Extending the Regular Restriction of Resolution to Non-Linear Subdeductions

Bruce Spencer and J. D. Horton
University of New Brunswick
P.O. Box 4400, Fredericton, New Brunswick, Canada E3B 5A3
bspencer@unb.ca, jdh@unb.ca, http://www.cs.unb.ca

Abstract

A binary resolution proof, represented as a binary tree, is irregular if some atom is resolved away and reappears on the same branch. We develop an algorithm, linear in the size of the tree, which detects whether reordering the resolutions in a given proof will generate an irregular proof. If so, the given proof is not minimal. A deduction system that keeps only minimal proofs retains completeness. We report on an initial implementation.

Introduction

The regular restriction of resolution (Tseitin 1969) states that a resolution step resolving on a given literal should not be used to deduce a clause containing that literal. That is, both steps should not be in the same branch (or linear subdeduction) of the proof tree. We extend this restriction so that it applies steps to that are not on one branch, but can be made linear by rotating edges of the tree, as long as those rotations do not weaken the proof. If a proof cannot be made irregular by rotations, we call it *minimal.*

The first section below presents the regular restriction on binary resolution trees. Following that, minimal binary resolution trees are introduced. It is important to decide quickly if a given binary resolution tree is minimal. In the next section we give an algorithm that requires a linear number of set operations, and so is effectively linear in the size of the tree. Next, we report on our theorem prover that makes use of this and other restrictions, and give some preliminary timings.

Note that in this paper we consider resolution only on literals and not on formulae.

Binary Resolution Trees

We use standard definitions (Chang & Lee 1973) for atom, literal, substitution, unifier and most general unifier. In the following a **clause** is an unordered disjunction of literals[2]. The clause $c_1 \vee \ldots \vee c_m$ **subsumes** the clause $d_1 \vee \ldots \vee d_n$ if there exists a substitution θ such that for every $i = 1, \ldots, m$ there exists $j = 1, \ldots, n$ such that $c_i\theta = d_j$. The **resolvent** of two clauses $C_1 \vee a_1$ and $C_2 \vee \neg a_2$ is $(C_1 \vee C_2)\theta$ where θ is a unifier of a_1 and a_2. An atom a **occurs in** a clause C if either a or $\neg a$ is one of the disjuncts of the clause. The **merge** operation on a clause $C \vee a_1 \vee a_1$ produces $C \vee a_1$. The resolution operation is usually defined as the combination of building a resolvent followed by as much merging as possible. A **factoring operation** on a clause $C \vee a_1 \vee a_2$ produces $(C \vee a_1)\theta$, where θ unifies a_1 and a_2.

A binary resolution derivation is commonly represented by a binary tree, drawn with its root at the bottom. Each edge joins a **parent** node, drawn above the edge, to a **child** node, drawn below it. The **ancestors** (**descendants**) of a node are defined by the reflexive, transitive closure of the parent (child) relation.

Definition 1 *A* **binary resolution tree** *on a set S of input clauses is a labeled binary tree. Each node N in the tree is labeled by clause, called a* **clause label** *denoted $cl(N)$. Each node either has two parents and then its clause label is the resolvent of their clause labels after zero or more merges, or has no parents and is labeled by an input clause from S. In the case of a resolution, the substitution is applied to all labels in the tree. The clause label of the root of the binary resolution tree is called the* **result** *of the tree. A binary resolution tree is* **closed** *if its result is the empty clause, $\square$. For an internal (non-leaf) node, we define two more labels that are produced by the resolution between the parents. The* **atom label** *of an internal node N is the atom of the literals resolved upon, denoted $al(N)$. The* **merge label**, *written $ml(N)$, is the set of literals that were merged as part of this resolution.*

For the binary resolution tree in Figure 1 $S = \{\neg a, \neg b, \neg g, a \vee b \vee \neg c, c \vee \neg d, a \vee d, \neg a \vee b \vee \neg e, e \vee f \vee g\}$.

[2]We do not use set notation because we do not want multiple occurrences of a literal to collapse to a single literal automatically.

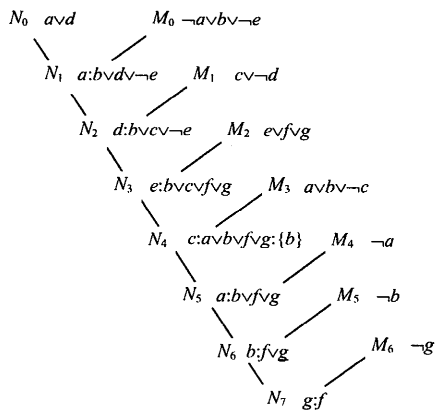

Figure 1: Part of a binary resolution tree.

