Completeness for Linear Regular Negation Normal Form Inference Systems *

Technical Report 97-2

April, 1997

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) is established, and the completeness of NC-resolution for NNF formulas under a linear restriction is proved.

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 were 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 [16]). 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. Proofs of Lemmas 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. Some basic properties of NNF formulas are reviewed in Section 3. In Section 4, non-clausal tableaux are defined and shown to be complete. This result is extended to connected, regular NNF tableaux. In Section 5 the completeness eof linear non-clausal resolution is proved.

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). □

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 $\mathcal{S}''$. □

Proof: Every clause in $\mathcal{S}'$ subsumes a clause in $\mathcal{S}$, so $\mathcal{S}'$ is unsatisfiable, and $\mathcal{S}''$ is unsatisfiable by the Pure Rule.

Consider an interpretation I_0 that falsifies C_0 but satisfies all other clauses in $\mathcal{S}$. There must be such an I_0, since otherwise C_0 would not be in a minimally unsatisfiable clause set. In particular, $I_0(p) = false$ and $I_0(C_0') = false$. Thus, in $\mathcal{S}$, I_0 satisfies a literal other than p in every clause other than C_0. This implies that I_0 satisfies every clause in $\mathcal{S}'$ other than C_0'. But this means that C_0' is in any minimally unsatisfiable subset of $\mathcal{S}'$.

Since C_0' is the only clause in $\mathcal{S}'$ falsified by I_0, and since $\mathcal{S}'' \subset \mathcal{S}'$, C_0' must be in the falsified $\mathcal{S}''$.

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[4] 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 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

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

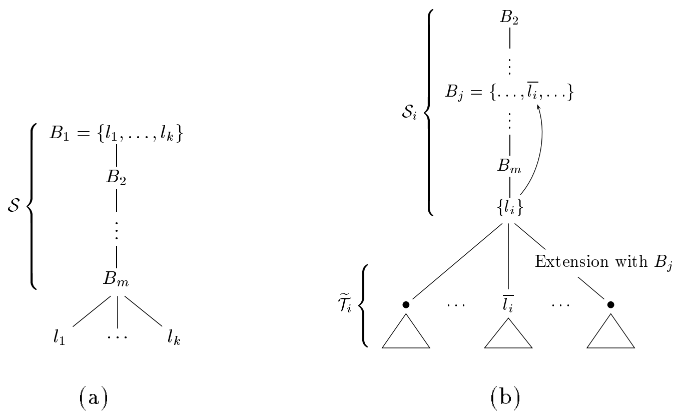

Fig. 1. Tableau T and forest $\tilde{T}$ from the proof.

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. 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 Negation Normal Form

We now consider formulas in *negation normal form* (NNF): The only connectives are conjunction, disjunction, and negation, and all negations are at the atomic level. *Semantic graphs* are a graphical representation of NNF formulas that will be useful for the analysis that follows.

Definition. A *semantic graph* is

1. one of the constants *true* or *false*,
2. an atom p or $\overline{p}$,
3. a *c-arc*, which is a conjunction of two semantic graphs, or
4. a *d-arc*, which is a disjunction of two semantic graphs.

We use the notation $(X,Y)_c$ for the c-arc from X to Y and similarly use $(X,Y)_d$ for a d-arc; the subscript may be omitted when no confusion is possible. Each semantic graph used in the construction of a semantic graph $\mathcal{G}$ is called an *explicit subgraph* of $\mathcal{G}$. If $\mathcal{G} = (X,Y)_c$, then X (respectively, Y) is a *fundamental subgraph* of $\mathcal{G}$ if X (respectively, Y) is not a c-arc; otherwise the fundamental subgraphs of X (respectively, Y) are fundamental subgraphs of $\mathcal{G}$. Similarly if $\mathcal{G} = (X,Y)_d$, then X (respectively, Y) is a *fundamental subgraph* of $\mathcal{G}$ if X (respectively, Y) is not a d-arc; otherwise the fundamental subgraphs of X (respectively, Y) are fundamental subgraphs of $\mathcal{G}$.

If we view conjunction (and disjunction) as n-ary commutative and associative operators, then the fundamental subgraphs of an explicit subgraph are its arguments. For example, the fundamental subgraphs of the upper disjunction in Figure 2 are $(\overline{r} \wedge p)$ and the literals s and t. A *full block* is a subset of the fundamentals of one explicit subgraph. Full blocks may be treated as essentially explicit subgraphs (up to the order of arguments), and the Isomorphism Theorem (see [9]) assures us that they are the only structures that may be so treated.

In essence, the only difference between a semantic graph and a formula is the point of view, and we will use either term depending upon the desired emphasis. For a more detailed exposition, see [10]. Unless otherwise stated, we will assume that for any formula, the following simplification rules have been applied; in them, p is a literal and $\mathcal{S}$ is an arbitrary formula.

