A HOLE IN GOAL TREES: SOME GUIDANCE FROM RESOLUTION THEORY†

D. W. Loveland* and M. E. Stickel

June 1973

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

This paper is to be presented at the Third International Joint Conference on Artificial Intelligence.

†Research supported in part by NSF Grant GJ-28457X1.

*Present address: Computer Science Department, Duke University, Durham, N. C. 27706.

A HOLE IN GOAL TREES: SOME GUIDANCE FROM RESOLUTION THEORY†

D. W. Loveland* and M. E. Stickel
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

The representation power of goal-subgoal trees and the adequacy of this form of problem reduction is considered. A number of inadequacies in the classical form are illustrated, and two versions of a syntactic procedure incorporating extensions are given. Although the form of the corrections are suggested from resolution theory results, and the value of this connection emphasized, the paper discusses the goal tree format and its extensions on an informal level.

Key words: theorem proving, goal trees, AND/OR trees, Geometry Theorem Machine, resolution, model elimination.

1. Introduction

After several years when almost all theorem proving systems, and many problem solving systems, were based on resolution, many researchers are returning to natural deduction type logics, often implemented via some form of goal-subgoal tree notation using a problem reduction approach. In this paper the goal-subgoal tree form (or AND/OR tree form) is considered. We show that if one wishes to use this syntactic form for representation of the deductions and search space as a full replacement for the resolution approach, one must make some additions to the classical problem reduction formulation.

To show that there exist holes in the classical goal-subgoal problem reduction method we need only present some examples, which we supply. To determine an appropriate correction and measure its power takes some theory. It turns out that resolution theory, in particular the model elimination procedure results, provides an adequate theoretical base. In this paper we only state the consequences for the problem reduction approach, omitting proofs. However, we want to stress the value of resolution theory for the insight it gives to the problem reduction method and remark that more information than is exploited here can certainly be pulled from existing resolution theory.

AND/OR trees, used as goal trees, are components of most problem solving systems that are not resolution based. We are hereafter concerned only with goal trees used for logical inference. We show, among other things, that the usual way of organizing goal-subgoal trees is incomplete yet one small change makes the mechanism complete, assuming equality substitution is not relevant, and if the equality predicate is used, several added rules gives completeness in general. By completeness, we mean that the goal trees and associated syntactic mechanism is capable of establishing a goal statement whenever the goal is valid given the assertions present. The systems we discuss are the search trees such as are used in the Geometry Theorem Machine (Gelernter et al.[2,3,4]), the Logic Theorist (Newell et al.[12]) and elsewhere. Indeed, when the equality predicate is not present, the mechanism of the Plane Geometry Machine is sufficient in structure and mechanism to be complete yet is not complete.

The subject of completeness is embroiled in controversy these days. We feel a developing consensus that total completeness is pointless to pursue, and for almost all problems, pursuit of the solution will be done by methods (particularly search methods) incomplete in themselves, yet the total reservoir of tools to be drawn upon should be complete if at all possible. In particular, one wishes the underlying organization and recording mechanism (this is what AND/OR goal trees are) to be capable of handling any situation. The worst possible situation is to be prevented from establishing a simple inference not because one is unable to thread through the search space but because the inference chain cannot even be represented. We claim this is particularly bad because the problem specific search tools are expected to be updated frequently while the underlying recording (proof) mechanism is viewed as far more stable. In analogy, inability to express concepts due to inadequate grammatical structure is worse than inadequacy due to a limited vocabulary, for one more readily adds to his (her) vocabulary. One wishes a grammar "complete" although no one expects a "complete" vocabulary.

As regards goal trees, one instance of inadequate understanding of goal trees and the associated mechanisms is reported in Gelernter[3]. This paper documents an instance where a mechanism mixing use of the STUCK and ESTABLISHED labels with goal elimination due to identical higher goal resulted in the inability to infer theorems whose known proof complexity suggested solution should be possible. As search spaces were relatively small, most runs could be carefully analyzed, so the flaw was probably discovered on the first theorem for which the flaw actually prevented the proof. However, the Geometry Theorem Machine had been in operation nearly a year at that time and many "production" runs were made prior to this discovery. Moral: flaws in infrequently used logical paths may be particularly bad because simple (and important?) results may be blocked long after the system is believed "debugged" in its basic routines.

We do not consider completeness proofs here but rely on examples to suggest the need and degree of applicability of extensions to the classical form for goal trees. Those familiar with resolution theory (in particular, model elimination as given in Loveland[10], also in Kowalski and Kuehner[8]) will be able to verify some claims. Other assertions are based on results to appear in a forthcoming book on theorem proving by one of the authors[11].

At this stage of development of the artificial intelligence field, we feel it is unnecessary to justify interest in theorem proving techniques themselves. The bibliography lists a small sample of papers that investigate theorem proving techniques or apply such techniques to robot guidance, question-answerer systems, automatic programming, etc.

