Automated Reasoning with Extended Linking and Left Merging

Chuen-Hsuen Jeff Ho
Department of Computer Science and Engineering
Chung-Cheng Institute of Technology
Ta-Hsi, Tao-Yuan, Taiwan, ROC
jeffho@cs.ccit.edu.tw

Lawrence Henschen
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60201, USA
henschen@eecs.nwu.edu

Abstract

Extended Linking Strategy (ELS) is a hyper-style strategy whose underlying principle is to control and perform (extend) a series of standard resolution on clauses. However it may be treated as a unique inference rule that serially links several resolution steps into one. We have defined the clause chain, introduced the ideas of ELS with left merging (LELS). We also presented the soundness and completeness proofs for ground LELS and used a fundamental theorem of logic (Herbrand's theorem) and facts about the unification algorithm to show that LELS is in fact complete for the first-order predicate calculus. We also touched on some consequences and benefits of a strategic nature that are present when employing LELS. Employment of LELS enables an automated reasoning program to draw conclusions in fewer steps that typically require many steps when unlinked inference rules are used.

1. Introduction

The content of this paper is standard in the literature on automated reasoning. We assume that the reader has familiarity with resolution and first order logic. For further details and references on the definitions and ideas of Extended Linking Strategy (ELS) introduced here see [5].

ELS starts linking from a start clause $C_0$ to clause $C_1$ by connecting a literal of $C_0$ and its counterpart in $C_1$. It then picks one of the remaining literals in $C_1$ and links (resolves with) to $C_2$, then from a literal of $C_2$ to $C_3$, etc. These links are called R-links. The unification follows the direction of the chaining. An R-path is a series of R-links in a clause chain. The length of the R-path is the number of R-links in the path. The start clause links to only one clause and the end clause is
linked only from a single clause. All intermediate clauses are linked to exactly two other clauses (the immediate predecessor and immediate successor). ELS chains will not be allowed to be cyclic. The prohibition of cycling is to assure that the linked clauses correspond to a sound deduction and to avoid tautologous results.

Merging identical and unifiable literals is allowed during the chaining process. Left merging is similar to the merge left operation used in OL-resolution (see [1]). Left merging keeps the very first occurrence of a literal and deletes the identities appearing at other clauses of a clause chain. Following is the formal definition of left merging.

**DEFINITION:** A literal $L$ of a clause $C_i$ in a chain $\&$ is left merged with a literal $L'$ of $C_j, j < i$ in $\&$ if $L$ is not in the $R$-path and $L, L'$ are unifiable with a unifier $\sigma$.

**DEFINITION:** ELS with left merging is denoted as LELS. A clause chain established by applying LELS is an LELS chain. The resolvent of a chain is denoted as CCR.

**DEFINITION:** Let $\&$ be a chain, and let $C_i$ be a clause in $\&$. $RLST(C_i)$ is the sequence $\{L_0, \cdots, L_{i-1}\}$ of literals from $C_j, j < i, C_j \in \&$ with outgoing $R$-links.

**DEFINITION:** A literal $L$ of a clause $C_i$ in a chain $\&$ is backward deleted by $\neg L$ of a clause $C_j, j < i$ if $\neg L$ is an element of $RLST(C_i)$.

The termination conditions of an LELS chain plays a very important role in proving theorems. A chain without termination cannot have a resolvent. It also destroys the completeness properties in proofs. When a chain terminates, its CCR can be generated by resolving along the $R$-path of the chain. Resolving a clause chain is to collect all free-literals in the chain. The CCR is the disjunction of these free-literals. A literal is called a free-literal during a chaining process (after unification and substitution) if it is neither in the $R$-path nor a deleted literal, and nor a merged literal.