1. $p \vee \cdots \vee p \;=\; p \wedge \cdots \wedge p \;=\; p$
2. $p \vee \neg p \;=\; true$; $\qquad p \wedge \neg p \;=\; false$
3. $\mathcal{S} \vee false \;=\; \mathcal{S} \wedge true \;=\; \mathcal{S}$
4. $\mathcal{S} \wedge false \;=\; false$; $\qquad \mathcal{S} \vee true \;=\; true$

The set of *nodes* of a semantic graph G consists of all literal occurrences used in its construction; the same holds for the *set of c-arcs* of G and the *set of d-arcs* of G; i.e., these sets include the nodes, c-arcs, and d-carc, respectively, occurring in the explicit subgraphs of G.

In the following, we identify a semantic graph G with the formula it represents[5]. We also identify a semantic graph with the triple consisting of its set of nodes, its set of c-arcs, and its set of d-arcs. The only exceptions, for which this latter identification is ambiguous, are the semantic graphs *true* and *false* — both correspond to $(\emptyset,\emptyset,\emptyset)$.

If p and q are nodes in a graph $\mathcal{G}$, and if $(X,Y)_\alpha$ is an arc ($\alpha = c$ or $\alpha = d$) with p in X and q in Y, we say that $(X,Y)_\alpha$ is the arc *connecting* p and q, and that p and q are *α-connected.* A *link* is a pair of c-connected complementary nodes. A *partial c-path through* $\mathcal{G}$ is a set of nodes such that any two are c-connected, and a *c-path*

[5] *true*, *false*, and positive literals represent themselves; a negative literal $\overline{A}$ represents $\neg A$; $(X,Y)_c$ represents $X \wedge Y$; and $(X,Y)_d$ represents $X \vee Y$

through $\mathcal{G}$ is a partial c-path that is not properly contained in any partial c-path. (*Partial*) *d-paths* are defined accordingly using d-arcs instead of c-arcs. The c-paths of a formula correspond to the clauses of one of its disjunctive normal form (DNF) equivalents. Similarly, the d-paths correspond to the clauses of a CNF equivalent. The next lemma (from [9]) is easy to prove.

Lemma 7. A formula is satisfiable if and only if some c-path in it is satisfiable, and a c-path is satisfiable if and only if it does not contain a link. □

For example, consider the semantic graph depicted in Figure 2. The node r is c-connected to each of q, p, $\overline{r}$, s, and t, and is d-connected to $\overline{p}$. The c-paths of the graph in Figure 2 are: $\{\overline{r}, p, \overline{p}\}$, $\{\overline{r}, p, q, r\}$, $\{s, \overline{p}\}$, $\{s, q, r\}$, $\{t, \overline{p}\}$, $\{t, q, r\}$.

$$((\overline{r} \wedge p) \vee s \vee t) \wedge (\overline{p} \vee (q \wedge r)) \qquad \equiv \qquad \begin{array}{llll} \overline{r} & & & \\ \wedge \vee s \vee t & & & \\ p & & & \\ & \wedge & & \\ & & q & \\ \overline{p} \vee \wedge & & & \\ & & r & \end{array}$$

Fig. 2. NNF Displayed in Two Dimensions

We will find it useful to consider subgraphs that are not explicit; that is, given any set of nodes, we would like to examine that part of the graph consisting of exactly that set. In Figure 3, the example of Figure 2 is shown on the left; the subgraph relative to the set $\{\mathrm{p}, \mathrm{s}, \overline{\mathrm{p}}\}$ is on the right.

$$\begin{array}{llll} \overline{\mathrm{r}} & & & \\ \wedge \vee \mathrm{s} \vee \mathrm{t} & & & \\ \mathrm{p} & & & \\ & \wedge & & \\ & & \mathrm{q} & \\ \overline{\mathrm{p}} \vee \wedge & & & \\ & & \mathrm{r} & \end{array} \qquad\qquad \begin{array}{c} \mathrm{p} \vee \mathrm{s} \\ \wedge \\ \overline{\mathrm{p}} \end{array}$$

Fig. 3.

Given a semantic graph G and a non-empty subset N of the nodes of G, the *subgraph* of G that corresponds to N is that part of G that consists of nodes from N, where the logical structure of that part is preserved.

A non-empty subset N of nodes corresponds unambiguously to one subgraph of G. For a precise definition of subgraph, see [9].

While relaxing the restriction to clause form employed in the previous section, we can allow negations at any level; implications also pose no difficulties. Formulas of this nature can be converted to NNF in linear time. Alternatively, we could introduce additional tableau rules to account for implications and arbitrary negations. To simplify the discussion, we assume that formulas are in NNF. We will consider a set $\mathcal{S}$ of NNF formulas to be the conjunction of its members. Thus, complementary literals residing in different formulas are in fact c-connected and thus constitute a link. Links may also reside within a single formula.

The completeness argument in Theorem 11 requires the removal of syntactic objects from formulas in an unsatisfiability preserving way. This is more complicated for NNF formulas than it is for clause form, where all that is necessary is the removal of literal occurrences from the clauses in which they appear. One way to remove an object $\mathcal{P}$ from an unsatisfiable formula $\mathcal{S}$ to produce an unsatisfiable formula $\mathcal{S}_{\mathcal{P}}$ is to remove all c-paths containing $\mathcal{P}$. Since $\mathcal{S}$ is unsatisfiable, all c-paths of $\mathcal{S}$ contain a link. Since the c-paths of $\mathcal{S}_{\mathcal{P}}$ will be a subset of the c-paths of $\mathcal{S}$, $\mathcal{S}_{\mathcal{P}}$ will also be unsatisfiable. It turns out that a good choice for $\mathcal{S}_{\mathcal{P}}$ is the *c-path complement* (see [10]) of $\mathcal{P}$ in $\mathcal{S}$, denoted $CC(\mathcal{P},\ \mathcal{S})$.

