Genetic Algorithms for Finding Polynomial Orderings

Jürgen Giesl, Fernando Esponda, and Stephanie Forrest

Computer Science Department
University of New Mexico, Albuquerque, NM 87131, USA
{giesl|fesponda|forrest}@cs.unm.edu

Abstract. Polynomial orderings are a well-known method to prove termination of term rewriting systems. However, for an automation of this method, the crucial point is to find suitable coefficients by machine. We present a novel approach for solving this problem by applying genetic algorithms.

1 Introduction

One essential property of many programs is their termination, that is, do they terminate or not? To study termination problems, many researchers consider *term rewriting systems*, since this is essentially the most fundamental and basic functional programming language.

1.1 Term Rewriting Systems

A term rewriting system (TRS) is a set of equations between terms. However, these equations are oriented and therefore, one writes an arrow $\to$ instead of the equality symbol $=$. These oriented equations are called *rules*. As an example consider the following simple TRS consisting of two rules:

$$\mathsf{plus}(x, \mathsf{0}) \to x$$
$$\mathsf{plus}(x, \mathsf{s}(y)) \to \mathsf{s}(\mathsf{plus}(x, y))$$

The rules of a TRS can be used to *evaluate* or to *compute* expressions. For example, the first rule states that every term starting with plus whose second argument is $\mathsf{0}$ can be replaced by its first argument. The evaluation of terms works by replacing subterms by new subterms according to the rules.

The TRS above implements an addition algorithm for natural numbers. Here, numbers are represented by the function symbols $\mathsf{0}$ and s, where s stands for the successor function. So the number 0 corresponds to the term $\mathsf{0}$, the number 1 is represented as the term $\mathsf{s}(\mathsf{0})$, the number 2 is represented as $\mathsf{s}(\mathsf{s}(\mathsf{0}))$, etc.

Formally, a term s *rewrites* to a term t w.r.t. a TRS $\mathcal{R}$ (denoted $s \to_{\mathcal{R}} t$) iff there exists a subterm of s which is an instance of the left-hand side of a rule and if t results from s by replacing this subterm by the corresponding instance of the right-hand side of this rule. For a detailed introduction to term rewriting systems see e.g. [2].

In this way, the above TRS can be used for computations. For example, in order to compute "1 + 2" we would have to evaluate the term $\mathsf{plus}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{s}(\mathsf{0})))$. We obtain

$$\mathsf{plus}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{s}(\mathsf{0}))) \to_{\mathcal{R}} \mathsf{s}(\mathsf{plus}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(\mathsf{plus}(\mathsf{s}(\mathsf{0})), \mathsf{0})) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))).$$

Hence, the result of "1 + 2" is 3.

A term rewriting system $\mathcal{R}$ is *terminating* if there does not exist any infinite rewrite sequence. While in general this problem is undecidable, several methods for proving termination have been developed. For an overview on classical methods see for example [7, 15] and for recent, more powerful developments see [1]. In the following, we will only investigate whether all rewrite sequences of *ground terms* (i.e., terms without variables) are finite. It is well known (and straightforward) that this is already enough for proving termination.

1.2 Polynomial Orderings

One of the classical approaches for termination proofs is the use of *polynomial orderings* [12]. Several methods for generating polynomial orderings automatically have been suggested, which also led to the implementation of several systems for this task [4, 5, 8, 11, 14] . Methods for generating polynomial orderings are also very useful in more modern and powerful termination techniques, where such orderings can be used as "base orderings" (see [1, 5]).

The classical approach to termination proofs is to find a suitable ordering such that terms always decrease w.r.t. this ordering whenever a rewrite rule is applied. In other words, one has to find an ordering $\succ$ such that $s \to_{\mathcal{R}} t$ implies $s \succ t$ for all ground terms s and t. If this ordering is *well founded* (i.e., if there exist no infinite decreasing sequences $t_0 \succ t_1 \succ t_2 \succ \ldots$ w.r.t. this ordering), then this is sufficient to prove termination of $\mathcal{R}$.

In order to find such an ordering, it is of course impractical to investigate all possible (infinitely many) rewrite sequences. Instead, one should rather look at the rules and find an ordering such that the left-hand side of each rule is greater than the corresponding right-hand side (for all instantiations of the variables with ground terms). In this case, we say the ordering is *compatible* with the TRS. In our example, this means that we have to find an ordering $\succ$ satisfying

$$\begin{aligned} \mathsf{plus}(t, \mathsf{0}) &\succ t \\ \mathsf{plus}(t_1, \mathsf{s}(t_2)) &\succ \mathsf{s}(\mathsf{plus}(t_1, t_2)) \end{aligned}$$

for all ground terms t, t_1, t_2.

However, in order to ensure that with every rewrite step the terms become smaller w.r.t. $\succ$ we have to demand something more. Since rewrite steps can be applied on subterms, we must demand that the ordering $\succ$ is *monotonic*, i.e.

$$s \succ t \quad \text{must imply} \quad f(\ldots s \ldots) \succ f(\ldots t \ldots)$$

for all function symbols f.

One idea of defining such orderings is the following: We define a mapping POL which maps ground terms to natural numbers. For example, we could define POL inductively as follows:

$$
\begin{aligned}
POL(\mathsf{0}) &= 0 \\
POL(\mathsf{s}(t)) &= POL(t) + 1 \\
POL(\mathsf{plus}(t_1, t_2)) &= POL(t_1) + 2 * POL(t_2) + 1
\end{aligned}
$$

Since the number corresponding to a term is composed from the numbers corresponding to its subterms by using a polynomial function, such a mapping is called a *polynomial interpretation*. The corresponding *polynomial ordering* $\succ_{POL}$ is simply defined by comparing terms according to the numbers they are mapped to, i.e., $s \succ_{POL} t$ iff $POL(s) > POL(t)$ (where "$>$" is the usual greater-relation on naturals).

Obviously, every polynomial ordering is well founded, since $t_0 \succ_{POL} t_1 \succ_{POL} \ldots$ would mean that there is an infinite decreasing sequence of natural numbers (which is impossible). Hence, if a polynomial ordering is monotonic and compatible with the TRS, then termination is proved. It will turn out that our polynomial ordering indeed satisfies these two requirements for the plus-TRS. Hence, this TRS is terminating.