A clause chain $\& = \{C_1, \cdots, C_k\}$ terminates if the last clause $C_k$ is

1. a unit clause.
2. a nonunit clause with no available literals.
3. a nonunit with available literals and
   3a. cycle termination occurs; or
   3b. they are already in RLST; or
   3c. there are no more available target clauses in the search space.
4. the end clause because $k$ is the chain length limit set by the user.
The number of *free*-literals of a chain can be determined by the following: let \( l \) be the total number of literal occurrences in a clause chain \( \& \), let \( r \) be the length of the \( R \)-path of \( \& \), and let \( m \) be the number of merged literals and \( d \) be the number literals which are involved in the cycle termination condition. Then the number of *free*-literals of \( \& \) is \( l - 2r - m - d \). Note that \( d > 0 \) for chains ending by cycle termination and \( d = 0 \) for other LELS termination conditions.

Note that no intermediate resolvents are generated during the chaining. Only the CCR is generated, and this only when the chain terminates. The length of each clause directly effects the length of the CCR. Hence clause length is good to use as a guide when choosing which clause to link with. Either the unit-preference strategy [11] or the smallest first strategy can be easily adopted as a guiding rule.

2. Ground LELS

Propositional (or ground) LELS is applied only to propositional (or ground) clauses. It leads naturally to a general principle of LELS for first-order clauses to be discussed later. It is thus natural to introduce first-order LELS by first expressing the rule at the simpler ground level.

**Definition:** A ground LELS chain is a LELS chain containing only ground clauses.

**Definition:** Let \( S \) be a ground set and \( C \in S \). An LELS-derivation from \( S \) is a sequence of clause chains \( \&_1, \ldots, \&_n \) with \( R_1, \ldots, R_n \) as the CCRs satisfying the following conditions:

1. \( \&_1 \) is a chain starting with \( C \).
2. \( \&_{i+1} \) is a chain starting with \( R_i, i \geq 1 \).
3. Each \( \&_i \) satisfies one of the chain termination conditions.

**Definition:** A clause chain is a *contradiction* if it has no *free*-literals.

**Definition:** An LELS-refutation of \( S \) is an LELS-derivation of a contradictory chain. Equivalently, it is a derivation whose last chain resolvent is the empty clause (\( \emptyset \)). We denote it by \( S \vdash_{\text{lels}} \emptyset \).

Most of the theorems in this paper are proved by using the *Splitting Rule* of Davis and Putnam[2]: Let \( S \) be a ground set and let \( P \) be an atom occurring in \( S \). \( S \) can be put into the form \( P \lor A_1, \ldots, P \lor A_m, \neg P \lor B_1, \ldots, \neg P \lor B_n, C_1, \ldots, C_k \), where \( A_i, B_i \), and \( C_i \) are free of \( P \) and \( \neg P \). Let \( S_1 = \{ A_1, \ldots, A_m, C_1, \ldots, C_k \} \) and \( S_2 = \{ B_1, \ldots, B_n, C_1, \ldots, C_k \} \). \( S \) is unsatisfiable if and only if both \( S_1 \) and
$S_2$ are unsatisfiable.

**Remarks:** Let $k(S)$ denote the number of occurrences of literals in $S$ minus the number of clauses in $S$. Let $S'$ be formed from $S$ by deleting a literal from a non-unit clause. Then $k(S') < k(S)$. Similarly, if $S'$ is formed from $S$ by deleting a clause from $S$, then $k(S') \leq k(S)$, and the comparison is strict if the deleted clause is non-unit. Finally, let $S_1(S_2)$ be one of the sets formed by splitting for a minimally unsatisfiable set $S$ of clauses. If $k(S) > 0$, then $k(S_i) < k(S)$ for $i = 1, 2$.

**Lemma 2.1** Let $C_0, \ldots, C_k$ be ground clauses and let $& = \{C_0, \ldots, C_k\}$ be a clause chain. Let $I$ be an interpretation. If $I$ satisfies each $C_i$, then $I$ satisfies the CCR of $&$.

