DELETION-DIRECTED SEARCH IN RESOLUTION-BASED PROOF PROCEDURES

David Gelperin*

Department of Computer and Information Science**
The Ohio State University
Columbus, Ohio

Abstract

The operation of a deletion-directed search strategy for resolution-based proof procedures is discussed. The strategy attempts to determine the satisfiability of a set of input clauses while at the same time minimizing the cardinality of the set of retained clauses. E-representation, a new clause deletion rule which is fundamental to the operation of the search strategy, is also described.

Descriptive terms

Formal logic, automatic theorem proving, resolution, heuristic search, clause deletion, E-representation, deletion-directed search, mate tables.

1. Introduction

This report describes some new techniques which can be used by resolution-based proof procedures. Since all subsequent references will be to resolution-based procedures, the adjective will be understood. In addition, familiarity with the terminology and results of J. A. Robinson[1] is assumed.

In order to provide a framework for discussion, both a structural and an operational description of a proof procedure will be considered. From a structural viewpoint, & proof procedure can be denoted by a triple <T,A,z) where T is a finite, non-empty set of clause generation rules, A is a finite (possibly empty) set of clause deletion rules, and *Z* is a search strategy i.e. a procedure for applying the rules in TUfi. A clause generation rule, e.g. clash resolution[2], specifies the conditions constraining the clauses of an admissible resolution. A clause deletion rule, e.g. subsumption deletion, specifies the conditions under which a clause may be eliminated without affecting the unsatisflability of a set of clauses. A search strategy, e.g. diagonal search[3], sequences the generation and deletion of clauses as a proof procedure attempts to determine the satisfiability of an input clause set. Within a search strategy, clause generation is controlled by a generation strategy and clause elimination by a deletion strategy.

Turning to an operational viewpoint, a proof precedure can be characterized as a mapping which associates a finite set of clauses (i.e. the input set) with a non-empty sequence of clauses and pointers, called a trace. A trace begins with some ordered occurrence of the input set. This is followed by an ordered set of generated clauses and pointers to deleted clauses. All of the parents of a generated clause must precede it in the trace and all deleted clauses must precede their respective pointers. Associated with each non-input trace element is a set of retained clauses which is composed of all of the clauses in the corresponding partial trace which have not been deleted i.e. which do not have corresponding pointers in the partial trace. A trace records the operation of a proof procedure and in particular its search strategy on a specific problem.

The search space associated with a set of input clauses and a proof procedure <T,A,I> can be represented by a labelled tree in which the input set labels the root, retained sets label the other nodes and elements of TUA label the edges. A trace represents the particular path selected by £ in the search tree. Note that the elements of TUA are not well-defined as operators on clause sets but can be viewed as operator schemata.

Most of the work dealing with resolution has concentrated on the problem of demonstrating the unsatisflability of an unsatisfiable set of input clauses. The problem of demonstrating the satisfiability of a satisfiable input set has been largely ignored. This is probably due, in part, to the fact that no procedure can identify all satisfiable sets. However, many solvable cases of the decision problem have been identified[4] and the relationship between these cases and resolution-based proof procedures has not been explored. When considering applications utilizing resolution-based proof procedures, such as question-answering systems[5] or robot planning systems[6], situations involving the absence of sufficient information make the recognition of satisfiable sets an area of interest. Satisfiability is indicated by the generation of the empty set of clauses and its demonstration results from the utilization of clause deletion rules. A proof procedure must utilize such rules in order to maximize the domain of formulas that it can classify. While deletion rules are theoretically unnecessary in the domain of unsatisfiable formulas, in actual practice their use can help to increase the effectiveness of a proof search. Such rules can be used to reduce the number of candidate resolutions while at the same time preserving completeness. Careful consideration is necessary, however, since their indiscriminate use may not be cost-effective and can result in a loss of completeness .

