Tsukuba Termination Tool*

Nao Hirokawa and Aart Middeldorp

University of Tsukuba, Tsukuba 305-8573, Japan

We present a tool for automatically proving termination of first-order rewrite systems. The tool is based on the dependency pair method of Arts and Giesl [1]. It incorporates several new ideas that make the method more efficient. The tool produces high-quality output and has a convenient web interface. If T$_{\text{T}}$T succeeds in proving termination, it outputs a proof script which explains in considerable detail how termination was proved. This script is available in both HTML and LaTeX format. In the latter, the approximated dependency graph is visualized using the *dot* tool of the Graphviz toolkit. T$_{\text{T}}$T is written in Objective Caml. We tested the various options of T$_{\text{T}}$T on numerous examples. The results, as well as a comparison with other tools that implement the dependency pair method and some implementation details, can be found in [2, 3].

We describe some of the features of the tool (T$_{\text{T}}$T in the sequel) by means of its web interface, displayed in Fig. 1.

TRS The user inputs a TRS by typing the rules into the upper right text area or by uploading a file via the browse button. The exact input format is obtained by clicking the TRS link.

Comment and Bibtex Anything typed into the upper right text area will appear as a footnote in the generated LaTeX code. This is useful to identify TRSs. LaTeX `\cite` commands may be included. In order for this to work correctly, a corresponding bibtex entry should be supplied. This can be done by typing the entry into the appropriate text area or by uploading an appropriate bibtex file via the browse button.

Base Order The current version of T$_{\text{T}}$T supports the following three base orders: LPO with strict precedence, LPO with quasi-precedence, and KBO with strict precedence.

Dependency Pairs T$_{\text{T}}$T supports the basic features of the dependency pair technique (argument filtering, dependency graph, cycle analysis) described below. Advanced features like narrowing, rewriting, and instantiation are not yet available. Also innermost termination analysis is not yet implemented.

* `http://www.score.is.tsukuba.ac.jp/ttt/`

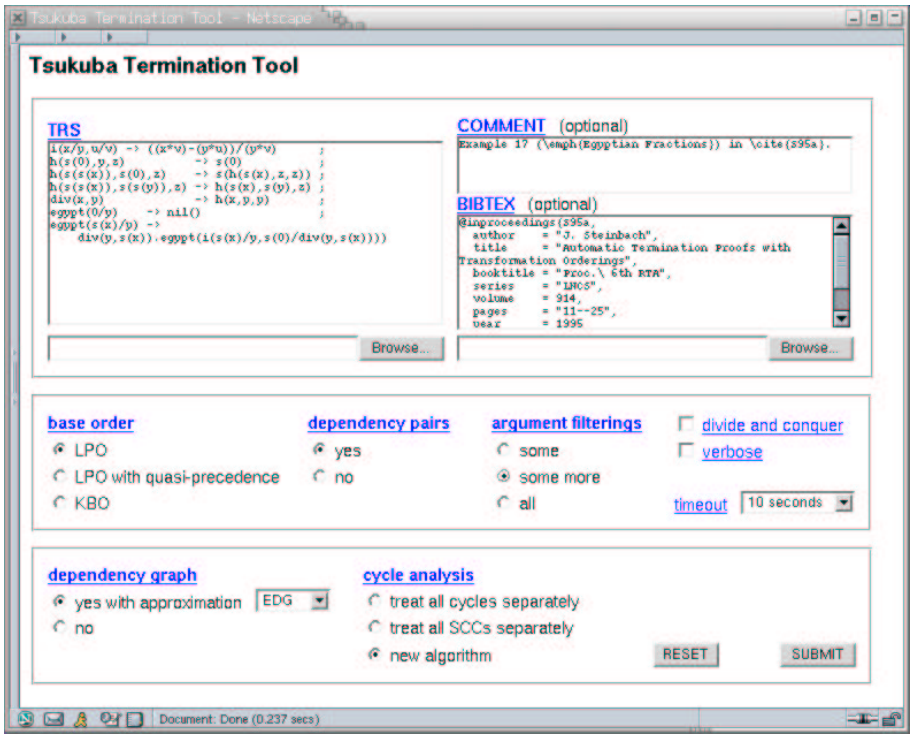

Fig. 1. A screen shot of the web interface of TTT.

