Completeness for Linear Regular Negation Normal Form Inference Systems ⋆

Reiner Hähnle[1], Neil V. Murray[2], Erik Rosenthal[3]

[1] Department of Computer Science, University of Karlsruhe, 76128 Karlsruhe, Germany, reiner@ira.uka.de, +49-721-608-4329
[2] Department of Computer Science, State University of New York, Albany, NY 12222, USA, nvm@cs.albany.edu, (518) 442-3393
[3] Department of Mathematics, University of New Haven, West Haven, CT 06516, brodsky@charger.newhaven.edu, (203) 932-7463

Abstract. Completeness proofs that generalize the Anderson-Bledsoe excess literal argument are developed for calculi other than resolution. A simple proof of the completeness of regular, connected tableaux for formulas in conjunctive normal form (CNF) is presented. These techniques also provide completeness results for some inference mechanisms that do not rely on clause form. In particular, the completeness of regular, connected tableaux for formulas in negation normal form (NNF), and the completeness of NC-resolution under a linear restriction, are established.

Keywords: *Logic for Artificial Intelligence, completeness, tableau method, resolution, non-clausal inference, negation normal form*

⋆ This research was supported in part by the National Science Foundation under grants CCR-9404338 and CCR-9504349.

1 Introduction

Robinson developed semantic tree arguments to provide completeness proofs for resolution and related inference systems; these arguments are not entirely transparent. The excess literal technique discovered by Anderson and Bledsoe [1], which is strictly syntactic, was a considerable simplification. Their technique, which is essentially an induction on the size of the formula, was the basis for the first completeness proofs of certain refinements of resolution. Their technique has been considered by other authors but has seldom been applied to non-resolution systems. (There have been some non-resolution applications: Baumgartner and Furbach [2], where a similar technique is applied to model elimination, and Letz [7], where such an induction is made implicitly. Bibel's connection graph resolution completeness result [3] uses a related technique.)

Reducing the size of the search space is an important consideration in most automated deduction systems. With resolution, this is often done with subsumption checking and with the linear restriction. Analogously, the search space for the tableau method can be reduced using regularity and connectivity. The first proof that the tableau method is complete with these restrictions is due to Letz [7] (proofs for closely related systems were presented previously in [6,8]; other restrictions have been investigated by Wallace and Wrightson [15]). While elegant and insightful, Letz' proof is not the easiest to follow. The proof presented here provides a slight generalization by disallowing extensions by unit clauses, while at the same time being considerably shorter and simpler. The slight generalization is not the focus of this paper; the proof technique, which may be of interest in other settings, and the results for negation normal form are.

Effective proof methods that do not rely on conjunctive normal form (CNF) are often desirable for AI applications since reliance on clause form can create an exponential blow-up even before inference procedures are applied. Efficient clause form translations commonly used in theorem provers preserve unsatisfiability but not logical equivalence. Thus, any system — for example, the diagnosis system reported in [12], based on Reiter's theory [13] — that depends on logical equivalence cannot use these translations. One difficulty with non-clausal proof methods is that relatively few refinements have been developed for restricting the search space. Among the many refinements for CNF based systems, *linear* restrictions are of considerable importance.

In this paper, the Anderson-Bledsoe technique is adapted to paradigms that employ negation normal form (NNF). Completeness is proved for NNF inference techniques under a kind of linearity restriction as well as with a regularity condition, while preserving the simplicity and elegance of their technique. In particular, the completeness of *connected tableaux* and *connected regular tableaux* for NNF formulas (defined below) and the completeness of linear non-clausal resolution are established. Several proofs are omitted for lack of space and can be found in [5].

In Section 2 the Anderson-Bledsoe excess literal technique for proving completeness of resolution is described; it is generalized and applied to prove the completeness of linear resolution. An alternative generalization is used to demonstrate the completeness of regular connected tableaux with CNF input. The completeness of non-clausal tableaux and of linear non-clausal resolution is discussed in Section 3.