Definition. Let $\mathcal{H}$ be a subgraph of the semantic graph $\mathcal{G}$. Then the *c-path complement* of $\mathcal{H}$ with respect to $\mathcal{G}$, denoted $CC(\mathcal{H},\mathcal{G})$, is the subgraph of $\mathcal{G}$ consisting of all literals in $\mathcal{G}$ that lie on c-paths that contain no nodes from $\mathcal{H}$. If no such literal exists, $CC(\mathcal{H},\mathcal{G}) = \text{false}$.

In [10] it is shown that the c-path complement can also be characterized by the following:

Let $\mathcal{G} = (X,Y)_\alpha$ be a semantic graph, and let $\mathcal{H}$ be a subgraph of $\mathcal{G}$. Then

$$CC(\mathcal{H},\mathcal{G}) = \begin{cases} \mathcal{G} & \text{if } \mathcal{H} \text{ does not meet } \mathcal{G} \\ \text{false} & \text{if } \mathcal{H} = \mathcal{G} \\ (CC(\mathcal{H}_X, X), CC(\mathcal{H}_Y, Y))_\alpha & \text{otherwise} \end{cases}$$

It turns out (see Theorem 8 below) that when the first argument of the c-path complement operator is a full block $\mathcal{P}$, then $CC(\mathcal{P},\ \mathcal{S}) = \mathcal{S} - CE(\mathcal{P})$, where CE($\mathcal{P}$) is the *c-extension* — defined below — of $\mathcal{P}$, and $\mathcal{S} - CE(\mathcal{P})$ is the subgraph of $\mathcal{S}$ consisting only of nodes not in $CE(\mathcal{P})$. (See [10] for a precise definition of subgraph.) In other words, we merely remove the c-extension of $\mathcal{P}$ from $\mathcal{S}$ to produce $\mathcal{S}_\mathcal{P}$. In this paper, the objects we remove from formulas will always be full blocks. To define c-extension (and d-extension), let $\mathcal{P}$ be a full block in a semantic graph $\mathcal{G}$; then $\mathcal{P}$ is a conjunction or a disjunction of fundamental subgraphs of some explicit subgraph $\mathcal{H}$. If the final arc of $\mathcal{H}$ is a conjunction, then the *c-extension* of $\mathcal{P}$ is $\mathcal{H}$, and the *d-extension* of $\mathcal{P}$ is $\mathcal{P}$ itself. The definitions are reversed if the final arc of $\mathcal{H}$ is a d-arc. We use the $CE(\mathcal{P})$ and $DE(\mathcal{P})$ to denote the c- and d-extensions, respectively, of $\mathcal{P}$.

Theorem 8. Let $\mathcal{P}$ be a full block in a formula $\mathcal{S}$. Then $CC(\mathcal{P},\ \mathcal{S}) = \mathcal{S} - CE(\mathcal{P})$. □

Proof: We proceed by induction on the size of $\mathcal{S}$. If $\mathcal{S}$ is a literal, then $\mathcal{P} = \mathcal{S}$ or $\mathcal{P} = \emptyset$; in either case, the result is trivial.

If $\mathcal{S}$ is larger than a single literal, then the final arc of $\mathcal{S}$ may be a c-arc or a d-arc; suppose first that $\mathcal{S} = (X,Y)_c$. If $\mathcal{P}$ is a subset of the fundamentals of $\mathcal{S}$, then the third condition in the definition of CC amounts to $CC(\mathcal{P},\ \mathcal{S}) = CC(\mathcal{P},\ \mathcal{P})$, in which case $CC(\mathcal{P},\ \mathcal{S}) = \emptyset$ (false). Also, $CE(\mathcal{P}) = \mathcal{S}$, and so $\mathcal{S} - \mathcal{P} = \emptyset$.

Otherwise, since $\mathcal{P}$ is a full block, $\mathcal{P}$ must be a subset of the fundamentals of an explicit proper subgraph $\mathcal{H}$ of $\mathcal{S}$. Thus, $\mathcal{H}$ must be contained in X or in Y; say in X. Then the induction hypothesis applies to X. Since $CC(\mathcal{P},\ \mathcal{S}) = Y \wedge CC(\mathcal{P},\ X)$, and since $CE(\mathcal{P})$ is contained entirely within X, the theorem holds for $\mathcal{P}$ in X, and hence for $\mathcal{P}$ in $\mathcal{S}$.