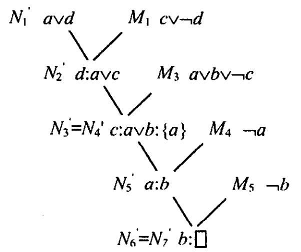

Figure 2: Surgery: Operation 3 on Figure 1.

The labels of a node N are displayed as $al(N) : cl(N) : ml(N)$. If the merge label is empty, it is not shown. The node N_4 has atom label c, clause label $a \vee b \vee f \vee g$ and merge label $\{b\}$. The order between the parents of a node is not defined.

Definition 2 (Regular) *A binary resolution tree T is* **regular** *if there does not exist a node N of T and a descendant M of N, such that $al(N)$ occurs in $cl(M)$.*

Operation 3 (Surgery on irregular trees) *Suppose N_1 and N_j are internal nodes in T such that $al(N_1) = a$ occurs in $cl(N_j)$, that N_n is the root of T, and for $0 \leq i < n$, N_i is one parent of N_{i+1}, and M_i is the other. Assume without loss of generality that a occurs in the clause label of no node in $(N_2, \ldots, N_{j-1})$. Let N_0 be the parent of N_1 chosen so that the occurrence of a in $cl(N_0)$ agrees in sign with the occurrence in $cl(N_j)$. Construct T' by constructing the path $(N_1', \ldots, N_n')$ of not necessarily distinct nodes as follows. Let $N_1' = N_0$, and consider all ancestors of N_0 to be in T'. For $i = 2, \ldots, n$ let N_i' be N_{i-1}' if $al(N_i)$ does not occur in $cl(N_{i-1}')$. Otherwise define a new node N_i' and make it the child of N_{i-1}' and M_{i-1}. Consider M_{i-1} and all of its ancestors as nodes of T'. Let the clause label, atom label and merge label of N_i' be defined by the resolution on $al(N_i)$ so that all merges that can be done at N_i' are done. Note that a new merge is done at N_j'.*

Figure 2 shows the effect of this operation on Figure 1. Surgery is performed between N_1 and N_4, and M_0, M_2 and M_6 are not needed in T'.

Lemma 4 *Suppose T is an irregular binary resolution tree on a set S of clauses and suppose T' is constructed by Operation 3. Then T' is also a binary resolution tree on S, T' is smaller than T and the result of T' subsumes the result of T.*

Proof. Each leaf in T' has the same label as a leaf in T and therefore T' is defined on S. Also, each internal node is defined by a resolution of its parents, so T' is a binary resolution tree. Note that $cl(N_1') = cl(N_0)$ subsumes $cl(N_1) \vee a$ because $cl(N_1)$ contains all the literals in $cl(N_0)$ except a. By a simple induction $cl(N_i')$ subsumes $cl(N_i) \vee a$ for $i = 2, \ldots, n$. If there is only one occurrence of a in $cl(N_j)$ and that is resolved upon when creating $cl(N_{j+1}')$ we know that a is not in $cl(N_{j+1}')$. Thus $cl(N_{j+1}')$ subsumes $cl(N_{j+1})$. Then $cl(N_i')$ subsumes $cl(N_i)$ for $i = j+1, \ldots, n$, so the result of T' subsumes that of T. Since M_0 is not in T' and since all other nodes in T' are taken at most once from T, it follows that T' has fewer nodes than T. □

Theorem 5 (Completeness (Tseitin 1969)) *If S is unsatisfiable there exists a closed regular binary resolution tree on S. Furthermore the smallest closed binary resolution tree is regular.*

Proof. If S is unsatisfiable, there exists a closed binary resolution tree (Robinson 1965). If it is irregular, apply Operation 3 repeatedly until it is regular. This process must terminate since the tree is smaller at each step. If the smallest closed binary resolution tree is not regular, surgery can be applied to it, making a smaller closed tree. □

Minimal Binary Resolution Trees

A **rotation** of an edge in a binary tree is a common operation, for example with AVL trees (Adelson-Velskii & Landis 1962). Before we apply it to binary resolution trees, we review the operation on binary trees. Given the binary tree fragment on the left of Figure 3, a rotation is the reassignment of edges so that the tree on the right of Figure 3 is produced. The parent C of E becomes the child of E and the parent B of C becomes the parent of E. If some node has E for a

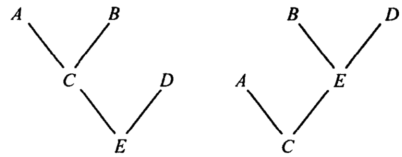

Figure 3: A binary tree rotation

parent, that node will now use C instead of E for this parent.

