Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

A Method to Prove the Positiveness of Polynomials

Joachim Steinbach

SEKI Report SR-91-13 (SFB)

A Method to Prove the Positiveness of Polynomials[1)]

Joachim STEINBACH
Universität Kaiserslautern
FB Informatik
Postfach 3049
D-6750 Kaiserslautern
Germany
e-mail: steinba@informatik.uni-kl.de

December 17, 1991

[1)] This research was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D4-Projekt)

Abstract

Termination is an important property for programming and particularly for term rewriting systems. The well-known polynomial orderings can be used for proving the termination of term rewriting systems. The proof of the positiveness of a polynomial presents a crucial point concerning polynomial orderings. There exists a powerful non-deterministic method for performing such proofs that has been developed by BenCherifa and Lescanne. We describe some observations on the time complexity of this approach. In addition, a deterministic version is presented which has the same power as the original one. We also deal with a modification for signatures which do not contain any constant symbols. Finally, we discuss an improvement of the non-deterministic method.

Contents

1 **Introduction** 2

2 **Notations** 2

3 **The Method of BenCherifa and Lescanne** 4

4 **Towards the BenCherifa and Lescanne Approach** 6

5 **The 'No Constants'-Case** 11

6 **A Modification of the Approach of BenCherifa and Lescanne** 13

7 **A Modified Simplex Algorithm** 15

8 **Integrating the Simplex Method into Algorithm 6.1** 19

9 **Improving the BenCherifa and Lescanne Method** 22

A **Appendix : Proofs** 27

B **An Example** 32

List of Figures

1 Procedure $\mathcal{POSITIVE}$ of [BL87] 5
2 Procedure $\mathcal{CHANGE}$ of [BL87] 5
3 The sequences corresponding to example 3.1 6
4 Procedure $\mathcal{CHOOSE}$ of [Ben86] 7
5 The sequence, belonging to example 3.2, generated by $\mathcal{CHOOSE}$ 7
6 Number of paths for the polynomial of example 4.1 9
7 The worst case for procedure $\mathcal{POSITIVE}$ 10
8 The paths for the polynomial of example 4.3 12
9 Division process of algorithm 6.1 14
10 Deterministic version of algorithm $\mathcal{POSITIVE}$ 15
11 The application of algorithm 6.1 to the example of section 3 16
12 Merging algorithm 6.1 and procedure $\mathcal{POSITIVE}$ 17
13 The 1^{st} phase of the Simplex method 20
14 Connection between algorithm 6.1 and the Simplex method 20
15 The set of inequalities generated by algorithm 6.1 22

1 Introduction

Term rewriting systems (TRS, for short) provide a powerful tool for expressing non-deterministic computations. As programs they have a very simple syntax and their semantic is based on equalities that are used as reduction rules with no explicit control. For this purpose it is essential that a TRS has the property of termination.

There exist various methods for proving termination of TRS. Most of these are based on reduction orderings which are well-founded, compatible with the structure of terms and stable with respect to (w.r.t., for short) substitutions. The notion of reduction orderings leads to the following description of termination of TRS (see [Lan77]):

> *A TRS $\mathcal{R}$ terminates if, and only if, there exists a reduction ordering $\succ$ such that $l \succ r$ for each rule $l \to r$ of $\mathcal{R}$.*

One way of constructing reduction orderings consists of the specification of a well-founded set $(\mathcal{W}, \succ)$ and a mapping φ (called termination function) from the set of terms into $\mathcal{W}$, such that $\varphi(s) \succ \varphi(t)$ whenever t can be derived from s ([MN70]). The well-known Knuth-Bendix orderings ([KB70]) are thus defined using $\mathcal{W} := \mathbb{N}$, $\succ := >$[1)] and φ as the so-called weight function. Polynomial orderings proposed by Lankford ([Lan75a],[Lan79]) are based on the set of polynomials over $\mathbb{N}$ (representing $\mathcal{W}$) where φ denotes a polynomial interpretation (also called norm function) and $\succ$ represents an ordering on polynomials (which is, in the case of ground terms, equivalent to $>$).

The use of polynomial orderings reduces the proof of termination of TRS to finding appropriate interpretations orienting the given system, on the one hand, and to deciding whether a given polynomial is greater than another one, on the other hand. The first problem is treated, for example, in [Ben86] and [Ste91]. The topic of this paper concerns the second problem. In the literature, there exist a few methods ([Lan79], [BL87], [Rou88] and [Rou91], for example) for handling the proof of whether a polynomial is greater than zero. In this report, we present some modifications of the technique contained in [BL87].

In the following section we briefly recapitulate the most essential notions used in connection with TRS and termination[2)]. Section 3 deals with the description of the constructive factors concerning the technique of BenCherifa and Lescanne ([BL87]). An examination of this method, related to the time complexity, is given in section 4. In section 5, we provide a modification for signatures which do not contain any constant symbols. A deterministic version of [BL87] is presented in section 6. It is based on the transformation of a polynomial into a set of linear inequalities (according to the coefficients of the polynomial) which can be solved by applying, for example, the first phase of the well-known Simplex algorithm. The most important reason for the development of this technique lies in the fact, that it can be used for (automatically) generating a polynomial interpretation for a given TRS (see [Ste91], [SZ90]) whereas the technique of [BL87] cannot. Section 7 contains a brief description of the first phase of the Simplex algorithm. Subsequently, we deal with the time complexity of the Simplex method because the Simplex method forms the main part of the approach given in section 6. Finally, we present an improvement of the BenCherifa/Lescanne technique.

