TR-664

Upside-Down Meta-InterPretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction

by
Mark Stickel

July, 1991

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

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

May 1991

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 method also allows incorporation of more flexible search ordering strategies, such as merit-ordered search, that are commonly available in bottom-up but not top-down interpreters; this advantage is not so readily or fully achievable in the alternative approach of adding caching to a top-down interpreter. Neiman has developed a similar method for deduction with Horn clauses and Bry has explored the idea in the context of query evaluation in databases. This work extends theirs to both non-Horn clauses and abductive reasoning and can be regarded as an extension of the magic set method to these cases. Model elimination extended by the ability to make assumptions for abduction and its restriction to Horn clauses are the top-down methods adapted here.

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 [19, 20] (for arbitrary clauses, used by PTTP [41]) are typical top-down reasoning systems. We assume the reader is already familiar with these inference procedures.

[1]This research was supported by the National Science Foundation under Grant CCR-8922330. 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 or the United States government.

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. A subset of the positive clauses is sometimes used as the set of support in conjunction with hyperresolution [46, 47]. For example, hypotheses of the theorem, but not general axioms of a theory, may be put in the set of support to make the derivation more goal-directed by requiring clauses to be derived only directly or indirectly from the hypotheses. Although this often works, it is not guaranteed to, since the combination of hyperresolution and the set of support strategy is incomplete in general.

Top-down reasoning 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 and 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. There are convincing arguments that failure to control redundancy by some method such as subsumption will lead to failure to prove difficult theorems [28], despite the high inference rate possible for top-down reasoning systems such as Prolog and PTTP. Although methods such as subsumption are costly and can drastically reduce the rate of inference, reduced search-space size 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. For example, Prolog uses depth-first search with backtracking while hyperresolution can maintain a list of facts in order of preference for inference.

In many instances, simply using a bottom-up reasoning method is the right solution to the problem. For example, problems in group theory or implicational calculus benefit little from a top-down approach, since irrelevant axioms are absent and top-down reasoning quickly produces very general goals. 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.

It is 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 they 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. Caching can be added to top-down reasoning methods to eliminate search-space redundancy as subsumption does in bottom-up methods. However, the extra code required to index formulas to make caching efficient is substantial [44], while such code is already present in good implementations of bottom-up reasoning. Greater flexibility in ordering the search space can be obtained in many top-down reasoning systems such as Prolog only by discarding the current implementation approach.

The approach we adopt is similar to that of Neiman's subgoal extraction method [23] for nonredundant top-down deduction with Horn sets of clauses and Bry's backward fixpoint procedure [4] for query evaluation in recursive databases. Bry presents the magic set method [3] as a specialization of the backward fixpoint procedure to particular database rules. We will translate Horn clauses similarly to the magic set method, and then extend the translation to abductive reasoning and non-Horn clauses.

We use this approach with 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 and low storage requirements, we can make search much less redundant by means of "upside-down meta-interpretation", i.e., by interpreting 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.

The intent is to translate clauses used in Prolog and model elimination inference for execution by a bottom-up interpreter using metapredicates *goal*, *fact*, and *cont* (for "continuations", which concisely encode what goals are we 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 top-down reasoning systems 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 similar to the methods of Neiman and Bry and 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.

2 Previous Approaches

There are several previous approaches to eliminating redundancy in the 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. While factoring can shorten proofs and is clearly beneficial and can be made mandatory in the propositional case, 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. 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. This is a sort of lazy factoring, because it delays factoring until after one of the goals is proved. This is much better, since pairs of unproved goals are no longer instantiated by factoring. For example, if a pair of factorable literals 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 proved.

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 [19, 20] are extra clauses derivable by the model elimination procedure that contain proved goals. Lemmas are not required for completeness, but their use can shorten proofs in much the same way as factoring or C-reduction can. 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 save information about successful but not unsuccessful proof attempts.

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. Only caching of the methods we have described uniformly replaces this 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. Caching has been used in database query evaluation [4, 11] methods and in other theorem proving procedures [12, 25, 32]. We have experimented with it in the Prolog subset of PTTP and achieved substantial reductions in the number of inferences, but at the expense of increased time, since cache storage and retrieval is slow compared to PTTP's compiled inference. Unless cache operations are sped up, a net saving appears possible only for quite large problems. Recent results with another implementation of model elimination have been more favorable [1, 2].

