 Rev. December, 1992

Resolution Theorem Proving

Lifting is one of the most important general techniques for accelerating difficult computations. Lifting was used by Alan Robinson in the early 1960's to accelerate first order inference [Robinson, 1965]. A complete inference procedure had been done by Martin Davis and Hillary Putnam a few years prior to Robinson's work [Davis and Putnam, 1960]. Unfortunately, it was a ground procedure and suffered from a large search space generated by unguided selection of gound instances of quantified formulas. The general method of lifting any ground procedure was not known at the time. Without an understanding of the general technique, it is difficult to lift the Davis-Putnam procedure. Rather than lift the Davis-Putnam procedure, Robinson designed a ground procedure for which lifting was particularly simple. The lifted inference process became known as resolution. Because of the tremendous advantage of a lifted procedure over a ground procedure, resolution immediately became the central paradigm of automated first order inference. Over the years resolution has been greatly refined. The basic procedure is defined below and several refinements are described. Although lifting is a good technique for accelerating difficult computations, lifting alone has not brought computers near human-level competence in mathematical reasoning. It is still instructive, however, to study the classical resolution theorem proving techniques.

In order to using lifting one must first construct a ground procedure. The first step in constructing the ground procedure is to convert inference problems to a simple normal form.

1 Clausal Normal Form

Resolution is a refutation technique. To show that $\Sigma \models \Phi$ resolution derives a contradiction from $\Sigma \cup \{\neg\Phi\}$. We will simply assume that we are given a finite set of formulas Γ and that we wish to show that Γ is unsatisfiable. The first step in resolution theorem proving is to put the formulas in Γ in a certain normal form. To define this normal form we need some additional terminology.

> **Definition** An *atomic formula* is either a proposition symbol, a predicate application of the form $P(t_1, \cdots, t_n)$, or an equation $t_1 = t_2$ between two terms.

> **Definition** A *literal* is either an atomic formula or the negation of an atomic formula.

> **Definition** A *clause* is a first order proposition of the form
>
> $$\forall x_1, x_2, \cdots x_n(\Phi_1 \vee \Phi_2 \vee \cdots \Phi_m)$$
>
> where each Φ_i is a literal and such that the proposition has no free variables, i.e., the only variables that appear in the clause are the variables $x_1, \cdots x_n$ that appear in the leading universal quantifiers.

For example, the universally quantified implication $\forall x(P(x) \rightarrow Q(x))$ is equivalent to the clause $\forall x(\neg P(x) \vee Q(x))$. The statement that a relation R is transitive can be written as

$$\forall x, y, z[(R(x,\ y) \wedge R(y,\ z)) \rightarrow R(x,\ z)]$$

which is equivalent to the clause

$$\forall x, y, z[\neg R(x,\ y) \vee \neg R(y,\ z) \vee R(x,\ z)].$$

If $\Psi_1, \cdots, \Psi_n$ and Φ are literals, then we will treat the formula

$$\forall x_1, \cdots, x_n[(\Psi_1 \wedge \cdots \wedge \Psi_n) \rightarrow \Phi]$$

as an alternate notation for the equivalent clause

$$\forall x_1, \cdots, x_n[\neg\Psi_1 \vee \cdots \vee \neg\Psi_n \vee \Phi].$$

A clause may contain any number of literals — including one or zero. A clause with a single literal is called a unit clause.

Definition A *unit clause* is a clause that contains only one literal.

Definition A *ground clause* is a clause with no variables.

A *unit clause* is a formula of the form $\forall x_1, \cdots x_n \Phi$ where Φ is a literal. For example, the proposition $\forall x\ P(x, f(x))$ is a unit clause. The empty clause is defined to be the formula that is false in all models.

