# On the Completeness of FLH-Resolution

Marshall R. Mayberry, III
Department of Computer Science
University of Texas at Austin

**Abstract**

First-literal hyper-resolution (FLH) is a refinement of ordinary hyper-resolution in that positive clauses are only resolved upon their *first* literals. In this respect, positive literals are ordered in every clause, but negative literals are not. Due to the ordering of literals in this resolution method, standard techniques for demonstrating ground completeness have proven ineffective. In this paper we prove both the ground completeness and the general completeness of first-literal hyper-resolution for first-order logic (FOL) using a *clause-ordering and modeling* procedure. This technique also suggests a new proof procedure, ETOS-FLH-resolution, a refinement of FLH-resolution, which is more restrictive in that clauses are ordered in a tree with the consequent advantage that only one resolvent is generated at each resolution step. This proof also has an important implication in the completeness of a version of the interactive prover IMPLY.

## 1 Introduction

First-literal hyper-resolution (FLH) is a refinement on hyper-resolution [7], requiring that all positive literals be ordered in every clause in which they occur. There is no pre-defined order; consequently, an arbitrary order might be dictated by implementation issues. Accordingly, literals in one input clause might be ordered differently than literals in another. The technique derives its name from the requirement that resolution only proceed on the *first* (i.e. leftmost in an ordered clause) literals of positive clauses. An obvious advantage of this is that search is thereby restricted to these literals because only positive clauses are generated as resolvents; the initial set of negative clauses is never changed. To maintain the ordering through the resolution process, the further device of merging literals right

is employed. It is this ordering and merging right of literals which has made the completeness of this method difficult to prove by traditional techniques, such as those given in [1, 6]. The method of proof used in this paper employs a *clause-ordering and modeling* procedure, which not only provides a technique for showing the completeness of first-literal hyper-resolution, but also suggests a refinement of this resolution method. This technique may prove to be even more powerful by its device of ordering positive clauses in a tree and thereby specifying a particular resolvent at each resolution step, while retaining the advantage of confining the search space during the proof process.

First-Literal hyper-resolution is closely related to the interactive prover, IMPLY, of Bledsoe, et al [2]. IMPLY can also act as a stand-alone prover. The question of whether the stand-alone mode of IMPLY is *complete* for FOL was first raised by Bledsoe several years ago. He determined that IMPLY itself is equivalent to FLH, if the theorem being proved is first "clausified" (put into disjunctive-normal form after it is skolemized), and if iterative deepening, variable renaming, and factoring are employed. So this provides an added motivation for showing that FLH is complete for FOL, because it will then follow that the above version of IMPLY is also complete. The ground completeness of first-literal hyper-resolution has been an open question for about ten years.

This technique also bears some similarities to Locking [3] and Indexing [5] in that one can impose an arbitrary order on the literals of a clause by means of indices. However, it differs from these in two important respects: (1) no literal ever appears more than once in a clause, and (2) the order of the positive literals in any nucleus are preserved upon resolution.

A primary objective of this paper is the proof of the following theorem, from which we are then able to show, with the appropriate lifting lemma, the general completeness of FLH for first-order logic.

**Theorem 1. (Ground Completeness of FLH-resolution).** If $S$ is a finite unsatisfiable set of ordered ground clauses, then $\Box \in \text{res-flh}(S, m)$ for some $m$.

The proof of this theorem is given in Section 3. We outline the proof as follows:

1. Since $S$ is a finite set of ground clauses, it follows that $\text{res-flh}(S, m+1) = \text{res-flh}(S, m)$ for some $m$. Let $S' = \text{res-flh}(S, m)$. Thus, if FLH is *complete*, then $\Box \in S'$.
2. If $\Box \notin S'$, then there is a flh-resolvent $R'$ of $S'$ which is not in $S'$ and is not subsumed by any member of $S'$ (i.e., $\text{res-flh}(S, m+1) \neq \text{res-flh}(S, m)$). Contradiction.
3. Such an $R$ is generated as follows:

(a) Since $S'$ is unsatisfiable and $\Box \notin S'$, we are able to describe an interpretation $\mathcal{I}$ of $S'$ which satisfies all the electrons in $S'$ and falsifies one of the nuclei $N$ (non-negative clause) in $S'$.

(b) In order to define the interpretation $\mathcal{I}$, we first *ETOS-order* the electrons of $S'$. This procedure is described in full in Section 2.

(c) We can then use this interpretation $\mathcal{I}$ to show that $N$ can be flh-resolved against a specific set of electrons to obtain an $R'$ that is not subsumed by any electron in $S'$, but which cannot be placed in $S'$ consistently according to $\mathcal{I}$.

In Section 2 we give definitions and lemmas which will be used in Section 3 to prove Theorem 1. In Section 4 we give some definitions for the predicate case from Section 2 and finish the completeness proof of *general* flh-resolution by proving and using the appropriate lifting lemma. In Sections 5 we define ETOS-FLH-Resolution for the propositional case. Section 6 describes future research toward generalizing ETOS-FLH-Resolution to the predicate level. In Section 7 we make concluding remarks.

# 2 Definitions and Lemmas

In this paper, we employ the convention of juxtaposition to indicate the order of literals in a clause. When this is not clear, or if emphasis is desired, we will use the concatenation operator '|' to indicate the ordering. Thus, to show that the literal $a$ is followed by a sequence of literals $\beta$, which is in turn followed by literal $c$ in a clause, we can write $a|\beta|c$, or, simply, $a\ \beta\ c$. It should be understood that each literal is connected by the disjunction '$\vee$,' as in ordinary resolution. Moreover, small Roman letters are used to indicate specific literals while small Greek letters are used to represent unspecified literal sequences. The subscripted capital letter $L$ represents a single unspecified literal.

**Definition.** Let $C = C'\ C''$ be a sequence of literals $L_n\ L_{n-1} \ldots L_2\ L_1$. Then $C''$ is a *suffix* of $C$. That is, $C'' = L_i \ldots L_1$ for some $0 \leq i \leq n$[1].

**Definition.** An *electron* is a *positive* ground clause. This is a ground clause in which every literal is positive.

**Definition.** A *nucleus* is a *non-positive* ground clause. This is a ground clause in which at least one literal is negative.

[1] If $C = C'$, then $C'' = \epsilon = \Box$ is the empty clause, which is a suffix of every clause. Also, every clause is a suffix of itself.

We require that no literal appear more than once in any given clause. This leads us to the following definitions:

**Definition.** The *sequence difference* of two sequences of literals $C_1$ and $C_2$, denoted by $C_1 \rightsquigarrow C_2$, is analogous to the notion of set difference, with the added stipulation that order is preserved.

