

A HIERARCHICAL APPROACH TO THEOREM-PROVING *

Malcolm C. Harrison

Courant Institute of Mathematical Sciences, New York University, New York, N. Y. 10012

1. Introduction

A considerable amount of recent work in theorem-proving has been concerned with methods for increasing the power of inferences which can be made in special cases. This appears to be in response to the widespread recognition that general purpose theorem provers, particularly those using resolution [10] as their inference rule, have extreme difficulty with particular aspects of proofs for which there are fairly effective algorithms and heuristics. Outstanding examples are equality, commutativity, and associativity. It is generally recognized that such properties are more efficiently dealt with if they are somehow 'built in' to the theorem prover, rather than provided in the form of axioms. There is, however, an understandable reluctance to destroy generality or completeness by ad hoc mechanisms.

The usual approach has been to search for methods of replacing an axiom or a set of axioms with an inference rule. This has resulted in a number of suggested methods for treating equality [1,2,5,6,7,8,11,12], associativity and commutativity [1,4,15], and other properties [14,16]. One of the problems with equality is that proving two terms equal is in general as difficult as proving an arbitrary theorem -- equality is undecidable, and any theorem can be expressed using just the equality predicate. Associativity is not as difficult, but the fact that there are an infinite number of ways of unifying terms involving associative functions gives rise to significant problems.

Recently we have been looking at a method of incorporating axioms into inference rules which approaches this problem in a different way. The idea is to split an unsatisfiable set S of clauses into two disjoint subsets, say U and $S-U$, and to use a theorem prover working on clauses of U to generate inference rules which are applicable to clauses of $S-U$. If these inference rules are generated on demand, this method is not necessarily restricted to values of U for which an equivalent finite set of

inference rules are known. In this paper we discuss a simple inference rule called U -generalized-resolution which is complete for any subset U of S .

This approach grew out of some discussions with David Yarmush and Norman Rubin concerning the problem of equality. It was suggested that an appropriate generalization of resolution to deal with equality consisted of generalizing the unification procedure so that literals could be unified if they had the same predicate and a substitution could be found which would make the arguments equal, rather than identical. Since equality is in general undecidable, such a procedure would require the use of a theorem-prover within the unification procedure. We discuss this approach to equality, which we call U -equality-generalized resolution, in Section 2. The more general approach, which we call U -generalized resolution and which gives a framework for dealing with properties other than equality, is treated in Section 3.

2. U -Equality-generalized Resolution

Given two clauses (written as sets of literals)

$$A \cup \{L(x_1, \dots, x_\ell)\} \quad (1)$$

$$B \cup \{\neg L(y_1, \dots, y_\ell)\} \quad (2)$$

we can infer

$$[A\lambda - \{L(x_1, \dots, x_\ell)\lambda}]$$

$$\cup [B\lambda - \{\neg L(y_1, \dots, y_\ell)\lambda\}] \cup \bigcup_i C_i \lambda \quad (3)$$

if we can prove

$$C_i \cup \{r_i = s_i\}, \quad i = 1, 2, \dots, \ell \quad (4)$$

and λ is the most general simultaneous unifier (mg-su) of the pairs of terms

$$(x_i, r_i), \quad (y_i, s_i), \quad i = 1, 2, \dots, \ell.$$

If (4) is deduced from a set U of clauses, we refer to (3) as a U -equality-generalized resolvent of (1) and (2).

* This work was supported by grant number N0014-75-C-0571 from the Office of Naval Research; and by ERDA, contract number EY-76-C-02-3077.*000.

In a similar way, from the clause

$$A \cup \{L(x_1, \dots, x_\ell), L(y_1, \dots, y_\ell)\} \quad (5)$$

we can infer

$$A\lambda \cup \{L(x_1, \dots, x_\ell)\} \cup \bigcup_i C_i\lambda \quad (6)$$

if we can prove

$$C_i \cup \{r_i = s_i\}, \quad i = 1, 2, \dots, \ell \quad (7)$$

and λ is the mgsu of the pairs (x_i, r_i) and (y_i, s_i) , $i=1, 2, \dots, \ell$. We call (6) a U-equality-generalized factor of (5) if (7) are deduced from a set U of clauses.