In order to find out whether our polynomial ordering is compatible with our TRS, one has to check the following constraints for all ground terms t, t_1, t_2:

$$
\begin{aligned}
POL(\mathsf{plus}(t, \mathsf{0})) &> POL(t) \\
POL(\mathsf{plus}(t_1, \mathsf{s}(t_2))) &> POL(\mathsf{s}(\mathsf{plus}(t_1, t_2)))
\end{aligned}
$$

In other words, this means

$$
\begin{aligned}
POL(t) + 2 * 0 + 1 &> POL(t) \\
POL(t_1) + 2 * (POL(t_2) + 1) + 1 &> POL(t_1) + 2 * POL(t_2) + 1 + 1
\end{aligned}
$$

which is obviously true. Monotonicity of our polynomial ordering is also straightforward from the fact that making $POL(t)$ greater also increases the value of $POL(\mathsf{s}(t))$ and that making $POL(t_1)$ or $POL(t_2)$ greater increases the value of $POL(\mathsf{plus}(t_1, t_2))$.

1.3 Generating Polynomial Orderings

Of course, in general the question is how can one find suitable polynomial orderings automatically for any given TRS. We follow the approach of [8] to transform this problem into a constraint satisfaction problem [1] , which in turn refines an earlier suggestion from [12].

[1] This approach is more powerful and much simpler than the ones of [4, 14], see [8, 11].

For each TRS $\mathcal{R}$ our aim is to synthesize a polynomial interpretation POL such that $\succ_{POL}$ is compatible with $\mathcal{R}$. For that purpose one has to determine the inductive definition of POL for terms built with the different function symbols.

Of course, first one has to choose the degrees of the polynomial functions used to compose the numbers corresponding to the subterms in order to obtain the number corresponding to the whole term. If we decide on polynomials of degree 1 (i.e., linear polynomials), then any such polynomial interpretation has the form

$$\begin{aligned} POL(\mathsf{0}) &= n \\ POL(\mathsf{s}(t)) &= s_1 * POL(t) + s_0 \\ POL(\mathsf{plus}(t_1, t_2)) &= p_1 * POL(t_1) + p_2 * POL(t_2) + p_0 \end{aligned}$$

The question is how to find suitable coefficients $n, s_0, s_1, p_0, p_1, p_2$ such that the resulting polynomial ordering is monotonic and compatible with the TRS.

Compatibility with the TRS means

$$\begin{aligned} POL(\mathsf{plus}(t, \mathsf{0})) &> POL(t) \\ POL(\mathsf{plus}(t_1, \mathsf{s}(t_2))) &> POL(\mathsf{s}(\mathsf{plus}(t_1, t_2))) \end{aligned}$$

or in other words

$$\begin{aligned} p_1 * POL(t) + p_2 * n + p_0 &> POL(t) \\ p_1 * POL(t_1) + p_2 * (s_1 * POL(t_2) + s_0) + p_0 &> \\ s_1 * (p_1 * POL(t_1) + p_2 * POL(t_2) + p_0) + s_0 \end{aligned}$$

for all ground terms t, t_1, t_2. Of course, this can be re-formulated as

$$(p_1 - 1) * POL(t) + p_2 * n + p_0 > 0 \tag{1}$$

$$(1 - s_1) * p_0 + (p_2 - 1) * s_0 + (1 - s_1) * p_1 * POL(t_1) > 0 \tag{2}$$

for all ground terms t, t_1, t_2.

In order to simplify these constraints further later on, we will introduce a new variable μ and we will add constraints which make sure that μ is smaller than or equal to the minimal number that a ground term is ever mapped to.

$$n - \mu \geq 0 \tag{3}$$

$$s_1 * POL(t) + s_0 - \mu \geq 0 \tag{4}$$

$$p_1 * POL(t_1) + p_2 * POL(t_2) + p_0 - \mu \geq 0 \tag{5}$$

Now we can simplify the constraints (1) - (5) in the following way. We know that for any ground term t, $POL(t)$ is a number greater than or equal to μ. Hence, instead of demanding (1) for all numbers $POL(t)$ corresponding to ground terms, it is sufficient to demand that (1) holds if $POL(t)$ is replaced by μ and that the polynomial on the left-hand side of (1) is not decreasing when

$POL(t)$ is increasing. In other words, the partial derivative of this polynomial w.r.t. $POL(t)$ should be non-negative. Hence, (1) can be replaced by

$$(p_1 - 1) * \mu + p_2 * n + p_0 > 0 \tag{6}$$
$$p_1 - 1 \geq 0 \tag{7}$$

Note that these constraints are much simpler than (1). In (1) we had to find coefficients p_0, p_1, p_2, n such that the constraint is satisfied *for all* numbers $POL(t)$. In contrast, for (6) and (7) we simply have to find coefficients p_0, p_1, p_2, n, μ satisfying these two inequalities. Thus, for a particular choice of p_0, p_1, p_2, n, μ it is straightforward to check whether (6) and (7) are satisfied, whereas it is not straightforward (and in fact, undecidable in general) to check whether (1) is satisfied.

By applying the same transformation on (2) - (5) we obtain the additional constraints

$$(1 - s_1) * p_0 + (p_2 - 1) * s_0 + (1 - s_1) * p_1 * \mu > 0 \tag{8}$$
$$(1 - s_1) * p_1 \geq 0 \tag{9}$$
$$n - \mu \geq 0 \tag{10}$$
$$(s_1 - 1) * \mu + s_0 \geq 0 \tag{11}$$
$$s_1 \geq 0 \tag{12}$$
$$(p_1 + p_2 - 1) * \mu + p_0 \geq 0 \tag{13}$$
$$p_1 \geq 0 \tag{14}$$
$$p_2 \geq 0 \tag{15}$$

Obviously, the constraints (12), (14), and (15) are always satisfied if we restrict ourselves to natural numbers. But in fact, this method also allows the use of integer coefficients (and there is an extension in [8] where one can even use real coefficients).

