FUNDAMENTAL STUDIES
IN COMPUTER SCIENCE 6

automated theorem proving: a logical basis

donald w. loveland

north-holland

Automated Theorem Proving: A Logical Basis

DONALD W. LOVELAND

Duke University,
Durham, North Carolina

1978

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM · NEW YORK · OXFORD

North-Holland ISBN for the Series: 0 7204 2500 X
North-Holland ISBN for this Volume: 0 7204 0499 1

Published by:
NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM · NEW YORK · OXFORD

Sole Distributors for the U.S.A. and Canada:
ELSEVIER NORTH-HOLLAND INC.
52 VANDERBILT AVENUE
NEW YORK, NY 10017

Library of Congress Cataloging in Publication Data

Loveland, Donald W
Automated theorem proving.

(Fundamental studies in computer science; 6)
Bibliography: p. 395
Includes index.
1. Automatic theorem proving. I. Title. II. Series.
QA76.9.A96L68 001.53'5 76-54345
ISBN 0-7204-0499-1

PRINTED IN HUNGARY

To my parents
My original and beloved instructors

PREFACE

The purpose of this book is to organize, augment when necessary, and record the major conceptual advances in an aspect of automated theorem proving that peaked during the decade of the 1960's. There were several reasons for this decade of intense activity: the general excitement of the subject matter, the discovery of a particularly attractive computer-oriented inferecne principle called the *resolution* principle, and the experimentation with automated theorem provers as providers of "intelligent" behavior within first-generation question-answering systems, robots and automated computer program-writing systems. The field itself is just two decades old, having been born in the mid-1950's with the arrival of digital computers on university campuses where the academicians could play with them. We include material spanning the full two decades; however, the center of gravity is the work done in the mid-to-late 1960's.

Admittedly, the thrust of investigation was one-sided. From a fixed format for the presentation of the problem (a particular normal form within the language of first-order logic), a variety of computer-oriented inference structures were developed. The goal was finding "natural" and "efficient" deduction schemes still general enough to handle any valid formula of first-order logic when in the proper format. Almost totally neglected was the problem of successful selection of a few promising deductions from among the many possibilities generated at each stage of execution. In part this was due to the difficulty of solving this selection problem. This is a prime research area for the 1970's and beyond, undoubtedly to be recorded by numerous books in the future.

It is the author's contention that the efforts expended on the computer-oriented inference systems to date provide a flexible basis upon which the future theorem provers will be built, using search control structures and other devices just beginning to be understood. Thus this book is written, not simply to report historically on an exciting intellectual excursion, but

also to propogate the acquired knowledge that may well form the logical bases of automated theorem provers to come.

The reader will readily note the author's bias towards a rigorous development of the material presented. This is characterized both by the full proofs of almost all stated theorems, and the attention to the analysis of the scope of the formats or procedures presented. However, it is clear that many readers are likely to be computer systems builders rather than mathematicians. These readers are expected to skip over many proofs, but are encouraged to read all theorem statements. Extensive discussions and the many examples are especially intended for the less mathematically-inclined reader.

Although this book is not a textbook in format it can be used in instructor-student settings quite easily. Some exercises of varied difficulty are incorporated in the text and the instructor will find it easy to compose routine exercises to test the understanding of a new format or procedure.

Anyone who has seen an axiom system in first-order logic, who knows the meaning of a valid formula and also has understood some proof by induction is adequately prepared to read this book. Part of Chapter 1 is devoted to the review of the basic concepts and notation from modern logic used in this book.

Chapter 1 includes the review just mentioned, a discussion of sample problem domains and the formulation of possible theorems, and the manner of translation of formulas to the standardized form we use. Section 1.1 gives a general orientation to the field and the book.

