SAT Solving for Termination Analysis with Polynomial Interpretations*

Carsten Fuhs[1], Jürgen Giesl[1], Aart Middeldorp[2], Peter Schneider-Kamp[1], René Thiemann[1], and Harald Zankl[2]

[1] LuFG Informatik 2, RWTH Aachen, Germany,
[2] Institute of Computer Science, University of Innsbruck, Austria,

1 Termination of TRSs and Polynomial Interpretations

Most termination methods for TRSs transform the termination problem into a set of inequalities between terms. For example, a classical approach is to generate inequalities $\ell \succ r$ for all rules $\ell \to r$ or, if one uses the dependency (DP) framework [1, 3, 5], then one generates strict inequalities for the DPs and non-strict ones for the usable rules. Consider the following example for subtraction.

$$\begin{array}{rr} \mathsf{p}(\mathsf{0}) \to \mathsf{0} & \mathsf{minus}(x, \mathsf{0}) \to x \\ \mathsf{p}(\mathsf{s}(x)) \to x & \mathsf{minus}(x, \mathsf{s}(y)) \to \mathsf{minus}(\mathsf{p}(x), y) \end{array}$$

For the DP of the recursive minus-call we get the following constraints. Here, M is the tuple-symbol of minus.

$$\mathsf{p}(\mathsf{0}) \succsim \mathsf{0} \quad (1) \qquad \mathsf{p}(\mathsf{s}(x)) \succsim x \quad (2) \qquad \mathsf{M}(x, \mathsf{s}(y)) \succ \mathsf{M}(\mathsf{p}(x), y) \quad (3)$$

A popular method to search for relations $\succ$ and $\succsim$ automatically are *polynomial interpretations* [8]. A polynomial interpretation $\mathcal{P}ol$ maps each n-ary function symbol f to a polynomial $f_{\mathcal{P}ol}$ over n variables $x_1, \ldots, x_n$ with coefficients from $\mathbb{N} = \{0, 1, 2, \ldots\}$. It is extended to a mapping $[\cdot]_{\mathcal{P}ol}$ on terms where $[x]_{\mathcal{P}ol} = x$ for variables x and $[f(t_1, \ldots, t_n)]_{\mathcal{P}ol} = f_{\mathcal{P}ol}\{x_1/[t_1]_{\mathcal{P}ol}, \ldots, x_n/[t_n]_{\mathcal{P}ol}\}$. We often write $[\cdot]$ if $\mathcal{P}ol$ is clear from the context. Now a term u is greater (resp. greater-equal) than v iff $[u] \geq [v] + 1$ (resp. $[u] \geq [v]$) holds for all instantiations of the variables with natural numbers. For instance, the constraints of the example are satisfied using the polynomial interpretation $\mathcal{P}ol_1$ with $\mathsf{M}_{\mathcal{P}ol_1} = x_2$, $\mathsf{p}_{\mathcal{P}ol_1} = x_1$, $\mathsf{s}_{\mathcal{P}ol_1} = x_1 + 1$, and $\mathsf{0}_{\mathcal{P}ol_1} = 0$. Thus, termination of the example is proved.

To find such interpretations automatically, one starts with an *abstract* polynomial interpretation. In the linear case we obtain

$$f_{\mathcal{P}ol} = f_0 + f_1 x_1 + \cdots + f_n x_n \quad \text{for each } f \text{ with arity } n \qquad (4)$$

where the coefficients f_i are left open. Then one translates the term constraints into polynomial constraints. In the example we obtain

$$\mathsf{p}_0 + \mathsf{p}_1\mathsf{0}_0 \geq \mathsf{0}_0 \quad (5) \qquad (\mathsf{p}_0 + \mathsf{p}_1\mathsf{s}_0) + (\mathsf{p}_1\mathsf{s}_1) * x \geq x \quad (6)$$
$$(\mathsf{M}_0 + \mathsf{M}_2\mathsf{s}_0) + \mathsf{M}_1 * x + \mathsf{M}_2\mathsf{s}_1 * y \geq (\mathsf{M}_0 + \mathsf{M}_1\mathsf{p}_0 + 1) + \mathsf{M}_1\mathsf{p}_1 * x + \mathsf{M}_2 * y \quad (7)$$

Next one simplifies these constraints by deleting the variables $x, y, \ldots$ that are

* Supported by the DFG grant GI 274/5-1 and the FWF project P18763.