Caching will surely be more complicated and less effective for the full model elimination procedure than for the Prolog subset. 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 reduces the dependency on ancestor goals and may make looking up solutions in the cache succeed more frequently [33]. Although caching can eliminate redundant search, it can contribute little to the other problem of top-down reasoning systems, the inflexibility of their search strategy.

Typical bottom-up reasoning systems eliminate redundancy effectively and allow general search strategies. Although they have the disadvantage of a much lower inference rate than top-down reasoning systems like Prolog or PTTP, adding caching or more flexible search strategies to the latter would eliminate much of their speed advantage by forcing them to use similar data structures to those of bottom-up systems. The advantages of top-down reasoning systems seem to be their high inference rate, low memory consumption, and goal-directedness. It appears to be necessary to compromise on the first two to eliminate redundancy. Thus, if bottom-up systems, which already have general search strategies and the ability to eliminate redundancy, can be given greater goal-directedness, there is little incentive for sticking with the top-down approach. We believe our upside-down meta-interpretation procedure, which executes the top-down model elimination procedure by a bottom-up interpreter, achieves the desired objective.

Our approach is to use bottom-up execution with top-down filtering. This is conceptually similar to the use of relevancy testing [45, 16] in the bottom-up SATCHMO [21] and MGTP [15] theorem provers that use hyperresolution and case-splitting on nonunit derived clauses. Case splitting is practical because derived clause are required to be ground. This is guaranteed in the case of range-restricted clauses (those in which every variable in a positive literal also appears in a negative literal). 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.

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 bottom-up or top-down fashion. 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. 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.

In the following, we assume a bottom-up reasoning system such as hyperresolution with subsumption. The rule $A_1 \wedge \cdots \wedge A_m \rightarrow C$ 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.

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

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

and

$$\begin{aligned} goal(C) &\rightarrow goal(A_1) \\ &\vdots \\ goal(C) &\rightarrow goal(A_m) \end{aligned}$$

respectively.

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

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

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 $goal(G)$. If a method like hyperresolution is used as the bottom-up interpreter for the translated rules, execution can be terminated when a solution to the initial goal is found by also including the clause $\neg fact(G)$. This can be resolved with $fact(G)$ when it is finally derived to produce the empty clause.

This translation of the problem is sometimes sufficient. However, it is often better 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 early 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.

A left-to-right execution order for subgoals can be imposed so that $goal(A_{i+1})$ is not introduced until a solution to $goal(A_i)$ has been found:

$$goal(C) \rightarrow goal(A_1)$$
$$goal(A_1) \wedge fact(A_1) \rightarrow goal(A_2)$$
$$\vdots$$
$$goal(A_{m-1}) \wedge fact(A_{m-1}) \rightarrow goal(A_m)$$
$$goal(A_m) \wedge fact(A_m) \rightarrow fact(C)$$

If $m = 1$, only the first and last clause are present.

Actually, the code above is not quite correct. It allows $goal(A_{i+1})$ to be derived from $goal(A_i)$ and $fact(A_i)$ even if $goal(A_i)$ was introduced by some other clause, such as $\cdots \wedge A_i \wedge \cdots \rightarrow D$. There may also be some confusion if A_i is unifiable with another A_j in the same clause, so it is necessary to distinguish between different goals from the same clause as well as goals from different clauses. Moreover, the code above does not insure that a consistent set of variable values is used in goal C and facts $A_1, \ldots, A_m$.

Let k be a unique number for the rule $A_1 \wedge \cdots \wedge A_m \supset C$ and let V be a term that contains all the variables of the rule except those in the head. The rule number, antecedent-literal number, head and variables are put into a continuation predicate as follows:[2]

[2]Some of these rules have multiliteral consequents $goal(A_i) \wedge cont_{k,i}(C, V)$, which means that both

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

The literals $cont_{k,i}(C,V)$ identify which subgoal is being solved with what substitution. Actually, k, i, V alone are sufficient if V includes the variables of the head (the magic set method does not include C in such literals). Our representation includes C as well to facilitate generalized subsumption.