2 Conjunctive Normal Form

Recall that a *clause* is a disjunction of literals, and that a formula in *conjunctive normal form* (CNF) is a conjunction of clauses. Such a conjunction is often referred to as a *set* of clauses. A *link* is a complementary pair of literals occurring in different clauses, and a literal occurrence is said to be *pure* if it is not linked to any literal in the clause set. It is easy to verify the *Pure Rule*:

Lemma 1. (**Pure Rule**) If a clause set with a pure literal is unsatisfiable, then so is the set of clauses produced by removing the clause containing the pure literal. □

2.1 Resolution

The work described here was inspired by the Anderson-Bledsoe [1] excess literal proof of the completeness of resolution, and we begin with that proof.

Theorem 2. (**Anderson-Bledsoe**) Binary resolution (with merging) is refutation complete for propositional logic.

Proof: Let $\mathcal{S} = \{C_1, C_2, \ldots, C_m\}$ be an unsatisfiable set of clauses. We must show that there is a refutation of $\mathcal{S}$ using resolution. Let n be the number of *excess literals* in $\mathcal{S}$, i.e., the number of literals in $\mathcal{S}$ minus the number of clauses in $\mathcal{S}$, and proceed by induction on n. If $n = 0$, then every clause is a unit clause. Since $\mathcal{S}$ is unsatisfiable, there must be two clauses consisting of complementary literals. Since they are unit clauses, their resolvent is the empty clause.

Suppose now that there is a resolution proof of any unsatisfiable clause set with at most n excess literals, and suppose that $\mathcal{S}$ has $n + 1$ excess literals. At least one clause, say C_1, contains at least two literals. Let C_1' be the result of removing one literal, say p, from C_1, and let $\mathcal{S}'$ be the result of replacing C_1 by C_1' in $\mathcal{S}$. Since any satisfying interpretation for $\mathcal{S}'$ would satisfy $\mathcal{S}$ (i.e., if C_1' is *true*, so is C_1), $\mathcal{S}'$ must be unsatisfiable. Since $\mathcal{S}'$ has n excess literals, there exists a resolution proof of $\mathcal{S}'$. That proof produces the empty clause from $\mathcal{S}'$. Thus, if it is applied to $\mathcal{S}$, it will either produce the empty clause or a clause containing only the literal p. That clause may contain several copies of p, but they can be merged to create the clause $\{p\}$.

We now have a set of clauses containing $\mathcal{S}'' = \{\{p\}, C_2, C_3, \ldots, C_m\}$, which is unsatisfiable since any satisfying interpretation would satisfy $\mathcal{S}$. Finally, $\mathcal{S}''$ has fewer excess literals than $\mathcal{S}$, so there is a proof of $\mathcal{S}''$; the two proofs together provide a proof of $\mathcal{S}$. □

An interesting variation of this proof can be obtained by applying the induction to the number of distinct atoms that appear in $\mathcal{S}$. This can be demonstrated by proving that resolution with the linear restriction is complete. Formally, a resolution proof is *linear* if one parent of each resolvent, except for the first, is the most recently derived clause. We require the following two lemmas. The first is [8, Lemma 2.3.2, p.63], and the second is a variant.

Lemma 3. Let $\mathcal{S}$ be a minimally unsatisfiable set of clauses, let B be a subset of the clause $C \in \mathcal{S}$, and let $\mathcal{S}' = (\mathcal{S} - \{C\}) \cup \{B\}$. Then $\mathcal{S}'$ contains a minimally unsatisfiable set of clauses that includes B (and does not include C). $\square$

Lemma 4. Let $\mathcal{S} = \{C_0, C_1, C_2, \ldots, C_k\}$ be a minimally unsatisfiable set of clauses, and suppose $C_0 = \{p\} \cup \{q_1, \ldots, q_n\}$, where $n \geq 0$. Obtain $\mathcal{S}'$ from $\mathcal{S}$ by deleting every occurrence of p in $\mathcal{S}$, and obtain $\mathcal{S}''$ by applying the Pure Rule to $\mathcal{S}'$, i.e., by deleting clauses containing $\overline{p}$. Let $C_0' = \{q_1, \ldots, q_n\}$. Then

1. $\mathcal{S}'$ is unsatisfiable;
2. $\mathcal{S}''$ is unsatisfiable;
3. C_0' is a member of any minimally unsatisfiable subset of $\mathcal{S}'$;
4. C_0' is a member of any minimally unsatisfiable subset of $\mathcal{S}''$. $\square$

Theorem 5. Linear binary resolution (with merging) is refutation complete for propositional logic.