It is clear that for any inconsistent set S of clauses, the inference rules of U-e-g-resolution and U-e-g-factoring are sound and complete if U is any subset of S to which is added the equality reflexive axiom $x = x$, since any resolvent or factor can be obtained by U-e-g-resolution or U-e-g-factoring using just the clauses $x = x$ inferred from U . What we would like to be able to do is to prove completeness results for sets U when U-e-g-resolution and U-e-g-factoring is only permitted on clauses of $S-U$ or their U-e-g-resolvents or U-e-g-factors. A preliminary result of this form is given in (5); this says that if U contains just the axioms of equality, possibly together with some or all positive equality unit clauses from S , a refutation of $(S-U) \cup \{x = x\}$ can be obtained by U-e-g-resolution alone if S has a refutation tree. We say that a set of clauses has a refutation tree if and only if it has a refutation graph, using the terminology of Shostak [13], which is a tree (i.e. each bridge connects exactly two literals); having a refutation tree is equivalent to having a unit refutation or an input refutation. This result shows that, at least in the case when S has a refutation tree, the use of the equality axioms can be restricted to within the unification procedure.

This method of dealing with equality would appear to have a number of advantages over other methods such as paramodulation. These include:

- The application of aimless inferences involving equality substitutions is restricted to cases when such substitutions will permit a resolution which would not be possible in the absence of the equality clauses.

- The equality subproblem is specified by two terms which must be made equal. In heuristic search terms this is a problem with a well defined start node and a well defined goal node, and so should be

subject to more powerful techniques such as bi-directional search, and operator-difference methods (such as those used with considerable effectiveness in FDS [9], for example). Note that in the standard resolution approach, the goal node is the null clause, which does not provide very much information which can be used for heuristic guidance.

- Properties such as associativity and commutativity are specified by positive equality unit clauses, so that their use can be restricted to within the equality-prover. The use of specialized techniques for dealing with these properties, such as the use of parenthesis-free notation, can therefore be restricted to within the equality-prover.

So far, investigations carried out to determine the effectiveness of this equality method are very encouraging. For sets of clauses with input refutations, some results are given in [5]. A more extensive, but also a more complex, set of experiments have been carried out by Rubin [12]. This involved the construction of a theorem-proving/programming system called DILEMMA, using e-g-resolution (called coercion), evaluable predicates, and heuristic guidance functions. This was used to prove theorems in geometry, giving results similar to those of the well known work of Gelernter [3]. In fact in some respects the results were superior, since the construction procedure used by Gelernter dropped out automatically from the unification procedure.

3. U-generalized-resolution

U-generalized resolution can be defined in the following way. From the clauses

$$A \cup \{L(x_1, \dots, x_\ell)\} \quad (8)$$

$$B \cup \{M(y_1, \dots, y_m)\} \quad (9)$$

we will infer

$$[A\lambda - \{L(x_1, \dots, x_\ell)\lambda}] \cup [B\lambda - \{M(y_1, \dots, y_m)\lambda}] \cup C\lambda \quad (10)$$

if we can deduce the clause

$$C \cup \{\neg L(r_1, \dots, r_\ell), \neg M(s_1, \dots, s_m)\} \quad (11)$$

from a set U of clauses, where λ is the mgsu of the pairs

$$\begin{aligned} (x_i, r_i) &, & i = 1, 2, \dots, \ell, \\ (y_i, s_i) &, & i = 1, 2, \dots, m. \end{aligned}$$

In the above definition we will normally not permit the L and M literals to be the constant literals "true" or "false", since every resolvent would also be a U-g-resolvent for any U. However, in the special case when (B) is null and when the U-g-resolvent (10) subsumes either $A\lambda$ or $C\lambda$ we will permit the M literal to be 'true'. We will then refer to (10) as a unary U-g-resolvent. Unless stated to the contrary, the term U-g-resolution will be intended to include unary U-g-resolution.

