exists\nu\phi$, substituting a new name (Skolem constant) for all occurrences of the variable in ϕ, as in Existential Instantiation. But note that we are applying the rule more generally now – e.g., in embedded occurrences like

   P(A) $\vee$ ((∃x Q(B,x)) $\wedge$ (∀y ¬R(y))).

   This would become P(A) $\vee$ (Q(B,C) $\wedge$ (∀y ¬R(y))) upon Skolemization, with new constant C. If we had a wide-scope universal, as in

   (∀y (P(y) $\vee$ ((∃x Q(B,x)) $\wedge$ ¬R(y)))),

   we would get (∀y (P(y) $\vee$ (Q(B,f(y)) $\wedge$ ¬R(y)))) after Skolemization, with new Skolem function term f(y). The intuitive justification for functional Skolemization is this. First of all, it can be shown that at this point of the transformation to clause form, all the formulas we obtain can be equivalently rewritten in *prenex form*, i.e., with all quantifiers moved to the left. For example, the formula above can be rewritten equivalently as

   (∀y (∃x (P(y) $\vee$ (Q(B,x) $\wedge$ ¬R(y))))).

So we have formulas of form

$(\forall\nu(\forall\omega...(\exists\sigma\phi)...))$,

after Skolemizing top-level existentials. This says that corresponding to each ν, and each ω, and each ..., there exists a σ such that ϕ holds. But if that is so, then there also exists a *function* which *supplies* such a value σ corresponding to arguments $\nu, \omega, ...$ So we can make up a name like f$(\nu, \omega, ...)$ for such a function and substitute it for σ.

For instance, suppose it is true that for every pair of persons in the world, there is a third person that speaks all the languages that the first two persons speak, which we can symbolize as

(∀x (∀y (∃z S(x,y,z)))),

assuming our domain of discourse consists of the world's people. If that is so, then for each pair of people x, y, we should be able to supply a specific third person, z (perhaps chosen from many possibilities), who speaks all the languages of x and y, something like this:

x	y	f(x,y)
Smith	Lafrance	Trudeau
Smith	Glukov	Yevtushenko
Merkel	Yevtushenko	Yevtushenko
Trudeau	Yevtushenko	Nabokov
	etc.	

So then we can rewrite the original statement as

(∀x (∀y S(x,y,f(x,y)))).

Intuitively, saying that z exists for all x, y is the same as saying that f(x,y) has a value for all x, y. By introducing F, we say the latter automatically, since in FOL a function is *total*, i.e., defined on the entire domain of discourse.

Final note: Recall that the B&L text reserves the term "*sentence*" for *closed* wffs, i.e., containing no free variables (see p. 18), and it defines *truth* (in a model ($\mathcal{D}$,$\mathcal{I}$)) only for sentences (see p. 22). In class, we did not in principle disallow open wffs in the definition of truth; since truth is defined in terms of satisfaction by *all* variable assignments, this implies that a wff with free variables behaves semantically as if all its free variables were universally quantified. Thus, for instance,

Physically-detectable(x)∨ ¬Exists(x)

would mean "Everything is either physically detectable or doesn't exist", just as if there were a quantifier (∀x ...) prefixed to the formula. This is precisely how we can interpret formulas in clause form. However, in resolution principle theorem-proving, we assume that we always *start out* with closed formulas (i.e., with sentences). This avoids complications in Skolemization.

5. **Drop $\forall$ quantifiers**. After this, the free variables are "understood" to be universally quantified.

6. **Move $\vee$ inward**, i.e., distribute $\vee$ over $\wedge$:

   $((\phi \wedge \psi) \vee \chi)$ becomes $((\phi \vee \chi) \wedge (\psi \vee \chi))$
   $(\chi \vee (\phi \wedge \psi))$ becomes $((\chi \vee \phi) \wedge (\chi \vee \psi))$

7. **Drop $\wedge$'s**. This leaves us with a *set* of *clauses*, where each clause is just a disjunction of literals, i.e., atomic formulas or negated atomic formulas (possibly with free variables). We already saw examples like

   $\neg$Barks(x) and $\neg$Dog(x) $\vee$ Barks(x)

   earlier. A single-literal clause like the first one here is called a *unit* clause. Sometimes a final step is taken, in which clauses are reformulated as *sets of literals*, e.g.,

   {$\neg$Barks(x)}, {$\neg$Dog(x), Barks(x)}.

   This has the advantage that duplicate literals are "automatically" eliminated (assuming appropriate set-manipulation functions are implemented). On the other hand, it can be a little confusing, since now set formation at the level of clauses means *conjunction*, while set formation at the level of literals means *disjunction*. But either notation is acceptable in this course.

Example of conversion to clause form

Let's look at the following example (using some different bracket shapes for improved clarity):

(∀x [(∃y [P(x,y) ∨ Q(x,y)]) ⇒ (∃y R(x,y))])
Eliminate ⇒:
(∀x [¬(∃y [P(x,y) ∨ Q(x,y)]) ∨ (∃y R(x,y))])
Move ¬ inward:
(∀x [(∀y ¬[P(x,y) ∨ Q(x,y)]) ∨ (∃y R(x,y))])
(∀x [(∀y [¬P(x,y) ∧ ¬Q(x,y)]) ∨ (∃y R(x,y))])
Standardize apart:
(∀x [(∀y [¬P(x,y) ∧ ¬Q(x,y)]) ∨ (∃z R(x,z))])
Skolemize:
(∀x [(∀y [¬P(x,y) ∧ ¬Q(x,y)]) ∨ R(x,f(x))])
Drop ∀: [[¬P(x,y) ∧ ¬Q(x,y)] ∨ R(x,f(x))]
Move ∨ inward: [[¬P(x,y) ∨ R(x,f(x))] ∧ [¬Q(x,y) ∨ R(x,f(x))]]
Drop ∧: ¬P(x,y) ∨ R(x,f(x)), ¬Q(x,y) ∨ R(x,f(x))

Thus we obtain two clauses.