Operation 6 *Suppose T is a binary resolution tree with an edge (C, E) between internal nodes such that C is the parent of E and C has two parents A and B. Further, suppose $al(E)$ occurs in $cl(B)$ but not in $ml(C)$. Then the result of a* **rotation** *on this edge is the binary resolution tree T' defined by resolving $cl(B)$ and $cl(D)$ on $al(E)$ giving $cl(E)$ in T' and then resolving $cl(E)$ with $cl(A)$ on $al(C)$ giving $cl(C)$ in T'. Any merges in T of literals in $cl(A)$ and $cl(B)$ are done in T' at C. Likewise merges between $cl(A)$ and $cl(D)$ are done at C in T', and merges between $cl(B)$ and $cl(D)$ are done at E in T'. This defines the merge label and refines the clause label of E and C in T'. Furthermore, the child of E in T is the child of C in T'.*

A rotation changes the order of two resolutions. It also changes the clause labels and merge labels of C and E, but not their atom labels. Since the substitutions arising from a resolution are applied to all labels in the tree, the instances of atoms are not changed by a rotation. A rotation may introduce tautologies or duplication to clause labels of internal nodes. For instance, if $al(C)$ occurs in $cl(D)$ then $cl(E)$ in T' will be tautological or contain a duplicate literal. However the clause label of the root is not changed.

Definition 7 *A binary resolution tree T is* **minimal** *if no sequence of rotations of edges generates a tree T' that is irregular.*

Theorem 8 *If a binary resolution tree T on S is non-minimal, there exists a minimal binary resolution tree T' on S which is smaller than T and the result of T' subsumes the result of T.*

Proof. If T is not minimal, apply Operation 6 and Operation 3 so that a regular tree is produced. If this tree is minimal then let T' be this tree. Otherwise repeat from the beginning until T' is defined. This process must terminate because the tree is getting smaller at each application of Operation 3. Also the old result is subsumed by the new result at each step. □

Thus a smallest binary resolution tree is minimal. Goerdt has shown (Goerdt 1993) that a smallest regular binary resolution directed acyclic graph (DAG) may be exponentially larger than an irregular binary resolution DAG. Thus in some cases regularity, and hence minimality, will slow a theorem prover. However, there are also cases where minimality reduces the search.

Checking Minimality

Determining whether a given binary resolution tree is minimal seems to be labourious, since the straightforward application of the definition, as is done in the proof of Theorem 8, checks every possible sequence of rotations, and there can be exponentially many. In this section we define the notion visibility for binary resolution trees, first defined for clause trees (Horton & Spencer 1997). We also give a linear algorithm for deciding whether two minimal binary resolution trees can be combined to give a minimal tree.

Definition 9 (History Path) *A* **history path** *P for an atom a in a binary resolution tree T is a sequence $(N_0, \ldots, N_n)$ of nodes such that N_0 is a leaf, each N_i is the parent of N_{i+1} for $i = 1, \ldots, n-1$ and a occurs in the clause label of each node. The* **head** *of P is N_n. We say that P* **closes** *at N_{n+1} if N_n is the parent of N_{n+1} and $a = al(N_{n+1})$.*

For example in Figure 1, (M_1, N_2, N_3) is a history path for c which closes at N_4. Note that if there are multiple occurences of a in the clause label of some node N_i, they are on separate history paths. Also a rotation does not change the nodes at which history paths close, although any node on the path, except the leaf, may be changed. Thus a history path P is identified by its leaf and its closing node, so after a rotation P', the **image** of P, is the path with the same leaf and closing node as P.

Definition 10 (Precedes) *A history path P* **directly precedes** *a history path Q if P and Q have no nodes in common, and P closes at some node in Q. We write $P \prec Q$. Moreover we say P* **precedes** *Q, and write $P \prec^* Q$ if there is a sequence of history paths $(P_1, \ldots, P_k)$ with $P = P_1$ and $Q = P_k$ and P_i directly precedes P_{i+1} for $i = 1, \ldots, k-1$.*

The relation precedes is the reflexive and transitive closure of directly precedes. In particular a history path precedes itself, even though it does not directly precede itself. Also note that precedes defines a partial order on the set of history paths.