2 Notations

We assume familiarity with the standard definitions of the set of *function symbols* (or operators) $\mathcal{F}$ and their arities[3)], the set of *variables* $\mathcal{X}$, the set of *terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$, the set of *ground terms* $\mathcal{G}(\mathcal{F})$ as well as with the definition of a *substitution* $t\sigma$ of a term t and *rewriting systems* $\mathcal{R} = \{l_i \to r_i \mid i \in I\}$[4)].

A *partial ordering* $\succ$ is a transitive and irreflexive binary relation. It is said to be *well-founded* if there exists no infinite descending sequence. A partial ordering on $\mathcal{T}(\mathcal{F},\mathcal{X})$ is called a *term ordering.* A *reduction ordering* $\succ$ is a well-founded term ordering which is *stable* w.r.t. substitutions ($s \succ t \;\leadsto\; s\sigma \succ t\sigma$) and *monotonic* w.r.t. (or

[1)] In this paper, $>$ represents the natural ordering on $\mathbb{N}$.

[2)] For details see, for example, [HO80] and [Der87].

[3)] An operator with no arguments (i.e. whose arity is zero) is called a *constant* (symbol).

[4)] I is a set of indices.

compatible with) the structure of terms ($s \succ t \rightsquigarrow f(\ldots,s,\ldots) \succ f(\ldots,t,\ldots)$).

Polynomial orderings are special reduction orderings and have been studied by Manna & Ness ([MN70]), Lankford ([Lan75a], [Lan75b], [Lan76], [Lan79]), Dershowitz ([Der79], [Der83], [Der87]), Huet & Oppen ([HO80]), BenCherifa & Lescanne ([Ben86], [BL87]) and Rouyer ([Rou88], [Rou91]). Manna & Ness, Lankford, Huet & Oppen and BenCherifa & Lescanne have proposed a method which maps the set of terms into a well-founded set by attaching monotonic functions to operators[5]. Let us now describe this technique.

The set of all *polynomials* with an arbitrary number of variables and with coefficients from $\mathbb{N}$ is denoted by $\mathcal{P}ol(\mathbb{N})$. A polynomial is composed of a sum of *monomials*[6] $\alpha_{r_1\ldots r_n} \cdot x_1^{r_1} \cdot \ldots \cdot x_n^{r_n}$. $p(x_1,\ldots,x_n)$ represents the polynomial $\sum \alpha_{r_1\ldots r_n} x_1^{r_1} \cdot \ldots \cdot x_n^{r_n}$ based on n distinct variables. The ordering $\succ_{MON}$ on two monomials is defined as[7]

$\alpha_{i_1\ldots i_n} \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \succ_{MON} \alpha_{j_1\ldots j_n} x_1^{j_1} \cdot \ldots \cdot x_n^{j_n}$ if, and only if, $(i_1,\ldots,i_n) >^n_{lex} (j_1,\ldots,j_n)$.

Since every ground polynomial is equal to a natural number, we will identify the set of ground polynomials with $\mathbb{N}$. A polynomial p possesses a *strict arity* n if n variables occur in p that differ by pairs.

Definition 2.1 *A polynomial interpretation* $[.]$: $\mathcal{F} \cup \mathcal{X} \mapsto \mathcal{P}ol(\mathbb{N})$ *(for variable terms) maps each n-ary function symbol* $f \in \mathcal{F}$ *into a polynomial* $p \in \mathcal{P}ol(\mathbb{N})$ *of strict arity n and each variable* $x \in \mathcal{X}$ *over terms into a variable* $X \in \mathcal{V}$ *over* $\mathbb{N}$. *This mapping can be extended to* $[.]$: $\mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{P}ol(\mathbb{N})$ *by defining* $[f(t_1,\ldots,t_n)] = [f]([t_1],\ldots,[t_n])$.

Definition 2.2 ([Lan75a]) *Let* $\mathcal{M}$ *be any non-empty set such that* $[\mathcal{G}(\mathcal{F})] \subseteq \mathcal{M} \subseteq \mathbb{N}_+$. *The polynomial ordering* $\succ_{POL}$ *on two terms s and t is defined as*

$$s \succ_{POL} t \iff [s] \sqsupset [t]$$

with[8] $p \sqsupset q \iff$

$$(\forall X_1,\ldots,X_n \in \mathcal{M})\, p(X_1,\ldots,X_n) > q(X_1,\ldots,X_n)$$

Example 2.1 *We prove the termination of*

$$\mathcal{R} = \begin{cases} f(x,y) & \rightarrow & g(x,y) \\ g(h(x),y) & \rightarrow & h(f(x,y)) \end{cases}$$

by using $\succ_{POL}$ *based on the interpretations*

$$\begin{aligned} [f](X,Y) &= 2X+Y+1 \\ [g](X,Y) &= 2X+Y \\ [h](X) &= X+2. \end{aligned}$$

Note that $[f(x,y)](X,Y) = 2X+Y+1$ *and* $[g(x,y)](X,Y) = 2X+Y$, $[g(h(x),y)](X,Y) = 2X+Y+4$ *and* $[h(f(x,y))](X,Y) = 2X+Y+3$. *Since* $\mathcal{R}$ *contains no constant symbols, the non-empty set* $\mathcal{M}$ *can arbitrarily be chosen. We have to show that* $(\forall X,Y \in \mathcal{M})\, 2X+Y+1 > 2X+Y \wedge 2X+Y+4 > 2X+Y+3$ *which is valid for any* $\mathcal{M} \subseteq \mathbb{N}_+$.

