A Note on the Completeness of Certain Refinements of Resolution

Jean Goubault-Larrecq

J. Goubault–Larrecq

A Note on the Completeness of Certain Refinements of Resolution

Research Report LSV–02–8, Jul. 2002

Laboratoire Spécification et Vérification

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Research Report LSV–02–8, Lab. Spécification et Vérification, CNRS & ENS de Cachan, France, Jul. 2002

A Note on the Completeness of Certain Refinements of Resolution

Jean Goubault-Larrecq

LSV/CNRS UMR 8643, ENS Cachan
61, av. du président-Wilson
94235 Cachan Cedex, France

Abstract. Resolution and many refinements thereof have now been known for a long time. Completeness is usually proved by semantic means (e.g., semantic trees, Bachmair-Ganzinger forcing), or by syntactic tricks (Bledsoe's excess literal technique). The purpose of this paper is to note that there is a completely proof-theoretic way of proving completeness for several refinements of resolution, resembling Gentzen's method of cut-elimination. While this technique has a number of shortcomings compared to the semantic arguments cited above, it is valuable in that the completeness proofs for different refinements are the same. We have found this proof technique to be effective in teaching the ins and outs of refinements of resolution to masters level students. This can also be used to extract propositional proofs in one resolution format from resolution proofs in some other format automatically; in the first-order case, the same technique allows one to extract ordered resolution or hyperresolution proofs from proofs in any other resolution format mechanically.

1 Introduction

Context and related work. Resolution [10] has several refinements, among which hyperresolution, semantic resolution, ordered strategies, and combinations thereof, most of which being complete [2]. The classical proofs of their completeness is different in each case, and sometimes feel ad hoc. Amongst these techniques, we find Kowalski and Hayes' semantic tree technique [9], which works nicely for ordered resolution, hyperresolution and semantic resolution; Bachmair and Ganzinger's forcing technique [1], which is often used for ordered resolution with selection and provides an explicit model construction in case no refutation can be found. These semantic methods have several advantages, including the fact that it is easy to show that several deletion strategies (tautology elimination, subsumption) as well as additional rules (e.g., splitting [5] or condensation [8]) can be used without destroying the completeness of the base calculus. One syntactic method of proving completeness is Boyer's *excess literal technique*, which applies notably to Boyer's locking and to semantic resolution, including hyperresolution: see [2].

It is then usually somewhat awkward to teach students a course in resolution techniques, as refinements look like a hodge-podge of ad hoc tricks, with specific semantic or syntactic completeness arguments. The purpose of this paper is threefold:

– First, to show one unifying intuition behind several refinements of resolution: resolution, at least in the propositional case, is Gentzen's cut rule, and there are many ways that a proof using only cuts can be rearranged by permuting and distributing cuts across each other. Normal proofs will then be of specific forms, allowing one to reduce non-determinism in proof search. This is very much in the spirit of Gentzen's Hauptsatz for sequent calculi [13]. While permuting and distributing cuts is easy, showing that this process terminates is slightly more challenging. (Note that weak termination would be enough, but it is not much harder to show strong termination: all rewrites terminate.)
– Second, to provide a simple argument by which termination is ensured, hence from which completeness follows. This simple argument (condition (5) below) is enough to retrieve some of the most well-known refinements of resolution.
– Third, to provide effective translations between resolution formats. For example, our argument allows one to rewrite any ordered resolution refutation into a positive hyperresolution refutation from the same set of clauses. This can be used to provide human-readable proofs from machine-generated proofs; positive hyperresolution derivations, notably, tend to be more readable than ordered resolution proofs.

On the other hand, it is not the purpose of this paper to show that resolution refinements are still complete in the presence of such or such deletion strategy, for which semantic trees or Bachmair and Ganzinger's technique are preferable. This is mostly an orthogonal concern. For example, it is still possible to show that subsumed clauses and tautologies can be eliminated, when they can, by syntactic methods [7]. Neither is it the purpose of this paper to introduce new refinements of resolution, or to introduce a universal completeness proof. In particular, it seems that certain refinements of resolution, e.g., ordered resolution with free selection of negative literals, are not easily amenable to the technique described here.

Outline. Because we can always rest on lifting arguments, we mostly deal with *propositional* resolution in this paper—that is, until Section 5. We fix notations and recall the resolution principle in Section 2. We then introduce our proof transformation rules in Section 3, and give sufficient conditions for them to terminate, thus implying completeness. Section 4 illustrates a number of known refinements that can be shown complete by this technique. Although the stress is put on the propositional case in this paper, we deal with the first-order case in Section 5; this is more difficult to tackle without going through lifting first. This effort pays up: our technique provides an effective translation from any first-order resolution refutation to ordered, or to hyperresolution refutations. We conclude in Section 6.

2 Resolution

Let us fix a vocabulary of *atoms* A, B, ...; *literals* L are either positive atoms $+A$ or negative atoms $-A$. *Clauses* C, D, ..., are finite sets of literals, seen as disjunctions.

On propositional formulas, which is the case we deal with except in Section 5, the *resolution rule* is nothing else than Gentzen's Cut rule:

$$\frac{C, +A \quad C', -A}{C, C'}\,(Cut)$$

where comma denotes union, and disjoint union in premises (in particular, $+A \notin C$, $-A \notin C'$).

We write $\Box$ the empty clause, $\overline{L}$ the negation of literal L, i.e., $\overline{+A} = -A$, $\overline{-A} = +A$. In the Cut rule above, A is called the *cut formula*. A *resolution proof* of a clause C from the set S of clauses is any finite tree of resolution inferences (instance of Cut) whose leaves (at the top) are clauses in S and whose root (at the bottom) is C. A *refutation* from S is a resolution proof of $\Box$ from S.

The completeness of resolution, i.e., that there is a refutation from S whenever S is inconsistent, can be established by semantic means, or by appealing to the syntactic device of cut elimination: let $\mathbf{LK} + S$ denote Gentzen's sequent calculus for classical logic augmented with non-logical axioms taken from S (clauses being read as sequents), then eliminating cuts from any $(\mathbf{LK} + S)$-proof of the empty sequent yields one where the only rule is (Cut) [6,7]. That resolution is complete will be assumed in the sequel. To show that some refinement of resolution is complete, we only need to rewrite any given refutation of S into a refutation of S that obeys the constraints of the refinement. We shall do this by using rewrite rules ((1)–(2) below) that express all possible ways of permuting one cut past another.

$$\dfrac{\dfrac{\vdots\,\pi_0}{C, L} \quad \dfrac{\dfrac{\vdots\,\pi_1}{C_1', \overline{L}, L'} \quad \dfrac{\vdots\,\pi_2}{C_2, \overline{L'}}\,(\dagger)}{C_1', C_2, \overline{L}}\,(*)}{C, C_1', C_2} \longrightarrow \dfrac{\dfrac{\vdots\,\pi_0}{C, L} \quad \dfrac{\vdots\,\pi_1}{C_1', \overline{L}, L'}\,(\ddagger)}{C, C_1', L'} \quad \dfrac{\vdots\,\pi_2}{C_2, \overline{L'}}\,(**)}{C, C_1', C_2} \tag{1}$$

$$\dfrac{\dfrac{\vdots\,\pi_0}{C, L} \quad \dfrac{\dfrac{\vdots\,\pi_1}{C_1', \overline{L}, L'} \quad \dfrac{\vdots\,\pi_2}{C_2', \overline{L}, \overline{L'}}\,(\dagger)}{C_1', C_2', \overline{L}}\,(*)}{C, C_1', C_2'} \longrightarrow \dfrac{\dfrac{\vdots\,\pi_0}{C, L} \quad \dfrac{\vdots\,\pi_1}{C_1', \overline{L}, L'}\,(\ddagger)}{C, C_1', L'} \quad \dfrac{\dfrac{\vdots\,\pi_0}{C, L} \quad \dfrac{\vdots\,\pi_2}{C_2', \overline{L}, \overline{L'}}\,(\ddagger)}{C, C_2', \overline{L'}}\,(**)}{C, C_1', C_2'} \tag{2}$$

The $(*)$, $(**)$, $(\dagger)$, $(\ddagger)$ marks have been added for future reference. For example, we shall say that the $(\ddagger)$-*cuts* are those marked with $(\ddagger)$ (the topmost cuts in each rule, on the right).

