A note on the analysis of theorem-proving strategies

Maria Paola Bonacina *
Department of Computer Science
University of Iowa
Iowa City, IA 52242-1419, USA
bonacina@cs.uiowa.edu

Abstract

In recent work David Plaisted proposed an approach to analyze the search efficiency of theorem-proving strategies, and applied it to Horn propositional logic. In this note we comment on this approach. We point out a number of problematic issues in the modelling of search and the analysis of strategies, including the formalization of the search plan, the representation of contraction, and the duality of forward and backward reasoning. These issues are especially relevant for the analysis of strategies in first-order logic, where the search space is infinite.

1 Introduction

The evaluation of theorem-proving strategies has been done traditionally in an empirical manner: a strategy is implemented in a theorem prover, the prover is applied to a number of theorem-proving problems, and the running times are reported and possibly compared with those of existing theorem provers. In recent years, a growing effort has been devoted to make the empirical evaluation of theorem provers more systematic. The need for a standard collection of theorem-proving problems and a standard set of empirical measures has been recognized. The first requirement was met in part with the creation of the TPTP library [23]. The second issue is currently debated in the theorem-proving community, with different measures, involving running time, number of theorems proved, and similar data, being considered for different purposes [22].

While benchmarking of theorem provers is necessary, and the current progress in the methodology of empirical evaluation is of critical importance for the field, the problem of *strategy evaluation* remains open. A theorem-proving system is made of many components in addition to the theorem-proving strategy, including for instance data structures, indexing techniques and service algorithms such as those for unification or term replacement. The performance of a theorem prover depends on all these components, and, furthermore, on the overall engineering of the system. It is very difficult to establish quantitatively how different components contribute to the observed performance. Therefore, empirical evaluation is evaluation of theorem-proving systems, not theorem-proving strategies. The goal of evaluating strategies independent of implementation

*Supported in part by the National Science Foundation with grant CCR-94-08667.

requires the development of a theory of "strategy analysis," comparable to algorithm analysis, and with potentially similar beneficial consequences.

David Plaisted recently presented an approach to strategy analysis [20], which studies the search efficiency of theorem-proving strategies applied to Horn propositional logic. In this paper we comment on this approach. The emphasis is on the method, not on the results. We use the framework of [20] as a starting point to discuss methodological issues that need to be addressed in order to analyze theorem-proving strategies. Our intent is to call attention to these issues, highlighting open problems. An alternative approach to strategy analysis may be found in [5, 4], where it is applied to analyze the effect of contraction on the search space. Since the present paper concentrates on method, and not on specific analyses, it is self-contained. This note is organized in four sections: preliminary notions, a summary of the approach of [20], the critical comment, and final remarks.

2 Basic notions

A theorem-proving strategy is made of an inference system and a search plan. The inference system defines the possible inferences, and the search plan selects at each step of a derivation the inference rule and the premises for the next step. We distinguish *contraction-based strategies* and *subgoal-reduction strategies*. The first group includes strategies based on resolution, Knuth-Bendix completion and term rewriting (e.g., [3, 6, 10, 21] and [9] for a survey). The second group includes model elimination [15], and therefore Prolog technology theorem proving (e.g., [2]), the MESON strategy [16], and the problem-reduction-format strategies [18]. Tableau-based methods also belong to the latter group, since they can be interpreted in terms of model elimination (e.g., [14, 25]). Clause linking [13] reduces to the Davis-Putnam procedure on propositional problems. In this section we recall the basic characterizations of these strategies, because they are relevant to the following discussion.

