$\mathsf{T_TT_2}$ with Termination Templates for Teaching*

Jonas Schöpf and Christian Sternagel

University of Innsbruck
Innsbruck, Austria
`{jonas.schoepf|christian.sternagel}@uibk.ac.at`

Abstract

On the one hand, checking specific termination proofs by hand, say using a particular collection of matrix interpretations, can be an arduous and error-prone task. On the other hand, automation of such checks would save time and help to establish correctness of exam solutions, examples in lecture notes etc. To this end, we introduce a template mechanism for the termination tool $\mathsf{T_TT_2}$ that allows us to restrict parameters of certain termination methods. In the extreme, when all parameters are fixed, such restrictions result in checks for specific proofs.

Keywords and phrases teaching, termination tools, templates, proof checker

1 Introduction

Many of us are familiar with the following two (or at least similar) situations:

> *Enthusiastically we call on our favorite termination tool in order to create an example for a lecture or an exercise for an exam. But is it really necessary that the weights of this KBO are higher than four? And wouldn't it be nicer if the successor symbol was actually interpreted as the successor function in that polynomial termination proof.*

> *Hard pressed for time, you have to correct 20 term rewriting exams and each of the students seems to have chosen different matrix interpretations in Example 2. How will you manage before the deadline? Maybe you should just give each student full points.*

As a more concrete example, consider the following well-known term rewrite system (TRS) implementing addition on natural numbers:

$$0 + y \to y \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

If you ask $\mathsf{T_TT_2}$ [3] whether this TRS is terminating by KBO (`ttt2 -s 'kbo'`), you will get a `YES` with $w_0 = 1$, weights $w(\mathsf{s}) = w(0) = 1$ and $w(+) = 0$, and precedence $+ > \mathsf{s} \sim 0$.[1] In a lecture, you might want to restrict to the basic version of KBO with total precedences and moreover it might be nicer for a presentation if all function symbols had the same weight, say 1. Using our templates, this is now possible via

```
ttt2 -s 'kbo -prec "+ > s > 0" -w0 1 -weights "+ = s = 0 = 1"'
```

An instance of the other kind of example we mention above would be the question whether the following matrix interpretations prove termination of the addition TRS (we can verify by `ttt2 -s 'matrix -direct -dim 2'` that there is a similar proof, but not using exactly the same interpretations):

$$[0] = \begin{pmatrix}0\\0\end{pmatrix} \qquad [\mathsf{s}](x_0) = x_0 + \begin{pmatrix}1\\1\end{pmatrix} \qquad [+](x_0, x_1) = \begin{pmatrix}1 & 1\\0 & 1\end{pmatrix} x_0 + \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} x_1 + \begin{pmatrix}1\\0\end{pmatrix}$$

With our template mechanism you can check the given proof by

* This work is supported by the Austrian Science Fund (FWF) project P27502.
[1] Where $\sim$ is "don't care" for strict and equivalence of function symbols for quasi-precedences.

Proceedings of the 16th International Workshop on Termination;
Editor: Salvador Lucas

```
ttt2 -s 'matrix -inters "0 = 0, s = x0 + 1, + = [1,1;0,1]x0 + x1 + [1;0]"'
```

The goal of our work is to successfully handle situations like the above. More specifically, our contributions are as follows:

- Based on a simple idea (Section 2), we devised a template mechanism (Section 3) for four of the most prominent encoding-based termination methods that allows us to employ TTT2 as a "proof checker."
- Moreover, we extended TTT2's web interface (Section 4, where also ☑ is explained): on the one hand, to support our template mechanism and on the other hand, by the possibility to generate URLs that fix input TRSs and custom settings. The latter is especially useful for lectures, where it provides a fast and easy way to demonstrate examples.

2 Main Idea

Many of the termination methods that TTT2 supports are based on SAT/SMT encodings, notably the *lexicographic path order* (LPO), the *Knuth-Bendix order* (KBO), *polynomial interpretations* (PIs), and *matrix interpretations* (MIs).[2]

These methods have the following implementation detail in common: constraints on their parameters (like a precedence for LPO or KBO and the maximal value of matrix entries for MIs) are encoded into a SAT/SMT formula ϕ and then a SAT/SMT-solver is used to obtain a concrete instance. As a consequence such concrete instances of methods seem entirely random to the casual user, which is not always desirable.

For example, we might only be interested in KBO proofs where a certain symbol has least precedence, or we might want to interpret a unary function symbol s as successor function without restricting the interpretations of other symbols. In the extreme, we already have a specific proof at hand (say using specific matrix interpretations) and want to use a tool like TTT2 to verify its correctness.

In all of the above cases, an obvious solution is to modify the encoding ϕ in such a way that the desired constraints are fulfilled by every satisfying assignment. This is easily achieved by appending a formula χ that represents these constraints, resulting in $\phi' = \phi \wedge \chi$.