Recall that we assume that in premises such as $C_1', \overline{L}, L'$, the literals $\overline{L}$ and L' are distinct and not in C_1'.

These rules do not terminate in general. However we shall find a series of conditions that ensure termination in the next section, and demonstrate that several known refinements of resolution obey these conditions.

3 Completeness via Selection Functions

Let us specify a refinement of resolution by means of a *selection function* ς mapping each clause C to a subset of literals that we are allowed to take as cut formulas in the (Cut) rule.

For example, if ς returns the set of all $>$-maximal literals for some ordering $>$, and $>$ compares literals by comparing the underlying atoms, then we get ordered resolution. We shall give more examples of ς functions in Section 4.

We ask the selection function ς to obey the following axioms. First, ς selects from literals in the clause:

$$\varsigma(C) \subseteq C \tag{3}$$

Then, ς should select the unique literal from each *unit* (one-literal) clause:

$$\varsigma(L) = \{L\} \tag{4}$$

for every literal L. That is, it is not allowed to select nothing from a unit clause.

Before we introduce the last condition, define the ς-*cut* rule as the restriction of the (Cut) rule where $+A \in \varsigma(C, A)$ and $-A \in \varsigma(C', -A)$. A resolution proof is a ς-*resolution proof* if and only if it only uses ς-cuts. We then require the following condition:

In (1), (2), if the $(*)$-cut is a ς-cut, and the $(\dagger)$-cut is not, then the $(\ddagger)$-cuts are ς-cuts. (5)

If π is a non-ς-resolution refutation from S, there must be a lowest instance of (Cut) that is not a ς-cut. This lowest non-ς-cut cannot be the last one. Indeed, because $\varsigma(L) = \{L\}$ for every unit clause L, any final instance of (Cut) in a refutation, which derives $\square$ and therefore must be a cut between two unit clauses, is a ς-cut. So there must be a ς-cut below the lowest non-ς-cut. That is, π contains the following configuration:

$$\dfrac{\dfrac{\vdots\,\pi_0}{C, L} \qquad \dfrac{\dfrac{\vdots\,\pi_1}{C_1, L'} \qquad \dfrac{\vdots\,\pi_2}{C_2, \overline{L}'}}{C', \overline{L}}\ \text{non-}\varsigma\text{-cut}}{C, C'}\ \varsigma\text{-cut} \tag{6}$$

which we call a *redex*. Condition (5) says that (1) and (2) rewrite redexes to configurations where the ς-cut was permuted upwards: all topmost cuts in right-hand sides are required to be ς-cuts.

Since the non-ς-cut in the redex (6) is an instance of (Cut), the clause C_1, C_2 must equal $C', \overline{L}$. Then either C_1 contains $\overline{L}$ but not C_2 (or by symmetry C_1 does not but C_2 does), or both C_1 and C_2 contain L: these yield the left-hand sides of rules (1) and (2) respectively.

We claim that under the assumptions (3), (4), (5), these rewrite rules (1) and (2) restricted to apply to redexes (6) terminate. It will follow that we can always transform any resolution refutation into a ς-resolution refutation; in particular, ς-resolution will be complete.

To establish termination, define an interpretation of proofs as first-order terms built on one constant a and two binary function symbols f_0 and f_1, representing ς-cuts and non-ς-cuts respectively. Recall that S is the set of clauses that we start from:

$$[\![C]\!] = a \qquad (C \in S)$$

$$\left[\!\!\left[\dfrac{\begin{matrix}\vdots\,\pi_1 & \vdots\,\pi_2\\ C, L & C', \overline{L}\end{matrix}}{C, C'}\ \varsigma\text{-cut} \right]\!\!\right] = f_0([\![\pi_1]\!], [\![\pi_2]\!])$$

$$\left[\!\!\left[\dfrac{\begin{matrix}\vdots\,\pi_1 & \vdots\,\pi_2\\ C, L & C', \overline{L}\end{matrix}}{C, C'}\ \text{non-}\varsigma\text{-cut} \right]\!\!\right] = f_1([\![\pi_1]\!], [\![\pi_2]\!])$$

We assume f_0 and f_1 to be commutative, i.e., $f_i(s,t) = f_i(t,s)$ to avoid ambiguity in this translation. (We might also untie the knot by imposing that s be, say, the translation of the premise where the cut formula is positive, but this would unnecessarily duplicate the cases to handle.)

Translating (1) and (2) through $[\![\_]\!]$ yields the following rewrite rules:

$$f_0(u_0, f_1(u_1, u_2)) \longrightarrow f_i(f_0(u_0, u_1), u_2) \tag{7}$$

$$f_0(u_0, f_1(u_1, u_2)) \longrightarrow f_i(f_0(u_0, u_1), f_0(u_0, u_2)) \tag{8}$$

where i ranges over $\{0, 1\}$. That is, $\pi \longrightarrow \pi'$ by rules (1) or (2) implies $[\![\pi]\!] \longrightarrow [\![\pi']\!]$ by rules (7) or (8); this is an easy check, using condition (5).

Lemma 1 (Termination). *The rewrite system (7), (8) terminates.*

Proof. Let SN be the set of terms u that are terminating, i.e. such that every rewrite starting from u is finite. Let the *contexts* be terms $\mathcal{C}$ with one hole, denoted $[]$ ($\mathcal{C} ::= [] | f_i(t, \mathcal{C})$ where t ranges over terms); $\mathcal{C}[u]$ denotes $\mathcal{C}$ with the hole replaced by the term u. Similarly, let $\mathcal{C}[\mathcal{C}']$ be the context obtained by replacing the hole of $\mathcal{C}$ by the context $\mathcal{C}'$. Define *SN-contexts* inductively by: $[]$ is an SN-context, and if $\mathcal{C}$ is an SN-context and $v \in SN$, then $f_0(v, \mathcal{C})$ is an SN-context. Finally, say that a term u is *reducible* if and only if $\mathcal{C}[u] \in SN$ for every SN-context $\mathcal{C}$; the set of reducible terms is written RED.

Observe that: (a) $RED \subseteq SN$. This is because $[]$ is an SN-context.