Suppose now that $\mathcal{S} = (X,Y)_d$. If $\mathcal{P}$ is a subset of the fundamentals of $\mathcal{S}$, then, from the definition of CC, $CC(\mathcal{P},\ \mathcal{S})$ is the disjunction of those fundamentals of $\mathcal{S}$ that are not in $\mathcal{P}$. From the definition of CE, $CE(\mathcal{P}) = \mathcal{P}$, and $\mathcal{S} - CE(\mathcal{P})$ is also the disjunction of those fundamentals of $\mathcal{S}$ that are not in $\mathcal{P}$.

If $\mathcal{P}$ is not a subset of the fundamentals of $\mathcal{S}$, then again $\mathcal{P}$ is a subset of X or of Y — say of X. From the definition of CC, $CC(\mathcal{P},\ \mathcal{S}) = Y \vee CC(\mathcal{P},\ X)$, which completes the proof since the induction hypothesis ensures that the theorem holds in X.

The Pure Rule is valid for NNF formulas [10] but requires some care. In essence, the NNF equivalent of the clause containing a given literal p is $DE(p)$.

Lemma 9. **(Pure Rule for Negation Normal Form)** If p is a pure literal in an unsatisfiable NNF formula $\mathcal{S}$, then $\mathcal{S} - DE(p)$ is unsatisfiable. □

The tableau method works with NNF formulas as well as with formulas in clause form. In this 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. As in the clausal case, an unsatisfiable set of NNF formulas is *minimally unsatisfiable* if removing any member produces a satisfiable set.

The following lemma is useful for the proof of Theorems 11 and 12 below. The former establishes completeness for unrestricted NNF tableaux. The latter shows that NNF tableaux with a connectivity restriction are complete; it is a generalization of Lemma 4 to NNF formulas.

Lemma 10. Let $\mathcal{S} = \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$ be a minimally unsatisfiable set of NNF formulas, and let p be a literal in $\mathcal{F}_0$. Construct $\mathcal{S}'$ from $\mathcal{S}$ by removing every occurrence of CE(p) in $\mathcal{S}$; construct $\mathcal{S}''$ by applying the Pure Rule to $\mathcal{S}'$, i.e., by removing every occurrence of DE($\overline{p}$). Let $\mathcal{F}_i'$ and $\mathcal{F}_i''$ be the formulas in $\mathcal{S}'$ and $\mathcal{S}''$, respectively, corresponding to $\mathcal{F}_i$ in $\mathcal{S}$ in the obvious way.

1. Both $\mathcal{S}'$ and $\mathcal{S}''$ are unsatisfiable;
2. $\mathcal{F}_0'$ is a member of any minimally unsatisfiable subset of $\mathcal{S}'$;
3. $\mathcal{F}_0''$ is a member of any minimally unsatisfiable subset of $\mathcal{S}''$. □

Proof: Let I_0 be an interpretation that falsifies F_0 but satisfies all other formulas in $\mathcal{S}$. (There must be such an I_0, since otherwise $\mathcal{F}_0$ would not be in a minimal unsatisfiable formula set.) Let $D_0 = \{p_0, p_1, ..., p_n\}, n \geq 0$, be a d-path in $\mathcal{S}$ in $\mathcal{F}_0$ that is falsified by I_0.

Parts 1 and 2: Every d-path of $\mathcal{F}_i'$ subsumes a d-path of $\mathcal{F}_i, 0 \leq i \leq n$, so $\mathcal{S}'$ is obviously unsatisfiable, and $\mathcal{S}''$ is unsatisfiable by Lemma 9.

To prove part 3, observe that $\mathcal{F}_0'$ in $\mathcal{S}'$ contains the d-path $D_0' = D_0 - \{p\} \subseteq \{p_0, ..., p_n\}$. Obviously, $I_0(p) = I_0(D_0') = \mathit{false}$, so $I_0(\mathcal{F}_0') = \mathit{false}$. Let C_i denote the c-path of $\mathcal{F}_i$ satisfied by I_0, $1 \leq i \leq n$. Since $I_0(C_i) = \mathit{true}$, C_i cannot contain p; furthermore, $\mathcal{F}_i'$ contains C_i because by Theorem 8, $\mathcal{F}_i' = \mathcal{F}_i - \mathrm{CE}(p) = \mathrm{CC}(p, \mathcal{F}_i)$, which by definition has all c-paths of $\mathcal{F}_i$ that do not include p. Thus I_0 satisfies $\mathcal{F}_i'$, $1 \leq i \leq n$. As a result, $\mathcal{F}_0'$ is in any minimal unsatisfiable subset of $\mathcal{S}'$.

Finally, we must establish part 4 of the lemma: D_0' cannot contain $\overline{p}$, and $\mathcal{F}_0'' = \mathcal{F}_0' - \mathrm{DE}(\overline{p}) = \mathrm{DC}(\overline{p}, \mathcal{F}_0')$ by the dual of Theorem 8. Thus $\mathcal{F}_0''$ contains all d-paths not containing $\overline{p}$, in particular D_0'. Thus I_0 falsifies $\mathcal{F}_0''$.