Of course, it would be *really* cumbersome if we had to write χ by hand. Not to mention that we would have to know implementation details concerning the encoding ϕ, for example, which arithmetic variable represents the precedence position of a given function symbol?

Instead we provide a method-specific template mechanism whose main work is to parse and translate constraints that a user provides in form of a human-readable string.

3 Templates for SAT/SMT-Based Methods

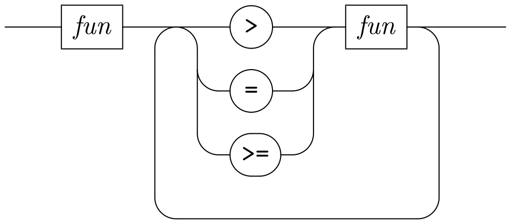
In the following we give an overview of the various templates that we provide for the four encoding-based methods from the introduction.

---

[2] There are others, like *arctic interpretations*, the *(generalized) subterm criterion*, etc. But for the moment templates are only supported by the four methods mentioned above.

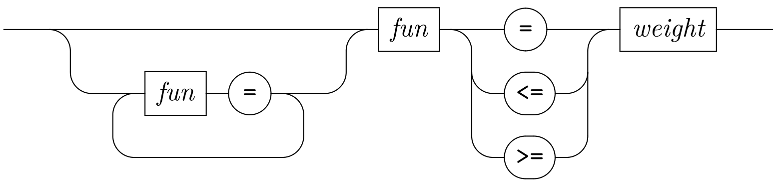
Lexicographic Path Orders The only parameter of an LPO (`ttt2 -s 'lpo'`) is its precedence. It can be specified using the flag `-prec` that takes a *prec* template as argument. A *prec* template represents a (partial) precedence by listing its constituent function symbols in decreasing order (separated by `>`, `=`, or `>=`), according to the syntax diagram on the right-hand side (as soon as `=` or `>=` is used, we implicitly switch from strict to quasi-precedences).

prec



Knuth-Bendix Orders In addition to a precedence, KBO (`ttt2 -s 'kbo'`) is also parameterized by weights, which can be specified using the flags `-w0` (for the weight of variables) and `-weights` that take a single weight and a *weights* template, respectively, as arguments. Weights are natural numbers and can be specified for multiple function symbols at once, as depicted in the following syntax diagram:

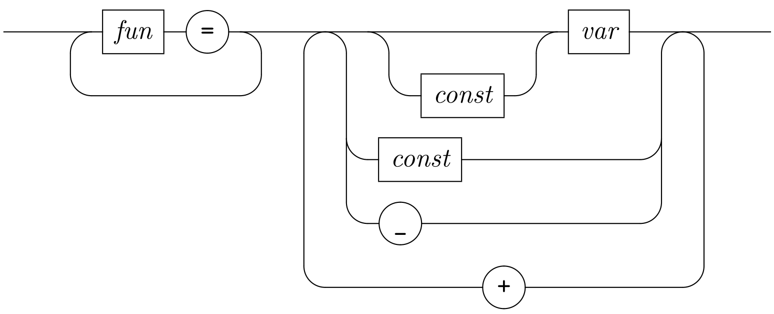
weights



If the last relation symbol of a *weights* template is `<=` or `>=`, then the specified weight is an upper and lower bound, respectively, of all preceding function symbols of the same template.

Linear Interpretations Linear interpretations can be specified using the flag `-inters` (supported by PIs and MIs) that takes an *inters* template as argument. Such interpretations are sums of linear monomials that may either be an optional coefficient followed by a variable, a constant part, or an underscore, according to the syntax diagram:

inters

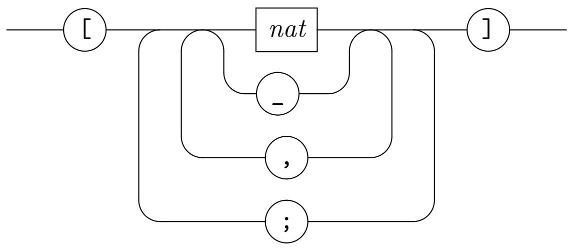


Here, underscores denote arbitrary (unrestricted) parts of an interpretation and variables are associated to function arguments via their index (starting from 0). For example, given a binary function symbol `f`, the template `2x0 + x1` fixes its interpretation to $[\mathtt{f}](x, y) = 2x + y$.

Polynomial Interpretations Polynomial interpretations (`ttt2 -s 'poly'`) are parameterized by linear polynomials as interpretations for function symbols. We obtain them by taking natural numbers for "const" in the *inters* template.