The above definition requires that the only variables involved in a clause are the ones bound by the universal quantifiers. This makes it possible to drop the quantifiers from the representation of a clause. A clause can be represented by a set of literals where the variables appearing in the set of literals are implicitly universally quantified. For example, the statement that the relation R is transitive can be expressed with the clause

$$\neg R(x,\ y) \vee \neg R(y,\ z) \vee R(x,\ z).$$

If b, c, and d are constant symbols then the following is a *ground instance* of the above transitivity clause.

$$\neg R(b,\ c) \vee \neg R(c,\ d) \vee R(b,\ d)$$

In practice mathematical theorems are often translated into clausal normal form by hand so that the input to a resolution theorem prover is already a set of clauses. However, it is not difficult to design a procedure for efficiently converting arbitrary first order formulas into clauses. In fact we have the following theorem.

> **Normal Form Theorem:** Given any finite set of first order propositions Γ, one can construct, in linear time in the written size of Γ, a set of clauses Γ' such that Γ' is satisfiable if and only if Γ is satisfiable.

The first step in the conversion is to replace all the variables that appear free in Γ by constant symbols. Recall that an occurrence of a variable x is free in a formula Φ if that occurrence is not inside a formula of the form $\forall x \Psi(x)$. for each variable x that occurs free in some member of Γ, we select a new constant symbol (one that does not already appear in Γ) and replace all free occurrences of x by c. This replacement does not affect the satisfiability of Γ. The syntactic distinction between constants and variables is used to keep track of which symbols occur free and which symbols occur bound.

The next step is to replace each proposition in Γ by a set of clauses. For each proposition Ψ in Γ we construct a new proposition symbol P and call the procedure `DEFINE` given below to "define" P to be a representation of Ψ. The procedure `DEFINE` recursively generates clauses that enforce the definition. After defining P to be a representation of Ψ we then add the unit clause P to set of output clauses. After this is done for each element of Γ the set of output clauses is satisfiable if and only if Γ is satisfiable.

The procedure `DEFINE` takes two arguments — an atomic formula that is being defined and an arbitrary formula that is its definition. A proposition symbol can be treated as a predicate of no arguments, so we need only consider the general case where the first argument is an atomic formula of the form $P(x_1, \cdots x_n)$. The second argument to the procedure is a proposition $\Psi(x_1, \cdots, x_n)$ such that every free variable of the proposition is one of the variables $x_1, \cdots, x_n$. The procedure creates clauses which define the given predicate symbol P in a way that simulates the definition

$$\forall x_1, \cdots, x_n [P(x_1, \cdots, x_n) = \Psi(x_1, \cdots, x_n)].$$

The procedure can not create the definition directly because the definition is not a clause — but the definition can be simulated with a set of clauses. In the procedure for creating definitional clauses we assume that the only Boolean operators in Ψ are $\vee$ and $\neg$ — all other Boolean connectives can be viewed as abbreviations for expressions constructed with $\vee$ and $\neg$. We also assume that the only quantifier is $\forall$ — an existential proposition can always be written as a negation of a universal proposition.

Procedure `DEFINE` for defining $P(x_1, \cdots, x_n)$ as $\Psi(x_1, \cdots, x_n)$.

1. If $\Psi(x_1, \cdots, x_n)$ is a literal then add the two clauses

   $$P(x_1, \cdots, x_n) \rightarrow \Psi(x_1, \cdots, x_n)$$

   and

   $$\Psi(x_1, \cdots, x_n) \rightarrow P(x_1, \cdots, x_n).$$

2. If $\Psi(x_1, \cdots, x_n)$ is a negation of the form $\neg\Theta(x_1, \cdots, x_n)$ then

   (a) Let Q be a new n-ary predicate symbol.

   (b) Call `DEFINE` on $Q(x_1, \cdots, x_n)$ and $\Theta(x_1, \cdots, x_n)$.

   (c) Add the two clauses

   $$Q(x_1, \cdots, x_n) \rightarrow \neg P(x_1, \cdots, x_n)$$

   and

   $$\neg Q(x_1, \cdots, x_n) \rightarrow P(x_1, \cdots, x_n).$$

3. If $\Psi(x_1, \cdots, x_n)$ is a disjunction of the form $\Theta_1(x_1, \cdots, x_n) \vee \Theta_2(x_1, \cdots, x_n)$ then

   (a) Let Q_1 and Q_2 be new n-ary predicate symbols.