Note also: (b) if $s \in SN$ and $t \in RED$, then $f_0(s,t) \in RED$. Indeed, for every SN-context $\mathcal{C}$, $\mathcal{C}[f_0(s, [])]$ is an SN-context by construction and $t \in RED$, so $\mathcal{C}[f_0(s,t)] \in SN$.

We also have: (c) if $s \in RED$ and $s \longrightarrow s'$ then $s' \in RED$. Indeed, for any SN-context $\mathcal{C}, \mathcal{C}[s] \in SN$, so the one-step reduct $\mathcal{C}[s']$ is in SN, too.

We claim that: (d) if $s, t \in RED$, then $f_1(s,t) \in RED$. To this end, let $\mathcal{C}$ be any SN-context, and let us show that $\mathcal{C}[f_1(s,t)] \in SN$. To fix ideas, write $\mathcal{C}$ as $f_0(u_1, f_2(u_2, \ldots f_0(u_n, []), \ldots))$, and let us show the claim by induction on $n, s, t, u_1, \ldots, u_n$ ordered lexicographically, where terms s, t, u_1, …, u_n are compared via the $\longrightarrow$ relation—which is well-founded on $s, t \in RED \subseteq SN$ and on

$u_1, \ldots, u_n \in SN$. Then look at one-step reducts from $\mathcal{C}[f_1(s,t)]$. Some of them are obtained by contracting a redex in s, in t, or in some u_i, $1 \leq i \leq n$: then by (c) and the induction hypothesis the obtained one-step reduct is in SN; another is obtained by contracting the redex $f_0(u_n, f_1(s,t))$, provided $n \geq 1$. If this reduces by (8), the reduct is $\mathcal{C}'[f_i(f_0(u_n,s), f_0(u_n,t))]$ where $\mathcal{C}' = f_0(u_1, f_0(u_2, \ldots f_0(u_{n-1}, []), \ldots))$. Note that $f_0(u_n,s)$ and $f_0(u_n,t)$ are in RED by (b). So, if $i = 1$, we may conclude by the induction hypothesis (with n decreased by 1, and $\mathcal{C}$ replaced by $\mathcal{C}'$); if $i = 0$, then $\mathcal{C}'[f_0(f_0(u_n,s), f_0(u_n,t))]$ is in SN because $\mathcal{C}'[f_0(f_0(u_n,s), f_0(u_n,[]))]$ is an SN-context ($f_0(u_n,s)$ in particular is in SN by (c) and (a)) and $t \in RED$. Similarly, every one-step-reduct obtained by contracting $f_0(u_n, f_1(s,t))$ by (7) is in SN: if $i = 1$, $\mathcal{C}'[f_1(s, f_0(u_n,t))]$ is in SN by induction hypothesis, using the fact that $s \in RED$ and $f_0(u_n,t) \in RED$ by (b); if $i = 0$, $\mathcal{C}'[f_0(s, f_0(u_n,t))]$ is in SN since $\mathcal{C}'[f_0(s, f_0(u_n,[]))]$ is an SN-context (using $s \in RED \subseteq SN$) and $t \in RED$. Since every one-step reduct of $\mathcal{C}[f_1(s,t)]$ is in SN, $\mathcal{C}[f_1(s,t)]$ is in SN, too. Since $\mathcal{C}$ is arbitrary, $f_1(s,t)$ is reducible.

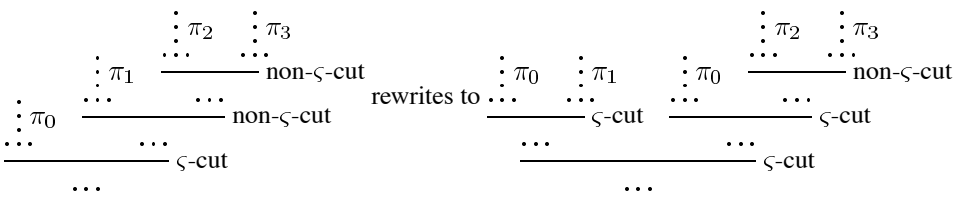
It is easy to see that $a \in RED$. Indeed, for every $u_1, \ldots, u_n \in SN$, $f_0(u_1, f_2(u_1, \ldots f_0(u_n, a), \ldots))$ is indeed in SN (an easy induction on $u_1, \ldots, u_n$).

It follows that every term u is reducible, by structural induction on u. We have just dealt with the base case $u = a$, and the inductive cases are dealt with by (d) when $u = f_1(s,t)$, by (b) and (a) when $u = f_0(s,t)$. Since every term is reducible, by (a) every term is in SN. □

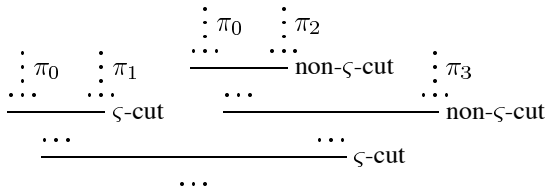
Most standard methods in rewriting fail to prove Lemma 1. In particular, the *recursive path ordering* [3] cannot deal with rules (7) or (8) when $i = 0$. In fact, this rewrite system is not *simply terminating*, and being included in a recursive path ordering implies simple termination. Recall that a rewrite system R is simply terminating if and only if R plus the *simplification* rules $f_i(s,t) \longrightarrow s$, $f_i(s,t) \longrightarrow t$ is terminating. A counter-example is:

$$\begin{aligned} f_0(f_1(a,a), f_1(a,a)) &\longrightarrow f_0(f_0(f_1(a,a),a), f_0(f_1(a,a),a)) && \text{by (8)} \\ &\longrightarrow^2 f_0(f_1(a,a), f_1(a,a)) && \text{by simplification} \end{aligned}$$

Condition (5) is also maximal in that any liberalization leads to non-termination; in fact even weak termination (existence of normal forms) fails with any liberalized form of condition (5). Allowing some $(\ddagger)$-cuts to be non-ς-cuts while the $(**)$-cut is a ς-cut would mean creating redexes on the right-hand side, leading immediately to non-terminating behavior. More subtly, allowing some $(\ddagger)$-cuts as well as the $(**)$-cut to be non-ς-cuts, which might seem a benign extension, also leads to non-termination. Consider the case of rule (1) for example, then allowing the latter would enable the following non-terminating behavior. We have elided the actual clauses, which are unimportant. First



by (2), then to



if we allow for the indicated liberalization of rule (1). The latter derivation is then again a ς-cut under a non-ς-cut under a non-ς-cut, which allows us to start this cycle of reductions all over again.

Having proved Lemma 1, we are now done:

Theorem 1. *Every resolution refutation from S can be effectively transformed into a ς-resolution refutation from S, provided (3), (4), (5) hold.*

Corollary 1. *Every refinement of resolution based on a selection function ς satisfying (3), (4), (5) is complete.*

4 Applications

We first show that Corollary 1 allows us to justify some standard refinements of resolution. As announced in the introduction, we won't deal with every known refinement of resolution. In particular, free selection functions [1], even in the propositional case that we are now considering, do not seem to fit well in this framework.

4.1 Ordered resolution.

Let $>$ be a strict ordering on atoms. Ordered resolution is the case where $\varsigma(C)$ is the set of all literals $\pm A$ such that A is *maximal* in C: i.e., there is no $\pm B$ in C, with a possibly different sign, such that $B > A$.