In the next section, a new clause deletion rule is introduced and an example in section 3 shows that this rule allows a demonstration of satisfiability not previously possible. The main topic of section 3 is the operation of a deletion-directed search strategy which is built around the new deletion rule and attempts to minimize the cardinality of retained sets.

2. E-representation

In order to motivate the formal presentation below, consider the unsatisfiable set S - {AB, AB, AC, BC, C}. Notice that each binary resolution which involves the literal occurrence A in AB produces a clause which is already in S. This observation, viewed as a generalization of the notion

* Current address: Computer Science Department, Technion, Haifa, Israel.

** This research was supported by National Science Foundation Grant GN-534.1

of a pure literal[1], suggests that AB can be deleted from S without affecting unsatisfiability. A new deletion rule, called E-representation, validates this conjecture and the definitions which follow formalize the observation which produced it.

Associated with every ordered pair $\langle C,D\rangle$ of clauses is a finite set (possibly empty) of ordered triples $\langle L,M,N\rangle$ where L, M, and N are all non-empty finite sets of literals. These ordered triples are called key triples of $\langle C,D\rangle$ and their component sets satisfy the following properties: (1) $L \subseteq C$ (2) $M \subseteq D$ (3) $N = L\xi_C \cup \bar{M}\eta_D$ is unifiable with most general unifier σ_N where ξ_C and η_D are the x-standardization and y-standarization of C and D respectively. A resolvent of C and D is any clause of the form: $(C-L)\xi_C\sigma_N \cup (D-M)\eta_D\sigma_N$ where $\langle L,M,N\rangle$ is a key triple of $\langle C,D\rangle$. As an example, let $C = \bar{P}aPxQxgx$ and $D = PaPfx$ be two parent clauses. Their resolvents are $QagaPfy_1$, $\bar{P}x_1Qx_1gx_1Pfy_1$, and $\bar{P}aQfy_1gfy_1Pa$ with corresponding key triples $\langle\{\bar{P}a,\bar{P}x\}, \{Pa\}, \{\bar{P}a,\bar{P}x_1\}\rangle$, $\langle\{\bar{P}a\}, \{Pa\}, \{\bar{P}a\}\rangle$ and $\langle\{\bar{P}x\}, \{Pfx\}, \{\bar{P}x_1,Pfy_1\}\rangle$.

The notion of a represented key triple generalizes the relationship between a key triple and its *associated resolvent* to include *any* clause which subsumes that resolvent. Let C and D be two clauses such that $T = \langle L,M,N\rangle$ is a key triple of $\langle C,D\rangle$ and $R = (C-L)\xi_C\sigma_N \cup (D-M)\eta_D\sigma_N$ is the resolvent of C and D corresponding to T. If R is subsumed by a clause in a set S of clauses, then T of $\langle C,D\rangle$ is represented in S.

An occurrence of a literal is termed exhausted when a particular set of triples is represented. Let S be a set of clauses. Let E be a literal in a clause C. Let P be the set of all key triples T of $\langle C,D\rangle$, where D is any clause in S-{C}, which satisfy the following conditions: (1) T has the form $\langle\{E\},M,N\rangle$ and (2) the most general unifier σ_N associated with T is such that $D\eta_D\sigma_N$ is not a tautology. If all key triples of P associated with non-tautologous resolvents are represented in S-{C}, then the occurrence of E in C is exhausted in S. Informally, the idea behind exhaustion is that all of the necessary resolvents corresponding to a particular literal occurrence already appear in the current set of clauses i.e. are represented. A clause C is E-represented in a set S of clauses iff some literal in C is exhausted in S. In other words, if (almost) all of the non-tautologous resolvents that can be generated from C by only cancelling instances of E are subsumed in S-{C}, then E is exhausted and C is E-represented. The 'almost' results from the fact that a non-tautologous resolvent of C and D which involves a tautologous unification instance of D need not be considered.

