Computational Properties of Resolution and First-Order Logic

Soundness and Completeness:

Definition: The symbol $\vdash_{Res1}$ denotes the inference mechanism of first-order resolution (without paramodulation).

Theorem: First-order resolution is sound. That is, for any set of clauses Φ and for any clause φ,

$$\Phi \vdash_{Res1} \varphi \quad \text{implies} \quad \Phi \models \varphi.$$

Proof: The proof is straightforward, and similar to the case for propositional resolution. □

Theorem: First-order resolution is complete for refutation. That is, for any set of clauses Φ,

$$\Phi \models \bot \quad \text{implies} \quad \Phi \vdash_{Res1} \bot.$$

Proof: The proof will be examined shortly.

Decidability:

Theorem: There is no algorithm which can take as input an arbitrary set Φ of clauses and determine whether or not Φ is satisfiable; that is, whether or not $\Phi \models \perp$ holds.
Proof: It can be shown that this problem is equivalent to the well-known halting problem. The proof will not be provided here. □

Corollary: Resolution for first-order logic cannot possibly be an algorithm which decides whether or not $\Phi \models \perp$ holds. □

Q: What can go wrong?

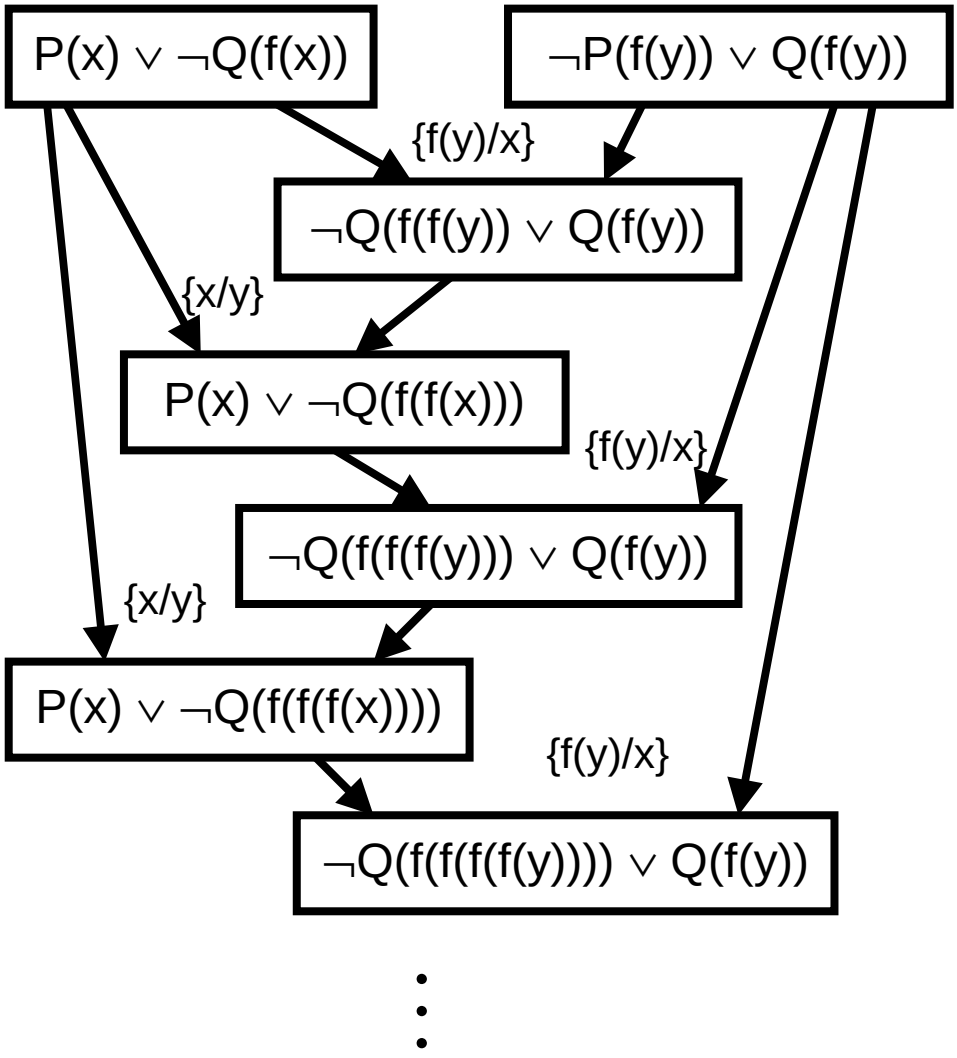
A: The process can loop forever generating new resolvents. In general, it is impossible to determine whether or not this generation will be productive in arriving at a deduction of $\perp$.