Let $\mathcal{S} = \{C_1, C_2, \ldots, C_m\}$ be an unsatisfiable set of clauses. We assume that $\mathcal{S}$ is minimally unsatisfiable; otherwise, restrict attention to a minimally unsatisfiable subset. We must show that there is a refutation of $\mathcal{S}$ using linear resolution. We will prove the following slightly stronger result: There is a refutation of $\mathcal{S}$ in which any clause may be used as the top clause, i.e., the one used in the first step.

We proceed by induction on the number of distinct atoms in $\mathcal{S}$. If there are none, then $\mathcal{S}$ contains the empty clause, and we are done. Otherwise, suppose that all unsatisfiable sets of clauses with at most n atoms can be refuted with linear resolution, and assume that $\mathcal{S}$ has $n+1$ atoms including the atom p. If $\mathcal{S}$ contains the unit clause $\{p\}$, fine; otherwise, remove all occurrences of p from $\mathcal{S}$. This formula is unsatisfiable since any satisfying interpretation would also satisfy $\mathcal{S}$. Consider a minimally unsatisfiable subset; by Lemma 4, every clause that had contained p in $\mathcal{S}$ is in this set. Also, since no occurrence of $\overline{p}$ is linked, no clause containing $\overline{p}$ is present. By the induction hypothesis, there is a refutation $\mathcal{R}_p$ by linear resolution. Note that if that refutation is applied to $\mathcal{S}$ — call the resulting refutation $\mathcal{R}_p'$ — none of the clauses containing $\overline{p}$ are resolved upon, and that the result is either the empty clause[1] or the clause $\{p\}$; (merging several copies of p may be required). This clause is of course the last resolvent.

Analogously, if we begin by deleting $\overline{p}$, a proof $\mathcal{R}_{\overline{p}}$ can be found that, when applied to $\mathcal{S}$ — let the resulting proof be $\mathcal{R}_{\overline{p}}'$ — will produce the empty clause or

[1] In fact, this cannot happen because of minimality, but this is not really relevant.

the clause $\{\overline{p}\}$. If either $\mathcal{R}'$ or $\mathcal{R}'_{\overline{p}}$ did produce the empty clause, then we are done. Otherwise, we may assume that the latter proof began with a clause C containing $\overline{p}$. The two proofs are linear, and we can put them together, maintaining linearity, by resolving $\{p\}$ and C. This proof may still produce the unit clause $\{\overline{p}\}$ because of multiple occurrences of $\overline{p}$. However, by resolving with the unit $\{p\}$, linearity is maintained and the empty clause is produced. □

The above induction is somewhat reminiscent of the Davis-Putnam procedure [4]. Refutations are obtained from the induction hypothesis by removing all occurrences of a given atom. Completeness for connected CNF tableaux can be proved with this technique. Below the size-based induction of Theorem 2 is adapted to obtain Letz' result [7] (with a slight strengthening) that completeness also holds when a regularity restriction is imposed along with connectivity. In subsequent sections, these induction techniques are employed to prove completeness for various non-clausal systems.

2.2 Analytic Tableaux

Definition. A *tableau proof tree* for a set (conjunction) $\mathcal{S}$ of clauses is a tree labeled with formulas, constructed as follows:

1. The tree consisting of the single node $\mathcal{S}$ is a tableau; this tree is the *initial* tableau.
2. Suppose T is a tableau containing a node on the branch Θ labeled $\mathcal{C}$, the conjunction of $C_1, C_2, \ldots, C_n$, (each of which is a clause). Then a tableau is obtained by replacing the node labeled $\mathcal{C}$ with n new nodes labeled $C_1, C_2, \ldots, C_n$ on the branch Θ. This is the *alpha rule*; it is applied automatically to nodes labeled with conjunctions.
3. Suppose T is a tableau containing a node labeled C, a clause containing the literals $l_1, l_2, \ldots, l_n$. Then a tableau is obtained by extending any branch Θ below C by appending n new leaves to Θ labeled $\{l_1\}, \{l_2\}, \ldots, \{l_n\}$. This is the *beta rule*; it is sometimes referred to as a *beta extension*.