We note in passing the similarity of this code to a translation of hyperresolution nuclei that seeks to reduce the number of unification operations. Partial matches of the antecedent are encoded in continuations as above. The rule $A_1 \wedge \cdots \wedge A_m \rightarrow C$ with $m \geq 3$ is translated into:

$$A_1 \wedge A_2 \rightarrow cont_{k,3}(V)$$
$$cont_{k,3}(V) \wedge A_3 \rightarrow cont_{k,4}(V)$$
$$\vdots$$
$$cont_{k,m-1}(V) \wedge A_{m-1} \rightarrow cont_{k,m}(V)$$
$$cont_{k,m}(V) \wedge A_m \rightarrow C$$

Derivation of $cont_{k,i}(V)$ indicates that literals matching $A_1, \ldots, A_{i-1}$ exist with the unifier recorded in V. Hyperresolution can be viewed as performing a sequence of ordered binary resolution steps, but without storing the intermediate clauses $A_2 \wedge \cdots \wedge A_m \supset C$, $A_3 \wedge \cdots \wedge A_m \supset C$, etc. They are in effect recomputed by repeated unification operations when making inferences from new literals A_i. The literal $cont_{k,i}(V)$ is a compact encoding of an instance of $A_i \wedge \cdots \wedge A_m \supset C$. These continuations generally result in more clauses being derived and stored as well as fewer unifications being performed, so there is a tradeoff. Continuations correspond to intermediate nodes in the RETE network for pattern matching [14]. When operations more complex than ordinary unification are used, as when computing labels for an ATMS [9, 10], a great deal of effort can be saved by creating such intermediate nodes or continuations [27]. Note that the transformed hyperresolution nuclei here and the clauses created for upside-down meta-interpretation have at most two negative literals.

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

a1. $plus(0,x,x)$
a2. $plus(x,y,z) \supset plus(s(x),y,s(z))$
a3. $fib(0,0)$
a4. $fib(s(0),s(0))$
a5. $fib(x,y) \wedge fib(s(x),z) \wedge plus(y,z,w) \supset fib(s(s(x)),w)$

which can be translated to:[3]

$goal(A_i)$ and $cont_{k,i}(C,V)$ are to be derived. If standard, clausal hyperresolution is used as the bottom-up interpreter, they must be split into separate rules $\cdots \rightarrow goal(A_i)$ and $\cdots \rightarrow cont_{k,i}(C,V)$.

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

b1. $fact(plus(0,x,x))$

b2. $goal(plus(s(x),y,s(z))) \rightarrow goal(plus(x,y,z))$
b3. $goal(plus(s(x),y,s(z))) \rightarrow cont_{a2,1}(plus(s(x),y,s(z)))$
b4. $cont_{a2,1}(plus(s(x),y,s(z))) \wedge fact(plus(x,y,z)) \rightarrow fact(plus(s(x),y,s(z)))$

b5. $fact(fib(0,0))$

b6. $fact(fib(s(0),s(0)))$

b7. $goal(fib(s(s(x)),w)) \rightarrow goal(fib(x,y))$
b8. $goal(fib(s(s(x)),w)) \rightarrow cont_{a5,1}(fib(s(s(x)),w))$
b9. $cont_{a5,1}(fib(s(s(x)),w)) \wedge fact(fib(x,y)) \rightarrow goal(fib(s(x),z))$
b10. $cont_{a5,1}(fib(s(s(x)),w)) \wedge fact(fib(x,y)) \rightarrow cont_{a5,2}(fib(s(s(x)),w),y)$
b11. $cont_{a5,2}(fib(s(s(x)),w),y) \wedge fact(fib(s(x),z)) \rightarrow goal(plus(y,z,w))$
b12. $cont_{a5,2}(fib(s(s(x)),w),y) \wedge fact(fib(s(x),z)) \rightarrow cont_{a5,3}(fib(s(s(x)),w),y,z)$

b13. $cont_{a5,3}(fib(s(s(x)),w),y,z) \wedge fact(plus(y,z,w)) \rightarrow fact(fib(s(s(x)),w))$

whose execution is substantially less redundant. These clauses, with minor syntactic changes, are interpretable by the bottom-up hyperresolution inference procedure implemented in the OTTER theorem prover [22].