Contraction-based strategies work primarily by forward reasoning. They use *expansion rules* such as resolution to generate consequences from existing clauses. Initially, the existing clauses are those obtained by transforming in clausal form the axioms of the problem and the negation of the target theorem. Forward-reasoning methods do not generally distinguish goal clauses from the others, seeking to obtain a contradiction - the empty clause - from the whole set. Generated clauses are kept and used for further generations, so that the strategy works on a *database of clauses*. The major source of redundancy for these strategies is that they generate and retain clauses that do not contribute to proving the theorem. Contraction-based strategies counter this problem by using *contraction rules*, such as simplification and subsumption, to delete redundant clauses. Since the inference system typically features multiple inference rules, and generated clauses are kept, creating many possible choices of premises, the search plan plays an important role. Forward-reasoning strategies admit a variety of search plans, which sort the clauses in the database by different criteria or orderings. Contraction-based strategies employ *eager-contraction* search plans, that give priority to contraction over expansion, in order to maintain the database minimal with respect to the contraction rules.

Subgoal-reduction strategies work by reducing the goal to subgoals. Each inference step involves the current goal and at most another premise. The current goal is initially an input clause, usually a goal clause. Then, it is the most recently generated goal. If no rule applies to the most recently generated goal, the strategy backtracks to the previous current goal. Since generated clauses (goals) are retained only for the purpose of backtracking, the strategy does not form a database of clauses, and works on a *stack of goals*. Since there is no database, there is no contraction, and the search plan is fixed to be *depth-first search with iterative deepening* [11]. The major source of redundancy for these strategies is that by focusing only on the most recently generated goal, with no memory of previous steps, they reduce repeatedly the same subgoals. Subgoal-reduction methods counter this problem by using techniques for *lemmaizing* or *caching* (e.g., [2, 7, 14, 15, 19, 25]), that enable them to keep track of already solved or failed subgoals.

3 Plaisted's approach to the analysis of strategies

Most of the research in complexity of theorem proving studies the complexity of propositional proofs (e.g., see [24] for a survey), while most of the work on search concentrates on the design of heuristics (e.g., [17]). The classical source for the modelling of search in theorem proving is [12], which, however, was not concerned with evaluating the complexity of the strategies. The object of [20] and this note is the complexity of *searching for a proof*. Thus, the first step is to have a *model of the search space* (the realm of possibilities) and search process. The second step is to define *measures of complexity* in such a model, so that strategies can be evaluated and compared in terms of such measures. The third step is to carry out the analysis itself. In this section we extract the relevant elements from [20] according to this view.

3.1 The model of the search space

A theorem-proving strategy is formalized in [20] as a 5-tuple $< S, V, i, E, u >$, where S is a set of states, and E is a set of pairs of states, in such a way that $< S, E >$ forms a directed graph. The component V is a set of labels for the states in S. The component i is a mapping from the set of input clauses to S, which identifies the initial state(s). The component u is a function from S to $\{true, false\}$, where $u(s) = true$ if and only if s is a state where unsatisfiability is detected. It is assumed that u is an almost trivial test, such as recognizing that a set of clauses contains the empty clause. A more precise definition of V and u depends on the specific strategy. A function *label*, such that $label(s)$ denotes the label of state s, is also used. It is required that no two distinct edges (s_1, t_1) and (s_2, t_2) in E have $t_1 = t_2$. This implies that the directed graph is a set of *trees*. A strategy is said to be *linear* if for all states s, there is a unique state t such that $(s, t) \in E$, that is, every state has unique successor.

Given a strategy $G =< S, V, i, E, u >$ and an input set of clauses R, a state $s \in S$ is *reachable* from R if there is a path from a state in $i(R)$ to s. $S(R)$ denotes the subset of S containing all states that are reachable from R, and $E(R)$ denotes the restriction of E to $S(R)$. Then, $G(R) =< S(R), E(R) >$ is the search space for the theorem-proving problem R according to the strategy G.

3.2 The application of the model to the strategies

For resolution, the labels of the states are finite sets of clauses. If R is the input set of clauses, and $s_0 \in S$ is the state with label R, the input function $i\colon R \to S$ is the function such that $i(\psi) = s_0$ for all $\psi \in R$. In other words, all input clause are mapped to a single initial state, whose label is the set of input clauses. It follows that $G(R)$ is *a tree* with R as the label of the root. An arc connects state s to state t if the label of t is the union of the label of s and all the resolvents in s according to the inference system. This means that each state has a unique successor, that is, the tree $G(R)$ for resolution degenerates to *a list*. Accordingly, all resolution-based strategies are *linear* in [20]. The function u is defined by $u(s) = true$ if and only if the label of s contains the empty clause.