If $\mathcal{F}$ contains at least one constant symbol, i.e. $\mathcal{G}(\mathcal{F}) \neq \emptyset$, the polynomial ordering $\succ_{POL}$ strongly depends on $[\mathcal{G}(\mathcal{F})]$ since definition 2.2 requires $[\mathcal{G}(\mathcal{F})] \subseteq \mathcal{M}$. Because of the interpretations of ground terms being natural numbers, $\mathcal{M}$ has a unique minimal (w.r.t. $>$ on $\mathbb{N}$) element. In the remaining part of this paper, the *minimum* of the set $\mathcal{M}$ is denoted by μ.

Remark 2.1 *Note that* $\mu = min\{[c]() \mid c$ *is a constant symbol of* $\mathcal{F}\}$ *if there exists at least one constant symbol in* $\mathcal{F}$. *If* μ *is strictly greater than the minimal interpretation of all constant symbols, the induced polynomial ordering might no longer be stable w.r.t. substitutions and, as a consequence, does not need to be well-founded. This can be illustrated by a simple example: Let* $[g](x) = x^2$, $[h](x) = x+1$ *and* $[a]() = 1$. *Then,*

$$\begin{aligned} g(x) &\succ_{POL} h(x) \quad if\ \mu = 2 \\ h(a) &\succ_{POL} g(a) \end{aligned}$$

---

[5] Dershowitz uses an arbitrary set by requiring the functions to possess the subterm property.

[6] We often use $\alpha_{r_1\ldots r_n}$ for referring to the exponents of the variables (e.g., $\alpha_{210}x^2y + \alpha_{101}xz$).

[7] The ordering $>^n_{lex}$ denotes the lexicographical extension of $>$ to tuples of n natural numbers.

[8] Note that $[s]$ and $[t]$ are polynomials. Thus, $\sqsupset$ is an ordering on polynomials (p,q).

The stability of $\succ_{POL}$ must guarantee that $g(a) \succ_{POL} h(a)$ holds if $g(x) \succ_{POL} h(x)$. It is obvious that the system $\{g(x) \rightarrow h(x), h(a) \rightarrow g(a)\}$ does not terminate. Anyway, the above condition concerning μ must be guaranteed (which implies that $g(x)$ and $h(x)$ are incomparable w.r.t. $\succ_{POL}$).

Instead of using the set $\mathcal{M}$, we require $\sqsupset$ to be defined as $p \sqsupset q$ iff $(\forall X_i \geq \mu)\ p(X_1, \ldots, X_n) > q(X_1, \ldots, X_n)$. In the remaining part of this paper, we no longer differentiate between capital letters for denoting variables of polynomials and lower case for representing variables of terms. We adopt the lower case form for simplicity.

3 The Method of BenCherifa and Lescanne

The use of polynomial orderings reduces the proof of termination of TRS to finding appropriate interpretations orienting the given system, on the one hand, and to deciding whether a given polynomial is greater than another one, on the other hand. In general, the comparison of two polynomials ($p \sqsupset q$) is reduced to the proof of the positiveness of just *one* polynomial ($p - q \sqsupset 0$). The problem whether a given polynomial in n variables is positive over real numbers is generally decidable, although in exponential time ([Tar51], [Col75]). However, if we restrict the domain of a polynomial to a proper subset of $\mathbb{R}$, such as $\mathbb{N}$, the problem is generally undecidable ([Dav73]). There are a few well-known approaches concerning this problem (see for example [Lan79], [BL87], [Rou88] and [Rou91]). In this section, we briefly present the technique of [BL87].

The main idea of the method proposed by BenCherifa and Lescanne is to prove $p \sqsupset 0$ by finding polynomials $p_1, \ldots, p_n$ such that

$$p = p_0 \sqsupseteq p_1 \sqsupseteq \ldots \sqsupseteq p_n \sqsupset 0.$$

The positiveness of p_n is checked by a basic principle like 'all coefficients are positive'. The transformation of p_i into p_{i+1} is performed by merging a negative monomial and an appropriate positive one. This process consists of two tasks. Firstly, we have to *choose* a pair of monomials, one having a positive and the other one containing a negative coefficient. Secondly, we must *transform* the negative and the positive one into one singular monomial. Since these two procedures, named $\mathcal{CHOOSE}$ and $\mathcal{CHANGE}$, are the essential points of the problem, we will discuss them in detail after the presentation of the whole algorithm.

Algorithm 3.1 ([BL87]) *We assume that a polynomial can be represented as a set of monomials each realized as a tuple $(\alpha_{e_1 \ldots e_n}, e_1 \ldots e_n)$ where e_i stands for the exponent of the variable x_i and $\alpha_{e_1 \ldots e_n}$ for the coefficient of the monomial. Figure 1 represents algorithm $\mathcal{POSITIVE}$ of [BL87].*

As noted previously, the main idea of the procedure $\mathcal{POSITIVE}$ is to consider a monomial N with a negative coefficient and try to find a monomial M with a positive coefficient which is an upper bound of it. When such a monomial M is found, we divide it into two parts M_1 and M_2 with $M_1 + M_2 \leq M$ such that M_1 forms an upper bound of N. Thus, the positiveness of the whole polynomial p can be guaranteed by proving the positiveness of a polynomial p', which is derived from p by replacing the monomials N and M by M_2. For example, we prove $2xy - xy - x$ to be positive by transforming $2xy$ to $xy + xy$, replacing $2xy$ and $-xy$ by xy, and proving $xy - x$ to be positive.