Readers who read the end of detective stories before the middle and artificial intelligence buffs are encouraged to read Section 6.1 before proceeding to Chapter 2. Here the traditional problem reduction method is examined, shown lacking as a full proof search structure, and the nature of the corrections given. This intuitive system is very different in appearance from the resolution procedures that are our main focus of attention. That a variant of resolution developed in earlier chapters can be used to indicate and confirm an appropriate extension to the problem reduction format, done in Section 6.2, is a demonstration of the robustness of the basic concepts behind the resolution variations. This reinforces our belief that the ideas, or the formats themselves, will be the basis of future theorem provers of general capability.

Chapter 2 introduces the basic resolution inference scheme, the primary inference structure that we study. Chapter 3 organizes and studies the important variants of resolution. These have been organized to unify concepts as much as possible; for example, clause-ordering rules are handled in a

uniform manner throughout. Also, the concepts of ordered clauses, settings or linearity are seen to underlie most known variants.

Chapter 4 contains a study of subsumption as it applies to the important variants of resolution. Much of this is new to the literature. Chapter 5 concerns the introduction of equality as a built-in relation. Some of this material is also new to the literature; in particular, the addition of equality to the model elimination procedure. In Chapter 6 the model elimination format is seen to relate to the problem reduction format.

Our stylistic conventions are generally familiar. We use "iff" for "if and only if". The symbol ▮ designates the end of proofs. Our definitions are introduced in either of two formats: embedded within the text itself or displayed. This is not meant to indicate relative importance, but more often reflects the complexity of the definition or the desire to break up a span of uninterrupted text.

We have made no attempt to present a full bibliography of the field. Instead, we list entries of direct relevance to the topics we consider in this book and include a reference to the most comprehensive bibliography on automated theorem proving known to this author, prepared in 1971 at the University of Maryland.

ACKNOWLEDGEMENTS

My appreciation extends to many more people than can be named here. Special thanks are due Michael O'Donnell, C. R. Reddy, Robert Daley and H. P. Edmundson for careful readings and criticism of the initial manuscript, the latter two using it in seminars on mechanical theorem proving. Mark Stickel contributed both criticism of the manuscript and substantial aid in the conception and structure of Chapter 6. Numerous typists helped prepare the manuscripts, with Dorothy Josephson, Sandy Thomas and Mary Kirkland undertaking major portions of the task.

The National Science Foundation and the Advanced Research Project Agency Grant to Carnegie-Mellon University provided research support, some results of which appear here for the first time in detail. A month's haven for writing was provided by Eugene Laska of the Information Sciences Division, Rockland Research Institute, Orangeburg, New York, with support shared by the National Institute for Mental Health and the New York State Department of Mental Hygiene. I am particularly indebted to North-Holland Editor Einar Fredriksson, whose initial encouragement and then patience were vital to the beginning and finishing of this book. Finally, many thanks are due my family for their patience during my long period of involvement with this book.

CONTENTS

CHAPTER 1

THE ROLE OF LOGICAL SYSTEMS

1.1. Orientation

The dream of a mechanical marvel that reasons on the level of human thought is a dream as old as the concept of machine. Not surprisingly, therefore, computer programs that prove theorems appeared as soon as electronic computers passed the prototype stage, in the early 1950's.

The interest in automated theorem proving stems from the ability to cast many of the tasks associated with human intellect as applications of theorem proving. An environment, be it a mathematical theory, a data bank of census data, or a physical environment for a robot, can be presented as axioms with the task goal as the asserted goal. (We discuss examples of this in this chapter.) The realization that powerful theorem proving techniques could provide a key component of many "intellingent machines" has drawn many computer scientists and mathematicians to the computer rooms to implement a theorem prover.

The first significant computer program for theorem proving was the Logic Theory Machine of Newell, Shaw, and Simon [NS1], which appeared in the mid–1950's. The problem domain under study was a particular formalism for the propositional calculus. They were interested in human problem solving techniques, and so devised a proof search organization distinct from the natural proof enumeration and from standard decision methods such as truth tables. One of the several important techniques that they introduced to the field of automated theorem proving was the technique of "working backwards" from problem goal to subproblem, a process we amplify shortly. The performance level was quite favorable relative to humans undertaking the same task, but only over a very modest portion of the domain of propositional calculus problems. The problem of mechanizing theorem proving was roundly attacked but certainly not demolished.