In most cases a rotation does not change the precedes relation on history paths.

Lemma 11 *Let the history path P precede the history path Q in the binary resolution tree T and suppose that P', Q' and T' are the images of P, Q and T respectively after a rotation of the edge CE as in Definition 6. Further suppose that the head of Q is not C. Then P' precedes Q' in T'.*

Proof. If $P = Q$ then $P' = Q'$ so $P' \prec^* Q'$. Assume $P \neq Q$. Let M be the node at which P closes, and N be the head of Q. Note that M must be an ancestor of N, and consider the path $path(M, N)$ with tail M and head N. There exist paths $P_1, \ldots, P_n$ such that $P = P_1 \prec \ldots \prec P_n = Q$ and these paths close at distinct nodes on $path(M, N)$. Thus for a given node on this path there is a unique $i, 1 \leq i \leq n$ such that the node is on P_i. We consider where the edge CE occurs in relation to $path(M, N)$.

(Case 1) If neither C nor E is on $path(M, N)$ then the rotation has not affected this path and P' precedes Q' in T' as in T.

(Case 2) Suppose C is on $path(M, N)$. Recall that we have eliminated the case where $N = C$, so E is also on $path(M, N)$. Let P_i be the history path on which

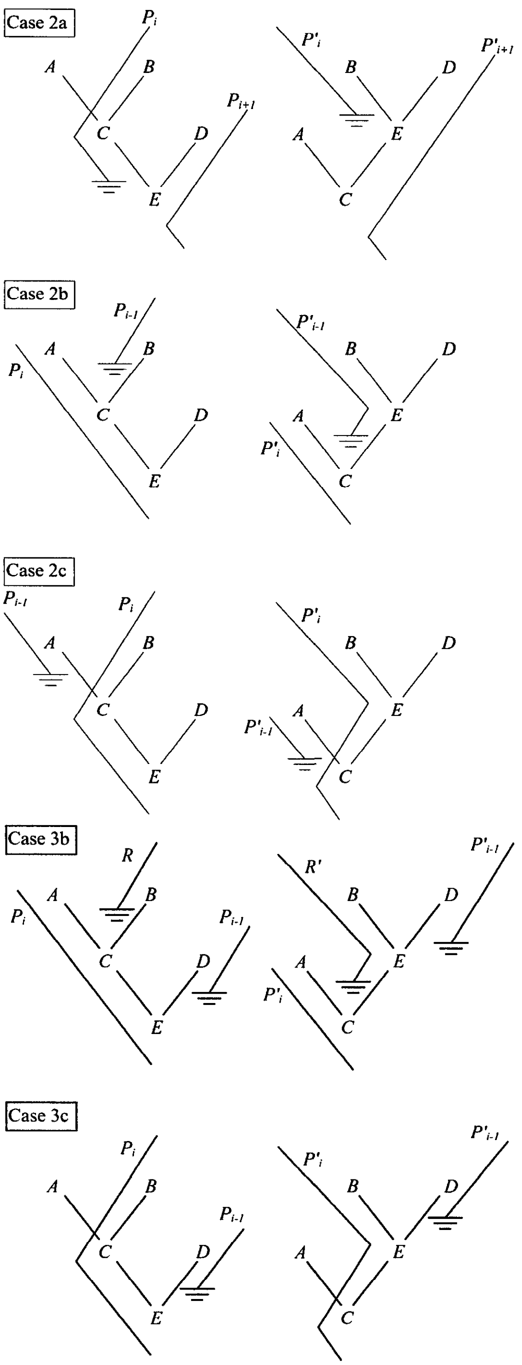

Figure 4: Cases of Lemma 11

C occurs. If the head of P_i is C and P_i contains A, then the rotation is impossible, since no history path through A may close at E. The remaining subcases are illustrated in Figure 4, where the ground symbol (three lines) on a history path indicates the edge between the head of the path and where the path closes.

(2a) Suppose the head of P_i is C and P_i contains B. Then P_{i+1} contains D and E, and after the rotation P'_{i+1} contains D, E, C and closes below C. Thus $P'_i \prec P'_{i+1}$. Also $P_1 \prec^* P_i$ and $P'_{i+1} \prec^* P'_n$ since these paths have not changed. Thus $P'_1 \prec^* P'_n$.

