Termination Competition (termCOMP 2015)

Jürgen Giesl[1⋆], Frédéric Mesnard[2], Albert Rubio[3⋆⋆], René Thiemann[4⋆⋆⋆], and Johannes Waldmann[5]

[1] RWTH Aachen University, Germany
[2] Université de la Réunion, France
[3] Universitat Politècnica de Catalunya - BarcelonaTech, Spain
[4] Universität Innsbruck, Austria
[5] HTWK Leipzig, Germany

Abstract. The termination competition focuses on automated termination analysis for all kinds of programming paradigms, including categories for term rewriting, imperative programming, logic programming, and functional programming. Moreover, the competition also features categories for automated complexity analysis. In all categories, the competition also welcomes the participation of tools providing certified proofs. The goal of the termination competition is to demonstrate the power of the leading tools in each of these areas.

1 Introduction

The termination competition has been organized annually since 2004. There are usually between 10 and 20 participating termination/complexity/certification tools. Recent competitions were executed live during the main conferences of the field (at VSL 2014, RDP 2013, IJCAR 2012, RTA 2011, and FLoC 2010).

2 Competition Categories

2.1 Termination Analysis

The termination competition features numerous categories for different forms of languages. These languages can be classified into real programming languages (Sect. 2.1.2) and languages based on rewriting and/or transition systems (Sect. 2.1.1). Termination of such languages is also of great practical interest, because they are often used as *back-end* languages. More precisely, one can prove termination of programs by first translating them into such a back-end language automatically and by analyzing termination of the resulting rewrite or transition system afterwards.

⋆ This author is supported by the Deutsche Forschungsgemeinschaft (DFG) grant GI 274/6-1.
⋆⋆ This author is supported by the Spanish MINECO under the grant TIN2013-45732-C4-3-P (project DAMAS).
⋆⋆⋆ This author is supported by the Austrian Science Fund (FWF) project Y757.

2.1.1 Rewriting and Transition Systems

There are several categories for termination analysis of different variants of term rewriting. This includes classical term rewriting, conditional term rewriting, term rewriting under specific strategies (innermost rewriting, outermost rewriting, and context-sensitive rewriting), string rewriting (where all function symbols are unary), relative term or string rewriting (where one has to prove that certain rules cannot be used infinitely often), term rewriting modulo equations, and higher-order rewriting.

In 2014, the competition also had categories for systems with built-in integers for the first time. More precisely, there was a category for term rewriting extended by integers and a category for integer transition systems (which do not feature terms, and where one has to prove absence of infinite runs originating in designated start states).

2.1.2 Programming Languages

The termination competition has categories for termination of programs in several languages from different paradigms. This includes functional languages (Haskell), logic languages (Prolog), and imperative languages. While a category for termination of Java programs has already been part of the competition since 2009, since 2014 there is also a category for the analysis of C programs.

2.2 Complexity Analysis

Since 2008, the termination competition has categories for asymptotic worst-case complexity analysis of term rewriting. Here, one tries to find an upper bound on the function that maps any natural number n to the length of the longest possible derivation starting with a term of size n or less. In the competition, different forms of complexity are investigated, depending on whether one regards full or innermost rewriting. Moreover, these forms of complexity differ in the shape of the possible start terms. For *derivational complexity*, one allows arbitrary start terms. In contrast, for *runtime complexity*, one only allows start terms of the form $f(t_1, \ldots, t_n)$, where a defined symbol f (i.e., an "algorithm") is applied to "data objects" $t_1, \ldots, t_n$ (i.e., the terms t_i may not contain any defined symbols). So runtime complexity corresponds to the notion of complexity typically used for programs.

2.3 Certified Categories

It regularly occurred during previous competitions that bugs of tools have been detected by conflicting answers. However, even if there are no conflicting answers there is the potential of wrong answers. To this end, the termination competition provides a certification option. If enabled, tools must generate their proofs in a machine-readable and fully specified *certification problem format*. These proofs will then be validated by certifiers whose soundness has to be justified, e.g., by a machine-checked soundness proof of the certifier itself, or via on-the-fly generation of proof scripts for proof-assistants like Coq, Isabelle, or PVS.

The certification option is currently supported for most categories on first-order term rewriting, for both termination and complexity analysis.