Now consider the formulas $\mathcal{F}_i'', 1 \leq i \leq n$, comprising $\mathcal{S}'' - \{\mathcal{F}_0''\}$. Each $\mathcal{F}_i''$ is formed by removing DE($\overline{p}$) from $\mathcal{F}_i'$. As a result, each c-path in $\mathcal{F}_i''$ is a (not necessarily proper) subset of a c-path in $\mathcal{F}_i'$. In particular, each c-path C_i' in $\mathcal{F}_i'$, is satisfied by I_0, and thus each corresponding $C_i'' \subseteq C_i'$ is also. Therefore, I_0 satisfies each $\mathcal{F}_i''$. So I_0 falsifies $\mathcal{F}_0''$, satisfies all formulas of $\mathcal{S}'' - \{\mathcal{F}_0''\}$, and $\mathcal{F}_0''$ must be in any minimal unsatisfiable subset of $\mathcal{S}''$.

4 NNF Tableaux

We begin this section with a definition of 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 11. The tableau method is a complete refutation procedure for unsatisfiable (finite) sets of (ground) NNF formulas.

Proof: Let $\mathcal{S}$ be an unsatisfiable set of ground NNF formulas; we proceed by induction on the number n of distinct atoms in $\mathcal{S}$.

If $n = 0$, then $\mathcal{S} = \mathit{false}$, and there is nothing to prove; so assume the theorem holds for all formulas with at most n distinct atoms, and let $\mathcal{S}$ be a set of formulas with $n + 1$ distinct atoms.

Suppose first that one of the formulas in $\mathcal{S}$ is the unit p. After the alpha rule is applied to the initial tableau, the result is a tableau in which the one branch is open and contains a node labeled p.

Otherwise, let $\mathcal{S}_p$ be the set of formulas produced by removing from $\mathcal{S}$ all occurrences of $CE(p)$ and then applying the Pure Rule (Lemma 9) to the resulting unlinked occurrences of $\overline{p}$, thus removing the d-extensions of all occurrences of $\overline{p}$. By Lemma 10 this preserves unsatisfiability. By the induction hypothesis, there is a proof T_p for $\mathcal{S}_p$. Let T'_p be the tableau tree produced by applying each extension in T_p to the corresponding formulas in $\mathcal{S}$. If T'_p is closed, we are done. If not, then all open branches in T'_p must result from formulas containing p or $\overline{p}$.

A formula that contains p is a formula of $\mathcal{S}$ from which the c-extension of p was removed to produce the corresponding formula of $\mathcal{S}_p$. That c-extension must occur in a disjunction since a conjunction would be part of the c-extension. When such a disjunction is used for extension, an open branch emerges — one that is simply not in T_p. The open branch will contain $CE(p)$, which is either the unit p or a conjunction that automatically produces the unit p on the branch by the alpha rule.

Consider now extensions by formulas containing $\overline{p}$. Recall that in the completeness proof for linear resolution for formulas in clause form (Theorem 5), the clauses containing $\overline{p}$ were never used in the proof provided by the induction hypothesis. Such clauses are in effect removed by the Pure Rule since $DE(\overline{p})$ is precisely the clause containing $\overline{p}$. In the non-clausal case, $DE(\overline{p})$ is also deleted, but this in general is a *proper* sub-formula of the formula containing $\overline{p}$. In that case, $DE(\overline{p})$ is always an argument of a conjunction. Let $DE_c(\overline{p})$ be the conjunction, and let $DE'_c(\overline{p})$ be $DE_c(\overline{p})$ with $DE(\overline{p})$ removed. Thus, $DE_c(\overline{p}) = DE'_c(\overline{p}) \wedge DE(\overline{p})$. If $DE'_c(\overline{p})$ is used in an extension in T_p, then, when $DE_c(\overline{p})$ is extended in T'_p, $DE(\overline{p})$ is conjoined to $DE'_c(\overline{p})$. In particular, there are no new branches and therefore no open branches due to the formulas of $\mathcal{S}$ that contain $\overline{p}$. Hence, all open branches in T'_p contain nodes labeled p.

Similarly, by deleting occurrences of $CE(\overline{p})$ to produce the set of formulas $\mathcal{S}_{\overline{p}}$, the induction hypothesis provides a proof $T_{\overline{p}}$ of $\mathcal{S}_{\overline{p}}$ and a corresponding proof tree $T'_{\overline{p}}$ of $\mathcal{S}$. The leaves of all open branches of $T'_{\overline{p}}$ are labeled $\overline{p}$. Finally, to obtain a closed tableau from $T'_{\overline{p}}$ and T'_p, we append $T'_{\overline{p}}$ to all open branches of T'_p and observe that any branch not closed by $T'_{\overline{p}}$ or by T'_p has nodes labeled p and labeled $\overline{p}$ and thus is closed. □

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 11 with a connectedness restriction; the proof is a reasonably straightforward adaptation of the proof of Theorem 11.

Theorem 12. Connected tableaux are a complete inference procedure for finite sets of ground NNF formulas.

Proof: Let $\mathcal{S}$ be a minimally unsatisfiable set of NNF formulas; (otherwise, restrict attention to a minimally unsatisfiable subset). We shall prove the following slightly stronger result: Given any formula $\mathcal{F}$ in $\mathcal{S}$, there is a connected closed tableau for $\mathcal{S}$ in which $\mathcal{F}$ is the first formula used for a beta extension. We proceed by induction on the number n of distinct atoms in $\mathcal{S}$, noting that the result is trivial for $n = 0$ or $n = 1$.