In order to guarantee monotonicity, one simply has to demand that the partial derivatives of $POL(\mathsf{s}(t))$ w.r.t. $POL(t)$ and of $POL(\mathsf{plus}(t_1, t_2))$ w.r.t. $POL(t_1)$ and $POL(t_2)$ are greater than 0.

$$s_1 > 0 \tag{16}$$
$$p_1 > 0 \tag{17}$$
$$p_2 > 0 \tag{18}$$

In this way, one has the following transformation procedure: Given a TRS and given the desired degree of the polynomial interpretation one can automatically generate a set of constraints containing the "variable coefficients" of the polynomial interpretation. If there exist numbers satisfying these constraints, then the TRS is terminating.

In our example one has to find numbers $n, s_0, s_1, p_0, p_1, p_2, \mu$ satisfying the constraints (6) - (18). For example, the following numbers would be a solution:

$$n = 0$$

$$
\begin{aligned}
s_0 &= 1\\
s_1 &= 1\\
p_0 &= 1\\
p_1 &= 1\\
p_2 &= 2\\
\mu &= 0
\end{aligned}
$$

In fact, with this choice we would obtain the polynomial interpretation from the beginning of this section.

For further details on this transformation procedure we refer to [8]. This procedure has been implemented in the POLO system for automated termination proofs of TRSs [10].

2 Using Genetic Algorithms to find Polynomial Orderings

All previous techniques for synthesizing polynomial orderings first generate inequalities from the TRS under consideration which ensure compatibility and then they perform certain algebraic transformations (like the one sketched in the previous section). But finally one has to search for a suitable choice of coefficients satisfying the resulting constraints. Here, in most cases, the previous systems only used a trial-and-error approach by simply trying combinations of the numbers 0, 1, and 2. (In [8, 10] one can alternatively also use a simplified variant of Collins' decision algorithm [6], but this turns out to be too inefficient in many cases.)

The aim of the present paper is to develop a more sophisticated approach which is more powerful than just trying the numbers 0, 1, and 2 and more efficient than using decision procedures from computer algebra. For that purpose we make use of genetic algorithms [17][18].

2.1 The Problem and its Representation

Essentially, the problem sketched in the previous section can be re-stated as follows. We have a set of inequalities (or constraints)

$$
\begin{aligned}
P_1[n_1, \ldots, n_k] &\geq 0\\
&\vdots\\
P_m[n_1, \ldots, n_k] &\geq 0,
\end{aligned}
$$

where $P_i[n_1, \ldots, n_k]$ is a polynomial with integer coefficients over the variables $n_1, \ldots, n_k$. (Strict inequalities $P_i[n_1, \ldots, n_k] > 0$ can easily be turned into non-strict inequalities by replacing them by $P_i[n_1, \ldots, n_k] - 1 \geq 0$.) The goal is to find integer numbers $n_1, \ldots, n_k$ such that these inequalities are valid. However, since there are hardly any TRSs whose termination proofs require the use of non-negative coefficients, in order to decrease the search space, we will restrict ourselves only to solutions where $n_1, \ldots, n_k$ are non-negative. (As mentioned before, a variant of this problem can also be formulated where $n_1, \ldots, n_k$ are allowed to be real numbers.)

The aim here is not to find a particularly "good" solution, but to find any solution satisfying the constraints. As soon as one solution is found, one can stop the search process.

At the beginning of the search procedure, the user has to fix the possible range of the numbers $n_1, \ldots, n_k$. As mentioned in the previous section, most implementations only search for solutions in the interval $[0, 2]$. In our approach, the user is allowed to choose any interval $[0, 2^q - 1]$. Then a particular solution $n_1, \ldots, n_k$ is represented as a bit string of $q * k$ bits. The first q bits represent the value of n_1, the second q bits represent the value of n_2, etc.

For the success of genetic algorithms, it is important that bit positions with a strong logical connection occur near to each other in the representation chosen. Obviously, the q bits representing the value of some n_i are strongly inter-related, and indeed they occur next to each other in our representation. But moreover, the coefficients of the polynomial interpretation of one particular function also have a strong relation to each other. For instance, in the plus-example of the previous section, the values of p_0, p_1, and p_2 are strongly related to each other (more strongly than they are to n or to s_0 and s_1). Thus, in the representation, coefficients of one function should be represented adjacent to each other. Finally, function symbols which occur in the same rule (or the same inequality) also have more connection to each other than function symbols which do not. Thus, if one would add two more rules

$$\begin{aligned} \mathsf{append}(\mathsf{nil}, y) &\to y \\ \mathsf{append}(\mathsf{add}(n, x), y) &\to \mathsf{add}(n, \mathsf{append}(x, y)) \end{aligned}$$

to the plus-TRS, then the coefficients corresponding to the function symbols 0, s, and plus, should be adjacent and the coefficients corresponding to nil, add, and append should be adjacent, too. In other words, if possible, then one should keep the coefficients of those function symbols near to each other which occur in the same parts of the TRS.

2.2 The Fitness Function and the Genetic Algorithm

The crucial point in developing a successful genetic algorithm is the choice of a suitable fitness function. In other words, given a particular sequence of numbers $n_1, \ldots, n_k$ (represented as a bit string), one has to find a measure to decide how "good" this sequence is.

We define the fitness $f(n_1, \ldots, n_k)$ of such a sequence by adding up the values of those polynomials $P_i[n_1, \ldots, n_k]$ which are still below 0. Thus, the fitness function is always less than or equal to 0. As soon as it is 0, we have reached a solution and the algorithm stops.

By defining the fitness function in this way, only inequalities which are still violated contribute to the fitness function. However, a solution where the inequalities are only violated a little is considered better than a solution where the polynomials $P_i[n_1, \ldots, n_k]$ are extremely small negative numbers. In this way, individuals which are only "a little bit false" can point into the direction of a valid solution.