For model elimination, each state is labelled by a single chain[1], which is the current goal in that state. Equivalently, we can think of a label as a singleton set containing one chain. If $R = \{\psi_1, \ldots, \psi_n\}$ is the input set of clauses, there are states $s_1, \ldots, s_n$ with labels $\{\psi_1\}, \ldots, \{\psi_n\}$, respectively, and $i(\psi_j) = s_j$, for all $1 \leq j \leq n$. In other words, there is an initial state for each input clause. This captures the fact that any input clause can be chosen as the initial chain. It follows that $G(R)$ is *a set of trees*, one for each possible initial chain. An arc connects state s to state t if the chain labelling t is generated from the chain labelling s in one ME-step (ME-extension or ME-reduction or ME-contraction [15, 16]). The function u is defined in the same way as for resolution. The strategies based on model elimination are *not linear*, because an arc represents a single step, and more than one step may be applicable to a chain (e.g., ME-extension steps with different input clauses).

3.3 The complexity measures

The measures of complexity of search proposed in [20] aim at measuring *the total size of the search space*. The total size of $G(R)$ is defined as $||G(R)|| = \sum_{s \in S(R)} |label(s)|$. For resolution, the labels are finite sets of clauses, and therefore $||G(R)||$ is the sum of the cardinalities of the sets of clauses labelling the nodes of $G(R)$. For model elimination, the labels are singleton sets, so that $||G(R)||$ is equal to the number of nodes in $G(R)$.

The measure $||G(R)||$ is refined further into three measures, called *duplication by iteration*, *duplication by case analysis* and *duplication by combination*. Duplication by iteration is the maximum length of a path in $G(R)$. Duplication by case analysis is the maximum size of a subset of $S(R)$ no two elements of which are on the same path. Duplication by combination is the maximum cardinality of a label of a state in $S(R)$, i.e., $max_{s \in S(R)} |label(s)|$. Since $G(R)$ is a tree or a set of trees, duplication by case analysis reduces to the number of paths, and $||G(R)||$ is bounded by the product of duplication by iteration, duplication by case analysis and duplication by combination. The analysis proceeds by establishing whether these measures are exponential or linear or constant in the length of the input set R read as a single string.

[1] A *chain* is a sorted clause with plain literals, called B-literals, and framed literals, called A-literals for ancestors [15, 16].

3.4 The analysis

According to the analysis in [20], basic resolution strategies have linear duplication by iteration and exponential duplication by combination. The duplication by case analysis is trivially constant (more precisely, equal to 1), because each state has unique successor. The exponential duplication by combination captures the complexity of generating and keeping clauses, since resolvents are generated by combining in different ways the input literals. Since one of the measures is exponential, resolution strategies are regarded as inefficient. For positive hyperresolution, however, the duplication by combination is linear, because positive hyperresolvents in Horn logic are unit clauses, so that all generated clauses are unit clauses, and literals are not combined. Also, the duplication by combination of positive resolution reduces from exponential to linear if the strategy is equipped with an ordering on predicate symbols, and only the negative literals with smallest predicate are resolved upon. In summary, resolution strategies are either efficient, but not goal-sensitive (e.g., positive hyperresolution and positive resolution), or goal-sensitive, but not efficient (e.g., negative resolution), or neither efficient nor goal-sensitive (e.g., ordered resolution). These results are worst-case results. For instance, ordered resolution is efficient for some sets of clauses and orderings (e.g., all well-ordered sets[2]), but there are sets for which an ordering that makes the strategy efficient cannot be found. The same is true for goal-sensitivity.

Model elimination has exponential duplication by iteration and exponential duplication by case analysis. The duplication by combination is trivially constant (equal to 1), because each state is a singleton. The exponential duplication by iteration captures the redundancy of solving subgoals repeatedly. Duplication by case analysis is also exponential because a state may have multiple successors. Thus, resolution and model elimination are sort of dual: resolution has low duplication by case analysis and iteration, but high duplication by combination, and vice versa for model elimination.

