Upside-Down Meta-Interpretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction[1]

Mark E. Stickel

Artificial Intelligence Center
SRI International
Menlo Park, California

Institute for New Generation
Computer Technology
Tokyo, Japan

September 14, 1992

Abstract

Typical bottom-up, forward-chaining reasoning systems such as hyperresolution lack goal-directedness while typical top-down, backward-chaining reasoning systems like Prolog or model elimination repeatedly solve the same goals. Reasoning systems that are goal-directed and avoid repeatedly solving the same goals can be constructed by formulating the top-down methods metatheoretically for execution by a bottom-up reasoning system (hence, "upside-down meta-interpretation" is being used). This also facilitates the use of flexible search strategies, such as merit-ordered search, that are common to bottom-up interpreters. The model elimination theorem proving procedure, its extension by an assumption rule for abduction and its restriction to Horn clauses, are adapted here for such upside-down meta-interpretation. This work can be regarded as an extension of the magic set method for query evaluation in deductive databases to both non-Horn clauses and abductive reasoning.

1 Introduction

Bottom-up, forward-chaining reasoning systems derive new facts from already established ones. The implication $A_1, \ldots, A_m \supset C$ is interpreted procedurally by such systems to derive the fact C from the facts $A_1, \ldots, A_m$. Hyperresolution [38, 46] is a typical bottom-up reasoning system. Top-down, backward-chaining reasoning systems, on the other hand, derive new subgoals from existing goals. The implication $A_1, \ldots, A_m \supset C$ is interpreted procedurally by such systems to derive each of the subgoals $A_1, \ldots, A_m$ from the goal C. Ordered input resolution (for Horn clauses, used by Prolog) and the model elimination procedure [22, 23] (for arbitrary clauses, used by PTTP [41]) are typical top-down reasoning systems. We assume the reader is already familiar with these inference procedures.

Both bottom-up and top-down methods have well known weaknesses. Bottom-up reasoning is often not goal-directed. For example, if the initial goal is translated for refutation into a negative clause, hyperresolution can use the goal only in the final step of a proof. Nevertheless, simply using a bottom-up reasoning method is often the right approach. For

[1]This research was supported by the National Science Foundation under Grant CCR-8922330 and by the Defense Advanced Research Projects Agency under Office of Naval Research Contract N00014-90-C-0220. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the National Science Foundation, the Defense Advanced Research Projects Agency, or the United States government.

example, group theory or condensed detachment problems benefit little from a top-down approach, since irrelevant axioms are absent and top-down reasoning quickly produces very general goals that fail to constrain the search. On the other hand, in deductive database, logic programming, and artificial intelligence applications, the lack of goal-directedness of pure bottom-up reasoning is a crucial defect. In principle, it would require enumeration of all consequences of the axioms until a fact matching the query is derived, a foolish approach in the presence of many irrelevant axioms.

The major problem with top-down reasoning is that it often results in goals being derived and proved more than once, which may result in large, redundant search spaces. For example, when Prolog tries to prove $P \wedge Q$, backtracking search will cause it to try to prove Q once for every proof of P it finds. This repeated work can be extraordinarily costly. "Intelligent backtracking" can reduce but not eliminate the problem. Redundancy can also occur in bottom-up methods in the form of facts being derived more than once. However, there the redundancy is controlled by subsumption, which deletes duplicate or less general facts. Although methods such as subsumption are costly and can drastically reduce the rate of inference, reduced search-space size often compensates for the lower inference rate.

A second problem with top-down reasoning systems is that they typically have much less flexibility in specifying order of search than bottom-up reasoning systems. Prolog and PTTP, for example, use depth-first search with backtracking for an efficient implementation with minimal storage requirements by representing only the goals on a single branch of the search space at a time, but this makes it impossible to direct search by jumping to a more favorable branch. Hyperresolution, on the other hand, can maintain a list of facts in order of preference for inference. Lack of control over search is not a necessary limitation of top-down reasoning systems, but rather an observation about typical ones. It is thus possible to adapt either bottom-up or top-down reasoning methods to produce a goal-directed reasoning system with a nonredundant search space and flexible search strategy. We choose to adapt bottom-up reasoning methods because their implementations appear to be closer to this ideal already. The prototypical bottom-up reasoning system hyperresolution already possesses effective methods for controlling redundancy (subsumption) and ordering the search space (merit-ordered search). As we shall see, it is feasible to make this bottom-up reasoning method more goal-directed.

The approach we adopt is basically an extension of the magic set method [3, 44] for query evaluation in deductive databases. We will translate Horn clauses similarly to the magic set method, and then extend the translation to abductive reasoning and non-Horn clauses.

The extension to non-Horn clauses is based on the model elimination theorem-proving procedure. Model elimination is a complete theorem-proving procedure for the full first-order predicate calculus that possesses the desirable properties of linear proofs, literal ordering, set of support, and no need for factoring. PTTP's implementation of the model elimination procedure has as well a high inference rate with minimal storage requirements. The largest problem with model elimination and PTTP is the failure to control search-space redundancy. Here, we demonstrate how, while unfortunately sacrificing PTTP's implementation approach with its high inference rate and minimal storage requirements, we can make search much less redundant by means of "upside-down meta-interpretation", i.e., by executing the top-down model elimination procedure by a bottom-up interpreter.

Our approach is to start with top-down, backward-chaining input resolution and transform the clauses for execution by a bottom-up interpreter such as hyperresolution. Instead of a goal-subgoal tree being created, literals of the form $goal(G)$ are derived. Use of the implication $A_1 \wedge \cdots \wedge A_m \supset C$ to derive the fact C from facts $A_1, \ldots, A_m$ is made contingent on the existence of $goal(C)$ by use of the translated clause $goal(C) \wedge fact(A_1) \wedge \cdots \wedge fact(A_m) \rightarrow fact(C)$. The translation is extended to impose a requirement of left-to-right solution, as in Prolog and the model elimination procedure. In many cases, this can substantially reduce the search space as solutions for earlier goals instantiate later goals.

Abductive reasoning (abduction) is becoming an important application of extended Prolog and model elimination systems. Abduction extends deduction to the case of partial proofs with assumptions that, if they could be proved, would allow a proof to be completed. We extend our translation method to abduction. The added possibility in abduction of assuming as well as proving formulas makes the search space for abduction problems larger than for deduction problems with the same axioms. This, plus the fact that many applications of abduction demand rich knowledge bases with many irrelevant clauses, means that there may be an even bigger payoff for this method in the case of abduction than deduction.

Clauses used in Prolog and model elimination inference are translated for execution by a bottom-up interpreter using metapredicates *goal*, *fact*, and *cont* (for "continuations", which concisely encode what goals we are trying to solve, which of their subgoals have been solved, and which subgoals remain). New facts, goals, and continuations are derived by bottom-up inference in a faithful encoding of the Prolog or model elimination search tree, possibly in a different order depending on the chosen search strategy, and with redundant subtrees eliminable by the reuse of facts derived earlier and by subsumption. The time complexity in the worst case, when there is no eliminable redundancy, should be the same order as that of Prolog or model elimination when the latter's search strategy is imitated (a three-fold increase in length of the proof may occur, as a single literal in the search tree may be represented by *goal*, *fact*, and *cont* literals in the encoding). In the case of Horn clauses, the procedure closely resembles hyperresolution in behavior, except hyperresolution operations are allowed only if they derive a fact that matches a top-down derived goal.