Clearly, (3) and (4) hold. For (5), assume that $L = \pm A$, $L' = \pm A'$, and $(*)$ is a ς-cut in (1):

$$\dfrac{\dfrac{\vdots\,\pi_0}{C, \pm A} \quad \dfrac{\dfrac{\vdots\,\pi_1}{C'_1, \overline{\pm A}, \pm A'} \quad \dfrac{\vdots\,\pi_2}{C_2, \overline{\pm A}'}}{C'_1, C_2, \overline{\pm A}}\,(\dagger)}{C, C'_1, C_2}\,(*) \longrightarrow \dfrac{\dfrac{\vdots\,\pi_0}{C, \pm A} \quad \dfrac{\vdots\,\pi_1}{C'_1, \overline{\pm A}, \pm A'}}{C, C'_1, \pm A'}\,(\ddagger) \quad \dfrac{\vdots\,\pi_2}{C_2, \overline{\pm A}'}}{C, C'_1, C_2}\,(**)$$

In other words, (a) A is maximal in $C, \pm A$, and (b) no atom in C'_1, C_2 is greater than A in $>$. If $(\ddagger)$ is not a ς-cut, then there must be an atom greater than A in $C'_1, \overline{\pm A}$. It cannot be in C'_1 by (b), so we must have $A' > A$. So A' is maximal in $C'_1, \overline{\pm A}, \pm A'$, otherwise there would be a greater atom B in $C'_1, \overline{\pm A}$: then $B > A' > A$, contradicting (b). A' must also be maximal in $C_2, \overline{\pm A}'$, otherwise there would be a greater atom B in C_2, so $B > A' > A$, contradicting (b) again.

The argument for rule (2) is similar:

$$\dfrac{\dfrac{\vdots\,\pi_0}{C,\pm A}\quad\dfrac{\vdots\,\pi_1\qquad\vdots\,\pi_2}{C_1',\overline{\pm A},\pm A'\quad C_2',\overline{\pm A},\overline{\pm A}'}\,(\dagger)}{C,C_1',C_2'}\,(*)\longrightarrow\dfrac{\dfrac{\vdots\,\pi_0\qquad\vdots\,\pi_1}{C,\pm A\quad C_1',\overline{\pm A},\pm A'}\,(\ddagger)\qquad\dfrac{\vdots\,\pi_0\qquad\vdots\,\pi_2}{C,\pm A\quad C_2',\overline{\pm A},\overline{\pm A}'}\,(\ddagger)}{C,C_1',C_2'}\,(**)$$

Since $(*)$ is a ς-cut, (a) A is maximal in $C,\pm A$, and (b) $\overline{\pm A}$ is maximal in $C_1',C_2',\overline{\pm A}$. If some $(\ddagger)$-cut is not a ς-cut, say the left one by symmetry, then A is not maximal in $C_1',\overline{\pm A},\pm A'$, so by (b) $A'>A$. Then A' is maximal in both premises of $(\dagger)$. Therefore (5) holds.

In particular by Corollary 1 ordered resolution is complete.

There was in fact an easier syntactic proof of termination here. Modify the $[\![\_]\!]$ translation so that:

$$\left[\!\!\left[\dfrac{\vdots\,\pi_1\qquad\vdots\,\pi_2}{C,\pm A\quad C',\overline{\pm A}}\right]\!\!\right]=f_A([\![\pi_1]\!],[\![\pi_2]\!])$$

where for each atom A there is a new binary commutative function symbol f_A. Then the termination of the rewrite system on proofs of Section 3 in this case can be shown by using a multiset path ordering [3] with the precedence $f_A\succ f_B$ iff $A>B$. This applies since there are only finitely many atoms in any given refutation, therefore $\succ$ is well-founded.

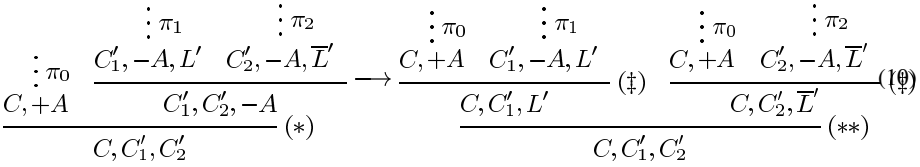
4.2 Positive hyperresolution.

Let $\varsigma(C)$ be the set of *negative* atoms $-A$ in C if any, otherwise $\varsigma(C)=C$. A ς-cut between two parent clauses $C,+A$ and $C',-A$ is then such that $C,+A$ only contains positive atoms, otherwise $+A$ cannot be selected. Conversely, every positive hyperresolution step defined as (Cut) where one premise is a positive clause (a clause containing only positive atoms) is a ς-cut with this definition of ς.

Again, conditions (3) and (4) are clear. If the left-hand side of (1) (or (2)) is a redex, we claim that C,L must be a positive clause. Indeed, since the $(*)$-cut is a positive hyperresolution step, the only other possibility is that $C_1',C_2,\overline{L}$, resp. $C_1',C_2',\overline{L}$, is a positive clause; then one of the premises of the $(\dagger)$-cut must be positive, so the $(\dagger)$-cut would be a ς-cut, contradicting the fact that we have got a redex. So C,L is a positive clause, hence the $(\ddagger)$-cut is a positive hyperresolution step, i.e., a ς-cut. Similarly for (2). Since C,L is a positive clause in any case, condition (5) holds. Therefore positive hyperresolution is complete.

In fact, we can spell out the rewrite rules (1), (2) in this case as follows:

$$\dfrac{\dfrac{\vdots\,\pi_0}{C,+A}\quad\dfrac{\vdots\,\pi_1\qquad\vdots\,\pi_2}{C_1',-A,L'\quad C_2,\overline{L}'}}{C,C_1',C_2}\,(*)\longrightarrow\dfrac{\dfrac{\vdots\,\pi_0\qquad\vdots\,\pi_1}{C,+A\quad C_1',-A,L'}\,(\ddagger)\qquad\vdots\,\pi_2}{C,C_1',C_2}\,(**)\qquad(9)$$



where it should be clearer that the $(\ddagger)$-cuts must be positive hyperresolution steps. Here C is positive, however we shall use the same rules in Section 4.4 without this restriction.

The alternate form of hyperresolution where *macro-steps* with $n+1$ premises are used, consisting of a non-positive clause (the *nucleus*) and n positive clauses (the *electrons*) [2] which are resolved in n steps to yield a new *positive* clause is complete, too. It is enough to notice that in a positive hyperresolution refutation, as defined above, if some ς-cut has a non-positive conclusion, then going down the refutation we eventually reach a positive clause: at the latest, $\Box$ is a positive clause. Hence we can extract a refutation consisting entirely of macro-steps from any ς-resolution refutation.

We let the interested reader check that the only role of $\Box$ in the argument of Section 3 is to show that the last instance of (Cut) in a refutation is a ς-cut. In positive hyperresolution we may generalize: any resolution derivation of any *positive* clause C_0 must end in a positive hyperresolution step (a ς-cut). We can then replay Section 3: every resolution derivation of C_0 can be effectively transformed into a positive hyperresolution derivation of C_0. In particular, we get the well-known fact that positive hyperresolution derives exactly the same positive clauses as unconstrained resolution.