The Geometry Theorem Machine of Gelernter et al. [Ge1], [GH1], which followed in the late 1950's, was similar to the Logic Theory Machine in proof search organization but novel in its use of problem heuristics. A

heuristic is a rule-of-thumb, sometimes very useful but not always applicable. The theorem prover could prove a substantial class of the theorems considered in a high school plane geometry course. It employed the "diagram" that traditionally is offered with the problem statement at the high school level to narrow the alternatives in the proof search. The proof search worked backward from the goal, using a search form now often called the *problem reduction format*. This format is of interest to us in this book and is the subject of Chapter 6. We outline it here.

Given a statement of the problem, i.e. premises and a conclusion believed to follow logically from the premises, we seek theorems in our collection of established theorems with conclusions that match the problem conclusion. It suffices to satisfy the hypothesis of any such known theorem to establish the conclusion of the problem, because the truth of both the known theorem and its hypotheses establishes the truth of the theorem conclusion, which is also the problem conclusion. The hypotheses of the known theorem can be taken as new problem conclusions to be solved because their confirmation is seen to yield directly the intended conclusion. This inspires the name "problem reduction" since the problem goal (conclusion) is "reduced" to other goals, hopefully easier to establish. A reduction to a premise allows successful termination of that particular sequence of reductions.

The problem reduction method outlined above is one method of *proof search organization*. A proof search organization permits a framework in which the problem can be stated and a process executed that may lead to a signal (and "proof") that the conclusion follows from the premises. If success is never indicated unless the conclusion logically follows from the premises and rules of inference (here the established theorems), then the search organization system is said to be *sound*. If the search organization permits a theorem to be established whenever the conclusion logically follows from the premises and the asserted rules then the proof search organization system is called *complete*. We see in Chapter 6 that the above described problem reduction system is sound but not complete. (Admittedly, the terms have not been precisely defined here; they will be made precise later.)

Proof search organization is but one component of the design of a sophisticated automated theorem prover. A mechanism that will determine the direction of search development is also needed. We picture this in terms of the problem reduction method outlined above. A goal is matched with the conclusion of perhaps a large number of theorems initially, to generate all possible ways the goal could be achieved. This often generates a large

number of alternate subproblems of which only one (perhaps compound) subproblem need be established. Certain information may allow deletion of some of the alternatives. Indeed, the diagram served this purpose elegantly in the Geometry Theorem Machine by allowing deletion of any subproblem with a statement not true in the diagram. Other trimming rules also exist that depend on the previously generated subproblems, as we see later. These rules usually leave a sizable set of alternatives from which to select the more promising ones if further pursuit is necessary. The problem of determining an order of investigation by assignment of a priority order to alternate subproblems we call the *search control* or *search guidance* problem. The Geometry Theory Machine first used selection of subproblems by order of generation for those subproblems not removed by the diagram or other tests. A selection rule was introduced later based on the number of line segments implicitly and explicitly named in both the subproblem and the premises. This often realized a substantial speed-up in the proof search.

The conceptual distinction between search organization and search control should be emphasized, along with the realization that the boundary is fuzzy. Search organization has as its essence the proof search representation and record, largely a function of the chosen language and inference rules, of which many forms are possible. Search control or guidance is a selection process that converts an inherently nondeterministic process to a deterministic (sequential) process. (A small amount of parallelism does not essentially alter this characterization.)

We illustrate the separate roles of search organization and search control in terms of human problem solving. To establish that one can travel from Durham, North Carolina (U.S.A.) to Dunkirk, Nord (France), a simple depth-first "forward chaining" search organization is very appropriate. That is, at each stage we select a city, and then check that travel links exist between the previously selected city (which initially is Durham) and the newly selected city. This is iterated until the selected city is Dunkirk or travel links between "adjacent" cities cannot be established. In the latter case, an alternate selection is made and the process continues as described. This describes the mechanism of search organization. Search control, or guidance, involves the selection of the city from a set of alternatives, some subset of the cities of the world. A powerful heuristic is to insist that the selected city always be closer to Durham than Dunkirk is, unless that set of cities is exhausted. Other information, such as whether or not a city is an international port, is also useful selection information. From experience

in assembling travel routes, we know that the heuristics above often yield first-try success. Without such successful control superimposed on the chosen search organization, the depth-first organization might not be very "appropriate" at all. Indeed, for difficult mathematical theorems where heuristic guidance is weak, an organization directed towards parallel probing is more in order.

