High Performance Reasoning Systems: Some Opportunities[1]

John Slaney[2]

1 Introduction

The notion of reasoning comprises several activities not all of which have received the same degree of attention from those of us whose approach is from logic. Three of these activities occupy my attention today. They are deduction or the proving of theorems, its dual model generation or the satisfaction of constraints, and planning or the transformation of one model of a theory into another through a series of intermediate states linked by an action relation. These three fields have grown up to be the professional concern of three rather different intellectual communities. It has long been part of my purpose, and is so again today, to attack the barriers between these communities in the interests not only of cross-fertilisation at the peripheries but of significant advance in the heart of each enterprise.

I want to examine several examples of fruitful or potentially fruitful interactions between aspects of reasoning. Along the way, I want to urge that we direct energies both into exploring what research in one kind of reasoning can offer to another and into implementing our ideas in such a way that we strive for performance, aiming the best code we can write at the best hardware available. By doing this we may take advantage of real opportunities for achievement which stand before us at this stage of development of the discipline.

In choosing case studies to illustrate there themes, I have no wish to slight other examples which could have been used. For instance, I say nothing about constraint logic programming, although that is the most oustandingly successful application of constraint satisfaction techniques in deduction. I also say nothing about nonmonotonic reasoning, nor about inductive reasoning. The reason is that I prefer to illustrate from research in which I have been actively involved, and in any case I expect to be more entertaining on the subjects which excite me today than those which did so last year or may do so next.

2 Constraint Satisfaction

My first example is the use of propositional satisfiability techniques from the world of theorem proving to solve pure constraint satisfaction problems in finite algebra. The problems in question concern the existence or non-existence of quasigroups (in

[1]The ideas in this talk were first presented at the "Automated Reasoning Day" mini-conference at Bribie Island, Queensland in September 1994. The present account is another attempt to explore the relationship between aspects of our field. Thanks are due to the participants in the Automated Reasoning Day for their comments and suggestions.

[2]John.Slaney@anu.edu.au. Automated Reasoning Project, Australian National University, Canberra, ACT 0200, Australia.

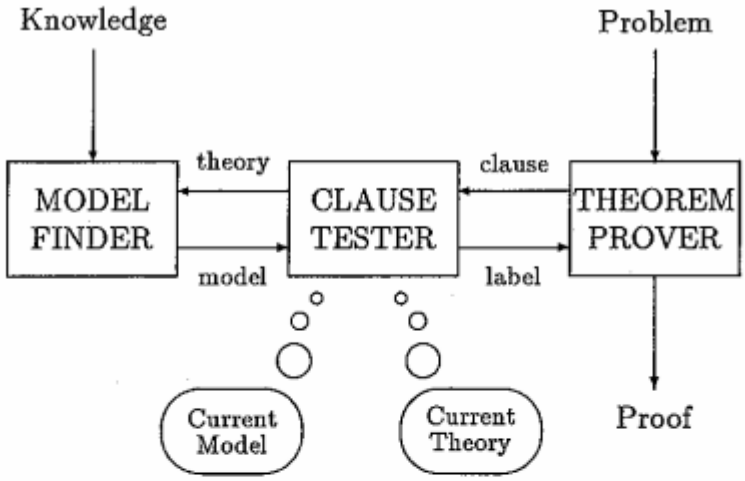

Figure 1: Semantics in Theorem Proving

effect, Latin squares) satisfying certain simply specified conditions. This work, which began in a serious way at ICOT in 1992, using MGTP-G, and has continued to the present with the involvement of a number of researchers elsewhere.

I shall not detail the quasigroup problems or the methods applied to them, since they have been described elsewhere[2, 9] and other speakers in today's workshop are best able to supply the most recent information concerning them. I shall merely say briefly that by bringing methods from one area of automated reasoning to bear on problems in what has been regarded as a different one we were able to achieve significant results, and that high performance implementation has enabled us to extend our work far enough into the field to solve a large number of open problems. Some of the solutions have been published in the mathematics literature, and others will certainly follow.

3 Theorem Proving

The second case study is the theorem prover SCOTT (Semantically Constrained Otter) described in [7]. This was made by linking together the existing theorem prover OTTER[6] and the existing model generator FINDER[8]. In 1992–3 my colleague Dr Mark Grundy and I spent time at ICOT working on the idea of a similar system using MGTP-G to generate the models and MGTP-N to find the proofs.

There are several ways in which a bottom-up prover can use semantic information to help guide its proof search. In SCOTT, two of these are exploited. The first is model resolution, and the second is the false preference strategy whereby in choosing parent clauses for inferences the prover gives some preference to those which are false in a guiding model. In each case the model is not given in advance but found dynamically as the search progresses. The model is generated by the constraint satisfaction module, attempting to model formulae deduced during the proof search. The model may from time to time be replaced by a better one as more information

becomes available. In this way prover and modeller work together to explore the search space.