**Example.** Let $C_1 = c\ e\ f\ e\ b$ and $C_2 = g\ b\ a\ e\ d$. Then

$$C_1 \rightsquigarrow C_2 = (c\ e\ f\ e\ b) \rightsquigarrow (g\ b\ a\ e\ d) = c\ f.$$

**Definition.** If $C$ is a sequence of literals, $L_1\ L_2 \ldots L_n$, then

$$\text{merge-right}(L_1\ L_2 \ldots L_n)$$

is defined recursively as

$$\begin{aligned} \text{merge-right}(L) &= L \\ \text{merge-right}(L_1\ L_2 \ldots L_n) &= [\text{merge-right}(L_1\ L_2 \ldots L_{n-1}) \rightsquigarrow L_n] \mid L_n \end{aligned}$$

**Example.**

$$\text{merge-right}(c\ e\ f\ e\ b\ g\ b\ a\ e\ d) = c\ f\ g\ b\ a\ e\ d.$$

**Definition.** Let

$$N = N'\ \sim L_0\ \sim L_1 \ldots \sim L_{n_N}$$

be a nucleus in $S$ and $\mathcal{E}_N$ be the electron set (depending on $N$) composed of the electrons[2]

$$\begin{aligned} E_0 &= L_0\ E_0' \\ E_1 &= L_1\ E_1' \\ &\vdots \\ E_{n_N} &= L_{n_N}\ E_{n_N}'. \end{aligned}$$

Then

$$R = \text{flh-resolvent}(N, \mathcal{E}_N) = \text{merge-right}(N'\,[E_0', E_1', \ldots, E_{n_N}'])$$

[2]We use $E_i$ and $E_i'$ to stand for literal sequences while $L_i$ is the first literal of each $E_i$.

is a *flh-resolvent* of clauses $E_0, \ldots, E_{n_N}$ and $N$. Note the brackets "[ ]" signify that the enclosed arguments may be permuted in any order. Thus, for any $n$ electrons, there may be at most $n!$ flh-resolvents. That there is more than one possible flh-resolvent, unlike ordinary resolution, is a consequence of the ordering of literals within a clause. This permutation is required for completeness.[3] (However, if ETOS-ordering is used (described below), it will be shown that a single flh-resolvent can be specified and completeness preserved. This is the basis for ETOS-FLH-resolution, which is described in Section 5.)

**Example.** Let

$$\begin{aligned} E_0 &= d\ b\ f \\ E_1 &= a\ g \\ E_2 &= h\ b\ g \end{aligned}$$

be electrons and

$$N = g\ c\ \sim a\ \sim d\ \sim h$$

be a nucleus. Then the following are possible flh-resolvents of $E_0, E_1, E_2$, and $N$:

$$\begin{aligned} R &= c\ f\ b\ g \\ \text{or } R &= c\ g\ b\ f \\ \text{or } R &= c\ b\ f\ g \end{aligned}$$

Note that it is possible for two or more permutations to yield the same flh-resolvent.

Initially, all clauses are merged-right (i.e., the positive literals are strictly ordered and no positive literal appears more than once in the clause).

**Definition.** flh-resolvents$(S)$ = $\{R : R =$ flh-resolvent$(N, \mathcal{E}_N)$ for some nucleus $N$ and electron set $\mathcal{E}_\mathcal{N} = \{E_0, E_1, \ldots, E_{n_N}\}$ in $S\}$.

**Definition.** If $C_1$ and $C_2$ are two ordered ground clauses, then $C_1$ *flh-subsumes* $C_2$ if $C_2 = C\ C_1$ for some literal sequence $C$ (i.e. if $C_1$ is a suffix of $C_2$).

**Definition.** res-flh$(S, m)$ is defined recursively as follows:

1. res-flh$(S, 0) = S$.

[3] Consider the set of clauses $\{\{a\ b\}, \{b\ c\}, \{c\ a\}, \{\sim a\ \sim b\}, \{\sim b\ \sim c\}, \{\sim c\ \sim a\}\}$

2. res-flh$'(S, i+1)$ = res-flh$(S, i) \cup$ flh-resolvents(res-flh$(S, i)$)

3. res-flh$(S, i+1)$ = res-flh$'(S, i+1)$ with all flh-subsumed clauses removed

for $0 \leq i \leq m$.

**Definition.** Let $S$ be a list, $C_0, C_1, \ldots, C_n$, of ordered electrons. Then $S$ is said to be *ETOS-ordered*, if, for each $i, j, k$, and $C'$, if $i \leq j \leq k$ and $C'$ is a suffix of both $C_i$ and $C_k$, then $C'$ is also a suffix of $C_j$.

Thus, if two members of $S$ have a common suffix, then any clauses between them has that same suffix.

**Example.** One possible ETOS-ordering of the set

$$
\begin{array}{r}
b\ g\ d\ e\ a \\
a\ c\ f\ d \\
f\ b\ g \\
b\ e\ f\ d \\
f\ c\ d\ e\ a \\
c\ a\ b\ g \\
g\ e\ f\ d \\
h\ e\ a \\
e\ b\ f\ d
\end{array}
$$

could be

$$
\begin{array}{r}
b\ g\ d\ e\ a \\
f\ c\ d\ e\ a \\
h\ e\ a \\
a\ c\ f\ d \\
b\ e\ f\ d \\
g\ e\ f\ d \\
e\ b\ f\ d \\
c\ a\ b\ g \\
f\ b\ g.
\end{array}
$$

Note that this is one of many possible ETOS-orderings. Also observe that, although $b\ e\ f\ d$ and $e\ b\ f\ d$ have the same literals, neither flh-subsumes the other since neither is a suffix of the other.

**Lemma 1.** Any finite set of ground clauses $P$ can be arranged into an ETOS-ordered list.

**Proof.** One way to do this would be to perform a reverse lexicographical sort of the clauses. That is, sort the clauses according to the last literal, then according to the penultimate literal, and so on. Any arbitrary ordering of the atoms may be taken beforehand.

**Lemma 2.** Let

$$R = \text{merge-right}(\alpha_n \ldots \alpha_1 \alpha_0).$$

Then $R$ can be represented by

$$R = \alpha'_n \ldots \alpha'_1 \alpha'_0$$

where

$$\begin{aligned} \alpha'_0 &= \alpha_0 \\ \alpha'_1 &= \alpha_1 \rightsquigarrow \alpha_0 \\ \alpha'_2 &= \alpha_2 \rightsquigarrow (\alpha_1 \alpha_0) \\ &\vdots \\ \alpha'_n &= \alpha_n \rightsquigarrow (\alpha_{n-1} \ldots \alpha_1 \alpha_0). \end{aligned}$$

**Proof.** The primed literal sequences $\alpha'_i$ follow from the definition of merge-right. Thus, any literal which appears in both any given $\alpha_i$ and $\alpha_{i-1} \ldots \alpha'_0$ will be merged-right into the literal sequence $\alpha_{i-1} \ldots \alpha'_0$ on its right, effectively removing it from $\alpha_i$, which we denote by $\alpha'_i$. Note that $\alpha'_i$ is a subsequence of $\alpha_i$.

