Automatic Generation of Implied Constraints: Initial Progress

Simon Colton, Lyndon Drake, Alan M. Frisch, Ian Miguel and Toby Walsh
Artificial Intelligence Group
Department of Computer Science, University of York
York, YO10 5DD, United Kingdom
{simonco, lyndon, frisch, ianm, tw}@cs.york.ac.uk

Constraint satisfaction is a highly successful technology for tackling a wide variety of search problems including resource allocation, transportation and scheduling. Several recent studies show that implied constraints added by hand to a problem representation can lead to significant reductions in search (e.g. [8]). The aim of this project is to develop, analyse and evaluate methods for generating useful implied constraints automatically. Our initial progress comprises three strands, which are outlined below.

Proof Planning

Implied constraints are simply logical consequences of the problem specification. Bundy's "proof planning" approach [1] appears to be one of the most promising techniques for dealing with the combinatorially explosive problem of generating useful logical consequences. Proof planning has often been associated with "rippling" [2], a powerful heuristic for guiding search in inductive proof. However, proof planning can easily be adapted to other mathematical tasks like finding closed form sums to series [9] or, as here, generating logical consequences which may make useful implied constraints.

Consider the following fragment of a constraint satisfaction problem, where the goal is to assign values to the variables whose domains consist of a finite set of integers:

- $A = \frac{B+C+D}{E}$
- $B \leq C \leq D$

Given a method, `eliminate`, a generalisation of Gaussian elimination whose preconditions identify a non-linear constraint and find a sub-term to eliminate, we can eliminate C and D in favour of B:

- $A \geq \frac{2B+D}{E}$ since $B \leq C$
- $A \geq \frac{3B}{E}$ since $B \leq D$

A second ternary constraint can be obtained similarly by eliminating in favour of D. These constraints are likely to prune more of the search tree than the initial 5-ary one due to their small arity.

Proof planning offers several potential advantages over other theorem proving techniques for this task. First, methods can be given very strong preconditions which limit the generation of logical consequences to those that are likely to make useful implied constraints. Second, methods can act at a very high level. For example, they can perform complex rewriting, simplifications, and transformations. Such steps might require very long and complex proofs to justify at the level of individual inference rules. And third, the search control in proof planning is cleanly separated from the inference steps. We can therefore easily try out a variety of different search strategies like best-first search or limited discrepancy search.

Theorem Generation

We are also exploring an alternative approach to the automatic generation of implied constraints, exploiting the HR mathematical discovery program [4]. HR is provided with background information about a type of mathematical object such as quasigroups and used to make conjectures which it, in turn, uses Otter [6] to try and prove. A constraint solver can use the resulting theorems as implied constraints.

A limitation of this approach is that, if completeness is an issue, only problem classes within which conjectures can be proven by existing theorem provers may be considered. Nevertheless, some encouraging results have been obtained. For example, in quasigroup class 3, HR produced the implied constraint that elements of the leading diagonal must be pairwise distinct. Adding this constraint to the basic problem representation produces a 3-fold

reduction in search effort on average when determining the existence of a quasigroup of a particular size from that class. We hope that similar positive results will be obtained in other quasigroup classes and related problems.

Implied Clauses in Propositional Satisfiability

Proposition satisfiability (SAT) is a special case of a constraint satisfaction problem, consisting of Boolean variables constrained by a set of clauses. Given a SAT problem, it is generally possible to infer additional clauses. These implied clauses, while they include no new information, may alter the behaviour of a search algorithm on the problem instance so that the search terminates more quickly [3, 5]. We will explore methods of automatically generating such implied clauses, initially for use by Davis-Putnam solvers. A generally useful algorithm will have to find a balance between the effort spent on inference and search, and this balance will be dependent on the problem type and instance [7].

The standard DP algorithm uses only unit resolution. We plan to examine other forms of resolution and inference. Possible resolution strategies include: ordered resolution; resolutions that remove all occurrences of a variable; resolutions that produce a short resolvent; and resolutions that produce a resolvent which subsumes one or both of its parents. The strategy chosen is likely to largely determine the usefulness of the generated clauses.

Next Steps

Our immediate plans include the development of a more sophisticated means of gauging the expected utility of implied constraints/clauses. Currently, simple criteria such as constraint arity and tightness are used to select implied constraints for use. However, it is envisaged that more informed criteria will eventually be used, for example taking into account the effect of adding an implied constraint to the connectivity of the constraint graph.

Acknowledgements

This project is supported by EPSRC Grant GR/N16129[1]. The first author is supported by EPSRC Grant GR/M98012. The final author is supported by an EPSRC advanced research fellowship.

References

[1] A. Bundy. A science of reasoning. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 178–198. MIT Press, 1991.

[2] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.

[3] B. Cha and K. Iwama. Adding new clauses for faster local search. *Proceedings of AAAI-96*, 1996.

[4] S. Colton, A. Bundy, and T. Walsh. Automatic concept formation in pure mathematics. *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 786 – 791, 1999.

[5] J. P. Marques Silva and K. A. Sakallah. Conflict analysis in search algorithms for satisfiability. *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, Nov 1996.

[6] W. McCune. The Otter user's guide. Technical report ANL/90/9, Argonne National Laboratory, 1994.

[7] I. Rish and R. Dechter. *Resolution versus Search: Two Strategies for SAT*, pages 215–259. Frontiers in Artificial Intelligence and Applications. IOS Press, 2000.

[8] B. Smith, K. Stergiou, and T. Walsh. Modelling the Golomb ruler problem. *Proceedings of the IJCAI-99 Workshop on Non-binary Constraints*, 1999.

[9] T. Walsh, A. Nunes, and A. Bundy. The use of proof plans to sum series. In *Proceedings of the 11th Conference on Automated Deduction*, pages 325–339. Springer Verlag Lecture Notes in Computer Science 607, 1992.

[1] `http://www.cs.york.ac.uk/aig/projects/implied/index.html`