The Generalized Subterm Criterion in TTT2*

Christian Sternagel

University of Innsbruck, Austria
`christian.sternagel@uibk.ac.at`

Abstract

We present an SMT encoding of a generalized version of the subterm criterion and evaluate its implementation in TTT2.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases termination, subterm criterion, SMT encodings

1 Preliminaries

We assume basic familiarity with term rewriting [1] in general and the dependency pair framework [3] for proving termination in particular. We start with a recap of terminology and notation that we use in the remainder.

By $\mathcal{M}(A)$, we denote the set of *finite multisets* ranging over elements from the set A. We write $M(x)$ for the *multiplicity* (i.e., number of occurrences) of x in the multiset M, use $+$ for multiset sum, but otherwise use standard set-notation.

Given a relation $\succ$, its *restriction to the set* A, written $\succ_{\downarrow A}$, is the relation defined by the set $\{(x,y) \mid x \succ y, x \in A, y \in A\}$. Moreover, for any function f, we use $x \succ^f y$ as a shorthand for $f(x) \succ f(x)$.

The *multiset extension* $\succ_{\mathsf{mul}}$ of a given relation $\succ$ is defined by:

$$M \succ_{\mathsf{mul}} N \text{ iff } \exists X\ Y\ Z.\ X \neq \varnothing, M = X + Z, N = Y + Z, \forall y \in Y.\ \exists x \in X.\ x \succ y$$

A useful fact about the multiset extension is that we may always "maximize" the common part Z in the above definition.

▶ **Lemma 1.** *Consider an irreflexive and transitive relation $\succ$ and multisets M, N such that $M \succ_{\mathsf{mul}} N$. Moreover, let $X = M - M \cap N$ and $Y = N - M \cap N$. Then $X \neq \varnothing$ and $\forall y \in Y.\ \exists x \in X.\ x \succ y$.*

While intuitively obvious, a rigorous proof of this fact does not seem to be widely known.[1] In preparation for the proof, we recall the following easy fact about finite relations.

▶ **Lemma 2.** *Every finite, irreflexive, and transitive relation is well-founded.*

Proof. Let $\succ$ be a finite, irreflexive, and transitive relation. For the sake of a contradiction, assume that $\succ$ is not well-founded. Then there is an infinite sequence $a_1 \succ a_2 \succ a_3 \succ \cdots$ whose elements are in the finite (since $\succ$ is finite) field of $\succ$. But then, by the (infinite) pigeonhole principle, there is some recurring element a_i, i.e., $\cdots \succ a_i \succ \cdots \succ a_i \succ \cdots$. By transitivity we obtain $a_i \succ a_i$ contradicting the irreflexivity of $\succ$. ◀

* This work was supported by FWF (Austrian Science Fund) project P27502.

[1] An alternative proof of this fact is indicated in Vincent van Oostrom's PhD thesis [7].

Noting that the converse of any finite, irreflexive, and transitive relation is again finite, irreflexive, and transitive, Lemma 2 allows us to employ well-founded induction where the induction hypothesis holds for "bigger" elements, as exemplified in the following proof.

Proof of Lemma 1. Since $M \succ_{\mathsf{mul}} N$ we obtain $I \neq \varnothing$, J, and K such that $M = I + K$, $N = J + K$, and $\forall j \in J.\, \exists i \in I.\, i \succ j$. Let $A = I - I \cap J$, $B = J - I \cap J$, and consider the finite set D of elements occurring in either of I and J. Now, appealing to Lemma 2, we employ well-founded induction with respect to $\prec_{\downarrow D}$ in order to prove:

$$\forall j \in J.\, \exists a \in A.\, a \succ j \tag{†}$$

Thus we assume $j \in J$ for some arbitrary but fixed j and obtain the induction hypothesis (IH) $\forall c \succ_{\downarrow D} j.\, c \in J \longrightarrow \exists a \in A.\, a \succ c$. From $j \in J$ we obtain an $i \in I$ with $i \succ j$. Now if $i \in A$, then we are done. Otherwise, $i \in J$ and by IH we obtain an $a \in A$ with $a \succ i$. Since $\succ$ is transitive, this implies $a \succ j$, concluding the proof of (†). But then also $\forall b \in B. \exists a \in A. x \succ b$ and $A \neq \varnothing$. We conclude by noting the following two equalities:

$$X = M - M \cap N = (I + K) - (I + K) \cap (J + K) = I - I \cap J = A,$$
$$Y = N - M \cap N = (J + K) - (I + K) \cap (J + K) = J - I \cap J = B.$$ ◀

2 A Generalized Subterm Criterion

Recall the subterm criterion – originally by Hirokawa and Middeldorp [4] and later reformulated as a processor for the dependency pair framework – which is a particularly elegant technique (due to its simplicity and the fact that the $\mathcal{R}$-component of a dependency pair problem $(\mathcal{P}, \mathcal{R})$ may be ignored).