**Lemma 3.** If

$$R = \alpha'_n \ldots \alpha'_1 \alpha'_0$$

and $R'$ flh-subsumes $R$, then

$$R' = \alpha''_j \ldots \alpha'_1 \alpha'_0$$

for $0 \leq j \leq n$.

**Proof.** Since $R'$ is a suffix of $R$, then $R = C\ R'$ for some sequence of literals $C$, by definition. Now,

$$R = \alpha'_n \ldots \alpha'_j \ldots \alpha'_1 \alpha'_0.$$

Hence, for some $j$ such that $0 \leq j \leq n$,

$$R' = \alpha''_j \ldots \alpha'_1 \alpha'_0$$

where $\alpha''_j$ is a suffix of $\alpha'_j$ (so that $C$ would be $\alpha'_n \ldots \alpha'''_j$ where $\alpha'_j = \alpha'''_j \alpha''_j$).

**Lemma 4.** Every subsequence of an ETOS-ordered sequence is ETOS-ordered.

**Proof.** Although this is obvious, we give a short proof of the fact. Assume this is false. Then there is some subsequence $C_0, \ldots, C_m$ of an ETOS-ordered sequence $E_0 \ldots E_n$ which is not ETOS-ordered. In particular, there is a subsequence $C_i, C_j, C_k$, $i < j < k$ such that $C_i$ and $C_k$ have a common suffix which is not a suffix of $C_j$. However, $C_i$, $C_j$, and $C_k$ correspond to some $E_{i'}$, $E_{j'}$, and $E_{k'}$, respectively such that $i' < j' < k'$. Then $E_{i'}$ and $E_{k'}$ have a common suffix which is not a suffix of $E_{j'}$. This contradicts the fact that $E_0 \ldots E_n$ is ETOS-ordered.

**Lemma 5.** If, in an ETOS-ordered sequence of electrons,

$$R' = \alpha''_j \ldots \alpha'_k \ldots \alpha'_1 \alpha'_0$$

and

$$\begin{aligned} C_0 &= L_0\, \alpha_0 \\ C_1 &= L_1\, \alpha_1 \\ &\vdots \\ C_k &= L_k\, \alpha_k \\ R' &= \alpha''_j \ldots \alpha'_k \ldots \alpha'_1 \alpha'_0 \end{aligned}$$

is a subsequence of this sequence, where $0 \leq i \leq k < j$, then

$$\begin{aligned} \alpha_1 &= \alpha'_1 \alpha'_0 \\ \alpha_2 &= \alpha'_2 \alpha'_1 \alpha'_0 \\ &\vdots \\ \alpha_k &= \alpha'_k \ldots \alpha'_1 \alpha'_0. \end{aligned}$$

**Proof.** Note that $\alpha'_0 = \alpha_0$. We establish this by induction.

**Base Case.** Since $C_1$ lies between $C_0$ and $R'$, both of which have the common suffix $\alpha_0$, and since the clauses are ETOS-ordered, then $C_1$ also has this suffix. Hence,

$$C_1 = L_1\, \alpha_1 = L_1\, \alpha'_1 \alpha'_0.$$

Due to merging-right, we find that $\alpha'_1$ is primed, as it is in $R'$.

Now, as the induction hypothesis, we assume that, for each $C_i$ between $C_{i-1}$ and $R'$,

$$C_i = L_i\, \alpha'_i \ldots \alpha'_1 \alpha'_0.$$

Then, $C_{i+1}$ is between $C_i$ and $R'$. Thus, it, too, has the suffix $\alpha'_i \ldots \alpha'_1\alpha'_0$, by the ETOS-ordering of the clauses. Hence,

$$C_{i+1} = L_{i+1}\ \alpha'_{i+1}\alpha'_i \ldots \alpha'_1\alpha'_0.$$

Merging right of the literals in $\alpha_{i+1}$ with the suffix $\alpha'_i \ldots \alpha'_1\alpha'_0$ results in that literal sequence also being primed: $\alpha'_{i+1}$.

Note that, if in the statement of the lemma, $k = j$, then

$$R' = \alpha''_j \ldots \alpha'_1\alpha'_0,$$

would flh-subsume $C_j$ since it would then be a suffix of this clause. This is an important fact that will feature in the completeness proof which follows in the next section. The upshot of this is that, if $R'$ is this suffix, it cannot follow $C_j$ in the list of ETOS-ordered electrons.

# 3 Proof of Ground Completeness

**Theorem 1.** If $S$ is a finite unsatisfiable set of ordered ground clauses, then $\Box \in$ res-flh$(S, m)$ for some $m$.

**Proof.** Since $S$ is finite, it follows that res-flh$(S, m+1)$ = res-flh$(S, m)$ for some $m$. Let $S'$ = res-flh$(S, m)$. We show that $\Box \in S'$.

The proof proceeds by contradiction. Assume that $\Box \notin S'$.

Let $P$ be the set of electrons in $S'$. Observe that, by the definition of *res-flh*, no clause in $S'$ properly flh-subsumes another clause. This gives us the following fact:

**Fact0.** For each electron $E$ in $P$, no proper suffix of $E$ occurs in $P$.

That is, there is no clause $E'$ in $P$ which is a proper suffix of $E$.

By Lemma 1, The set of electrons, $P$, can be arranged into an ETOS-ordered sequence of clauses, $E_0, E_1, \ldots, E_p$.

We now use the following *modeling procedure* to define an interpretation, $\mathcal{I}$, which satisfies every electron in $P$. (Since $\Box \notin P$, each of the $E_i$'s is non-empty.) We use the term *FL-satisfy* to denote the action of assigning the truth value `T` to the first (leftmost)

literal of an electron. When such assignment also satisfies a subsequent clause in which that literal occurs (not necessarily first), we say that that clause is "knocked out". We proceed clause by clause down the ETOS-ordered sequence of clauses.

1. FL-satisfy $E_0$. (At this point $\mathcal{I}$ contains only the first literal of $E_0$.)

   The interpretation $\mathcal{I}$ now satisfies $E_0$ and perhaps other $E_i$'s.

2. FL-satisfy the *next* clause $E_i$ which has not been already satisfied (knocked out) under $\mathcal{I}$. (I.e., add the first literal of this $E_i$ to $\mathcal{I}$.)

