Termptation

Cristina Borralleras[1] and Albert Rubio[2]

[1] Universitat de Vic, Spain
Email: cristina.borralleras@uvic.es
[2] Universitat Politècnica de Catalunya, Barcelona, SPAIN
Email: rubio@lsi.upc.es

TERMPTATION (TERMination Proof Techniques automATION) is a fully automated system for proving termination of term rewrite systems (TRS) which is based on the *monotonic semantic path ordering* method (MSPO) [BFR00] and its translation to an ordering constraint solving problem [BR03], called the MSPO-constraint method (see [Bor03] for a detailed description). It is available at `//www.lsi.upc.es/~albert/`.

The system has been implemented in Prolog and has three main steps. First it tries to ease the termination proof of the TRS by applying modularity results: disjoint unions, constructor-sharing unions and, in some cases, hierarchical unions of sets of rules. Once this splitting is made, for every subsystem we apply the MSPO-constraint method to obtain the a disjunction of ordering constraints. Finally the system tries to find a satisfiable constraint in the disjunction.

1 Options of the system

TERMPTATION has four parameters to be set:

– *The prover*.
The user can choose among five provers. Prover 5 is an implementation of the RPO. Provers 1 – 4 are implementations of the MSPO-constraint method which differ in

- the way the constraints are generated: by groups in provers 1 and 3, which chooses in a clever way the constraints to be tried first, and one by one in provers 2 and 4;
- whether modularity for hierarchical unions are applied or not. Provers 1 and 2 do not apply such modularity results while provers 3 and 4 do. Note that, if this modularity reuslt are applied then only a subset of the constraints generated by the MSPO-constraint method can be used.

In general, provers 2 and 4 work better for TRS's with large size rules (since in that case generating the constraints by groups can be very expensive in time and space), and provers 3 and 4 are recomended for TRS's with a large number of rules. The default option is prover 1 which treats the constraints by groups and it does not apply modularity result for hierarchical unions.

– *Standard or innermost rewriting*.
The system can be used for proving termination of rewriting or termination of innermost rewriting. Note that for innermost rewriting more powerful constraint solving techniques can be applied.

– *Special treatment of arguments*.
  By default the system applies the definition of the MSPO using the semantic path ordering (SPO) with a multiset comparison of arguments. If a special treatment of arguments is chosen then the system uses the lexicographic semantic path ordering (SPO-lex) with a lexicographic comparison of the arguments.
– *Deeper analysis*.
  If this option is activated then the system uses more powerful techniques to analyze each constraint that is considered. Therefore, constraints that are considered unsolvable without the deeper analysis can become solvable if this option is activated. Due to this, if the system fails in both cases then, of course, the system with the deeper analysis takes more time to produce the answer.
  On the other hand, if the system succeeds in both cases, since they may found different solutions, it is not clear which one will be faster.

2 The Output

The output provides part of the explanation about the proof of termination found by the system.

By now, except if we have used prover 5 (which is only RPO), in many cases part of the output can only be followed by users knowing about MSPO and its translation to constraints and with a deep knowledge about the applied constraint handling methods (some of them are similar to the ones used by the dependency pair method [AG00]). The provided information is intended to allow expert users to reconstruct the termination proof by hand.

3 Some implementation details

Our method basically tries to prove the satisfiability of one of the constaints generated. Since there can be many of those constraints, the system tries to consider first the ones that look simpler to be solved.

The techniques that are used for solving the obtained constraints are based on the analysis of the dependency graph [AG00] and some original techniques to check the potential cycles. After analayzing the graph and its potential cycles, the resulting ordering constraints are finally solved using term interpretations and the recursive path ordering (RPO).

3.1 Generating term interpretations and RPO

The system considers two different kinds of interpretations, projections and selections of arguments (so called argument filtering interpretations). The set of possible interpretations is finite, but it is in general very large. Our process consists of proving or discarding possible interpretations for each symbol. If all the interpretations for a symbol are discarded then the process fails and backtracks.

Some heuristics are used to choose which symbols are considered first for deciding its interpretation. These decisions are taken be analyzing the ordering restrictions that

have to be fulfiled. After any intermediate decision in the process, the system discards all remaining interpretations that are incompatible with the constraint.

Additionally, while producing the interpretations we keep track of all forced decisions on the precedence and status of the function symbols for the RPO. For instance, if in a particular state the interpretation of $g(x)$ is either g or $g(x)$ and we have a condition $a > g(b)$ in the constraint then the precedence should include $a \succ_F g$. If a latter decision on the interpretation forces $g \succeq_F a$ in the precedence, the process fails and backtracks.

Finally, the automatic generation of the RPO is implemented following the Davis-Putnam style for solving SAT [DLL62]. In general, the system takes first those decisions for which there are less alternatives to consider. After every decision, the system applies (propagates) all forced decisions about the precedence and the status for the function symbols and simplifies the remaining set of ordering conditions with respect to such decisions.

References

[AG00] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[BFR00] C. Borralleras, M. Ferreira, and A. Rubio. Complete monotonic semantic path orderings. *Proc. of the 17th International Conference on Automated Deduction*, LNAI 1831:346–364, Pittsburgh, USA, 2000. Springer-Verlag.

[Bor03] C. Borralleras. Ordering-based methods for proving termination automatically. PhD thesis, Universitat Politècnica de Catalunya, Dept. LSI., 2003. Available at `//www.lsi.upc.es/~albert/cristinaphd.ps`.

[BR03] C. Borralleras and A. Rubio. Proving Termination by the Reduction Constraint Framework. *Proc. of the 6th International Workshop on Termination*, Valencia, Spain, 2003.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. In *Communications of the ACM*, 5(7):394–397, July 1962.