It is important not to take those polynomials $P_i[n_1, \ldots, n_k]$ into account which are already positive. The reason is that otherwise, the algorithm would generate sequences $n_1, \ldots, n_k$ where a few $P_i[n_1, \ldots, n_k]$ are extremely high, but where others are still negative. Now our genetic algorithm works as follows:

1. **Initialization**: Generate the initial population randomly(i.e., a set of arbitrary $q * k$ bit strings).
2. **Evaluation**: Compute the fitness of each individual.
3. **Selection**: For this application we used binary tourney selection which works as follows: Pick a pair of individuals from the population at random, keep the one with higher fitness, replace them in the population and then select its partner for crossover in a similar way (binary tourney selection).
4. **Crossover**: Perform crossover between selected individuals using one point crossover and ensuring that the crossover point falls between the representation of n_j and n_{j+1}.
5. **Mutation**: Flip some bits in the population with low probability.
6. **Repeat**: If the fitness of some individual is 0 or a maximum number of evaluations has been reached, then stop. Otherwise, go to Step 2.

2.3 Experimental Results

We tested the GA on a set of examples taken from [10]. The GA was run with crossover and mutation probabilities of 0.8 and 0.03 respectively. Each experiment consists of 60 runs using a population of 10 individuals for a maximum of 100 generations, yielding a total of 1000 function evaluations per run. Additionally, we include the results obtained using the Generate & Test (G&T) method and the incomplete Collins algorithm as implemented in the POLO system [10], as well as those from a random mutation hill climber [9] executing for 1000 steps. Table 1 reports the number of successful runs that obtained satisfactory values for the polynomial's coefficients as well as the mean and standard deviations of the number of evaluations per experiment. Note that the Generate & Test and Collins algorithms yield the same results each time they are run. The constraints for each TRS where generated using the simple-mixed polynomial option of the POLO package.

TRS	GA			RMHC			G&T	Collins		
	s-runs	$\overline{x}$	σ	s-runs	$\overline{x}$	σ	s-runs	s-runs	range	var
Boolean Ring	37	577.50	400.59	43	374.26	402.70	60	60	[0,2]	25
Associativity	60	10.0	0.0	58	35.71	180.61	60	60	[0,2]	7
Reverse	36	589.33	374.61	28	570.13	464.40	60	60	[0,2]	17
Symb.Differentiatn	60	237.66	178.21	4	938.06	233.68	0	0	[0,2]	23
Fibonacci Group	4	976.16	98.15	0	1000.0	0.0	0	0	[0,4]	10
Taussky Group	60	71.0	89.49	38	406.35	456.62	0	0	[0,2]	17

Table 1. Number of successes (s-runs) out of 60 runs, the average number of evaluations and standard deviation for each experiment, the range of possible variable assignment values, and the number of variables in each set of constraints.

3 Discussion

The GA seems to be a plausible approach for solving the polynomial orderings problem. The stochastic nature of the GA is immediately apparent from the fact that not all runs for a particular experiment proved successful. In some cases there is a major incidence of success form the other three methods, however, only the GA got answers for all the TRSs, suggesting that it is a more robust approach for trying to solve this kind of constraint satisfaction problem. Nevertheless, it is important to point out that the examples presented in this paper are not of particular difficulty and it would be worthwhile to test the performance of the GA on harder instances, which in turn may require the use of more sophisticated operators and bigger population sizes.

A Examples

This appendix contains a collection of examples for termination proofs with polynomial orderings.

A.1 plus

The first example is the running example plus from the text.

$$\begin{aligned}
\mathsf{plus}(x, \mathsf{0}) &\to x \\
\mathsf{plus}(x, \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{plus}(x, y))
\end{aligned}$$

As explained in the text, for its termination proof one has to find numbers $n, s_0, s_1, p_0, p_1, p_2, \mu$ satisfying the following constraints:

$$\begin{aligned}
(p_1 - 1) * \mu + p_2 * n + p_0 - 1 \geq 0 \\
p_1 - 1 \geq 0
\end{aligned}$$

$$
\begin{aligned}
(1-s_1)*p_0+(p_2-1)*s_0+(1-s_1)*p_1*\mu-1 &\geq 0\\
(1-s_1)*p_1 &\geq 0\\
n-\mu &\geq 0\\
(s_1-1)*\mu+s_0 &\geq 0\\
s_1 &\geq 0\\
(p_1+p_2-1)*\mu+p_0 &\geq 0\\
p_1 &\geq 0\\
p_2 &\geq 0\\
s_1-1 &\geq 0\\
p_2-1 &\geq 0
\end{aligned}
$$

A solution is

$$
\begin{aligned}
n &= 0\\
s_0 &= 1\\
s_1 &= 1\\
p_0 &= 1\\
p_1 &= 1\\
p_2 &= 2\\
\mu &= 0
\end{aligned}
$$

This corresponds to the polynomial interpretation

$$
\begin{aligned}
POL(\mathsf{0}) &= 0\\
POL(\mathsf{s}(t)) &= POL(t)+1\\
POL(\mathsf{plus}(t_1,t_2)) &= POL(t_1)+2*POL(t_2)+1.
\end{aligned}
$$

A.2 Endomorphism and Associativity

The next example is from [3] and [4][2].

$$\mathsf{comp}(\mathsf{comp}(x,y),z) \to \mathsf{comp}(x,\mathsf{comp}(y,z)) \tag{19}$$
$$\mathsf{comp}(\mathsf{map}(x),\mathsf{map}(y)) \to \mathsf{map}(\mathsf{comp}(x,y)) \tag{20}$$
$$\mathsf{comp}(\mathsf{map}(x),\mathsf{comp}(\mathsf{map}(y),z)) \to \mathsf{comp}(\mathsf{map}(\mathsf{comp}(x,y)),z) \tag{21}$$

[2] This example comes from the area of functional programming. The intended meaning of comp is composition, variables represent functions and map is a *mapcar*-like operator (whose result is a function).