   (b) Call `DEFINE` on $Q_1(x_1, \cdots, x_n)$ and $\Theta_1(x_1, \cdots, x_n)$.

   (c) Call `DEFINE` on $Q_2(x_1, \cdots, x_n)$ and $\Theta_2(x_1, \cdots, x_n)$.

   (d) Add the clauses

   $$Q_1(x_1, \cdots, x_n) \rightarrow P(x_1, \cdots, x_n)$$
   $$Q_2(x_1, \cdots, x_n) \rightarrow P(x_1, \cdots, x_n)$$
   $$(\neg Q_1(x_1, \cdots, x_n) \wedge \neg Q_2(x_1, \cdots, x_n)) \rightarrow \neg P(x_1, \cdots, x_n).$$

4. If $\Psi(x_1, \cdots, x_n)$ is a quantified formula $\forall x_{n+1}\Theta(x_1, \cdots, x_n, x_{n+1})$ then

   (a) Let Q be a new (n+1)-ary predicate symbol and let f be a new n-ary function symbol.

   (b) Call `DEFINE` on $Q(x_1, \cdots, x_n, x_{n+1})$ and $\Theta(x_1, \cdots, x_n, x_{n+1})$.

   (c) Add the clauses

$$\neg P(x_1, \cdots, x_n) \rightarrow \neg Q(x_1, \cdots, x_n, f(x_1, \cdots x_n))$$
$$P(x_1, \cdots, x_n) \rightarrow Q(x_1, \cdots, x_n, x_{n+1})$$

The correctness of the overall normalization process follows from two statements about the definition procedure. Suppose the definition procedure is called on $P(x_1, \cdots, x_n)$ and $\Psi(x_1, \cdots, x_n)$ and generates the clause set C as a result of this call. First, one can show that the clauses generated by the procedure are *sufficiently strong* to define P, i.e., any model of C satisfies the definitional proposition

$$\forall x_1, \cdots, x_n[P(x_1, \cdots, x_n) = \Psi(x_1, \cdots, x_n)].$$

Second, one can show that the clauses are *sufficiently weak* that they only define the given predicate — for any first order structure $\mathcal{M}$ there exists a structure $\mathcal{M}'$ that is identical to $\mathcal{M}$ except for the interpretation of P, and other symbols introduced by the definition procedure, such that $\mathcal{M}'$ satisfies the generated clause set C.

A rigorous proof of the correctness of the definition procedure, and in particular the correctness of the definition of universally quantified formulas, involves the following "Skolemization principle".

> **Skolemization Principle:** If a first order model $\mathcal{M}$ satisfies the proposition
> $$\forall x_1, \cdots, x_n \exists z \Psi(x_1, \cdots x_n, z)$$
> , and f is a function symbol that does not appear in the proposition, then there exists a first order model $\mathcal{M}'$ that is identical to $\mathcal{M}$ except for the interpretation of f, and such that $\mathcal{M}'$ satisfies
> $$\forall x_1, \cdots, x_n \Psi(x_1, \cdots x_n, f(x_1, \cdots, x_n))$$

.

The skolemization principle is often stated more simply by saying that

$$\forall x_1, \cdots, x_n \exists z \Psi(x_1, \cdots x_n, z)$$

is satisfiable if and only if

$$\forall x_1, \cdots, x_n \exists z \Psi(x_1, \cdots x_n, f(x_1, \cdots, x_n))$$

is satisfiable. The function symbol f used in step 4 of the procedure, or referenced in the above Skolemization principle, is called a *Skolem function.* The above form of the Skolemization principle is equivalent to the set theoretic axiom of choice. Because the axiom of choice can be used to prove some surprising results in mathematics, some mathematicians have suggested that it should be rejected. It turns out that one can construct a foundation for mathematics based on ZF set theory in which the Skolemization principle is false. Most mathematicians, however, accept the validity of the Skolemization principle.

2 The Davis-Putnam Procedure

Ground procedures for first order inference are based on Herbrand's theorem. The statement of the theorem involves the notion of a ground instance of a clause.

> **Definition** A *ground instance* of a clause is the result of replacing each variable in the clause by a ground (variable free) term.