3 Termination Problem Data Base

The *Termination Problem Data Base* (TPDB) is the collection of all the examples used in the competition. Its structure is closely related to the categories in the competition. Each example in the TPDB is sufficiently specified to precisely determine a Boolean answer for termination (or an optimal answer for complexity). For instance, although we aim to detect duplicates and eliminate them from the TPDB (modulo renaming and order of rewrite rules), the data base may contain two examples with the same program which differ in their evaluation strategy or in the set of start terms. These details are important in the competition, where the tools are asked to investigate the termination and complexity behavior for exactly the given evaluation strategy and initial terms. Although there is a unique correct answer for each example, these answers are not stored in the TPDB and might even be unknown. For instance, the TPDB also contains Collatz' open termination problem of the "$3n + 1$" function.

New problems for the TPDB can be submitted at any time and will be added after a short reviewing process of the steering committee. This steering committee consists of representatives of the participating research groups. It is in charge of strategic decisions for the competition and its future. Currently, the examples in the TPDB are distributed as follows w.r.t. their source languages: term rewriting (10755), Haskell (1671), integer transition systems (953), Java (859), Prolog (492), and C (480).

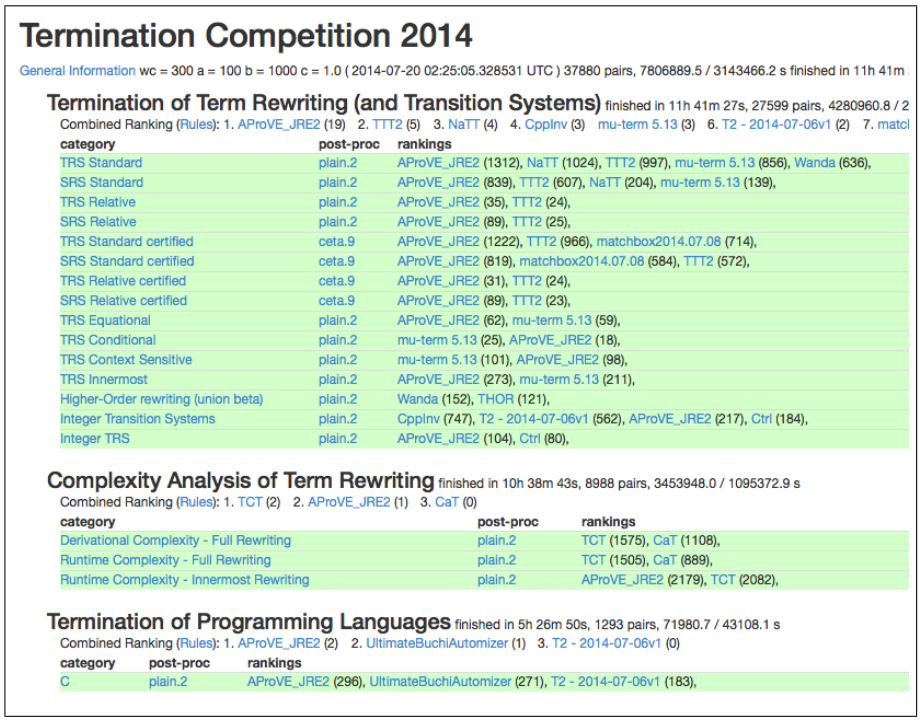
4 Running The Competition

Here is a brief description of the rules of the competition:

- For termination tools: given an input program from the TPDB, try to determine whether it terminates or not within a given time limit. Positive and negative answers are equally scored when determining the winner of a category.
- For complexity tools: try to figure out the worst-case complexity of an input program from the TPDB within a given time limit in big-O notation. Here, the scoring depends on the precision of the answer in comparison to the answers of the other competing tools.
- For certifiers: try to check as many machine-readable termination/complexity proofs as possible.

Both termination and complexity tools must provide a human- or machine-readable proof in addition to their answer. The input problems and tools are partitioned w.r.t. the categories presented in Sect. 2. A category is only scheduled in the competition if there are at least two participating tools for that category. Other categories may be scheduled for demonstration purposes.