**Proof:** Because computing the CCR from the clauses of the chain involves backward deletion, the proof is not a trivial consequence of the soundness of ordinary resolution. The proof is by induction on the length $k$ of $&$.

**Base Case:** $k = 1$. In this case the chain is just $\{C_0, C_1\}$, and the CCR is just an ordinary binary resolvent. The result follows from resolution theory.

**Induction Case:** $k > 1$. Suppose the result is true for chains of length less than $k$. Then the chain resolvent $\text{CCR}_{k-1}$ of $\{C_0, \ldots, C_{k-1}\}$ is satisfied by $I$. Now in computing the CCR for $&$, one literal from $\text{CCR}_{k-1}$ (in fact from $C_{k-1}$) is chosen to resolve with a literal from $C_k$. Let the chosen literal be $P$, and let $C_k$ be $\neg P \lor L_1 \lor \cdots \lor L_m$. By hypothesis, $I$ satisfies $C_k$.

a. If $I$ makes $P$ false, then some other literal of $\text{CCR}_{k-1}$ must be true, and all these literals also occur in $\text{CCR}_k$. Clearly, $\text{CCR}_k$ is satisfied by $I$.

b. Suppose now that $I$ makes $P$ true. Then one of the literals $L_m$ must be true because $I$ satisfies $C_k$. If any such $L_m$ remains in $\text{CCR}_k$, then again $I$ satisfies $\text{CCR}_k$. The only way this can fail is that each such $L_m$ is backward deleted. For $L_m$ to be backward deleted, there must have been clauses $C_i$ and $C_{i+1}$ earlier in the chain such that $C_i = \neg L_m \lor C_i'$ and $C_{i+1} = L_m \lor C_{i+1}'$ and the R-link from $C_i$ originated from $\neg L_m$. Moreover, since $L_m$ is true in $I$ in this subcase, some other literal of $C_i$ must be true, say $M$.

b1. Some true $M$ from $C_i$ is not itself backward deleted. If such an $M$ is not backward deleted, then either it occurs in $\text{CCR}_k$ or it is right merged and then resolved because every new R-link must emanate from a free literal of the most recent clause in the chain. If $M$ occurs in $\text{CCR}_k$, the $I$ satisfies $\text{CCR}_k$. If in fact $M$ had been right merged and resolved, then $M$ would occur on RLST($C_k$) to the right of $\neg L_i$ from $\text{CCR}_k$. Thus, $L_i$ would in fact occur in $\text{CCR}_k$, and $I$ would satisfy $\text{CCR}_k$.  

76 BISFAI-95
b2. Suppose all the true literals of \( C_i \) were themselves backward deleted.

By the induction hypothesis, \( C_{CR_i} \) is satisfied by \( I \). Recall, \( C_i = \neg L_m \lor M_1 \lor \ldots \lor C_v \) where \( M_j \) are all the true literals of \( C_i \). (Recall, \( L_m \) was assumed true in \( I \).) Therefore, some true literal from clause \( C_{v'}, v < i \) must occur in \( C_{CR_i} \). This literal can now play the role of \( M \) in subcase b1. 

Q.E.D.

The intention behind the application of LELS is to derive a contradictory chain from a ground clause set, proving the inconsistency of the set; this then proves the unsatisfiability of the set, given that LELS is sound. Soundness can be stated as follows:

**THEOREM 2.1** LELS is sound.

Proof: Trivial because each LELS chain resolvent is formed from the clauses by a sequence of ordinary resolutions. Q.E.D.

3. The Completeness of LELS Resolution

We prove the completeness of LELS resolution in two steps. Because the cycle termination property requires special consideration, we first prove the completeness of a modified LELS in which the cycle termination condition is not used.

**DEFINITION:** Simple LELS (SLELS) resolution is LELS in which cycle terminations are ignored.

Thus, chains are terminated in SLELS only when there are no available literals in the last clause from which to generate another \( R \)-link.

