Termination of Term Rewriting:
Foundation, Formalization, Implementation, and Competition

Akihisa Yamada
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract
Automated termination analysis is a central topic in the research of term rewriting. In this talk, I will first review the theoretical foundation of termination of term rewriting, leading to the recently established tuple interpretation method. Then I will present an Isabelle/HOL formalization of the theory. Although the formalization is based on the existing library IsaFoR (Isabelle Formalization of Rewriting), the present work takes another approach of representing relations (predicates rather than sets) so that the notation is more human friendly. Then I will present a unified implementation of the termination analysis techniques via SMT encoding, leading to the termination prover NaTT. Many tools have been developed for termination analysis and have been competing annually in termCOMP (Termination Competition) for two decades. At the end of the talk, I will share my experience in organizing termCOMP in the last five years.

2012 ACM Subject Classification Theory of computation → Rewrite systems; Theory of computation → Automated reasoning

Keywords and phrases Term rewriting, Termination, Isabelle/HOL, Competition

1 Foundation

Ensuring the termination [10] of term rewrite systems (TRSs) has many important applications and actively studied for a half century. A foundational way of ensuring termination used to be reduction orders, in short, overestimating the rewrite step by a term ordering whose termination (well-foundedness) is known. Later, the dependency pair method [1] generalized reduction orders to reduction pairs [1, 17, 15], where, in short, one part overestimates the rewrite step and the other part ensures termination.

The semantic method (interpretation method) [29] for defining reduction orders/pairs uses universal algebras, mapping terms into a set with a known terminating relation, such as \( (\mathbb{N}, >) \). Instances of semantic approaches are polynomial interpretations [27], matrix interpretations [18, 11], max-polynomial interpretations [12], and arctic interpretations [24]. Tuple interpretations [43, 23] encompass these instances, and the use of derivers [41, 28] allows to uniformly prove monotonicity (essential for overestimating rewrite steps) [43].

The syntactic approach such as recursive path ordering [9] and lexicographic path ordering [21] analyzes the term structure purely syntactically, and usually ensure termination by Kruskal’s tree theorem [26]. The Knuth-Bendix ordering [22] mixes the semantic and syntactic approaches. The weighted path order (WPO) [45] gives a unified way of capturing syntactic and semantic methods.
2 Formalization

There are numerous methods for ensuring termination, and they are implemented in automated termination analysis tools such as AProVE [13], TTT2 [25], matchbox [40], MU-TERM [16], etc. However, it cannot be taken for granted that the implementations and the paper proofs are correct. Therefore, there have been efforts in formally verifying the paper proofs and tool outputs using proof assistants such as Coq [4] and Isabelle/HOL [30].

Coccinelle/CiME3 [8] and CoLoR/Rainbow [5] are pairs whose first components are Coq libraries of proofs for termination methods, and the second components are tools that translate termination tool's outputs into Coq theories whose validity are then checked by Coq. IsaFoR/CeTA [38] is another pair, whose first component is an Isabelle/HOL library of termination methods (and more), and the second is a tool that checks the tool outputs. The correctness of the latter is verified by Isabelle; therefore achieves the same level of assurance as the Coq ones without needing to execute the proof assistants in the run time. In this talk I will present a recent development based on IsaFoR for sorted term rewriting and algebras, which formalizes the correctness of the deriver-based tuple interpretation method.

3 Implementation

NaTT started as an OCaml [31] code for experimenting WPO. This implementation follows the “lazy” approach [47, 32, 48, 7] and delegates “solving” tasks to SMT solvers [3]. The code was turned into a full termination prover by adding other basic techniques and improving efficiency by being more lazy in calling SMT solvers [44]: for instance, in the SMT interface of NaTT, the OCaml code
\[
x *^\text{Delay} (\text{fun } _ \rightarrow \text{hardFormula})
\]
represents an SMT formula for \(x\) multiplied by \(\text{hardFormula}\), but \(\text{hardFormula}\) will not be evaluated if \(x = 0\) is known. NaTT participates in the Termination Competition (since “full run” of 2013) and keeps winning the second place in the TRS Standard category after the champion, AProVE. Since then more techniques are implemented into NaTT [19, 46, 2, 35, 43, 42], making the implementation harder and harder to maintain. In the meantime, the C++ programming language kept evolving, and hence I fully reimplement NaTT in the modern C++20 [20]. For instance, in the new C++ implementation, the above lazy formula is achieved by
\[
x * \{ \text{return hardFormula; } \}.
\]

4 Competition

At the International Workshop on Termination (WST) 2003, Albert Rubio organized a session where termination tool developers demonstrate their tools on problems written on a blackboard. The community decided to turn this event into the International Termination Competition (termCOMP) [14]. Benchmarks are collected, using an agreed textual format called the WST format [6], into the Termination Problem DataBase (TPDB) [39]. Editions from 2004 to 2007 were run on local servers organized by Claude Marché, and from 2008 to 2013 are by René Thiemann. In 2009, the benchmark format has been changed to an XML format (see [39]). From 2014 to 2017 are organized by Johannes Waldmann, who migrated the tool execution platform to the StarExec environment [36], and wrote the web server code, star-exec-presenter [33], using Haskell and SQL. In 2018 I succeeded the organization and developed starexec-master [34] web frontend in pure PHP (so that most web servers and engines can understand). Since then I have gradually improve visuals and collected metrics (Figure 1 on page 3), keep past results in a public repository [37], and migrated TPDB also into GitHub. In the talk I would like to share some lessons learned by organizing the competition.
Figure 1 An excerpt of results of termCOMP 2022. Rows with check mark indicates the certified configurations. The “news” scores indicate improvements over the virtual best solver of the past, that is, the number of open problems closed by the solver this year. Those scores were used to award the special “advancing-the-state-of-the-art” medals.

References

4:4 Termination of Term Rewriting: Foundation, Formalization, Implementation, ...


31 Ocam. URL: https://v2.ocaml.org/.

star-exec-presenter (software), 2014. URL: https://github.com/jwaldmann/star-exec-presenter/.


The termination problem data base. URL: http://termination-portal.org/wiki/TPDB.