Because this method is a new approach to implementing standard theorem proving procedures (Prolog, model elimination, and their extensions for abduction) instead of a new theorem-proving procedure, we will omit soundness and completeness results. The benefit of the new approach in eliminating redundancy should be obvious. Gains from eliminating redundancy can be arbitrarily large.

In Section 2, we recount past approaches to the problem of redundancy in the model elimination procedure and cite their disadvantages, which are absent in the new approach. In Section 3, we describe upside-down meta-interpretation of Prolog-style deduction with Horn clauses. This, except for the remarks on generalizing subsumption and generalizing derived facts, is essentially the magic set method. In Section 4, the method is extended to abduction by allowing conditional facts accompanied by assumptions sufficient to establish them. In Section 5, the method for abduction with Horn clauses is transformed by different handling of assumptions into a method for upside-down meta-interpretation of model-elimination-style deduction with non-Horn clauses. Deduction with non-Horn clauses is then extended to abduction with non-Horn clauses in Section 6. Related work is described in Section 7.

2 Other Methods for Eliminating Redundancy

There are several other approaches to eliminating redundancy in model elimination and similar procedures. Factoring is the earliest method for eliminating duplicate goals and is required for completeness in many resolution procedures, though not for Prolog or model elimination. It is clearly beneficial and can be made mandatory in the propositional case. However, in the first-order case when goals must be unified during factoring, factoring must be optional and proofs with and without the goals factored must both be sought. This results in an increase in the breadth of the search space; the depth of the search space is reduced in compensation only if a shorter proof can be found with factoring than without, which is too rarely the case. Unifying goals often results in clauses becoming overinstantiated and not usable in a proof.

The graph construction procedure [39] adds the C-reduction operation to the model elimination procedure. C-reduction resembles factoring except that it unifies an unproved goal with a proved goal instead of another unproved goal. This is an improvement because unprovable goals are never factored. For example, if a pair of factorable goals do not happen to have a common provable instance, factoring them will ultimately result in failure. If, as in the graph construction procedure, it is necessary for one to be proved before being factored with the other, the goals will no longer be factorable after one of them is instantiated by its proof.

Both factoring and C-reduction affect only the descendants of the factored clause. No information about provable goals is made available to other parts of the search space. Lemmas [22, 23] are extra clauses derivable by the model elimination procedure that contain proved goals. Lemmas are essentially previously solved goals. They are not required for completeness, but their use can shorten proofs by matching a goal with a lemma instead of reproving it. Unlike factoring and C-reduction, lemmas are available throughout the search space after they are derived, not just in descendant clauses. However, like factoring and C-reduction, lemmas increase the breadth of the search space, by allowing proofs from lemmas as well as axioms. Lemmas in the model elimination procedure save information about successful but not unsuccessful proof attempts. There nevertheless is the obvious notion of "failure lemmas"—remembered goals that could not be proved. Lemmas have been used in database query evaluation [5, 12, 43] methods and in other theorem proving procedures [13, 27, 32], often under the name "caching", although we use the that term to refer to a slightly different concept.

Caching is the most complete approach for eliminating redundancy in top-down reasoning systems. By saving goals as well as solutions, caching can record information about both success and failure. In a depth-bounded reasoner like PTTP, the cache would contain goals and associated depth bounds asserting that the cache contains all solutions to the goal discoverable with that depth bound. When attempting to prove a goal with a depth bound, if the goal or a more general one with the same or greater depth bound is stored in the cache, solutions are retrieved from the cache instead of searching for solutions by backward-chaining. The difference between lemmas and caching is that lemmas are treated as extra axioms and broaden the search while caching replaces the search for solutions of a goal by cache lookup. Only caching of the methods we have described uniformly replaces search instead of adding alternatives to it in the hope of finding a shorter proof. Caching

can easily reduce the size of the search space even if the proof found is not shorter. Many successful experiments with lemmas and caching are reported on in [1].

Caching will surely be more complicated and less effective for the full model elimination procedure than for the Prolog subset on which it has been successfully tested. In the full procedure, solutions to a goal no longer depend on the goal formula alone, but also on its ancestor goals. Even if goals recur frequently, they may rarely recur with a set of ancestor goals that can be found in the cache. A refinement of the model elimination procedure that uses negative but not positive ancestor goals may make looking up solutions in the cache succeed more frequently [33], but probably still not often enough. Although caching can eliminate redundant search, it can contribute little to solving the other problem of top-down reasoning systems, the inflexibility of their search strategy.

3 Deduction with Horn Clauses

A Horn clause problem is composed of a set of facts F, a set of rules $A_1 \wedge \cdots \wedge A_m \supset C$ with $m \geq 1$, and a goal G, where F, A_i, C, and G are all atomic formulas. Requiring the goal to be atomic is not a significant restriction. A conjunctive goal $G_1 \wedge \cdots \wedge G_n$ can be converted into the rule $G_1 \wedge \cdots \wedge G_n \supset G$ for atomic goal G.

A rule $A_1 \wedge \cdots \wedge A_m \supset C$ can be interpreted in top-down or bottom-up fashion. The top-down interpretation is:

> From the goal C derive the goals $A_1, \ldots, A_m$.

A problem is solved when one can recursively derive from the goal G a set of subgoals all of which match initial facts F. Input resolution, as in Prolog, is a standard top-down reasoning method. The bottom-up interpretation is:

> Derive the fact C from the facts $A_1, \ldots, A_m$.

A problem is solved when a fact matching the goal G is derived from the initial facts F. Hyperresolution, for example, is a standard bottom-up reasoning method.

In the following, we assume a bottom-up reasoning system such as hyperresolution with subsumption. The rule $A_1 \wedge \cdots \wedge A_m \rightarrow C$[2] is interpreted as: if $A_1, \ldots, A_m$ are present, then C can be derived. The separate roles of an atomic formula L as a fact or goal will be distinguished by putting L as an argument of the *fact* or *goal* metapredicate.

Top-down and bottom-up interpretations of $A_1 \wedge \cdots \wedge A_m \supset C$ are expressed metatheoretically by

$$goal(C) \rightarrow goal(A_1)$$
$$\vdots$$
$$goal(C) \rightarrow goal(A_m)$$

and

$$fact(A_1) \wedge \cdots \wedge fact(A_m) \rightarrow fact(C)$$

[2] Note the use of $\rightarrow$ for executable rules versus $\supset$ for assertions.

respectively.

We now connect the *goal* and *fact* rules. The *fact* rule can be modified and used in conjunction with the *goal* rules to provide bottom-up execution with top-down filtering:

$$goal(C) \rightarrow goal(A_1)$$
$$\vdots$$
$$goal(C) \rightarrow goal(A_m)$$
$$goal(C) \wedge fact(A_1) \wedge \cdots \wedge fact(A_m) \rightarrow fact(C)$$

Goals are generated in simulated top-down fashion, but bottom-up reasoning is constrained: $fact(C)$ can only be derived if $goal(C)$ is present. Note that the clauses resulting from this translation and all the extensions we present are Horn. Thus, a bottom-up interpreter such as hyperresolution will derive only unit clauses using them.