4.3 Negative hyperresolution, semantic resolution.

Negative hyperresolution is obtained similarly by letting $\varsigma(C)$ be the set of all *positive* atoms $+A$ in C if any, otherwise $\varsigma(C) = C$. In general, I-resolution, a.k.a. *semantic resolution*, where I is a set of atoms, that is, a Herbrand interpretation, is obtained by letting $\varsigma(C)$ be the set of all literals that are true in I if any, otherwise $\varsigma(C) = C$. Just as in the case of positive hyperresolution, C, L in rules (1) and (2) must be restricted to be false in I.

We do not need to use Corollary 1 here, though: completeness of I-resolution follows from that of positive hyperresolution by renaming every atom $\pm A$ to $\overline{\pm A}$ in clauses whenever $A \in I$, and noticing that such a renaming preserves the existence of refutations.

4.4 Semi-ordered hyperresolution

While imposing an ordering constraint on both premises of positive hyperresolution steps destroys completeness, it is well-known that imposing that A be maximal only in the positive clause $C, +A$ leads to a complete refinement of resolution. This is called *semi-ordered* hyperresolution in [7] to distinguish it from the aforementioned incomplete ordered refinement of hyperresolution. This is obtained by letting $\varsigma(C)$ be the set of all negative atoms $-A$ in C if any, otherwise $\varsigma(C)$ is the set of *maximal* (positive) atoms $+A$ in C.

Curiously, Corollary 1 does not apply directly. The reason is that condition (5) is not satisfied. Indeed, note that ς-cuts are cuts where one premise is positive $C, +A$ and the cut formula $+A$ is maximal. Then in rule (1) it might be the case that $C'_1, C_2, \overline{L}$ is a positive clause with $L = -A$ and A maximal in $C'_1, C_2, +A$ (whence $(*)$ is a ς-cut), but $C_2, \overline{L'}$ is also positive with $L' = -A'$ but A' is not maximal in $C_2, +A'$ (whence $(\dagger)$ is not a ς-cut); in this case there is no reason why $(\ddagger)$ should be a ς-cut: neither $C, L = C, -A$ nor $C'_1, \overline{L}, L' = C'_1, +A, -A'$ is positive, in particular.

It might be that there is still a way of showing that (1) and (2) terminate in this case, too. However, an easier way of converting any resolution refutation π into a semi-ordered positive hyperresolution refutation is to proceed in two steps. First convert π into an ordered refutation π' as in Section 4.1, then convert π' into a positive hyperresolution refutation π'' as in Section 4.2, using rules (9) and (10). Then apply the following lemma.

Call an instance of (Cut) *$+$-ordered* if and only the cut formula A is maximal in the one premise where A is positive. A resolution derivation is $+$-ordered if every step in it is. It is clear that ordered resolution derivations are $+$-ordered, while positive hyperresolution derivations that are $+$-ordered are exactly the semi-ordered positive hyperresolution derivations. For convenience, assume that $>$ is total here: if $+A$ is maximal in $C, +A$ then A is greater than or equal to all atoms in C.

Lemma 2. *If $\pi \longrightarrow \pi'$ by rules (9) or (10) and π is $+$-ordered, then π' is $+$-ordered.*

Proof. The $(\ddagger)$-cuts are clearly $+$-ordered. It remains to show that the $(**)$-cuts are, too. If L' is a positive literal $+A'$ in (9), then by assumption $A' \geq C'_1, -A$; since $A \geq C$, in particular $A' \geq C'_1, C$, so the $(**)$-cut is $+$-ordered. If L' is a negative literal $-A'$ in (9) then by assumption $A' \geq C_2$, so again the $(**)$-cut is $+$-ordered. In the case of rule (10), by symmetry we may assume that $L' = +A'$; then by assumption $A' \geq C'_1, -A$, so $A' \geq C, C'_1$ since $A \geq C$; so the $(**)$-cut is $+$-ordered. □

It follows that the end refutation π'' obtained in the two-step process above is both a positive hyperresolution refutation and $+$-ordered, so it is a semi-ordered positive hyperresolution refutation.

Again, similar arguments show that semi-ordered negative hyperresolution or semi-ordered I-resolution are complete, and give an effective procedure to transform any resolution refutation in one of the required format.

5 The First-Order Case

While this is not completely immediate, the rewrite rules (1) and (2) generalize to the first-order case, where the resolution rule reads:

$$\frac{C, +A_1, \ldots, +A_m \quad C', -A'_1, \ldots, -A'_n}{(C, C')\sigma}$$

provided σ is the *most general unifier* of $A_1, \ldots, A_m, A'_1, \ldots, A'_n$. For any substitution σ, we write $A\sigma$ the result of applying the substitution σ to the atom A, and we use

similar notations for substituting in clauses. As is traditional, we leave implicit the fact that the parent clauses $C, +A_1, \ldots, +A_m$ and $C', -A'_1, \ldots, -A'_n$ are first renamed so that they have no free variable in common.

For short, write $\boldsymbol{A}$ for the set $A_1, \ldots, A_m$. Such sets will always be assumed to be non-empty, i.e., $m \geq 1$. Let $\mathrm{mgu}(\boldsymbol{A} = \boldsymbol{A'})$ denote the most general unifier of $A_1, \ldots, A_m, A'_1, \ldots, A'_n$. Extend this to literals and to several equality signs: $\mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L'}_1, \ldots, \boldsymbol{L}_k = \boldsymbol{L'}_k)$ is the most general common substitution (if any) that instantiate $L_{1\,1}, \ldots, L_{1\,m_1}, L'_{1\,1}, \ldots, L'_{1\,n_1}$ to the same literal L_1, and $\ldots$, and $L_{k\,1}, \ldots, L_{k\,m_k}, L'_{k\,1}, \ldots, L'_{k\,n_k}$ to the same literal L_k.

Let $d(\sigma)$, the *domain* of σ, be the set $\{x | x\sigma \neq x\}$. We shall always assume that most general unifiers are *idempotent*, i.e., for every $x \in d(\sigma)$, x is not free in $x\sigma$. If σ is idempotent, letting $E(\sigma)$ be the set of all equations $x = x\sigma$, $x \in d(\sigma)$, then $\sigma = \mathrm{mgu}(E(\sigma))$; furthermore, if σ and σ' are idempotent and $d(\sigma) \cap d(\sigma') = \emptyset$, then $\sigma\sigma' = \mathrm{mgu}(E(\sigma) \cup E(\sigma'))$ (we write $\sigma\sigma'$ the substitution mapping every variable x to $(x\sigma)\sigma'$), and if two sets of equations E and E' have the same unquantified equational consequences, then $\mathrm{mgu}(E) = \mathrm{mgu}(E')$. In particular if $\sigma = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2)$ then $\mathrm{mgu}(\boldsymbol{L}'_1\sigma = \boldsymbol{L}'_2, \boldsymbol{L}_1 = \boldsymbol{L}_2) = \mathrm{mgu}(\boldsymbol{L}'_1 = \boldsymbol{L}'_2, \boldsymbol{L}_1 = \boldsymbol{L}_2)$.