We now discuss procedure $\mathcal{CHANGE}$. As mentioned in section 2, the realization of $\mathcal{CHANGE}$ strongly depends on the minimum of the interpretations of the constant symbols contained in $\mathcal{F}$. In [BL87], the constants will be interpreted as natural numbers greater than or equal to[9] 2. Thus, a monomial M having $\alpha_{e_1 \ldots e_n}$ as positive coefficient forms an upper bound of a monomial N consisting of the coefficient $\alpha_{f_1 \ldots f_n}$ (which is negative) if $\alpha_{e_1 \ldots e_n} \cdot 2^{\sum(e_i - f_i)} > |\alpha_{f_1 \ldots f_n}|$[10]. If M is not an upper bound of N, this number can be added to $\alpha_{f_1 \ldots f_n}$ to minimize the negative coefficient[11].

Algorithm 3.2 ([BL87]) *Analogous with algorithm 3.1, let $\alpha_{e_1 \ldots e_n}$ ($\alpha_{f_1 \ldots f_n}$, respectively) be the*

[9] This implies that $\mathcal{M}$ represents the set $\{k \mid k \in \mathbb{N}, k \geq 2\}$, i.e. $\mu = 2$.

[10] Note that $(\forall i \in [1, n])\, e_i \geq f_i$. This condition is required in $\mathcal{POSITIVE}$.

[11] For example, $p' + x^2y - 4xy$ will lead to $p' - 2xy$ if $\mu = 2$, since $x^2y \geq 2xy$.

$\mathcal{POSITIVE}$ = **proc** *(p : polynomial)* **returns** *(string)*
while *there exists a negative coefficient in p* **do**
if *there exist* $\alpha_{e_1...e_n} > 0$ *and* $\alpha_{f_1...f_n} < 0$ *with* $(\forall i \in [1..n])\, e_i \geq f_i$
then $(\alpha_{e_1...e_n}, \alpha_{f_1...f_n}) := \mathcal{CHOOSE}(p)$
$\mathcal{CHANGE}(\alpha_{e_1...e_n}, \alpha_{f_1...f_n})$
else **return** *('no answer')*
end
return *('positive')*
end

Figure 1: Procedure $\mathcal{POSITIVE}$ of [BL87]

$\mathcal{CHANGE}$ = **proc**$(\alpha_{e_1...e_n}, \alpha_{f_1...f_n}$ *: monomial)*
if $\alpha_{e_1...e_n} > |\alpha_{f_1...f_n} \cdot 2^{\sum(f_i - e_i)}|$
then $\alpha_{e_1...e_n} := \alpha_{e_1...e_n} + \alpha_{f_1...f_n} \cdot 2^{\sum(f_i - e_i)}$
$\alpha_{f_1...f_n} := 0$
else $\alpha_{f_1...f_n} := \alpha_{f_1...f_n} + \alpha_{e_1...e_n} \cdot 2^{\sum(e_i - f_i)}$
$\alpha_{e_1...e_n} := 0$
end

Figure 2: Procedure $\mathcal{CHANGE}$ of [BL87]

representation of a monomial. Procedure $\mathcal{CHANGE}$ is contained in figure 2.

It is obvious that algorithm 3.1 can be adapted to a general minimum μ of $\mathcal{M}$. Then, all the integers '2' of procedure $\mathcal{CHANGE}$ will have to be replaced by μ. By increasing μ, this method becomes more powerful. However, remember that μ is bounded by the minimum of the interpretations of all constant symbols.

Remark 3.1
A general version of procedure $\mathcal{CHANGE}$ transforms the monomials $\alpha_{e_1...e_n}$ (positive) and $\alpha_{f_1...f_n}$ (negative) in the following way:

$$(\alpha_{e_1...e_n}, \alpha_{f_1...f_n}) = \begin{cases} (\alpha_{e_1...e_n} + \alpha_{f_1...f_n}\mu^{\sum(f_i - e_i)}, 0) \\ if\ \alpha_{e_1...e_n} > |\alpha_{f_1...f_n}\mu^{\sum(f_i - e_i)}| \\ \\ (0, \alpha_{f_1...f_n} + \alpha_{e_1...e_n}\mu^{\sum(e_i - f_i)}) \\ otherwise \end{cases}$$

After merging two monomials, an increase of μ would lead to a greater positive coefficient (a smaller negative coefficient, respectively). More precisely: For compactness, let $a = \alpha_{e_1...e_n}$, $b = \alpha_{f_1...f_n}$ and $c = \sum(f_i - e_i)$. Then, $a + b\mu_1^c > a + b\mu_2^c$ if $a > |b\mu_i^c| \wedge \mu_1 > \mu_2$ as well as $b + a\mu_1^{-c} < b + a\mu_2^{-c}$ if $a \leq |b\mu_i^c| \wedge \mu_1 > \mu_2$. Thus, the greater the μ is, the more powerful procedure $\mathcal{POSITIVE}$ will be.

Before discussing procedure $\mathcal{CHOOSE}$ we illustrate the steps, presented up to now, by an example.