Assume now that the theorem holds for any formula with at most n distinct atoms, and assume that $\mathcal{S}$ has $n + 1$; let p be a literal in $\mathcal{F}$. By Lemma 10, we may remove CE(p) and DE($\overline{p}$) from $\mathcal{S}$ to produce $\mathcal{S}_p$, a minimally unsatisfiable set of formulas (with one fewer atom) containing the formula $\mathcal{F}_p$ corresponding to $\mathcal{F}$. The induction hypothesis provides a closed non-clausal connected tableau T_p for $\mathcal{S}_p$, in which $\mathcal{F}_p$ is used for the first extension. By applying the extensions of T_p to $\mathcal{S}$, a connected tableau T'_p is produced in which the first formula extended is $\mathcal{F}$, and all open branches contain a node labeled p. Each such open branch has p occurring *after* the last branch point, because these branches do not exist in T_p and thus are never extended in T'_p.

Similarly, we may obtain $T'_{\overline{p}}$, a connection tableau in which the first formula extended is one containing $\overline{p}$, and all open branches contain nodes labeled $\overline{p}$. By appending $T'_{\overline{p}}$ to all open branches of T'_p, a closed connected NNF tableau is produced. □

It is worth noting that the apparent asymmetry between the removal of p and $\overline{p}$ is in fact part of a larger symmetry. When removing p, we remove the c-extensions of its occurrences, while it is the d-extensions that are removed for $\overline{p}$. But it is easy to see that removing the c-extension of a literal is equivalent to replacing the literal by *false* and then applying the simplification rules. Similarly, removing the d-extension of a literal is equivalent to replacing the literal by *true* and simplifying.

4.1 Regularity and u-connectivity

There are two problems with Theorem 12: 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. It is tempting to characterize extension by a formula ψ as irregular when it is entailed by the *other* formulas on that branch. However, there is an immediate problem with such an approach: The initial branch of an NNF tableau is unsatisfiable (in cases where a proof can indeed be found). Thus, as the proof tree grows, any formula ψ labeling a node below the initial branch is entailed by some subset of the formulas above it. Indeed, each such ψ is *individually* unnecessary in some sense; we can avoid extensions with it by restricting attention to a minimally unsatisfiable subset of formulas on that branch not containing ψ. By doing so, however, we may merely embark on a new derivation of ψ.

The point is, we may restrict attention to a minimally unsatisfiable subset not including ψ, provided that the proof proceeds from that subset under no further restrictions. However, if we continuously remove semantically entailed formulas, we may forever postpone the completion of a proof. One way to ensure that this doesn't happen is to verify that any proof obtainable after extension with ψ could have been constructed with no additional (and perhaps less) effort prior to extension with ψ.

An analogous situation occurred in the early study of subsumed clauses. Removal of clause B when it is subsumed by C from a set containing both is obviously unsatisfiability preserving. This was assumed to be essentially completeness preserving. Sibert [14] was the first to point out that this semantic observation, combined with completeness of resolution, guaranteed only that a refutation was available at any point, *provided that only resolution was employed from that point onward.* These issues were subsequently investigated in great detail — see [8].

The definition of regularity below encompasses the notion of proof theoretic redundancy and accounts for automatic applications of the alpha rule. 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.

In the proof of Theorem 14, we require yet another variant of [8, Lemma 2.3.2, p.63].

Lemma 13. Let $\mathcal{S} = \{\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_n\}$ be a minimally unsatisfiable set of NNF formulas, where the $\mathcal{F}_i$'s are either literals or disjunctions. Suppose $\mathcal{F}_1 = D_1 \vee ... \vee D_r, r \geq 1$. Let B be the disjunction of a subset of the D_i's, and let $\mathcal{S}' = (\mathcal{S} - \{\mathcal{F}_1\}) \cup \{B\}$. Then every minimally unsatisfiable subset of $\mathcal{S}'$ includes B. □

Proof: The formula $\mathcal{F}_1$ is a consequence of B; in particular B is *false* whenever $\mathcal{F}_1$ is. So there is an interpretation that falsifies B and satisfies $\mathcal{F}_i, 2 \leq i \leq n$. Thus B is in any minimally unsatisfiable subset of $\mathcal{S}'$.

Observe that in the lemma we treat B as an element of $\mathcal{S}'$. However, if B is the disjunction of a singleton subset, say $B = D_i$, and if D_i is not a literal, then B is a conjunction $D_{i1} \wedge ... \wedge D_{is_i}$. In this case, any minimally unsatisfiable subset of $(\mathcal{S} - \{\mathcal{F}_1\}) \cup \{D_{i1}\} \cup ... \cup \{D_{is_i}\}$ contains at least one D_{ij}.

Theorem 14. 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.

Proof: Let $\mathcal{S}$ be an unsatisfiable set of NNF formulas. 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) formula in $\mathcal{S}$, there is a closed regular u-connected tableau for $\mathcal{S}$ in which that formula 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 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 literals.