Subsumption is used to eliminate duplicate or less general facts or goals. Facts, once derived, can be used again in the solution of other goals. The *goal* derivation rules employ upside-down meta-interpretation, since the meaning of the rules is the top-down generation of subgoals, but the rules themselves are executed bottom-up. Each initial fact F is translated to $fact(F)$ and the initial goal is translated to $goal(G)$. Proofs are completed by deriving instances of $fact(G)$.

This translation of the problem is often sufficient. However, it is sometimes better in the case of clauses with more than one antecedent literal to create subgoals sequentially, e.g., to generate (an appropriate instance of) $goal(A_i)$ only after goals $A_1, \ldots, A_{i-1}$ have been solved. This is especially important in logic-programming problems, in which some subgoals compute values used as inputs to later subgoals. For example, the rule $fib(x, y) \wedge fib(s(x), z) \wedge plus(y, z, w) \supset fib(s(s(x)), w)$ for computing Fibonacci numbers could be used to create the subgoals $fib(3, y)$, $fib(4, z)$, and $plus(y, z, w)$ from the goal $fib(5, w)$. It would be better to delay creating the subgoal $plus(y, z, w)$ until after $fib(3, y)$ and $fib(4, z)$ are solved, thus instantiating y and z.

To impose a left-to-right execution order for subgoals so that $goal(A_{i+1})$ is not introduced until a solution to $goal(A_i)$ has been found, "continuation" predicates are used to encode the state of matching antecedent literals of a rule. Let k be a unique number for the rule $A_1 \wedge \cdots \wedge A_m \supset C$ with $m \geq 2$ and let V be a term that contains variables of the rule except those in the head.[3] The rule is translated as follows:[4]

$$goal(C) \rightarrow goal(A_1)$$
$$goal(C) \wedge fact(A_1) \rightarrow cont_{k,2}(C, V) \wedge goal(A_2)$$
$$cont_{k,2}(C, V) \wedge fact(A_2) \rightarrow cont_{k,3}(C, V) \wedge goal(A_3)$$
$$\vdots$$
$$cont_{k,m-1}(C, V) \wedge fact(A_{m-1}) \rightarrow cont_{k,m}(C, V) \wedge goal(A_m)$$
$$cont_{k,m}(C, V) \wedge fact(A_m) \rightarrow fact(C)$$

[3] Not all variables need be included in every continuation. For $cont_{k,i}$, it is sufficient to include $(Vars(\{A_1, \ldots, A_{i-1}\}) \cap Vars(\{A_i, \ldots, A_m\})) - Vars(C)$, where $Vars(X)$ is the set of variables appearing in literal or set of literals X.

[4] Some of these rules have multiliteral consequents $cont_{k,i}(C, V) \wedge goal(A_i)$, which means that both $cont_{k,i}(C, V)$ and $goal(A_i)$ are to be derived. If standard, clausal hyperresolution is used as the bottom-up interpreter, they can be split into separate rules $\cdots \rightarrow cont_{k,i}(C, V)$ and $cont_{k,i}(C, V) \rightarrow goal(A_i)$.

The literals $cont_{k,i}(C, V)$ identify which subgoal is being solved with what substitution.

A classic example of poor, highly redundant top-down execution behavior is the computation of Fibonacci numbers. The computation can be defined by:

a. $plus(0, x, x)$
b. $plus(x, y, z) \supset plus(s(x), y, s(z))$
c. $fib(0, 0)$
d. $fib(s(0), s(0))$
e. $fib(x, y) \wedge fib(s(x), z) \wedge plus(y, z, w) \supset fib(s(s(x)), w)$

which can be translated to:[5]

1. $fact(plus(0, x, x))$
2. $goal(plus(s(x), y, s(z))) \rightarrow goal(plus(x, y, z))$
3. $goal(plus(s(x), y, s(z))) \wedge fact(plus(x, y, z)) \rightarrow fact(plus(s(x), y, s(z)))$
4. $fact(fib(0, 0))$
5. $fact(fib(s(0), s(0)))$
6. $goal(fib(s(s(x)), w)) \rightarrow goal(fib(x, y))$
7. $goal(fib(s(s(x)), w)) \wedge fact(fib(x, y)) \rightarrow$
   $cont_{e,2}(fib(s(s(x)), w), y) \wedge goal(fib(s(x), z))$
8. $cont_{e,2}(fib(s(s(x)), w), y) \wedge fact(fib(s(x), z)) \rightarrow$
   $cont_{e,3}(fib(s(s(x)), w), y, z) \wedge goal(plus(y, z, w))$
9. $cont_{e,3}(fib(s(s(x)), w), y, z) \wedge fact(plus(y, z, w)) \rightarrow fact(fib(s(s(x)), w))$

whose execution is substantially less redundant because Fibonacci numbers do not need to be recomputed.

3.1 Generalizing Subsumption

Subsumption is the principal mechanism for eliminating redundancy in bottom-up reasoning. If $fact(L)$ and $fact(L\sigma)$ are both derived, then $fact(L\sigma)$ can be deleted. Likewise, if $goal(L)$ and $goal(L\sigma)$ are both derived, then $goal(L\sigma)$ can be deleted. Similarly for $cont_{k,i}(C, V)$ and $cont_{k,i}(C\sigma, V\sigma)$. These deletions can be accomplished by ordinary subsumption.

It is beneficial to generalize this. The following instances of generalized subsumption are possible:

- $fact(L)$ subsumes $goal(L')$, where $L' = L\sigma$ for some substitution σ. Goals can be deleted if they are the same as or more specific than a fact.[6]
- $fact(L)$ subsumes $cont_{k,i}(C, V)$, where $C = L\sigma$ for some substitution σ. Continuations can be deleted if they lead only to the derivation of facts the same as or more specific than an existing one.

[5] Instead of a variable-containing term V, we write all the variables as separate arguments of $cont_{k,i}$.

[6] Although derived facts are always instances of the goals that lead to them, an initial fact might be more general than a goal, and it is also possible to modify the method to derive more general facts (see Section 3.2).

A stronger deletion strategy would also delete subgoals of deleted goals. Goal-subgoal relationships would have to be recorded so that a subgoal is deleted only if all the goals of which it is a subgoal have been deleted.

3.2 Generalizing Derived Facts

Although unnecessary recomputation of Fibonacci numbers is successfully eliminated in the example, bottom-up interpretation unfiltered by goals could yield a still shorter proof that uses fewer, more general derived facts. The problem is that derived facts are sometimes overly specific. This is a result of their having been derived with top-down filtering.

It is possible to derive $plus(1, y, s(y))$ from clauses a and b, and it is likewise possible to derive $fact(plus(1, y, s(y)))$ from 1–3 when given the general goal $goal(plus(1, y, z))$. However, if more specific goals such as $goal(plus(1, 1, z))$, $goal(plus(1, 2, z))$, and $goal(plus(1, 3, z))$ are given, as they are when these rules are used to compute Fibonacci numbers, only the more specific facts such as $fact(plus(1, 1, 2))$, $fact(plus(1, 2, 3))$, and $fact(plus(1, 3, 4))$ will be derived. Computing larger Fibonacci numbers results in many more repeated instances of computing $x + y_1$, $x + y_2$, ... The length of each of these derivations is linear in the size of x.

