Increasing the Versatility of Heuristic Based Theorem Provers

INCREASING THE VERSATILITY OF HEURISTIC BASED THEOREM PROVERS

Alistair Manning, Andrew Ireland and Alan Bundy

DAI Research Paper No. 633
June 1993

Published in LPAR'93

Increasing the Versatility of Heuristic Based Theorem Provers *

Alistair Manning   *Andrew Ireland*   *Alan Bundy*

Abstract

Heuristic based theorem proving systems typically impose a fixed ordering on the strategies which they embody. The ordering reflects the general experience of the system designer. As a consequence, there will exist a variety of specific instances where the fixed ordering breaks down. We present an approach which liberates such systems by introducing a more versatile framework for organising proof strategies.

1 Introduction

Theorem proving, in general, is a goal-directed activity. In the case of heuristic based theorem provers goal decomposition is achieved by the application of *proof strategies.* A proof strategy can be viewed as having two components: a heuristic component and a guarantee component. The heuristic component determines the applicability of the strategy in a given situation while the guarantee component ensures the soundness of the resulting proof steps.

There are however many situations where more than one strategy is applicable to a given goal. As a result strategies are ordered. This ordering is determined by the system designer and usually reflects their general experience of the domain to which the system is being applied. As a consequence, there will usually exist a variety of specific instances where the fixed ordering of strategies breaks down. To overcome this problem we have developed a more versatile framework for organising proof strategies which supports the re-ordering of strategies dynamically.

Our ideas have been applied to the domain of inductive proof and in §2 we outline two heuristic based theorem provers upon which our work builds. In reviewing

*This paper was published in: *Logic Programming and Automated Reasoning LPAR'93*, St.Petersburg, Lecture Notes in Artificial Intelligence Springer-Verlag 1993. The research reported in this paper was supported by SERC grant GR/H/23610 and SERC studentship 91425046 to the first author.

the traditional approach to organising strategies (§3) we outline the basic flaw in adopting the approach of a fixed ordering of strategies. More detailed motivation for our work is presented in §4 where the benefits of the dynamic re-ordering of strategies is illustrated. Our versatile framework for strategy selection is presented in §5 while implementation details are outlined in §6. Preliminary results are presented in §7 and further extensions are discussed in §8.

2 Heuristic based theorem proving

The technique being presented is applicable in any domain which requires a heuristic based theorem proving approach. Our technique has been applied within the domain of inductive proof and builds upon CLAM-OYSTER [van Harmelen 89], a composite theorem proving system for inductive proof based upon the notion of proof plans [Bundy 88]. The OYSTER component is a re-implementation of the NUPRL [Constable *et al* 86] proof development environment which supports *tactic* based theorem proving [Gordon *et al* 79]. Tactics support goal-directed proof. Primitive tactics correspond to the primitive rules of the logic. In addition, such systems provide tactic combinators which enable the construction of sophisticated strategies built up from the primitive tactics. The CLAM component automates the search for a proof within the OYSTER logic using a predefined set of tactic specifications called *methods*. A method describes explicitly, through declarative preconditions, the general context in which a tactic is applicable. For a given conjecture CLAM constructs a composite method (or proof plan) from the list of methods. This composite method is tailored to the particular conjecture and the corresponding tactic controls the construction of a proof at the OYSTER level.

The strategies captured within CLAM-OYSTER are a rational reconstruction and extension of those embedded within NQTHM [Boyer & Moore 79]. In general terms both systems share the same basic ordering of strategies[1]:

- simplification
- generalisation
- induction

Note that simplification refers to a class of rewriting techniques which preserve equivalence. Typically, simplification involves the application of definitional rewrites. The form of goal generalisation exploited is the replacement of common subexpressions by a new universally quantified variable.

Both systems differ significantly in the way in which strategies are represented. In the case of CLAM-OYSTER the heuristic component is made explicit through a

[1]For presentational purposes we ignored the cross-fertilization strategy which controls the use of hypotheses. Our implementation, however, includes this strategy.

method's preconditions while the guarantee component is encoded separately as a tactic. In contrast, the strategies of NQTHM are "black-boxes". Both the control heuristics and guarantee components for each strategy are integrated with a single subroutine.

While both systems impose a fixed ordering on strategies the "glass-box" approach to expressing proof strategies taken by CLAM-OYSTER gives greatest potential for building flexibility into the application of strategies. It is for this reason that we chose to build directly upon the CLAM-OYSTER system. We will use both systems, however, as the basis for comparison in §7.