A tableau is *closed* if each branch contains a pair of nodes labeled with complementary literals. In this case we speak of a *proof* or a *refutation* of $\mathcal{S}$.

Observe that, for a set of clauses, the alpha rule is applied exactly once: to the initial tableau. Many authors choose to omit alpha rules, thus simplifying the definition of a tableau proof tree. We employ the alpha rule because it makes CNF tableaux and the definitions in the next paragraph special cases of NNF tableaux. Also, there is a slight technical advantage: All unit clauses are placed (by the first and only alpha step) on the initial branch and therefore need never be used for extensions.

A tableau proof tree with CNF input is *weakly connected* if each time a beta rule extends a branch, the formula to which the rule has been applied is linked to a node along the branch. (This property is present in the fully condensed proof trees of [11], but its presence is not sufficient for a proof tree to be fully condensed.) The tableau is *connected* if immediately prior to the extension the node to which the selected formula is linked appears after the last branch point. Note that after the first beta extension of a non-unit clause, this link must be to a leaf. Insisting that tableaux be connected is similar to the linear restriction for resolution.

A tableau proof tree with CNF input is *regular* if no branch contains distinct nodes labeled with identical *literals*. Observe that even a clause set with two identical clauses can have a regular tableau proof tree. This is the only way in which a regular (thus alpha) step can introduce repeated formulas on a branch when the input is in CNF. Such repeated formulas can be introduced much more easily with NNF input. In fact, repetitions along a branch due to alpha rules cannot be avoided in NNF. However, obvious ones such as repeated clauses can: If a conjunction (or disjunction, for that matter) has identical arguments, all but one can simply be deleted. Henceforward we assume all formulas have been so condensed.

It is interesting to note that regularity excludes extensions by unit clauses. This is no problem. Indeed, this is desirable: No closure is ever enabled by such an extension since all such unit clauses label nodes on every branch in the tree. This effect is produced by the first (and only) alpha step. The formal inclusion of the alpha rule also makes possible connected proofs that are free of (unnecessary!) unit extensions. As with Theorem 5, it is essential that we work with a *minimally* unsatisfiable set of clauses.

The next theorem demonstrates that Letz' result can be obtained in a straightforward manner by adapting the induction argument from Theorem 2. Theorem 6 slightly enhances Letz' in that extensions with unit clauses are disallowed.

Theorem 6. The tableau method restricted to regular connected tableaux, free of unit extensions, is complete for unsatisfiable (finite) sets of (ground) clauses.

Proof: Let $\mathcal{S}$ be an unsatisfiable set of clauses. We assume that $\mathcal{S}$ is minimal; otherwise, restrict attention to a minimal subset. We will prove the following slightly stronger result: Given any (non-unit) clause in $\mathcal{S}$, there is a closed tableau for $\mathcal{S}$ in which that clause is the first to which a beta rule is applied. We proceed by induction on the number n of literal occurrences in $\mathcal{S}$.

If $n = 0$, the result is trivial. The case $n = 1$ is not possible by minimality. If $n = 2$, there must be two unit clauses containing complementary literals, no beta rule is necessary, and again the result is trivial. Observe, again because of minimality, that this is the only case in which $\mathcal{S}$ contains only unit clauses.

Assume now that there is a closed regular connected tableau for every unsatisfiable formula with at most n literal occurrences, and let $\mathcal{S} = \{B_1, B_2, \ldots, B_m\}$ be a minimally unsatisfiable clause set with $n+1$ literal occurrences ($n \geq 2$). Suppose that $B_1 = \{l_1, l_2, \ldots, l_k\}$ is the (non-unit) clause in $\mathcal{S}$ selected for the first extension step; let T be the resulting tableau — see Figure 1a.

Observe that T is both regular and connected: Were T irregular, some branch would have duplicate literals, which is to say, for some $i, 1 \leq i \leq k$, $\mathcal{S}$ contains the unit clause $\{l_i\}$. But that clause would subsume B_1, contrary to minimality.