3. Repeat Step 2 until all members of $P$ are satisfied by $\mathcal{I}$.

4. Assign `F` to all literals not in $\mathcal{I}$. (I.e., leave $\mathcal{I}$ as it is.)

Since $S'$ is unsatisfiable and $\square \notin S'$, it follows that there is a nucleus $N$ in $S'$ which is falsified by $\mathcal{I}$. Let

$$N = \alpha_n \sim L_{n-1} \ldots \sim L_1 \ \sim L_0,$$

where $\alpha_n$ is the (ordered) positive literals of $N$, if any, and $\sim L_{n-1} \ldots \sim L_1 \sim L_0$ are the negative literals of $N$.

Note that no literal in $\alpha_n$ was assigned the value `T` by the modeling procedure above. Also, every literal $L_{n-1}, \ldots, L_1, L_0$ in $N$ was assigned `T` at some point during this procedure. This means there is a subsequence of clauses, $C_0, C_1, \ldots, C_{n-1}$, in the ETOS-ordered sequence, $E_0, E_1, \ldots, E_p$ which were FL-satisfied by the assignment of `T` to some $L_i$. Since the negative literals are *not ordered*, it follows that the nucleus $N$ is not changed in value if we permute its negative literals. So, without loss of generality, we assume that $L_i$ *corresponds* to the first literal of $C_i$, for each $i$. Then, we can represent the relative position of the $C_i$'s to each other as follows:

| | |
|---|---|
| $C_0 = L_0 \ \alpha_0$ | The first place in the ETOS-ordering where $L_0$ is assigned `T` . |
| {Other clauses} | |
| $\vdots$ | |
| $C_i = L_i \ \alpha_i$ | The first place in the ETOS-ordering where $L_i$ is assigned `T` . |
| $\vdots$ | |
| {Other clauses} | |
| $C_{n-1} = L_{n-1} \ \alpha_{n-1}$ | The first place in the ETOS-ordering where $L_{n-1}$ is assigned `T` . |

Let $R$ be the flh-resolvent produced by the nucleus $N$ and the electrons, $C_0, \ldots, C_{n-1}$. Then

$$\begin{aligned} R &= \text{merge-right}(\alpha_n \ldots \alpha_1 \alpha_0) \\ &= \alpha'_n \ldots \alpha'_1 \alpha'_0 \end{aligned}$$

by Lemma 2.

As we have assumed that res-flh$(S, m+1)$ = res-flh$(S, m)$ for some $m$, it follows that $R$ is either already in $P$ or is (properly) flh-subsumed by some member $R'$ of $P$. In either case, $R$ is flh-subsumed by a member $R'$ of $P$. By Lemma 3, $R'$ can be represented by

$$R' = \alpha''_j \ldots \alpha'_1 \alpha'_0$$

where $0 \leq j \leq n$.

We will show that $R'$ could not have been satisfied during the modeling procedure and complete the proof by contradiction.

Since $R'$ is an electron, it is positioned somewhere in the ordered electron list, $E_1 \ldots E_p$ (i.e., it is one of the $E_i$'s), and some literal $L_{R'}$ in $R'$ is assigned `T` at some point in the modeling procedure. This could have occurred in either of two different ways:

1. $R'$ was FL-satisfied by the procedure
2. $R'$ was knocked out by the FL-satisfaction of another clause occurring before $R'$ in the procedure.

In either case, we let $E_q$ be the clause in which $L_{R'}$ is first assigned `T` . Thus, $L_{R'}$ is the first literal of $E_q$.

Now, $L_{R'}$ occurs in $\alpha'_k$ for some $0 \leq k \leq j$ in $R'$[4]. Note that $L_{R'}$ cannot occur in $\alpha'_n$ because no literal in $\alpha'_n$ is ever assigned `T` in the modeling procedure. This implies that $R'$ does not occur before $C_k$ in the sequence $E_0 \ldots E_p$ of electrons. If it did, then $C_k$ would have been knocked out by the assignment of `T` to $L_{R'}$ (recall that $E_q$, the clause in which this assignment first occurs, either *is* $R'$ or occurs before $R'$). By the same reasoning, $E_q$ also does not occur before $C_k$. Furthermore, $E_q \neq C_k$ since $L_{R'} \neq L_k$. If this were not the case, then $L_{R'}(= L_k)$ would be merged-right into its own suffix since $L_{R'} \in \alpha'_k \subset \alpha_k$. Therefore, $E_q$ necessarily occurs after $C_k$. Since $R'$ either is $E_q$ or follows it, it also occurs after $C_k$.

[4] If $k = j$, then $L_{R'}$ occurs in $\alpha''_j$.

We now show that $E_q$ could not have been the clause in which $L_{R'}$ was first assigned T (and, consequently, no clause could have been since we chose $E_q$ based on the assumption that it *was* the clause in which this assignment first occurred). This, in turn, implies that $R'$ was never satisfied in the modeling procedure, contrary to our construction of $\mathcal{I}$.

Now, $E_q$ has the form

$$E_q = L_{R'}\ \gamma.$$

If $E_q$ is $R'$, then $\gamma = \alpha''_j \ldots \alpha'_1 \alpha'_0$, where $\alpha''_j$ is a suffix of $\alpha'_j$. Otherwise, we have the following subsequence:

$$\begin{aligned}
C_0 &= L_0\ \alpha_0 \\
&\vdots \\
C_1 &= L_1\ \alpha_1 \\
&\vdots \\
C_k &= L_k\ \alpha_k \\
&\vdots \\
E_q &= L_{R'}\ \gamma \\
&\vdots \\
R' &= \alpha''_j \ldots \alpha'_k \ldots \alpha'_1 \alpha'_0,
\end{aligned}$$

By Lemma 4, this subsequence is ETOS-ordered. By Lemma 5, then, $\alpha_k = \alpha'_k \ldots \alpha'_1 \alpha'_0$, and hence

$$C_k = L_k\ \alpha'_k \ldots \alpha'_1 \alpha'_0.$$

Since $E_q$ lies between $C_k$ and $R'$, it also has the suffix $\alpha'_k \ldots \alpha'_1 \alpha'_0$ so that

$$E_q = L_{R'}\ \gamma' \alpha'_k \ldots \alpha'_1 \alpha'_0.$$

where $\gamma' = \gamma \rightsquigarrow (\alpha'_k \ldots \alpha'_1 \alpha'_0)$. But $L_{R'} \in \alpha'_k$. Therefore, it is merged into $\alpha'_k$, in which case, if $\gamma'$ is non-empty, $L_{R'}$ is not the first literal of $E_q$ as assumed. If $\gamma'$ is empty, then, as above, $E_q$ has the form

$$E_q = L_{R'}\ \alpha''_k \ldots \alpha'_1 \alpha'_0.$$

But $L_{R'} \alpha''_k$ is a suffix of $\alpha'_k$. Accordingly, $E_q$ flh-subsumes $C_k$, contradicting our assumption that no clause in $P$ is flh-subsumed by another.