If model elimination is enriched with unit lemmaizing[3], duplication by iteration becomes linear, because lemmaizing prevents solving repeatedly the same successful subgoals. Lemmatization adds a forward-reasoning character to the strategy, because lemmas are generated and retained. In the framework of [20] this means exponential duplication by combination and constant duplication by case analysis, so that model elimination with lemmaizing has the same duplication as the resolution strategies. In Horn logic model elimination can be enhanced with caching[4]. The use of caching further reduces the duplication, because not only successful subgoals (success caching), but also failed subgoals (failure caching) are not repeated. Assuming depth-first search with iterative deepening (and therefore depth-dependent caching, e.g. [2, 7]), duplication by combination becomes linear (more precisely, quadratic). Similar results apply to the other subgoal-reduction strategies. In summary, subgoal-reduction strategies are inefficient, but goal-sensitive. Subgoal-reduction strategies with caching are efficient and goal-sensitive.

[2]A set of Horn clauses is *well-ordered* if there is a partial ordering $<$ such that if $P :- P_1, \ldots, P_n$ is a clause in the set then $P_i < P$ for all i, $1 \leq i \leq n$.

[3]In Horn logic all lemmas generated by lemmaizing are unit lemmas.

[4]Caching is not consistent with ME-reduction, which is necessary for the completeness of model elimination in first-order logic (e.g., [2, 7]). In first-order logic one may use lemmaizing or more complicated caching schemes (e.g., [1, 7, 25]) that are not analyzed in [20].

4 Discussion of Plaisted's approach

In this section we point out and discuss a few aspects of the analysis in [20]: the omission of the role of the search plan, the absence of contraction, the problem of encompassing properly contraction-based and subgoal-reduction strategies in a model of search, and the choice of worst-case analysis. The comment is written having first-order logic in mind, since this is also the final purpose of the approach of [20].

4.1 The role of the search plan

The first remark to the model of search in [20] is that it ignores the search plan. For resolution-based strategies, the search plan chooses the parents of the next resolution step among the clauses in the database. In the model of search in [20] neither the search plan itself nor its application are represented. For resolution, *even the existence of multiple choices is not represented*, because each state has a unique successor defined by the addition of all resolvents.

The choice of not including the search plan in the model is motivated in [20] by the intent of *measuring the total size of the search space* generated by an inference system, not the behaviour of specific search plans. (It is remarked that the total size of the search space is also relevant for concrete search plans that are exhaustive, such as breadth-first search and depth-first search with iterative deepening.) However, omitting the search plan is not appropriate for a general framework for strategy analysis:

1. The total size of the search space can be measured only if the search space is *finite*, as in the propositional problems considered in [20]. For *infinite* search spaces, such as those of first-order problems, the total size cannot be measured, and the role of the search plan is even more important. Thus, a model of search needs to make the analysis of the behaviour of specific search plans possible.

2. The total size of the search space depends on the inference system, not on the search plan. Measuring the total size of the search space and omitting the search plan in the formulation of search problems imply that it is not possible to differentiate between strategies that have the same inference system but different search plans, even in finite search spaces.

3. Last and most important, in order to do strategy analysis, it is not sufficient to have a representation of the search space as a *static* space, given prior to the derivation, because in the general case such space is infinite and cannot be measured. In order to deal with infinity, we need to represent the *search process*, or the evolution of the search space during the derivation. For this purpose, the model needs to include the search plan as an explicit component, and to represent the effect of its selections on the search space.

4.2 The problem of contraction

The formalism of [20] does not include contraction inferences as a basic element. Inferences are represented by the arcs in the directed graph, and the arcs considered in [20] cover only expansion

inferences, such as the addition of resolvents, or the generation of a successor chain from a chain in model elimination. The effect of contraction inferences does not appear in the representation of the search space. The choice of representing expansion inferences by arcs that add all resolvents is also an obstacle to including contraction in the model. For instance, the search space of a strategy that applies subsumption after every resolution step cannot be represented in this model, because subsumption by a resolvent may delete the parent clauses of other resolution steps that were originally enabled.