**THEOREM 3.1** Let \( S \) be a minimally unsatisfiable set of clauses. Let \( D \) be a clause of \( S \), and let \( P \) be a literal of \( D \). There exists an SLELS refutation of \( S \) starting from \( P \) in \( D \).

Proof: The proof is by induction on \( k(S) \).

**Base case:** \( k(S) = 0 \). Then \( S \) consists of two conflicting unit clauses, and the proof is trivial.

**Induction case:** \( k(S) > 0 \). Assume the result holds for any minimally unsatisfiable set \( S' \) with \( k(S') < k(S) \). Let the clauses of \( S \) be \( P \lor A_1, \ldots, P \lor A_n, \neg P \lor B_1, \ldots, \neg P \lor B_m, C_1, \ldots, C_k \), as usual, and suppose \( D \) is the clause \( P \lor A_1 \). There are two subcases.

1. \( A_1 \) is empty. Then the unit clause \( P \) occurs in \( S \), and there are no other clauses containing \( P \). Consider the set \( S_2 = \{B_1, \ldots, B_m, C_1, \ldots, C_k\} \). \( S_2 \)
linking, even if it got a Q added back. The result is a modified chain with exactly the same R-links and whose resolvent is $Q \lor CCR(\&_1)$. Again, the same argument can be repeated for the remaining chains of R. The result is an SLELS deduction $R'$ of the unit clause $Q$. We now consider the set $S_3$ formed by deleting all clauses containing $Q$ and adding the unit clause $Q$. Let $S_4$ be a minimally unsatisfiable subset of $S_3$. Again, $Q$ must occur in $S_4$. Because there is at least one non-unit clause from $S$ containing $Q$ (namely $P \lor Q \lor A'_1$), $k(S_4) < k(S)$. By induction, there is an SLELS refutation $RQ$ of $S_4$ starting from $Q$. Then the refutation formed by appending $RQ$ to $R'$ is an SLELS refutation of $S$ starting from $P$ in $D$.

Q.E.D.

**THEOREM 3.2** Let $S$ be a minimally unsatisfiable set of clauses, and let $D$ be a clause in $S$ with literal $P$. Then there is an LELS refutation of $S$ starting from $P$ in $D$.

**Proof:** By the previous theorem there is an SLELS refutation of $S$ starting from $P$ in $D$. In each chain we trace the chain to find any cycle termination conditions. Whenever one is found, the chain is broken, the chain resolvent formed, and a new chain starting from that chain resolvent using the same links as in the original chain is begun. Clearly the final resolvent for any such broken chain is the same as before. Of course, the resolvent for the last chain of the original refutation is then still the empty clause.

Q.E.D.

The proof for ground clauses may be lifted to first-order clauses by means of the standard "lifting lemma" explained in [7]. This lemma states that for any ground instances of clauses $C$ and $D$ each of their resolvents are instances of some resolvent of $C$ and $D$. Since, according to Herbrand’s theorem, a set of clauses is unsatisfiable if and only if there is a set of ground instances of them that is unsatisfiable, the lifting lemma assures that the use of the most general unifier is sufficient to achieve completeness. Note that the standard lifting lemma did not handle any merging situation. As stated in section 2 the ground LELS carries the left merging and possible cycle termination in a chaining process. For first order cases there will be no forced merging on two literals unless they are identical.

**LEMMA 3.1** If $C'_0, \ldots, C'_r$ are instances of $C_0, \ldots, C_r$, respectively, and $R'$ is a CCR of a clause chain $\&'$ of $C'_0, \ldots, C'_r$, then there is a CCR $R$ of $\&$ of $C_0, \ldots, C_r$ such that $R'$ is an instance of $R$.