2. Goal Trees

By a goal tree we mean an AND/OR tree developed by a problem reduction mechanism. A "classical" treatment of goal trees occurs in Nilsson[13] and Slagle[19], for example. We review this notion briefly by outline

† Research supported in part by NSF Grant GJ-28457X1.

* Present address: Computer Science Department, Duke University, Durham, N. C. 27706.

and example.

Let us represent our syntactic, or semantic, atoms by capital Latin letters: A, B, C, ..., with subscripts if necessary. Of course, A may be a complex formula, e.g., $(\forall y)(\forall x)P(x,y) \supset Q(y)$, but we agree not to consider its interior structure relevant to the particular problem so it is "packaged" as A. We consider our primary, or top, goal G to be the atom to be established. Assertions (facts) are of the form $A_1 \wedge \ldots \wedge A_n \rightarrow C$ (implications) or P (premises). The A_i are antecedents and C is the consequent of the implication. For notational convenience, we define the consequent of a premise to be the premise itself and the set of antecedents of a premise to be the empty set. For a particular problem we begin with a goal to be established and a set of assertions. We consider the expression format more closely later.

A goal tree records the development of the search to establish G by linking it to the premises via the implications. G is the top goal; if it is also a premise, G is established. Otherwise all implications with consequent G are located and the antecedents of each such implication become new goals, subgoals of G. G is the parent of each new goal and each new goal is the successor of G. If each new goal for one of the implications can be established, G is then established (by asserting the implication). The antecedents of one implication form partner goals. We also refer to a conjunction of goals meaning the set of antecedents from one implication. Any single set of partner goals (goals in conjunction) at this level that can be established establishes G. This yields a disjunction of partner goal sets. If no partner goal set corresponds to a set of premises, some partner goal set is selected and each of the partners not a premise is again matched against implication conclusions to create (possibly) new subgoal sets (not necessarily as a single parallel action). This proceeds in iteration until a sufficient set of premise matches are found, or the search stops. The conjunction/disjunction relationship above leads to the name AND/OR tree.

A goal A is an ancestor of goal B if A is the parent of B or A is an ancestor of the parent of B. A partner of an ancestor of the goal A is called an ancestor partner of A.

We give an elementary example from plane geometry in the spirit of the Geometry Theorem Machine (GTM); see Figure 1. Immediate subgoals lie below their goal and are connected by a slanted line. Partner goals are connected by a horizontal line. In Figure 1 the bottom leftmost conjunction of goals is rejected even though two goals are premises because the third goal also occurs as the top goal, thus it is an ancestor of itself. Any goal that occurred as an ancestor goal of itself was rejected at the lower level in the GTM structure because if it could be established at all, it could be established from the higher level. Also in the GTM structure was a way of discarding a conjunction of goals if a higher conjunction containing an ancestor was easier to prove. We do not elaborate for we handle this somewhat differently. The key point is that interaction with ancestor goals existed, and was very important due to the "depth first" search which meant not leaving a branch until you could go no further.

We now enlarge our format for expressions. This is done by allowing our atoms to be literals, atoms possibly preceded by a negation sign. Thus if A is a complex expression, we look inside only to check if the leftmost symbol is a "not" operator of propositional logic. If so, it is displayed. We let A, B, C,... (possibly with subscripts) represent literals. To emphasize that B is A proceeded by a "not" we will sometimes write B as $\sim A$. A and $\sim A$ are complement literals. Otherwise, our expression format is as before.

The use of negated goals has not appeared in the classical inference programs using goal-subgoal systems. The Geometry Theorem Machine avoided the need to recognize complementary goals almost by accident, for concepts like "XYZ is collinear" and "XYZ is not collinear" both appeared but did not interact. However, in general situations particularly in robot systems, question answerer systems, etc. interaction between complementary literals is to be expected. Certain recent systems of a goal-subgoal format have been designed to handle negated formulas so that complemented literals interact; see Bledsoe et al.[1], Reiter[15]. These systems are less in the classical goal-subgoal format than the system considered here and also appear to be incomplete.

We consider in Figure 2 a simple example in which the goal follows from the assertions but the goal-subgoal mechanism so far illustrated will not establish the goal. One reason is that the contrapositive of one of the assertions is needed. We add the contrapositive as an explicit assertion. We note, however, that there is no way of proceeding to a premise! Yet the problem is simple enough so that one can read the intended meaning of the assertions and see that the goal follows. We claim that because $\sim C$ occurs as an (indirect) subgoal of C, we can treat $\sim C$ as if it were a premise and terminate that branch. That is, $\sim C$ is now marked contradicted and considered established. As A is a premise, B is established, so C is established, as desired.