It follows that the analysis in [20] first considers all resolution-based methods as expansion-only strategies. Results for strategies with subsumption and clausal simplification are obtained by modifying the results for the expansion-only versions of the strategies. For some strategies with specific restrictions, and because of the simplicity of Horn logic, it is sometimes possible to capture the effect of contraction even if it is not represented in the model. One such instance is positive resolution: since in Horn logic positive clauses are unit clauses, each positive-resolution step generates a resolvent that subsumes its non-unit parent. It follows that if a positive resolution strategy applies subsumption after each resolution step, the duplication by combination reduces from exponential to linear. For most strategies, the analysis of contraction in [20] consists in showing that subsumption and clausal simplification do not apply in the sets of clauses used to establish the worst-case results. This is the case for the parametric sets S_n^2 and A_n that give the exponential upper bounds on duplication by combination for negative resolution and ordered resolution, respectively. This way of reasoning cannot be extended to first-order logic and equational logic, where contraction is most useful, because the search space is *infinite*, and worst-case analysis does not apply (see Section 4.4).

Since contraction may make a great difference in practice, the analysis of strategies needs to explain such observations from the experiments. Contraction makes the search space *dynamic*, by deleting clauses. Therefore, the analysis cannot be done in a framework where the search space is static. In this sense, the issue of contraction is related to the issue of modelling search as a dynamic process of Item 3 in Section 4.1.

4.3 The treatment of contraction-based and subgoal-reduction strategies

The search space of subgoal-reduction strategies has been traditionally represented as a tree (e.g., AND/OR-tree). The nodes are labelled by the goals (e.g. the chains of model elimination), and the input clauses are viewed as "operators" that may be applied to reduce the goals, rather than as part of the description of the state. The treatment of model elimination in [20] follows this pattern.

The representation of the search space of contraction-based strategies, however, is more problematic. The representation in [20] depends dramatically on the stipulation that an arc represents the addition of all the resolvents. If we drop this convention, and let an arc represent a single resolution inference, as for model elimination, the search space is neither a list nor a tree, but a general graph, because each state has many successors and a state may be reached through different paths. The choice of [20] of associating to an arc the generation of all resolvents, and collapsing the search space of resolution to a list, has several consequences:

1. The non-uniform interpretation of the elements of the formalism: an arc represents an inference for model elimination and all the possible inferences in the given state for resolution.

2. The counterintuitive result that resolution is linear, whereas model elimination is not.

3. The difficulty with strategies interleaving expansion and contraction as pointed out in Section 4.2.

4. The duality of the measures of duplication: all resolution strategies have constant duplication by case analysis; all model elimination strategies have constant duplication by combination; no strategy has both duplication by case analysis and duplication by combination.

In summary, the search space formalism of [20] is modelled after an idea of theorem proving which is essentially subgoal reduction:

1. The tree structure, which is natural for subgoal-reduction strategies, but not for contraction-based strategies, is assumed already in the general characterization of a strategy as a tuple $< S, V, i, E, u >$. Then, the search space of subgoal-reduction strategies is represented properly as a set of trees, while the search space of resolution is reduced to a list.

2. Resolution has a very high degree of non-determinism, because of all the possible selections of clauses in the database. However, it is represented as a completely deterministic strategy. On the other hand, the "degrees of freedom" of model elimination (e.g., choice of input clause for ME-extension steps[5] and choice of initial chain) are represented (e.g., a state may have multiple successors and there is a tree for each choice of initial chain).

3. If the existence of multiple choices is not represented in the model, there is no need for the search plan. Indeed, the search plan, which is fundamental for contraction-based strategies, is not included in the model. This does not affect the analysis of subgoal-reduction strategies, because their search plan is fixed to be depth-first search with iterative deepening.