(2b) The head of P_i is below C and P_i contains A. If $M = C$ or M is an ancestor of B then after the rotation $P'_1 \prec^* P'_{i-1}$ and $P'_i \prec P'_n$ because these parts of the tree have not changed. Also $P'_{i-1} \prec P'_i$ since P'_{i-1} closes at C and is disjoint from P'_i. Thus $P'_1 \prec^* P'_n$. On the other hand if M is an ancestor of A the one change to $path(M, N)$ is that P'_i is one edge shorter than P_i, but no head of any P_i is different, so $P_1 \prec^* P_n$.

(2c) The head of P_i is below C and P_i contains B. If $M = C$ or M is an ancestor of A then after the rotation $P'_1 \prec^* P'_{i-1}$ and $P'_i \prec P'_n$ because these parts of the tree have not changed. Also $P'_{i-1} \prec P'_i$. Thus $P'_1 \prec^* P'_n$. On the other hand if M is an ancestor of B then no head of any P_i is different, so $P_1 \prec^* P_n$.

(Case 3) Suppose C is not on $path(M, N)$ but E is. Let P_i be the history path on which E occurs. Note that the rotation has not changed the heads of history paths below E, so $P'_i \prec^* P'_n$. Thus we need only show that $P'_1 \prec^* P'_i$.

(3a) If D is on P_i then after the rotation, D, E and C are on P'_i. Since the heads of history paths below E have not changed, $P'_1 \prec P'_i$.

(3b) If D is on P_{i-1} while A, C and E are on P_i then after the rotation the heads of history paths above D have not changed so $P'_1 \prec^* P'_{i-1}$. There must be a history path R with head B closing at C. After the rotation its head is E, so $P'_{i-1} \prec R' \prec P'_i$.

(3c) If D is on P_{i-1} while B, C and E are on P_i then after the rotation the heads of history paths above D have not changed so $P'_1 \prec^* P'_{i-1}$. Also $P'_{i-1} \prec P'_i$. □

Definition 12 (Hold) *An unordered pair $\{P, Q\}$ of history paths* **holds** *an internal node M of a binary resolution tree if there exist history paths P_1 and Q_1 such that $P_1 \prec^* P, Q_1 \prec^* Q$ and M is the first node that occurs on both P_1 and Q_1, that is, the parent of M does not occur on both. A node N* **holds** *M if history paths P and Q hold M and they both close at N.*

Definition 13 (Visible) *In a given binary resolution tree with internal nodes N and M, we say that M is* **visible** *from N, and that N can* **see** *M, if there exists a sequence of rotations such that M is a descendant of N. Otherwise M is invisible from N.*

Theorem 14 *The nearest common descendant of M and N holds M if and only if M is invisible from N.*

Proof. We show that if the nearest common descendant of M and N holds M, then after a rotation, the nearest common descendant of M and N holds M. Note that the nearest common descendant may be changed by the rotation. Thus M can never be a descendant of N for if it were then the nearest common descendant would be M, and a node cannot hold itself.

Let F be the nearest common descendant of N and M, and let the rotated edge CE, and nodes A, B and D adjacent to it, be as defined in Operation 6. Let P and Q hold M and close at F, while $P_1 \prec^* P$ and $Q_1 \prec^* Q$ are the paths for which M is the highest common node. Consider the case where $F \neq E$, so that after the rotation F is still the nearest common descendant of M and N. By Lemma 11, $P_1' \prec^* P'$ and $Q_1' \prec^* Q'$. Suppose $M \neq E$. Then after the rotation, M is still the first common node on P_1' and Q_1', so F still holds M. Now suppose that $M = E$. Without loss of generality assume that P_1 contains C and Q_1 contains D. If P_1 contains B then after the rotation, P_1' and Q_1' still hold E, so F holds M. If P_1 contains A then consider the path R containing B and closing at C. After the rotation, $R' \prec P_1'$, so that R' and Q_1' hold E, so again F holds M.

