

# Semantically Guiding a First-Order Theorem Prover with a Soft Model

**Arnold Binas**

Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112  
Tel: 979-862-6801  
arnold-binas@neo.tamu.edu

**John Slaney**

Computer Science Laboratory, RSISE  
The Australian National University  
Canberra, ACT, Australia  
John.Slaney@anu.edu.au

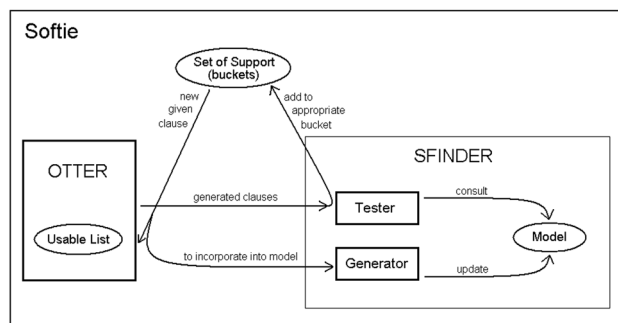
Various versions of our first-order logic theorem prover SCOTT have been developed over the past decade to employ the concept of semantic guidance for improving the underlying system OTTER by McCune (Slaney, Lusk, & McCune 1994; Hodgson & Slaney 2001; 2002). We introduce our latest attempt to speed up OTTER's proof search, Softie. While the various SCOTTs consulted an ordinary constraint solver to gain information about the problem to be solved, Softie is implemented from scratch and uses a solver capable of handling soft constraints.

The first-order theorem prover OTTER searches for refutation proofs by the means of the *given clause algorithm*, which partitions the clauses of a problem into the *usable list* (UL) and the *set of support* (SOS). So-called *given clauses* are selected from the SOS, moved to the UL, and react under inference rules to generate new clauses (McCune 1994). While OTTER uses purely syntactic criteria to select the next given clause, SCOTT bases that decision on the truth-values of clauses in several models of a subset of the problem's clauses. The models are supplied by the constraint solver FINDER (Slaney 1994). As the goal of a refutation proof search is to arrive at the empty clause, clauses false in the guiding model are preferred as next given clauses. This approach is known as the *false preference strategy*.

When dealing with hard constraints, multiple models improve the quality of semantic guidance, but their maintenance is very costly. The new Softie uses a guiding model that is generated by SFINDER, a version of FINDER that handles soft constraints. While the underlying idea of semantic guidance remains the same as in previous versions of SCOTT, Softie is built from scratch and maintains a single, largely soft model of almost the whole current UL.

The figure below illustrates the basic workings of Softie. The SOS is partitioned into buckets to differentiate both between lengths of clauses and their truth-values in the current model. A new given clause is selected from the SOS mostly<sup>1</sup> using the false preference strategy, incorporated into SFINDER's model, and used by OTTER to generate new clauses. The new clauses that OTTER generates are tested

against the current model and put into the appropriate buckets of the SOS. Testing a clause against the current model is relatively cheap as the model is very small (we use a domain of only around three elements). The process is then repeated until the empty clause has been derived and a proof found.



As its predecessor FINDER, SFINDER uses a generic sort with a finite domain for all variables and functions in the theory it is modeling. In Softie's proof search, the clauses initially in the UL (before the proof search starts) are treated by SFINDER as hard constraints to give the model a general direction. In most cases, the clauses in the initial UL are consistent with each other as they are axioms, so a model is still guaranteed to be found. Clauses that are selected as given clauses during the proof search are then added to the set of clauses being modeled. Due to the way SFINDER works, only so-called short clauses can be incorporated into the model as soft constraints. The number of literals a clause can have and still be considered short is set as a parameter in Softie. All selected given clauses that are short are incorporated into the model as soft constraints. Treating the constraints imposed by short clauses as soft opens a way to consider a larger portion of the problem's clauses while still finding a model. Whenever a new clause is true in the current model, it is added without any changes to the model. Otherwise, a new model including the new clause is sought. Thus, SFINDER always supplies Softie with a model that satisfies as many instances as possible, fitting the model to most of the UL clauses and making it more representative of the problem. As another parameter in Softie, SFINDER is given a time limit after which it returns the best model found up to that point.

Softie has been tested with various parameter settings on

<sup>1</sup>Every now and then a clause is selected without semantic guidance to ensure the completeness of the approach.

several eligible problems from the CADE-18 ATP System Competition (Sutcliffe & Suttner 2002) and compared to runs of its underlying system OTTER on the same set of problems. At this early stage, the results are mixed, yet encouraging. While there are problems that OTTER solves within a time limit of ten minutes that Softie cannot solve in that period, there are several problems that Softie solves in well under ten minutes and which OTTER cannot prove in the given time. For harder problems that both systems solve within the time limit, Softie generally takes fewer given clauses before finding a proof than OTTER, which indicates that the semantic guidance is successfully focusing the search. In many cases, however, that comes at a high cost in time spent on the model searches, effectively causing Softie to take longer than OTTER for many problems. These experiments show that Softie is headed in the right direction, but it is far from finished. However, it is already many times faster than the previous versions of SCOTT. A major potential improvement we plan to implement in the future is for Softie to utilize its guiding model not only for the false preference strategy, but also for *semantic resolution* (Slagle 1967). Under this approach, clauses in the UL are labeled with a truth-value as well. Semantic resolution then calls for at least one parent of each newly generated clause to be false in the guiding model. Other areas of future work include tuning the various settings and details of the clause selection algorithm as well as making Softie learn features of problems and thereby enabling it to adapt to different problem classes autonomously.

The presented system exposes the potential of using soft models in semantically guided theorem proving. We consider it a good start in further exploring how to speed up the proof search of contemporary first-order provers.

### Acknowledgements

This work was supported by the National ICT Australia. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

### References

- Hodgson, K., and Slaney, J. 2001. System Description: SCOTT-5. In *Proc. First International Joint Conference on Automated Reasoning*, 443–447. Berlin, Heidelberg: Springer Verlag.
- Hodgson, K., and Slaney, J. 2002. TPTP, CASC and the Development of a Semantically Guided Theorem Prover. *AI Communications* 15(2):135–146.
- McCune, W. W. 1994. OTTER 3.0 Reference Manual and Guide. Technical Report, ANL-94/6, Argonne National Laboratory.
- Slagle, J. R. 1967. Automatic Theorem Proving With Renamable and Semantic Resolution. *Journal of the Association for Computing Machinery* 14(4):687–697.
- Slaney, J.; Lusk, E. L.; and McCune, W. 1994. SCOTT: Semantically Constrained Otter System Description. In

*Proc. 12th Conference on Automated Deduction*, 764–768. Berlin, Heidelberg: Springer Verlag.

Slaney, J. 1994. FINDER: Finite Domain Enumerator (System Description). In *Proc. 12th Conference on Automated Deduction*, 798–801. Berlin, Heidelberg: Springer Verlag.

Sutcliffe, G., and Suttner, C. 2002. The CADE-18 ATP System Competition. Retrieved April 1, 2004 from the World Wide Web  
<http://www.cs.miami.edu/~tptp/CASC/18/>.