The rationale for the mechanism above is not hard to find. Either C is true or $\sim C$ is true. If $\sim C$ is true, then we can establish C (after establishing other pertinent subgoals), which is impossible. Thus C is true. This is an argument by contradiction. We observe that the check for this is trivial if possible identity with ancestor goals is checked as in the GTM. One simply checks for identity and then complementation.

The not-so-immediate fact is that we now have a propositionally complete system. That is, if no substitution inside literals is allowd so as to make distinct literals alike (or complementary) no further gimmicks will be necessary. In Figure 2, we note a possible alternate argument to produce establishment is that one of D and $\sim D$ is true so one of the two ways of establishing C should be permitted. Is this sufficient also? Probably so, we are not sure. In any event, it is generally a much more difficult check as the occurrences of D and $\sim D$ are on different disjunctive branches and can be made to appear at an arbitrary depth by making the inference connecting C and D more complex. Thus instead of a nearly free check one has a relatively complex tree search. But might such a tree search be necessary anyway, for some case where ancestor complements do not occur? No. That is the meaning of our statement that the system is now propositionally complete. The proof is a consequence of the completeness of model elimination (ME).

In general problem solvers will not be constrained to work propositionally. The expressions we have considered, goal and assertions, will in general have free variables and functions, including Skolem functions which build in universal quantifiers. We do not consider in detail the process of general conversion to our chosen format (generalized somewhat below). It is basically the conversion to disjunctive normal form with Skolem functions, the dual to the "conversion" in Nilsson[13], for example. The general structure of the goal-subgoal mechanism when operating in the presence of free (individual) variables and substitution is the same but with direct comparison replaced by the notion of unification from resolution (see Robinson[17], Nilsson[13], or Slagle[19]).

One of the common substitution situations involves equality. If we have goal P(a) and assertion a=b we certainly consider P(b) a subgoal whose establishment would yield P(a). Indeed, some readers may wonder why we need to write P(b) explicitly. P(a) might be interpreted as all statements equivalent to P(a) under equality substitution. This has disadvantages when substitutions use numerous derived equations so we reject this here although a use of such identification might be satisfactory. Such a treatment is compatible with our main points but requires a modified organization to that given below.

In Figures 3, 4, and 5 we give examples where the goal should be inferred from the assertions presented but cannot be inferred under the simple format of the preceding paragraph. These figures suggest the format in which we propose to handle such problems. That is, in our general description below the problems stated would generate the goal tree presented. Note that in Figure 4 an alternate form of implication 1 is needed. We supply it here as assertion 4. We call 4 a <u>general contrapositive</u> of 1. We remark that we would expect the situation of Figure 4 to arise very infrequently so such an inference route should be investigated only when desperate.

Again, if we adopt the few rules for handling equality given below, of which three instances have been displayed, we have completeness of the goal tree procedure when equality substitution is included. The completeness proof comes from the appropriate form of ME with paramodulation (an equality handling mechanism) whose proof appears in Loveland[11].

A number of other features for goal tree analysis can be gleaned from results concerning ME. Most are natural in this setting such as the removal of a conjunction of goals when one goal matches an ancestor goal. We noted this was incorporated in the GTM. A non-intuitive situation is that a conjunction of goals can be eliminated if one of the component goals is complementary to an unexpanded ancestor partner goal, i.e., a goal with no subgoals yet recorded, but completeness is not assured unless a goal is marked <u>displaced</u>, and treated as established, whenever it matches an unexpanded partner or an unexpanded ancestor partner goal. Displacement is illustrated in Figure 6. Displacement avoids expanding the same subgoal twice. One has no need for the displacement device if the coincident ancestor partner has been expanded and established. The matching subgoal can directly be marked "established".

Figure 7 is an example of another situation we must handle. If S is an unsatisfiable formula, $S \rightarrow C$ is valid for any formula C. We use the device of the contradictory formula $\not\ell$, which may be considered a shorthand for formula $P \wedge \sim P$. This device allows a natural extension of our notion of assertion and goal and suffices to handle cases where the goal, or subgoal, cannot be directly derived although it is a valid consequence of the assertions.

We write the general format for our goal tree system as if a propositional system is our concern. That is, all comparisons of literals are by identity or complementarity. However, the word <u>matches</u> is used for this identity check. By interpreting <u>matches</u> as using a most general unifying substitution, the general form is realized when substitution for (individual) free variables is permitted. We include in our format the substitution of equality but, again, with the ambiguity which may or may not allow free variables in those terms.

For convenience we label the problem reduction procedure below the MESON (<u>M</u>odel <u>E</u>limination <u>S</u>ubgoal <u>O</u>rie<u>n</u>ted) procedure.