When $goal(L)$ leads to the derivation of $fact(L\sigma_1)$, the problem of possible overspecificity of $fact(L\sigma_1)$ can be overcome by reexecuting the same inference steps starting with $goal(x)$ (i.e., with a free variable as goal formula) instead of $goal(L)$ and ending with $fact(x\sigma_2)$, which is stored instead of $fact(L\sigma_1)$. The result $fact(x\sigma_2)$ is an equally valid conclusion that is either a generalization of or equivalent to $fact(L\sigma_1)$. There is no danger in deriving these more general facts. They are more easily used, but top-down filtering still prevents their use except in the presence of a relevant goal.

Note that the problem of deriving overly specific goals is not universal. From ground facts and range-restricted rules (those in which every variable in a positive literal also appears in a negative literal), which are customary in databases, bottom-up reasoning can derive only ground facts, and top-down filtering cannot result in anything more specific. The magic set method for range-restricted databases thus has no need for fact generalization.

4 Abduction with Horn Clauses

We shall now extend the method to abduction with Horn clauses. First, we give a general description of abduction, not restricted to Horn clauses. We will then extend the method in Section 3 to a method for abduction with Horn clauses. Section 6 describes abduction with non-Horn clauses.

Abduction is the form of reasoning that allows us to hypothesize that P is true if we know that $P \supset Q$ is true and we are trying to explain why Q is true [31]. It can naturally be viewed as an extension of deduction. Instead of being required to prove a formula, abduction allows us to identify sets of hypotheses that, if they could be proved, would allow a proof of the formula to be completed. This style of reasoning has been applied to diagnosis [8, 29, 30, 36], design synthesis [15], theory formation [35], default and circumscriptive reasoning [35, 37], and natural language interpretation [7, 18, 28, 42].

A widespread approach for implementing abduction is top-down, backward-chaining reasoning with some literals being allowed to be assumed instead of proved [8, 18, 19, 35, 36, 37, 40, 42], i.e., an inference rule that assumes a literal is added to Prolog-like inference (in the case of Horn clauses) or the model elimination procedure. Standard top-down reasoning can be viewed as operating on a list of goals, removing goals when they match facts, adding subgoals when a goal matches the head of a rule, and succeeding only when the list becomes empty. Abductive reasoning allows this process to "skip" certain goals [19]. An abductive proof or explanation is found when only skipped goals remain. These are the assumptions that would allow completion of the proof.

The presence of an additional inference rule that allows literals to be either assumed or proved makes the search space for abduction even larger than that for deduction. This provides a strong motivation for upside-down meta-interpretation of the top-down inference rules for abduction in order to eliminate search-space redundancy. Recent work on using an ATMS [9, 10] to cache results of abductive reasoning [26] has the same objective as ours of eliminating redundant work on duplicate goals and has already demonstrated significant improvement. This is done for the case of Horn clauses with some limitation on unification as a result of using an ATMS.

For some theory T and goal G, abduction consists of finding sets of assumptions H and substitutions θ such that $G\theta$ is a consequence of $T \cup H$, i.e., $H \supset G\theta$ is a consequence of T. We require that H consist of *assumable* atomic formulas with designated predicate symbols.

We focus on only one element of abduction here, namely, finding H and $G\theta$. It is a nearly universal requirement that H be consistent with T, but this must be determined by some other means (e.g., by attempting to refute $T \cup H$ and failing) and is undecidable in general. Many abductive proofs can usually be found, and selection of a preferred abductive proof is a vital part of abduction. One criterion is that an abductive proof that requires a subset of the assumptions required by another one is preferred. Generalized subsumption of derived facts allows us to discard such less general proofs. Assigning costs to assumable formulas is a popular method to help choose among alternative proofs and is the focus of much recent work on abduction [6, 18, 42]. We believe the top-down meta-interpretation approach for abduction can be adapted to such cost-based abduction, but this is outside the scope of the present work.

To support abductive reasoning, the metatheoretic predicate *fact* is extended to two arguments: an atomic formula and a set of assumptions sufficient to prove it. Bottom-up interpretation of the rule $A_1 \wedge \cdots \wedge A_m \supset C$ can be expressed by

$$fact(A_1, H_1) \wedge \cdots \wedge fact(A_m, H_m) \rightarrow fact(C, H_1 \cup \cdots \cup H_m)$$

If each A_i is true, assuming H_i, then C is true, assuming the union of the assumptions. Each initial fact F is translated to $fact(F, \emptyset)$. If atomic formula L is assumable, $fact(L, \{L\})$ is included in the initial facts; its meaning is that L is allowed to be proved by assuming L. Note that $fact(X, \{X\})$ is a tautology, i.e., $L \supset L$ or $L \vee \neg L$.

Our rules for Horn clause deduction by bottom-up execution with top-down filtering and left-to-right solution of goals can also be easily adapted to Horn clause abduction. The general case of the translation of $A_1 \wedge \cdots \wedge A_m \supset C$ is

$$goal(C) \leftarrow goal(A_1)$$
$$goal(C) \wedge fact(A_1, H_1) \leftarrow cont_{k,2}(C, H_1, V) \wedge goal(A_2)$$
$$cont_{k,2}(C, H, V) \wedge fact(A_2, H_2) \leftarrow cont_{k,3}(C, H \cup H_2, V) \wedge goal(A_3)$$
$$\vdots$$
$$cont_{k,m-1}(C, H, V) \wedge fact(A_{m-1}, H_{m-1}) \leftarrow cont_{k,m}(C, H \cup H_{m-1}, V) \wedge goal(A_m)$$
$$cont_{k,m}(C, H, V) \wedge fact(A_m, H_m) \leftarrow fact(C, H \cup H_m)$$

where $H, H_1, \ldots, H_m$ are variables whose values during execution will be sets of assumptions used in deriving a continuation or fact.

The procedure is complete: for any H and $G\theta$ such that H is composed of assumable literals, $H \supset G\theta$ is a consequence of theory T, and H is consistent with T,[7] this procedure can derive some $fact(G', H')$ such that $G'\sigma = G\theta$ and $H'\sigma \subseteq H$ for some substitution σ.

Subsumption can be further generalized to take account of assumptions. The following instances of generalized subsumption are possible:

- $fact(L, H)$ subsumes $fact(L', H')$, where $L' = L\sigma$ and $H' \supseteq H\sigma$ for some substitution σ.
- $fact(L, H)$ subsumes $cont_{k,i}(C, H', V)$, where $C = L\sigma$ and $H' \supseteq H\sigma$ for some substitution σ.
- $fact(L, \emptyset)$ subsumes $goal(L')$, where $L' = L\sigma$ for some substitution σ.
- $cont_{k,i}(C, H, V)$ subsumes $cont_{k,i}(C', H', V')$, where $C' = C\sigma$, $H' \supseteq H\sigma$, and $V' = V\sigma$ for some substitution σ.