This demonstrates that $R'$ cannot be consistently satisfied under the modeling procedure given, which contradicts the fact that all electrons are satisfied by the procedure. Since the modeling procedure ensures that all electrons are satisfied, no nucleus can be falsified under $\mathcal{I}$ and, so, $\mathcal{I}$ is actually a model for the set $S$, which contradicts its unsatisfiability. Therefore, first-literal hyper-resolution is complete in the ground case.

# 4 Proof of General Completeness

In this section we define first-literal hyper-resolution for the predicate case, extend the definitions of Section 3 to FOL, and prove the lifting lemma and ground completeness for FLH.

**Definition.** A clause $C'$ is said to be an *flh-factor* of a clause $C$ if there is a most general unifier $\theta$ such that $C' = \text{merge-right}(C\theta)$.

**Definition.** A clause $C'$ is said to be an *flh-instance* of a clause $C$ if there is a substitution $\sigma$ such that $C' = \text{merge-right}(C\sigma)$.

**Definition.** A *ground flh-instance* is a flh-instance in which no variable occurs in any term.

**Definition.** Let

$$N = N' \ \sim L_0 \ \sim L_1 \ldots \sim L_{n_N}$$

be a nucleus (or flh-factor of a nucleus) in $S$ and $E_N$ be the electron set (depending on $N$) composed of the electrons (or flh-factors of electrons)

$$\begin{aligned} E_0 &= L'_0 \ E'_0 \\ E_1 &= L'_1 \ E'_1 \\ &\vdots \\ E_{n_N} &= L'_{n_N} \ E'_{n_N}. \end{aligned}$$

If the set of literals $\{L_0, L'_0, \ldots L_{n_N}, L'_{n_N}\}$ have a most general unifier $\sigma$, then

$$R = \text{flh-resolvent}(N, E_N) = \text{merge-right}((N' \ [E'_0, E'_1, \ldots, E'_{n_N}])\sigma)$$

is a *flh-resolvent* of clauses $E_0, \ldots, E_{n_N}$ and $N$. As in the ground case, the brackets "[ ]" signify that the enclosed arguments may be permuted in any order.

**Example.** Let

$$\begin{aligned} E_0 &= P(f(y), x) \ Q(x, a) \\ E_1 &= P(g(z), z) \ R(z) \ P(g(c), v) \\ E_2 &= Q(b, f(c)) \end{aligned}$$

be electrons and

$$N = R(a) \ \sim P(f(a), b) \ \sim R(w) \ \sim Q(u, f(w))$$

be a nucleus. Since $R(c)\ P(g(c),c)$ is a flh-factor of $E_1$, we obtain

$$\begin{aligned} R &= R(a)\ Q(b,a)\ P(g(c),c) \\ \text{or} &= R(a)\ P(g(c),c)\ Q(b,a) \end{aligned}$$

as possible flh-resolvents.

**Lemma 6.** If $C^g$ is a ground flh-instance of $C$, then there exists a flh-factor $C^f$ of $C$ such that $|C^f| = |C^g|$.

**Proof.** Since $C^g$ is a ground flh-instance of $C$, there exists a substitution $\sigma$ such that $C^g = C\sigma$. For each $L_i^g$ in $C^g$, $\sigma$ unifies a set of one or more literals $\mathcal{L}_i$ in $C$ to $L_i^g$. Let $\tau$ be the most general unifier that unifies all of these $\mathcal{L}_i$'s. By the definition of merge-right, the literals in any flh-instance of $C$ (such as a ground flh-instance or flh-factor) cannot be a rearrangement of the literals in $C$ because order is preserved. Then $C^f$ is the desired flh-factor and $|C^f| = |C^g|$. Note that, due to the preservation of the order of literals, each literal in $C^g$ is an instance of the corresponding literal in $C^f$.

We now state and prove the lifting lemma as applied to flh clauses.

**Lemma (Lifting Lemma).** If $R^g$ is a flh-resolvent of a nucleus

$$\begin{aligned} N^g &= N^{g\prime} \sim L^g_{N_0} \ \sim L^g_{N_k} \ldots \sim L^g_{N_n} \\ &\text{and electrons} \\ E_0^g &= L^g_{E_0}\ E_0^{g\prime} \\ &\vdots \\ E_n^g &= L^g_{E_n}\ E_n^{g\prime}, \\ &\text{so that} \\ R^g &= \text{merge-right}(N^{g\prime}\ E_0^{g\prime} \ldots E_n^{g\prime}) \end{aligned}$$

where the order of the $E_i^{g\prime}$'s in the merge-right argument is as shown, and where $N^g$ and the $E_i^g$'s are ground flh-instances of $N$ and $E_0, \ldots, E_n$, respectively, then there exists a flh-resolvent $R$ of $N$ and $E_0, \ldots, E_n$ such that $R^g$ is a ground flh-instance of $R$.

**Proof.** We can rename the variables so that no two clauses among $N, E_0, \ldots, E_n$ have any variables in common. Let $N^g$ be a ground flh-instance of $N$ and $E_0^g, \ldots, E_n^g$ be ground flh-instances of $E_0, \ldots, E_n$. Since there are no variables in common among the clauses, we can assume there is a most general unifier $\sigma$ such that $N^g = N\sigma$ and, for $0 \le j \le n$, $E_j^g = E_j\sigma$.

That is, we can assume that the ground flh-instances of the nucleus and electrons

$$N^g \quad E_0^g \quad E_1^g \quad \ldots \quad E_n^g$$

is derived from the set

$$N \quad E_0 \quad E_1 \quad \ldots \quad E_n$$

by applying $\sigma$.