The rewrite rules (1) and (2) change as follows in the first-order case. The left-hand side of rule (1) now reads:

$$\dfrac{\dfrac{\vdots\,\pi_0 \quad C, \boldsymbol{L}_1}{C\sigma', C'_1\sigma\sigma', C_2\sigma\sigma'} \quad \dfrac{\vdots\,\pi_1 \quad \vdots\,\pi_2}{C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1 \quad C_2, \overline{\boldsymbol{L}'_2}} \; \sigma = \mathrm{mgu}(\boldsymbol{L}'_1 = \boldsymbol{L}'_2)}{C'_1\sigma, C_2\sigma, \overline{\boldsymbol{L}_2\sigma}} \; \sigma' = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2\sigma) \tag{11}$$

Now, because clauses are renamed apart, $C\sigma' = C\sigma\sigma'$. Also, $\sigma\sigma' = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2\sigma, \boldsymbol{L}'_1 = \boldsymbol{L}'_2) = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2, \boldsymbol{L}'_1 = \boldsymbol{L}'_2)$. It follows in particular that $\sigma'' = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2)$ exists, that $\sigma''' = \mathrm{mgu}(\boldsymbol{L}'_1\sigma'' = \boldsymbol{L}'_2)$ exists and that $\sigma''\sigma''' = \mathrm{mgu}(\boldsymbol{L}'_1\sigma'' = \boldsymbol{L}'_2, \boldsymbol{L}_1 = \boldsymbol{L}_2) = \mathrm{mgu}(\boldsymbol{L}'_1 = \boldsymbol{L}'_2, \boldsymbol{L}_1 = \boldsymbol{L}_2) = \sigma\sigma'$. We can then rewrite the redex above to:

$$\dfrac{\dfrac{\vdots\,\pi_0 \quad C, \boldsymbol{L}_1}{C\sigma'', C'_1\sigma'', \boldsymbol{L}'_1\sigma''} \quad \dfrac{\vdots\,\pi_1 \quad \vdots\,\pi_2}{C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1 \quad C_2, \overline{\boldsymbol{L}'_2}}}{C\sigma''\sigma''', C'_1\sigma''\sigma''', C_2\sigma'''} \tag{12}$$

where $C_2\sigma''' = C_2\sigma''\sigma'''$. Since $\sigma''\sigma''' = \sigma\sigma'$, the conclusion of (12) is the same as that of (11).

The case of rule (2) is slightly more complicated. The left-hand side is:

$$\dfrac{\dfrac{\vdots\,\pi_0 \quad C, \boldsymbol{L}_1}{C\sigma', C'_1\sigma\sigma', C'_2\sigma\sigma'} \quad \dfrac{\vdots\,\pi_1 \quad \vdots\,\pi_2}{C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1 \quad C'_2, \overline{\boldsymbol{L}_3}, \overline{\boldsymbol{L}'_2}} \; \sigma = \mathrm{mgu}(\boldsymbol{L}'_1 = \boldsymbol{L}'_2)}{C'_1\sigma, C'_2\sigma, \overline{\boldsymbol{L}_2\sigma}, \overline{\boldsymbol{L}_3\sigma}} \; \sigma' = \mathrm{mgu}(\boldsymbol{L}_1 = (\boldsymbol{L}_2, \boldsymbol{L}_3)\sigma) \tag{13}$$

where $\boldsymbol{L}_2, \boldsymbol{L}_3$ denotes the union of the sets $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$. Recall that both $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ are assumed not empty. Again $C\sigma' = C\sigma\sigma'$, and $\sigma\sigma' = \mathrm{mgu}(\boldsymbol{L}_1 = (\boldsymbol{L}_2, \boldsymbol{L}_3)\sigma, \boldsymbol{L}'_1 = \boldsymbol{L}'_2) = \mathrm{mgu}(\boldsymbol{L}_1 = (\boldsymbol{L}_2, \boldsymbol{L}_3), \boldsymbol{L}'_1 = \boldsymbol{L}'_2)$.

Let now ρ be a renaming (one-to-one, mapping variables to variables) substitution so that $C\rho, \boldsymbol{L}_1\rho$ as no free variable in common with $C, \boldsymbol{L}_1$. Then $\sigma\sigma'$ is a unifier of $\boldsymbol{L}_1 = \boldsymbol{L}_2$, so $\sigma'' = \mathrm{mgu}(\boldsymbol{L}_1, \boldsymbol{L}_2)$ exists; $\rho^{-1}\sigma\sigma'$ is a unifier of $\boldsymbol{L}_1\rho = \boldsymbol{L}_3$, so $\sigma''' = \mathrm{mgu}(\boldsymbol{L}_1\rho, \boldsymbol{L}_3)$ exists. Then Note that $\sigma\sigma'$ and $\rho^{-1}\sigma\sigma'$ have disjoint domains. So the union $\sigma\sigma' \cup \rho^{-1}\sigma\sigma'$ makes sense, and we claim that it unifies $\boldsymbol{L}'_1\sigma'' = \boldsymbol{L}'_2\sigma'''$. Indeed, $\sigma\sigma' \cup \rho^{-1}\sigma\sigma'$ unifies $\boldsymbol{L}_1 = (\boldsymbol{L}_2, \boldsymbol{L}_3)$ and $\boldsymbol{L}'_1 = \boldsymbol{L}'_2$ (because $\sigma\sigma'$ is their common most general unifier), and also $\boldsymbol{L}_1\rho = (\boldsymbol{L}_2, \boldsymbol{L}_3)$ and $\boldsymbol{L}'_1 = \boldsymbol{L}'_2$. In particular, $\sigma\sigma' \cup \rho^{-1}\sigma\sigma'$ unifies $\boldsymbol{L}_1 = \boldsymbol{L}_2, \boldsymbol{L}_1\rho = \boldsymbol{L}_3, \boldsymbol{L}'_1 = \boldsymbol{L}'_2$, and is therefore an instance of $\mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2, \boldsymbol{L}_1\rho = \boldsymbol{L}_3, \boldsymbol{L}'_1 = \boldsymbol{L}'_2) = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2, \boldsymbol{L}_1\rho = \boldsymbol{L}_3, \boldsymbol{L}'_1\sigma'' = \boldsymbol{L}'_2\sigma''')$. (For any atom, clause or substitution X, an *instance* of X is any atom, clause or substitution $X\theta$, for any substitution θ. We then say that X is *more general* than Y if and only if Y is an instance of X.)

We may then generate the derivation:

$$\dfrac{\dfrac{\begin{array}{cc}\vdots\,\pi_0 & \vdots\,\pi_1 \\ C, \boldsymbol{L}_1 & C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1\end{array}}{C\sigma'', C'_1\sigma'', \boldsymbol{L}'_1\sigma''} \quad \dfrac{\begin{array}{cc}\vdots\,\pi_0 & \vdots\,\pi_2 \\ C\rho, \boldsymbol{L}_1\rho & C'_2, \overline{\boldsymbol{L}_3}, \overline{\boldsymbol{L}'_2}\end{array}}{C\rho\sigma''', C'_2\sigma''', \overline{\boldsymbol{L}'_2}\sigma'''}}{C\sigma''\sigma'''', C\rho\sigma'''\sigma'''', C'_1\sigma''\sigma'''', C'_2\sigma'''\sigma''''} \tag{14}$$

Now because all clauses are renamed apart, the bottom clause in (14) is also

$$C\sigma^2\sigma'''', C\rho\sigma^2\sigma'''', C'_1\sigma^2\sigma'''', C'_2\sigma^2\sigma''''$$

Then $\sigma^2\sigma'''' = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2, \boldsymbol{L}_1\rho = \boldsymbol{L}_3, \boldsymbol{L}'_1\sigma'' = \boldsymbol{L}'_2\sigma''') = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2, \boldsymbol{L}_1\rho = \boldsymbol{L}_3, \boldsymbol{L}'_1 = \boldsymbol{L}'_2)$, and we have already noticed that $\sigma\sigma' \cup \rho^{-1}\sigma\sigma'$ is an instance of this.