To clarify the above definitions, consider the set $S = \{\bar{P}x\bar{Q}yRxy, \bar{R}xaSx, \bar{P}xSx, Px\bar{S}x\}$. The key triple $T = \langle\{Rxy\}, \{\bar{R}xa\}, \{Rx_1x_2, Ry_1a\}\rangle$ of $\langle\bar{P}x\bar{Q}yRxy, \bar{R}xaSx\rangle$ is represented in S because $\bar{P}xSx$ subsumes the corresponding resolvent $\bar{P}x_1\bar{Q}aSx_1$. Rxy is exhausted in S because the only corresponding non-tautologous resolvent $\bar{P}x_1\bar{Q}aSx_1$, is represented in S-{$\bar{P}x\bar{Q}yRxy$}. Analogously, $\bar{R}xa$ is exhausted in S. $\bar{Q}y$ is exhausted in S because all of its corresponding non-tautologous resolvents, i.e. none (it is pure), are represented in S. Both literals in $\bar{P}xSx$ are exhausted in S since neither has a corresponding non-tautologous resolvent. The clauses $\bar{P}x\bar{Q}yRxy$, $\bar{R}xaSx$ and $\bar{P}xSx$ are each E-represented in S because each contains at least one of the above mentioned exhausted literals. $Px\bar{S}x$ is not E-represented in S.

The concept of an E-represented clause *is* utilized in the E-representation TheoremB which states: If S is a set of clauses and C is a clause which is E-represented in S, then S is satisfiable iff S-{C} is satisfiable. This theorem implies that if any literal in a clause is exhausted, then the whole clause may be deleted without affecting unsatieflability.

In the next section, a search strategy is described which is built around the notion of exhaustion. The strategy operates by performing those resolutions which cause some nearly exhausted literal occurrence to become exhausted, i.e. the strategy generates E-represented clauses.

3. Deletion-Directed Search

Consider the following unsatisfiable set of clauses which resulted from an attempt to prove the proposition: If a diagonal of a trapezoid bisects a lower base angle, then the corresponding upper inscribed triangle is Isosceles.

1. Tabcd
2. Eabddbc
3. Iabd
4. TwxyzPwzxy
5. $\bar{P}$wxyzEwxyxyz
6. $\bar{E}$xyzxzyIxyz
7. $\bar{E}$rstxyz$\bar{E}$uvwxyz$\bar{E}$rstuvw
8. $\bar{E}$uvwxyz$\bar{E}$uvwuvw (basic factor of 7)

Let simple binary resolution denote a generation rule with the constraint that only one literal occurrence from each parent can be selected for a unification set, i.e. the first two components of all key triples are unit sets. Let a basic factor of a clause C be any clause C6 where O is an mgu of exactly two literals In C. A proof procedure utilizing basic factoring, simple binary resolution, E-representation deletion, and deletion-directed search generated the following modified trace:

9. Padbc $\langle 1,4\rangle$

Delete 1 and A by E-rep.

10. Eabdadb $\langle 3,6\rangle$

Delete 3 and 6 by E-rep.

11. Eadbdbc $\langle 5,9\rangle$

Delete 5 and 9 by E-rep.

12. EabdxyzEadbxyz $\langle 7,10\rangle$

Delete 10 by E-rep.

At this point, the set of retained clauses contains

2. Eabddbc
7. $\bar{E}$rstxyz$\bar{E}$uvwxyzErstuvw
8. $\bar{E}$uvwxyzEuvwuvw
11. Eadbdbc
12. EabdxyzEadbxyz

Continuation of the procedure yields

13. $\bar{E}$adbdbc $\langle 2,12\rangle$
14. □ $\langle 11,13\rangle$

The principle which guides deletion-directed search can be stated as follows: Given a set of clauses to which no deletion rule can be applied, attempt to derive another set of clauses by adding as few resolvents as possible such that at least one of the original clauses can be deleted from the new set. The following description of the search strategy assumes that the associated generation rules are basic factoring and simple binary resolution. Given an input set, search is initiated by building a data