Since $N^g$ is a ground flh-instance of $N$, it follows from Lemma 6 that there is a flh-factor $N^f$ of $N$ such that $N^g$ is a ground flh-instance of $N^f$. Similarly, for each $k$, since $E_k^g$ is a ground flh-instance of $E_k$ it follows from Lemma 6 that there is a flh-factor $E_k^f$ of $E_k$ such that $E_k^g$ is a ground flh-instance of $E_k^f$. Therefore, since $N$ and $E_0 \ldots, E_n$ have no variables in common, there exists a substitution $\tau$ for which $N^f = N\tau$ and $E_k^f = E_k\tau$ for each $k$. For each $k$, let $L_{E_k}^f$ be the first literal of $E_k^f$, and let $L_{N_k}^f$ be the corresponding literal of $N^f$. We see this as

$$\left.\begin{array}{cccccc} N & E_0 & E_1 & \ldots & E_n & \\ N^f & E_0^f & E_1^f & \ldots & E_n^f & \Big\}\tau \\ N^g & E_0^g & E_1^g & \ldots & E_n^g & \end{array}\right\}\sigma$$

Each member of the second line is a factor of the first, and has the same number of literals as the corresponding member of the third. Since members of the third line are ground flh-instances of the second, there exists a substitution $\mu$ such that $N^g = N^f\mu$, and $E_k^g = E_k^f\mu$, for each $k$.

Let $L_{E_k}^f$ be the first literal of $E_k^f$, for each $k$. Then

$$\begin{array}{rcl} N^g & = & N^{g\prime} \sim L_{N_0}^g \ \sim L_{N_k}^g \ldots \sim L_{N_n}^g \\ E_0^g & = & L_{E_0}^g \ E_0^{g\prime} \\ & \vdots & \\ E_n^g & = & L_{E_n}^g \ E_n^{g\prime}, \end{array}$$

and for each $k$, $L_{E_k}^f\mu = L_{E_k}^g = L_{N_k}^g = L_{N_k}^f\mu$. And therefore, $L_{E_k}^f\mu = L_{N_k}^f\mu$, which means that there is a mgu $\theta$ for which $L_{E_k}^f\theta = L_{N_k}^f\theta$, for each $k$. Therefore,

$$R = \text{merge-right}((N^{f\prime} \ E_0^{f\prime} \ldots E_n^{f\prime})\theta)$$

is a flh-resolvents of factors of $N$, $E_0, \ldots, E_n$. And also $R^g$ is a ground flh-instance of $R$, since $\theta$ is at least as general as $\sigma$. This completes the proof.

**Theorem 2. (Completeness of General First-Literal Hyper-resolution)** If $S$ is an unsatisfiable set of ordered clauses, then there is a deduction of $\Box$ from $S$ by first-literal hyper-resolution.

**Proof.** Let $S$ be an unsatisfiable set of clauses where the positive literals are ordered in each clause. By Herbrand's theorem, there is a finite unsatisfiable set $S'$ of ground instances of the clauses in $S$. By Theorem 1, there is a first-literal hyper-resolution deduction of $\Box$ from $S'$. Using the previous lifting lemma, we can obtain a general flh-hyper-resolution deduction of $\Box$ from $S$.

# 5 ETOS-FLH-Resolution

The proof method described in the preceding sections suggests a refinement to first-literal hyper-resolution.

We repeat the definition of *ETOS-ordering* here for convenience:

**Definition.** Let $S$ be a sequence, $C_0, C_1, \ldots, C_n$, of ordered electrons. Then $S$ is said to be *ETOS-ordered*, if, for each $i, j, k$, and $C'$, if $i \leq j \leq k$ and $C'$ is a suffix of both $C_i$ and $C_k$, then $C'$ is also a suffix of $C_j$.

The power of the method that we will now describe lies in the restriction on the number of resolvents that may be generated during resolution. Under first-literal hyper-resolution, all permutations of the electrons must be explored to ensure completeness. If there are $n$ electrons to be resolved against a given nucleus, this can mean up to $n!$ flh-resolvents. If, instead, the electrons are initially ETOS-ordered and that ordering is maintained during the resolution process, only one flh-resolvent is generated. The primary advantage of first-literal hyper-resolution is retained, that of confining the search to the first literals of electrons. Moreover, ETOS-FLH-Resolution can be efficiently implemented as a tree structure as described below.

The ground version of ETOS-FLH-Resolution is defined below, but the general version is not. This work is currently being researched.

The proof of the ground completeness of FLH-Resolution presented in Section 3 is based on ETOS-FLH-Resolution. Consequently, it assures us that ETOS-FLH-Resolution itself

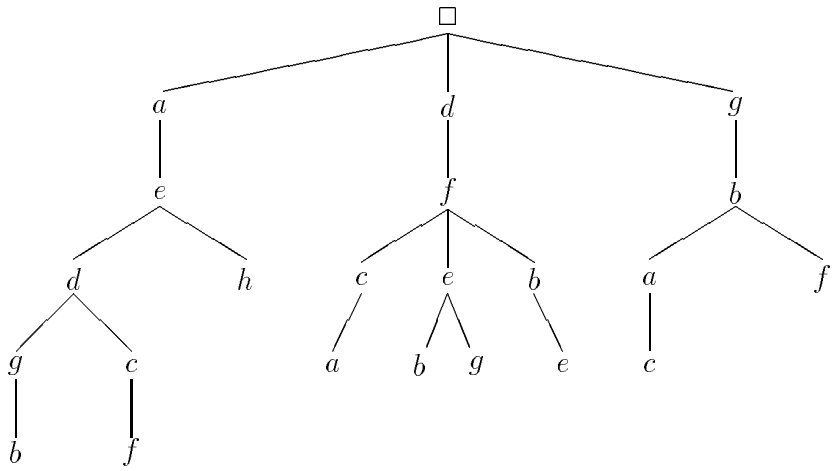
is complete. Thus, the essential elements of the proof, ETOS-ordering, suffix, and how a resolvent is produced, are all motivated by the proof method described in this section. The ETOS-ordering of electrons used by the proof gives rise to an important structure in ETOS-FLH-Resolution called an *electron tree*. Because ETOS-FLH-Resolution is based on positive hyper-resolution, the set of nuclei never changes. However, electrons are generated and subsumed during the proof process. We capture this process and the essential concept of *suffix* with the following definition:

**Definition.** An *electron tree* is a tree in which every path from a leaf to the root represents a unique electron in a set $P$ of ETOS-ordered electrons. The root is $\Box$, which is a suffix of every electron. Each leaf is the first literal of the electron and each node from the leaf to the root is each subsequent literal in the electron.

**Example.** Let $P$ contain the following electrons from an example in Section 2.

$$
\begin{aligned}
b\ g\ d\ e\ a\\
f\ c\ d\ e\ a\\
h\ e\ a\\
a\ c\ f\ d\\
b\ e\ f\ d\\
g\ e\ f\ d\\
e\ b\ f\ d\\
c\ a\ b\ g\\
f\ b\ g.
\end{aligned}
$$

Then one electron tree for $P$ is:




**Definition.** We say that a sequence of literals $L_j \ldots L_n$ *occurs* in an electron tree $T$ if there is a path from $L_j$ to $L_n$ and $\square$ is the parent node of $L_n$. This simply means that $L_j \ldots L_n$ is a suffix of some electron represented in $T$.