We consider again the expression format. An arbitrary first order formula can be converted to the appropriate expression format, preserving validity. A formula, or (finite) set of formulas, not already suitably expressed should be converted to the following form:

$$B_1 \wedge \ldots \wedge B_n \rightarrow G,$$

where B_i is of the form $A_1 \wedge \ldots \wedge A_m \rightarrow C$ or C and G is of the form $L_1 \wedge \ldots \wedge L_k$, where the A_i's, L_i's and C are literals. This is readily obtained from the disjunctive normal form of the original formula. G then defines the goal: if K=1, L_1 is the single goal, otherwise $L_1, \ldots, L_k$ are top level partner goals all of which must be eventually established. We can tackle one at a time (though they may be linked by common variables) so hereafter we consider a single goal C. $A_1 \wedge \ldots \wedge A_m \rightarrow C$ is an assertion implication, and C a premise. An important equivalence for format preparation is $(A \rightarrow B \vee C) \equiv (A \wedge \sim B \rightarrow C)$. This is used to form the various general contrapositives needed for completeness. We extend this to generate $\sim A \rightarrow \not\ell$ from A, for example.

If the goal is believed to follow directly from the assertions (as is usually the case) the use of $\not\ell$ may be avoided. Otherwise, add $\not\ell \rightarrow G$, the <u>assertion generated from the goal</u>, to the assertions and for each assertion $A_1 \wedge \ldots \wedge A_m \rightarrow C$ add the general contrapositive $A_1 \wedge \ldots \wedge A_m \wedge \sim C \rightarrow \not\ell$, and for each premise add $\sim C \rightarrow \not\ell$. Only one such formula need be added to the assertion list if some version of that assertion is believed necessary to establish the result.

It is necessary to consider, for each assertion implication $A_1 \wedge \ldots \wedge A_m \rightarrow C$, m general contrapositives plus the original assertion if completeness is to be preserved. There should be one general contrapositive $A_1 \wedge \ldots A_{i-1} \wedge \sim C \wedge A_{i+1} \ldots \wedge A_m \rightarrow \sim A_i$ for each i. The order of antecedents in any assertion is immaterial.

3. The MESON Procedures

The procedures presented here are for propositional (variable free) problems. We will make occasional reference to the requirements of the procedures utilizing variables.

The procedures represent syntactic systems for adding to a goal tree information about goal-subgoal relationships and establishment of goals. The procedures return "success" or "failure" according to whether the top goal can be established or not respectively. Of course, the ability to return "failure" disappears when substitution is allowed, e.g., first-order formulations. A returned value of "failure" for a problem indicates either the top goal does not follow from the assertions or the search ordering and goal generation and deletion strategies specified by the planning routine are inadequate for the problem. (It is possible to write a complete planning routine which theoretically always returns "success" for soluable problems.)

We will now present two MESON procedures for goal tree analysis incorporating the new rules discussed above* The procedures are logically divided into four subprocedures with labels "initialize", "loop", "updatejnarks" and "updatejgoals".

The instructions placed at the label "initialize" define GOALS (the set of goals to be attempted) to be the set consisting of only the top goal and also initialize the goal tree.

The instructions placed at the label "loop" select a goal G from GOALS, an operation to be performed and an assertion D if needed. The selected operation is then performed for the goal G and assertion D. Those operations try to establish goals or create subgoals.

The instructions placed at the label "update^ marks" mark a goal "established" if each of a lilt of partner successors is marked "established", "contradicted" or "displaced". Thus, if each of a conjunction set of subgoals of a goal is established, the goal is established.

The instructions placed at the label "update_ goals" add newly generated subgoals to the tree and GOALS provided certain acceptance criteria are met.

The selection of the next goal in GOALS to be operated upon and the selection of the operation and the assertion to be used in operating on that goal are assumed to be accomplished by some externally specified planning routine ("the planner"). The planner, in addition to specifying a search strategy, may restrict or totally eliminate use of some of the operations. For example, traditional goal tree procedures without the contradiction mechanism correspond to a planner which never uses the operation at "op3".

The planner, by applying the operation at "op5" to a goal, removes the goal from GOALS and thereby signifies that no more operations will be applied to the goal.

If one wishes to insure completeness, the planner must in some order process all operations (except the operation at "op5") for each goal and potentially applicable assertion. The planner may select the goals of a conjunctive set of goals in any desired order to attempt their establishment. The procedure(s) make no assumption as to whether the search is depth-first, breadth-first, or some mixture of these.