For example, if b is a constant symbol then $\neg P(b) \vee Q(b, f(b))$ is a ground instance of the clause $\forall x(\neg P(x) \vee Q(x, f(x)))$.

> **Herbrand's Theorem:** If Γ is an unsatisfiable set of clauses then there exists a finite set Δ of ground instances of clauses in Γ such that Δ is unsatisfiable.

The proof of Herbrand's theorem is similar to the proof of the refutation completeness theorem for the sequent proof discussed earlier. It suffices to show that if every finite set of ground instances of clauses in Γ is satisfiable, then Γ is satisfiable. This is done by constructing a model of Γ whose domain elements are equivalence classes of closed terms. The model constructed in the proof of Herbrand's theorem is called a Herbrand model. The proof will be given in a later version of the notes.

We can now give a ground semi-decision procedure for determining whether $\Sigma \models \Phi$ where $\models$ is the entailment relation of first order logic.

The Davis-Putnam Theorem Proving Procedure: To determine whether $\Sigma \models \Phi$.

1. Construct a set of clauses Γ such that Γ is satisfiable if and only if $\Sigma \cup \{\neg\Phi\}$ is satisfiable.
2. Select a set of ground instances of clauses in Γ.
3. If the selected set of ground clauses is not satisfiable, then return "success" (indicating that $\Sigma \models \Phi$).
4. Otherwise return "failure" (indicating that we have failed to determine whether or not $\Sigma \models \Phi$).

Step 3 must determine the satisfiability of a set of ground clauses. The most efficient procedure for doing this seems to be a procedure based on a combination of Boolean constraint propagation and congruence closure. The original Davis-Putnam procedure did not handle equality, so congruence closure was not used. However, the original Davis-Putnam procedure did use a form of Boolean constraint propagation. Herbrand's theorem implies that there exists a successful execution of this procedure if and only if $\Sigma \models \Phi$.

The Davis-Putnam procedure can be lifted using the general lifting transformation. In the lifted version the clauses selected at step 2 contain expression variables. During the execution step 3, each equality test between expressions is nondeterministic and each nondeterministic outcome of an equality test places constraints on the expression variables. The congruence closure procedure requires performing hash table operations on expressions as well as equality tests. Although this complicates lifting, it still seems possible to lift the efficient congruence closure implementation (this needs to be verified). As in the ground case, there exists a successful execution of the lifted procedure if and only if $\Sigma \models \Phi$.

When Robinson attempted to improve the ground Davis-Putnam procedure, lifting was unknown. Before inventing lifting, Robinson constructed a very simple ground procedure for determining the satisfiability of a set of ground clauses. Robinson was then able to invent lifting as a way of accelerating the simplified ground procedure. By "simple" I mean a procedure that is easy to understand — not necessarily a procedure that runs efficiently.

3 Ground Resolution from Ground Premises

Resolution, in its original form, does not handle equality. In this section we only consider clauses that do not involve equality. Resolution is first defined as a ground procedure. Resolution is an inference rule that can be used to derive a new clause from two previously derived clauses. The resolution inference rule for ground clauses can be stated as follows:

$$\frac{\begin{array}{l}\Phi \vee \Psi_1 \vee \cdots \Psi_n \\ \neg\Phi \vee \Theta_1 \vee \cdots \Theta_k\end{array}}{\Psi_1 \vee \cdots \Psi_n \vee \Theta_1 \vee \cdots \Theta_k}$$

To see that the rule is sound consider any model in which the two antecedent clauses are true. In that model, the proposition Φ must be either true or false. If Φ is true, then one of $\Theta_1, \cdots, \Theta_k$ must be true. If Φ is false, then one of $\Psi_1, \cdots, \Psi_n$ must be true. In either case, the conclusion clause must be true.

Definition We define R be a set of three inference rules — the above resolution rule, an inference rule that allows the literals in a clause to be rearranged in any order, plus an inference rule that allows for the removal of extra copies of the same literal.