From 2004 to 2007, the competition was hosted by the University of Paris-Sud, France. From 2008 to 2013, the competition was hosted by the University of Innsbruck, Austria. In 2014, the competition was run for the first time on the StarExec platform (`https://www.starexec.org/`) at the University of Iowa, USA, while results were presented on the web front-end star-exec-presenter (see Fig. 1) running at HTWK Leipzig, Germany. The same infrastructure will be used for the 2015 competition.



Termination Competition 2014

General Information wc = 300 a = 100 b = 1000 c = 1.0 (2014-07-20 02:25:05.328531 UTC) 37880 pairs, 7806889.5 / 3143466.2 s finished in 11h 41m

Termination of Term Rewriting (and Transition Systems) finished in 11h 41m 27s, 27599 pairs, 4280960.8 / 2

Combined Ranking (Rules): 1. AProVE_JRE2 (19) 2. TTT2 (5) 3. NaTT (4) 4. CppInv (3) mu-term 5.13 (3) 6. T2 - 2014-07-06v1 (2) 7. matcl

category	post-proc	rankings
TRS Standard	plain.2	AProVE_JRE2 (1312), NaTT (1024), TTT2 (997), mu-term 5.13 (856), Wanda (636),
SRS Standard	plain.2	AProVE_JRE2 (839), TTT2 (607), NaTT (204), mu-term 5.13 (139),
TRS Relative	plain.2	AProVE_JRE2 (35), TTT2 (24),
SRS Relative	plain.2	AProVE_JRE2 (89), TTT2 (25),
TRS Standard certified	ceta.9	AProVE_JRE2 (1222), TTT2 (966), matchbox2014.07.08 (714),
SRS Standard certified	ceta.9	AProVE_JRE2 (819), matchbox2014.07.08 (584), TTT2 (572),
TRS Relative certified	ceta.9	AProVE_JRE2 (31), TTT2 (24),
SRS Relative certified	ceta.9	AProVE_JRE2 (89), TTT2 (23),
TRS Equational	plain.2	AProVE_JRE2 (62), mu-term 5.13 (59),
TRS Conditional	plain.2	mu-term 5.13 (25), AProVE_JRE2 (18),
TRS Context Sensitive	plain.2	mu-term 5.13 (101), AProVE_JRE2 (98),
TRS Innermost	plain.2	AProVE_JRE2 (273), mu-term 5.13 (211),
Higher-Order rewriting (union beta)	plain.2	Wanda (152), THOR (121),
Integer Transition Systems	plain.2	CppInv (747), T2 - 2014-07-06v1 (562), AProVE_JRE2 (217), Ctrl (184),
Integer TRS	plain.2	AProVE_JRE2 (104), Ctrl (80),

Complexity Analysis of Term Rewriting finished in 10h 38m 43s, 8988 pairs, 3453948.0 / 1095372.9 s

Combined Ranking (Rules): 1. TCT (2) 2. AProVE_JRE2 (1) 3. CaT (0)

category	post-proc	rankings
Derivational Complexity - Full Rewriting	plain.2	TCT (1575), CaT (1108),
Runtime Complexity - Full Rewriting	plain.2	TCT (1505), CaT (889),
Runtime Complexity - Innermost Rewriting	plain.2	AProVE_JRE2 (2179), TCT (2082),

Termination of Programming Languages finished in 5h 26m 50s, 1293 pairs, 71980.7 / 43108.1 s

Combined Ranking (Rules): 1. AProVE_JRE2 (2) 2. UltimateBuchiAutomizer (1) 3. T2 - 2014-07-06v1 (0)

category	post-proc	rankings
C	plain.2	AProVE_JRE2 (296), UltimateBuchiAutomizer (271), T2 - 2014-07-06v1 (183),

Fig. 1. The web front-end star-exec-presenter summarizing the 2014 competition

In order to run the competition within the duration of a conference, in the last years only a subset of termination problems from the TPDB was selected for each competition. Separate "full runs" of all tools on all TPDB problems were also executed, which took around a week. In 2014, StarExec provided enough computing power to execute a full run in 12 hours. Time-out per problem was 5 minutes. A total of 377880 problem/tool pairs were executed using $8 \cdot 10^6$ seconds (2200 hours) CPU time and running on almost 200 nodes in parallel.

For further details, we refer to the main web site of the termination competition (`http://termination-portal.org/wiki/Termination_Competition`).