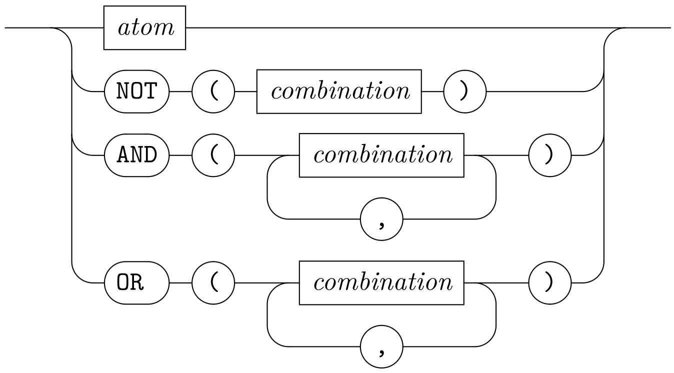
Matrix Interpretations For matrix interpretations (`ttt2 -s 'matrix'`), we instantiate "const" in the *inters* template by matrices of natural numbers as specified by the diagram on the right-hand side.[3] Moreover, we provide the shorthands `0` and `1` for the zero-vector and one-vector, respectively.

matrix



Boolean Combinations of Constraints For all of the above templates (as atoms), we actually support arbitrary boolean combinations of atomic constraints as specified below:

combination



As a common special case a comma-separated list of atoms is allowed at the toplevel, which is then interpreted as logical conjunction of atomic templates.

Examples We conclude this section by giving some example templates:

- Let us first consider an LPO such that `f` is not at the same time bigger than `g` and bigger than `h` in the precedence: `-prec "NOT(AND(f > g, f > h))"`
- Can we find (linear) polynomial interpretations such that the constant part of [`+`] is 2 and whenever `0` is interpreted as 0, then `s` should also obtain its natural interpretation? `-inters "AND(+ = _ + 2, OR(NOT(0 = 0), s = x0 + 1))"`
- How about matrix interpretations with upper triangular matrices of dimension three that have only ones on their diagonals, for unary `f` and binary `g` and `h`? `-inters "f=g=h=[1,_,_;0,1,_;0,0,1]x0+_, g=h=[1,_,_;0,1,_;0,0,1]x1+_"`

4 Teaching with the Web-Interface

It is the experience of the second author that for teaching, while live demonstrations of tools are often appreciated by students, properly setting up such examples often takes way too much time. To remedy this situation at least for $\mathsf{T_TT_2}$, we integrated a mechanism into

[3] When the flag `-inters` is present, the matrix dimension is read from the template, which usually means that `-dim` is not required anymore.

its web interface[4] that allows a user to store the current configuration (including the input TRS) into the query part of the web interface URL.

Now, we can easily copy such a URL and turn it, for example, into a PDF hyperlink on slides or in a paper. Incidentally, this is exactly the purpose of the ☑ symbols in this paper. The first one, for example, is generated by the following (incomplete) LaTeX code:

```
\href{http://colo6-c703.uibk.ac.at/ttt2/web/?problem=(VAR\%20x\%20y)...
```

Moreover, our termination templates are accessible through dedicated input fields of the web interface. Thus, a user does not have to type (and know) the corresponding flags.

5 Conclusion and Future Work

We presented the extension of TTT2 by a template mechanism for four of the most common termination techniques that are taught in classes. These templates allow us to narrow down the search space such that in the extreme, we are left with a specific instance of a termination technique. In this way, TTT2 can be used as a "proof checker" for termination proofs (if you want to have near absolute certainty that some termination proof is correct, you should of course validate TTT2's output by a formally verified certifier like CeTA). Furthermore, we demonstrated a simple but useful addition to TTT2's web interface to generate URLs that make the input TRS and current configuration persistent.

We leave templates for methods like *arctic interpretations* [2], the *(generalized) subterm criterion* [4], and *finding loops* [5] as future work. An orthogonal issue that requires further investigation is: which extensions to the current templates would be most useful for teaching?

Some similar options to our termination templates were available through a GUI in an old version of AProVE [1, Section 2]. Unfortunately, this GUI was abandoned in later versions.

References

1 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Automated Termination Proofs with AProVE. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications (RTA)*, volume 3091 of *Lecture Notes in Computer Science*, pages 210–220. Springer, 2004. 10.1007/978-3-540-25979-4_15.

2 Adam Koprowski and Johannes Waldmann. Arctic Termination ... Below Zero. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications (RTA)*, volume 5117 of *Lecture Notes in Computer Science*, pages 202–216. Springer, 2008. 10.1007/978-3-540-70590-1_14.

3 Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean Termination Tool 2. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304. Springer, 2009. 10.1007/978-3-642-02348-4_21.

4 Christian Sternagel. The generalized subterm criterion in TTT2. In *Proceedings of the 15th Workshop on Termination (WST)*, 2016. `arXiv:1609.03432`.

5 Harald Zankl, Christian Sternagel, Dieter Hofbauer, and Aart Middeldorp. Finding and certifying loops. In *Proceedings of the 36th International Conference on Theory and Practice of Computer Science (SOFSEM)*, volume 5901 of *Lecture Notes in Computer Science*, pages 755–766. Springer, 2010. 10.1007/978-3-642-11266-9_63.

[4] `http://colo6-c703.uibk.ac.at/ttt2/web/`