Here, we use an approach with a *simple-mixed* polynomial interpretation,[3] as recommended by Steinbach [13]. More precisely, we search for a polynomial interpretation of the following form:

$$\begin{aligned} POL(\mathsf{map}(t)) &= m_0 + m_1 POL(t) \\ POL(\mathsf{comp}(t_1, t_2)) &= c_0 + c_1 POL(t_1) + c_2 POL(t_2) + c_3 POL(t_1) POL(t_2) \end{aligned}$$

To ensure compatibility of the polynomial ordering with the TRS, initially we obtain the following inequalities for all ground terms t_1, t_2, t_3:

$$\begin{aligned} &(c_1 - c_2)c_0 \\ &+(c_1^2 - c_1 - c_0c_3)POL(t_1) \\ &+(c_2 - c_2^2 + c_0c_3)POL(t_3) \\ &+(c_1 - c_2)c_3POL(t_1)POL(t_3) > 0 \\ &(1 - m_1)c_0 + (c_1 + c_2 - 1)m_0 + c_3m_0^2 \\ &+c_3m_0m_1POL(t_1) + c_3m_0m_1POL(t_2) + (m_1^2 - m_1)c_3POL(t_1)POL(t_2) > 0 \\ &c_0c_2 + (c_1c_2 + c_0c_3)m_0 + (c_3m_0^2 - c_0m_1)c_1 \\ &+(c_1 - c_1^2 + c_0c_3 + c_1c_3m_0)m_1POL(t_1) \\ &+c_1c_3m_0m_1POL(t_2) \\ &+(c_2^2 + (2c_2c_3 - c_3)m_0 + c_3^2m_0^2 - c_2 - c_0c_3m_1)POL(t_3) \\ &+(m_1^2 - m_1)c_1c_3POL(t_1)POL(t_2) \\ &+(c_2c_3 + c_3^2m_0 - c_1c_3)m_1POL(t_1)POL(t_3) \\ &+c_3^2m_0m_1POL(t_2)POL(t_3) \\ &+(m_1^2 - m_1)c_3^2POL(t_1)POL(t_2)POL(t_3) > 0 \end{aligned}$$

By applying the transformation from [8], we finally obtain the following constraints:

$$(c_1 - c_2)c_0 +(c_1^2 - c_1 + c_2 - c_2^2)\mu +(c_1 - c_2)c_3\mu^2 - 1 \geq 0 \quad (22)$$

$$c_1^2 - c_1 - c_0c_3 +(c_1 - c_2)c_3\mu \geq 0 \quad (23)$$

$$(c_1 - c_2)c_3 \geq 0 \quad (24)$$

$$c_2 - c_2^2 + c_0c_3 +(c_1 - c_2)c_3\mu \geq 0 \quad (25)$$

[3] A non-unary polynomial p is *simple-mixed* iff all its exponents are not greater than 1 (i.e., $2xy - 5xyz$ is simple-mixed while $3x^2y$ is not). A unary polynomial p is simple-mixed if it has the form $p_0 + p_1x$ or $p_0 + p_2x^2$. In [13], Steinbach conducted 320 experiments with TRSs from the literature and noticed that 96 % of those TRSs which are compatible with a polynomial ordering are compatible with a *simple-mixed* polynomial ordering.

$$
\begin{aligned}
(1-m_1)c_0 + (c_1+c_2-1)m_0 + c_3m_0^2 & \\
+2c_3m_0m_1\mu + (m_1^2-m_1)c_3\mu^2 - 1 \geq 0 & \quad (26)\\
c_3m_0m_1 + (m_1^2-m_1)c_3\mu \geq 0 & \quad (27)\\
(m_1^2-m_1)c_3 \geq 0 & \quad (28)\\
c_0c_2 + (c_1c_2+c_0c_3)m_0 + (c_3m_0^2-c_0m_1)c_1 & \\
+(\ (c_1-c_1^2+2c_1c_3m_0)m_1 + c_2^2 + (2c_2c_3-c_3)m_0 + c_3^2m_0^2 - c_2\ )\mu & \\
+(\ (m_1^2-2m_1)c_1c_3 + (c_2c_3+2c_3^2m_0)m_1\ )\mu^2 & \\
+(m_1^2-m_1)c_3^2\mu^3 - 1 \geq 0 & \quad (29)\\
(c_1-c_1^2+c_0c_3+c_1c_3m_0)m_1 & \\
+(\ (m_1^2-2m_1)c_1c_3 + (c_2c_3+c_3^2m_0)m_1\ )\mu & \\
+(m_1^2-m_1)c_3^2\mu^2 \geq 0 & \quad (30)\\
(m_1^2-m_1)c_1c_3 + (m_1^2-m_1)c_3^2\mu \geq 0 & \quad (31)\\
(m_1^2-m_1)c_3^2 \geq 0 & \quad (32)\\
(c_2c_3+c_3^2m_0-c_1c_3)m_1 & \\
+(m_1^2-m_1)c_3^2\mu \geq 0 & \quad (33)\\
c_1c_3m_0m_1 & \\
+(\ (m_1^2-m_1)c_1c_3 + c_3^2m_0m_1\ )\mu & \\
+(m_1^2-m_1)c_3^2\mu^2 \geq 0 & \quad (34)\\
c_3^2m_0m_1 + (m_1^2-m_1)c_3^2\mu \geq 0 & \quad (35)\\
c_2^2 + (2c_2c_3-c_3)m_0 + c_3^2m_0^2 - c_2 - c_0c_3m_1 & \\
+(c_2c_3+2c_3^2m_0-c_1c_3)m_1\mu & \\
+(m_1^2-m_1)c_3^2\mu^2 \geq 0 & \quad (36)\\
m_0 + (m_1-1)\mu \geq 0 & \quad (37)\\
m_1 \geq 0 & \quad (38)\\
c_0 + (c_1+c_2-1)\mu + c_3\mu^2 \geq 0 & \quad (39)\\
c_1 + c_3\mu \geq 0 & \quad (40)\\
c_3 \geq 0 & \quad (41)\\
c_2 + c_3\mu \geq 0 & \quad (42)\\
m_1 - 1 \geq 0 & \quad (43)\\
c_1 + c_3\mu - 1 \geq 0 & \quad (44)\\
c_2 + c_3\mu - 1 \geq 0 & \quad (45)
\end{aligned}
$$