The rule set R allows a clause to be viewed as an unordered set of literals. As an example, suppose Γ contains the two clauses $P \vee Q$ and $\neg P \vee Q$. Resolution on these two clauses yields the unit clause $Q \vee Q$. Removal of redundant literals then allows the derivation of the unit clause Q. So it is possible that $\Gamma \vdash_R Q$ even if there are no unit clauses in Γ. If $\Gamma \vdash_R Q$ and $\Gamma \vdash_R \neg Q$ then the resolution rule can be applied to these two unit clauses resulting in a derivation of the empty clause. We write $\Gamma \vdash_R \mathbf{F}$ if there exists a derivation of the empty clause from the clauses in Γ.

Ground Completeness Theorem for Resolution If Γ is an unsatisfiable set of ground clauses then $\Gamma \vdash_R \mathbf{F}$.

Proof The proof is by induction on the number of atomic formulas that appear in Γ. We have assumed that no literal appears twice in the same clause. If an atomic formula is mentioned twice in the same clause then the clause must contain both a literal and its negation and can thus be eliminated while preserving the satisfiability of Γ. So we can assume without loss of generality that no atomic formula occurs twice in the same clause. Select an atomic formula p AND let Γ' be all clauses in Γ that do not mention p plus all clauses that can be derived by resolving two elements of Γ on the formula p. Note that no clause in Γ' mentions p. We will now show that if Γ' is satisfiable then so is Γ. Consider a model $\mathcal{M}$ of Γ'. Suppose this model makes p true. In this case the model must satisfy all clauses in Γ other than those containing the literal $\neg p$. Suppose that there is some clause $C_{\neg p}$ in Γ containing $\neg p$ that $\mathcal{M}$ does not satisfy $C_{\neg p}$. In this case one can show that $\mathcal{M}$ must satisfy every clause C_p containing p by virtue of some literal other than p. To see this it suffices to note that $\mathcal{M}$ must satisfy the result of resolving $C_{\neg p}$ with C_p. Since the model does not satisfy any of the literals in $C_{\neg p}$ it must satisfy some literal in C_p other than p. But if every clause C_p

that contains p is satisfied by some literal other than p, then the model $\mathcal{M}'$ that results from switching p from true to false satisfies every clause in Γ. Thus if Γ' has a model $\mathcal{M}$ that assigns p true then, either $\mathcal{M}$ is a model of Γ, or the result $\mathcal{M}'$ of setting p to false is a model of Γ. A similar argument applies if Γ' has a model that assigns p false. Thus if Γ' has a model then so does Γ. But, by assumption Γ is unsatisfiable. So Γ' must also be unsatisfiable. But Γ' involves one fewer atomic formula than Γ and, by the induction hypothesis, if Γ' is unsatisfiable then the empty clause can be derived from Γ' using the resolution rule.

4 Ground Resolution from Lifted Premises

Now consider a set of clauses Γ where the clauses in Γ need not be ground. The resolution inference rule can be generalized to allow for the derivation of a new non-ground clause from two already derived non-ground clauses. To define the general resolution rule we adopt the convention that all variables in a clause are implicitly universally quantified. This allows us to drop the universal quantifiers from the representation of non-ground clauses. The general resolution rule can then be defined in terms of the notions of substitution and unification. Recall that a substitution is a mapping from variables to terms. If x is a variable then we write $\rho(x)$ to denote the image of x under the mapping ρ. More generally, for any expression s we write $\rho(s)$ to denote the result of replacing each variable in s by its image under ρ. Also recall that a ground term is a term that does not contain variables. A substitution ρ is called *ground* if $\rho(x)$ is a ground term for all variables x. Note that if ρ is a ground substitution then $\rho(s)$ is a ground term for any term s. We can construct a complete inference relation for clauses by adding the following "ground instance" rule. This rule allows one to derive any ground instance of a non-ground clause. In this rule ρ must be a ground substitution.

$$\frac{\Psi_1 \vee \cdots \Psi_n}{\rho(\Psi_1) \vee \cdots \vee \rho(\Psi_n)}$$

We now have the following corollary to Herbrand's theorem.