MESON procedure

initialize: Let GOALS be a set consisting of only the top goal. Initialize the goal tree to the top goal.

loop: If GOALS is empty, exit procedure with "failure". Let G be a goal in GOALS selected by the planner. The planner selects one of the following operations to be performed on G and selects D, a premise, implication or general contrapositive of implication, as required by the operation.

opl: If G matches the premise D, mark G "established" and go to updatejnarks. Otherwise go to loop. ~

op2: If G matches the consequent of D, where D is an implication or general contrapositive of implication, let A be the set of the antecedents of D and go to updatejgoals. Otherwise go to loop.

op3: If G matches the complement of an ancestor of G, mark G "contradicted" and go to updatejnarks. Otherwise go to loop.

op4: If G matches an unexpanded partner of G not tnarked "displaced" or an unexpanded ancestor partner of G, mark G "displaced" and go to updatejnarks. Otherwise go to loop.

op5: Delete G from GOALS and go to loop.

updatejnarks: If G is top goal, exit procedure with "success". If all partner goals of G are marked "established", "contradicted" or "displaced", let G, be the parent of G, set G-Gj, mark G "established" and go to updatejnarks. Otherwise go to loop.

updatejgoals:

test 1: If a member of A is identical to G or an ancestor of G, go to loop.

test 2: If a member of A is complementary to another member of A, an unexpanded partner of G or an unexpanded ancestor partner of G, go to loop.

Otherwise add the members of A to GOALS and to the goal tree as a conjunctive set of successors of G and go to loop.

The MESON procedure for equality incorporates rules for handling the equality relation. It differs from the MESON procedure in that three new operations are added. Also, the rules for disregarding newly generated subgoals (at "test 1" and "test 2") have not been proven to preserve completeness although we believe completeness is preserved with these rules applied. We maintain the updatejgoals subprocedure in the MESON procedure for equality with the admonition that if completeness is to be preserved these rules should be bypassed (at present).

For technical reasons, it is necessary to put in premises of the form a»a for each term a or, if in a setting using free variables and substitutions, one must put in x«x and f(x^,...,X,) - f(x.,...,X,) for each n-ary function symbol f. Such axioms can be replaced by appropriate procedure rules if desired.

MESON procedure with equality

initialize: (same as for MESON procedure)

loop: (preface and operations 1-5 same as for MESON procedure; only change is the addition of the following operations)

op6: If G contains a term matching term a where a»b or b»a is the consequent of D, where D is a premise, implication or general contrapositive of implication, let A be the set consisting of G with a single instance of a replaced by b plus the antecedents of D and go to update^ goals. Otherwise go to loop.

op7: If the consequent of D, where D is a premise, implication or general contrapositive of implication contains a term matching term a where G is a/b or b^a, let A be the set consisting of the complemented consequent of D with a single instance of a replaced by b plus the antecedents of D and go to update_goals. Otherwise go to loop.

op8: If H is an ancestor of G or G itself and H (resp. G) contains a term matching term a where G (resp. H) is a≠b or b≠a, let A be the set consisting of H (resp. G) with a single instance of a replaced by b and go to update_goals. Otherwise go to loop.
(Note: see examples below.)

update_marks: (same as for MESON procedure)

update_goals: (same as for MESON procedure)

We attempt to clarify op8 and shed light on its usefulness. Consider the case that H is G and G is a≠b, a and b simple constants. Then, reading the "respectively" case, we see that G contains term a and H is a≠b. Then A, the possible new subgoal, is G with replacement, i.e., b≠b. Taking the other case (ignoring the "respectively") yields the same possible subgoal. This is certainly unproductive and could actually be deleted with no risk involved. However, in a free variable setting with substitution it is important. Suppose the goal is f(x)≠x and the sole premise is f(f(x))≠x. By op8 where H is G is f(x)≠x, ignoring the respectively's (for variation), we have H containing a term x matching f(y), under substitution f(y) for x, where G is f(y)≠y (the change of variable name is a necessary detail); now a≠b is f(y)≠y. Then A consists of H with replacement, i.e., f(f(y))≠y. This subgoal matches the premise and the desired result is obtained.

It is impossible to give an adequate discussion within this paper of the modifications required to handle first order formulas, i.e., allowing quantification of individual variables in the problem statement. This is best done elsewhere where space permits a full discussion. The modifications are generally straightforward if the reader is familiar with resolution theory, in particular ME. See Loveland[11]. Subtle points do arise, however, as suggested below.

Performing matching by use of the general unification algorithm is an important idea and, although we can conceive of reasons to select less general substitutions under certain conditions, the advantages of obtaining the most general substitution should not be given up lightly. This is an important aspect where knowledge from resolution theory can enhance the problem reduction method.

We make two further points, really warnings, concerning adopting the above description of the MESON procedure to first order expressions. If the goal has a free variable in it, the negation of the goal should be made a (hypothetical) premise. To see this, consider the following example: Goal: P(y) (i.e., we want to know if $\exists y P(y)$). Assertion: $\sim P(f(a)) \rightarrow P(a)$. Clearly either P(a) or P(f(a)) holds. We need $\sim P(x)$ as a premise to realize this. A second point: a substitution may occur in a subgoal when applying an assertion implication. This substitution must be made at each occurrence of the replaced variable throughout the goal tree. Thus copies of the goal tree must be retained in such instances for back up in case of failure. A good format for handling this involves adopting the ME format to the MESON procedure organization.