3 Fixing the ordering of strategies

As mentioned in §2 both NQTHM and CLAM-OYSTER impose a strict ordering on strategies and only the first successful candidate is applied. Any other strategy not yet tested will not be considered except as a result of backtracking[2]. However, given that some strategies may lead to non-termination one cannot rely upon backtracking to find a proof. Consequently it is very important that the list of allowable strategies is ordered with great care. The general basis of this ordering depends, to differing degrees, upon the following:

- The consequences of a strategy being applied inappropriately
- The degree of complexity involved in testing for the applicability of the strategy;
- The number of subgoals the strategy produces.

Note that the inappropriate application of a strategy may on the one hand lead to unnecessary steps in the resulting proof while on the other hand it may result in the non-termination of the theorem prover. The extent to which a strategy can be inappropriately applied is a very general concept and one that is not considered within this paper. It has been included simply to highlight the problems associated with strategy ordering.

If a strategy produces no subgoals then it should naturally occur early on in the list. Alternatively, if multiple subgoals are produced, as in the case of induction where base and step case subgoals are generated, it should be considered later on in the list since it will lead to further work.

As with any general purpose routine there are usually examples of exceptions to the norm. The wide scope of possibilities that can occur within theorem proving give rise to a large range of examples contrary to the general rules. With this

[2]Note that in the case of NQTHM backtracking is only permitted in very restricted circumstances.

knowledge in mind it is therefore quite apparent that generating a fixed order of strategies to cope with all situations is tricky if not impossible. If the only option is further refinement of a strategy, or more precisely its preconditions, then not only do the strategies become more complicated, they begin to lose their appealing declarative nature. Allowing for flexibility in the ordering of the strategies may therefore be a good way to overcome some of the problems encountered due to specific exceptions to the general rules.

4 Dynamic re-ordering of strategies

In §3 the shortcomings of a fixed strategy ordering within heuristic based theorem proving systems was outlined. We now present concrete examples which demonstrate the need for a framework in which strategies can be dynamically re-ordered.

4.1 Generalisation *vs* Induction

Consider the following conjecture

$$\forall x, y, z : nat.\ x \times (z \times 0) \quad y \times (z \times 0)$$

to which generalisation and induction are applicable:

Generalisation: replacement of the common subexpression $(z \times 0)$ by a new universal variable p;

Induction: on either x, y or z is possible.

If the order in which the strategies are tested for applicability is fixed, as presented in §2, then the generalisation will be applied. Note however that the generalisation presented above gives rise to the following subgoal

$$\forall x, y, p : nat.\ x \times p = y \times p$$

which is easily shown to be a non-theorem.

This problem can be avoided by either reversing the order of the two strategies or by refining the generalisation preconditions to prevent applicability. Changing the re-ordering displaces rather than solves the problem. To illustrate this point consider the conjecture

$$\forall x, y, z : nat.\ x \times (z \times y) = (z \times y) \times x$$

Both generalisation and induction are required to prove (2). However, performing generalisation before induction significantly reduces the complexity of subsequent inductions. Generalisation is thus more appropriate than induction in this case. Increasing the strength of the generalisation preconditions[3], to prevent incorrect applications, without affecting any appropriate usage is extremely difficult (if not impossible) as illustrated in [Hesketh 91]. However, if a more flexible approach could be found which enabled the strategies to be dynamically re-ordered then some of these problems can be addressed.

4.2 Simplification *vs* Generalisation

The above examples illustrate the problems associated with ordering induction and generalisation. This problem is not unique to these two strategies. For instance, consider the conjecture

$$\forall n : nat.\ s(s(n)) \times 1 = s(s(n)) \tag{3}$$

to which both simplification and generalisation are applicable:

Simplification: given the following defining equations for $\times$

$$\begin{aligned} 0 \times Y &\quad 0 \\ s(X) \times Y &\quad\ X \times Y) + Y \end{aligned}$$

$$+ 1) + 1 \quad s(s(n))$$

Generalisation: replacing the subexpression $s(s(n))$ in (3) by a new universal variable, say p, to give

$$\forall p : nat.\ p \times \quad = p$$

In this case, the simplification is unnecessary and a proof can be achieved more directly using the generalisation.

Now consider the conjecture

$$\forall n : nat.\ 0 \times s(s(n)) = (0 + 0) \times s(s(n)) \tag{4}$$

again simplification and generalisation are applicable:

Simplification: given the following defining equations for $+$:

$$\begin{aligned} 0 &+ \\ s(X) &+ \end{aligned}$$

and the definition of $\times$ given above, a proof of (4) is immediate;

Generalisation: replacing the subexpression $s(s(n))$ in (4) by a new universal variable gives

$$\forall p : nat.\ 0 \times p \quad (0+0) \times p$$

In this case the generalisation is unnecessary since simplification goes directly to a proof.