Proof. Consider any resolution refutation R of S , consisting of operations of factoring or resolution, and consider the resolutions of R in order. Each such resolution is between two clauses, each of which is either supported by $(S-U)$, or is not. Each resolution in R is thus one of the following

- between two (S-U)-supported clauses
- between two non-(S-U)-supported clauses
- between an (S-U)-supported clause and a non-(S-U)-supported clause

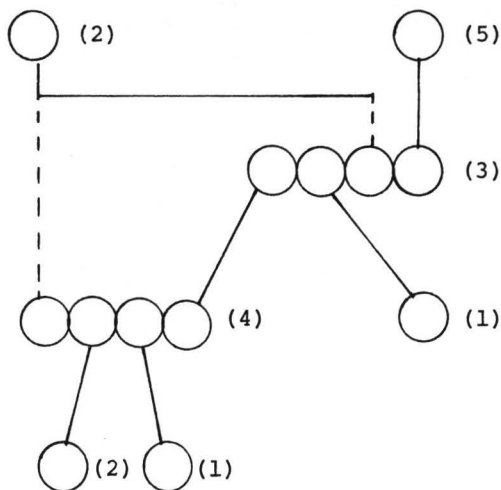
In the first case the resolvent is also a binary U-g-resolvent with the linking clause being a tautology containing only the resolved literal and its negation. In the second case we can regard the resolvent as part of the process of deducing a linking clause from U; this clause is also non-(S-U)-supported. In the third case we can construct the resolvent by U-g-resolution in the following way. If $A \cup \{L\}$ is the (S-U)-supported clause and $B \cup \{\neg L\}$ is the non-(S-U)-supported clause, with L being the literal resolved on, construct the U-g-resolvent of $A \cup \{L\}$ and $A \cup \{L\}$ with $B \cup \{\neg L\} \cup \{L\}$ as the linking clause. Thus each resolvent in R can be constructed by U-g-resolution, which with factoring is therefore complete. $\square$

of refutations in the form of refutation graphs [13]. For example, consider the following set of clauses (actually the clauses for a version of the theorem that in a group the left inverse is a right inverse):

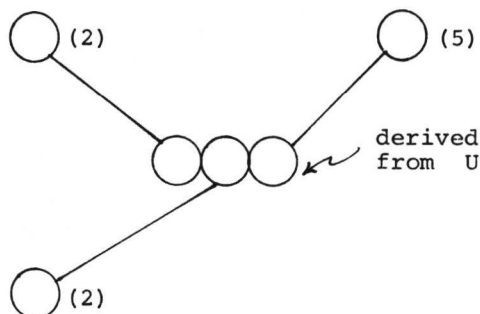
1. $P(e, x, x)$ ◦ identity
2. $P(x^{-1}, x, e)$ ◦ left inverse
3. $\bar{P}(u, v, x), \bar{P}(v, w, y), \bar{P}(u, y, z), \bar{P}(x, w, z)$
 ◦ associativity
4. $\bar{P}(u, v, x), \bar{P}(v, w, y), \bar{P}(x, w, z), \bar{P}(u, y, z)$
 ◦ associativity
5. $\bar{P}(a, a^{-1}, e)$ ◦ negated theorem

(Circles represent literals, touching circles are in the same clause, solid lines represent resolution steps, dotted lines represent merges.)

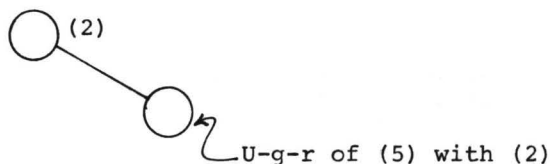
Theorem Proving-2: Harrison
531



Initial graph.



Intermediate step showing U-g-r of (5) with (2).

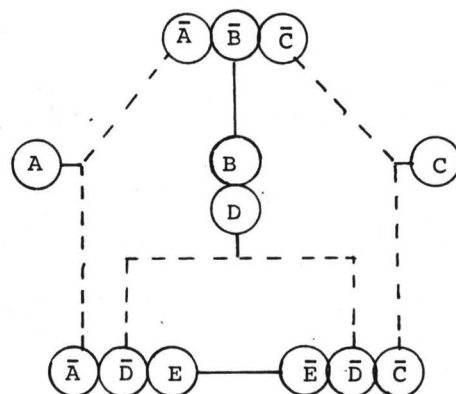


Graph after one step.

□ Proof obtained after 2 steps.

In fact it can be shown [5] that any refutation for which the refutation graph is a tree can be collapsed by U-g-resolution without using any applications of U-g-resolution in which the two (S-U)-supported clauses are identical in the graph (we will call these weak U-g-resolutions).