**Proof:** Assume the variables are all separated. Let $R'_0, \ldots, R'_r$ be the literals in the RLST of $\&'$. Let $\gamma$ be the substitution that maps $C_i$ onto $C'_i$, $i = 0..r$. We construct a chain $\&$ starting from $C_0$. 
is, in fact, also minimally unsatisfiable, and of course \( k(S_2) < k(S) \). By induction, there is an SLELS refutation \( R \) of \( S_2 \) starting from some literal in \( B_1 \). Let the chains of \( R \) be \&1, \&2, \ldots, \&i. We consider these chains one by one.

\&1 : \( B_1 \xrightarrow{R} \cdots \xrightarrow{R} D_i \)

where there are no literals available in \( D_i \) for continued linking. Append \( \neg P \) to all the \( B_i \) clauses used in this chain and append a new \( R \)-link to the beginning as follows:

\[ P \xrightarrow{R} \neg P \lor B_1 \xrightarrow{R} \cdots \xrightarrow{R} D'_i \]

where each \( D'_i \) is either \( D_i \) or \( D_i \lor \neg P \).

Clearly the ordinary binary resolvent of the first link is exactly the beginning of \&1. Now suppose some of the \( D' \) clauses actually contain \( \neg P \). Let \( D_i \) be the first of these. We replace the outgoing link of \( D_i \) with a new \( R \)-link to the unit clause \( P \). If \( D_i \) is \( D_n \) (with no outgoing link) we simply add a new \( R \)-link. In either case, this results in a termination because the new clause is a unit. However, by our earlier remarks about literals always remaining in the chain resolvent if they are not connected to an \( R \)-link, it is clear that the chain resolvent formed at the point where \( P \) is added is exactly the same as the resolvent up to the corresponding point in \&1. If \( D_i \) was not \( D_n \), then we start another chain from the chain resolvent just formed using the same literal as in \&1 and linking to the same clauses as in \&1. We repeat the process until all occurrences of \( \neg P \) have been removed. The result is exactly \( \text{CCR}(\&1) \). (Note that in the modified deduction there may be several chains, but the last one will have the resolvent identical to \( \text{CCR}(\&1) \)). We now repeat the argument for the remaining chains.

2. \( A_1 \) is not empty. Let \( Q \) be a literal of \( A_1 \). Then \( D \) is the clause \( P \lor Q \lor A'_1 \). Let \( S_1 \) and \( S_2 \) be the split sets obtained by splitting on \( Q \), and let \( S_{1m} \) and \( S_{2m} \) be corresponding minimally unsatisfiable subsets. Again, note that \( P \lor A'_1 \) must be in \( S_{1m} \). \( k(S_{1m}) < k(S) \), so by induction there is an SLELS refutation \( R \) of \( S_{1m} \) starting from \( P \) in \( P \lor A_1 \). Now add \( Q \) back to all the clauses from which it was deleted, including the start clause. Consider the first chain of \( R_i \) and \&1. Again, by our remarks about literals remaining in the chains, it is clear that \( Q \) remains in \&1. Moreover, if any other clause in that chain gets the literal \( Q \) added back, it will be merged left. Of course all other mergings will be exactly the same as before. Thus, all occurrences of \( Q \) are merged, and no other literal in \&1 that was not merged before will be newly merged. This means that for each clause \( D_i \) in this chain exactly the same literals as before are available for linking, and therefore each former \( R \)-link can still be made. Finally, the last clause has no literals available for
Let $R_0$ be the set of literals from $C_0$ mapped onto $R'_0$ in $C'_0$ by $\gamma$. Let $-R'_0$ be the set of literals from $C_1$ mapped onto $-R'_0$ in $C'_1$ by $\gamma$. Form the resolvent in the normal way with $\alpha_0$ as mgu. If there were any left-merged literals, then if necessary unify the corresponding literals by additional substitution. Let $\tau_0$ be the final substitution. Clearly $\tau_0 \cdot \lambda_0 = \gamma$ from some $\lambda_0$. Now, if there was a termination condition in $\&'$, then there will be a corresponding termination condition in the first-order chain being constructed. For example, if all free-literals of $C'_1$ had been left merged, then all the literals of $C_1$ would have been optionally merged too. Similarly, if a literal in $C'_1$ formed a cycle, the corresponding literal(s) of $C_1$ would (after unification) still form a cycle. If there was no termination in $\&'$, then we extend the first-order chain using the same process. At each step, there will be a $\lambda_i$ such that $\tau_i \cdot \lambda_i = \gamma$. The free-literals remaining in the first-order chain will map onto $R'$ by the last $\lambda$, $\lambda_r$. 