4. Conclusion

This paper can be read simply for the illustrations of possible extensions for the problem reduction method. However, we have attempted to convey informally that resolution theory can contribute to the understanding of alternate syntactic methods. Other devices of resolution such as linear representation of goal trees and use of unit clauses from premises may also be of use. We do believe that the MESON format, which simply extends classical goal tree representation, may present a very useful way of incorporating resolution ideas in future problem solving programs.

Bibliography

1. Bledsoe, W. W., R. S. Boyer and W. H. Henneman. Computer proofs of limit theorems. Artificial Intelligence, 3 (1972), 27-60.

2. Gelernter, H. Realization of a geometry theorem-proving machine. In Feigenbaum, E. A. and J. Feldman (eds.) Computers and Thought. McGraw-Hill, New York, 1963, 134-152.

3. Gelernter, H. Machine generated problem solving graphs. Proc. Symp. Math. Theory of Automata (1963), 179-203,

4. Gelernter, H., J. R. Hansen, and D. W. Loveland. Empirical explorations of the geometry theorem machine. In Feigenbaum, E. A. and J. Feldman (eds.), Computers and Thought. McGraw-Hill, New York, 1963, 153-163.

5. Green, C. Application of theorem proving to problem solving. Proc. Int. Joint Conf. on Artificial Intelligence (1969), 219-239.

6. Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. MIT Artificial Intelligence Laboratory report AI TR-258 (April 1972).

7. Kowalski, R. AND/OR graphs, theorem-proving graphs and bi-directionsl search. Machine Intelligence 7, 1973.

8. Kowalski, R. and D. Kuehner. Linear resolution with selection function. Artificial Intelligence 2, (1971), 227-260.

9. Loveland, D. W. A simplified format for the model elimination theorem-proving procedure. J. ACM 16, 3 (July 1969), 349-363.

10. Loveland, D. W. A unifying view of some linear Herbrand procedures. J.ACM 19, 2 (April 1972), 366-384.

11. Loveland, D. W. Forthcoming book on mechanical theorem proving. North-Holland (early 1974).

12. Newell, A., J. Shaw and H. Simon. Empirical explorations of the logic theory machine. In Feigenbaum, E. A. and J. Feldman (eds.) Computers and Thought. McGraw-Hill, New York, 1963, 109-133.

13. Nilsson, N. J. Problem Solving Methods in Artificial Intelligence. McGraw-Hill, New York, 1971.

14. Reiter, R. Two results on ordering for resolution with merging and linear format. J. ACM 18 (October 1971), 630-646.

15* Reiter, R. The use of models in automatic theorem-proving. University of British Columbia Department of Computer Science report 72-09 (September 1972).

16. Robinson, G. A. and L. T. Wos. Paramodulation and theorem-proving in first-order theories with equality* In Meltzer, B. and D. Michie (eds.) Machine Intelligence 4. American Elsevier, New York, 1969, 135-150.

17. Robinson, J. A. A machine-oriented logic based on the resolution principle. J. ACM 12, 1 (January 1965), 23-41.

18. Robinson, J. A. A review of automatic theorem proving. Proc. Symp. Appl. Math. 19 (1966), 1-18.

19. Slagle, J. R. Artificial Intelligence: The Heuristic Programming Approach. McGraw-Hill, New York, 1971.

20. Waldinger, R. J. and R. C. T. Lee. PROW: a step toward automatic program writing. Proc. Int. Joint Conf. on Artificial Intelligence (1969), 241-252.

21. Wos, L. T., D. F. Carson and G. A. Robinson. The unit preference strategy in theorem proving. AFIPS 25 (Fall, 1964). Spartan Books, Washington, 615-621.

Problem Statement

Prove the base angles of an isosceles triangle are equal.

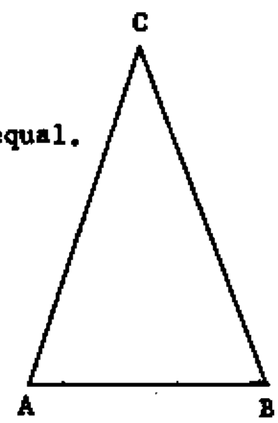
Prove: $\angle BAC = \angle ABC$

Given: $\overline{AC} = \overline{CB}$

$\triangle ABC$

Theorems:

1. $\triangle XYZ \cong \triangle UVW \rightarrow \angle XYZ = \angle UVW$
2. $\angle XYZ = \angle RST \wedge \angle UVW = \angle RST \rightarrow \angle XYZ = \angle UVW$
3. $\overline{XY} = \overline{UV} \wedge \overline{YZ} = \overline{VW} \wedge \angle XYZ = \angle UVW \rightarrow \triangle XYZ \cong \triangle UVW$
4. $\overline{XY} = \overline{UV} \wedge \overline{YZ} = \overline{VW} \wedge \overline{XZ} = \overline{UW} \rightarrow \triangle XYZ \cong \triangle UVW$