Argument Filtering A single function symbol f of arity n gives rise to $2^n + n$ different argument filterings (AFs): $f(x_1, \ldots, x_n) \to f(x_{i_1}, \ldots, x_{i_m})$ for all $1 \leqslant i_1 < \cdots < i_m \leqslant n$ and $f(x_1, \ldots, x_n) \to x_i$ for all $1 \leqslant i \leqslant n$. A moment's thought reveals that even for relatively small signatures, the number of possible AFs is huge. TTT supports two simple heuristics to reduce this number. The *some* option considers for a function symbol f of arity n only the 'full' AF $f(x_1, \ldots, x_n) \to f(x_1, \ldots, x_n)$ and the n 'collapsing' AFs $f(x_1, \ldots, x_n) \to x_i$ $(1 \leqslant i \leqslant n)$. The *some more* option considers in addition the argument filtering $f(x_1, \ldots, x_n) \to f$ (when $n > 0$).

Dependency Graph The dependency graph determines the ordering constraints that have to be solved in order to guarantee termination. Since the dependency graph is in general not computable, a decidable approximation has to be adopted. The current version of TTT supplies two such approximations: EDG is the original estimation of Arts and Giesl; EDG$*$ is an improved version of EDG described in [4, latter half of Section 6].

Cycle Analysis Once an approximation of the dependency graph has been computed, some kind of cycle analysis is required to generate the actual ordering

constraints. T$_\text{T}$T offers three different methods:

1. The method described in the literature is to treat cycles in the approximated dependency graph separately. For every cycle $\mathcal{C}$, the dependency pairs in $\mathcal{C}$ and the rewrite rules of the given TRS must be weakly decreasing and at least one dependency pair in $\mathcal{C}$ must be strictly decreasing (with respect to some AF and base order).
2. Another method is to treat all strongly connected components (SCCs) separately. For every SCC $\mathcal{S}$, the dependency pairs in $\mathcal{S}$ must be strictly decreasing and the rewrite rules of the given TRS must be weakly decreasing. Treating SCCs rather than cycles separately improves the efficiency at the expense of reduced termination proving power.
3. The third method available in T$_\text{T}$T combines the termination proving power of the cycle method with the efficiency of the SCC method. It is described in [2].

Divide and Conquer The default option to find a suitable AF that enables a group of ordering constraints to be solved by the selected base order is *enumeration*, which can be very inefficient, especially for larger TRSs where the number of suitable AFs is small. Setting the *divide and conquer* option computes all suitable AFs for each constraint separately and subsequently merges them to obtain the solutions of the full set of constraints. This can (greatly) reduce the execution time at the expense of an increased memory consumption. The divide and conquer option is described in detail in [2]. At the moment of writing it is only available in combination with LPO.

Verbose Setting the verbose option generates more proof details. In combination with the divide and conquer option described above, the total number of AFs that enable the successive ordering constraints to be solved are displayed during the termination proving process.

Timeout Every combination of options results in a finite search space for finding termination proofs. However, since it can take days to fully explore the search space, (the web version of) T$_\text{T}$T puts a strong upper bound on the permitted execution time.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *TCS*, 236:133–178, 2000.
2. N. Hirokawa and A. Middeldorp. Automating the depencency pair method. In *Proc. 19th CADE*, LNAI, 2003. To appear.
3. N. Hirokawa and A. Middeldorp. Tsukuba termination tool. In *Proc. 14th RTA*, LNCS, 2003. To appear.
4. A. Middeldorp. Approximations for strategies and termination. In *Proc. 2nd WRS*, volume 70(6) of *ENTCS*, 2002.