4.3 Simplification *vs* Induction

Finally we consider the relative ordering of simplification and induction. Given the conjecture

$$\forall x : nat.\, 0 \times x \quad x \times 0 \tag{5}$$

the alternatives are as follows:

Simplification: reduces the goal to

$$\forall x : nat.\ 0 = x \times 0$$

Induction: on x is possible.

Here simplification is preferred since it leads to a simplier inductive proof. If an induction is performed first then an additional simplification step is required.

In contrast, consider the conjecture

$$\forall x, y, z : nat.\ x \times (s(s(z)) + y) \quad x \times (z + s(s(y))) \tag{6}$$

here the alternatives are as follows:

Simplification: gives rise to

$$\forall x, y, z : nat.\, x \times s(s(z+y)) \quad x \times (z + s(s(y)))$$

Induction: on x or z is possible.

A proof of (6) requires induction on x. The simplification of the expression $s(s(z))+y$ is unnecessary for the induction to succeed. Induction should therefore be preferred over simplification.

We have shown that in a variety of situations involving different strategies some additional work is performed as a direct consequence of the fixed ordering approach. It could be argued at this point that if it is only extra work and not outright failure which results, then maybe the fixed ordering is after all acceptable. Unfortunately for the current systems this is not so, as illustrated by (1) and more interestingly by:

$$\forall x, y, z : nat.\ x + (s(s(z)) \times y) = x + (y \times s(s(z))) \tag{7}$$

$$\forall x, y, z : nat.\ (s(s(z)) \times x) \times y = s(s(z)) \times (x \times y) \tag{8}$$

In these examples an application of simplification or induction in preference to a generalisation leads to non-termination (see §7).

5 Achieving versatility in strategy ordering

We achieve the dynamic re-ordering of strategies illustrated in §4 by introducing the notion of a strategy's *total promise*, a numeric score, which we defined in terms of the following three features:

- The original rank of the strategy within the theorem prover's fixed ordering;
- The size or complexity of the goal to which the strategy is to be applied;
- A measure of the merit of a strategy which is dependent upon the extent to which its preconditions are satisfied.

Below we describe each of these features in detail together with how the total promise is calculated.

The original ranking of the strategies

This is obviously very important when one considers that most general theorem provers use the fixed ordering of strategies as their only means of assessing a strategy's promise. The higher the position of the strategy within this list the greater the promise the strategy is deemed to have. This rudimentary scoring has value in its simplicity since the strategies are assessed in ranking order, as soon as a strategy is found to be applicable, every other untried strategy is considered to have less promise and hence need not be consulted unless as a result of failure and backtracking.

The new basic versatile promise value follows the same pattern as the old ranking system, with more promise given to the simplification strategy than to the generalisation strategy. This in itself has no effect, it is simply a means of coordinating the score given to the overall strategy's importance, with that of the merit measure of the strategy. In other words only if a strategy's micro promise (merit measure) is striking enough will the original order of the strategies be disturbed.

Goal size and complexity

Consider the expressions $a + b$ and $a^c \times b$. Which is the more complicated? Obviously the second but the actual processes in determining just how much more complex is outwith the scope of this paper. Here it will suffice to explain the general empirical concepts behind this value, as opposed to the actual factors that influence it. In determining this value for any expression, two factors should be considered, namely,

- How many functors and variables are present and where each is located in the expression;

- The complexity of each functor,
  e.g. exponential is defined in terms of $\times$ which is in turn defined using $+$.

The reasoning behind employing this measure is simple enough. If a choice has to be made between pursuing the proof of a goal that is complicated and one that is not, then it is common sense to follow the latter (at least initially). If a proof can be found via this route then it is more likely to be shorter and more compact than one derived via the alternative.