Now suppose that $F = E$. Consider the case where M is an ancestor of C and N is an ancestor of D. Since no history path can contain A and C and close at E, $M \neq C$. For the same reason, P and Q contain B and close at E. If M is an ancestor of A then the paths that directly precede P and Q close at C and hold M. Thus C holds M. After the rotation the nearest common descendant of M and N is C, and C still holds M. Otherwise if M is an ancestor of B then after the rotation the nearest common descendant of M and N is E and E still holds M. Finally consider the case where M is an ancestor of D. If N is an ancestor of B then after the rotation, E still holds M and is still the nearest common descendant of N and M. If N is C or is an ancestor of A, then consider path R with head at B which closes at C. After the rotation, the nearest common descendant of M and N is C, while R' directly precedes both P' and Q'. Thus C holds M after the rotation. □

Note that the proof of the converse of Theorem 14, which was omitted for lack of space, constructs a sequence of rotations so that a non-minimal tree becomes irregular, thus allowing surgery to be applied. We leave the implementation of surgery to future work. Now we turn our attention to a theorem prover that keeps only minimal binary resolution trees.

Definition 15 *Let T be a binary resolution tree. Then $al(T) = \{al(N) | N$ is a node of $T\}$ is called the* **set of atoms** *of T. A* **subbrt** *of T is a binary resolution tree rooted at some node other than the root of T. For a subbrt T' of T, $vis(T') = \{al(N) | N$ is visible from the root of $T\}$ is called the set of* **visible atoms of** T'.

Theorem 16 *Let binary resolution tree T consist of a root node R and two subbrt's T_1 and T_2. T is minimal if and only if*

1. *T_1 and T_2 are minimal;*
2. *no atom in $cl(R)$ is in $al(T_1) \cup al(T_2)$;*
3. *$al(T_1) \cap vis(T_2) = \phi$; and*
4. *$al(T_2) \cap vis(T_1) = \phi$.*

Proof. Assume that T is minimal. If T_1 or T_2 were not minimal, then there would be a sequence of edge-rotations which would make the subbrt irregular. The same sequence performed on T would make T irregular as well. Hence the first condition is true. If the second condition were false, then T would be irregular immediately. Assume that the third condition is false. Then there are two nodes, $N \in T_1$ and $M \in T_2$ whose atom labels are the same, and M is visible from R. Hence M can be rotated below R, and so below N, making T irregular. The fourth condition is symmetric.

Conversely, assume that T is not minimal. Then there is a sequence of rotations that create an irregular tree T'. Node N has a descendant M_1 in T' such that $al(N)$ occurs in $cl(M_1)$. Since the rotations do not change $cl(R)$, if $al(N)$ occurs in the result of T', it occurs in $cl(R)$ in T and then T violates the second condition. Thus $al(N)$ does not occur in $cl(R)$ so there is a descendant M of N in T such that $al(N) = al(M)$. If M and N are in the same T_i then T_i violates the first condition. Assume M and N are in different T_i. Since M has been rotated below N, M is visible from N in T, and by Theorem 14 M is not held by the nearest common descendant R of M and N. Thus M is visible from R. Therefore $al(M)$ is in $vis(T_i)$ while $al(N)$ is in $al(T_{3-i})$. □

Since our theorem prover keeps only minimal binary resolution trees, the first condition in Theorem 16 is already satisfied for any newly constructed binary resolution tree. It is easy to check that the new result does not contain an atom in $al(T_1) \cup al(T_2)$. What is left is to find is an easy way to calculate those atoms which are visible in each subbrt T_i. This condition for this is given by Theorem 14, and computed by Procedure 17, by calling $vis(T_i, \{al(R)\})$.

The idea in Procedure 17 is that a node is visible in a subbrt if and only if it is not held by paths that close at the root. So we need to calculate for each node N in T_i the history paths P_N going through N that precede history paths that close at the root. If some of these paths go through one parent of N, and some go through the other, then N is held by the root; otherwise N is visible from the root. We use sets of atoms to represent history paths. Thus the history paths going through a parent N of the root and closing at the root R are represented by $P_N = \{al(R)\}$. As we go from a node N to its parent A, to calculate the paths through A, we first remove any paths that do not go through A. This is done by intersecting the atoms of P_N with the atoms of $cl(A)$. Then we add $al(N)$ to P_A if there is some path in P_N that does not go through A, because then that path precedes paths in P_N, and thus precedes paths that close at the root.

