DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSIDAD POLITÉCNICA DE VALENCIA

P.O. Box: 22012 E-46071 Valencia (SPAIN)

Informe Técnico / Technical Report

Ref. No.:	DSIC-II/16/07	**Pages:** 14
Title:	Fast SAT-based polynomial constraint solving for termination tools	
Author(s):	Salvador Lucas, Rafael Navarro	
Date:	Jul 26, 2007	
Keywords:	SAT, polynomial interpretation, constraint solving, termination.	

V^o B^o
Leader of research Group Author(s)

Fast SAT-based polynomial constraint solving for termination tools*

Salvador Lucas Rafael Navarro

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{slucas,rnavarro}@dsic.upv.es

Abstract

Proofs of termination in term rewriting involve solving constraints between terms coming from (parts of) the rules of the term rewriting system. Several recent works develop connections between these problems and more standard constraint solving problems for which well-known and efficient techniques apply. In particular, SAT techniques are receiving increasing attention in the field. A very common way to deal with constraints in termination tools is treating them as *polynomial constraints*. Accordingly, a recent paper by Fuhs et al. shows how to encode polynomial constraints as propositional constraints which are managed by using state-of-the-art SAT solvers. In this paper we show how to improve this encoding to obtain faster SAT-based algorithms for solving polynomial constraints in restricted (but still relevant) domains of coefficients.

1 Introduction

Proofs of termination in term rewriting involve solving constraints between terms s and t coming from (parts of) the rules of the Term Rewriting System (TRS [18, 20]). For instance, in proofs of termination using the dependency pairs approach [2], given a rewrite rule $l \to r$ of a TRS $\mathcal{R}$, we get dependency pairs $l^\sharp \to s^\sharp$ for all subterms s of r which are rooted by a defined symbol[1]; the notation $t^\sharp$ for a given term t means that the root symbol f of t is *marked* thus becoming $f^\sharp$ (often just capitalized: F).

Example 1 *Consider the following TRS $\mathcal{R}$ from [2, Example 1]:*

*Work partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and HA 2006-0007, and by the Generalitat Valenciana under grant GV06/285. Rafael Navarro was partially supported by the Spanish MEC under FPU grant AP2006-02676.

[1] A symbol f is said to be *defined* in a TRS $\mathcal{R}$ if $\mathcal{R}$ contains a rule $f(l_1, \ldots, l_k) \to r$.

```
[1] minus(x,0) -> x
[2] minus(s(x),s(y)) -> minus(x,y)
[3] quot(0,s(y)) -> 0
[4] quot(s(x),s(y)) -> s(quot(minus(x,y),s(y)))
```

This TRS contains three dependency pairs:

```
[5] MINUS(s(x),s(y)) -> MINUS(x,y)
[6] QUOT(s(x),s(y)) -> QUOT(minus(x,y),s(y))
[7] QUOT(s(x),s(y)) -> MINUS(x,y)
```

The dependency pairs conform a new TRS $\mathsf{DP}(\mathcal{R})$ which (together with $\mathcal{R}$) determines the so-called *dependency chains*. The absence of infinite dependency chains characterize termination of $\mathcal{R}$. The dependency pairs can be presented as a *dependency graph* (DG), where the absence of infinite chains can be analyzed by considering the *cycles* in the graph.

Example 2 *Consider the TRS $\mathcal{R}$ in Example 1. There are two cycles in the dependency graph:* $\{5\}$ *and* $\{6\}$.

Basically, given a *cycle* in the dependency graph, we require $l \succeq r$ for all rules in the TRS $\mathcal{R}$, $u \succeq v$ for all dependency pairs in $\mathfrak{C}$ and $u > v$ for *at least one* dependency pair $u \to v \in \mathfrak{C}$. Here, $\succeq$ is a quasi-ordering on terms and $>$ is a *well-founded* ordering.

Example 3 *Consider the TRS $\mathcal{R}$ in Example 1 and the cycle $\mathfrak{C} = \{6\}$ (see Example 2) consisting of the dependency pair*

```
QUOT(s(x),s(y)) -> QUOT(minus(x,y),s(y))
```

In order to prove termination of $\mathcal{R}$ we have to find a reduction pair $(\succeq, >)$ which satisfies the following constraints:

$$\begin{array}{rcl}
\texttt{minus(x,0)} & \succeq & \texttt{x} \\
\texttt{minus(s(x),s(y))} & \succeq & \texttt{minus(x,y)} \\
\texttt{quot(0,s(y))} & \succeq & \texttt{0} \\
\texttt{quot(s(x),s(y))} & \succeq & \texttt{s(quot(minus(x,y),s(y)))} \\
\texttt{QUOT(s(x),s(y))} & > & \texttt{QUOT(minus(x,y),s(y))}
\end{array}$$

Many termination tools (AProVE [7], C*i*ME 2.02 [3], MU-TERM [13], TTT [9],...) use polynomials as a principal ingredient to achieve termination proofs. In this setting, each k-ary symbol $f \in \mathcal{F}$ is given a *parametric* polynomial like, e.g., $a_k x_k + \cdots + a_1 x_1 + a_0$.

