Call Subsumption Mechanisms for Tabled Logic Programs

Flávio Manuel Fernandes Cruz

Projecto/Dissertação realizado sob a orientação do *Prof. Ricardo Rocha*
no *CRACS / FCUP*

1. Background

Tabling is a particularly successful resolution mechanism that overcomes some limitations of the SLD resolution method found in Prolog systems [1], namely, in dealing with recursion and redundant sub-computations. In comparison to the traditional SLD resolution method, tabling can reduce the search space to cut redundant computations, avoids looping and has better termination properties [2]. In tabling, first calls to tabled subgoals are evaluated through program resolution, while *similar calls* are evaluated by consuming answers stored in the table space by the corresponding similar subgoal. In general, we can distinguish between two main approaches to determine if a subgoal A is similar to a subgoal B: *variant-based tabling* and *subsumption-based tabling*. In variant-based tabling, A is similar to B if they are the same up to variable renaming. In subsumption-based tabling, A is similar to B when A is subsumed by B (or B subsumes A). This stems from a simple principle: if A is subsumed by B and S_A and S_B are the respective answer sets, then $S_A \subseteq S_B$. While subsumption-based tabling (or *call subsumption*) can yield superior time performance by allowing greater answer reuse, its efficient implementation is harder than variant-based tabling, which makes tabling engines with variant checks much more popular in the logic community.

2. Purpose

This thesis first addresses the porting and integration of the *Time Stamped Tries* (TST) mechanism from the SLG-WAM [3] into YapTab [4]. This mechanism was proposed by Ernie Johnson *et al.* [5, 6] and implements the algorithms and data structures that support subsumption-based tabling. The TST technique is based on the idea of extending the table space with time information to distinguish between new answers from old answers.

In the second part of this thesis we present the design, implementation, and evaluation of a novel extension based on subsumption-based tabling called *Retroactive Call Subsumption* (RCS) [7]. RCS overcomes some limitations of traditional call subsumption, namely, the fact that the call order of the subgoals can greatly affect its success and applicability. RCS allows full sharing of answers, independently of the order they are called by selectively pruning and restarting the evaluation of subsumed subgoals. To implement retroactive-based tabling we developed a few novel ideas: (1) a novel algorithm to efficiently retrieve running *instances* of a subgoal; (2) a novel table space organization, where answers are represented only once; and (3) a new evaluation strategy capable of pruning and transforming generator subgoals into consumer subgoals.

3. Results

Our performance results show that the integration of TST mechanisms and algorithms from the SLG-WAM to YapTab was largely successful, with comparable speedups when using subsumptive-based tabling against variant-based tabling.

For the RCS engine, our results show that the overhead of the new mechanisms for RCS support are low enough in programs that do not benefit from it, which, combined with considerable gains for programs that can take advantage of them, validates this new evaluation technique.

4. Conclusions

The main contributions of this thesis are the following: (1) support for subsumption-based tabling in Yap Prolog; (2) the novel RCS technique that permits bidirectional reuse of answers; (3) a novel table space organization that enhances answer reuse by allowing answer reuse on a predicate basis; and (4) a tabling engine capable of mixing multiple evaluating strategies, such as variant, subsumption and retroactive-based tabling. Our final system enables the programmer to choose the best evaluation strategy per predicate, which arguably can augment the power of tabling for real world programming.

References

[1] W. Chen and D. S. Warren. Tabled Evaluation with Delaying for General Logic Programs. *Journal of the ACM*, 43(1):20–74, 1996.

[2] H. Tamaki and T. Sato. OLDT Resolution with Tabulation. In *International Conference on Logic Programming*, number 225 in LNCS, pages 84–98. Springer-Verlag, 1986.

[3] K. Sagonas. *The SLG-WAM: A Search-Efficient Engine for Well-Founded Evaluation of Normal Logic Programs*. PhD thesis, Department of Computer Science, State University of New York, 1996.

[4] R. Rocha, F. Silva, and V. Santos Costa. YapTab: A Tabling Engine Designed to Support Parallelism. In *Conference on Tabulation in Parsing and Deduction*, pages 77–87, 2000.

[5] E. Johnson, C. R. Ramakrishnan, I. V. Ramakrishnan, and P. Rao. A Space Efficient Engine for Subsumption-Based Tabled Evaluation of Logic Programs. In *Fuji International Symposium on Functional and Logic Programming*, number 1722 in LNCS, pages 284–300. Springer-Verlag, 1999.

[6] Ernie Johnson. Interfacing a Tabled-WAM Engine to a Tabling Subsystem Supporting Both Variant and Subsumption Checks. In *Conference on Tabulation in Parsing and Deduction*, 2000.

[7] F. Cruz and R. Rocha. Retroactive Subsumption-Based Tabled Evaluation of Logic Programs. In T. Janhunen and I. Niemelä, editors, *Proceedings of the 12th European Conference on Logics in Artificial Intelligence, JELIA'2010*, LNCS, Helsinki, Finland, September 2010. Springer-Verlag.