Hence, we have to find a sequence $m_0, m_1, c_0, c_1, c_2, c_3, \mu$ such that the above 24 inequalities are satisfied. (Of course, one could first simplify this set of inequalities, since, e.g., (43) implies (38), etc.) A solution is

$$
\begin{aligned}
m_0 &= 0\\
m_1 &= 2\\
c_0 &= 0
\end{aligned}
$$

$$
\begin{aligned}
c_1 &= 1\\
c_2 &= 0\\
c_3 &= 1\\
\mu &= 2
\end{aligned}
$$

This corresponds to the following polynomial interpretation:

$$
\begin{aligned}
POL(\mathsf{map}(t)) &= 2POL(t)\\
POL(\mathsf{comp}(t_1,t_2)) &= POL(t_1) + POL(t_1)POL(t_2)
\end{aligned}
$$

A.3 reverse

The next example is from [13, Ex. 8.5]. It contains both the iterative and the "standard recursive" version of the reverse function on lists.

$$
\begin{aligned}
\mathsf{append}(\mathsf{nil}, y) &\to y,\\
\mathsf{append}(\mathsf{add}(x,y), z) &\to \mathsf{add}(x, \mathsf{append}(y,z)),\\
\mathsf{append}(\mathsf{append}(x,y), z) &\to \mathsf{append}(x, \mathsf{append}(y,z)),\\
\mathsf{reverse}(\mathsf{nil}) &\to \mathsf{nil},\\
\mathsf{reverse}(\mathsf{add}(x,y)) &\to \mathsf{append}(\mathsf{reverse}(y), \mathsf{add}(x, \mathsf{nil})),\\
\mathsf{reviter}(\mathsf{nil}, y) &\to y,\\
\mathsf{reviter}(\mathsf{add}(x,y), z) &\to \mathsf{reviter}(y, \mathsf{add}(x,z)),\\
\mathsf{append}(\mathsf{reverse}(x), y) &\to \mathsf{reviter}(x,y),\\
\mathsf{reverse}(x) &\to \mathsf{reviter}(x, \mathsf{nil})
\end{aligned}
$$

Again, we use a simple-mixed polynomial interpretation. In order to avoid the choice with unary polynomials, here we use an arbitrary polynomial of degree 2 for reverse.

$$
\begin{aligned}
POL(\mathsf{nil}) &= n\\
POL(\mathsf{add}(t_1,t_2)) &= a_0 + a_1POL(t_1) + a_2POL(t_2) + a_3POL(t_1)POL(t_2)\\
POL(\mathsf{append}(t_1,t_2)) &= p_0 + p_1POL(t_1) + p_2POL(t_2) + p_3POL(t_1)POL(t_2)\\
POL(\mathsf{reverse}(t)) &= r_0 + r_1POL(t) + r_2POL(t)^2\\
POL(\mathsf{reviter}(t_1,t_2)) &= i_0 + i_1POL(t_1) + i_2POL(t_2) + i_3POL(t_1)POL(t_2)
\end{aligned}
$$

For compatibility, the polynomial ordering has to satisfy the following inequalities for all ground terms t_1, t_2, t_3:

$$
\begin{aligned}
p_0 + p_1n + (p_2-1)POL(t_1) + p_3nPOL(t_2) &> 0\\
(1-a_2)p_0 + (p_1-1)a_0&\\
+((p_1-1)a_1 - p_0a_3)POL(t_1)&\\
+((1-a_2)p_2 + a_0p_3)POL(t_3)&
\end{aligned}
$$

$$
\begin{aligned}
+(a_1p_3 - a_3p_2)POL(t_1)POL(t_3) &> 0 \\
(p_1 - p_2)p_0 & \\
+(p_1^2 - p_1 - p_0p_3)POL(t_1) & \\
+(p_2 + p_0p_3 - p_2^2)POL(t_3) & \\
+(p_1 - p_2)p_3POL(t_1)POL(t_3) &> 0 \\
r_0 + (r_1 - 1)n + r_2^2n &> 0 \\
(1 - p_1)r_0 + a_0r_1 + a_0^2r_2 - (a_0 + a_2n)(p_2 + p_3r_0) & \\
+(\ (r_1 + 2a_0r_2)a_1 - (a_1 + a_3n)(p_2 + p_3r_0)\ )POL(t_1) & \\
+(\ (a_2 - (a_0 + a_2n)p_3 - p_1)r_1 + 2r_2a_0a_2\ )POL(t_2) & \\
+(\ (a_3 - (a_1 + a_3n)p_3)r_1 + 2(a_0a_3 + a_1a_2)r_2\ )POL(t_1)POL(t_2) & \\
+r_2a_1^2POL(t_1)^2 & \\
+(a_2^2 - p_1 - (a_0 + a_2n)p_3)r_2POL(t_2)^2 & \\
+2r_2a_1a_3POL(t_1)^2POL(t_2) & \\
+(2a_2a_3 - (a_1 + a_3n)p_3)r_2POL(t_1)POL(t_2)^2 & \\
+r_2a_3^2POL(t_1)^2POL(t_2)^2 &> 0 \\
i_0 + i_1n + (i_2 - 1)POL(t_2) + i_3nPOL(t_2) &> 0 \\
(i_1 - i_2)a_0 & \\
+(i_1 - i_2)a_1POL(t_1) & \\
+(i_1a_2 - i_3a_0)POL(t_2) & \\
+((1 - a_2)i_2 + i_3a_0)POL(t_3) & \\
+(i_1a_3 - i_3a_1)POL(t_1)POL(t_2) & \\
+(i_3a_1 - i_2a_3)POL(t_1)POL(t_3) &> 0 \\
p_0 + p_1r_0 - i_0 & \\
+(p_1r_1 - i_1)POL(t_1) & \\
+(p_2 + p_3r_0 - i_2)POL(t_2) & \\
+p_1r_2POL(t_1)^2 & \\
+(p_3r_1 - i_3)POL(t_1)POL(t_2) & \\
+p_3r_2POL(t_1)^2POL(t_2) &> 0 \\
r_0 - i_0 - i_2n + (r_1 - i_1 - i_3n)POL(t_1) + r_2POL(t_1)^2 &> 0
\end{aligned}
$$