Trimming, or deletion, rules are properly part of the search control component since they aid in subproblem selection. However, to the extent that the subproblems deleted are logically redundant with respect to remaining subproblems, such rules are also part of the representation or search organization component. We need not remove such fuzziness, but rather accept such ambiguities.

In general, present automated theorem provers have weak search guidance. The design of strong guidance systems is very difficult, indeed beyond our present capabilities except for very small, highly structured domains. The apparent difficulty is that there is no alternative to the use of a large number of heuristics for search guidance, many of these heuristics of a quite specialized nature. Human problem-solving has this character. Such a structure is very hard to synthesize, however. The best work in this domain to date falls outside the formal theorem proving area, in the domain of checker and chess playing programs. Theorem proving researchers presently are addressing this component more seriously than they have in the past, but success undoubtedly will come slowly and in little pieces.

Two efforts in the 1960's deserve mention because they confronted the search control problem, in different ways. Norton [No1] devised a theorem prover for the theory of groups, which used primarily the problem reduction organization and included a fair number of heuristics for search control. Guard et al. [G01] developed a theorem prover where crucial search control was supplied by humans. Far from "cheating", this may be the manner in which search control is best realized, at least in the near future.*

In contrast with the difficulty in developing an understanding of the search guidance problem, our understanding of the proof search organization component has progressed quite well. It is this component that this book emphasizes. It is now appropriate to return to our mini-history of automated theorem proving.

* An open mathematical problem was solved using a result disconvered by their computer program. The program was in the theorem generation mode; it took a human to appreciate the value of one printed theorem among a collection of less valuable theorems.

In the late 1950's several logicians independently became interested in the challenge and potential of automated theorem proving. Gilmore [Gi1], Wang [Wa1], [Wa2], [Wa3], Davis and Putnam [DP1], Davis et al. [DL1], Dunham and North [DN1], and Prawitz [Pr1] represent some of the better known work of this period. These investigations were largely motivated by the conviction that classical mathematical logic was a good springboard from which to leap into this new research domain. First-order logics provide a universal language and semantics for everyday discourse and most of mathematics. Using these given qualities and appropriately designing the inference machinery, these logicians felt that progress beyond that directly implied by the Logic Theory Machine could be made by the proper search organization alone. Strong search guidance was postponed as a problem (although see Wang [Wa2]); a better search organization via tailored inference machinery was the first step. The argument for use of the universal language and semantics of first-order logics is that we should learn our capabilities in universal systems before moving to restricted systems. First-order logics provide the best understood "universal" deductive systems.

Valuable contributions appeared in this period. The use of Herbrand's results allowed translation of a first-order logic task to a propositional logic task (in a perhaps infinitary logic), which meant that quantifiers were explicitly handled only initially. An apparently efficient and reasonably natural process for checking tautologyhood appeared in the Davis–Putnam procedure. We consider this process in Chapter 2. Prawitz introduced to this discipline the notion of term matching, which in refinement plays a central role in the efforts on which we will focus.

Computer implementations showed improvement over what had been previously possible but success was largely confined to problems found in logic books rather than mathematics books. Investigation continued.

The design of inference systems for automated theorem proving received a big boost in 1965 with the publication of J. A. Robinson's paper on the resolution inference system [Ro1]. This provided a single inference rule and no requirement for logical axioms. Of course, axioms associated with the specific problem (proper axioms) are needed. The inference rule is extremely simple propositionally (the "cut" rule in essence) with an encompassing substitution mechanism attached. Its simplicity makes it often the easiest way to hand check a simple formula for validity (or refutability as it is actually formatted).