(implicitly) universally quantified. To this end, instead of an inequality between polynomials we only compare the respective coefficients ("absolute positiveness" [7]). In the example, the resulting constraints are (5), (8) and (9) (using $x = 0 + 1 * x$), and (10) – (12).

$$\mathsf{p}_0 + \mathsf{p}_1\mathsf{s}_0 \geq 0 \quad (8) \qquad \mathsf{p}_1\mathsf{s}_1 \geq 1 \quad (9)$$
$$\mathsf{M}_2\mathsf{s}_0 \geq \mathsf{M}_1\mathsf{p}_0 + 1 \quad (10) \qquad \mathsf{M}_1 \geq \mathsf{M}_1\mathsf{p}_1 \quad (11) \qquad \mathsf{M}_2\mathsf{s}_1 \geq \mathsf{M}_2 \quad (12)$$

Now to prove termination one has to show the *satisfiability* of such *Diophantine constraints* over the naturals. In the example, a solution of the constraints is $\mathsf{0}_0 = \mathsf{p}_0 = \mathsf{M}_0 = \mathsf{M}_1 = 0$ and $\mathsf{p}_1 = \mathsf{s}_0 = \mathsf{s}_1 = \mathsf{M}_2 = 1$. In this way, the abstract interpretation is turned into the polynomial interpretation $\mathcal{P}ol_1$.

In the next section, we show how to check this satisfiability using SAT solvers.

2 Encoding Diophantine Constraints to SAT

To encode Diophantine constraints into SAT we first present a mapping $||\cdot||$ from polynomials to tuples of propositional formulas which are interpreted as *binary representations* of the polynomials. We restrict the search to coefficients in the range $\{0, \ldots, 2^k - 1\}$ for a fixed k. Then each coefficient f is encoded into $||f|| = \langle f^{k-1}, \ldots, f^0 \rangle$ where $f^0, \ldots, f^{k-1}$ are propositional variables. Similarly, a natural number $n = b_\ell * 2^\ell + \ldots b_1 * 2^1 + b_0$ is encoded into $||n|| = \langle b_\ell, \ldots, b_1, b_0 \rangle$ where 0 and 1 are identified with false and true. So if $k = 2$ then $||\mathsf{s}_0|| = \langle \mathsf{s}_0^1, \mathsf{s}_0^0 \rangle$ and $||6|| = \langle 1, 1, 0 \rangle$. For addition and multiplication, we introduce operations B^+ and B^* on tuples of propositional formulas and define

$$||p + q|| = B^+(||p||, ||q||) \quad \text{and} \quad ||p * q|| = B^*(||p||, ||q||)$$

for all polynomials p and q. For B^+ we essentially use the idea of a ripple-carry-adder. The details are presented in [2]. For example $||\mathsf{s}_0 + 6|| = \langle \mathsf{s}_0^1, \neg\mathsf{s}_0^1, \neg\mathsf{s}_0^1, \mathsf{s}_0^0 \rangle$. We encode multiplication by summing up partial products as follows:

- $B^*(\langle \varphi_1, \ldots, \varphi_n \rangle, \langle \psi \rangle) = \langle \varphi_1 \wedge \psi, \ldots, \varphi_n \wedge \psi \rangle$
- $B^*(\langle \varphi_1, \ldots, \varphi_n \rangle, \langle \psi_1, \ldots, \psi_m \rangle) = B^+(\langle \varphi_1 \wedge \psi_1, \ldots, \varphi_n \wedge \psi_1, \overbrace{0, \ldots, 0}^{m-1 \text{ times}} \rangle,$
  $B^*(\langle \varphi_1, \ldots, \varphi_n \rangle, \langle \psi_2, \ldots, \psi_m \rangle)\,)$ if $m \geq 2$.