5.3 The measure of a strategy's merit

The general approach with regards to the preconditions of a strategy is that all of the conditions must be satisfied in order for the strategy to be deemed applicable. This is a very stringent test and one that allows little flexibility. The flaw behind this approach lies in the fact that all of the preconditions are given the same importance. If any one is falsified the strategy is not applicable. On the one hand, if the preconditions are too strong then potentially fruitful parts of the search space will not be explored. On the other hand, if preconditions are too weak there is an increased risk of unprofitable and infinite branches of the search space being explored with obvious consequences. The strategy designer is thus forced into a compromise. Our approach liberates the strategy designer from this compromised position by introducing three categories of preconditions:

Standard: conditions which are necessary and sufficient for the legal application of the logical inference which underpins the strategy;

Strengthening: conditions which *increase* the likelihood of the strategy resulting in a proof;

Weakening: conditions which *decrease* the likelihood of the strategy resulting in a proof

Evaluating the applicability of the strategy is now a two part process. Firstly, the standard preconditions are evaluated. These must all succeed for the strategy to be applicable. Secondly, given that the standard preconditions hold, then both the strengthening and weakening preconditions are evaluated. Strengthening preconditions contribute a positive score while weakening preconditions contribute in a negative sense. Consequently, the second part of evaluating the applicability of a strategy may cause the dynamic re-ordering of strategies.

To illustrate in more detail consider the following, simplified, preconditions for the generalisation of subexpressions in the context of equality goals:

Standard: There exists a non-atomic term t in the goal;

Strengthening: The term t occurs on both sides of the equality;

Weakening: No occurrence of t is in a recursive argument position.

Now consider again conjecture (1). Evaluating the standard precondition for generalisation gives rise to three candidate terms for t. The successful evaluation of the strengthening precondition ranks the term $(z \times 0)$ as the most promising candidate for generalising. Note, however, that neither occurrence of $(z \times 0)$ is in a recursive argument position. Consequently, the weakening precondition succeeds and reduces the merit score associated with the application of the strategy. It is this weakening precondition which achieves the dynamic re-ordering of generalisation and induction illustrated in §4.1.

5.4 Calculating total promise

In order to combine the three measures each value is individually weighed against all the others so that large disparities do not arise to upset the global theorem proving system. Most weight, however, is still given to the original strategy ordering; minor abnormalities within a strategy's preconditions should have little if any overall effect.

6 Implementation

The ideas presented above have been implemented as an extension to the CLAM-OYSTER system [Manning 92]. The flexibility of the resulting system manifests itself in the form of a best first search algorithm in which the heuristic score is given by promise of a method (strategy).

As a conjecture is investigated a tree structure is developed in which each branch represents a different disjunctive choice made within the search. Each leaf shows the extent to which its branch has been explored, with the success of a search indicated when a leaf contains no goals. At such a point the complete proof plan of the original conjecture can be determined by following the branch from the leaf to the root.

At every stage of the search the most promising leaf is extended by one step, thus creating new leaves, each with its own promise value. At no stage is a leaf discarded, even the most unlikely ones are maintained, albeit with a low priority. In this manner the search can opportunistically traverse the search space, each time examining only the most promising leaf.

Each leaf is comprised of a conjunction of methods each with its own input goal and promise value. When a leaf is extended, each method is applied to its respective input goal generating a series of subgoals. Each subgoal is tested for applicability with respect to the method set, thus creating a list of applicable methods and related promise values, one for each subgoal. These lists are combined together in all possible ways to generate a number of sets, such that one and only one element from each list is found in each set. A set represents a new leaf with a promise value equal to its lowest priority element.

7 Comparative study

In both the NQTHM and CLAM-OYSTER systems limited "ad-hoc" versions of strengthening conditional merits have been implemented within some strategies. In neither case, however, does this allow the merit measure of a strategy to alter the actual order of the strategies, nor to incorporate weakening merits.

The ranking of strategies is such that our implementation performs as well as, if not better than, CLAM-OYSTER on its library of theorems[4]. The implementation was also tested with the examples given in §4. The results together with a comparison with CLAM-OYSTER and NQTHM are summarised in table 1.

Examples (1), (3), (4), (5) and (6) all illustrate that the versatile CLAM-OYSTER system either produces the same or an improved proof over those generated by the other two systems. Note the extra work, however, required by CLAM-OYSTER in the case of examples (3) and (6).

The other examples (2), (7) and (8) are more illuminating as they illustrate failures of the fixed strategy ordering systems. It is interesting to note that NQTHM and CLAM-OYSTER have different approaches to overcoming initial failure.

NQTHM has an "ad-hoc" pre-programmed mechanism which appears to force an induction if a simplification fails. This, as (7) and (8) exemplify, can result in an opportunity for generalisation being missed with fatal consequences.