structure called a mate table (cf. classification trees[9]). Two literals L_1 and L_2 are mates iff there exist substitutions θ_1 and θ_2 such that $L_1\theta_1$ $L2\theta2$ *are* complementary. A mate table is constructed by attempting to resolve each literal occurrence L appearing in an input clause C with every other literal occurrence in the input set. The result is a set of lists such that every literal occurrence has an associated, list. Each list has two types of entries depending upon whether the literal occurrence resolving with L i.e. L's mate, is in C or outside of C. An Outside entry is created when all of the following conditions hold: (1) C resolves on L with some clause D = C (2) Neither the resolvent of C and D nor the associated instance of D is a tautology (3) The resolvent is not subsumed by D. The entry identifies a mate literal in D and contains the corresponding" resolvent. An inside entry, on the other hand, just identifies one of L's mate literals within C but doesn't contain the corresponding resolvent.

The pointer portion of the initial mate table for the last example is represented below.

1.1	2.1	3.1	4.1	4.2	5.1	5.2
4.1	7.1	6.2	1.1	5.1	4.2	6.1
	7.2					7.1
	8.1					7.2
						8.1

6.1	6.2	7.1	7.2	7.3	8.1	8.2
5.2	3.1	2.1	2.1	6.1	2.1	6.1
7.3		5.2	5.2	8.1	5.2	7.1
8.2		8.2	7.3	7.1	7.3	8.1
		7.3		7.2	8.2	

The *proof procedure examines the table looking* for Indications of a double parent elimination possibility and finds that the first literal in clause 4 is the only candidate for resolution with the first literal in clause 1 and vice versa. By adding the corresponding resolvent which is already stored in the table to the retained set, a new set can be created in which both parent clauses contain an exhausted literal. Since a double elimination is the best result that can be planned, the corresponding resolvent is added. After deletion, the mate table is updated and becomes

2.1	3.1	5.1	5.2	6.1	6.2
7.1	6.2	9.1	6.1	5.2	3.1
7.2			7.1	7.3	
8.1			7.2	8.2	
			8.1		

7.1	7.2	7.3	8.1	8.2	9.1
2.1	2.1	6.1	2.1	6.1	5.1
5.2	5.2	8.1	5.2	7.1	
8.2	7.3	7.1	7.3	8.1	
7.3		7.2	8.2		

Two more double eliminations are detected and the corresponding resolvents are added to the retained set. After the addition of clause 11, the table becomes

2.1	7.1	7.2	7.3	8.1	8.2	10.1	11.1
7.1	2.1	2.1	8.1	2.1	7.1	7.3	7.1
7.2	8.2	11.1	10.1	7.3	8.1		7.2
8.1	11.1	7.3	7.1	11.1			8.1
	7.3		7.2	8.2			

At this point no double eliminations are predictable, but an immediate single elimination results from adding the resolvent of clauses 7 and 10 to the retained set and leads to

2.1	7.1	7.2	7.3	8.1	8.2	11.1	12.1	12.2
7.1	2.1	2.1	8.1	2.1	7.1	7.1	2.1	7.3
7.2	8.2	11.1	12.1	7.3	12.1	7.2	7.3	8.2
8.1	11.1	7.3	12.2	11.1	12.2	8.1	8.2	11.1
12.1	7.3		7.1	8.2	8.1	12.2		
			7.2					

The table now indicates that a minimum of three resolvents must be added to the retained set before a deletion by E-rep is possible. The procedure arbitrarily selects the first literal in clause 12 and adds the shortest resolvent corresponding to a mate i.e. the resolvent of clause 2. The immediate testing of all newly added unit clauses (the end test) yields the null clause.

In the realm of satisfiable sets, consider the following problem which results when AT(x,under-bananas,s_{01}) is omitted from a formulation of the monkey and bananas problem due to C. Green[10].