As an example, consider the following theory used to explain a bicycle's wobbly wheel [21]. Here, *broken-spokes*, *punctured-tube*, and *leaky-value* are assumable predicates that can be used to create an explanation.

a. $flat\text{-}tire \supset wobbly\text{-}wheel$
b. $broken\text{-}spokes \supset wobbly\text{-}wheel$
c. $punctured\text{-}tube \supset flat\text{-}tire$
d. $leaky\text{-}valve \supset flat\text{-}tire$

The translation is

1. $fact(broken\text{-}spokes, \{broken\text{-}spokes\})$
2. $fact(punctured\text{-}tube, \{punctured\text{-}tube\})$
3. $fact(leaky\text{-}valve, \{leaky\text{-}valve\})$
4. $goal(wobbly\text{-}wheel) \leftarrow goal(flat\text{-}tire)$
5. $goal(wobbly\text{-}wheel) \wedge fact(flat\text{-}tire, H) \leftarrow fact(wobbly\text{-}wheel, H)$
6. $goal(wobbly\text{-}wheel) \leftarrow goal(broken\text{-}spokes)$
7. $goal(wobbly\text{-}wheel) \wedge fact(broken\text{-}spokes, H) \leftarrow fact(wobbly\text{-}wheel, H)$
8. $goal(flat\text{-}tire) \leftarrow goal(punctured\text{-}tube)$
9. $goal(flat\text{-}tire) \wedge fact(punctured\text{-}tube, H) \leftarrow fact(flat\text{-}tire, H)$

[7] Although the procedure may generate abductive proofs with hypotheses inconsistent with T, it is not guaranteed to and we would not want it to generate all sets of inconsistent hypotheses.

10. $goal(flat\text{-}tire) \leftarrow goal(leaky\text{-}valve)$
11. $goal(flat\text{-}tire) \wedge fact(leaky\text{-}valve, H) \leftarrow fact(flat\text{-}tire, H)$

Execution of these rules with the goal of explaining a wobbly wheel follows. Explanations are found on lines 16, 19, and 21, e.g., if there was a punctured tube, then there would be a wobbly wheel.

12. $goal(wobbly\text{-}wheel)$	initial goal
13. $goal(flat\text{-}tire)$	subgoal of 12 by 4
14. $goal(punctured\text{-}tube)$	subgoal of 13 by 8
15. $fact(flat\text{-}tire, \{punctured\text{-}tube\})$	solution of 13 by 2,9
16. $fact(wobbly\text{-}wheel, \{punctured\text{-}tube\})$	solution of 12 by 15,5
17. $goal(leaky\text{-}valve)$	subgoal of 13 by 10
18. $fact(flat\text{-}tire, \{leaky\text{-}valve\})$	solution of 13 by 3,11
19. $fact(wobbly\text{-}wheel, \{leaky\text{-}valve\})$	solution of 12 by 18,5
20. $goal(broken\text{-}spokes)$	subgoal of 12 by 6
21. $fact(wobbly\text{-}wheel, \{broken\text{-}spokes\})$	solution of 12 by 1,7

5 Deduction with Non-Horn Clauses

Using the method for abduction with Horn clauses as a starting point, we now extend our upside-down meta-interpretation method to deduction with possibly non-Horn clauses. Abduction will be added again in Section 6. Facts, goals, and rules can be written with literals instead of just atomic formulas. We require that contrapositives of the rules be present. That is, if $A_1 \wedge \cdots \wedge A_m \supset C$ is a rule, then m other rules of the form $A \supset \neg A_i$ must also be provided, where A is the conjunction of $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m, \neg C$, and, for any literal L, $\neg L$ denotes its complement.

The model elimination (ME) theorem-proving procedure has a single inference rule in addition to Prolog's:

> If the current goal is unifiable with the complement of one of its ancestor goals, then apply the unifying substitution and treat the current goal as if it were solved.

This added inference operation is the ME *reduction* operation. The normal Prolog inference operation is the ME *extension* operation. The two together comprise a complete inference procedure for the full first-order predicate calculus, not just the Horn-clause subset. Unless the unifying substitution (unifier) is empty (i.e., the goal and its ancestor goal are exactly complementary), the reduction operation is used as an alternative to, not a substitute for, solving the goal by extension or by reduction with a different ancestor goal.

Similarly to abduction with Horn clauses, we begin by formulating model elimination procedure in terms of deriving facts that follow from a set of assumptions.

The metatheoretic predicate *fact* has two arguments: a literal and a set of assumptions sufficient to prove it. Bottom-up interpretation of the rule $A_1 \wedge \cdots \wedge A_m \supset C$ can be expressed by

$$fact(A_1, H_1) \wedge \cdots \wedge fact(A_m, H_m) \rightarrow fact(C, (H_1 \cup \cdots \cup H_m) - \neg C)$$

If each A_i is true, assuming H_i, then C is true, assuming the union of the assumptions, excluding $\neg C$. This description is accurate for the ground case. In the nonground case, it is necessary to consider unifying $\neg C$ with other assumptions to derive alternative results. In that way, different instances of C can be shown to follow from different sets of assumptions. For example, suppose $\neg C$ is not a member of $H_1 \cup \cdots \cup H_m$. We conclude that C is true, assuming $H_1 \cup \cdots \cup H_m$. If $\neg C$ is unifiable (by unifier σ) with a member of $H_1 \cup \cdots \cup H_m$, we can also conclude that $C\sigma$ is true, assuming the smaller set $(H_1\sigma \cup \cdots \cup H_m\sigma) - \neg C\sigma$.

Single-literal facts F are translated to $fact(F, \emptyset)$. The single literal $fact(x, \{x\})$ is also included. Its interpretation is that any literal x is a consequence of its own assumption. Note again that $fact(x, \{x\})$ is a tautology, i.e., $x \supset x$ or $x \vee \neg x$.

This differs from upside-down meta-interpretation of abduction with Horn clauses because all literals are treated as assumable (because any literal might be solvable by reduction with a complementary ancestor goal) and because $\neg C$ can be omitted from the set of assumptions used.

Top-down filtering by goals along with left-to-right execution order for subgoals can be accomplished almost exactly as in the case of abduction for Horn clauses:

$$\begin{aligned}
&goal(C) \rightarrow goal(A_1) \\
&goal(C) \wedge fact(A_1, H_1) \rightarrow cont_{k,2}(C, H_1, V) \wedge goal(A_2) \\
&cont_{k,2}(C, H, V) \wedge fact(A_2, H_2) \rightarrow cont_{k,3}(C, H \cup H_2, V) \wedge goal(A_3) \\
&\qquad \vdots \\
&cont_{k,m-1}(C, H, V) \wedge fact(A_{m-1}, H_{m-1}) \rightarrow cont_{k,m}(C, H \cup H_{m-1}, V) \wedge goal(A_m) \\
&cont_{k,m}(C, H, V) \wedge fact(A_m, H_m) \rightarrow fact(C, (H \cup H_m) - \neg C)
\end{aligned}$$

Note the use of $\neg C$ in the final clause.

Performance of this code is likely to be very poor. Assumptions can be made easily but can be removed only in the presence of a complementary ancestor goal; a proof is complete only when the assumption-free $fact(G\sigma, \emptyset)$ is derived for $goal(G)$. It is apparent that more control over the generation of facts is required. Top-down filtering is done above using only the form of the goal; we propose top-down filtering also take account of the goal's ancestors, so that a fact will not be derived unless a goal exists whose ancestor list includes all the fact's the assumptions.

