Connecting remote termination tools*

María Alpuente and Salvador Lucas

DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
{alpuente,slucas}@dsic.upv.es

Abstract. After more than thirty years of development of the theory of termination of rewriting, a number of tools for automatically proving termination of TRSs have recently emerged. Their application to real programming languages and systems is of course desirable and has been envisaged in many of these tools. The World Wide Web makes it possible to gain access to the different resources in a number of ways, ranging from remote downloads followed by local executions to remote execution via WWW services. Unfortunately, though, few of the existing systems and tools (including termination tools) are readily connectable. We advocate systematically considering interoperability across platforms, applications and programming languages when developing termination tools and related language processors. This is only possible if a number of common practices (and standards) are seriously put into use by the international community.

1 Introduction

The quest for a *verifying compiler* is a classic but still hot, challenging goal for both the Software Industry and the Computer Science community [4]. Of course, termination analysis would be an essential component of such a tool. As mentioned in [4], the effective development of such a system will require an incremental and cooperative effort from different work teams all around the world. Thanks to the WWW, the physical distance among those teams is becoming less and less important. However, many existing systems and tools are not easily conciliable for working together, even if they address closely related problems or rely on similar theoretical bases. The WWW technology, however, is full of possibilities for removing barriers to integration and for dramatically improving the re-usability of previous theoretical results and development efforts.

2 Termination of programs and termination tools

As a motivating example, we consider the termination analysis of programs written in programming languages such as CafeOBJ, Elan, Erlang, Maude, OBJ, etc., whose operational principle is based on term rewriting. Proofs of termination of TRSs can be used for proving the termination of programs written in these languages. A number of systems for proving termination are available on the WWW: e.g.,

Tool	WWW site (use prefix `http://`)
AProVE	`www-i2.informatik.rwth-aachen.de/AProVE`
CiME	`cime.lri.fr`
Hasta-La-Vista	`www.cs.kuleuven.ac.be/~dtai/dpl/systems-E.shtml`
Matchbox	`theo1.informatik.uni-leipzig.de/matchbox`
TALP	`bibiserv.techfak.uni-bielefeld.de/talp`
TerminWeb	`lvs.cs.bgu.ac.il/~mcodish/suexec/terminweb`
Termptation	`www.lsi.upc.es/~albert/term.tar.gz`
Tyrolean/Tsukuba Termination Tool	`cl2-informatik.uibk.ac.at`

* Work partially supported by MCyT project TIC2001-2705-C03-01, MCyT Acción Integrada HU 2003-0003 and AVCyT grant GR03/025.

Other systems, like the Mercury compiler `www.cs.mu.oz.au/research/mercury`, include a termination checker.

Unfortunately, however, it is not easy to connect independently developed analysis tools to a practical environment such as the Maude interpreter[1]: the syntax of Maude programs does not correspond to the syntax of any of existing termination tools (see [5] for descriptions of many of them); the systems have quite different interfaces (for instance, stream-based input output or graphical interfaces); the Maude interpreter is written in C++, whereas the tools are written in different languages, namely CAML, Java, SICSTUS Prolog, etc.

From a semantic point of view, there are also other relevant issues which have to be considered when trying to bridge term rewriting tools and Maude: Maude programs are syntactically richer than term rewriting systems. Maude features that are not necessarily managed by these rewriting tools include: sorts, conditional rules, pattern matching expressions, programmable evaluation strategies, associative/commutative/idempotent symbols, modules, etc. The previous tools for proving termination of rewriting hardly deal with these features, which still have to be properly managed if we want to (accurately) use the existing tools to prove properties of real programs. Eventually, this can be done by means of program transformations which preserve the focused property (e.g., termination). This is somehow related to the transformational approaches for proving termination of (well-moded) logic programs which reduce the termination problem of logic programs to that of TRSs. These findings are the basis for logic programming termination tools consisting of a front-end which implements the considered transformation and a back-end for proving termination of the generated TRS.

3 Web services for termination analysis

Interoperability (i.e., to make it possible for *a program on one system to access programs and data on another system* [1]) is a general problem in Software Engineering and a number of solutions have been devised up to now (namely, *middleware systems* [1,6]). The XML WWW services (or just WWW services) are the most recent proposal for interconecting systems [7]. They provide a flexible architecture for achieving interoperability of loosely-coupled systems that use different internal data structures to represent the information, and that have been developed in different programming languages. The exchange of information, the implementation of the necessary calls, and the localization of services on the web is based on different XML dialects (XML for representing data, SOAP for packaging messages, WSDL for defining the available services, and UDDI for finding them on the Web [3]). Even in the case that the systems are available on the same local machine, most of these interoperability facilities could play a relevant role in gluing together different tools which are often developed in different programming languages as well as combining architectural approaches where more tightly-coupled techniques like RPC or CORBA are not viable in many cases. Innovation is moving towards the so-called WWW services choreography, which pursues new standards to support the specification of complex systems out of simpler ones.