1. $\overline{\text{HAS}}$(monkey,bananas,x)
2. MOVABLE(box)
3. AT(box,place,s_0)
4. CLIMBABLE(monkey,box,x)
5. $\overline{\text{REACHABLE}}$(x,y,z)HAS(x,y,reach(x,y,z))
6. $\overline{\text{AT}}$(x,y,z)$\overline{\text{CLIMBABLE}}$(w,x,z)AT(x,y,climb(w,x,z))
7. $\overline{\text{AT}}$(x,y,z)$\overline{\text{CLIMBABLE}}$(w,x,z)ON(w,x,climb(w,x,z))
8. $\overline{\text{AT}}$(box,under-bananas,x)$\overline{\text{ON}}$(monkey,box,x)
   REACHABLE(monkey,bananas,x)
9. $\overline{\text{AT}}$(w,x,z)$\overline{\text{MOVABLE}}$(w)AT(skl(w,x,y,z),y,z)
   AT(w,y,move(monkey,w,y,z))
10. $\overline{\text{AT}}$(w,x,z)$\overline{\text{MOVABLE}}$(w)AT(skl(w,x,y,z),y,z)
    AT(monkey,y,move(monkey,w,y,z))

A proof procedure using basic factoring, simple binary resolution, E~representation deletion, subsumption deletion, and deletion-directed search produced the following modified trace. All of the clauses generated while processing the mate table do not appear in the trace.

11. $\overline{\text{REACHABLE}}$(monkey,bananas,z) ⟨1,5⟩

Delete (1,5)

12. $\overline{\text{AT}}$(box,under-bananas,x) ⟨8,11⟩
    $\overline{\text{ON}}$(monkey,box,x)

Delete (8,11)

13. $\overline{\text{AT}}$(box,y,z) ⟨7,12⟩
    $\overline{\text{CLIMBABLE}}$(monkey,box,z)
    $\overline{\text{AT}}$(box,under-bananas,climb(monkey,box,z))

Delete (7,12)

14. $\overline{\text{AT}}$(box,y,z)AT(box,y,climb(monkey,box,z)) ⟨4,6⟩

Delete (6)

15. $\overline{\text{AT}}$(box,y,z) ⟨4,13⟩
    AT(box,under-bananas,climb(monkey,box,z))

Delete (4,13)

16. $\overline{\text{AT}}$(box,under-bananas,z)$\overline{\text{AT}}$(box,y,z) ⟨14,15⟩

Delete (15)

17. $\overline{\text{AT}}$(box,under-bananas,z) basic factor of 16

Delete (16)

18. $\overline{AT}(box,x,z)AT(skl(box,x,y,z),y,z)$ $\langle 2,9 \rangle$
$AT(box,y,move(monkey,box,y,z))$

Delete (9); (10); (2); (18); (14); (3); (17)

The satisfiability of the input set has been demonstrated since the retained set is empty. The notation following clause 18 means that the deletion of clause 9 causes a literal in clause 10 to be exhausted and therefore permits the deletion of that clause. In the same way, the deletion of clause 10 permits the deletion of clause 2, etc.

Although not illustrated by either of the examples above, deletion-directed search must be constrained by a level bound. As in the case of the unit preference strategy11 , a level bound must be imposed in order to avoid an infinite depth-first search.

Now consider whether deletion-directed search can always find a literal occurrence to exhaust. Can resolvents always be added to a set of clauses so that some original clause will be E-represented? While the answer is "no", e.g. {Pxpfx, PxPgx}, all of the counter-examples contain neither an all positive nor an all negative clause and are therefore easily recognized as satisfiable sets. For all other sets, which includes all unsatisfiable sets, some identifiable literal occurrence is exhaustible and the required resolvents are identifiable and of finite number. Therefore, for all unsatisfiable sets, deletion-directed search will always be able to find an exhaustible literal occurrence.