The following is an OTTER derivation of $fib(5)$ from clauses b1–b13 (for readability, $s(0)$ has been replaced by 1, $s(s(0))$ has been replaced by 2, etc.).

```
14 [] goal(fib(5,A)).
15 [] -fact(fib(5,A)).
16 [hyper,14,8] cont501(fib(5,A)).
17 [hyper,14,7] goal(fib(3,A)).
18 [hyper,17,8] cont501(fib(3,A)).
20 [hyper,18,10,6] cont502(fib(3,A),1).
21 [hyper,18,9,6] goal(fib(2,A)).
22 [hyper,21,8] cont501(fib(2,A)).
24 [hyper,22,10,5] cont502(fib(2,A),0).
25 [hyper,24,12,6] cont503(fib(2,A),0,1).
27 [hyper,25,13,1] fact(fib(2,1)).
28 [hyper,27,12,20] cont503(fib(3,A),1,1).
29 [hyper,27,11,20] goal(plus(1,1,A)).
30 [hyper,29,3] cont201(plus(1,1,s(A))).
31 [hyper,30,4,1] fact(plus(1,1,2)).
32 [hyper,31,13,28] fact(fib(3,2)).
33 [hyper,32,10,16] cont502(fib(5,A),2).
34 [hyper,32,9,16] goal(fib(4,A)).
```

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.

```
35 [hyper,34,8] cont501(fib(4,A)).
36 [hyper,35,10,27] cont502(fib(4,A),1).
37 [hyper,36,12,32] cont503(fib(4,A),1,2).
38 [hyper,36,11,32] goal(plus(1,2,A)).
39 [hyper,38,3] cont201(plus(1,2,s(A))).
41 [hyper,39,4,1] fact(plus(1,2,3)).
42 [hyper,41,13,37] fact(fib(4,3)).
43 [hyper,42,12,33] cont503(fib(5,A),2,3).
44 [hyper,42,11,33] goal(plus(2,3,A)).
45 [hyper,44,3] cont201(plus(2,3,s(A))).
46 [hyper,44,2] goal(plus(1,3,A)).
47 [hyper,46,3] cont201(plus(1,3,s(A))).
49 [hyper,47,4,1] fact(plus(1,3,4)).
50 [hyper,49,4,45] fact(plus(2,3,5)).
51 [hyper,50,13,43] fact(fib(5,5)).
52 [binary,51,15] .
```

To see that our objective of reducing redundancy has been achieved, note the occurrence of the goals $fib(3)$ and $fib(2)$ on lines 17 and 21, their solutions on lines 32 and 27, and the reuse (instead of rederivation, as top-down evaluation would require) of their solutions on lines 36–38.

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. 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.
- $fact(L)$ 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.

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 a1 and a2, and it is likewise possible to derive $fact(plus(1, y, s(y)))$ from b1 b4 when given the 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 on lines 29, 38, and 46 of the example), more specific facts such as $fact(plus(1, 1, 2))$, $fact(plus(1, 2, 3))$, and $fact(plus(1, 3, 4))$ will be derived (as on lines 31, 41, and 49). 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, which are customary in databases, bottom-up reasoning can derive only ground facts, and top-down filtering cannot result in anything more specific.

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 [7, 8, 29, 30, 36], design synthesis [13], theory formation [34, 35], default and circumscriptive reasoning [34, 35, 37], and natural language interpretation [6, 17, 26, 42, 43].

A widespread approach for implementing abduction is top-down, backward-chaining reasoning with some literals being allowed to be assumed instead of proved [7, 8, 17, 18, 34, 35, 36, 37, 40, 42, 43], 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 [18]. 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 [24] has the same objective

as ours of eliminating redundant work on duplicate goals and has already demonstrated significant improvement when compared to their implementation of a noncaching top-down algorithm [42]. 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 (predicate specific abduction [13, 34, 35, 42, 43]). It is, of course, permissable to designate all atomic formulas to be assumable. If nonatomic assumptions are desired, and the form of the assumable formula F is known in advance, a new atomic formula P can be created and then $P \supset F$ used and P assumed, instead of assuming the nonatomic formula F directly.

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 [5, 17, 42, 43]. We believe the top-down meta-interpretation approach for abduction can be adapted to such cost-based abduction without difficulty, 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.

Our rules for Horn clause deduction by bottom-up execution with top-down filtering and left-to-right solution of goals can be easily adapted to Horn clause abduction:

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

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.

For any H and $G\theta$ such that H is composed of assumable literals, $H \supset G\theta$ is a conse-

quence of T, and H is consistent with T,[4] 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.

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

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

[4] 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.

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.