We give a lemma here which is important to the Electron Tree Ordering Strategy (ETOS) refinement of FLH-Resolution presented below.

**Lemma 7.** Given any electron tree, we can always add a new electron to the tree and maintain the ETOS-ordering of the electrons represented in the tree.

**Proof.** Let

$$E = L_0 \ldots L_n$$

be an electron we wish to add to an existing electron tree $T$. We examine $T$ to find the longest suffix $L_i \ldots L_n$ of $E$ occurring in $T$. There are three possibilities:

1. $\square$ is the only suffix of $E$ occurring in $T$, but $T \neq \square$ (i.e., there is at least one non-empty branch of $T$ representing another electron).

   Then, we simply add $E$ to $T$ so that $L_0$ is a leaf and $\square$ is the parent node of $L_n$ and the path read from $L_0$ to $\square$ denotes $E$.

2. $L_i \ldots L_n$ occurs in $T$ and $L_i$ is a leaf.

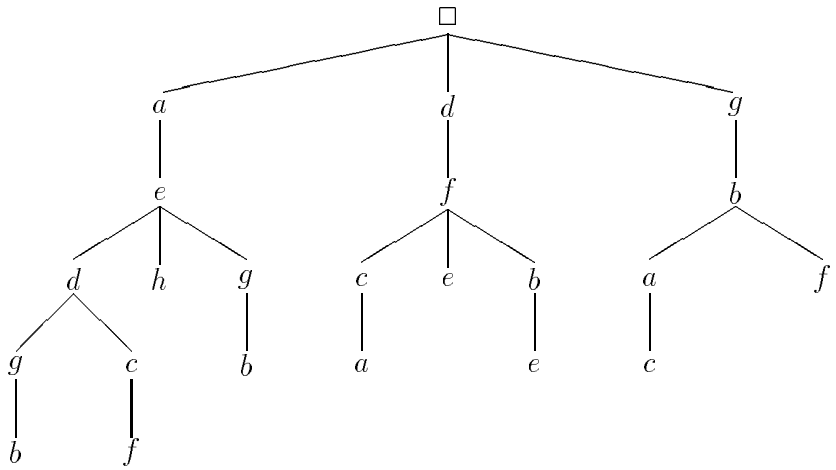
   Then, $L_i \ldots L_n$ is an electron represented in $T$ which flh-subsumes $E$. In this case, we do not add $E$ to $T$.

3. $L_i \ldots L_n$ occurs in $T$, but $L_i$ is not a leaf.

Then, $L_i \ldots L_n$ is a suffix of some electron represented in $T$. We add $L_0 \ldots L_{i+1}$ to $T$ by attaching $L_{i+1}$ to the node labeled $L_i$. $E$ is now represented as a new electron in $T$ and the ETOS-order is maintained because we attached $E$ to the longest suffix of it occurring in $T$. Hence, if $E$ now occurs between two clauses with a common suffix, then $E$ has that suffix as well. Moreover, if $L_i$ is $L_0$, then $E$ subsumes all electrons for which $L_0 \ldots L_n$ is a suffix. In this case, all branches for which $L_0$ is the parent node are removed from $T$. $L_0$ then becomes a new leaf in $T$.

Since item 1 is a special case of item 3 ($\square$ is a suffix of every clause), these are the only possibilities which need to be addressed. The procedure is analogous to adding a new word to a dictionary except that we use suffixes instead of prefixes to determine placement, and we are not concerned about the relative order of the clauses, only that common suffixes occur together.

**Example.** Suppose we wish to add the clauses $b\ g\ e\ a$ and $e\ f\ d$ to the tree given in the previous example. Since $e\ f\ d$ occurs in the tree and $e\ a$ is the longest suffix of $b\ g\ e\ a$ occurring in the tree, we add $e\ f\ d$ and $b\ g\ e\ a$ as below:




Since $e\ f\ d$ subsumes $b\ e\ f\ d$ and $g\ e\ f\ d$, the leaves $b$ and $g$ are removed so that $e$ becomes a new leaf; $b\ g\ e\ a$ is simply added because it neither subsumes nor is subsumed by any electron in the tree.

With this ordering comes a refinement to flh-resolvent, that of *etos-flh-resolvent*:

**Definition.** We produce an etos-flh-resolvent in the following manner:

Let

$$\begin{aligned} E_0 &= L_0\ E_0' \\ E_1 &= L_1\ E_1' \\ &\vdots \\ E_{N_n} &= L_{N_n}\ E_{N_n}' \end{aligned}$$

be electrons in $S$, read from left to right in the tree. Then the flh-resolvent resulting from hyper-resolution with the nucleus

$$N = N' \ \sim L_0 \ldots \sim L_{N_n}$$

is

$$R = \text{merge-right}(N'\ E_{N_n}' \ldots E_0')$$

is called an *etos-flh-resolvent* of $N$ and $E_0, \ldots, E_{N_n}$. That is, we resolve from right to left in the tree. This will result in $R$ being added to the same subtree as $E_0$ (if it is not subsumed by an electron already there. If, instead, it subsumes one or more clauses, their literals up to the node where the first literal of $R$ begins are removed. This node then becomes a leaf, by Lemma 7.

**The Electron Tree Ordering Strategy (ETOS)**

Given a set $S$ of ground clauses

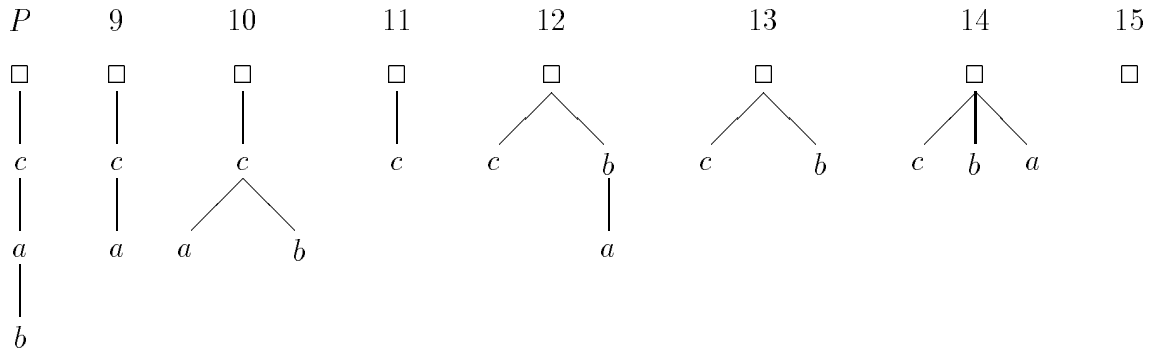
1. Construct an electron tree $T$ for the set $P$ of electrons in $S$.
2. If there is an etos-flh-resolvent not subsumed by an electron already in $T$, then we add it to $T$, as described in Lemma 7.
3. We continue in this manner until either the root of the tree is reached, or no new etos-flh-resolvent can be produced.