The preceding examples outlined the operation of a deletion-directed strategy, but they did not reveal why and when it works effectively. The following observations give some insight into these matters. Consider an arbitrary unsatisfiable set S of clauses. If some literal occurrence L in a clause C has only one mate, then a resolution involving L and its mate (or a descendant of that mate) must occur in every refutation of S which contains C. If every refutation of S contains C (e.g. C is the negation of the theorem) then a resolution on L is essential to a refutation of S. In a minimal unsatisfiable set, any resolution which allows double parent elimination by E-rep is essential and the eliminations preserve minimality. Any resolution which allows single parent elimination by E-rep in a minimal unsatisfiable set is essential but the elimination may not preserve minimality. A sequence of resolutions starting from a minimal unsatisfiable set and culminating in a deletion by E-rep contains at least one essential resolution. These observations suggest that deletion-directed search will be most effective for a minimal unsatisfiable set containing many single-mate literals — as in the first example. Its effectiveness will diminish, however, as the set in which it operates becomes increasingly non-minimal and as the population of literals with only a few mates decreases.

If necessary, the effect of non-minimality can be substantially mitigated by using a clause generation rule which is restricted by set of support constraints12. The effect of the absence of literals having only a few mates is much more serious since the mate structure is the dominant source of guidance information. When the mate structure doesn't provide effective guidance, not only is deletion-directed search blind, but it also fails to effectively restrict the growth of the set of retained clauses. The appropriate action in such a situation is not clear but the use of some other strategy is probably the best choice.

The reader who is familiar with the Davis-Putnam procedure for testing the consistency of propositional calculus expressions13 may observe that the notion of deletion-directed search can be viewed as a generalization to the predicate calculus of their rule for eliminating atomic formulas. Other approaches *to a* general notion of deletion-directed search can be found in reports by B. Meltzer14 and R. Reiter15.

References

1. Robinson, J. A. A machine-oriented logic based on the resolution principle. J.ACM 12, l(Jan. 1965), 23-41.
2. Robinson, J. A. A review of automatic theorem proving. Proc. of Symposia in Appl. Math. 19, 1967, 1-18.
3. Kowalski, R. Search strategies for theorem-proving, in B. Meltzer and D. Michie (eds.), Machine Intelligence 5, American Elsevier Pub. Co., New York, 1970, 181-200.
4. Ackermann, W. Solvable Cases of the Decision Problem, North-Holland Pub. Co., Amsterdam, 1954.
5. Minker, J., J. R. McSkimin and D. H. Fishman MRPPS-An interactive refutation proof procedure system for question-answering. Computer Science Center TR 228, Univ. of Maryland, 1973.
6. Fikes, R. and N. Nilsson Strips: A new approach to the application of theorem-proving to problem solving. Proc, Int. Joint Conf. on A.I., London, 1971.
7. Kowalski, R. Studies in the completeness and efficiency of theorem proving by resolution. Ph.D. Thesis, Univ. of Edinburgh, 1970.
8. Gelperin, D. Clause deletion in resolution theorem proving. Ph.D. Thesis, Ohio State Univ., 1973.
9. Kowalski, R. and D. Kuehner Linear resolution with selection function. Artificial Intellignece 2, 3-4 (1971), 227-260.
10. Green, C. The application of theorem proving to question-answering systems. Ph.D. Thesis, Stanford Univ., 1969.
11. Wos, L., D. Carson and G. Robinson The unit preference strategy in theorem proving. Proc. AFIPS 1964 FJCC 26, 1964, 616-621.
12. Wos, L., G. Robinson and D. Carson Efficiency and completeness of the set of support strategy in theorem proving. J.ACM 12, 4(0ct. 1965), 536-541.
13. Davis, M. and H. Putnam A computing procedure for quantification theory. J.ACM 7, 3 (July 1960), 201-215.
14. Meltzer, B. Some notes on resolution strategies, in D. Michie (ed.), Machine Intelligence 3, American Elsevier Pub. Co., New York, 1968, 71-75.
15. Reiter, R. The predicate elimination strategy in theorem proving. Proc. 2nd ACM Symposium on Theory of Computing, 1970, 180-183.