DELTA — A Bottom-up Preprocessor for Top-Down Theorem Provers — System Abstract —*

Johann M. Ph. Schumann

Institut für Informatik,
Technische Universität München
D-80290 München
email: schumann@informatik.tu-muenchen.de

Top-down theorem provers with depth-first search (e.g., PTTP [Sti88], METEOR [AL91], SETHEO [LSBB92]) have the general disadvantage that during the search the same goals have to be proven over and over again, thus causing a large amount of redundancy. Resolution-based bottom-up theorem provers (e.g., OTTER [McC90]), on the other hand, avoid this problem by performing backward and forward subsumption and by using elaborate storage and indexing techniques. Those provers, however, often lack the goal-orientedness of top-down provers.

In order to combine the advantages of top-down and bottom-up theorem proving, we have developed the preprocessor DELTA. DELTA processes one part of the search space (the "bottom" part) in a preprocessing phase, using bottom-up techniques (see also [?]). It generates unit-clauses (e.g., by applying UR-resolution) which are added to the original formula. Then, this formula is processed by a top-down theorem prover in the usual way.

During this top-down search, the additional unit clauses are used as generalized unit lemmata. Due to the structure of the search space and the combination of advantages of both approaches (subsumption in the preprocessing phase and goal-oriented search in the subsequent top-down search), a remarkable gain of efficiency can be achieved in many cases.

DELTA uses SETHEO [LSBB92] and its logic programming facilities for generating these unit clauses. In order to obtain a high efficiency for the bottom-up phase, the unit clauses are generated level by level. This technique is similar to delta iteration as it is used in the field of data-base research.

Starting with the original formula, the following iteration step is performed (first, we only consider Horn-formulae): we let SETHEO generate all new unit clauses which can be obtained by one UR-resolution step out of the current formula. This is accomplished by adding to the formula "most general queries" $\neg p(X_1, \ldots, X_n)$ for each predicate symbol p and variables $X_1, \ldots, X_n$. For those queries, SETHEO searches for all solutions within a low bound δ_{bu} (here, normally, a depth-bound (A-literal depth) of 2 is used, but it can be varied freely). For each obtained substitution σ, we generate the unit clause "$p(\sigma X_1 \ldots, \sigma X_n)$". Up to now SETHEO only uses locally effective subsumption techniques, based on SETHEO's built-in constraint mechanism. Although, in case of a depth bound of 2, anti-lemma constraints[2] simulate forward

* Submitted to CADE 12

[2] Anti-lemma constraints (for details see [LMG93]) are syntactic constraints which are

subsumption, all newly generated unit clauses must pass an additional subsumption test. Then, the remaining ones are added to the current formula, and the iteration step is executed again, unless until a stop condition (see below) has been reached. The original formula plus the unit clauses produced during the last iteration are used for the final top-down search.

Adding unit clauses to a formula does not affect completeness of the top-down proof procedure. Therefore, several ways of controlling the generation of unit clauses during the preprocessing phase are provided: the generation of new clauses can be restricted to specific predicate symbols, and unit clauses with an excessive size or complexity of terms can be filtered out. Furthermore, we can limit the number of generated unit clauses to a suitable value (usually around 100).

In the case of a Horn-formula, a proof can always be found in the final top-down step with a lower bound than needed for a pure top-down proof (assuming, however, all bottom-up unit clauses have been generated).

For non-Horn formulae, we use the same approach of bottom-up iteration. However, we generate unit clauses for p (as above) and for $\neg p$. Furthermore, we allow Model Elimination reduction steps to occur during the preprocessing phase. This increases the power of the bottom-up resolution step by introducing factorization. However, in the non-Horn case, we cannot assure that the resources needed in the final step are lower compared to those needed without DELTA. (There exists always the possibility that a leaf node in the tableau has to be closed by a reduction step almost up to the root.) Experiments showed that even with non-Horn formulae, in many cases, a gain in efficiency could be obtained, often because, there are only few reduction steps in a proof.

The following table shows the results of first experiments with well-known benchmark examples. We have used a simple prototype version of DELTA and SETHEO V3.1 as the top-down prover. This prototype has been implemented in UNIX shell, using awk [AKW88] scripts for manipulating the formula and filtering the unit clauses. As a resource bound for the preprocessing and the final top-down search, the depth of the proof tree (A-literal depth) has been used. DELTA has always been started with default parameters, except in cases where the number of newly generated clauses grew too rapidly. In that case, the maximal term complexity has been restricted to 2 († in the table). If, for Non-Horn formulae, too few unit clause have been generated, the depth bound for each iteration has been increased (‡).

As a comparison, run-times for SETHEO without bottom-up preprocessing are shown. As in the final top-down search, SETHEO has been started with its default parameters (iterative deepening with A-literal depth, and generation and usage of constraints). All run-times shown in this table are in seconds and have been obtained on a sun sparc II. They include full compilation and assembly times (for each iteration), not just the run-times of the prover itself. However, a more efficient implementation of DELTA could further substantially reduce the preprocessing time.

---

generated during the search to prevent solving a subgoal more than once with an identical substitution, as long as it remains in the tableau.

Example	iteration level	gener. clauses	run-time DELTA	run-time top-down	run-time total	run-time SETHEO *only*
wos1	1	36	0.87	3.59	4.46	**1.53**
wos4	1	93	1.79	23.35	25.14	**22.93**
wos15(H)[3]	2†	256	15.58	61.82	**77.40**	807.61
wos17(H)	2†	328	8.72	9.01	**17.73**	11081.93
wos20	1†	507	3.05	25.79	**28.84**	—
wos21(H)	2†	144	3.94	20.03	**34.17**	—
wos31	4‡	114	17.2	3.65	**20.85**	—
wos33	4	28	12.97	5.97	**18.94**	—
sam(H)	3	66	12.97	5.97	**18.94**	—
LS36(H)	1	148	1.35	45.28	**46.63**	87.25
LS37a(H)	3†	257	21.67	7.52	**29.19**	—
Bledsoe-1	2	67	1.96	4.40	**6.63**	6.67
Bledsoe-2	2	68	1.90	6.87	**8.77**	114.33

The results shown are quite remarkable, compared to pure top-down theorem proving, despite the straight-forward approach and primitive prototype of DELTA. The combined system could even solve several examples for which SETHEO alone could not find a proof.

This approach of combining top-down and bottom-up theorem proving is extremely flexible and thus allows for many enhancements and future developments, leading to a dramatic increase in the power of automated theorem proving.

The prototype version of DELTA together with a manual page is available via ftp. For information, please contact the author.

References

[AKW88] A. Aho, B. Kernighan, and P. Weinberger. *The AWK Programming Language*. Eddison Wesley, 19988.

[AL91] O.L. Astrachan and D.W. Loveland. METEORs: High Performance Theorem Provers using Model Elimination. In R.S. Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*. Kluwer Academic Publishers, 1991.

[LMG93] R. Letz, K. Mayr, and C. Goller. Controlled Integrations of the Cut Rule into Connection Tableau Calculi. Technical report, Technische Universität München, 1993. submitted to JAR.

[LSBB92] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

[McC90] W. McCune. *Otter 2.0 Users Guide*. National Technical Information Service, U.S. Department of Commerce, Springfield, VA, 1990.

[Sti88] M. E. Stickel. A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler. *Journal of Automated Reasoning*, 4:353–380, 1988.

[3] Horn formulae are marked by a "(H)".