We illustrate this process with a couple of examples.

**Example.** Let $S$ be the full set on three literals. Then, an ETOS-FLH-deduction of $\Box$ from $S$ would be as follows:

| | | |
|---|---|---|
| 1 | $b\ a\ c$ | |
| 2 | $a\ b\ \sim c$ | |
| 3 | $c\ a\ \sim b$ | |
| 4 | $a\ \sim b\ \sim c$ | |
| 5 | $c\ b\ \sim a$ | |
| 6 | $b\ \sim a\ \sim c$ | |
| 7 | $c\ \sim a\ \sim b$ | |
| 8 | $\sim a\ \sim b\ \sim c$ | |
| 9 | $a\ c$ | 3:1 |
| 10 | $b\ c$ | 5:9 |
| 11 | $c$ | 7:10,9 |
| 12 | $a\ b$ | 2:11 |
| 13 | $b$ | 6:12,11 |
| 14 | $a$ | 4:13,11 |
| 15 | $\square$ | 8:14,13,11 |

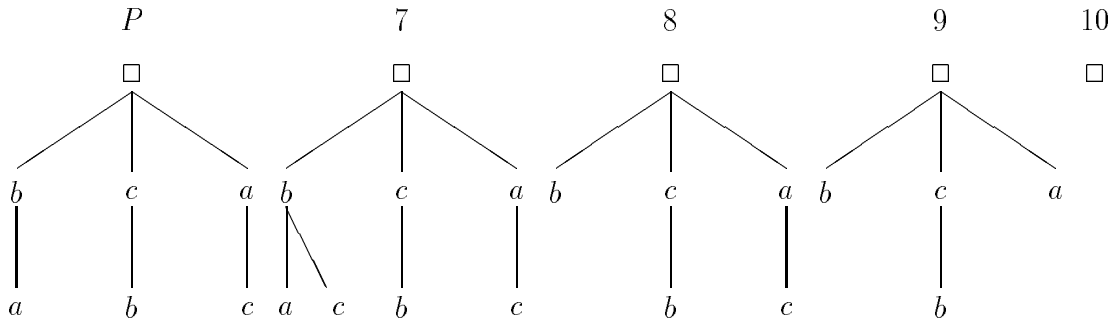
Letting $P$ be $\{\{b\ a\ c\}\}$, containing the only electron in $S$, and numbering each subsequent tree with the corresponding deduction step, we have the following series of electron trees:




**Example.** The following is an ETOS-FLH-deduction of $\square$ from the unsatisfiable set $S$ of clauses $\{\{a\ b\}, \{b\ c\}, \{c\ a\}, \{\sim a\ \sim b\}, \{\sim b\ \sim c\}, \{\sim c\ \sim a\}\}$:

| | | |
|---|---|---|
| 1 | $a\ b$ | |
| 2 | $b\ c$ | |
| 3 | $c\ a$ | |
| 4 | $\sim a\ \sim b$ | |
| 5 | $\sim b\ \sim c$ | |
| 6 | $\sim c\ \sim a$ | |
| 7 | $c\ b$ | 4:2,1 |
| 8 | $b$ | 6:7,1 |
| 9 | $a$ | 5:3,8 |
| 10 | $\square$ | 4:9,8 |

Letting $P$ denote the initial set of electrons in $S$ and numbering each subsequent tree with the deduction step it corresponds to, we have the following sequence of ETOS resolution trees:




# 6 Concluding Remarks

First-literal hyper-resolution and a refinement, ETOS-FLH-resolution (as given here for the ground case), suggest that the breadth of powerful resolution techniques have not yet been exhausted. These two methods offer advantages by restricting the scope of search for eligible clauses for resolution to the first literals of positive clauses, which can be matched against a table of negative literals indexed by nuclei. Since only positive clauses are generated as resolvents, this restriction is only enhanced during the proof process. It is also worth noting that the methods lend themselves handily to computer implementation. Many of the features of FLH-resolution can be implemented directly in LISP (e.g., merge-right translates easily into the LISP function `remove-duplicates`). Ground ETOS-FLH-resolution has the further advantage of specifying a tree structure for electrons and requiring only one resolvent be produced at each resolution step. The general case is still under investigation.

An important consideration of the completeness of FLH-resolution, aside from the interest of the proof in its own right because of its difficulty, is the implication of completeness of an equivalent resolution version of the interactive natural deduction prover, IMPLY.

## 6.1 Acknowledgements

Several people have lent their expertise to this paper in one way or another. Foremost, I would like to thank Dr. Bledsoe for his patient treatment of my many proof attempts, and for the problem itself. Ruben Gamboa helped me formalize my ideas on the method presented in the paper when the proof itself was still in embryonic form. Dr. Loveland provided valuable suggestions on the presentation and matters which needed to be addressed in comparison with existing resolution techniques. Both Drs. Larry Hines and Don Simon have offered many other useful suggestions on the structure of the proof as well.

# References

[1] Anderson, R. and Bledsoe, W.W. A linear format for resolution with merging and a new technique for establishing completeness. *J. ACM 17* (July 1970), 525-534.

[2] Bledsoe, W.W. and P. Bruell. A Man-machine Theorem Proving System, Proc 3rd IJCAI (1973), Stanford U.; also in AI Jour 5 (1974) 51-72. Bledsoe, W.W. and Mabry Tyson. The UT Interactive Prover. Memos ATP17A & ATP17B, Math Dep, Univ Texas, 1975, 1983. Bledsoe, W.W., Non-Resolution Theorem Proving, Artificial Intelligence 9 (1977) 55-77. In Reading in Artificial Intelligence (Webber, Nilsson, Eds), Tioga, Palo Alto, 1981, pp 91-108.

[3] Boyer, Robert S. Locking: a Restriction of Resolution. Ph. D. Thesis, University of Texas at Austin, Texas, 1971, 74 pp.

[4] Chang, C., and Lee, R. *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, Inc., 1973, 331 pp.

[5] Loveland, D.W. *Automated Theorem Proving: A Logical Basis*, North-Holland, Amsterdam, 1978, xiii + 405 pp.

[6] Robinson, J.A. A machine oriented logic based on the resolution principle. *J. ACM 12* (January 1965), 23-41.

[7] Robinson, J.A., Automatic deduction with hyper-resolution. *Internat. Jour. of Computer Math. 1* (1965), 227-234.