Example 3.1 *Suppose we have to prove that a term s is greater than a term t w.r.t. $\succ_{POL}$ based on a polynomial interpretation [.] such that*

$$\begin{aligned} [s] &= 3x^2y + 6xy^2, \\ [t] &= 2x^2 + 6y^2 + 12xy + 9x + 9y \ \ and \\ \mu &= 3. \end{aligned}$$

If we want to show $[s] \sqsupset [t]$, we need to check the positiveness of the polynomial $p = [s] - [t]$. Consider the two sequences of figure 3, both starting with p.

$$
\begin{array}{rcl}
p &=& p_0 = [s]-[t] = 3x^2y + \mathbf{6xy^2} - 2x^2 - 6y^2 - 12xy - \mathbf{9x} - 9y \\
&\sqsupseteq& p_1 = 3x^2y + \mathbf{5xy^2} - 2x^2 - 6y^2 - 12xy - \mathbf{9y} \\
&\sqsupseteq& p_2 = 3x^2y + \mathbf{4xy^2} - 2x^2 - 6y^2 - \mathbf{12xy} \\
&\sqsupseteq& p_3 = \mathbf{3x^2y} - \mathbf{2x^2} - 6y^2 \\
&\sqsupseteq& p_4 = \frac{7}{3}x^2y - 6y^2
\end{array}
$$

Note that we do not have any chance to show the positiveness of p_4 (p, respectively). However, consider the following sequence:

$$
\begin{array}{rcl}
p &=& p_0 = 3x^2y + \mathbf{6xy^2} - 2x^2 - \mathbf{6y^2} - 12xy - 9x - 9y \\
&\sqsupseteq& p_1 = \mathbf{3x^2y} + 4xy^2 - \mathbf{2x^2} - 12xy - 9x - 9y \\
&\sqsupseteq& p_2 = \frac{7}{3}x^2y + \mathbf{4xy^2} - \mathbf{12xy} - 9x - 9y \\
&\sqsupseteq& p_3 = \mathbf{\frac{7}{3}x^2y} - \mathbf{9x} - 9y \\
&\sqsupseteq& p_4 = \mathbf{\frac{4}{3}x^2y} - \mathbf{9y} \\
&\sqsupseteq& p_5 = \frac{1}{3}x^2y
\end{array}
$$

Note that $p_5 \sqsupset 0$ holds since all coefficients are positive.

Figure 3: The sequences corresponding to example 3.1

Procedure $\mathcal{CHOOSE}$ realizes a heuristic for finding an appropriate positive monomial for each negative monomial. At the end of this section, we present the version of $\mathcal{CHOOSE}$ as it is contained in [Ben86] and implemented in the REVE system. $\mathcal{CHOOSE}$ selects the greatest (w.r.t. $\succ_{MON}$) negative monomial as well as the greatest (w.r.t. $\succ_{MON}$) positive monomial such that the set of variables is minimal w.r.t. the negative monomial. For example, consider the polynomial $p = x^3yz + x^2y^2 - x^2y - xy^2z$. First of all, we merge the monomials x^2y^2 and $-x^2y$ since $x^2y \succ_{MON} xy^2z$ and the number of variables of x^2y^2 is smaller than that of x^3yz (although $x^3yz \succ_{MON} x^2y^2$ holds).

Algorithm 3.3 ([Ben86]) *Analogous with algorithm 3.1, we assume that a polynomial can be represented as a set of monomials each realized as a tuple $(\alpha_{e_1\ldots e_n}, e_1\ldots e_n)$ where e_i stands for the exponent of the variable x_i and $\alpha_{e_1\ldots e_n}$ for the coefficient of the monomial. Procedure $\mathcal{SORT\text{-}MON}$ receives a list of monomials as input and returns a sorted (w.r.t. $\succ_{MON}$, in descending order) copy of it. Procedure $\mathcal{CHOOSE}$ is featured by figure 4.*

Example 3.2 (Example 3.1 revisited) *By using procedure $\mathcal{CHOOSE}$, the sequence of figure 5, starting with the polynomial p of example 3.1, will be generated. Note that the length of such a sequence $p_0, p_1, \ldots$ strongly depends on procedure $\mathcal{CHOOSE}$[12]. For example, the length of the successful sequence presented in example 3.1 is 5 whereas $\mathcal{CHOOSE}$ generates a sequence of length 6.*

4 Towards the BenCherifa and Lescanne Approach

In this section, the empirical time complexity of procedure $\mathcal{POSITIVE}$ is discussed. Note that it is very difficult to give an exact complexity result since it depends on (the relationships between) the coefficients of the positive and the negative monomials. For the remaining part of this section, we assume that algorithm $\mathcal{POSITIVE}$ contains a *backtracking* component which will be employed whenever the chosen sequence cannot prove the positiveness of the given polynomial.