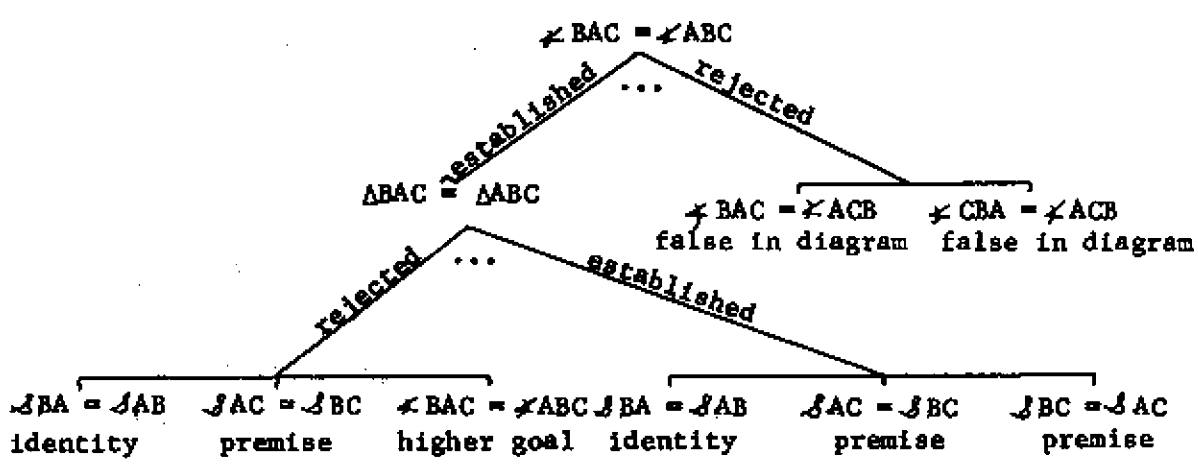


Figure 1

Problem Statement

I have a swimming pool.
If I have a swimming pool and it doesn't rain, I will go swimming.
If I go swimming, I will get wet.
If it rains, I will get wet.
Prove I will get wet.

A: I have a swimming pool.
B: I go swimming.
C: I get wet.
D: It rains.

Goal: 0. C

Assertions:
1. A
2. $A \wedge \sim D \rightarrow B$
3. $B \rightarrow C$
4. $D \rightarrow C$
[5. $\sim C \rightarrow \sim D$
(general contrapositive of 4)

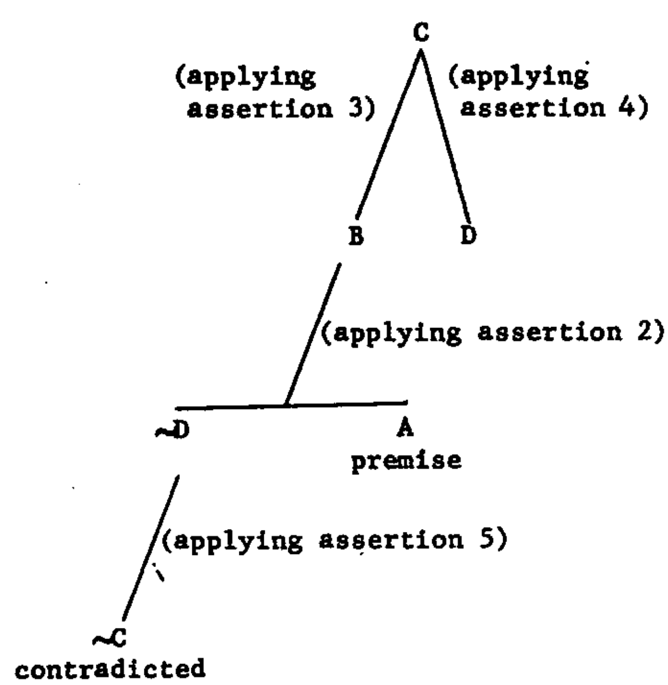


Figure 2

Goal:	0.	$b \geq 0$
Assertions:	1.	$a \leq b \wedge b \leq a \rightarrow a = b$
	2.	$a \geq 0$
	3.	$a \leq b$
	4.	$b \leq a$

