Theorem proving strategies: a search-oriented taxonomy*

(Position paper)

Maria Paola Bonacina†
Department of Computer Science
The University of Iowa

There are many ways of classifying theorem-proving strategies. In proof theory, one may question whether the strategy is *analytic* (i.e., it only generates formulae that are subformulae of the given problem $H \supset \varphi$, where H is the set of assumptions and φ the conjecture) or *generative* (i.e., not analytic). From the point of view of the language and its expressive power, one may be interested in whether the strategy works with equations, clauses, or sentences. From the point of view of the logic and its applicability, one may consider whether the strategy works for propositional logic, Horn logic, first-order logic, or higher-order logics. This talk presents a taxonomy of strategies, for fully-automated, general-purpose, first-order theorem proving, based on *how they search* [1].

Motivations for being interested in such a classification come from a variety of research problems, from parallelization of theorem proving (parallelism affects the control of theorem proving, and therefore its search aspect), to machine-independent evaluation of theorem-proving strategies (an analysis of strategies needs to analyze the search processes they may generate), and the engineering of theorem provers (theorem-proving methods are often specified in terms of inference rules only, and a significant amount of knowledge about search in theorem proving remains hidden in the code of the implementations, partly because of the lack of formal tools to discuss search in theorem proving).

The primary classification key in this taxonomy is to distinguish between those strategies that work on a set of objects (e.g., clauses) and develop implicitly many proof attempts, and those strategies that work on one object at a time (e.g., a goal clause, or a tableau) and develop one proof attempt at a time, backtracking when the current proof attempt cannot be completed into a proof. The strategies in the first group, on the other hand, never backtrack, because whatever they do may further one of the proof attempts.

The strategies in the first group are called in this taxonomy *ordering-based strategies*: exactly because they work with a set of objects, they use a *well-founded ordering* to order the objects, and possibly *delete* objects that are greater than and entailed by others. Thus, these strategies work by generating objects, *expanding* the set, and deleting objects, *contracting* the set. Also, since the set may grow very large, they employ *indexing techniques* to retrieve objects, and *eager-contraction* search plans to control the growth. The family of ordering-based strategies can be subdivided further into *expansion-oriented strategies*, *contraction-based strategies*, including in turn *target-oriented strategies*, and *semantic or supported strategies*. The strategies resulting from the merging of the resolution-

*Research supported in part by the National Science Foundation with grant CCR-97-01508.

†Dept. of Computer Science, MacLean Hall 15, University of Iowa, Iowa City, IA 52242-1419, U.S.A., bonacina@cs.uiowa.edu

paramodulation paradigm with the term-rewriting and Knuth-Bendix paradigm belong to these classes.

The strategies in the second family are called *subgoal-reduction strategies*, because if one considers the single object they work on as the goal, each step consists in reducing the goal to subgoals. Since they do not generate a set of objects, subgoal-reduction strategies do not use an ordering to sort it, neither do they use an object to delete another one. Because they need backtracking, a typical choice of search plan is *depth-first search with iterative deepening*. This family comprises *linear and linear-input clausal strategies*, and *tableaux-based strategies*. Model elimination, linear resolution, and problem reduction format methods, which also may have semantic variants, belong to these classes.

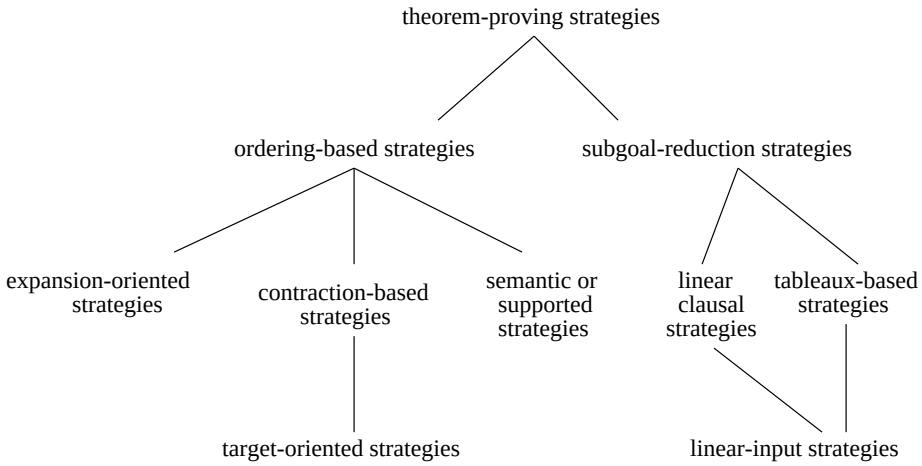

Figure 1: Classes of strategies

Central to this taxonomy is a formal notion of *search plan*. The availability of a sound and complete inference system guarantees the existence of a proof, but it remains the problem of how to generate one, and this is precisely the task of the search plan. Given the initial state of a refutational proof attempt containing H and $\neg\varphi$, the application of an inference rule to this state produces a new state. Thus, the problem can be seen as a *search problem*, with the inference rules as *transformation rules*, or *production rules*, *states* containing partial proofs, *successful states* containing complete proofs, and a *search plan* – or *computation rule* in terminology influenced by logic programming – to control the search. For instance, for a clausal ordering-based strategy, the states are sets or multisets of clauses, and a successful state contains the empty clause, while for a tableau-based strategy, the states are tableaux, and a successful state is a closed tableau.