4. The positive effect of lemmaizing and caching in reducing the duplication of subgoal-reduction strategies is successfully captured.

5. On the other hand, contraction, which counters the duplication of forward reasoning, and is absent in subgoal-reduction strategies, is not included in the model.

One could motivate the approach of [20] by observing that subgoal reduction is quite natural in Horn logic. However, while the analysis is applied in [20] to Horn propositional logic, the basic ideas on the modelling of search and on the nature of the complexity in theorem proving are intended for first-order logic. An additional evidence of this is that SLD-resolution is treated like all other resolution methods, even if SLD-resolution is a subgoal-reduction strategy for Horn logic. A plausible reason is that in first-order logic linear resolution [8] is not an input strategy, because it needs to include resolutions with the ancestors of the current goal. It follows that input

[5]ME-contraction and ME-reduction are deterministic rules, since ME-contraction simply removes the leftmost literal of a chain if it is an A-literal, and ME-reduction removes the leftmost literal of a chain if it is a B-literal that unifies with an A-literal of opposite sign in the chain.

clauses and generated clauses (the ancestors) are part of the state. On the other hand, model elimination is an input strategy also in first-order logic, and the state only needs to contain the current chain. This difference, however, emerges only in first-order logic, because linear input resolution is complete for Horn logic [8]. Thus, the basic choices on how to represent the search space of strategies in [20] are made having first-order logic in mind.

It seems that the approach of [20] implicitly envisions resolution as a subgoal-reduction strategy where the whole database of clauses is the goal (to be reduced to the empty clause) and the only operator is adding all resolvents. The defining features of methods that are not subgoal reduction strategies may not be captured and analyzed properly in a framework which casts all strategies in the mold of subgoal reduction. Strategy analysis needs a common framework which is sufficiently rich to capture different classes of strategies.

4.4 The type of analysis

The analysis in [20] is worst-case analysis. Working by worst-case analysis means trying to exhibit special sets of clauses where a strategy performs poorly. This raises the question of how significant these special sets, and therefore worst-case analysis, are in practice.

A first issue is that worst-case sets may display regularities of structure, such as symmetries, or even recurrence relations, that may not appear in real theorem-proving problems. The sets used in [20] to establish worst-case results incorporate recurrence relations on the indices of the literals in the clauses. The recurrence relations have exponential solution, and this is used in the proofs of the exponential behaviour of the strategies. It is suggested in [20] that these negative patterns occur often in real derivations, but are not detected by the experimenters because of the very high number of clauses generated by the theorem provers. This claim, however, may be difficult to verify for the same reason, and concrete examples are not reported.

A second issue is that in the quest for worst-case sets one may consider sets of clauses that are hard from a combinatorial point of view, but easy in practice. This is the case for the parametric sets S_n^2 and A_n used in [20] to defeat negative resolution and ordered resolution, respectively. S_n^2 is satisfiable and contains no positive clauses, so that positive hyperresolution or positive resolution would not apply a single inference and establish that the set is satisfiable in the time needed to read it. Similarly, A_n is satisfiable and contains neither positive nor negative clauses, so that it is trivial also for negative resolution.

One may suggest that satisfiable sets should not be considered. On the other hand, one may object to excluding them because satisfiable sets may occur in practice if the theorem is not true, or it has been misformulated. More importantly, the issue is whether worst-case analysis is appropriate for theorem proving, or whether it may be misleading, and emphasize combinatorial patterns in lieu of real difficulty. In addition, worst-case analysis cannot be extended to infinite search spaces. Therefore, the general study of theorem-proving strategies needs to develop different forms of analysis.

5 Summary

In this note we have commented on the approach to the analysis of theorem-proving strategies that was presented in [20]. The method of [20] applies to theorem proving the tools of classical algorithm analysis (e.g., finite search space, total size of the tree as the measure of complexity, worst-case analysis). We have pointed out a number of issues that are left open by this type of approach, including infinite search spaces, search as a dynamic process, search plans, contraction inferences, and an analysis other than worst-case analysis. In [5, 4] we have proposed an alternative approach to the modelling of search that works with infinite search spaces, contraction inferences and a different notion of complexity. More problems remain open, as the field of strategy analysis is only at the beginning.