Example 4 *Consider the constraints in Example 3. The following (parametric) polynomials are given to the symbols:*

$$\begin{array}{rcl}
[\texttt{0}](x) & = & a_0 \\
[\texttt{s}](X) & = & s_1 X + s_0 \\
[\texttt{minus}](X,Y) & = & m_1 X + m_2 Y + m_0 \\
[\texttt{quot}](X,Y) & = & q_1 X + q_2 Y + q_0 \\
[\texttt{QUOT}](X,Y) & = & q'_1 X + q'_2 Y + q'_0
\end{array}$$

Constraints $s \succeq t$ and $s > t$ are treated as *polynomial constraints* $P_{s,t} \geq 0$ and $P_{s,t} > 0$, respectively, where $P_{s,t} = [s] - [t]$ is the polynomial obtained from terms s and t by interpreting them as polynomials $[s]$ and $[t]$, see [4, 14] for further details.

Example 5 *The first constraint in Example 3 is translated into the polynomial constraint:*

$$(m_1 - 1)x + m_2 a_0 + m_0 \geq 0$$

Variables in terms s and t (e.g., $\mathbf{x}$ in the first constraint in Example 3) become universally quantified numeric variables in polynomial constraints $P_{s,t} \geq 0$ (e.g., x in Example 5). In contrast, the parametric coefficients become *existentially quantified variables* (e.g., a_0, m_0, m_1, and m_2 in Example 5). The use of non-negative numbers as interpretation domains and well-known positiveness criteria like Hong and Jakuš' [10] allows us to center the attention on *solving existential constraints* where all variables correspond to parametric coefficients.

Example 6 *According to [4, 14] and also [10], we have to solve the following (conjuntion of) polynomial constraints:*

$$\begin{array}{ll}
(1) & a_0 m_2 + m_0 \geq 0 \\
(2) & m_1 - 1 \geq 0 \\
(3) & m_1 s_0 + m_2 s_0 \geq 0 \\
(4) & m_1 s_1 - m_1 \geq 0 \\
(5) & m_2 s_1 - m_2 \geq 0 \\
(6) & a_0 q_1 + q_2 s_0 + q_0 - a_0 \geq 0 \\
(7) & q_2 s_1 \geq 0 \\
(8) & q_1 s_0 + q_2 s_0 + q_0 - m_0 q_1 s_1 - q_2 s_0 s_1 - q_0 s_1 - s_0 \geq 0 \\
(9) & q_1 s_1 - m_1 q_1 s_1 \geq 0 \\
(10) & q_2 s_1 - m_2 q_1 s_1 - q_2 s_1 s_1 \geq 0 \\
(11) & q_1' s_0 - m_0 q_1' \geq 0 \\
(12) & q_1' s_1 - m_1 q_1' \geq 0 \\
(13) & -m_2 q_1' \geq 0
\end{array}$$

Note that the variables *which have to be* solved *here are the* coefficients *of the parametric polynomials in Example 4. The previous set of constraints is sound regarding* DG*-based termination proofs (i.e., its satisfaction implies that the cycle is harmless) provided that* all such variables/parametric coefficients take non-negative values*, see [4, 14] for a justification of this claim.*

Now, the termination problem is just a standard *constraint solving* problem which can be treated by using standard algorithms and techniques.

1.1 Solving polynomial constraints in termination provers

Constraints like those showed in Example 6 are expected to be solved on a suitable domain D of *coefficients* because the intended meaning of the variables is to serve as particular coefficients of parametric interpretations like in Example 4. A recent paper by Fuhs et al. proposes the use of SAT techniques for solving (Diophantine) polynomial constraints in termination provers [6]. Fuhs et al.'s benchmarks show that, indeed, using $D = \{0, 1\}$ as the domain for coefficients in polynomial interpretations is already a very powerful option in comparison to bigger domains. The information in Table 1 has been taken from [6]. It corresponds to the benchmarks performed with the new version of AProVE which

implements a SAT-based solver for polynomial constraints (AProVE-SAT). The termination problems come from the 2006 Termination Problem Data Base (TPDB, version 3.2)[2]. 865 examples were considered. Three different ranges for coefficients were considered, corresponding to $N_1 = \{0, 1\}$, $N_2 = \{0, 1, 2\}$, and $N_3 = \{0, 1, 2, 3\}$.

Table 1 shows that AProVE-SAT increments the ratio of solved examples in

Coeff. Range:	1	2	3
# Success	421	431	434
% Success	49,1	49,8	50,2
Time	45.5 s.	91.8 s.	118.6 s.

Table 1: AProVE-SAT benchmarks

$0,7\%$ when using coefficients from N_2 instead of N_1, but the time for achieving the proofs is *duplicated!*

Thus, specifically considering the (small) domain N_1 to obtain an efficient solver on this particular domain still makes sense. In this paper we address this task[3].

2 Polynomial constraints over N_1

In this paper we focus on the case when variables in the considered polynomials range on N_1. Since for all $x \in N_1$ and all $n > 0$ we have $x^n = x$, when considering the representation of a polynomial P, we can *replace* each monomial $\mathsf{m} = cX_1^{\alpha_1} \cdots X_n^{\alpha_n}$ in P by the monomial $\rho(\mathsf{m}) = cX_1^{\beta_1} \cdots X_n^{\beta_n}$ where $\beta_i = 1$ if $\alpha_i \neq 0$ and $\beta_i = 0$ if $\alpha_i = 0$. Then, we obtain a simpler representation $\rho(P)$ of P:

$$\rho(P) = \begin{cases} K \text{ if } P = K \in \mathbb{R} \\ cX_1^{\beta_1} \cdots X_n^{\beta_n} + \rho(Q) \text{ if } P = cX_1^{\alpha_1} \cdots X_n^{\alpha_n} + Q \\ \quad \text{and there is } i, 1 \leq i \leq n, \alpha_i > 0 \end{cases}$$

The following result will be used to justify the correctness (and completeness) of this approach regarding constraint solving over N_1.