For top-down meta-interpretation of the model elimination procedure for deduction, we include another argument, P, in goals and continuations that specifies the set of assumptions (obtained from negations of ancestor goals) that are permitted to be made in the solution of a goal. Siegel likewise replaced model elimination's representation of goal-subgoal relationships in chains by directly associating a goal with its set of ancestors [40]. The translated rules will not be able to derive facts that require assumptions outside this set.

$$\begin{aligned}
&goal(C, P) \rightarrow goal(A_1, P \cup \{\neg C\}) \\
&goal(C, P) \wedge fact(A_1, H_1) \wedge H_1 \subseteq P \cup \{\neg C\} \rightarrow \\
&\qquad\qquad cont_{k,2}(C, H_1, P, V) \wedge goal(A_2, P \cup \{\neg C\})
\end{aligned}$$

$$cont_{k,2}(C,H,P,V) \wedge fact(A_2,H_2) \wedge H_2 \subseteq P \cup \{\neg C\} \rightarrow cont_{k,3}(C,H \cup H_2,P,V) \wedge goal(A_3,P \cup \{\neg C\})$$

$$\vdots$$

$$cont_{k,m-1}(C,H,P,V) \wedge fact(A_{m-1},H_{m-1}) \wedge H_{m-1} \subseteq P \cup \{\neg C\} \rightarrow cont_{k,m}(C,H \cup H_{m-1},P,V) \wedge goal(A_m,P \cup \{\neg C\})$$

$$cont_{k,m}(C,H,P,V) \wedge fact(A_m,H_m) \wedge H_m \subseteq P \cup \{\neg C\} \rightarrow fact(C,(H \cup H_m) - \neg C)$$

A single-literal goal is translated to $goal(G,\emptyset)$, i.e., an assumption-free proof of G is sought. Unification of members of H_i and $P \cup \{\neg C\}$ may be necessary to make $\subseteq$ hold and unification of members of $H \cup H_m$ with $\neg C$ may be necessary to derive facts with fewer assumptions. If this rule is invoked by $goal(G,P)$, it will derive literals of the form $fact(G',H)$, where $G' = G\sigma$ and $H \subseteq P\sigma$ for some substitution σ. Derived facts include only assumptions that are used (those in H_i), not all those that are permitted to be used (those in P). Thus, equally general facts can be derived even if P has extra members.

The following instances of generalized subsumption are possible:

- $fact(L,H)$ subsumes $fact(L',H')$, where $L' = L\sigma$ and $H' \supseteq H\sigma$ for some substitution σ. Facts that are less general or require more assumptions can be deleted.
- $fact(L,H)$ subsumes $goal(C,P)$ or $cont_{k,i}(C,H',P,V)$, where $C = L\sigma$ and $P \supseteq H\sigma$ for some substitution σ. Such facts solve the goal without instantiating it.
- $cont_{k,i}(C,H,P,V)$ subsumes $cont_{k,i}(C',H',P',V')$, where $C' = C\sigma$, $H' \supseteq H\sigma$, $P' = P\sigma$, and $V' = V\sigma$ for some substitution σ. Continuations that are less general or have made more assumptions can be deleted.

In addition, standard model elimination pruning rules imply that

- $goal(C,P)$ or $cont_{k,i}(C,H,P,V)$ can be deleted if $C \in P$, $\neg C \in P$, or P contains complementary literals.

As an example, consider the proof that $a \wedge b$ follows from $a \vee b$, $\neg a \vee b$, and $a \vee \neg b$. The problem is formulated with contrapositives as

a. $\neg a \supset b$
b. $\neg b \supset a$
c. $a \supset b$
d. $\neg b \supset \neg a$
e. $\neg a \supset \neg b$
f. $b \supset a$
g. $a \wedge b \supset q$

and the translation is

1: $fact(x,\{x\})$
2: $goal(b,P) \rightarrow goal(\neg a,P \cup \{\neg b\})$
3: $goal(b,P) \wedge fact(\neg a,H) \wedge H \subseteq P \cup \{\neg b\} \rightarrow fact(b,H - \{\neg b\})$
4: $goal(a,P) \rightarrow goal(\neg b,P \cup \{\neg a\})$

5: $goal(a, P) \wedge fact(\neg b, H) \wedge H \subseteq P \cup \{\neg a\} \rightarrow fact(a, H - \{\neg a\})$
6: $goal(b, P) \rightarrow goal(a, P \cup \{\neg b\})$
7: $goal(b, P) \wedge fact(a, H) \wedge H \subseteq P \cup \{\neg b\} \rightarrow fact(b, H - \{\neg b\})$
8: $goal(\neg a, P) \rightarrow goal(\neg b, P \cup \{a\})$
9: $goal(\neg a, P) \wedge fact(\neg b, H) \wedge H \subseteq P \cup \{a\} \rightarrow fact(\neg a, H - \{a\})$
10: $goal(\neg b, P) \rightarrow goal(\neg a, P \cup \{b\})$
11: $goal(\neg b, P) \wedge fact(\neg a, H) \wedge H \subseteq P \cup \{b\} \rightarrow fact(\neg b, H - \{b\})$
12: $goal(a, P) \rightarrow goal(b, P \cup \{\neg a\})$
13: $goal(a, P) \wedge fact(b, H) \wedge H \subseteq P \cup \{\neg a\} \rightarrow fact(a, H - \{\neg a\})$
14: $goal(q, P) \rightarrow goal(a, P \cup \{\neg q\})$
15: $goal(q, P) \wedge fact(a, H) \wedge H \subseteq P \cup \{\neg q\} \rightarrow cont_{g,2}(q, H, P)$
16: $cont_{g,2}(q, H, P) \rightarrow goal(b, P \cup \{\neg q\})$
17: $cont_{g,2}(q, H, P) \wedge fact(b, H_2) \wedge H_2 \subseteq P \cup \{\neg q\} \rightarrow fact(q, (H \cup H_2) - \{\neg q\})$

Execution of these rules leads to the following proof:

18: $goal(q, \emptyset)$	initial goal
19: $goal(a, \{\neg q\})$	subgoal of 18 by 14
20: $goal(b, \{\neg a, \neg q\})$	subgoal of 19 by 12
21: $fact(b, \{\neg a\})$	solution of 20 by 3,1
22: $fact(a, \emptyset)$	solution of 19 by 13,21
23: $fact(b, \emptyset)$	solution of 20 by 7,22
24: $cont_{g,2}(q, \emptyset, \emptyset)$	continuation of 18 by 15,22
25: $fact(q, \emptyset)$	solution of 18 by 17,23,24

Note that derived facts exactly correspond to lemmas in the model elimination procedure: they are conditionally solved goals, where the conditions are negations of ancestor goals used in their solution. Contrapositives of derived facts are also valid consequences, so facts like $fact(\neg b, \{a, c\})$ can be automatically derived from $fact(a, \{b, c\})$, or the procedure can be reformulated to use a neutral clause form $fact(\{a, \neg b, \neg c\})$ instead (this is done in Demolombe's similar method [11]).