Acknowledgements

I would like to thank David Plaisted, for answering my questions on his paper, and Jieh Hsiang, for many discussions on Plaisted's work.

References

[1] O. L. Astrachan. *Investigations in theorem proving based on model elimination*. PhD thesis, Dept. of Computer Science, Duke University, 1992.

[2] O. L. Astrachan and M. E. Stickel. Caching and lemmaizing in model elimination theorem provers. In D. Kapur, editor, *Eleventh Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 224–238. Springer Verlag, 1992.

[3] L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In M. E. Stickel, editor, *Tenth Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 427–441. Springer Verlag, 1991.

[4] M. P. Bonacina and J. Hsiang. On the notion of complexity of search in theorem proving. *Logic Colloquium*, San Sebastián, Spain, July 1996. Bulletin of Symbolic Logic, to appear.

[5] M. P. Bonacina and J. Hsiang. On the modelling of search in theorem proving – towards a theory of strategy analysis. Technical Report 12/95, Dept. of Computer Science, University of Iowa, 1995.

[6] M. P. Bonacina and J. Hsiang. Towards a foundation of completion procedures as semidecision procedures. *Theoretical Computer Science*, 146:199–242, 1995.

[7] M. P. Bonacina and J. Hsiang. On semantic resolution with lemmaizing and contraction. In N. Foo and R. Goebel, editors, *Fourth Pacific Rim Int. Conference on Artificial Intelligence*, volume to appear of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1996.

[8] C. L. Chang and R. C. Lee. *Symbolic logic and mechanical theorem proving*. Academic Press, New York, 1973.

[9] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of theoretical computer science*, volume B, pages 243–320. Elsevier, 1990.

[10] J. Hsiang and M. Rusinowitch. Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method. *Journal of the ACM*, 38:559–587, 1991.

[11] R. E. Korf. Depth-first iterative deepening: an optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[12] R. Kowalski. Search strategies for theorem proving. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 181–201. Edinburgh University Press, 1969.

[13] S.-J. Lee and D. A. Plaisted. Eliminating duplication with the hyperlinking strategy. *Journal of Automated Reasoning*, 9(1):25–42, 1992.

[14] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: a high performance theorem prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

[15] D. W. Loveland. A simplified format for the model elimination procedure. *Journal of the ACM*, 16(3):349–363, 1969.

[16] D. W. Loveland. *Automated theorem proving: a logical basis.* North Holland, New York, 1978.

[17] J. Pearl. *Heuristics – Intelligent search strategies for computer problem solving.* Addison Wesley, Reading, MA, 1984.

[18] D. A. Plaisted. A simplified problem reduction format. *Artificial Intelligence*, 18:227–261, 1982.

[19] D. A. Plaisted. Non-Horn clause logic programming without contrapositives. *Journal of Automated Reasoning*, 4(3):287–325, 1988.

[20] D. A. Plaisted. The search efficiency of theorem proving strategies. In A. Bundy, editor, *Twelfth Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 57–71. Springer Verlag, 1994. Full version: Technical Report MPI I-94-233.

[21] M. Rusinowitch. Theorem-proving with resolution and superposition. *Journal of Symbolic Computation*, 11(1 & 2):21–50, 1991.

[22] G. Sutcliffe and C. Suttner. The design of the CADE-13 ATP system competition. Technical Report 95/15 and AR-95-05, Dept. of Computer Science, James Cook University, Australia and Institut für Informatik, TU Munchen, Germany, 1996.

[23] G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In A. Bundy, editor, *Twelfth Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 252–266. Springer Verlag, 1994.

[24] A. Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1:425–467, 1995.

[25] K. Wallace and G. Wrightson. Regressive merging in model elimination tableau-based theorem provers. *Journal of the IGPL*, 3(6):921–937, 1995.