> **Definition** We define G to be the set of four inference rules that includes the three rules in R restricted to ground clauses, plus the above rule for generating ground instances of non-ground clauses.

> **Resolution Corollary to Herbrand's Theorem** If Γ is an unsatisfiable set of clauses then $\Gamma \vdash_G \mathbf{F}$.

> **Proof** By Herbrand's theorem, if Γ is unsatisfiable then there exists a finite set Γ' of ground instances of clauses in Γ such that Γ' is unsatisfiable. Since Γ' is unsatisfiable, the completeness theorem for ground resolution implies that $\Gamma' \vdash_R \mathbf{F}$. So $\Gamma \vdash_G \mathbf{F}$.

The inference relation $\vdash_G$ can be viewed as having two components that are analogous to steps 2 and 3 of the Davis-Putnam procedure. The instantiation inference rule corresponds to step 2 in which a set of ground instances of the given clause set is constructed. The three inference rules in R correspond to the decision procedure used in step 3 to determine if the generated set of ground clauses is satisfiable.

5 Fully Lifted Resolution

As an example, consider the unit clause $P(x, f(x))$ together with the unit clause $\neg P(f(y)f(z))$ (variables are implicitly universally quantified). The ground clause $P(f(a), f(f(a)))$ is a substitution instance of the first clause, and the ground clause $\neg P(f(a), f(f(a)))$ is a substitution instance of the second clause. So the original pair of clauses is unsatisfiable. However, to derive the empty clause from the two original clauses using the inference rules in G requires making a some lucky guesses about which ground instances should be constructed. The lifted version of resolution uses unification to construct the desired common instance — lifting avoids the guess work about which ground instance to consider.

Lifting is closely related to unification. Recall that a unification of two expressions s and t is a substitution ρ such that $\rho(s)$ is the same expression as $\rho(t)$. For example consider the terms $g(x,\ x)$ and $g(f(z), f(y))$. Consider a substitution ρ containing the mappings $x \mapsto f(y)$, $z \mapsto y$, and $y \mapsto y$. We have that $\rho(g(x,\ x))$ equals $\rho(g(f(z), f(y))$ equals $g(f(y),\ f(y))$ so ρ is a unifier of $g(x,\ x)$ and $g(f(z),\ f(y)$. But there are also other unifiers of these two terms. For example, consider the substitution τ containing the mappings $x \mapsto f(f(w))$, $z \mapsto f(w)$, $y \mapsto f(w)$ where w is a new variable. τ is also unifies $g(x,\ x)$ and $g(f(z),\ f(y))$. However, the substitution ρ is "less constraining" or "more general" than the substitution τ. Given two expressions s and t a *most general unifier* (MGU) of s and t is a unifier ρ of s and t that is more general (less constraining) than any other unification of s and t — ρ is an MGU of s and t if for any other unification τ of s and t there exists a "refinement" σ of ρ such that for any variable x, $\tau(x)$ equals $\sigma(\rho(x))$. In the above example, the refinement substitution σ contains the mapping $y \mapsto f(w)$.

> **Lemma** If there exists a unifier of s and t then there exists a most general unifier of s and t.

The lifted resolution inference rule is stated below. In this rule all variables in the clauses are implicitly universally quantified. The atomic formulas Φ_1 and Φ_2 must be unifiable and τ is a most general unification of Φ_1 and Φ_2.

$$\frac{\begin{array}{l}\Phi_1 \vee \Psi_1 \vee \cdots \Psi_n \\ \neg\Phi_2 \vee \Theta_1 \vee \cdots \Theta_k\end{array}}{\tau(\Psi_1) \vee \cdots \tau(\Psi_n) \vee \tau(\Theta_1) \vee \cdots \tau(\Theta_k)}$$

As in the case of ground resolution, we include inference rules for reordering the literals of a clause. The inference rule for removing extra copies of a literal from a clause needs careful consideration. An implementation of the ground version of this rule would involve an equality test between literals. In a lifted procedure an equality test produces a unification. The ground rule for eliminating repeated occurrence of the same literal becomes the lifted rule of *factoring* given below. In the factoring rule Φ_1 and Φ_2 are unifiable literals and τ is a most general unifier of Φ_1 and Φ_2.