[12] and thus on the values of the coefficients

$\mathcal{CHOOSE}$ = **proc**(*p : polynomial*) **returns** (*monomials*)
    NL : Negative-Monomial-List := ()
    while *there exists a negative monomial* $\alpha_{f_1 \ldots f_n}$ *in p* **do**
        $NL := NL \circ \alpha_{f_1 \ldots f_n}$
    end
    $NL := \mathcal{SORT\text{-}MON}(NL)$
    $\alpha_{f_1 \ldots f_n} := NL[1]$
    PL : Positive-Monomial-List := ()
    while *there exists a positive monomial* $\alpha_{e_1 \ldots e_n}$ *in p* **do**
        if $(\forall i \in [1, n])\, e_i \geq f_i$
        then $PL := PL \circ \alpha_{e_1 \ldots e_n}$
    end
    $PL := \mathcal{SORT\text{-}MON}(PL)$
    $\alpha_{e_1 \ldots e_n} := PL[i]$ *such that i and* $\sum_k (e_k - f_k)$ *are minimal*
    return $(\alpha_{e_1 \ldots e_n}, \alpha_{f_1 \ldots f_n})$
end

Figure 4: Procedure $\mathcal{CHOOSE}$ of [Ben86]

$$
\begin{array}{lclcl}
p & = & p_0 & = & \mathbf{3x^2y} + 6xy^2 - \mathbf{2x^2} - 6y^2 - 12xy - 9x - 9y \\
 & \sqsupseteq & p_1 & = & \mathbf{\tfrac{7}{3}x^2y} + 6xy^2 - 6y^2 - \mathbf{12xy} - 9x - 9y \\
 & \sqsupseteq & p_2 & = & \mathbf{6xy^2} - 6y^2 - \mathbf{5xy} - 9x - 9y \\
 & \sqsupseteq & p_3 & = & \mathbf{\tfrac{13}{3}xy^2} - 6y^2 - \mathbf{9x} - 9y \\
 & \sqsupseteq & p_4 & = & \mathbf{\tfrac{10}{3}xy^2} - \mathbf{6y^2} - 9y \\
 & \sqsupseteq & p_5 & = & \mathbf{\tfrac{4}{3}xy^2} - \mathbf{9y} \\
 & \sqsupseteq & p_6 & = & \tfrac{1}{3}xy^2
\end{array}
$$

Figure 5: The sequence, belonging to example 3.2, generated by $\mathcal{CHOOSE}$

As we have seen in the last section (example 3.1 and example 3.2), there exist different possibilities to *choose* negative and positive monomials. These multiple choices can be described by a tree[13] of which the nodes represent the polynomials p_i generated from a given polynomial p.

- The number $\Pi(p)$ of *paths* (i.e. the number of leaves) in this tree stands for the number of different sequences from p to the result *'positive'* or *'no answer'*, which procedure $\mathcal{POSITIVE}$ returns.
- The number $\Phi(p)$ of *nodes* (except the root) represents the number of computations of the sum of two monomials (i.e. it is identical[14] to the number of calls of procedure $\mathcal{CHANGE}$).

Example 4.1 (Example 3.1 revisited) *Figure 6 illustrates the number of polynomials which can be created when starting with the polynomial p of example 3.1. Due to the difficulty of graphically displaying all nodes, we present their numbers (beginning at level 4), only. Totally, 768 paths through this tree exist (although p is a relatively simple polynomial) of which 212 cannot be used to show the positiveness of p. Note that $\Phi(p) = 1967$.*

The best case[15] for the complexity of procedure $\mathcal{POSITIVE}$ is n, where n represents the number of negative monomials occurring in p. This complexity can only be achieved if p is a polynomial such that each positive monomial can cover all negative monomials, i.e.

- the exponent of each positive monomial is greater than or equal to the exponent of each negative monomial (for all variables) and
- each positive coefficient is greater than the sum (w.r.t. μ) of all negative monomials.

More formally: Let $p = \sum(\alpha_{i_1\ldots i_n} - \beta_{i_1\ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ such that $\alpha_{i_1\ldots i_n} \geq 0$, $\beta_{i_1\ldots i_n} \geq 0$ and $\alpha_{i_1\ldots i_n} \cdot \beta_{i_1\ldots i_n} = 0$[16]. Then, the following two conditions must hold:

- $(\forall \alpha_{i_1\ldots i_n} > 0)(\forall \beta_{j_1\ldots j_n} > 0)(\forall k \in [1,n])\, i_k \geq j_k$
- $(\forall \alpha_{i_1\ldots i_n} > 0)\, \alpha_{i_1\ldots i_n} > \sum \mu^{\sum(j_k - i_k)} \cdot \beta_{j_1\ldots j_n}$

A typical polynomial, fulfilling these conditions, is considered in example 4.2.

Example 4.2 *Let $p = 3x^2y + 6xy - 4x - 5y - 2$ and $\mu = 2$. Each positive monomial can cover all negative monomials since $3 > 4\cdot 2^{-2} + 5\cdot 2^{-2} + 2\cdot 2^{-3}$ and $6 > 4\cdot 2^{-1} + 5\cdot 2^{-1} + 2\cdot 2^{-2}$.*

The worst case for procedure $\mathcal{POSITIVE}$ will occur if p is not greater than zero[17] and the corresponding tree is maximal. Furthermore, the exponents of each positive monomial have to cover those of every negative one. The tree will be maximal if it is of the form presented in figure 7. Let n be the number of negative monomials, m be the number of positive monomials occurring in a polynomial p and w.l.o.g. let $n = m + k$, $k \geq 0$[18]. Figure 7 represents the number of nodes at each level of the tree which corresponds to p. First of all, the number of negative monomials must[19] be reduced to the number of positive monomials (since $n \geq m$). Then, n and m must be alternately decreased by 1[20]. Note that, in figure 7 we will select m instead of n if $n = m$ holds.