▶ **Definition 3** (Simple projections). A *simple projection* is a function $\pi : \mathcal{F} \to \mathbb{N}$ that maps every n-ary function symbol f to some natural number $\pi(f) \in \{1, \ldots, n\}$. Applying a simple projection to a term is defined by $\pi(f(t_1, \ldots, t_n)) = t_{\pi(f)}$.

▶ **Theorem 4.** *If $\mathcal{P} \subseteq \trianglerighteq^{\pi}$ for simple projection π, then $(\mathcal{P}, \mathcal{R})$ is finite iff $(\mathcal{P} \setminus \triangleright^{\pi}, \mathcal{R})$ is.* ◀

Recall that the appropriate notion of *finiteness* for the subterm criterion is "the absence of minimal infinite chains."

For an AC-variant of the subterm criterion (i.e., a variant for rewriting modulo associative and/or commutative function symbols), Yamada et al. [8] generalized simple projections to so-called multiprojections.

▶ **Definition 5** (Multiprojections). A *multiprojection* is a function $\pi : \mathcal{F} \to \mathcal{M}(\mathbb{N})$ that maps every n-ary function symbol f to a multiset $\pi(f) \subseteq \mathcal{M}(\{1, \ldots, n\})$. Applying a multiprojection to a term yields a multiset of terms as follows:

$$\pi(t) = \begin{cases} \pi(t_{i_1}) + \cdots + \pi(t_{i_k}) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } \pi(f) = \{i_1, \ldots, i_k\} \neq \varnothing, \\ \{t\} & \text{otherwise.} \end{cases}$$

We write $s \trianglerighteq^{\pi}_{\mathsf{mul}} t$ if either $s \triangleright^{\pi}_{\mathsf{mul}} t$ or $\pi(s) = \pi(t)$.

A compromise between simple projections and full multiprojections is to allow recursive projections (possibly through defined symbols). While theoretically subsumed by multiprojections, we included such recursive projections in our experiments in order to assess their performance in practice.

The following is a specialization of the AC subterm criterion by Yamada et al. [8, Theorem 33] to the non-AC case.

▶ **Theorem 6.** *Let π be a multiprojection such that $\mathcal{P} \subseteq \trianglerighteq^{\pi}_{\mathsf{mul}}$ and $f(\ldots) \trianglerighteq^{\pi}_{\mathsf{mul}} r$ for all $f(\ldots) \to r \in \mathcal{R}$ with $\pi(f) \neq \varnothing$. Then $(\mathcal{P}, \mathcal{R})$ is finite iff $(\mathcal{P} \setminus \triangleright^{\pi}_{\mathsf{mul}}, \mathcal{R})$ is.* ◀

This result (which is also formalized in IsaFoR [6]) states the soundness of a generalized version of the subterm criterion and thus gives the theoretical backing for implementing such a technique in a termination tool. In the following we are concerned with the more practical problem of an efficient implementation.

That is, given a DP problem $(\mathcal{P}, \mathcal{R})$ we want to find a multiprojection π that satisfies the conditions of Theorem 6 and orients at least one rule of $\mathcal{P}$ strictly by $\triangleright^{\pi}_{\mathsf{mul}}$.

Since the problem of finding such a multiprojection seems similar to the problem of finding an appropriate argument filter for a reduction pair [2], and the latter has been successfully tackled by various kinds of SAT and SMT encodings, we take a similar approach.

3 Implementation and Experiments

There are basically two issues that have to be considered: (1) how to encode a multiprojection π and thereby the multiset $\pi(s)$, and (2) how to encode the comparison between two encodings of multisets with respect to the multiset extension of $\triangleright$.

In the following we use lowercase sans serif for propositional and arithmetical variables, and UPPERCASE SANS SERIF for functions that result in formulas.