In other words:

- Resolution will find a proof if one exists, but may loop forever in a hopeless search if no such proof exists.

- It is impossible to determine, in general, whether or not the search is hopeless.

Example: Shown below is a plan for generating an infinite set of resolvents from a pair of clauses. In general, it is impossible to determine whether or not such a plan will be productive in generating the empty clause, and hence a refutation.



The Herbrand universe and the Herbrand base:

Informal definition (will be refined below): A *ground atom* is an atom of the form $P(t_1, t_2, .., t_n)$ in which none of the t_i's contains any variables. Put another way, ground atoms are built from terms consisting only of constant and function symbols.

Examples: P(f(a,g(b,c)), a) is a ground atom.
P(f(x,g(b,c)), a) is not a ground atom.

The notion of ground atom is extremely useful in establishing decidability, as well as the completeness of resolution, and deserves a somewhat fuller treatment.

Definition: Let $T = (V, K, F)$ be a first-order term algebra. The *Herbrand universe* $\mathcal{H}(T)$ for T is defined inductively as the smallest set satisfying the following conditions.

(a) If $a \in K$, then $a \in \mathcal{H}(T)$.

(b) If $K = \varnothing$, then a special constant symbol **1** is included in $\mathcal{H}(T)$.

(c) If $f \in F$ has arity n, and $t_1, t_2, .., t_n \in \mathcal{H}(T)$, then $f(t_1, t_2, .., t_n) \in \mathcal{H}(T)$.

Condition (b) is necessary to avoid having an empty Herbrand base, which would otherwise be the case were K empty.

Observation: As long as $\mathcal{T}$ contains at least one constant symbol, the Herbrand universe for $\mathcal{T}$ consists of precisely those terms which do not include any variables.

Example: If $\mathcal{T}$ is a function-free term algebra, then $\mathcal{H}(\mathcal{T}) = K$, provided that K is not empty. Otherwise, $\mathcal{H}(\mathcal{T}) = \{\mathbf{1}\}$.

Thus, for a function-free term algebra, the Herbrand base is finite, provided that the number of constant symbols is finite.

Example: Let $\mathcal{T} = (V, K, F)$ have $K = \{a\}$ and $F = \{f\}$, with f a unary function symbol. Then

$$\mathcal{H}(\mathcal{T}) = \{a, f(a), f(f(a)), f(f(f(a))), .. \}.$$

Observations:

- As long as $\mathcal{T}$ has at least one non-nullary function symbol, the Herbrand universe of $\mathcal{T}$ is infinite.

- If the number of constant symbols and function symbols is finite, then the Herbrand universe will be at most countably (denumerably) inifinite.

Example: Even for relatively simple term algebras, it can be complex to enumerate the elements of the Herbrand universe. Let K = {a, b}, and let F = {f, g}, with f unary and g binary. Then

$\mathcal{H}(T)$ = {a, b, f(a), f(b), g(a,a), g(a,b), g(b,a), g(b,b), f(f(a)), f(f(b)), f(g(a,a)), f(g(a,b)), f(g(b,a)), f(g(b,b)), g(f(a),a), g(a,f(a)), g(f(b), b), g(b, f(b)), … }.

Example: Consider the blocks world. The term algebra has constant set {B1, B2, P1, P2}, and, due to Skolemization, there is one unary function symbol, base. Thus, the Herbrand universe is

{B1, B2, P1, P2, base(B1), base(B2), base(P1), base(P2), base(base(B1)), … }.

Definition: Let $L = (R, C, A, T)$ be a first-order logic, with $T = (V, K, F)$.