6 Abduction with Non-Horn Clauses

The case of abduction with non-Horn clauses is nearly identical to that of deduction. The only change required is that assumptions are no longer restricted to those listed in goals as being permitted because their negations appeared in ancestor goals. This restriction is imposed by the test $H_i \subseteq P$. The test is modified in the case of abduction to apply only to literals that are not abductively assumable: $nonass(H_i) \subseteq P \cup \{\neg C\}$, where $nonass(H_i)$ is the largest subset of H_i that cannot be abductively assumed (those with nonassumable predicate names). In other words, any abductively assumable literal in H_i need not appear in $P \cup \{\neg C\}$, but others must.

We summarize the treatment of assumptions in these procedures. In the Horn case of abduction, $fact(L, \{L\})$ exists only for abductively assumable literals, so only they can be assumed. In the non-Horn case of deduction, $fact(x, \{x\})$ exists and any literal can be assumed, though top-down filtering permits only assumptions that match negated ancestor

goals to be used. In the non-Horn case of abduction, we again allow any literal to be assumed, but omit the requirement to match assumptions with negated ancestor goals in the case of abductively assumable literals.

Derivation of $fact(G\sigma, H)$ is an abductive proof of G, provided H consists entirely of abductively assumable literals. The procedure is complete: for any H and $G\theta$ such that H is composed of abductively assumable literals, $H \supset G\theta$ is a consequence of theory T, and H is consistent with T, this procedure can derive some $fact(G', H')$ such that $G'\sigma = G\theta$ and $H'\sigma \subseteq H$ for some substitution σ.

7 Related Work

Demolombe [11] also developed upside-down meta-interpretation of the model elimination theorem proving procedure. His method resembles the procedure in Section 5, but differs in that

- It uses literals like $goal(a \vee \neg b \vee \neg c)$ and $fact(a \vee \neg b \vee \neg c)$ instead of $goal(a, \{b, c\})$ and $fact(a, \{b, c\})$; contrapositives of facts are thus always available.
- It uses rules like $goal(C) \wedge fact(A_1) \wedge \cdots \wedge fact(A_{i-1}) \rightarrow goal(A_i)$ to generate subgoals, instead of using more concise continuation predicates.
- It doesn't keep track of which permitted assumptions are actually used, so $goal(a \vee b \vee c)$ can lead only to instances of $fact(a \vee b \vee c)$ being derived, instead of the more general $fact(a)$, $fact(a \vee b)$, etc., that can be derived if not all permitted assumptions are used.

Plaisted and Greenbaum [34] developed an upside-down meta-interpretation method for non-Horn clauses that is not based on the model elimination procedure. It doesn't require contrapositives and represents clauses by $A_1 \wedge \cdots \wedge A_m \supset C_1 \vee \cdots \vee C_n$, where $A_1, \ldots, A_m$ and $C_1, \ldots, C_n$ are all atoms. However, only negative clauses are used as initial goals. A key difference between their method and ours is that our translation yields a Horn set of clauses. The advantage of this is that if hyperresolution is used to execute the clauses, only single-literal results will be derived (though it must be said these single-literal facts or goals may contain multiple literals from the problem and thus still have a clause interpretation). Plaisted and Greenbaum's method requires derivation of non-unit positive clauses, such as $fact(a) \vee fact(b)$. They also developed an extension for equality, based on Brand's modification method [4]—something we haven't done yet.

Upside-down meta-interpretation has been applied to Horn clause theorem proving in Neiman's subgoal extraction method [25]. It closely resembles rewriting methods for query evaluation, as do the Horn clause case in Section 3 and the Demolombe and the Plaisted and Greenbaum methods restricted to Horn clauses. Neiman describes special data structures for more efficient execution.

There is a vast literature on such upside-down meta-interpretation methods for query evaluation in Horn clause databases. These methods (such as the magic set method) generally resemble each other abstractly, differing in details of the compilation and the extent to which the input rules are partially evaluated. Bry demonstrated upside-down meta-interpretation (i.e., rewriting-based query evaluation methods) and top-down evaluation

with lemmas (i.e., resolution-based query evaluation methods) are essentially equivalent instances of his backward fixpoint procedure [5]. There has also been a lot of work that extends magic sets to non-Horn deductive databases with negation as failure or closed world rather than classical semantics for negation (e.g., [2, 14, 20]).

Our approach is to use bottom-up execution with top-down filtering. This is conceptually similar to the use of relevancy testing [45, 17] in the bottom-up SATCHMO [24] and MGTP [16] theorem provers that employ hyperresolution and case-splitting on nonunit derived clauses. The use of range-restricted clauses guarantees that positive clauses are ground and makes case-splitting practical, since no variables are shared between cases. The relevancy test requires that each literal of a derived clause be relevant to the goal and can dramatically reduce the search space. The SATCHMO/MGTP approach appears to work very well on naturally range-restricted problems—better than model elimination. Problems that are not range-restricted can be easily converted into those that are, but this entails adding clauses that can generate all the terms of the Herbrand universe, and the SATCHMO/MGTP approach is usually ineffective for such problems.

8 Conclusion

The model elimination procedure is an effective theorem-proving procedure whose principal defect is the redundancy of its search space. Despite this defect, it has been used effectively for theorem proving and recently for abductive and related inference. Model elimination is a highly restrictive inference procedure that includes compatibility with set of support. This goal-directedness is crucial in the presence of many irrelevant axioms, such as in deductive database, logic programming, and artificial intelligence applications.

Upside-down meta-interpretation, the execution of the top-down model elimination procedure by a bottom-up interpreter like hyperresolution with subsumption, can basically reproduce the model elimination search space while eliminating much of its redundancy. Four variants of the method have been shown. The basic method for deduction with Horn clauses resembles the magic set method for query evaluation in databases. Extensions deal with non-Horn clauses and with abduction as well as deduction.

Upside-down meta-interpretation can be regarded as adding top-down filtering to a bottom-up interpreter thus making it more goal-directed. Its principal contribution is in applications with many irrelevant axioms, not for mathematical problems. Although non goal-directed methods such as hyperresolution might seem naive even for mathematical problems, they can actually be quite effective: when all the axioms are accessible from the initial goal and general subgoals are quickly generated, the top-down filtering provided by upside-down meta-interpretation is able to offer little or no goal-directedness.

The high inference rate and low memory consumption of top-down reasoning system such as Prolog and PTTP are lost in this move to upside-down meta-interpretation. This seems inevitable, since controlling redundancy requires storing more information about goals, solutions, etc., and the volume of information stored demands efficient, but still slow indexing. Efforts to make the inference rate of bottom-up interpreters more closely approach that of top-down interpreters will make the upside-down meta-interpretation approach more attractive. Writing a bottom-up interpreter specialized to the rules used in upside-down meta-interpretation can also improve performance. Neiman did this in the case of deduction

with Horn clauses when implementing his subgoal extraction method.

Acknowledgements

I would like to express my appreciation to ICOT for providing a friendly and supportive environment for doing this research and discussing and investigating many aspects of theorem proving. I would like to thank Masayuki Fujita and Francois Bry for our discussions of this work and Katsumi Inoue and Donald Loveland for their comments on an earlier draft of this paper.