Q.E.D.

**THEOREM 3.3** A set $S$ of clauses is unsatisfiable if and only if $S \vdash_{\text{ilel}} \varnothing$.

**Proof:** ($\iff$) As a consequence of the Herbrand theorem there exists a finite unsatisfiable set $S'$ of ground instances of $S$ since $S$ is unsatisfiable. Then the theorem holds by lifting Theorem 3.1 and Theorem 3.2.

($\implies$) Clearly first-order LELS is also sound.

Q.E.D.

4. Comments and Results

First, the LELS inference rules is clearly a hyper-type inference system. Many researchers have listed potential benefits of hyper-type inference rules - larger inferences are made in a single step, fewer clauses are actually added to long-term memory, clause processing such as demodulation and subsumption checks are made less often and on potentially more “useful” clauses, [8][9][13]. LELS is, however, quite different than other hyper methods. It does not have the syntactically oriented restrictions of positive/negative hyperresolution. Similarly for the restrictions on UR resolution [4][10], which requires that the end result be a unit clause. There are some aspects in common with the linking methodology because LELS can link in a chain through any number of 2-clauses (clauses with exactly 2 literals). However, unlike linked UR resolution [13], LELS concentrates only on one literal of the start clause.

LELS has several aspects in common with linear resolution and its variations. Any one chain is a linear deduction in which the selected literal always comes form the most recent side clause. However, by terminating a chain and forming a resolvent, we allow a breadth-first component to be available to the search. LELS allows all chain starting from the original start clause to interact after they are generated. In this sense, LELS method is like a cross between pure linear [6] and set-of-support resolution[12].
The cycle termination condition bears some relationship to the cancellation operation of OL resolution. A cycle termination condition occurs when OL cancellation is possible, but the LELS method form a new resolvent at the later point, that is, where the literal would have been canceled as opposed to where the framed literal had been introduced.

The problems we experiment with are taken from group theory, Boolean algebra, and Puzzle problems. They were chosen because they have received repeated attention in the literature [3][5][8][13]. The primary purpose of the experimentation was to determine, if possible, the conditions under which LELS would perform better than other hyper methods. A second purpose was to determine a better way for exploring the search space. A third purpose was to understand the relationship between the search space and LELS.

All of the experiments were performed on a Sun Work Station. The LELS-based automated theorem prover was implemented in C language[5]. The experimental results[5] seem to indicate that LELS explores the search space in a quite different way than other methods. LELS explores the search space according to the termination conditions, the free-literals on hand as well as the current merging situations. Moreover, there seems to be the potential for LELS to generate fewer obviously irrelevant clauses.

5. Conclusions

In this paper, we focused on the extended linking strategy with left merging for automated reasoning. We presented the detailed formalism, definition of the strategy. LELS is sound for any clause set. It is also proved that LELS is refutation complete for any ground sets. We also proved that LELS can be lifted to apply to the first order calculus. The use of LELS can have a dramatically positive effect on program performance, occasionally reducing the CPU time required to complete a proof by a large factor. Also, it reduced the search space for the program because a vast amount of rarely needed information is avoided (not generated) during the linking process. From the results of the test problems, we found that the proof steps and the search space are closely related to the clauses we selected to extend the linking process. LELS should be used with other heuristic strategies such as weighting and targeting to increase the efficiency of the automated theorem proving programs.

Acknowledgements

This research was supported in part by Taiwan NSF grant NSC 84-2213-E-014-007.