(a) A *ground atom* over L is any atom of the form $P(t_1, t_2, .., t_n)$, with $t_1, t_2, .., t_n \in \mathcal{H}(T)$. The set of all ground atoms for L is called the *Herbrand base for* L, and is denoted $\mathcal{H}(L)$.

(b) A *ground clause* is any clause built up from members of the Herbrand base.

(c) Given any clause φ, a *grounding* of φ is any ground clause of the form $\varphi\sigma$, in which σ is a substitution which replaces all variables in φ with ground terms.

Example:

$\neg$Is_pyramid(base(P1)) $\vee$ $\neg$On(B2,base(P1))

is a ground clause obtained by applying the substitution {base(P1)/x, B2/y} to the clause shown below

$\neg$Is_pyramid(x) $\vee$ $\neg$On(y,x)

while

$\neg$Is_pyramid(base(P1)) $\vee$ $\neg$On(B1,base(base(B2)))

is a ground clause which is not a grounding of the above clause with variables.

Definition: Let Φ be a set of clauses.

$$\text{Groundings}(\Phi)$$

denotes the set of all groundings of members of Φ.

Herbrand theorem: Let Φ be a finite set of clauses. Then

$$\Phi \models \bot$$

iff there is a finite set $\Psi \subseteq \text{Groundings}(\Phi)$ such that

$$\Psi \models \bot.$$

In words, Φ is unsatisfiable iff it has a finite set of groundings which is. □

- The Herbrand theorem is remarkable in that it asserts that questions of satisfiability within first-order logic may be reduced to questions of satisfiability within a propositional logic: the logic of ground clauses.

- The fly in the ointment, of course, is that the number of propositions in this grounded logic may be infinite.

- Even though we need to use only a finite number of ground clauses to establish undecidability, we do not know, in general, which finite set.

Application of the Herbrand theorem:

Theorem: Let Φ be a finite set of clauses in a first-order predicate logic. Then

$$\Phi \vdash_{Res1} \perp$$

iff there is a finite subset $\Psi \subseteq$ Groundings(Φ) such that

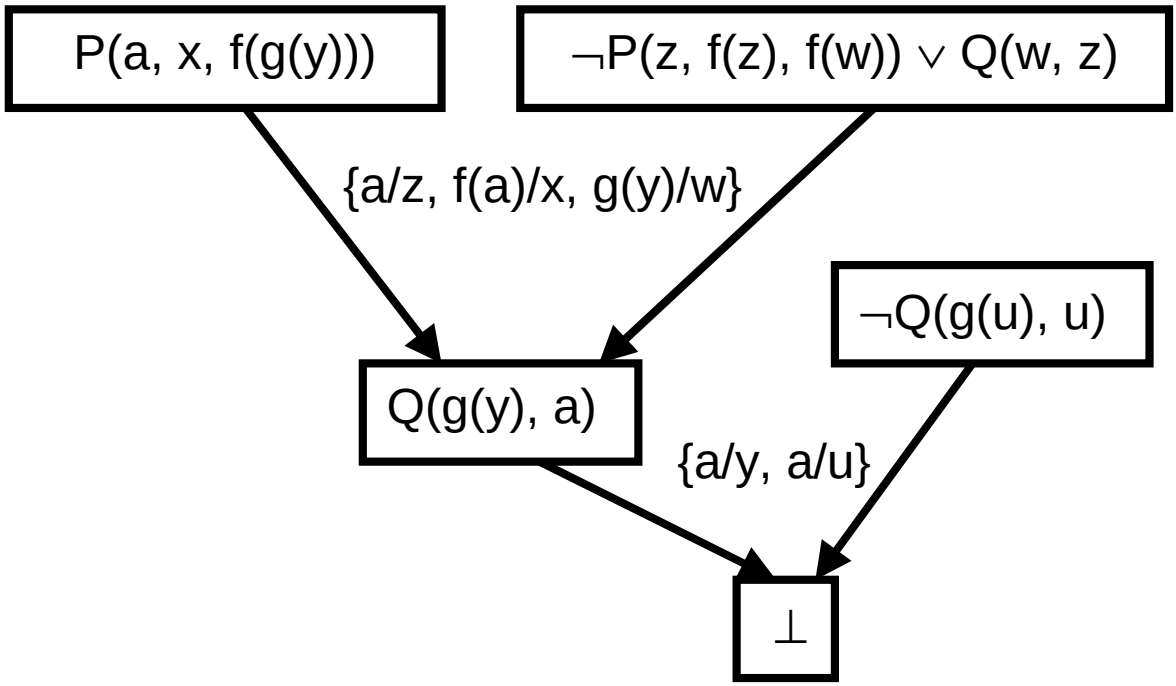
$$\Psi \vdash_{Res} \perp.$$

Here $\vdash_{Res}$ denotes resolution within propositional logic.
Proof: The full proof will not be provided here, but some illustration of the technique will be. □

Suppose that we are given the following clauses Φ:

$$P(a, x, f(g(y))),$$
$$\neg P(z, f(z), f(w)) \vee Q(w, z),$$
$$\neg Q(g(u), u).$$

Here is a resolution refutation, taken from previous slides:

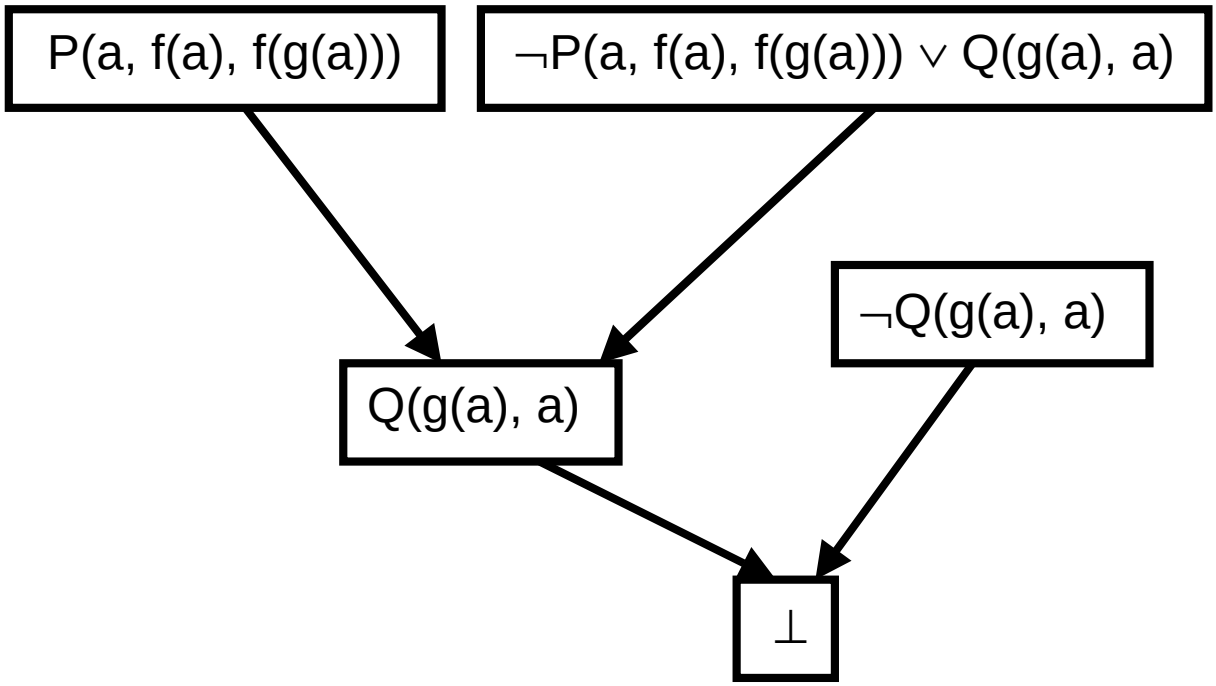


Here is an equivalent refutation from Groundings(Φ).

Think of the process as one of applying very strong substitutions to the initial axioms.

The trick is, of course, to identify appropriate substitutions, without knowing the original proof.

In this case, they have been constructed from the above proof.



If we can show that one can go back and forth between these two types of representations, then completeness of first-order resolution is established, based upon the completeness of propositional resolution.