References

[1] Astrachan, O.L. and M.E. Stickel. Caching and lemmaizing in model elimination theorem provers. *Proceedings of the 11th International Conference on Automated Deduction*, Sarasota Springs, New York, June 1992.

[2] Balbin, I., G.S. Port, and K. Ramamohanarao. Magic set computation for stratified databases. Technical Report 87/11, Department of Computer Science, University of Melbourne, Melbourne, Australia, 1987.

[3] Bancilhon, F., D. Maier, Y. Sagiv, and J. Ullman. Magic sets and other strange ways to implement logic programs. *Proceedings of the Fifth ACM Symposium on Principles of Database Systems,* 1986, 53–56.

[4] Brand, D. Proving theorems with the modification method. *SIAM Journal of Computing 4* (1975), 412–430.

[5] Bry, F. Query evaluation in recursive databases: bottom-up and top-down reconciled. *Data & Knowledge Engineering 5* (1990), 289–312.

[6] Charniak, E. and S. Husain. A new admissable heuristic for minimal-cost proofs. *Proceedings of the AAAI-91 National Conference on Artificial Intelligence*, Anaheim, California, July 1991.

[7] Charniak, E. and R. Goldman. A logic for semantic interpretation. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, June 1988, 87–94.

[8] Cox, P.T. and T. Pietrzykowski. Causes for events: their computation and applications. *Proceedings of the 8th Conference on Automated Deduction,* Oxford, England, July 1986, 608–621.

[9] deKleer, J. An assumption-based TMS. *Artificial Intelligence 28* (1986), 127–162.

[10] deKleer, J. Extending the ATMS. *Artificial Intelligence 28* (1986), 163–196.

[11] Demolombe, R. An efficient strategy for non-Horn deductive databases. *Theoretical Computer Science 78* (1991), 245–259.

[12] Dietrich, S.W. Extension tables: memo relations in logic programming. *Proceedings of the 1987 Symposium on Logic Programming,* San Francisco, California, August 1987, 264–272.

[13] Elkan, C. Conspiracy numbers and caching for searching and/or trees and theorem proving. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, Michigan, August 1989, 341–346.

[14] Fernández, J.A. and J. Minker. Bottom-up evaluation of hierarchical disjunctive deductive databases. *Proceedings of the Eighth International Conference on Logic Programming,* Paris, France, 1991, 660–675.

[15] Finger, J.J. *Exploiting Constraints in Design Synthesis.* Ph.D. dissertation, Department of Computer Science, Stanford University, Stanford, California, February 1987.

[16] Fujita, H. and R. Hasegawa. A model generation theorem prover in KL1 using a ramified-stack algorithm. Technical Report TR-606, Institute for New Generation Computer Technology, Tokyo, Japan, 1991.

[17] Fujita, M. Personal communication, 1991.

[18] Hobbs, J.R., M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. Technical Note 499, Artificial Intelligence Center, SRI International, December 1990. Revised version to appear in *Artificial Intelligence.*

[19] Inoue, K. Consequence-finding based on ordered linear resolution. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence,* Sydney, Australia, August 1991.

[20] Kemp, D.B., P.J. Stuckey, and D. Srivastava. Magic sets and bottom-up evaluation of well-founded models. *Proceedings of International Symposium on Logic Programming,* 1991.

[21] Kowalski, R.A. Problems and promises of computational logic. *Proceedings of the First Symposium on Computational Logic,* 1990, 1–36.

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

[23] Loveland, D.W. *Automated Theorem Proving: A Logical Basis.* North-Holland, Amsterdam, the Netherlands, 1978.

[24] Manthey, R. and F. Bry. SATCHMO: a theorem prover in Prolog. *Proceedings of the 9th International Conference on Automated Deduction,* Argonne, Illinois, May 1988.

[25] Neiman, V.S. Refutation search for Horn sets by a subgoal-extraction method. *Journal of Logic Programming 9* (1990), 267–284.

[26] Ng, H.T. and R.J. Mooney. An efficient first-order abduction system based on the ATMS. *Proceedings of the AAAI-91 National Conference on Artificial Intelligence,* Anaheim, California, July 1991.

[27] Nie, X. and D.A. Plaisted. A complete semantic back chaining proof system. *Proceedings of the 10th International Conference on Automated Deduction*, Kaiserslautern, Germany, July 1990, 16–27.

[28] Norvig, P. Inference in text understanding. *Proceedings of the AAAI-87 National Conference on Artificial Intelligence*, Seattle, Washington, July 1987.

[29] Peng, Y. and J.A. Reggia. A probabilistic causal model for diagnostic problem solving – part I: integrating symbolic causal inference with numeric probabilistic inference. *IEEE Transactions on Systems, Man, and Cybernetics SMC-17*, 2 (March/April 1987), 146–162.

[30] Peng, Y. and J.A. Reggia. A probabilistic causal model for diagnostic problem solving—part II: diagnostic strategy. *IEEE Transactions on Systems, Man, and Cybernetics SMC-17*, 3 (May/June 1987), 395–406.

[31] Pierce, C.S. Abduction and induction. In Buchler, J. (Ed.). *Philosophical Writings of Pierce*. Dover Books, New York, 1955, pp. 150–156.

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

[33] Plaisted, D.A. A sequent-style model elimination strategy and a positive refinement. *Journal of Automated Reasoning 6*, 4 (December 1990), 389–402.

[34] Plaisted, D.A. and S. Greenbaum. Problem representations for chaining and equality in resolution theorem proving. *Proceedings of the First Conference on Artificial Intelligence Applications*, Denver, Colorado, December 1984, 417–423.

[35] Poole, D. Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence 5* (1989), 97–110.

[36] Pople, H.E., Jr. On the mechanization of abductive logic. *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford, California, August 1973, 147–152.

[37] Przymusinski, T.C. An algorithm to compute circumscription. *Artificial Intelligence 38*, 1 (February 1989), 49–73.

[38] Robinson, J.A. Automatic deduction with hyper-resolution. *International Journal of Computer Mathematics*, 1 (1965), 227–234.

[39] Shostak, R.E. Refutation graphs. *Artificial Intelligence 7*, 1 (Spring 1976), 51–64.

[40] Siegel, P. *Représentation et Utilisation de la Connaissance en Calcul Propositionnel.* Thèse d'Etat, Université de Aix-Marseille II, 1987.

[41] Stickel, M.E. A Prolog technology theorem prover: implementation by an extended Prolog compiler. *Journal of Automated Reasoning 4*, 4 (December 1988), 353–380.

[42] Stickel, M.E. A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. *Annals of Mathematics and Artificial Intelligence* 4 (1991), 89–106.

[43] Tamaki, H. and T. Sato. OLD resolution with tabulation. *Proceedings of the Third International Conference on Logic Programming,* London, England, 1986, 84–98.

[44] Ullman, J.D. *Principles of Database and Knowledge-Base Systems.* Computer Science Press, Rockville, Maryland, 1989.

[45] Wilson, D.S. and D.W. Loveland. Incorporating relevancy testing in SATCHMO. Technical Report CS-1989-24, Department of Computer Science Duke University, Durham, North Carolina, November 1989.

[46] Wos, L., R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning.* Prentice-Hall, Englewood Cliffs, New Jersey, 1984.