Unfortunately, this result is not true in general. The following refutation graph, for example, in which the clauses in U are shown with a double line around them, cannot be collapsed by (nonweak) U-g-resolution steps:



However, by duplicating the C clause above, this graph can be collapsed by 4 U-g-resolution steps, still better than the best resolution refutation of 5 steps. An informal survey of a number of resolution graphs suggest that the number of weak U-g-resolution steps required is either zero or is very small. In fact we have not found a counterexample to the conjecture that if (S-U)-supported clauses can be duplicated, as in the above example, no weak U-g-resolutions will be required at all.

The use of U-g-resolution does not necessarily aid the search process required to construct a proof, though within U the search for a linking clause can be bidirectional rather than unidirectional, as in U-e-g-resolution. It is our feeling that U-g-resolution will be most effective when the clauses of U have a structure which facilitates this search. In fact it is our feeling that the domain-independent approach to theorem proving will not prove to be fruitful in any case, so we are more interested in specialized applications of U-g-resolution. We anticipate that these

specializations will come in the form of more efficient algorithms for inferring appropriate clauses from particular subsets U . We feel that our results with U -e-g-resolution offer substantial evidence that this approach is effective.

Another possibility is that the theorem proving technique used to deduce clauses from U could use U '-g-resolution as its inference rule, where $U' \subset U$. This suggests a possible scheme for a multilevel hierarchical theorem proving system, which we would expect to be organized along the following lines:

- Each level would deal with problems whose solutions are expressed in a small number of powerful inference steps.

- Each level would deal with a subset of the clauses concerned with a particular aspect of the problem, and would be concerned with solutions to the corresponding subproblems.

- Each level would use specialized techniques if these were known to be effective for the particular subproblem.

A hierarchical organization such as discussed here is somewhat more difficult to organize than a conventional single-level organization. In general it will not be possible for the lower-level theorem prover to terminate its search, so some mechanism must be provided for it to resume later. A simple implementation would just use some kind of cut-off parameter which could be incremented if the proof was not successful. A more sophisticated approach would be for the lower-level theorem prover to return resumption information, perhaps in the form of an incomplete search tree, or in the form of a suspended 'process'.

References

1. J. Darlington, 'Automatic Theorem-proving with Equality Substitutions and Mathematical Induction', in *Machine Intelligence 3*, Oliver & Boyd, 1968.
2. J. Dixon, 'Z-resolution: Theorem-proving with Compiled Axioms', *JACM*, January 1973.
3. H. Gelernter, 'Realization of a Geometry Theorem Proving Machine', in: E. Feigenbaum and J. Feldman, 'Computers and Thought', Mc-Graw-Hill, 1963.
4. J. Guard et al., 'Semi-automated Mathematics', *JACM*, January 1969.
5. M. C. Harrison and N. Rubin, 'A Generalization of Resolution', to be published in *JACM*.
6. J. B. Morris, 'E-resolution: Extension of Resolution to Include the Equality Relation', *IJCAI* #1, 1969.
7. A. J. Nevins, 'A Human Oriented Logic for Automatic Theorem Proving', *JACM*, October 1974.
8. G. Plotkin, 'Building in Equatorial Theories', *Machine Intelligence 7*, Halstead Press, 1973.
9. J. R. Quinlan and E. B. Hunt, 'A Formal Deductive Problem-Solving System', *JACM*, October 1968.
10. J. A. Robinson, 'A Machine Oriented Logic Based on the Resolution Principle', *JACM*, January 1965.
11. G. Robinson and L. Wos, 'Paramodulation and Theorem-Proving in First-order Theories with Equality', *Machine Intelligence 4*, American Elsevier, 1969.
12. N. Rubin, 'A Hierarchical Technique for Mechanical Theorem Proving and its Application to Programming Language Design; Courant Computer Science Report No. 10, November 1976.
13. R. E. Shostak, 'Refutation Graphs', *Artificial Intelligence 7*, 1976.
14. J. Slagle, 'Automatic Theorem-proving with Built-in Theories Including Equality, Partial Ordering and Sets', *JACM*, January 1972.
15. J. Slagle, 'Automated Theorem-proving for Theories with Simplifiers, Commutativity and Associativity', *JACM*, October 1974.
16. J. Slagle and L. Norton, 'Experiments with an Automatic Theorem-prover Having Partial Ordering Inference Rules', *CACM*, November 1973.