Proposition 1 *Let $P \in \mathbb{R}[X_1, \ldots, X_n]$ be a polynomial. For all $x_1, \ldots, x_n \in N_1$, $P(x_1, \ldots, x_n) = \rho(P)(x_1, \ldots, x_n)$.*

Proof. By induction on the number N of monomials with variables in P. If $N = 0$, then P is a constant monomial which is obviously identic to $\rho(P)$. If $N > 0$, let $P = cX_1^{\alpha_1} \cdots X_n^{\alpha_n} + Q$ where Q consists of all monomials in P but $cX_1^{\alpha_1} \cdots X_n^{\alpha_n}$. Since $x_i \in N_1$, we have that $x_i^{\alpha_i} = x_i^{\beta_i}$ for $\beta_i = 1$ if $\alpha_i \neq 0$ and $\beta_i = 0$ if $\alpha_i = 0$. Thus, for all $x_1, \ldots, x_n \in N_1$, $cx_1^{\alpha_1} \cdots x_n^{\alpha_n} = cx_1^{\beta_1} \cdots x_n^{\beta_n}$. By

[2] see `http://www.lri.fr/~marche/tpdb/`

[3] This paper is an extended and revised version of an extended abstract presented during the 9th International Workshop on Termination, WST'07.

I.H., $Q(x_1,\ldots,x_n) = \rho(Q)(x_1,\ldots,x_n)$. Since $P(x_1,\ldots,x_n) = cx_1^{\alpha_1}\cdots x_n^{\alpha_n} + Q = cx_1^{\beta_1}\cdots x_n^{\beta_n} + \rho(Q) = \rho(P)(x_1,\ldots,x_n)$, the conclusion follows. □

According to this, when solving constraints over N_1, we can use $\rho(P) \geq 0$ (or $\rho(P) > 0$) instead of $P \geq 0$ (resp. $P > 0$) without loosing anything.

Example 7 *The polynomial constraint* (10) *in Example 6 would be equivalently transformed into*

$$(10') \quad -m_2 q_1 s_1 \geq 0$$

if we are going to solve it over N_1.

3 SAT-solving for polynomial constraints over N_1

The Boolean satisfiability (SAT) problem is a well-known constraint satisfaction problem with many applications (see [16] for a recent survey). Given a propositional formula φ, the Boolean satisfiability problem posed on φ is to determine whether there exists a boolean assignment (i.e., a mapping from boolean variables into the booleans) under which φ evaluates to true. If this is the case, we say that the formula φ is satisfiable.

Along this paper, we often use 0 and 1 instead of $False$ and $True$, respectively, to denote boolean values. Furthermore, given a finite set $\{X_1,\ldots,X_n\}$ of (boolean) variables where some arbitrary total order is assumed, boolean assignments are represented as sequences $(x_1,\ldots,x_n) \in \{0,1\}^n$ where variable X_i is instantiated by x_i for $1 \leq i \leq n$.

When considering polynomial constraints over N_1, the arithmetics on N_1 become very close to boolean operations when 0 is interpreted as $False$ and 1 as $True$, respectively. In particular, the product of values in N_1 correspond to conjunction. Following this intuition, we have developed a simple encoding of polynomial constraints as propositional formulas.

The translation function τ is given in Figure 2, where Q is a polynomial, c and K are numeric constants (with $c \neq 0$), $\mathcal{X}$ is a set of variables, $\mathsf{rmM}_{\mathcal{X}}(P)$ removes all monomials in P which include *all* variables in $\mathcal{X}$, and $\mathsf{rmV}_{\mathcal{X}}(P)$ removes from P all occurrences of variables in $\mathcal{X}$. The formal definitions are given in Figure 1. The following results are used later. Their proofs are straightforward from the definitions in Figure 1.

Lemma 1 *Let* $P \in \mathbb{R}[X_1,\ldots,X_n]$ *be a polynomial, such that* $P = cX_1^{\alpha_1}\cdots X_n^{\alpha_n} + Q$ *for some* $c \in \mathbb{R}$, $\alpha_i \in \mathbb{N}$ *for* $1 \leq i \leq n$, *and* $Q \in \mathbb{R}[X_1,\ldots,X_n]$. *Let* $\mathcal{X} = \{X_i \mid \alpha_i > 0\}$ *and* $x_1,\ldots,x_n \in N_1$ *be such that* $x_i = 0$ *for at least one* $X_i \in \mathcal{X}$. *Then,* $\mathsf{rmM}_{\mathcal{X}}(P)(x_1,\ldots,x_n) = \mathsf{rmM}_{\mathcal{X}}(Q)(x_1,\ldots,x_n) = Q(x_1,\ldots,x_n)$.