By applying the transformation from [8], we finally obtain the following constraints:

$$
\begin{aligned}
p_0 + p_1n + (p_2 - 1 + p_3n)\mu - 1 &\geq 0 \qquad (46) \\
p_2 - 1 + p_3n &\geq 0 \qquad (47) \\
(1 - a_2)p_0 + (p_1 - 1)a_0 & \\
+((p_1 - 1)a_1 - p_0a_3 + (1 - a_2)p_2 + a_0p_3)\mu & \\
+(a_1p_3 - a_3p_2)\mu^2 - 1 &\geq 0 \qquad (48)
\end{aligned}
$$

$$
\begin{aligned}
(p_1-1)a_1-p_0a_3+(a_1p_3-a_3p_2)\mu &\geq 0 && (49)\\
(1-a_2)p_2+a_0p_3+(a_1p_3-a_3p_2)\mu &\geq 0 && (50)\\
a_1p_3-a_3p_2 &\geq 0 && (51)\\
(p_1-p_2)p_0+(p_1^2-p_1+p_2-p_2^2)\mu & \\
+(p_1-p_2)p_3\mu^2-1 &\geq 0 && (52)\\
p_1^2-p_1-p_0p_3+(p_1-p_2)p_3\mu &\geq 0 && (53)\\
p_2+p_0p_3-p_2^2+(p_1-p_2)p_3\mu &\geq 0 && (54)\\
(p_1-p_2)p_3 &\geq 0 && (55)\\
r_0+(r_1-1)n+r_2^2n-1 &\geq 0 && (56)\\
(1-p_1)r_0+a_0r_1+a_0^2r_2-(a_0+a_2n)(p_2+p_3r_0) & \\
+(\ (r_1+2a_0r_2)a_1-(a_1+a_3n)(p_2+p_3r_0) & \\
+(a_2-(a_0+a_2n)p_3-p_1)r_1+2r_2a_0a_2\ )\mu & \\
+(\ (a_3-(a_1+a_3n)p_3)r_1+2(a_0a_3+a_1a_2)r_2 & \\
+r_2a_1^2+(a_2^2-p_1-(a_0+a_2n)p_3)r_2\ )\mu^2 & \\
+(\ 2r_2a_1a_3+(2a_2a_3-(a_1+a_3n)p_3)r_2\ )\mu^3 & \\
+r_2a_3^2\mu^4-1 &\geq 0 && (57)\\
(r_1+2a_0r_2)a_1-(a_1+a_3n)(p_2+p_3r_0) & \\
+(\ (a_3-(a_1+a_3n)p_3)r_1+2(a_0a_3+a_1a_2)r_2+2r_2a_1^2\ )\mu & \\
+(\ 4r_2a_1a_3+(2a_2a_3-(a_1+a_3n)p_3)r_2\ )\mu^2 & \\
+2r_2a_3^2\mu^3 &\geq 0 && (58)\\
r_2a_1^2+2r_2a_1a_3\mu+r_2a_3^2\mu^2 &\geq 0 && (59)\\
r_2a_1a_3+r_2a_3^2\mu &\geq 0 && (60)\\
r_2a_3^2 &\geq 0 && (61)\\
(a_3-(a_1+a_3n)p_3)r_1+2(a_0a_3+a_1a_2)r_2 & \\
+(\ 4r_2a_1a_3+2(2a_2a_3-(a_1+a_3n)p_3)r_2\ )\mu & \\
+4r_2a_3^2\mu^2 &\geq 0 && (62)\\
(2a_2a_3-(a_1+a_3n)p_3)r_2+2r_2a_3^2\mu &\geq 0 && (63)\\
(a_2-(a_0+a_2n)p_3-p_1)r_1+2r_2a_0a_2 & \\
+(\ (a_3-(a_1+a_3n)p_3)r_1+2(a_0a_3+a_1a_2)r_2 & \\
+2(a_2^2-p_1-(a_0+a_2n)p_3)r_2\ )\mu & \\
+(\ 2r_2a_1a_3+2(2a_2a_3-(a_1+a_3n)p_3)r_2\ )\mu^2 & \\
+2r_2a_3^2\mu^3 &\geq 0 && (64)\\
(a_2^2-p_1-(a_0+a_2n)p_3)r_2 & \\
+(2a_2a_3-(a_1+a_3n)p_3)r_2\mu & \\
+r_2a_3^2\mu^2 &\geq 0 && (65)\\
i_0+i_1n+(i_2-1+i_3n)\mu-1 &\geq 0 && (66)
\end{aligned}
$$

$$
\begin{aligned}
i_2 - 1 + i_3 n &\geq 0 && (67)\\
(i_1 - i_2)a_0 & \\
+(\ (i_1 - i_2)a_1 + i_1 a_2 + (1 - a_2)i_2\ )\mu & \\
+(\ i_1 a_3 - i_2 a_3\ )\mu^2 - 1 &\geq 0 && (68)\\
(i_1 - i_2)a_1 & \\
+(i_1 a_3 - i_2 a_3)\mu &\geq 0 && (69)\\
i_1 a_3 - i_3 a_1 &\geq 0 && (70)\\
i_3 a_1 - i_2 a_3 &\geq 0 && (71)\\
i_1 a_2 - i_3 a_0 + (i_1 a_3 - i_3 a_1)\mu &\geq 0 && (72)\\
(1 - a_2)i_2 + i_3 a_0 + (i_3 a_1 - i_2 a_3)\mu &\geq 0 && (73)\\
p_0 + p_1 r_0 - i_0 & \\
+(p_1 r_1 - i_1 + p_2 + p_3 r_0 - i_2)\mu & \\
+(p_1 r_2 + p_3 r_1 - i_3)\mu^2 & \\
+p_3 r_2 \mu^3 - 1 \geq 0 & && (74)\\
p_1 r_1 - i_1 & \\
+(2p_1 r_2 + p_3 r_1 - i_3)\mu & \\
+2p_3 r_2 \mu^2 &\geq 0 && (75)\\
2p_1 r_2 + 2p_3 r_2 \mu &\geq 0 && (76)\\
p_3 r_2 &\geq 0 && (77)\\
p_3 r_1 - i_3 + 2p_3 r_2 \mu &\geq 0 && (78)\\
p_2 + p_3 r_0 - i_2 & \\
+(p_3 r_1 - i_3)\mu & \\
+p_3 r_2 \mu^2 &\geq 0 && (79)\\
r_0 - i_0 - i_2 n + (r_1 - i_1 - i_3 n)\mu + r_2 \mu^2 - 1 &\geq 0 && (80)\\
r_1 - i_1 - i_3 n + 2r_2 \mu &\geq 0 && (81)\\
r_2 &\geq 0 && (82)\\
n - \mu &\geq 0 && (83)\\
a_0 + (a_1 + a_2 - 1)\mu + a_3 \mu^2 &\geq 0 && (84)\\
a_1 + a_3 \mu &\geq 0 && (85)\\
a_3 &\geq 0 && (86)\\
a_2 + a_3 \mu &\geq 0 && (87)\\
p_0 + (p_1 + p_2 - 1)\mu + p_3 \mu^2 &\geq 0 && (88)\\
p_1 + p_3 \mu &\geq 0 && (89)\\
p_3 &\geq 0 && (90)\\
p_2 + p_3 \mu &\geq 0 && (91)\\
r_0 + (r_1 - 1)\mu + r_2 \mu^2 &\geq 0 && (92)
\end{aligned}
$$