The definition of standards is widely recognized as desirable in the industry. The scientific community would also benefit from some standarization regarding, e.g., interoperability of systems which are naturally and highly distributed. We do believe that more general frameworks are lacking, that is, abstract schemes for particular kinds of applications -abstract in the sense of isolating the common features of that class. In many framework

[1] `http://maude.cs.uiuc.edu`

designs, the applications are created by the combination of the framework with a number of application-dependent decisions so that they get permanently -and undesiderably- linked to the framework. Tackling the following specific issues seems desirable:

1. Defining suitable XML-like formats for expressing the most important features of currently used (families of) programming languages (in particular, rewriting–based programming languages).
2. XML sublanguages for expressing analysis/verification requirements to existing tools (e.g., innermost termination, simple termination, etc., possibly considering information regarding special operators, types, modules, strategies, conditions, etc.). We believe that this approach deserves great interest since it does contribute to the design of abstractions which are completely independent from the target language and are highly reusable.
3. Middleware translators from the existing programming languages to the lower level languages or formalisms which underly the program analysis tools. XSLT could be used for that purpose. Also declarative languages such as CiaoProlog, Haskell, SICStus Prolog, SWI-Prolog, etc., are well-suited for that.
4. Including the termination tools into the SOAP/WSDL/UDDI framework to get systematic access to their functionality.

As a concrete starting point, we think that the current format for specifying termination problems in the WST 2004 competition

`http://www.lri.fr/~marche/wst2004-competition`

could be further developed in the following directions:

1. separate the purely syntactic components of the problem (i.e., those referring the structure of the TRS) from the proof-oriented ones (i.e., those concerning a kind of termination property or termination proof which could be required from a tool);
2. use XML to express these different aspects of the problem; and
3. making some concrete, preliminary experiences to connect 'server' tools (e.g., termination provers) and 'client' tools (e.g., compilers, interpreters, or other systems which use auxiliary termination tools).

Beyond being the current trend of the international community, we believe that the use of XML-based technology for these purposes is also positive due to its flexibility and the existence of many tools and libraries which can be reused. Of course, other approaches, such as the ASF+SDF technology [2] could also provide useful ideas and solutions.

These tasks could probably be better addressed in coordination with existing working groups in international associations: for instance,

W3 Consortium	`http://www.w3c.org`
IFIP	`http://www.ifip.org`
ERCIM	`http://www.ercim.org`

4 Conclusion

The recent development of new and powerful termination tools is encouraging for the scientific community and suggests that they could be used by more complex computational systems such as advanced compilers and integrated software development environments.

However, the current tools are not commonly used or even easily reusable. The use of remote analysis tools (where remote means not only distant but also difficult to connect) is essential to be able to integrate powerful program analysis into programming environments. Not only should correctness or efficiency be taken into account while designing new tools, environments and frameworks, but interoperability issues should also be systematically considered as well. We honestly believe that putting greater effort into these issues will contribute to being successful with the current challenge of connecting analysis tools. In our opinion, these tasks should be appropriately coordinated by existing working groups in international associations.

References

1. P.A. Bernstein. Middelware: A Model for Distributed System Services. *Comm. of the ACM*, 39(2):86-98, 2002.
2. M.G.J. van den Brand, A. van Deursen, J. Heering, H.A. de Jong, M. de Jonge, T. Kuipers, P. Klint, L. Moonen, P.A. Olivier, J. Scheerder, J.J. Vinju, E. Visser, and J. Visser The ASF+SDF Meta-Environment: A component-based language development environment In R. Wilhelm, editor, *Proc. of 10th International Conference on Compiler Construction CC'01*, LNCS 2027:365-370, Springer-Verlag, Berlin, 2001.
3. M. Burner. The Deliberate Revolution. Creating Connectedness with XML Web Services. *ACM Queue* 1(1):29-37, March 2003.
4. T. Hoare. The Verifying Compiler: A Grand Challenge for Computing Research. *Journal of the ACM*, 50(1):63-69, 2003.
5. A. Rubio, editor, *Proc. of 6th Int'l Workshop on Termination, WST'03*, Tech. Report DSIC II/15/03, Valencia, Spain, 2003.
6. I. Sommerville. Software Engineering (6th Edition). Addison Wesley, 2001.
7. M. Stal. Web services: Beyond Component-Based Computing. *Comm. of the ACM*, 45(10):71-76, 2002.