Lemma 2 *Let* $P \in \mathbb{R}[X_1,\ldots,X_n]$ *be a polynomial, such that* $P = cX_1^{\alpha_1}\cdots X_n^{\alpha_n} + Q$ *for some* $c \in \mathbb{R}$, $\alpha_i \in \mathbb{N}$ *for* $1 \leq i \leq n$, *and* $Q \in \mathbb{R}[X_1,\ldots,X_n]$.

$$
\begin{array}{rcl}
\mathsf{rmM}_{\mathcal{X}}(K) & = & K \text{ if } K \text{ is a constant} \\
\mathsf{rmM}_{\mathcal{X}}(cX_1^{\alpha_1}\cdots X_n^{\alpha_n}+P) & = & \left\{\begin{array}{ll} \mathsf{rmM}_{\mathcal{X}}(P) & \text{if } \mathcal{X}\subseteq\{X_1,\ldots,X_n\} \\ cX_1^{\alpha_1}\cdots X_n^{\alpha_n}+\mathsf{rmM}_{\mathcal{X}}(P) & \text{otherwise}\end{array}\right. \\
\mathsf{rmV}_{\mathcal{X}}(K) & = & K \text{ if } K \text{ is a constant} \\
\mathsf{rmV}_{\mathcal{X}}(cX_1^{\alpha_1}\cdots X_n^{\alpha_n}+P) & = & cX_1^{\beta_1}\cdots X_n^{\beta_n}+\mathsf{rmV}_{\mathcal{X}}(P) \\
 & & \text{where, for all } i, 1\le i\le n, \beta_i = \left\{\begin{array}{ll} 0 & \text{if } X_i\in\mathcal{X} \text{ and} \\ \alpha_i & \text{otherwise}\end{array}\right.
\end{array}
$$

Figure 1: Definition of rmM and rmV

$$
\begin{array}{rcl}
\tau(K\ge 0) & = & True, \quad \text{if } K\ge 0 \\
\tau(K\ge 0) & = & False, \quad \text{if } K<0 \\
\tau(K> 0) & = & True, \quad \text{if } K>0 \\
\tau(K> 0) & = & False, \quad \text{if } K\le 0 \\
\tau(cX_1\ldots X_n+Q\ge 0) & = & \left(\left(\bigvee_{1\le i\le n}\neg X_i\right)\wedge\tau(\mathsf{rmM}_{\{X_1,\ldots,X_n\}}(Q)\ge 0)\right)\vee \\
 & & \left(\left(\bigwedge_{1\le i\le n} X_i\right)\wedge\tau(\mathsf{rmV}_{\{X_1,\ldots,X_n\}}(Q)+c\ge 0)\right) \\
\tau(cX_1\ldots X_n+Q> 0) & = & \left(\left(\bigvee_{1\le i\le n}\neg X_i\right)\wedge\tau(\mathsf{rmM}_{\{X_1,\ldots,X_n\}}(Q)> 0)\right)\vee \\
 & & \left(\left(\bigwedge_{1\le i\le n} X_i\right)\wedge\tau(\mathsf{rmV}_{\{X_1,\ldots,X_n\}}(Q)+c> 0)\right) \\
\tau(C\wedge C') & = & \tau(C)\wedge\tau(C')
\end{array}
$$

Figure 2: SAT encoding of polynomial constraints over N_1

Let $\mathcal{X}=\{X_i \mid \alpha_i>0\}$ *and* $x_1,\ldots,x_n\in N_1$ *be such that* $x_i=1$ *whenever* $X_i\in\mathcal{X}$. *Then,* $\mathsf{rmV}_{\mathcal{X}}(Q)(x_1,\ldots,x_n)=Q(x_1,\ldots,x_n)$.

According to the discussion in Section 2 (in particular, Proposition 1), we also assume that we only have to deal with polynomials consisting of monomials like $cX_1\cdots X_n$ (i.e., *without* any power greater than 1).

Example 8 *Consider the constraint* (9) *in Example 6. It is translated into a propositional formula as follows:*

$$
\begin{array}{l}
\tau(q_1s_1-m_1q_1s_1\ge 0) \\
\quad=((\neg q_1\vee\neg s_1)\wedge\tau(0\ge 0))\vee((q_1\wedge s_1)\wedge\tau(-m_1+1\ge 0)) \\
\quad=((\neg q_1\vee\neg s_1)\wedge True)\vee((q_1\wedge s_1)\wedge\tau(-m_1+1\ge 0))
\end{array}
$$

Since we have:

$$
\begin{array}{l}
\tau(-m_1+1\ge 0)=(\neg m_1\wedge\tau(1\ge 0))\vee(m_1\wedge\tau(0\ge 0)) \\
\quad=(\neg m_1\wedge True)\vee(m_1\wedge True) \\
\quad\Leftrightarrow \neg m_1\vee m_1 \\
\quad\Leftrightarrow True
\end{array}
$$

we conclude:

$$
\begin{array}{l}
\tau(q_1s_1-m_1q_1s_1\ge 0) \\
=((\neg q_1\vee\neg s_1)\wedge True)\vee((q_1\wedge s_1)\wedge((\neg m_1\wedge True)\vee(m_1\wedge True))) \\
\Leftrightarrow(\neg q_1\vee\neg s_1)\vee(q_1\wedge s_1) \\
\Leftrightarrow(\neg q_1\vee\neg s_1)\vee\neg(\neg q_1\vee\neg s_1) \\
\Leftrightarrow True
\end{array}
$$

The following result establishes the correctness of our technique.