$$\frac{\Phi_1 \vee \Phi_2 \vee \Psi_1 \vee \cdots \Psi_n}{\tau(\Phi_1) \vee \tau(\Psi_1) \vee \cdots \tau(\Psi_n)}$$

There are four inference rules that define the ground relation $\vdash_G$ — the instantiation rule and the three rules of ground resolution. We will define a lifted inference relation $\vdash_L$ that has four analogous lifted rules. We have just given lifted versions of the three ground resolution rule. The fourth ground rule generates ground instances of clauses. In the lifted procedure, generating another instance of a clause is replaced by generating a copy of the clause with brand new expression variables where the expression variables are further constrained during nondeterministic equality tests. The ground instantiation rule is replaced by the following *variable renaming rule.* A substitution ρ will be called a *renaming* ρ maps variables to variables and does not map any two variables to the same variable. For example, if ρ is a renaming then $\rho(h(f(z), g(y)))$ is a term of the form $h(f(z), g(w))$.

$$\frac{\Psi_1 \vee \cdots \Psi_n}{\rho(\Psi_1) \vee \cdots \vee \rho(\Psi_n)}$$

The renaming rule looks identical to the ground instantiation rule, but the nature of the substitution is extremely different in the two cases. The renaming rule corresponds to step 2 in the lifted version of the Davis-Putnam procedure. In most implementations of resolution, renaming is built into the lifted resolution rule — whenever a new clause is derived by the resolution rule the variables in that new clause are renamed to be brand new variables. In this way, no two clauses share variables.

Definition Let L be the set of four inference rules consisting of the renaming rule, the lifted resolution rule, the reordering rule, and the factoring rule.

Robinson proved the following lifting lemma.

Robinson's Lifting Lemma If $\Gamma \vdash_R C$ then C is a substitution instance of a clause C' such that $\Gamma \vdash_L C'$.

The proof of Robinson's lifting lemma is left as an exercise for the reader. Robinson's lifting lemma immediately implies that if $\Gamma \vdash_G \mathbf{F}$, i.e., the ground rules can derive the empty clause for Γ, then $\Gamma \vdash_L \mathbf{F}$. The completeness of ground resolution immediately implies the completeness of lifted resolution.

Resolution Completeness Corollary If Γ is an unsatisfiable set of clauses, then $\Gamma \vdash_L \mathbf{F}$.

6 Restrictions and Refinements of Resolution

Deriving an empty clause from a given set of clauses using resolution can be very difficult. The problem, of course, is the enormous number of clauses that can be generated. We can think of resolution as a search procedure — one searches the set of derivable clauses in an attempt to find the empty clause. During the late 1960's and early 1970's a large number of variants of resolution were derived. The basic intuition behind all of these variants was to reduce the size of the search space. One way of reducing at least the apparent search space is to restrict the inference rule so that fewer inferences are possible while maintaining the completeness of the procedure. Later versions of the notes will contain a description of the well known variants of resolution.

7 Problems

1. Suppose we wish to show that

$$\forall x \exists y (P(x,\ y) \wedge Q(x,\ y))$$

and

$$\forall x (\exists y P(x,\ y) \wedge \exists z Q(x,\ z) \rightarrow \exists w R(x,\ w))$$

entail

$$\forall x \exists w R(x,\ w).$$

Translate this entailment question into a question about the satisfiability of a first order formula in Skolem normal form. Your answer should be the same as that which would be produced by the general procedure for conversion to Skolem normal form described in the text.

2. Give a resolution proof that the Skolem normal form formula generated in problem 1 is unsatisfiable.

3. Give a resolution proof that the following two clauses are unsatisfiable.

$$P(x,\ y) \vee P(y,\ x)$$

$$\neg P(x,\ y) \vee \neg P(y,\ x)$$

Hint: consider the case where x equals y.

References

[Davis and Putnam, 1960] M Davis and H. Putnam. A computing procedure for quantification theory. *JACM*, 7(3), July 1960.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1), January 1965.