This differs from upside-down meta-interpretation of abduction with Horn clauses because all literals are treated as assumable (because any literal can be solved by reduction with a complementary ancestor goal) and if C is proved, $\neg C$ 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 as in the case of abduction for Horn clauses:

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

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

A single-literal goal can be translated as before to $goal(G)$. A deductive model elimination proof is complete when $fact(G\sigma, \emptyset)$ is derived, i.e., when an instance of G is derived with the empty set of assumptions. If a method like hyperresolution is used as the bottom-up interpreter for the translated rules, execution can be terminated when a solution to the initial goal is found by also including the clause $\neg fact(G, \emptyset)$. This can be resolved with $fact(G, \emptyset)$ when it is finally derived to produce the empty clause. When seeking indefinite answers to a goal, $fact(\neg G, \emptyset)$ is included as well as $goal(G, \emptyset)$.

We expect performance of this code to be poor, since assumptions can be made easily but can only be removed in the presence of a complementary ancestor goal. For example, if C is a goal, $fact(C, \{A_1, \ldots, A_m\})$ can be derived. But this fact can be used in a proof only if additional facts are derived from it in which $A_1, \ldots, A_m$ are all absent from the set of assumptions, having been removed by the $(H \cup H_m) - \neg C$ computations, an unlikely possibility. 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.

$$goal(C, P_0) \rightarrow goal(A_1, P_0 \cup \{\neg C\}) \wedge cont_{k,1}(C, \emptyset, P_0 \cup \{\neg C\}, V)$$

$$cont_{k,1}(C, \emptyset, P, V) \wedge fact(A_1, H_1) \wedge H_1 \subseteq P \rightarrow goal(A_2, P) \wedge cont_{k,2}(C, H_1, P, V)$$

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

$$\vdots$$

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

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

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 may be necessary to make H_i a subset of P and unification of members of $(H \cup H_m)$ and $\neg C$ 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. This justifies generalized subsumption that eliminates goals with permitted assumption sets that are a subset of those of another goal.

We give a sketch of a sample proof of p from $s \supset p$, $t \supset p$, and $s \vee t$. For clarity, continuations are omitted; they do not a large role in this proof anyway, since the rules all have single-literal antecededents.

```
1. goal(p,{})              initial goal
2. goal(s,{-p})            subgoal of 1 by s -> p
3. goal(-t,{-p,-s})        subgoal of 2 by -t -> s
4. goal(-p,{-p,-s,t})      subgoal of 3 by -p -> -t
5. fact(-p,{-p})           instance of fact(x,{x})
6. fact(-t,{-p})           solution of 3 by -p -> -t from 5
7. fact(s,{-p})            solution of 2 by -t -> s from 6
8. fact(p,{})              solution of 1 by s -> p from 7
```

We explain a few features of the proof. On line 2, the goal s is derived, with $\neg p$, the negation of the parent goal, listed as a permitted assumption for use in the proof of s. On line 6, it is derived that $\neg t$ is a consequence of assuming $\neg p$ because $\neg p$ is a consequence of assuming $\neg p$ (line 5) and $\neg p$ implies $\neg t$. Although $\neg p$ and $\neg s$ were both permitted as assumptions (line 6), only $\neg p$ was used and appears in the result. On line 8, the unconditional fact p is derived to complete the proof, because s is a consequence of assuming $\neg p$ (line 7) and s

implies p (if p is a consequence of assuming $\neg p$, then p is true, by excluded middle). The $\neg p$ assumption of line 7 is removed in the result on line 8 by the $(H \cup H_m) - \neg C$ computation.

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 $cont_{k,i}(C, H', P, V)$, where $C = L\sigma$ and $H' \supseteq H\sigma$ for some substitution σ. Continuations that lead only to the derivation of facts that are less general or require more assumptions can be deleted.
- $fact(L, \emptyset)$ subsumes $goal(C, P)$, where $C = L\sigma$ for some substitution σ. Goals can be deleted if an unconditional more general fact exists.
- $goal(L, P)$ subsumes $goal(L', P')$, where $L' = L\sigma$ and $P' \subseteq P\sigma$ for some substitution σ. Goals that are less general or allow fewer assumptions in their proof can be deleted.

In the ordinary model elimination procedure, a goal of the form $goal(G, P)$, where P is the set of negated ancestor goals and happens to include $\neg G$, can be rejected because a goal is identical to an ancestor goal. It is incorrect to use this rule in top-down meta-interpretation if we use the generalized subsumption rule that allows deletion of goals with smaller sets of permitted assumptions. The goal $goal(p, \{q\})$ might yield the conclusion $fact(p, \emptyset)$ or $fact(p, \{q\})$. The goal $goal(p, \{\neg p, q\})$ can yield the same conclusions, justifying its use to subsume the goal with fewer permitted assumptions, but only if evaluation of $goal(p, \{\neg p, q\})$ is not blocked by the identical ancestor pruning rule.