Lemma 4.1 *Let p be a polynomial with m positive monomials and n negative monomials. For the worst case of algorithm $\mathcal{POSITIVE}$, $\Pi(p)$ and $\Phi(p)$ are as follows:*

$$\begin{aligned}\Pi(p) &= (m!)^3 \cdot n! \cdot m^{n-m-1} \\ \Phi(p) &= m \cdot n! \cdot \sum_{j=0}^{n-m} \frac{m^j}{(n-j-1)!} + \\ &\quad \frac{n! \cdot m^{n-m+1}}{(m-1)!} \cdot \sum_{j=1}^{2m-2} \prod_{i=1}^{j} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor)\end{aligned}$$

---

[13] One and the same polynomial can appear more than once in the tree.

[14] Note that this section is based on algorithm $\mathcal{POSIT}$ *including* backtracking.

[15] independent of $\mathcal{CHOOSE}$, i.e. regardless of the chosen sequence

[16] The condition $\alpha_{i_1\ldots i_n}\beta_{i_1\ldots i_n} = 0$ provides a normalized polynomial where, for example, $2x^2y - x^2y + \ldots$ is not allowed.

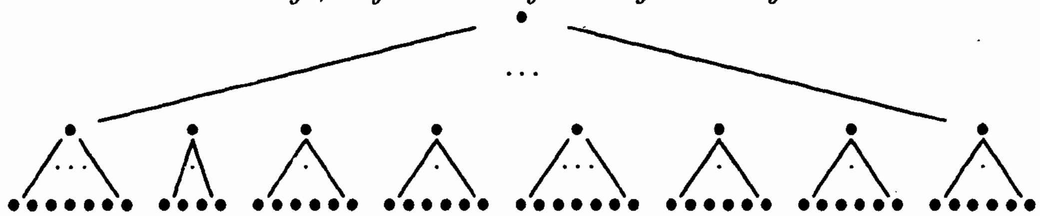
[17] This fact leads to the examination of all nodes of the corresponding tree

[18] see lemma 4.1

[19] in order to achieve the worst case

[20] This condition has to be guaranteed since for each real number $a > 0$, the following fact holds: $(\forall b \neq 0)\ a^2 > (a-b)(a+b)$, since $(a-b)(a+b) = a^2 - b^2$.

$3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$



								1
								8
								48
35	*12*	*25*	*25*	*32*	*24*	*25*	*25*	**203**
109	*24*	*64*	*64*	*91*	*54*	*64*	*64*	**534**
162	*24*	*91*	*91*	*148*	*54*	*87*	*87*	**744**
124	*0*	*43*	*43*	*96*	*6*	*59*	*59*	**430**

The following diagram provides a more detailed representation of the above tree. The nodes (i.e. the polynomials of a sequence) are identified by numbers. Each number of a node corresponds to its occurrence w.r.t. the tree. Thus, two identical numbers characterize the same subtree. The trees corresponding to the bold typed numbers are explicitly given, whereas the trees corresponding to the normal typed numbers are only pointed to. The vertical bars separate classes of subtrees belonging to different roots. The sign ⊕ (⊖) stands for a leaf of a successful (non-provable) path w.r.t. $\mathcal{POSITIVE}$.

0

1 2 3 3 4 5 6 6

11 12 12 13 1· 15 15 | 21 22 22 22 | 12 32 33 34 35 36 | 13 21 34 34 44 45 45 | 14 35 35 44 54 54 | 15 22 3· 45 54 66

111 112 113 113 | 121 122 123 124 125 | ·11 123 123 133 134 134 | 124 124 133 143 143 | 113 125 134 143 155 | 211 211 211 | 211 222 222 | 222 222 322 | 122 332 333 334 | 123 222 333 344 345 | 124 334 344 354 | 125 322 345 354 | 133 344 344 | 134 211 345 454 455 | 143 354 543 544 | 155 222 455 544

1111 1112 1112 | 1111 1122 1122 | 1112 1122 1133 | 1122 1212 | 1221 1222 1223 | 1231 1222 1233 1234 | 1223 1233 1243 | 1212 1234 1243 | 1233 1233 | 1112 1234 1343 1344 | 1243 1432 ⊖ | 1133 1344 ⊖ | 2111 2111 | 2111 2222 | 2222 2222 | 3321 2222 | 1222 3321 3333 | 1223 3333 | 1233 3333 | 1234 2222 3453 | 1243 3542 | 3453 3453 1343 | 1344 2111 4553 | 1432 3542 | ⊖

11111 11111 | 11111 11122 | 11221 | 11122 11221 | 12121 11221 | 12211 11221 | 12211 ⊕ | 11122 ⊕ | 11111 12121 | ⊕ | 12121 12342 | ⊖ | 13431 12342 | 11122 13442 | ⊖ | ⊖ | ⊖ | ⊖ | ⊕ | 12342 34532 | ⊖ | 13442 34532

⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕

Figure 6: Number of paths for the polynomial of example 4.1

$$1$$

$$nm$$

$$nm(n-1)m$$

$$nm(n-1)m(n-2)m$$

$$\vdots$$

$$nm(n-1)m(n-2)m \cdot \ldots \cdot (n-k)m =: l$$

$$l \cdot (m-1)m$$

$$l \cdot (m-1)m(m-1)(m-1)$$

$$\vdots$$

$$l \cdot (m-1)m(m-1)(m-1) \cdot \ldots \cdot (m-i)(m-i)$$

$$l \cdot (m-1)m(m-1)(m-1) \cdot \ldots \cdot (m-i)(m-i)(m-i-1)(m-i)$$

$$l \cdot (m-1)m(m-1)(m-1) \cdot \ldots \cdot (m-i)(m-i)(m-i-1)(m-i)(m-i-1)(m-i-1)$$

$$\vdots$$

$$l \cdot (m-1)m(m-1)(m-1) \cdot \ldots \cdot (m-i)(m-i)(m-i-1)(m-i)(m-i-1)(m-i-1) \cdot \ldots \cdot 1 \cdot 1$$

Figure 7: The worst case for procedure $\mathcal{POSITIVE}$

These two expressions are valid if $n \geq m$. In order to get $\Pi(p)$ and $\Phi(p)$ for $m > n$, we have to interchange the values of m and n, only.