If I denotes the given inference system, and $States$ denotes the set of all possible states, a search plan Σ is made of at least three components:

- A *rule-selecting function* $\zeta\colon States^* \to I$, which selects the next rule to be applied based on the history of the search so far;
- A *premise-selecting function* $\xi\colon States^* \to \mathcal{P}(\mathcal{L}_\Theta)$, which selects the elements of the current state the inference rule should be applied to;
- A *termination-detecting function* $\omega\colon States \to Bool$, which returns *true* if the given state is successful, false otherwise.

If the current state is not successful, ζ selects rule f and ξ selects premises $\psi_1 \dots \psi_n$, the next step will consist of applying f to $\psi_1 \dots \psi_n$. The sequence of states thus generated forms the *derivation* by I controlled by Σ from the given input. A derivation is *successful* if it terminates in a successful state.

It is important to appreciate that given an initial state with H and $\neg\varphi$, there are many derivations that an inference system I can generate from the initial state. In this sense, an inference system is *non-deterministic*. If I is coupled with a search plan Σ, there is one and only one derivation generated by I and Σ from the initial state. The combination of inference system and search plan forms a deterministic procedure called a *theorem-proving strategy*. While the inference system is required to be sound and refutationally complete, a search plan is expected to be *fair*: if there are proofs, or, equivalently, if there are successful states in the search space, one will be generated eventually.

By suitably specializing its components, this notion of search plan is shown to apply to all classes of strategies under consideration, including both ordering-based and subgoal-reduction strategies. Furthermore, it is applied to cover the concrete search plans of the ordering-based strategies implemented by the Argonne provers Otter and EQP. For all classes, the form of derivation is specified, and it is shown how inference system and search plan cooperate to generate it.

For ordering-based strategies, the modelling of the search space is developed beyond its description as a search space of states, summarized above. At a lower level of abstraction, the search space is modelled as a search graph of clauses, made dynamic by contraction. To see the relationship between these two levels of observation one can think of a magnifying lense: if one looks inside any state of the search space of states with a magnifying lense, one sees the underlying search graph of clauses. For the purposes of this taxonomy, this more detailed model helps to understand what it means that these strategies develop implicitly multiple partial proofs simultaneously, and, if the strategy succeeds, the proof that has been completed is extracted from the generated search space.

For subgoal-reduction strategies, the study of refinements include approaches to combine forward and backward reasoning, ways to import the notion of contraction from ordering-based strategies, and pruning techniques, as summarized in Table 1. The distinction between clausal and tableaux model elimination in this table is mostly one of terminology, since almost everything that can be done in one can be done in the other.

	Combination of forward and backward reasoning	Contraction	Pruning
Model elimination	lemmatization C-reduction success caching	lemma subsumption cache subsumption	identical ancestor pruning failure caching
Prolog Datalog	tabling/memoing magic sets		cut
Tableaux	regressive merging folding up UR-resolution hyperlinking	tableau subsumption subsumption tautology deletion purity deletion	irregularity anti-lemmas

Table 1: Refinements of subgoal-reduction strategies

At each stage of the derivation, subgoal-reduction strategies keep in memory (*active search space*) the current proof attempt (e.g., the current tableau), whereas ordering-based strategies keep in memory all generated clauses not deleted by contraction (see Table 2). On the other hand, because a subgoal-reduction strategy searches by backtracking, its *generated search space* is equal to the union of all the partial proofs it has attempted. In terms of proof, tableau-based strategies generate explicitly one proof attempt at a time, backtrack to modify it, and succeed when it is completed. Ordering-based strategies build their proof attempts implicitly, and when an empty clause is generated, extract the completed proof from the generated search space. If a relatively small active search space may be an advantage of subgoal-reduction strategies, ordering-based strategies may take advantage of contraction, which deletes existing redundant clauses, and also prevents their descendants in the search space from being generated.

	Ordering-based	Subgoal-reduction
Data	set of objects	one goal-object at a time
Proof attempts built	many implicitly	one at a time
Backtracking	no	yes
Contraction	yes	no
Generated search space	all generated clauses	all tried tableaux
Active search space	all kept clauses	current tableau
Generated proof	ancestor-graph of empty clause	closed tableau

Table 2: Two main classes of strategies.

All classifications contain some elements of arbitrariness. One may think that all ordering-based strategies are forward-reasoning strategies (i.e., strategies that reason from the assumptions), whereas all subgoal-reduction strategies are backward-reasoning strategies (i.e., strategies that reason from the goal), but this is not necessarily the case; for example, an ordering-based strategy with goal clauses in the set of support reasons backward, and a subgoal-reduction strategy with an assumption as top clause reasons forward. Similarly, it is not necessarily the case that subgoal-reduction strategies work with tableaux and ordering-based strategies work with clauses, although this is true in many cases. No classification can be complete, but connections are established with those strategies which may fit less obviously in this scheme, such as those based on *hyperlinking*, and with strategies that may use similar principles for other purposes, such as *model building*.

The full paper [1] provides the reader with a bibliography of over one hundred and forty entries: since it would have been impossible to include it in this summary and any selection would have been arbitrary, the interested reader is referred to the full paper which is available by contacting the author.

References

[1] Maria Paola Bonacina. A taxonomy of theorem-proving strategies. In Manuela Veloso, Mike Wooldridge (general), and Michael Fisher (logic reasoning knowledge representation area), editors, *Artificial Intelligence Today*, volume 1500. Springer Verlag, to appear.