Theorem 1 (Correctness) *Let $P \in \mathbb{R}[X_1, \ldots, X_n]$ be a polynomial. If $\tau(\rho(P) \geq 0)$ (resp. $\tau(\rho(P) > 0)$) holds for some $x_1, \ldots, x_n \in N_1$, then $P(x_1, \ldots, x_n) \geq 0$ (resp. $P(x_1, \ldots, x_n) > 0$).*

Proof. We give the proof for the 'weak' case; the strict one is analogous. We prove, by induction on the number N of monomials with variables in $\rho(P)$, that whenever $\tau(\rho(P) \geq 0)$ holds for some $x_1, \ldots, x_n \in N_1$, then $\rho(P)(x_1, \ldots, x_n) \geq 0$. By Proposition 1, this implies that $P(x_1, \ldots, x_n) \geq 0$.

If $N = 0$, then $\rho(P)$ is a constant polynomial $\rho(P) = K \in \mathbb{R}$. Thus, $\tau(\rho(P) \geq 0)$ is either $True$ or $False$ depending on the value of K. In particular, if $\tau(\rho(P) \geq 0)$ holds this means that $K \geq 0$. Hence $\rho(P) = K \geq 0$.

If $N > 0$, then we can write $\rho(P) = cX_1 \cdots X_m + Q$ for some $c \in \mathbb{R}$, $0 < m \leq n$, and $Q \in \mathbb{R}[X_1, \ldots, X_n]$. Let $\mathcal{X} = \{X_1, \ldots, X_m\}$. If $\tau(\rho(P) \geq 0)$ holds for the sequence $(x_1, \ldots, x_n) \in N_1^n$ viewed as a propositional assignment, then we have the following (mutually exclusive) cases:

1. $\bigvee_{1 \leq i \leq m} \neg X_i$ holds for the sequence $(x_1, \ldots, x_n)$. Hence, since $\tau(\rho(P) \geq 0)$ holds, by definition of τ we know that $\tau(\mathsf{rmM}_{\mathcal{X}}(Q) \geq 0)$ also holds for $(x_1, \ldots, x_n)$. Since $\mathsf{rmM}_{\mathcal{X}}(Q)$ contains at most as many monomials as Q, by the Induction Hypothesis, we know that $\mathsf{rmM}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) \geq 0$. Since $\bigvee_{1 \leq i \leq m} \neg X_i$ holds for the sequence $(x_1, \ldots, x_n)$, by Lemma 1, $\mathsf{rmM}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$. Thus, $Q(x_1, \ldots, x_n) \geq 0$. Furthermore, under the considered conditions, we have that $cx_1 \cdots x_n = 0$, hence $\rho(P)(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$ and $\rho(P)(x_1, \ldots, x_n) \geq 0$.

2. $\bigwedge_{1 \leq i \leq m} X_i$ holds for the sequence $(x_1, \ldots, x_n)$. Hence, we know that $\tau(\mathsf{rmV}_{\mathcal{X}}(Q) + c \geq 0)$ also holds for the sequence $(x_1, \ldots, x_n)$ and, by the Induction Hypothesis, we can assume that $\mathsf{rmV}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) + c \geq 0$. Since all variables in $\mathcal{X}$ take value 1 in $(x_1, \ldots, x_n)$, by Lemma 2 $\mathsf{rmV}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$. Therefore, $\mathsf{rmV}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) + c = Q(x_1, \ldots, x_n) + c \geq 0$. Since, under the considered conditions, $\rho(P)(x_1, \ldots, x_n) = c + Q(x_1, \ldots, x_n)$, we conclude that $\rho(P)(x_1, \ldots, x_n) \geq 0$.

□

Theorem 2 (Completeness) *Let $P \in \mathbb{R}[X_1, \ldots, X_n]$ be a polynomial and $x_1, \ldots, x_n \in N_1$. If $P(x_1, \ldots, x_n) \geq 0$ (resp. $P(x_1, \ldots, x_n) > 0$), then $\tau(\rho(P)(X_1, \ldots, X_n) \geq 0)$ (resp. $\tau(\rho(P)(X_1, \ldots, X_n) > 0)$) holds for the sequence $(x_1, \ldots, x_n)$ viewed as a truth assignment.*

Proof. Again, we give the proof for the 'weak' case only. Proposition 1 can be used to start with $\rho(P)(x_1, \ldots, x_n) \geq 0$ instead of $P(x_1, \ldots, x_n) \geq 0$. We also proceed by induction on the number N of monomials with variables in $\rho(P)$ to prove that that whenever $\rho(P)(x_1, \ldots, x_n) \geq 0$ for some $x_1, \ldots, x_n \in N_1$, then $\tau(\rho(P) \geq 0)$ holds.

If $N = 0$, then $\rho(P)$ is a constant polynomial $\rho(P) = K \in \mathbb{R}$. Thus, $\rho(P) = K \geq 0$ means that $\tau(\rho(P) \geq 0)$ is $True$ as required.