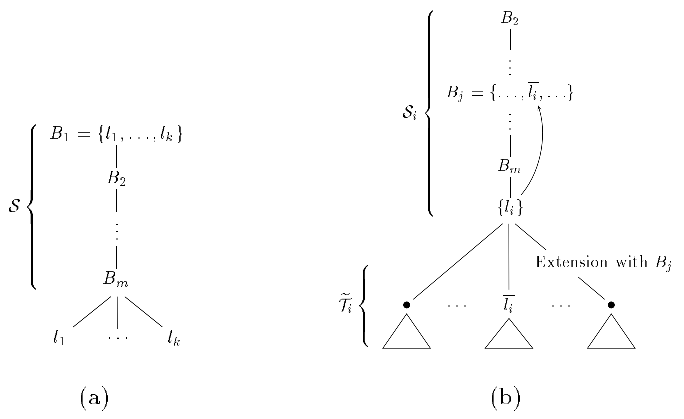

Fig. 1. Tableau T and forest $\tilde{T}$ from the proof.

That T is connected follows for the same reason: Were B_1 not linked, it would be unnecessary.

For each i, let $\mathcal{S}_i = (\mathcal{S} - \{B_1\}) \cup \{\{l_i\}\} = \{\{l_i\}, B_2, \ldots, B_m\}$. By Lemma 3, any minimally unsatisfiable subset of $\mathcal{S}_i$ contains $\{l_i\}$. The induction hypothesis applies to each $\mathcal{S}_i$, so there is a regular connected tableau T_i for each $\mathcal{S}_i$. In each T_i, beta rules are applied only to non-unit clauses in a minimally unsatisfiable subset. Furthermore, l_i is linked to a clause B_j in that subset. If B_j is not a unit, the induction hypothesis allows us to assume that the first application of a beta rule in T_i is to B_j. Let $\tilde{T}_i$ be the forest obtained from T_i by deleting the nodes labeled with the clauses from $\mathcal{S}_i$ — see Figure 1b. If B_j is a unit, i.e., if $B_j = \{\overline{l_i}\}$, then T_i closes with $\{l_i\}$, and $\tilde{T}_i$ is empty.

We expand T by, for every i, adjoining $\tilde{T}_i$ to the branch whose leaf is labeled $\{l_i\}$; i.e., the roots of the forest $\tilde{T}_i$ become the children of the leaf $\{l_i\}$. Observe that, if $\tilde{T}_i$ is empty, then l_i is linked to the unit clause $\{\overline{l_i}\}$; so the branch whose leaf is $\{l_i\}$ is closed in T. Thus, since every T_i is closed, T is a closed tableau for $\mathcal{S}$. Since there are no unit extensions in T, the proof will be complete if we show that T is regular and connected.

T is regular since the initial extension of B_j is regular and each T_i is regular. Finally, T is connected because the extension of B_j is connected and each l_i either closes its branch or is linked to the non-unit of the first extension in T_i.

□

3 Non-Clausal Methods

The techniques illustrated in the previous section can be used in non-clausal settings. The proofs are somewhat involved and are omitted here for lack of space; see [5].

We begin by recalling that a formula is in *negation normal form* (NNF) if the only connectives are conjunction, disjunction, and negation, and if all negations are at the atomic level. The tableau method works with NNF formulas as well as with formulas in clause form. In the NNF context, the phrase *set of formulas* means the conjunction of the formulas in the set. In particular, we assume the formulas in the set to be either disjunctions or literals.

We now define NNF tableaux. Since only atoms are negated, and since conjunction and disjunction are the only other logical operators, the rules for constructing an NNF tableau are the obvious generalization of the alpha and beta rules for CNF tableaux and are quite simple. The alpha rule, which applies to conjunctions, does not increase the number of branches and is applied automatically; the beta rule, which applies to disjunctions, does increase the number of branches and should not be applied automatically.