Although elegant, the basic resolution procedure usually produces intermediate clauses, in effect "subproblems", at an overwhelming rate.

Other investigators sought restricted forms of resolution that would also permit eventual detection of any valid formula (the *completeness property*) yet be more restrictive and, hopefully, more efficient in representation. It is primarily the basic resolution procedure and some important variants of resolution that we consider in this book.

The variety of resolution refinements is amazing, and a tribute to the richness of the basic procedure. We make no claims to present all known resolution refinements; indeed, new forms are still being discovered. The refinements known in the early 1970's that the author believes are conceptually interesting in format are included, and organized to emphasize key characteristics. The characteristics are important both as unifying concepts and as possible tools for search control mechanisms.

Because the equality relation is so significant a part of mathematics, resolution procedures have been extended to incorporate equality as part of the inference. We consider such extensions. We do not include more recent work that also incorporates other relations, such as partial ordering. (The interested reader may consult Slagle and Norton [SN1].)

It is our convinction that the importance of the resolution variants is in the flexibility of representational formats offered which in some form or forms may provide the appropriate search organization for future theorem provers. We do not believe that the precise formats treated here necessarily are the forms eventually to be used, but believe that many of the concepts considered here — unification, unit preference, clause ordering including specified settings (interpretations), linearity — will appear in the ultimate proof search organizations. For this reason we have chosen to omit results of implementations, not because they are not important (indeed, they are the ultimate evaluators) but because they are by nature dated. Addition of one search control mechanism of value will significantly alter performance in its domain of application. Likewise, we do not dwell on the method of application of theorem provers to question-answerers or other practical mechanisms, although limited discussion is included. We do present a conceptual application: to illustrate how a variant of resolution can provide insight into an apparently unrelated search organization system, we relate a resolution variant to the problem reduction method outlined earlier. This is a particularly significant illustration because some investigators who have "rejected" the resolution format imbed the problem reduction format within their devised inference system.

Because we deal with the search organization component, the soundness and completeness of the procedures studied is important. Soundness will

usually come easily. As we constrain the inference capability, it is the completeness that is endangered and that requires effort to verify.

Consideration of the value of completeness is appropriate since a fair portion of the book is concerned with this matter. First, everyone recognizes that a complete theorem proving system is meaningless because every proof search stops when time or space (or the machine) is exhausted. Moreover, no matter how much compression of time and memory space a super technology achieves, we know that we face basic limitations on the capabilities of complete procedures. For example, although procedures exist to determine the truth value of any arithmetic statement invoking only the addition and subtraction functions, any such procedure must be overwhelmed when given certain relatively short statements; see Rabin [Ra1].

Given this reality, many people have recently pointed out that we should not focus on complete systems, but should design systems tuned to perform well in small problem domains at first, and then expand the domains as we master further techniques. We can have incomplete, complete and redundantly complete systems, depending on what experimentation determines is best.

It is indeed appropriate to narrow our sights to specific problem domains and to shift our primary attention to the search control problem. It is also appropriate to use highly redundant yet incomplete inference systems when such systems are effective. But we should note that such statements consider the overall structure of complex systems. One should not assume that each component of the system uniformly adopts the characteristics of the overall system. We consider briefly the likely structure of the future automated theorem provers we seek.

Theorem provers will undoubtedly have a hierarchy of processes defining the search control and the search organization structure. For the search control, the lowest layer is composed of evaluation functions on subproblems, comparing the statement against established heuristic values, such as name comparisons against established facts or premises. Higher level heuristics measure the function values in the context of a plan, and a yet higher level structure continually reassesses the current search plan in the light of new discoveries via generated results or after assessing overly lengthy probes. We note that each level builds on the levels beneath it.

The proof search organization component, consisting of the inference system and the search representation framework, also can be layered. In particular, the inference system may be partly in the search representation framework, and partly in a data base of inference rules. (This interplay