If $N > 0$, then we write $\rho(P) = cX_1 \cdots X_m + Q$ for some $c \in \mathbb{R}$, $0 < m \leq n$, and $Q \in \mathbb{R}[X_1, \ldots, X_n]$. Let $\mathcal{X} = \{X_1, \ldots, X_m\}$. If $P(x_1, \ldots, x_n) \geq 0$ for some $x_1, \ldots, x_n \in N_1$, then we consider the following (mutually exclusive) cases:

1. $\bigvee_{1 \leq i \leq m} \neg X_i$ holds for the sequence $(x_1, \ldots, x_n)$. Thus, $cx_1 \cdots x_m = 0$ and $\widetilde{\rho}(P)(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n) \geq 0$. By Lemma 1, $Q(x_1, \ldots, x_n) = \mathsf{rmM}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) \geq 0$. Since $\mathsf{rmM}_{\mathcal{X}}(Q)$ does not contain more monomials than Q, by the Induction Hypothesis, $\tau(\mathsf{rmM}_{\mathcal{X}}(Q) \geq 0)$ holds for $(x_1, \ldots, x_n)$ and, by definition of τ, $\tau(\rho(P) \geq 0)$ also holds for $(x_1, \ldots, x_n)$.

2. $\bigwedge_{1 \leq i \leq m} X_i$ holds for the sequence $(x_1, \ldots, x_n)$. Hence, $cx_1 \cdots x_m = c$ and $\widetilde{\rho}(P)(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n) + c \geq 0$. By Lemma 2, we can write $\mathsf{rmV}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$. Therefore, $\mathsf{rmV}_{\mathcal{X}}(Q)(x_1, \ldots, x_n) + c = Q(x_1, \ldots, x_n) + c \geq 0$. By the Induction Hypothesis, $\tau(\mathsf{rmV}_{\mathcal{X}}(Q) + c \geq 0)$ holds for the sequence $(x_1, \ldots, x_n)$ and, by definition of τ, $\tau(\rho(P) \geq 0)$ also holds for $(x_1, \ldots, x_n)$.

□

4 Benchmarks

We have compared the behavior of MU-TERM when different polynomial constraint solving engines are used to prove termination of programs and the domain of coefficients is N_1:

1. MU-TERM-SAT uses the translation of polynomial constraints into propositional formulas described in Section 3. In order to obtain a propositional formula in CNF format, we call an external module implementing the algorithm in [19]. Then, we call MiniSat[4] to obtain a solution.

2. MU-TERM-ApSAT uses an external module implementing the SAT-based constraint solving algorithm described in [6] and implemented as part of AProVE, and which also uses MiniSat for solving the generated propositional constraints.

We have considered the 952 examples in the 'Standard' TRS subcategory of the 2007 Termination Competition[5] which are part of the 2007 Termination Problem Data Base (TPDB, version 4.0)[6]. The tools were executed under OS Linux Ubuntu 4.1.1-13ubuntu5, on a Intel Core 2 CPU at 2.13 GHz and 1 GByte of primary memory. Complete information about all benchmarks in the paper can be found here:

`http://www.dsic.upv.es/~rnavarro/satN1/benchmarks`

[4] `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/MiniSat.html`
[5] `http://www.lri.fr/~marche/termination-competition/2007`
[6] `http://www.lri.fr/~marche/tpdb`

4.1 Global results

Table 2 summarizes the proofs obtained by the different versions of MU-TERM.

	SAT	ApSAT
# YES	334	334
# NO	610	606
# TOs	8	12
YES Av.T.	0.22	1.28
NO Av.T.	0.83	3.79

Table 2: Different solvers for N_1

1. Row '# YES' indicates the number of successful proofs, i.e., the number of TRSs which MU-TERM can prove terminating by using polynomial interpretations over N_1.
2. Row '# NO' indicates the number of unsuccessful proofs, i.e., the number of TRSs that *cannot* be proved terminating by using polynomial interpretations over N_1. Here, we can say 'cannot' due to the completeness of the SAT encodings (Theorem 2 in our case).
3. Finally, row '# TOs' indicates the number of unfinished proofs interrupted by the time-out of 60 seconds.

Rows 'YES Av. T.' and 'NO Av. T.' indicate the average time of successful/unsuccessful proofs (in seconds). All benchmarks were made under the usual 60 seconds time out of the termination competition.

Further details. For the time-out of 60 seconds, both tools succeeded over the same examples (overall 334). Furthermore, the 8 time-outs reported by MU-TERM-SAT also correspond to time-outs of MU-TERM-ApSAT. Hence, whenever MU-TERM-ApSAT stops reporting an unsuccessful proof, MU-TERM-SAT also stops reporting an unsucessful proof.

Regarding speed, MU-TERM-SAT was faster than MU-TERM-ApSAT in 781 (82.0%) of the examples; MU-TERM-ApSAT was faster than MU-TERM-SAT in 138 (14.5%) of the examples. The biggest difference in speed favouring MU-TERM-SAT was of 39.93 seconds. The biggest difference in speed favouring MU-TERM-ApSAT was of 0,96 seconds.

4.2 Different time-outs

Tools for proving termination do not use a *single* technique for proving termination. Termination provers rather proceed stepwise by following some particular sequence of several techniques which are given 'partial' time-outs which are a (small) fraction of the global time-out.