Decidability and Finiteness issues:

Although full first-order logic is undecidable, there are some important subsets which are.

Theorem: Let Φ be a finite set of clauses which do not involve any function symbols in their terms. Then

$$\Phi \models \bot.$$

is decidable, and may in fact be decided by resolution.
Proof: If there are no function symbols, then the Herbrand base is finite, and so the problem is reduced to one of propositional resolution on a finite number of clauses. □

Note that this result applies to the set of clauses actually involved in the deduction process, which is the set obtained *after* normalization and Skolemization is performed.

- If there are any Skolem functions (not constants), then this decidability result does not apply.

Example: It does not apply to the simple blocks-world example, since there was a Skolem function (base).

Classes of decidable formulas:

Definition:

(a) A logical formula φ in prenex normal form is said to be *in class* $\exists^*\forall^*$ if every existential quantifier precedes every universal quantifier.

(b) A set of formulas is *function free* if none of the terms in any of the formulas contains a function symbol.

(c) The formulas which are both in class $\exists^*\forall^*$ and function free is called *Schönfinkel-Bernays.*

Example:

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

is in class $\exists^*\forall^*$, but not function free. The formula

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

is function free, but not in class $\exists^*\forall^*$. The formula

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

is both, hence Schönfinkel-Bernays.

Theorem: Let Φ be a finite set of formulas in the Schönfinkel-Bernays class. Then it is decidable whether or not $\Phi \models \perp$.
Proof: The Herbrand base is finite. $\square$

- In any practical application of logical deduction, decidability is of the utmost importance.

- While the Schönfinkel-Bernays class is a very useful, and has seen much application, it is also very limiting.

Q: Is there any way to extend the Schönfinkel-Bernays idea to more general classes of formulas?

A: Yes, by using types.

Example: Suppose that we have a modelling situation in which students attend universities, with the following sentence.

$$(\forall x)(\exists y)(\text{Is\_student}(x) \rightarrow \text{Attends}(x,y)).$$

This sentences says that every student attends a university. Upon normalizing and Skolemizing, we obtain

$$\text{Is\_student}(x) \rightarrow \text{Attends}(x,\text{univ}(x)),$$

with univ() the Skolem function obtained. In this simple situation, the Herbrand universe will be infinite, thus precluding its use to establish decidability.

Note, however, that it does not make sense semantically to apply the function univ more than once.

While univ(s) makes sense, univ(univ(s)) does not, because universities are not students and do not attend other universities.

We can preserve satisfiability by building this idea into the logic. To do so, we use a *typed logic.*

We assume:

- There are two disjoint types of objects, students and universities.
- Every term has a type, in particular.
  - V_S = variables of type student = $\{x^S, y^S, \ldots\}$.
  - V_U = variables of type university = $\{x^U, y^U, \ldots\}$.
  - K_S = constants of type student.
  - K_U = constants of type university.
  - $F_{S \to U}$ = function symbols of type [S] $\to$ [U].
  - Other types of function symbols, as needed. (See below).

Each function symbol has a type:

$$\text{univ: [student]} \to \text{[university]}$$

In general, there may be function symbols for each pair of types.

Write the constraint as:

$(\forall x^S)(\exists y^U)(\text{Is\_student}(x^S) \rightarrow \text{Attends}(x^S,y^U))$.

The normalization is:

$$\text{Is\_student}(x^S) \rightarrow \text{Attends}(x^S,\text{univ}(x^S)),$$

which is equivalent to

$$\text{Attends}(x^S,\text{univ}(x^S)).$$

Now the Herbrand base will remain finite, because the type constraint prevents terms such as

$$\text{univ}(\text{univ}(x^S)).$$

Thus, the logic with this single sentence is decidable.

In computational linguistics, it is critical that parsing algorithms be decidable. This idea has been applied successfully to such problems.

Hegner, S. J., “A family of decidable feature logics which support HPSG-style set and list constructions,” in *Logical Aspects of Computational Linguistics: First Annual Conference, LACL ’96, Nancy, France, September 1996, Selected Papers,* Springer-Verlag, 1997, pp. 208-227.

Remark: These techniques even apply to logics with equality.