Assume now there is a regular u-connected tableau for all unsatisfiable NNF formulas with at most n literal occurrences, and let $\mathcal{S} = \{\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_m\}$ be a minimally unsatisfiable set of formulas (where each $\mathcal{F}_i$ is a disjunction or a literal) with $n+1$ literal occurrences ($n \geq 2$). Suppose $\mathcal{F}_1 = (D_1 \vee D_2 \vee ... \vee D_k)$ is the (non-unit) formula in $\mathcal{S}$ selected for the first extension; let T be the resulting tableau. Unlike the CNF case, the D_i's are not in general literals but can be conjunctions; in that case, the automatic application of the alpha rule is required. Suppose $D_i = D_{i1} \wedge D_{i2} \wedge ... \wedge D_{iq_i}, q_i \geq 1$. Observe that if $q_i = 1$, then D_i is a literal.

For T to be irregular, some D_i (a disjunct of $\mathcal{F}_1$) would be contained in $\mathcal{S}$ itself. Thus $\mathcal{S} - \{\mathcal{F}_1\}$ would contain a minimally unsatisfiable set by Lemma 13, contrary to the minimality of $\mathcal{S}$. Connectivity follows for the same reason: Were $\mathcal{F}_1$ not linked to $\mathcal{S}$, either it or the rest of $\mathcal{S}$ would be unnecessary.

For each i, let $\mathcal{S}_i = (\mathcal{S} - \{\mathcal{F}_1\}) \cup \{D_i\} = \{D_{i1}, D_{i2}, ..., D_{iq_i}, \mathcal{F}_2, ..., \mathcal{F}_m\}$. Please note: When D_i is a conjunction, its fundamentals are fundamentals of $\mathcal{S}_i$. By Lemma 13, any minimally unsatisfiable subset of $\mathcal{S}_i$ contains some $\{D_{ij}\}$. The induction hypothesis applies to each $\mathcal{S}_i$, so there is a regular u-connected tableau T_i for each $\mathcal{S}_i$, and we may choose any non-unit formula — a disjunction, designated later in the proof — on the initial branch for the first extension.

Extend T by combining the tableaux for the $\mathcal{S}_i$'s. Regularity is preserved since the test for regularity along each branch of T is identical to the corresponding test in one of the T_i's. However, connectivity is a different matter.

If one of the D_{ij}'s is a linked literal, the branch is closed, and there is nothing more to prove. Otherwise, the key to ensuring connectivity in T is proper selection of the formula for the first extension in T_i. If one of the D_{ij}'s (in the minimally unsatisfiable subset) is linked to any non-unit formula on the initial branch in T_i, let that non-unit be the first extended. In that case, connectivity is preserved in T. If not, then every D_{ij} is linked only to literals, in which case extending with one such D_{ij} is connected in T_i and u-connected in T. Thus T is u-connected. □

Observation. The last case considered in the proof — every D_{ij} in a minimally unsatisfiable subset is linked only to literals — is an interesting special case: The branch can be closed with no additional extensions. The reason is being linked only to literals guarantees that every c-path through D_{ij} has a link. Thus, one D_{ij} so linked is enough to close the branch.

5 Linear Non-Clausal Resolution

We begin by providing a precise definition of NC-resolution (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 15. Linear non-clausal resolution is a complete inference procedure for finite sets of ground NNF formulas.

Proof: Let $\mathcal{S} = \{\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_m\}$ be an unsatisfiable set of NNF formulas. 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 NC-resolution. We will prove the following slightly stronger result: There is a linear refutation of $\mathcal{S}$ in which any formula may be used as the top formula, i.e., one used in the first step.

If $n = 0$, then $\mathcal{S} = \{false\}$, and there is nothing to prove; so assume the theorem holds for all formulas with at most n distinct atoms, and let $\mathcal{S}$ be a set of formulas with $n+1$ distinct atoms; let p be a literal that occurs in $\mathcal{F}_0$. If $\mathcal{S}$ contains the unit $\{p\}$, fine. Otherwise, remove from $\mathcal{S}$ all occurrences of $CE(p)$; furthermore, since all occurrences of $\overline{p}$ are unlinked, the Pure Rule applies, and the d-extensions of all occurrences of $\overline{p}$ may be removed, preserving unsatisfiability (Lemma 10). Let $\mathcal{S}^p$ be the resulting set of formulas, and let $\mathcal{F}_0^p$ be the formula in $\mathcal{S}^p$ corresponding to $\mathcal{F}_0$.

Consider a minimally unsatisfiable subset of $\mathcal{S}^p$; by Lemma 10, $\mathcal{F}_0^p$ is in this set. By the induction hypothesis, there is a refutation $\mathcal{R}^p$ by linear NC-resolution that begins with $\mathcal{F}_0^p$. (Note that $\mathcal{F}_0^p$ could be a conjunction; in that case, we simply restrict attention to those conjuncts — there is at least one — that are in the minimally unsatisfiable subset of $\mathcal{S}^p$.)