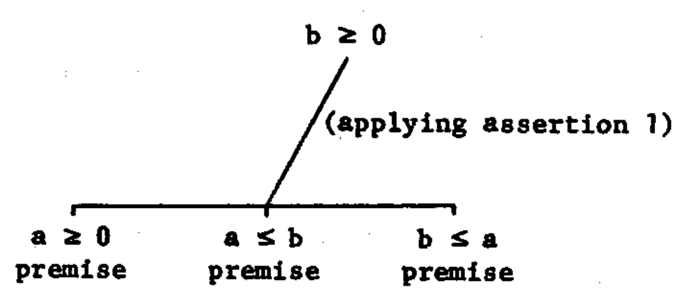


Figure 3

Goal: 0. $a \neq b$

Assertions: 1. $a + b = 2c \wedge a = b \rightarrow b = c$

2. $a + b = 2c$

3. $a \neq c$

[4. $a + b = 2c \wedge b \neq c \rightarrow a \neq b$ general contrapositive of 1]

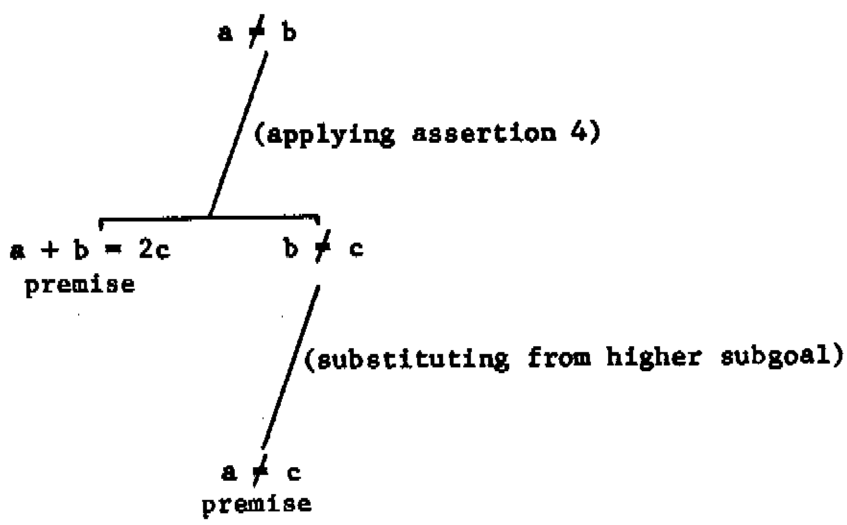


Figure 4

Goal:	0.	$a \neq b$
Assertions:	1.	$a > 0 \rightarrow a \geq 0$
	2.	$a > 0$
	3.	$b < 0$ i.e., $\sim(b \geq 0)$

$a \neq b$

(substituting into assertion 1)

$a > 0$ premise

$\sim(b \geq 0)$ premise

Figure 5

Goal: 0. A

Assertions:
1. B ∧ C → A
2. D → B
3. B → C
4. D

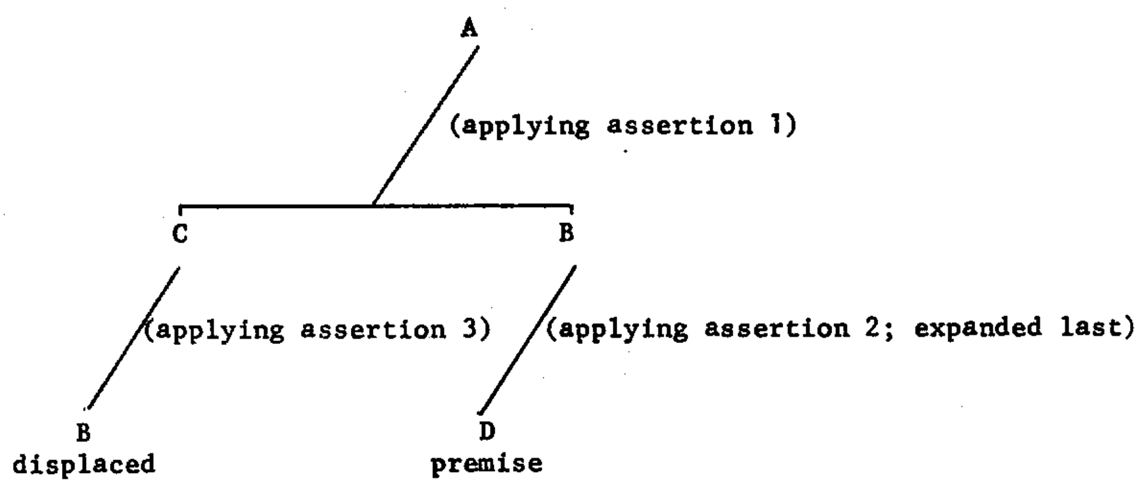


Figure 6

Goal: 0. C

Assertions: 1. A

2. ~A

[3. $\cancel{C} \rightarrow C$ assertion generated from 0]

[4. $\sim A \rightarrow \cancel{C}$ general contrapositive of 1]

C

(applying assertion 3)

$\cancel{C}$

(applying assertion 4)

~A

premise

Figure 7