Now we extend $||\cdot||$ to map each Diophantine constraint to a formula (not to a tuple). To this end, we define the operation $B^{\geq}$ which encodes comparisons.

$$||p \geq q|| = B^{\geq}(||p||, ||q||)$$

For $B^{\geq}$ we apply zero-padding and compare tuples lexicographically:

- $B^{\geq}(\langle \varphi \rangle, \langle \psi \rangle) = \psi \rightarrow \varphi$
- $B^{\geq}(\langle \varphi_1, \ldots, \varphi_n \rangle, \langle \psi_1, \ldots, \psi_n \rangle) = (\varphi_1 \wedge \neg\psi_1) \vee$
  $((\varphi_1 \leftrightarrow \psi_1) \wedge B^{\geq}(\langle \varphi_2, \ldots, \varphi_n \rangle, \langle \psi_2, \ldots, \psi_n \rangle))$ if $n \geq 2$.

So to determine the satisfiability of a set of Diophantine constraints $p_i \geq q_i$ with coefficients from $\{0, \ldots, 2^k - 1\}$, we encode it as a conjunction $\bigwedge_i ||p_i \geq q_i||$ of propositional formulas. Then we use a SAT solver to find an assignment for the

coefficients. Note that the space complexity of our encoding is polynomial. More precisely, whenever all numbers in "$p \geq q$" are smaller than $2^k - 1$, then the size of $||p \geq q||$ is in $\mathcal{O}(|p \geq q|^2 * k^2)$.

3 Polynomials with Negative Constant

Now we regard polynomials $f_{\mathcal{P}ol}$ which may have a negative constant coefficient (i.e., in (4) one may have $f_0 < 0$). All other coefficients still have to be natural numbers. This is needed if we replace the recursive minus-rule by $\mathsf{minus}(x, \mathsf{s}(y)) \to \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y)))$. Then the constraints are (1), (2), and

$$\mathsf{M}(x, \mathsf{s}(y)) \succ \mathsf{M}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y))), \tag{13}$$

which cannot be satisfied using non-negative polynomial interpretations. Thus, we use a polynomial interpretation $\mathcal{P}ol_2$ with $\mathsf{p}_{\mathcal{P}ol_2} = x_1 - 1$. To avoid that terms are mapped to negative integers (which would violate well-foundedness), [6] modified the interpretation of terms $[\cdot]$. Now one defines $[\mathsf{p}(x)] = \max(x-1, 0)$.

The problem is that with these interpretations, inequalities like $u \gtrsim v$ are transformed into $[u] \geq [v]$ where $[u]$ and $[v]$ are no polynomials anymore, as they contain "max". To solve this problem, let us first regard *concrete* polynomial interpretations (where the coefficients are actual numbers). Here, [6] presented an approach to transform inequalities like $[u] \geq [v]$ into ordinary polynomial inequalities without "max". The idea is to define an under-approximation $[.]^{left}$ and an over-approximation $[.]^{right}$ which do not contain "max" anymore. Then instead of $[u] \geq [v]$ one requires $[u]^{left} \geq [v]^{right}$.

Definition 1 ($[.]^{left}$ [6]). *For every polynomial q we denote its constant part by $con(q)$. For any term t, we define the polynomial $[t]^{left}$ as follows:*[3]

$$[t]^{left} = \begin{cases} t & \text{if } t \text{ is a variable} \\ 0 & \text{if } t = f(t_1, \ldots, t_n),\ q - con(q) = 0, \text{ and } 0 > con(q) \\ q & \text{if } t = f(t_1, \ldots, t_n), \text{ otherwise} \end{cases}$$

where $q = f_{\mathcal{P}ol}([t_1]^{left}, \ldots, [t_n]^{left})$.

For example, using $\mathcal{P}ol_2$ we obtain $[\mathsf{p}(x)]^{left} = x - 1 \leq \max(x - 1, 0) = [\mathsf{p}(x)]$. The reason is that $q = \mathsf{p}_{\mathcal{P}ol_2}(x) = x - 1$ and thus $con(q) = -1$ and $q - con(q) = x$. If $\mathcal{P}ol_2$ is defined like our previous interpretation $\mathcal{P}ol_1$ on all remaining function symbols except p, then the constraints (1), (2), and (13) are satisfied and termination of our modified example is proved.