Were we dealing with clauses, we could argue that if $\mathcal{R}^p$ is applied to $\mathcal{S}$, either false or the literal p will be derived. However, in the NNF case, this is not necessarily so. The reason is that a given formula $\mathcal{F}_i$ of $\mathcal{S}$ may contain several occurrences of p and of $\overline{p}$; the c-extension of each occurrence of p, and the d-extension of each occurrence of $\overline{p}$, is missing from $\mathcal{F}_i^p$. We must consider the effect of $\mathcal{R}^p$ on these structures when applying $\mathcal{R}^p$ to $\mathcal{S}$.

Consider first the c-extensions of p. The steps of $\mathcal{R}^p$ can change $\mathcal{F}_i$ outside $CE(p)$ and can change $CE(p)$ itself. Changes outside $CE(p)$ either leave it unchanged or delete it in its entirety due to a simplification rule. Changes inside $CE(p)$ can also delete it entirely. The other possibility is that what remains is a formula of the form $p \wedge \psi$. The point is, if $CE(p)$ is not completely deleted, then a formula of the form $p \wedge \psi$ remains.

In contrast, the d-extensions of $\overline{p}$ are of course disjunctions. When $\overline{p}$ occurs in $\mathcal{F}_j$, $DE(\overline{p})$ is either $\mathcal{F}_j$ itself or a proper subformula of $\mathcal{F}_j$. In the former case, $\mathcal{F}_j$ would have been removed in forming $\mathcal{S}^p$ and not have participated in $\mathcal{R}^p$. Otherwise, $DE(\overline{p})$ is one conjunct in some conjunction of $\mathcal{F}_j$. This conjunction is eliminated in $\mathcal{R}^p$ and will still be eliminated when $\mathcal{R}^p$ is applied to $\mathcal{S}$, unless it also contains p; in that case the conjunction is simply one of the c-extensions of p that remain as discussed above.

In any case, the result of applying $\mathcal{R}^p$ to $\mathcal{S}$ is $false$, in which case we are done, or a formula $\mathcal{C}^p$ made up of formulas of the form $p \wedge \psi$. Observe that $\mathcal{C}^p[false/p]$ will reduce to $false$ via the simplification rules since every $\mathcal{C}^p \wedge \psi$ has this property.

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 $false$ or a formula $\mathcal{C}^{\overline{p}}$. Note that $\mathcal{C}^{\overline{p}}$ is a formula built entirely from conjunctions that have $\overline{p}$ as a conjunct. Obviously, $\mathcal{C}^{\overline{p}}[false/\overline{p}]$ will reduce to $false$ via the simplification rules.

If either $\mathcal{R}'$ or $\mathcal{R}'^{\overline{p}}$ did produce $false$, then we are done. Otherwise, the two proofs must be put together in a linearity_preserving manner. By the induction hypothesis, we may assume that the latter proof began with a formula $\mathcal{F}_j^{\overline{p}}$ containing $\overline{p}$. The two proofs are linear, and we can combine them into one proof by NC-resolving $\mathcal{C}^p$ and $\mathcal{F}_j^{\overline{p}}$ on the literal p. The NC-resolvent is

$$\mathcal{C}^p[false/p] \quad \vee \quad \mathcal{F}_j^{\overline{p}}[true/p] \quad = \quad false \quad \vee \quad \mathcal{F}_j^{\overline{p}}[true/p] \quad = \quad \mathcal{F}_j^{\overline{p}}[true/p] \quad .$$

This formula is the top formula of $\mathcal{R}'^{\overline{p}}$ (when applied to $\mathcal{S}$) but with the occurrences of $CE(\overline{p})$ removed. As a result, the combined proof will produce a formula $\mathcal{C}^{\overline{p}'}$, which, though not identical to $\mathcal{C}^{\overline{p}}$, consists entirely of conjunctions that have $\overline{p}$ as a conjunct. If we add to this proof a final NC-resolution step with $\mathcal{C}^p$, the result is

$$\mathcal{C}^p[false/p] \quad \vee \quad \mathcal{C}^{\overline{p}'}[true/p] \quad = \quad false \vee false \quad = \quad false.$$

□

We note that the correspondence between a formula $\mathcal{F}_i$ in $\mathcal{S}$ and its counterpart $\mathcal{F}_i^p$ in $\mathcal{S}^p$ is crucial for the proof above. This correspondence is straightforward in all but one case: When $\mathcal{F}_i = D_1 \vee D_2$, $CE(p) = D_1$, and D_2 is a conjunction, then $\mathcal{F}_i^p$ is a conjunction and its conjuncts are separate formulas in $\mathcal{R}^p$. (Actually, $\mathcal{F}_i^p = D_2$ unless p occurs also in D_2.) This turns out to be a minor technicality; whenever a step in $calr_p$ involves any conjunct of $\mathcal{F}_i^p$, this is applied to the corresponding conjunct of D_2 in $\mathcal{F}_i$.

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. Sibert, E.E., A machine-oriented logic incorporating the equality relation, *Machine Intelligence 4* (Meltzer and Michie, eds.), Edinburgh University Press, Edinburgh, 1969, 103-133.
15. Smullyan, R.M., *First-Order Logic* (2^{nd} Edition), Dover Press, 1995.
16. 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.

This article was processed using the LaTeX macro package with LLNCS style