On a range of the problems for which OTTER and MGTP-N are very suitable, SCOTT speeds up OTTER by about a factor of 2, though in certain cases it wins by three orders of magnitude and of course in other cases it allows the user new ways to fail altogether. Since it has OTTER as a sub-program and can run with all else disabled, SCOTT need never lose in comparison with OTTER.

4 Planning

The last case study concerns classical planning problems. It is well known that such problems may be solved by theorem provers, and indeed that the search for a plan may be cast as a search for a proof. A plan in the sense considered here[3] is a sequence of actions, each of which suffices to convert a state into the next state in the sequence. From a description of the initial state and an axiomatisation of the conditions and effects of the actions, it follows that successive states, including the goal, are all reachable. A plan may be read off a proof of reachability.

Also obvious is that a classical planning problem may be represented as a demand not for a proof but for a model. If states and actions are objects of a domain, along with the objects on which the actions operate (e.g. the blocks in Blocks World), then a plan is a structure made up of those objects satisfying the axioms of state transition by actions, the first and last states in which are the initial and final states of the problem. Both backtracking methods such as the Davis-Putnam algorithm and non-backtracking methods such as GSAT have been applied with some success to small problems.[5]

The outstanding feature of these experiments in planning by logic, however, is the extremely poor performance of the systems. The example used is (as ever) Blocks World, and the largest problems examined in [5] have only nine blocks! Finding optimal (shortest) plans in Blocks World is NP-complete[1, 3] but it is rather easy to find algorithms which produce "good" solutions in polynomial time.[4, 10]. These go close to optimality with far more blocks than nine.

What is better is to find more imaginative ways of using automated reasoning, and constraint solving in particular, to augment methods already seen to be powerful in classical planning. In contrast to the cases noted above, here I have no proven solutions but only suggestions. The key is a theorem from [10] that the problem of generating Blocks World states is tractable. Standard algorithms like Davis-Putnam or arc consistency never backtrack when searching for such states, so generating models of Blocks World is $\mathcal{O}(n^3)$ and cheap compared with generating [near] optimal plans.

One suggestion for using model generation in planning is to devise states "intermediate" between the initial and goal states of large planning problems. Such ideas are of course common in AI. Now by iteratively adjusting the intermediate state, it

[3]Planning is not a field consisting of a single problem type. There are many other sorts of plan and other sorts of planner. Once again, I wish to point out that I am not attempting to talk about *everything*.

may be possible to arrive at a subgoal which is in some sense " not too far" from the initial one and "not too far" from the goal. We can use this to break an impossibly big planning problem into two more manageable ones. The resulting overall plan will not be optimal in general, but with large problems, as already pointed out, optimality is not attainable.

A second suggestion is a way of reducing certain non-classical planning problems to classical ones. Sometimes we are given a goal not as a conjunction of positive literals but just as a first order condition. Algorithms such as those in [4] and [10] cannot be used with such a formulation, but a finite domain constraint solver can generate models of the non-atomic description. By generating such a model and then refining it to make it "closer to" the initial state it is possible to turn such a generalisd planning problem into one suitable for standard methods.

5 And Finally...

The cases noted in this talk are meant to stand as instances of what can be done if the fragmentation of our research field is overcome to some extent and if we care about the quality of our implementation work. ICOT, with its ready interchange of ideas between sub-projects and its emphasis on leading edge technology, has been an exemplar in this regard. Let us see to it that this example is followed.

References

[1] S. Chenoweth, *On the NP-hardness of Blocks World*, **Proc. AAAI-91** (1991) 623–628.

[2] M. Fujita, J. Slaney & F. Bennett, *Automatic Generation of Some Results in Finite Algebra*, **Proc. 13th IJCAI** (1993) 52–57.

[3] N. Gupta & D. Nau, *Complexity Results for Blocks-World Planning*, **Proc. AAAI-91** (1991).

[4] N. Gupta & D. Nau, *On the Complexity of Blocks-World Planning*, **Artificial Intelligence** 56 (1992) 223–254.

[5] H. Kautz & B. Selman, *Planning as Satisfiability*, **Proc. 10th ECAI** (1992) 359–363.

[6] W. McCune, *OTTER 3.0 Users Guide*, Argonne National Laboratory, 1994.

[7] J. Slaney, *SCOTT: A Model-Guided Theorem Prover*, **Proc. 13th IJCAI** (1993) 109–114.

[8] John Slaney, *FINDER, Finite Domain Enumerator: System Description*, **Proc. 12th CADE** (1994) 798–801.

[9] J. Slaney, M. Fujita & M. Stickel, *Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems*, **Computers in Mathematics with Applications**, special issue on Automated Reasoning, forthcoming 1994.

[10] J. Slaney & S. Thiébaux, *Adventures in Blocks World*, Technical report TR-ARP-7-94, Automated Reasoning Project, Australian National University, 1994.