Abstract. This paper is about automatic searching for proofs, automatic searching for models and the potentially fruitful ways in which these traditionally separate aspects of reasoning may be made to interact. It takes its starting point in research reported in 1993 (Slaney, SCOTT: A Semantically Guided Theorem Prover, Proc. 13th IJCAI) on a system which combines a high performance first order theorem prover with a program generating small models of first order theories. The main theorem is an incompleteness result for a certain range of problems to which this combined system has been successfully applied. While the result may not be unexpected, the proof is worth examining and it is important to reflect on its relationship to the research program in combining methods.

1. Proof search, model search and their interaction

1.1. BACKGROUND

Traditional theorem provers search without much intelligence. They may reason forwards from the axioms or backwards from the desired theorem or both, but in either direction they rapidly find themselves in an exponentially growing search space of possible proof fragments which they explore in a manner at once admirably industrious and remarkably dull. Much good work in automatic proof search has gone into the discovery and refinement of methods for controlling the explosion of the search space or for reducing the amount of duplicated work undertaken in traversing it. At the same time, there has been vigorous research on heuristics for directing the search for proofs, but many of the most successful ideas in that field are either unexciting suggestions such as exploring short formulae first or else mysterious ones such as preferring some operator to be nested to the left rather
The root problem is that most powerful theorem provers work only locally, focussing on the specific formulae being transformed by an inference rather than on global aspects of the situation, and more significantly they are based on pure syntax. They may take into account questions like how many function symbols a formula has, which is the first literal in a clause, whether this unifies with that and the like, but they do not consider what the inference step is supposed to achieve, whether it is establishing a general law for the structures under consideration or an accidental property of the case, whether the conclusion of the inference says the same thing as one already proved or the like. That is, traditional provers do not understand what they are doing.

Still less do they understand the problems they are attempting to solve. The contrast with human theorem provers is striking. When we reason, we appeal constantly to conceptual structures within which particular problems make sense. We are not capable of exploring search spaces of millions of clauses, and it seems that we do not need to. What we are able to use is some sense of when a proof search is getting closer to the goal. What lies behind this capacity is not so much an ability to recognise syntactic patterns in formulae (though we importantly have that too) as an understanding of the problem: we know what the symbols, theories, axioms, goals and subgoals mean, and this puts us at an advantage in the investigation.

These remarks may be taken as leading to a recommendation that researchers in automated reasoning direct their efforts towards mechanical emulation of human cognitive processes. This, however, would be a mistake, certainly in the present state of development of the discipline. In the details of what they do—in what they find easy and what hard—computers are quite unlike us. Without becoming totally discouraged from projects in programming naturalistic intelligence, we should recognise that for practical purposes the indicated way ahead is to continue to let the machines be mechanical, doing what they do best at the high speeds of which they are capable. Algorithmic proof search is good for some things anyway; it no