Procedure 17 (Visibility) *Given a node N in a binary resolution tree and P_N a set of atoms representing history paths that precede history paths that close at the root of the tree, $vis(N, P_N)$ returns the atoms at and above N visible from the root.*
If (N is a leaf) return ϕ;
Let A and B be the parents of N, chosen so that $P_N \cap atom(cl(B)) \neq \phi$
$P_A = P \cap atom(cl(A))$;
$P_B = P \cap atom(cl(B))$;
If ($P_A \neq \phi$)
  //N is held
  return $vis(A, P_A \cup \{al(N)\}) \cup vis(B, P_B \cup \{al(N)\})$;
else ($P_A = \phi$)
  //N is not held, so it is visible
  return $\{al(N)\} \cup vis(A, \{al(N)\}) \cup vis(B, P_B)$;

Procedure 17 runs in a number of intersection calls which is proportional to the number of nodes in the tree. With hashing, these operations can in principle be performed in time proportional to the size of the clauses. Hence *vis* is a linear time algorithm, which is as fast as one could expect.

Implementations of BRTs

We have implemented a prototype theorem prover for propositional logic. It resembles OTTER(McCune 1994), but it retains only minimal binary resolution trees (so that the recursive calls in Theorem 16 are not needed), whereas the proofs built by OTTER correspond to non-minimal trees in some cases. We have combined the minimal restriction with an ordering restriction, different from those in (Kowalski & Hayes 1969), but our restriction has an additional feature: a given minimal binary resolution tree will be found exactly once. These two additional restrictions address the problem of redundancy in this type of theorem prover(Wos 1988). Finally we have defined a new type of subsumption that retains completeness when combined with the minimality restriction. Ordinary ubsumption combined with minimality is not complete.

We measured the number of clauses built by OTTER that were not tautologies, and the number of binary resolution trees built by the prototype that were minimal. The results are incomplete, but encouraging. For instance, OTTER accepts 10091 clauses to refute SYN094-1.005 from TPTP (Sutcliffe, Suttner, & Yemenis 1994), whereas the prototype allows only 359 minimal binary resolution trees. OTTER needs 35820 inferences for the four-pigeons problem, MSC007-1.004, while the prototype needs 577. In a minority of our experiments OTTER needed fewer inferences because the more restricted search space of the prototype did not contain the proof that OTTER found. OTTER's wins, so far, have not been as big.

Conclusion

The space of minimal binary resolution trees is interesting for three reasons: (1) it is refutationally complete, (2) it contains the smallest binary resolution tree and (3) non-minimal (sub)trees can be identified quickly. We define the novel notion of visibility between nodes in a binary resolution tree, and show that it is useful. We present an efficient algorithm to determine minimality, which uses a number of set operations that is linear in the size of the tree. We have implemented a theorem prover using this restriction, and it compares favorably to OTTER using binary resolution. We are continuing the implementation effort into first order logic.

Acknowledgments

The authors are grateful to NSERC for funding and to Joel Burrows for work on the prototype. The first author did some of this work in T.H. Darmstadt.

References

Adelson-Velskii, G. M., and Landis, E. M. 1962. An algorithm for the organizaton of information. *Soviet Math. Doklady* 3:1259–1263.

Chang, C.-L., and Lee, R. C.-T. 1973. *Symbolic Logic and Mechanical Theorem Proving.* New York and London: Academic Press.

Goerdt, A. 1993. Regular resolution versus unrestricted resolution. *SIAM Journal on Computing* 22:661–683.

Horton, J. D., and Spencer, B. 1997. Clause trees: a tool for understanding and implementing resolution in automated reasoning. *Artificial Intelligence* approximately 62 pages. Forthcoming 1997, available as http://www.cs.unb.ca/profs/bspencer/htm/clause_trees/TR95-95.ps.

Kowalski, R., and Hayes, P. 1969. Semantic trees in automated theorem proving. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4.* American Elsevier Publishing Company, Inc.

McCune, W. W. 1994. Otter 3.0 users guide. Technical Report ANL-94/6, Mathematics and Computer Science Division, Argonne National Laboratories, Argonne, IL.

Robinson, J. A. 1965. A machine-oriented logic based on the resolution principle. *J. ACM* 12:23–41.

Sutcliffe, G.; Suttner, C.; and Yemenis, T. 1994. The TPTP problem library. In Kapur, D., ed., *Automated Deduction CADE-12*, number 814 in Lecture Notes in Artificial Intelligence, 252–266. Springer-Verlag, Berlin.

Tseitin, G. S. 1969. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics*, Seminars in Mathematics: Mathematicheskii Institute, 115–125. Consultants Bureau.

Wos, L. 1988. *Automated Reasoning : 33 Basic Research Problems.* Englewood Cliffs, New Jersey: Prentice-Hall.