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: $nonass(H_i) \subseteq P$, 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, 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. For any H and $G\theta$ such that H is composed of abductively assumable literals, $H \supset G\theta$ is a consequence of 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 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 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. Of previous methods for eliminating redundancy—factoring, C-reduction, lemmas, and caching—only caching is capable of achieving the same goal. Efficient caching, however, requires many of the same data structures and operations as bottom-up interpretation, e.g., clause indexing. Typical top-down reasoners also lack the flexible search strategies possible for bottom-up reasoners. Thus, upside-down meta-interpretation instead of top-down interpretation plus caching makes it easier to incorporate flexible search strategies.

Such upside-down meta-interpretation has been applied to Horn clauses in Neiman's subgoal extraction method and Bry's backward fixpoint procedure, which specializes to rewriting-based query evaluation methods such as the magic set method. Upside-down meta-interpretation of the model elimination procedure in effect extends these to non-Horn clauses. We extended it further to allow literals to be assumed as well as proved during a proof and have thus developed an abductive extension as well.

In the context of query evaluation, Bry claimed upside-down meta-interpretation (i.e., rewriting-based query evaluation methods) and top-down evaluation with caching (i.e., resolution-based query evaluation methods) are essentially equivalent instances of his backward fixpoint procedure. However, for theorem proving instead of query evaluation, when a single proof instead of all answers is sought, and heuristic rather than exhaustive search is employed, upside-down meta-interpretation seems preferable.

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, especially in the case of abductive reasoning, where the search space is even greater than for deduction. Although non goal-directed methods such as hyperresolution seem naive even for mathematical problems, they are actually 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 add little goal-directedness.

The high inference rate and low memory consumption of top-down reasoning system such as Prolog and PTTP is 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 even 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 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. METEOR: model elimination theorem proving for efficient OR-parallelism. Masters Thesis, Department of Computer Science, Duke University, 1989.

[2] Astrachan, O. Personal communication, 1991.

[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 SIGMOD-SIGACT Symposium on Principles of Database Systems*, 1986.

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

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

[6] 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.

[7] 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.

[8] Cox, P.T. and T. Pietrzykowski. General diagnosis by abductive inference. *Proceedings of the 1987 Symposium on Logic Programming*, San Francisco, California, August 1987, 183–189.

[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] 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.

[12] 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.

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

[14] Forgy, C.L. RETE: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence 19,* 1 (1982), 17–37.

[15] 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.

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

[17] Hobbs, J.R., M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. Technical Note 499, Artificial Intelligence Center, SRI International, Menlo Park, California, December 1990.

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

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

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

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

[22] McCune, W.W. OTTER 2.0 Users Guide, Technical Report ANL-90/9, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, March 1990.

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

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

[25] 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.

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

[27] Ohta, Y. and K. Inoue. A forward-chaining multiple-context reasoner and its application to logic design. *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, Herndon, Virginia, November 1990, 386–392.

[28] Overbeek, R. and L. Wos. Subsumption, a sometimes undervalued procedure. In Lassez, J.-L. and G. Plotkin (Eds.). *Computational Logic, Essays in Honor of Alan Robinson.* MIT Press, Cambridge, Massachusetts, 1991.

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

[35] Poole, D.L. Compiling a default reasoning system into Prolog. *New Generation Computing 9*, 1 (1991), 3–38.

[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. To appear in *Annals of Mathematics and Artificial Intelligence.*

[43] Stickel, M.E. Rationale and methods for abductive reasoning in natural-language interpretation. In Studer, R. (Ed.). *Natural Language and Logic, International Scientific Symposium, Hamburg, FRG, May 1989, Proceedings.* Lecture Notes in Artificial Intelligence #459, Springer-Verlag, Berlin, Germany, 1990, pp. 233–252.

[44] Stickel, M.E. The path-indexing method for indexing terms. Technical Note 473, Artificial Intelligence Center, SRI International, Menlo Park, California, October 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.

[47] Wos, L., G.A. Robinson, and D.F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *Journal of the ACM 12*, 4 (October 1965), 536–541.