Remark 1 *Nowadays, the termination expert implemented in* MU-TERM *performs the proofs according to a sequence of 10 different techniques among which*

Tool:	MU-TERM-SAT			MU-TERM-ApSAT		
TO	YES	NO	TO	YES	NO	TO
1 s.	325	542	85	233	169	550
10 s.	332	598	22	325	551	76
30 s.	334	607	11	332	595	25
60 s.	334	610	8	334	606	12

Table 3: Different time-outs for N_1

we try different kinds of polynomial interpretations and different bounds for the coefficients. The global time-out is equitatively distributed among the different techniques. Hence, a global time-out of 60 s. amounts at each technique to have at most six seconds to obtain a proof.

Thus, we have also considered the behavior of the solvers when different time-outs (below 60 seconds) are considered. Table 3 shows our results.

5 Analysis of benchmarks

Over the last years, many algorithms, encodings and new variations of classical algorithms have been proposed. Often, the only way to get some evidence about their relative power or usefulness is performing some kind of empirical evaluation. Unfortunately, good evaluations are not done nearly enough. In order to compare two or more algorithms in a clear way, we should apply an appropriate statistical test. In that case, we could identify differences between the considered algorithms or techniques which are meaningful enough.

In order to apply the correct statistical test, we are going to discuss which types of statistical procedures are more suitable in our experiments. Following [5], we are going to see how to statistically compare two or more algorithms over multiple datasets. The common way to compare two algorithms is by means of the *Paired t-test*, which checks whether the means of two normally distributed interval variables differ from one another. Hence it is necessary to check that data are distributed normally (as well as other assumptions [5]). We do not have any clue that our data satisfy these assumptions. Then using this kind of test in a naïve way could be misleading.

Hence we should find a statistical test which is not based on any assumption, that is a *non-parametric* test. Wilcoxon's *signed rank sum test* [21] is the non-parametric version of the Paired t-test. We can use this test when we do not assume that the difference between the two variables is *interval and normally distributed*, but we *do* assume that the difference is ordinal (i.e., the differences between the individual results are meaninfully ordered). If the differences between variables are not ordinal but they can be classified as positive or negative, we should better consider a *sign test* [1] instead.

It is important to notice that, when the assumptions of the parametric method are met, they are more powerful than the non-parametric version. On the other hand, when the assumptions are violated, the non-parametric ones can be even more powerful.

In our experiments, we are going to deal with two different variables:

TO	60 s.	30 s.	10 s.	1 s.
# Proved	1.000	0.500	0.016	0.000

Table 4: Significance values (Sign test) for MU-TERM-ApSAT - MU-TERM-SAT

TO	60 s.	30 s.	10 s.	1 s.
# Proved	1.000	0.157	0.008	0.000
Time	0.000	0.000	0.000	0.000

Table 5: Significance values (Wilcoxon's test) for MU-TERM-ApSAT - MU-TERM-SAT

- "proved" represents if each problem has been solved (either positively or negatively); otherwise we do not know.
- "time" represents the time which necessary to compute the answer.

Since the difference over the "proved" variable is not ordinal, we are going to check it by means of a sign test (although Wilcoxon's signed rank sum test produces the same result, as we are going to see). For the difference over the "time" we are going to use the Wilcoxon's signed rank sum test.

As usual, statistical tests work trying to reject the *null-hypothesis* that assumes that the algorithms perform equally well. To *reject* that assumption (i.e., to statistically establish that they are different) we take into account the *significance value (sig)* computed for the statistical test. Whenever $sig < 0.05$, we can say that the algorithms actually differ (from a statistical point of view).

5.1 Evaluation of the experimental comparison

Our statistical analysis has been performed by using the statistical suite SPSS 14.0 [8]. The results are presented in Tables 4 and 5. They show that we have statistical evidence of the following facts:

1. MU-TERM-SAT is more powerful than MU-TERM-ApSAT when solving termination problems from a timeout less than or equal to 10 seconds.
2. MU-TERM-SAT is always faster than MU-TERM-ApSAT. Furthermore, MU-TERM-SAT is $\frac{1.28}{0.22} = 5.8$ times *faster* than MU-TERM-ApSAT in giving a *positive* answer and $\frac{3.79}{0.83} = 4.6$ times *faster* in giving a *negative* answer (in the average).

Thus, according to our experiments and the statistical analysis above, we conclude that our encoding of polynomial constraints over N_1 as propositional formulas and the use of state-of-the-art SAT solvers (e.g., MiniSat) is the best way to deal with such kind of constraints when N_1 is the domain of coefficients. Actually, this is crucial when small time-outs are used (as required by the 'experts' of termination tools).

Remark 2 *Note that, although* MU-TERM-*SAT directly implements the encoding described in Section 3 it still performs* two *calls to external tools (the CNF*

converter and MiniSat). Similarly, MU-TERM-*ApSAT actually performs two external calls (one to AProVE's SAT-solving engine which then calls to MiniSat). In this sense, we believe that comparing our SAT-encoding and Fuhs et al.'s one through* MU-TERM-*SAT and* MU-TERM-*ApSAT is fair.*

6 Conclusions

We have developed an encoding of polynomial constraints over $N_1 = \{0, 1\}$ as propositional formulas. Our encoding is correct and complete for solving polynomial constraints over N_1 (Theorems 1 and 2). We have compared our encoding with the recent (more general) encoding by Fuhs et al. [6] when applied to solve such polynomial constraints. Regarding their success when proving termination (i.e., the number of times that they give a positive or negative answer to the termination problem), we have provided statistical evidence that (within the conditions of our benchmarks) our encoding is, in general, equally powerful and, actually, *strictly more powerful* for time-outs below than or equal to 10 seconds. Furthermore, we have also provided statistical evidence that our encoding is *always* faster (around five times, in the average) than Fuhs et al.'s one.