Definition. A *tableau proof tree* for an NNF formula $\mathcal{S}$ is a tree labeled with formulas and constructed as follows:

1. The tree consisting of a single node labeled $\mathcal{S}$ is a tableau; this tree is the *initial* tableau.
2. If T is a tableau, and if N is a node in T labeled with $\mathcal{S} = \wedge_{i=1}^{n} \mathcal{S}_i$, where each $\mathcal{S}_i$ is a disjunction or a literal, then N is replaced by n new nodes on the same branch labeled $\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_n$. This is the *alpha rule* and is applied automatically to any node labeled with a conjunction.
3. If T is a tableau, and if N is a node in T labeled with $\mathcal{S} = \vee_{i=1}^{n} \mathcal{S}_i$, where each $\mathcal{S}_i$ is a conjunction or a literal, then a tableau may be obtained by appending n new nodes below any branch Θ containing N; each new node is uniquely labeled with some $\mathcal{S}_i$, $1 \leq i \leq n$. This is the *beta rule* and we say that Θ has been *beta-extended* by $\mathcal{S}$.

A tableau is *closed* if each branch contains a node labeled *false* or contains a pair of nodes labeled with complementary literals. The next theorem provides completeness for the tableau method with NNF formulas; the proof uses the Davis-Putnam style induction.

Theorem 7. The tableau method is a complete refutation procedure for unsatisfiable (finite) sets of (ground) NNF formulas. □

A tableau proof tree is *weakly connected* if each time a beta rule extends a branch, the formula to which the rule has been applied is linked to a node along the branch. The tableau is *connected* if immediately prior to the extension the node to which the selected formula is linked appears after the last branch point. This amounts to saying that the link is to a leaf or to a node created by an alpha rule applied to a leaf.

The next theorem strengthens Theorem 7 with a connectedness restriction; the proof is a reasonably straightforward adaptation of the proof of Theorem 7.

Theorem 8. Connected tableaux are a complete inference procedure for finite sets of ground NNF formulas. □

There are two problems with Theorem 8: No notion of regularity is enforced, and unit extensions are not excluded. The following example shows that unit extensions cannot be excluded if we demand a connected tableau. Let $\mathcal{S}$ be the conjunction of the two unit clauses $\{\overline{p}\}$ and $\{\overline{q}\}$ and the formula $B = (l \wedge (p \vee q)) \vee p$. After the first alpha rule application, the only possible non-unit extension is with B. To complete the proof tree without unit extensions, $(p \vee q)$ must be extended, and connectivity cannot be maintained.

Observe that both p and q are linked to unit clauses in $\mathcal{S}$. Therefore, violating connectivity (at least in this example) would seem to be both necessary and harmless. We shall see that this situation is more than fortuitous.

We say that the beta extension of a node labeled $\mathcal{F}$ is *u-connected* if every link in $\mathcal{F}$ is to a node labeled with a literal. A *tableau is u-connected* if every beta extension is either connected or u-connected. Note that u-connectivity for tableaux is a bit weaker than connectivity but stronger than weak connectivity.

In order to generalize CNF regularity to the NNF case in a useful way, both semantic and proof theoretic issues must be considered. Insufficient space precludes a detailed discussion, but the definition below is a generalization of the definition of regularity for CNF tableaux. Since nodes labeled by conjunctions are automatically replaced by the alpha rule, the nodes of an NNF tableau are always labeled by either literals or disjunctions.

Suppose we have a node labeled $\mathcal{F} = (\mathcal{F}_1 \vee \mathcal{F}_2 \vee ... \vee \mathcal{F}_n)$ on a branch Θ in an NNF tableau. Suppose further that we beta-extend Θ to create n new branches $\Theta_1, ..., \Theta_n$, where $\Theta_i = \Theta \cup \mathcal{F}_i$. Note that if $\mathcal{F}_i$ is a literal, then it labels the single new node on Θ_i; otherwise, $\mathcal{F}_i = \mathcal{F}_{i1} \wedge \mathcal{F}_{i2} \wedge ... \wedge \mathcal{F}_{iq}$, the alpha rule applies, and each $\mathcal{F}_{ij}$ labels a new node on Θ_i. We say that this beta extension is *irregular* if for some Θ_i, the new nodes on Θ_i are a subset of the nodes on Θ; i.e., the extension is irregular if for some $\mathcal{F}_i$, every node produced by the application of the alpha rule to $\mathcal{F}_i$ is labeled with a formula identical to that labeling some node on Θ. The step is *regular* if it is not irregular. Observe that all alpha steps are regular. A tableau proof tree is regular if every extension is regular.