The disadvantage of Def. 1 is that one can only compute $[t]^{left}$ and $[t]^{right}$ for concrete polynomial interpretations, since otherwise q contains coefficients and it depends on the assignment whether $q - con(q) = 0$ and $0 > con(q)$ is valid.

For example, we would use an abstract interpretation with $\mathsf{p}_{\mathcal{P}ol} = \mathbf{p_0} + \mathsf{p}_1 x_1$. Here, p_1 may only be instantiated by natural numbers, whereas a coefficient like $\mathbf{p_0}$ (written in **bold** face) may be instantiated by integers.

The idea is to introduce new coefficients $\boldsymbol{b}_t^{\boldsymbol{left}}$ and $\boldsymbol{b}_t^{\boldsymbol{right}}$ for any term t and to

[3] The definition of $[.]^{right}$ is similar to $[.]^{left}$, see [2, 6] for details.

create additional Diophantine constraints α_t^{left} and α_t^{right} which guarantee that $\boldsymbol{b}_t^{\boldsymbol{left}}$ and $\boldsymbol{b}_t^{\boldsymbol{right}}$ are instantiated correctly, depending on the possible values for the other coefficients. To this end, we express the conditions $q - con(q) = 0$ and $0 > con(q)$ from Def. 1 as Diophantine constraints and one can encode $u \succsim v$ by

$$\alpha_t^{left} \wedge \alpha_t^{right} \wedge [u]^{left} \geq [v]^{right}.$$

Here, $[\cdot]^{left}$ and $[\cdot]^{right}$ interpret terms with the help of the additional coefficients $\boldsymbol{b}_t^{\boldsymbol{left}}$ and $\boldsymbol{b}_t^{\boldsymbol{right}}$. For example, $\mathsf{p}(x) \succsim x$ is transformed into:

$$\begin{array}{rll} (\mathsf{p}_1 = 0 \wedge 0 > \mathbf{p_0}) \rightarrow \boldsymbol{b}_{\mathsf{p}(x)}^{\boldsymbol{left}} = 0 & \wedge & \\ \neg(\mathsf{p}_1 = 0 \wedge 0 > \mathbf{p_0}) \rightarrow \boldsymbol{b}_{\mathsf{p}(x)}^{\boldsymbol{left}} = \mathbf{p_0} & \wedge & \mathsf{p}_1 x + \boldsymbol{b}_{\mathsf{p}(x)}^{\boldsymbol{left}} \geq x \end{array}$$

So we obtain polynomial constraints containing **bold** coefficients like $\mathbf{p_0}$ and $\boldsymbol{b}_{\mathsf{p}(x)}^{\boldsymbol{left}}$ which may be instantiated by *integers*. In [2] it is shown how to remove these coefficients in order to apply our SAT encoding afterwards.

4 Implementation, Experiments, and Conclusion

We implemented our new SAT-based approach for polynomial interpretations in the termination prover AProVE [4] and used state-of-the art SAT solvers for the experiments. The SAT-based search for polynomial interpretations is by orders of magnitude faster than other non-SAT-based search algorithms. This holds in particular if one considers polynomials with higher coefficients or degrees. For details on the experiments and for the full paper [2] we refer to our evaluation web site at `http://aprove.informatik.rwth-aachen.de/eval/SATPOLO`.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT Solving for Termination Analysis with Polynomial Interpretations. In *Proc. SAT'07*, LNCS 4501, pages 340–354, 2007.
3. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In *Proc. LPAR'04*, LNAI 3452, pages 301–331, 2005.
4. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the Dependency Pair Framework. In *Proc. IJCAR'06*, LNAI 4130, pages 281–286, 2006.
5. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.
6. N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.
7. H. Hong and D. Jakuš. Testing positiveness of polynomials. *Journal of Automated Reasoning*, 21(1):23–38, 1998.
8. D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.