Encoding Multiprojections. We encode the multiplicity of a term t in the multiset $\pi(s)$, which is 0 if t does not occur in $\pi(s)$ at all, by $\mathsf{M}_s(t) = \mathsf{MUL}(1, s, t)$. The latter is defined as follows

$$\mathsf{MUL}(w, s, t) = \begin{cases} \left(\bigwedge_{1 \leq i \leq n} \neg \mathsf{p}^i_f\right) \mathrel{?} w : 0 & \text{if } s = t = f(t_1, \ldots, t_n) \\ w & \text{if } s = t \text{ and } t \text{ is a variable} \\ \sum_{1 \leq i \leq n} (\mathsf{p}^i_f \mathrel{?} \mathsf{MUL}(w \cdot \mathsf{w}^i_f, s_i, t) : 0) & \text{if } t \triangleleft s = f(s_1, \ldots, s_n) \\ 0 & \text{otherwise} \end{cases}$$

where $b \mathrel{?} t : e$ denotes *if* b *then* t *else* e and the intended meaning of variables is that $\mathsf{p}^i_f = \top$ precisely when π *projects to the* i*-th argument of* f, in which case w^i_f gives the *weight of* i *in* $\pi(f)$, i.e., its number of occurrences in $\pi(f)$.[2]

Encoding Multiset Comparison. Now consider the problem of finding π such that $s \triangleright^{\pi}_{\mathsf{mul}} t$ for given terms s and t. Noting that, independent of the exact π, $\pi(s)$ and $\pi(t)$ are multisets over the finite set of subterms of s and t, it suffices to find an encoding for comparing multisets over finite domains. This allows us to make use of the following observation.

▶ **Lemma 7** (Comparing multisets over finite domains). *Let D be a finite set, and $M, N \subseteq \mathcal{M}(D)$. Then, for irreflexive and transitive $\succ$, $M \succ_{\mathsf{mul}} N$ is equivalent to*

$$\forall d \in D.\, \mathrm{upper}(d) \longrightarrow M(d) \geq N(d) \text{ and } M \neq N \tag{$\star$}$$

where $\mathrm{upper}(x)$ *iff* $\forall d \in D.\, d \succ x \longrightarrow M(d) = N(d)$.

[2] In experiments, replacing $\mathsf{p}^i_f = \top$ by $\mathsf{w}^i_f > 0$ resulted in a slightly increased number of timeouts.

Proof. We start with the direction from ($\star$) to $M \succ_{\mathsf{mul}} N$. Assume ($\star$) for M and N, and define the multisets $Z = \{x \in M \cap N \mid \text{upper}(x)\}$, $X = M - Z$, and $Y = N - Z$ (i.e., $M = X + Z$ and $N = Y + Z$). Then, appealing to Lemma 2, we use well-founded induction with respect to $\prec_{\downarrow D}$ in order to prove

$$\forall y \in Y.\, \exists x \in X.\, x \succ y \tag{‡}$$

Thus we assume $y \in Y$ for some arbitrary but fixed y and obtain the induction hypothesis (IH) $\forall z \succ_{\downarrow D} y.\, z \in Y \longrightarrow \exists x \in X.\, x \succ z$. Also note that $\neg\text{upper}(y)$, since otherwise $M(y) \geq N(y)$ by ($\star$) and thus $Z(y) = N(y)$, contradicting $y \in Y$. Therefore, we obtain $z \succ y$ with $M(z) \neq N(z)$ by definition of upper. Now, either $M(z) > N(z)$ or $N(z) > M(z)$. In the former case $z \in X$ and we are done. In the latter case $z \in Y$ and thus we obtain an $x \in X$ such that $x \succ z$ by IH and conclude (‡) by transitivity of $\succ$. It remains to show $X \neq \varnothing$. Since $M \neq N$ there is some x with $M(x) \neq N(x)$. If $M(x) > N(x)$, then $x \in X$ and we are done. Otherwise, $N(x) > M(x)$ and thus $x \in Y$ and we conclude by invoking (‡).

For the other direction, assume $M \succ_{\mathsf{mul}} N$. Then for $Z = M \cap N$, $X = M - Z$, and $Y = M - Z$, we have $X \neq \varnothing$, $X \cap Y = \varnothing$, $M = X + Z$, $N = Y + Z$ and $\forall y \in Y. \exists x \in X. x \succ y$, using Lemma 1. This further implies $M \neq N$. Now assume $d \in D$ and upper(d). Then either $d \in Y$ or $d \notin Y$. In the latter case, clearly $M(d) \geq N(d)$, and we are done. In the former case, we obtain an $x \in X$ with $x \succ d$. Moreover, since $X \cap Y = \varnothing$, we have $x \notin Y$. But then $M(x) \neq N(x)$, contradicting upper(d). ◀