In general, $\sigma^2\sigma''''$ is not the same as $\sigma\sigma' \cup \rho^{-1}\sigma\sigma'$, however, contrarily to the case for rule (1). For example, take $\boldsymbol{L}_2$ be the atom $f(x_2, y_2)$, take $\boldsymbol{L}_3$ to be $f(y_3, x_3)$, $\boldsymbol{L}'_1$ to be $g(x_2, y_2)$, $\boldsymbol{L}'_2$ to be $g(x_3, y_3)$. Let us say that $\boldsymbol{L}_1$ is $f(x_1, y_1)$, and assume that every two variables with different names are distinct. Then $\sigma\sigma'$, and therefore also $\sigma\sigma' \cup \rho^{-1}\sigma\sigma'$, map x_2, y_2, x_3, y_3 to the same term. On the other hand, $\sigma^2\sigma''''$ only equates x_2 with x_3 and y_2 with y_3.

With the rules (11) $\rightarrow$ (12) and (13) $\rightarrow$ (14) we can now define a transformation on derivations as follows: $\pi \rightarrowtriangle \pi'$ if and only if either π is of the form (11) and π' is (12), or π is of the form (13) and π' is (14), or π is

$$\dfrac{\begin{array}{cc}\vdots\,\pi_1 & \vdots\,\pi_2 \\ C_1, \boldsymbol{L}_1 & C_2, \overline{\boldsymbol{L}_2}\end{array}}{(C_1, C_2)\sigma}\,\sigma = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2) \tag{15}$$

and $\pi_1 \rightarrowtriangle \pi_1'$, where π_1' derives a clause $C_1', \boldsymbol{L}_1'$ that is more general than $C_1, \boldsymbol{L}_1$—say, $C_1'\theta = C_1$ and $\boldsymbol{L}_1'\theta = \boldsymbol{L}_1$—, and π' is:

$$\dfrac{\begin{array}{cc}\vdots\,\pi_1' & \vdots\,\pi_2 \\ C_1', \boldsymbol{L}_1' & C_2, \overline{\boldsymbol{L}_2}\end{array}}{(C_1', C_2)\sigma'}\,\sigma' = \mathrm{mgu}(\boldsymbol{L}_1' = \boldsymbol{L}_2) \tag{16}$$

Note indeed that σ unifies $\boldsymbol{L}_1 = \boldsymbol{L}_2$, hence the substitution σ'' mapping every variable x free in $\boldsymbol{L}_1'$ to $x\theta\sigma$ and every variable y free in $\boldsymbol{L}_2$ to $y\sigma$ unifies $\boldsymbol{L}_1' = \boldsymbol{L}_2$. So σ' exists and is more general than σ''. Since $(C_1', C_2)\sigma'' = (C_1, C_2)\sigma$, $(C_1', C_2)\sigma'$ is also more general than $(C_1, C_2)\sigma$.

It follows that the relation $\rightarrowtriangle$ is well-defined and rewrites derivations of clauses C into derivations of more general clauses. In particular, it rewrites refutations into refutations.

As in Section 3, let ς be a selection function, and call a first-order resolution step a ς-*step* if and only if all literals resolved upon ($\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ in (15) for example) are selected in their respective clauses.

The last difficulty that awaits us in adapting the arguments of Section 3 is that the resolution step of (15) might be a ς-step while that of (16) fails to be, or conversely. It turns out that converting a non-ς-step into a ς-step is benign, while the converse leads to non-termination. The former requires us to add the following rewrite rule to (7), (8):

$$f_1(u_1, u_2) \longrightarrow f_0(u_1, u_2) \tag{17}$$

Then the termination Lemma 1 extends smoothly:

Lemma 3. *The rewrite system (7), (8), (17) is terminating.*

Proof. As for Lemma 1. The only additional case is in claim (d), where in $\mathcal{C}[f_1(s,t)]$ the subterm $f_1(s,t)$ rewrites to $f_0(s,t)$, with $\mathcal{C} = f_0(u_1, f_2(u_1, \ldots f_0(u_n, []), \ldots))$, and $s, t \in RED, u_1, \ldots, u_n \in SN$. Then the contractum $\mathcal{C}[f_0(s,t)]$ is $\mathcal{C}'[t]$, where t is in RED and $\mathcal{C}' = f_0(u_1, f_2(u_1, \ldots f_0(u_n, f_0(s, [])), \ldots))$ is clearly an SN-context, so that $\mathcal{C}[f_0(s,t)] \in SN$. □

At this point, the technique of Section 3 applies almost without modification. We only have to replace condition (5) by the condition that, whenever the bottom resolution step in (11) is a ς-step and the top step is not, then the top resolution step in (12) is a ς-step; and similarly, that whenever the bottom resolution step in (13) is a ς-step and the top step is not, then the top two resolution steps in (14) are ς-steps. By extension, call this condition (5) again. Then:

Theorem 2. *Let the selection function ς be* stable*: for every literal L, clause C, and substitution θ, if $L\theta \in \varsigma(C\theta)$ then $L \in \varsigma(C)$. Assume that conditions (3), (4) and (5) hold. Then every resolution derivation of C from S can be effectively transformed into a ς-resolution derivation of some clause C' more general than C from S.*

Proof. If (15) is a ς-step, then $\boldsymbol{L}_1 \subseteq \varsigma(C_1, \boldsymbol{L}_1)$. Letting as above $C'_1\theta$ be C_1 and $\boldsymbol{L}'_1\theta$ be L_1, it obtains $\boldsymbol{L}'_1\theta \subseteq \varsigma(C'_1\theta, \boldsymbol{L}'_1\theta)$, so by stability $\boldsymbol{L}'_1 \subseteq \varsigma(C'_1, \boldsymbol{L}'_1)$. Therefore (16) is a ς-step too. Using the obvious adaptation of the $[\![\_]\!]$ interpretation, $\twoheadrightarrow$ terminates if the rewrite system (7), (8), (17) does. Then apply Lemma 3. Finally, observe that $\twoheadrightarrow$-normal forms consist only of ς-steps, using condition (4). This requires showing that the last resolution step in a refutation is a ς-step, and taking bottommost non-ς-steps, as in Section 3. And indeed the last resolution step must be a resolution between clauses $\boldsymbol{L}_1$ and $\overline{\boldsymbol{L}_2}$, with $\sigma = \mathrm{mgu}(\boldsymbol{L}_1 = \boldsymbol{L}_2)$; by condition (4), $\varsigma(\boldsymbol{L}_1\sigma) = \boldsymbol{L}_1\sigma$ since this is a unit clause, and similarly $\varsigma(\overline{\boldsymbol{L}_2}\sigma) = \overline{\boldsymbol{L}_2}\sigma$, so by stability $\varsigma(\boldsymbol{L}_1) = \boldsymbol{L}_1$ and $\varsigma(\boldsymbol{L}_2) = \boldsymbol{L}_2$, therefore the last resolution step is a ς-step. □