Theorem 9. The tableau method restricted to regular u-connected tableaux free of unit extensions is a complete refutation procedure for unsatisfiable (finite) sets of (ground) NNF formulas. □

We now consider NC-resolution and begin by providing a precise definition (in the ground case). Let $\mathcal{F}$ and $\mathcal{G}$ be arbitrary unnormalized ground formulas, where $\mathcal{H}$ occurs as a subformula of both $\mathcal{F}$ and $\mathcal{G}$. If $\beta = true$ or $\beta = false$, we denote by $\mathcal{F}[\beta/\mathcal{H}]$ the result of replacing all occurrences of $\mathcal{H}$ in $\mathcal{F}$ by β and of performing truth-functional simplifications. Then the formula

$$\mathcal{F}[true/\mathcal{H}] \vee \mathcal{G}[false/\mathcal{H}]$$

is an NC-resolvent of $\mathcal{F}$ and $\mathcal{G}$ on the subformula $\mathcal{H}$. For the remainder of this paper we will consider only the case where $\mathcal{H}$ is a literal. Note that NC-resolution is defined for completely unnormalized formulas and does not require NNF. Here we restrict attention to NNF, so that the results of Section 3 can be employed.

The definition of linear NC-resolution is completely analogous to that of ordinary linear resolution. Formally, an NC-resolution proof is *linear* if one parent of each resolvent, except for the first, is the most recently derived formula.

Theorem 10. Linear non-clausal resolution is a complete inference procedure for finite sets of ground NNF formulas. □

References

1. Anderson, R., and Bledsoe, W., A linear format for resolution with merging and a new technique for establishing completeness, *J. ACM* 17(3) (1970), 525 – 534.
2. Baumgartner, P. and Furbach, U. Model elimination without contrapositives. *Proc. 12th Conference on Automated Deduction CADE, Nancy/France.* In *Lecture Notes in Artificial Intelligence*, (Bundy, Ed.), Springer-Verlag, Vol. 814, 87-101.
3. Bibel, W., On matrices with connections, *J. ACM* 28 (1981), 633 – 645.
4. Davis, M. and Putnam, H. A computing procedure for quantification theory. *J.ACM* **7**, 1960, 201-215.
5. Hähnle, R., Murray, N.V. and Rosenthal, E. On proving completeness. Tech. Rep. TR 96-6, Dept. of Comp. Sci., SUNY Albany, NY, December, 1996.
6. Kowalski, R. and Kuehner D., Linear Resolution with Selection Function. *Artificial Intelligence*, **2**(3), (1971), 227–260.
7. Letz, R., *First-order calculi and proof procedures for automated deduction*, Ph.D. dissertation, TU Darmstadt.
8. Loveland, D.W., *Automated Theorem Proving: A Logical Basis*, North-Holland, New York (1978).
9. Murray, N.V., and Rosenthal, E., Inference with path resolution and semantic graphs. *J. ACM* 34,2 (1987), 225–254.
10. Murray, N.V., and Rosenthal, E. Dissolution: Making paths vanish. *J.ACM* 40,3 (July 1993), 504-535.
11. Oppacher, F. and Suen, E. HARP: A tableau-based theorem prover, *Journal of Automated Reasoning* 4 (1988), 69–100.
12. Ramesh, A. and Murray, N.V. An application of non-clausal deduction in diagnosis. *Proceedings* of the *Eighth International Symposium on Artificial Intelligence*, Monterrey, Mexico, October 17-20, 1995, 378–385.
13. Reiter, R., A theory of diagnosis from first principles. *A.I. Journal*, **32** (1987), 57-95.
14. Smullyan, R.M., *First-Order Logic* (2^{nd} Edition), Dover Press, 1995.
15. Wallace, K. and Wrightson, G. Regressive merging in model elimination tableau-based theorem provers, *Journal of the Interest Group in Pure and Applied Logics*, (Special Issue: Selected Papers from Tableaux'94), **3** (Oct. 1995), 921-938.