Future work

An obvious question arising at this point is the following: what happens with more general domains of coefficients like $N_2 = \{0, 1, 2\}$, etc. Indeed, although considering specific encodings for polynomial constraints over such domains could be interesting, many of the good properties that are behind the simplicity and good behavior of our encoding for N_1 are not there for N_2, ... Still, as discussed in Section 1.1 the domain N_1 is important enough for termination tools as to justify our particular research. Furthermore, when considering the use of constraint solving algorithms for termination tools, an essential issue is *incrementality*: constraint solvers in termination tools usually consider not only one but several different domains of coefficients when trying to solve a given polynomial (or even symbolic) constraint. Such domains are often related. For instance, MU-TERM first try is for N_1 (using the encoding presented in this paper). If the constraint cannot be solved, then N_2 is considered, etc. But $N_1 \subseteq N_2$ and this means that, after failing with N_1, part of the search space for N_2 is known to be useless in advance. Current constraint solving algorithms implemented in termination tools like [4, 6, 15] do not take benefit from this fact. We plan to use our SAT-encoding for N_1 as a basis for incremental constraint-solving algorithms.

Another interesting aspect is that the SAT-encodings discussed here (both our new proposal in Section 3 and also [6]) do not take into account more sophisticated SAT frameworks like SMT (SAT modulo theories [17]) which seems to be a natural choice for polynomial constraints.

And, beyond Diophantine constraint solving, coefficients for solving poly-

nomial constraints could be taken from any subset of *real algebraic numbers*[7] [15]. For instance, $\{0, \frac{1}{2}, 1, 2\}$ is the default domain for coefficients in MU-TERM, and the CSP-like algorithm introduced in [15] is used for solving the polynomial constraints over such domain. An interesting open problem is how to encode polynomial constraints over such more general domains using SAT/SMT techniques. These are interesting subjects for future work.

Acknowledgements. We thank Jürgen Giesl and Peter Schneider-Kamp for providing an executable version of the SAT-based polynomial constraint solving engine implemented in AProVE-SAT [6].

References

[1] H. Abdi. Binomial Distribution: Binomial and Sign test. Salkind (Ed.): Encyclopedia of Measurement and Statistics. Thousand Oaks (CA): Sage.

[2] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236:133-178, 2000.

[3] E. Contejean and C. Marché, B. Monate and X. Urbain. Proving termination of rewriting with C*i*ME. In A. Rubio, editor, *Proc. of 6th International Workshop on Termination, WST'03*, pages 71-73, Technical Report DSIC II/15/03, Valencia, Spain, 2003. Available at `http://cime.lri.fr`.

[4] E. Contejean, C. Marché, A.-P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. Journal of Automated Reasoning, 34(4):325-363, 2006.

[5] J. Demsar. Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7:1-30, 2006.

[6] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT Solving for Termination Analysis with Polynomial Interpretations. In J. Marques-Silva and K.A. Sakallah, editors, *Proc. of the 10th International Conference on Theory and Applications of Satisfiability Testing, SAT'07*, LNCS 4501:340-354, Springer-Verlag, Berlin, 2007.

[7] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281-286, Springer-Verlag, Berlin, 2006. Available at `http://www-i2.informatik.rwth-aachen.de/AProVE`.

[8] S. B. Green, N. J. Salkind. Using SPSS for Windows and Macintosh: Analyzing and Understanding Data (4th Edition). Prentice Hall, 2004.

[9] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474-511, 2007.

[10] H. Hong and D. Jakuš. Testing Positiveness of Polynomials. *Journal of Automated Reasoning* 21:23-38, 1998.

[11] R. L. Iman and J. M. Davenport. Approximations of the critical region of the Friedman statistic. Communications in Statistics, pages 571-595, 1980.

[7] A real number $x \in \mathbb{R}$ is said to be *algebraic* if it satisfies an equation $x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$, of finite degree n where $a_i \in \mathbb{Q}$ for $0 \leq i \leq n-1$.

[12] D.S. Lankford. On proving term rewriting systems are noetherian. Technical Report, Louisiana Technological University, Ruston, LA, 1979.

[13] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting In V. van Oostrom, editor, *Proc. of 15h International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004.

[14] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547-586, 2005.

[15] S. Lucas. Practical use of polynomials over the reals in proofs of termination. In *Proc. of 9th International Symposium on Principles and Practice of Declarative Programming, PPDP'07*, pages 39-50, ACM Press, 2007.

[16] M.R. Prasad, A. Biere, and A. Gupta. A survey of recent advances in SAT-based formal verification. International Journal on Software Tools for Technology Transfer, 7(2):156-173, 2005.

[17] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Longemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937-977, 2006.

[18] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.

[19] D. Sheridan. The Optimality of a Fast CNF Conversion and its use with SAT. In *Proc. of 7th International Conference on Theory and Applications of Satisfiability Testing, SAT'04*, 2004.

[20] TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003.

[21] F. Wilcoxon. Individual comparisons by ranking methods. Biometrics, 1:80-83, 1945.