This can be used to extract ordered refutations from any resolution refutation:

Corollary 2. *Let $>$ be a stable ordering, i.e., $A > B$ implies $A\sigma > B\sigma$ for every substitution σ. Then every first-order resolution refutation can be effectively transformed into an ordered resolution refutation (wrt. $>$) from the same set of clauses.*

Proof. Apply a similar argument as in Section 4.1. Consider rule (11) $\rightarrow$ (12), and assume that (a) every literal in $\boldsymbol{L}_1$ is maximal in $C, \boldsymbol{L}_1$, and (b) every literal in $\boldsymbol{L}_2\sigma$ is maximal in $C'_1\sigma, C_2\sigma, \overline{\boldsymbol{L}_2}\sigma$. By stability, (b') every literal in $\boldsymbol{L}_2$ is maximal in $C'_1, C_2, \overline{\boldsymbol{L}_2}$. If the topmost resolution step in (12) is not ordered, then, because of (a), some literal in $\boldsymbol{L}_2$ is less than some literal in $C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1$. By (b') this literal L_{2i} in $\boldsymbol{L}_2$ must be less than some literal L'_{1j} in $\boldsymbol{L}'_1$: (c) $L_{2i} < L'_{1j}$.

We claim that every literal in $\boldsymbol{L}'_1$ is maximal in $C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1$. Otherwise there would be a literal L'_{1k} in $\boldsymbol{L}'_1$ that is less than some literal L'' in $C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1$. First, L'' cannot be in $\boldsymbol{L}'_1$: since σ instantiates all of $\boldsymbol{L}'_1$ to the same atom, if $L'_{1k} < L''$ then $L'_{1k}\sigma = L''\sigma$ by stability, contradicting $L'_{1k}\sigma = L''\sigma$. Second, L'' cannot be in C'_1: otherwise $L'_{1k} < L''$ implies $L'_{1k}\sigma < L''\sigma$ by stability; since $L'_{1k}\sigma = L'_{1j}\sigma > L_{2i}\sigma$ by (c) and stability, $L''\sigma > L_{2i}\sigma$, contradicting (b). Third, L'' cannot be in $\overline{L_2}$: otherwise $L''\sigma = \overline{L_2}\sigma = L_{2i}\sigma > L'_{1k}\sigma$ (by stability) $= \boldsymbol{L}'_1\sigma = L'_{1j}\sigma > L_{2i}\sigma$ (by (c) and stability) $= \overline{L_2}\sigma$, a contradiction. So every literal in $\boldsymbol{L}'_1$ is indeed maximal in $C'_1, \overline{\boldsymbol{L}_2}, \boldsymbol{L}'_1$, therefore the topmost resolution step in (12) is a ς-step.

The case of rule (13) $\rightarrow$ (14) is similar. Now since $\varsigma(C)$ defined as the set of maximal literals in C is stable in the sense of Lemma 3, the result follows. □

It is probably more interesting to rearrange resolution refutations into hyperresolution proofs instead. Recall that positive hyperresolution derivations can be seen as proofs deriving new facts from old facts [2]: arguably these proofs look like standard mathematical proofs, proceeding from assumptions to theorems. The following corollary can then be used to improve the readability of proofs obtained by any resolution theorem prover, provided it keeps a trace of the proof obtained, by extracting a positive hyperresolution proof from it.

Corollary 3. *Every first-order resolution refutation can be effectively transformed into a positive hyperresolution refutation from the same set of clauses.*

Proof. As in Section 4.2, this is the case where $C, \boldsymbol{L}_1$ is a positive clause in (11), (12), (13), (14). Condition (5) is then verified. Moreover, the corresponding selection function ς, which selects literals based on signs, is clearly stable. □

By the same remark as in Section 4.2, every first-order resolution derivation of a positive clause C (not just the empty clause) can be similarly effectively transformed into a positive hyperresolution derivation of some more general clause.

6 Conclusion

Our goal here is to demonstrate a simple effective transformation of resolution refutations, in the style of Gentzen's cut-elimination technique. With it, we are able to give constructive proofs of completeness of several important refinements of resolution, including ordered resolution, hyperresolution, semantic resolution and their semi-ordered variants. The technique is completely elementary, and although the termination result (Lemma 1) is more challenging, it still affords a simple proof from first principles.

Our point in this paper is not to compete in the race for ever more powerful arguments of completeness for refinements of resolution. Kowalski and Hayes' semantic trees and Bachmair-Ganzinger forcing are still to date the most powerful techniques for proving completeness. However our technique is simple and uniform enough, and perhaps more importantly, provides constructive completeness proofs from which one can extract effective transformations from one complete resolution format to another: see Section 4 for the propositional case, Corollary 2 for first-order ordered resolution, and Corollary 3 for first-order hyperresolution. This can prove useful for the purpose of explanation of proofs. There is still ample room for research here, notably in the first-order setting; we believe that our technique provides a valuable alternative to existing proof explanation paradigms (see [4] for some pioneering work). The termination criterion of Lemma 1, which only observes whether cuts are ς-cuts or non-ς-cuts, will probably have to be refined to this aim.

However, we believe that the main uses of this technique should be in one of the following domains. First, there are many logics which do not have a semantics as simple as Herbrand semantics, as required for standard semantic completeness proofs to go through [9, 1]; e.g., the technique presented here should pave the way to produce decision procedures for uniform provability in certain restricted classes of hereditary Harrop formulae; note that the backchaining rule used there is essentially a form of resolution already. Second, as usual with syntactic normalization proofs, this technique should have some meaning from a programming language point of view: if each clause in the initial set of clauses S is thought of as denoting the type of some primitive, then proof normalization is computation in a corresponding language, as in the Curry-Howard isomorphism for the λ-calculus. What this language of clauses, with normalization of resolution proofs as computation mechanism, really does is still to be discovered.

Acknowledgements

I thank Francis Klay, who expressed interest in this technique in 1999, as well as the students of the 1999 edition of my DEA course on automated deduction. They all encouraged me to publish it.

References

1. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.
2. C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science Classics. Academic Press, 1973.
3. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.
4. A. Felty and D. Miller. Proof explanation and revision. Technical Report MS-CIS-88-17, U. Pennsylvania, 1987.
5. C. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. *Resolution Decision Procedures*, chapter 25, pages 1791–1849. Volume II of Robinson and Voronkov [11], 2001.
6. J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.
7. J. Goubault-Larrecq and I. Mackie. *Proof Theory and Automated Deduction*, volume 6 of *Applied Logic Series*. Kluwer, May 1997.
8. W. H. Joyner Jr. Resolution strategies as decision procedures. *Journal of the ACM*, 23(3):398–417, July 1976.
9. R. Kowalski and P. J. Hayes. Semantic trees in automatic theorem-proving. *Machine Intelligence*, 4:87–101, 1969. Reprinted in [12].
10. J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, Jan. 1965.
11. J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*. North-Holland, 2001.
12. J. Siekmann and G. Wrightson, editors. *Automation of Reasoning: Classical Papers in Computational Logic*, volume 2, 1967–1970. Springer Verlag, 1983.
13. M. E. Szabo. *The Collected Papers of Gerhard Gentzen*. North-Holland Publishing Company, Amsterdam, 1969.