Encoding the Generalized Subterm Criterion. Putting everything together we obtain the encoding

$$(\forall s \to t \in \mathcal{P}.\, \mathsf{GEQ}(s,t)) \wedge (\exists s \to t \in \mathcal{P}.\, \mathsf{NEQ}(s,t)) \wedge$$
$$(\forall s \to t \in \mathcal{R}.\, \mathsf{RT}(s) \longrightarrow \mathsf{GEQ}(s,t)) \wedge (\forall f \in \mathcal{F}(\mathcal{P},\mathcal{R}).\, \mathsf{SAN}(f))$$

where

$$\begin{aligned}
\mathsf{GEQ}(s,t) &\text{ iff } \forall u \in \mathcal{S}\mathsf{ub}(s,t).\, \mathsf{UPPER}(u) \longrightarrow \mathsf{M}_s(u) \geq \mathsf{M}_t(u)\\
\mathsf{UPPER}(u) &\text{ iff } \forall v \in \mathcal{S}\mathsf{ub}(s,t).\, v \rhd u \longrightarrow \mathsf{M}_s(v) = \mathsf{M}_t(v)\\
\mathsf{NEQ}(s,t) &\text{ iff } \neg(\forall u \in \mathcal{S}\mathsf{ub}(s,t).\, \mathsf{M}_s(u) = \mathsf{M}_t(u))\\
\mathsf{RT}(f(s_1,\ldots,s_n)) &\text{ iff } \exists 1 \leq i \leq n.\, \mathsf{p}_f^i.\\
\mathsf{SAN}(f) &\text{ iff } \bigwedge_{1 \leq i \leq \mathsf{arity}(f)} \left(\mathsf{p}_f^i \longrightarrow \mathsf{w}_f^i > 0\right)
\end{aligned}$$

Here $\mathcal{S}\mathsf{ub}(s,t)$ denotes the set of all (i.e., including s and t themselves) subterms of s and t, and SAN is a "sanity check" that makes sure that propositional and arithmetical variables play well together. Every satisfying assignment gives rise to a multiprojection π satisfying the conditions of Theorem 6.

Experiments. We conducted experiments in order to assess our implementation. To this end we took all the 1498 TRSs in the standard (as in "standard term rewriting") category of the termination problem database (TPDB) version 10.3 and tried to prove their termination with the following strategy: first compute dependency pairs, then compute the estimated dependency graph $\mathcal{G}$, and finally try repeatedly to either decompose $\mathcal{G}$ into strongly connected components or apply the subterm criterion. For the subterm criterion we tried either *simple projections* (simple), *recursive projections* (recursive), *multiprojection* (multi), or a parallel combination of those (all).

Table 1 Experiments on 1498 standard TRSs of TPDB 10.3

	Yes		Maybe		Timeout		
Projections	#	(sec)	#	(sec)	#	(sec)	Total (sec)
simple	265	31.1	1184	226.8	49	254.0	502.9
recursive	292	35.4	1155	240.4	51	255.0	530.9
multi	351	61.2	1081	419.0	66	330.0	810.2
all	352	30.4	1099	230.3	47	235.0	495.7

In summary, the parallel combination of different kinds of projections results in a significant increase of power (i.e., number of yeses) and does not have a negative impact on the speed, compared to the original implementation of the subterm criterion (simple) of TTT2 [5].

Encouraged by this results, we incorporated our new implementation also into the competition strategy of TTT2 and compared it to its 2015 competition version. In this way, we were able to obtain 12 additional yeses. However, each of those 12 systems could already be handled by some other termination tool in the 2015 termination competition.

Acknowledgments. We thank Vincent van Oostrom for pointing us to Lemma 7 and Bertram Felgenhauer for helpful discussion concerning MUL. We further thank the Austrian Science Fund (FWF project P27502) for supporting this work.

References

1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. 10.1017/CBO9781139172752.

2 Michael Codish, Peter Schneider-Kamp, Vitaly Lagoon, René Thiemann, and Jürgen Giesl. SAT solving for argument filterings. In *Proc. 13th LPAR*, volume 4246 of *LNCS*, pages 30–44. Springer, 2006. 10.1007/11916277_3.

3 Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. 11th LPAR*, volume 3452 of *LNCS*, pages 301–331. Springer, 2005. 10.1007/978-3-540-32275-7_21.

4 Nao Hirokawa and Aart Middeldorp. Dependency pairs revisited. In *Proc. 15th RTA*, volume 3091 of *LNCS*, pages 249–268. Springer, 2004. 10.1007/978-3-540-25979-4_18.

5 Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean Termination Tool 2. In *Proc. 20th RTA*, volume 5595 of *LNCS*, pages 295–304. Springer, 2009. 10.1007/978-3-642-02348-4_21.

6 René Thiemann and Christian Sternagel. Certification of termination proofs using CeTA. In *Proc. 22nd TPHOLs*, volume 5674 of *LNCS*, pages 452–468. Springer, 2009. 10.1007/978-3-642-03359-9_31.

7 Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1994.

8 Akihisa Yamada, Christian Sternagel, René Thiemann, and Keiichirou Kusakari. AC dependency pairs revisited. In *Proc. 25th CSL*, LIPIcs. Schloss Dagstuhl, 2016. to appear.