In contrast CLAM-OYSTER relies on a pre-deterministic backtracking mechanism, i.e. one that returns to the fixed strategy list. This has weak points in that inappropriate strategies are applied simply due to their rank with no account taken of the current situation. In (7) and (8), an initial simplification prevents a profitable generalisation. Here the error occurred in its first step but its consequence (i.e. failure) does not arise until deep within the proof. The backtracking, being domain independent, is unable to return to this initial stage before an infinite branch of the search space is encountered.

The versatile CLAM-OYSTER system improves on these two approaches by being more sensitive in its application of strategies (i.e. by choosing only the most promising in each situation). It tries to avoid the inappropriate application of strategies which would otherwise lead to extra work or failure. In addition, because

[4]This library is a subset of the NQTHM corpus.

of its use of the Best First Search algorithm its backtracking is opportunistic. Even an inappropriate application anywhere within a search can be easily rectified.

8 Conclusions and further work

As can be seen from the discussion in §7 the incorporation of flexibility into strategy ordering has improved upon the performance of both the NQTHM and CLAM-OYSTER systems.

The technique examined in this paper can be extended in a variety of different ways. For example, consider the following conjecture

$$\forall y, z : nat.\ \ 0 + (s(s(z)) \times y) = y \times s(s(z))$$

Ideally we would like to only partially[5] simplify this goal using the definition of $+$ to give

$$\forall y, z : nat.\ \ s(s(z)) \times y = y \times s(s(z))$$

Now a generalisation sets us up for a relatively straightforward induction:

$$\forall y, p : nat.\ \ p \times y = y \times p$$

Such an extension requires only the inclusion of extra strengthening or weakening preconditions with associated merit scores.

In conclusion, our results serve to question the advantages of pursuing a fixed ordering on proof strategies. Can a strategy's preconditions be fine tuned so as to only be satisfied when it is profitable to do so? Is such fine tuning compatible with maintaining declarative preconditions? Empirical evidence will determine the answers to these questions. Our extension to the CLAM-OYSTER system provides a framework in which this evidence can be gathered.

References

[Boyer & Moore 79] R.S. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, 1979. ACM monograph series.

[Bundy 88] A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[5]Note that currently simplification is an "all or nothing" strategy.

Eqn	CLAM-OYSTER		Nqthm		Versatile CLAM-OYSTER	
1	$\surd$	generalises then backtracks and finds an inductive proof	$\surd$	avoids generalisation and finds an inductive proof	$\surd$	avoids generalisation and finds an inductive proof
2	$\surd$	generalises to commutativity of $\times$ then finds an inductive proof	χ	misses initial generalisation and attempts an induction *(non-termination)*	$\surd$	generalises to commutativity of $\times$ then finds an inductive proof
3	$\surd$	unnecessarily simplification followed by induction	$\surd$	unnecessarily simplification followed by induction	$\surd$	a generalisation instead of simplification is performed followed by an induction
4	$\surd$	simplification	$\surd$	simplification	$\surd$	simplification
5	$\surd$	simplification followed by induction	$\surd$	simplification followed by induction	$\surd$	simplification followed by induction
6	$\surd$	unnecessary simplification followed by induction	χ	unnecessary simplification followed by an induction attempt *(non-termination)*	$\surd$	Ignores unnecessary simplification and finds an inductive proof
7	χ	unnecessary simplification eliminates chances of generalisation and subsequent induction fails *(non-termination)*	χ	simplification then backtracks and attempts an induction *(non-termination)*	$\surd$	generalisation preferred over simplification followed by an inductive proof
8	χ	unnecessary simplification eliminates chances of generalisation *(non-termination)*	χ	simplification then backtracks and attempts an induction *(non-termination)*	$\surd$	generalisation preferred over simplification followed by an inductive proof

Table 1: Summary of results

The equation numbers refer to the examples given in §4. $\surd$ and χ indicate where proof attempts succeed and fail respectively.

[Constable *et al* 86] R.L. Constable, S.F. Allen, H.M. Bromley, *et al.* *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[Gordon *et al* 79] M.J. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF - A mechanised logic of computation*, volume 78 of *Lecture Notes in Computer Science.* Springer Verlag, 1979.

[Hesketh 91] J.T. Hesketh. *Using Middle-Out Reasoning to Guide Inductive Theorem Proving.* Unpublished PhD thesis, University of Edinburgh, 1991.

[Manning 92] A. Manning. Representing preference in proof plans. Unpublished M.Sc. thesis, Dept. of Artificial Intelligence, Edinburgh, 1992.

[van Harmelen 89] F. van Harmelen. The CLAM proof planner, user manual and programmer manual: version 1.4. Technical Paper TP-4, DAI, 1989.