$$
\begin{aligned}
r_1 + 2r_2\mu &\geq 0 & (93)\\
i_0 + (i_1 + i_2 - 1)\mu + i_3\mu^2 &\geq 0 & (94)\\
i_1 + i_3\mu &\geq 0 & (95)\\
i_3 &\geq 0 & (96)\\
i_2 + i_3\mu &\geq 0 & (97)\\
a_1 + a_3\mu - 1 &\geq 0 & (98)\\
a_2 + a_3\mu - 1 &\geq 0 & (99)\\
p_1 + p_3\mu - 1 &\geq 0 & (100)\\
p_2 + p_3\mu - 1 &\geq 0 & (101)\\
r_1 + 2r_2\mu - 1 &\geq 0 & (102)\\
i_1 + i_3\mu - 1 &\geq 0 & (103)\\
i_2 + i_3\mu - 1 &\geq 0 & (104)
\end{aligned}
$$

Hence, we have to find a sequence $n, a_0, a_1, a_2, a_3, p_0, p_1, p_2, p_3, r_0, r_1, r_2, i_0, i_1, i_2, i_3, \mu$ such that the above 59 inequalities are satisfied. A solution is

$$
\begin{aligned}
n &= 2\\
a_0 &= 2\\
a_1 &= 2\\
a_2 &= 2\\
a_3 &= 2\\
p_0 &= 0\\
p_1 &= 2\\
p_2 &= 0\\
p_3 &= 1\\
r_0 &= 2\\
r_1 &= 0\\
r_2 &= 2\\
i_0 &= 0\\
i_1 &= 2\\
i_2 &= 0\\
i_3 &= 1\\
\mu &= 2
\end{aligned}
$$

This corresponds to the following polynomial interpretation:

$$
\begin{aligned}
POL(\mathsf{nil}) &= 2\\
POL(\mathsf{add}(t_1, t_2)) &= 2 + 2POL(t_1) + 2POL(t_2) + 2POL(t_1)POL(t_2)\\
POL(\mathsf{append}(t_1, t_2)) &= 2POL(t_1) + POL(t_1)POL(t_2)
\end{aligned}
$$

$$POL(\mathsf{reverse}(t)) = 2 + 2POL(t)^2$$
$$POL(\mathsf{reviter}(t_1, t_2)) = 2POL(t_1) + POL(t_1)POL(t_2)$$

References

1. T. Arts and J. Giesl. Termination of Term Rewriting using Dependency Pairs. *Theoretical Computer Science*, 236:133-178, 2000.
2. F. Baader and T. Nipkow. *Term Rewriting and All That*, Cambridge University Press, 1998.
3. F. Bellegarde. Rewriting Systems on FP Expressions that reduce the Number of Sequences they yield. *Symposium on LISP and Functional Programming*, ACM, Austin, Texas, 1984.
4. A. Ben Cherifa and P. Lescanne. Termination of Rewriting Systems by Polynomial Interpretations and its Implementation. *Science of Computer Programming*, 9(2):137-159, 1987.
5. CiME 2. Pre-release available at http://www.lri.fr/~demons/cime-2.0.html, 1999.
6. G. E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, 1975.
7. N. Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, 3(1,2):69-116, 1987.
8. J. Giesl. Generating Polynomial Orderings for Termination Proofs. *Proceedings of the 6th International Conference on Rewriting Techniques and Applications, RTA-95*, LNCS 914, Springer-Verlag, pp. 426-431, 1995. Extended version available as Technical Report 95/23, Darmstadt University of Technology, Germany, 1995. http://www.inferenzsysteme.informatik.tu-darmstadt.de/~giesl
9. M. Mitchell, J.H. Holland and S. Forrest. Relative building-block fitness and the building block hypothesis. In D. Whitley, *Foundations of Genetic Algorithms 2*, 109-126. San Mateo, CA:Morgan Kaufmann, 1993.
10. J. Giesl. POLO – A System for Termination Proofs Using Polynomial Orderings. Technical Report 95/24, Darmstadt University of Technology, Germany, 1995. http://www.inferenzsysteme.informatik.tu-darmstadt.de/~giesl
11. H. Hong and D. Jakus. Testing Positiveness of Polynomials. *Journal of Automated Reasoning*, 21:23-38, 1998.
12. D. S. Lankford. On Proving Term Rewriting Systems are Noetherian. Technical Report Memo MTP-3, Louisiana Technical University, Ruston, LA, 1979.
13. J. Steinbach. Termination Proofs of Rewriting Systems — Heuristics for Generating Polynomial Orderings. SEKI-Report SR-91-14, University of Kaiserslautern, Germany, 1991.
14. J. Steinbach. Generating Polynomial Orderings. *Information Processing Letters*, 49:85-93, 1994.
15. J. Steinbach. Simplification Orderings: History of Results. *Fundamenta Informaticae*, 24:47-87, 1995.
16. A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, Berkeley, 1951.
17. J.H. Holland. Adaptation in Natural and Artificial Systems. The University of Michigan Press, 1975.
18. M. Mitchell. An Introduction to Genetic Algorithms. MIT Press, 1996.