A polynomial composed of 4 negative and 3 positive monomials can correspond to a tree with 5184 leaves and 13729 nodes (in the worst case). A typical example of a tree for representing the worst case of algorithm $\mathcal{POSITIVE}$ is the following one.

Example 4.3 *Let $p = 3x^2y + 3xy^2 - 4xy - 8x - 8y$ and $\mu = 2$. The symmetric tree of figure 8 corresponds to p. Note that all paths end with 0, i.e. p has a root at the position $(2,2)$ and thus $p \not\succ 0$ for $\mu = 2$.*

The following lemma provides a rough lower and a rough upper bound for the number $\Phi(p)$ of nodes determined in lemma 4.1.

Lemma 4.2 *Let $\Phi(p)$ be defined as in lemma 4.1. Then,*

$$\begin{array}{ll} \Phi(p) = 1 & \text{if } n = m = 1 \text{ or} \\ 2\Pi(p) \leq \Phi(p) \leq 3\Pi(p) & \text{otherwise.} \end{array}$$

The bounds of lemma 4.2 define the time complexity (for the worst case) of procedure $\mathcal{POSITIVE}$ (*with* backtracking) as $O(m! \cdot n! \cdot m^{n-m})$. By using the formula of Stirling[21], the complexity changes to $O((mn)^n)$.

We believe that, in many cases, procedure $\mathcal{CHOOSE}$ will directly generate a successful sequence proving the positiveness of a polynomial p if p is positive. In these cases, the time complexity of procedure $\mathcal{POSITIVE}$ is $O(k \cdot log\, k)$ where k represents the number of monomials occurring in p. More precisely, the number of calls of procedure $\mathcal{CHANGE}$ lies between n and $m+n-1$ (where n and m are the numbers of negative and positive monomials). The case $m+n-1$ will be achieved if $m-1$ positive monomials are covered by a part of the negative ones and the m^{th} positive monomial covers the remaining negative monomials. Procedure $\mathcal{SORT\text{-}MON}$[22] has the complexity[23] $(n \cdot log\, n + m \cdot log\, m) \cdot v$ where v is the number of different variables occurring in p.

[21] $n! \approx (\frac{n}{e})^n \cdot \sqrt{2\pi n}$.

[22] which represents the main part of procedure $\mathcal{CHOOSE}$

[23] if, for example, a heap sort algorithm is used

Note that this complexity can only be achieved if procedure $\mathcal{CHOOSE}$ *directly* finds a successful path through the corresponding tree of the polynomial. However, it is obvious that there exist positive polynomials (in the sense of $\mathcal{POSITIVE}$ *with* backtracking) that cannot be proved positive (*without* backtracking) by use of $\mathcal{CHOOSE}$:

Example 4.4 *Let $p = x^4y^2 + 3x^3yz - 8x^3 - 4x^2y^2$ and $\mu = 2$. Procedure $\mathcal{CHOOSE}$ creates the sequence*

$$\begin{array}{lll} p_0 & = & \mathbf{x^4y^2} + 3x^3yz - \mathbf{8x^3} - 4x^2y^2 \\ p_1 & = & 3x^3yz - 4x^2y^2 \end{array}$$

such that p_1 cannot be proved to be positive. However, the sequence

$$\begin{array}{lll} p_0 & = & \mathbf{x^4y^2} + 3x^3yz - 8x^3 - \mathbf{4x^2y^2} \\ p_1 & = & \mathbf{3x^3yz - 8x^3} \\ p_2 & = & x^3yz \end{array}$$

ends with a polynomial which is obviously positive.

5 The 'No Constants'-Case

This section deals with the effect of signatures *without* constant symbols on algorithm $\mathcal{POSITIVE}$. As mentioned in section 2, the constants of $\mathcal{F}$ are strongly influencing the application of polynomial orderings in general, as well as that of method $\mathcal{POSITIVE}$ in particular (see remark 2.1). For example, increasing the interpretations of constant symbols can lead to the orientation of an equation or to an easier interpretation of other operators.

Example 5.1 *Let*

$$\mathcal{R} = \{(x.y) \circ z \rightarrow x.(y \circ z)\}$$

and $\mu = 1$. With the help of the interpretations $[\circ](x,y) = x^2y$ and $[.](x,y) = x + 2y$, $\mathcal{R}$ can be oriented in the presented way. However, setting μ to 2, the interpretations $[\circ](x,y) = xy$ and $[.](x,y) = x + 2y$ are sufficient. Note that the latter interpretations cannot orient $\mathcal{R}$ if $\mu = 1$.

In the case of $\mathcal{R}$ not containing any constants, algorithm $\mathcal{POSITIVE}$ requires the minimum μ[24] to be set. However, if algorithm $\mathcal{POSITIVE}$ is changed such that μ is no longer considered, the resulting method is not only more efficient, but also more powerful. The improvement is based on the following observation.

[24] Remember that the original method sets μ to 2.

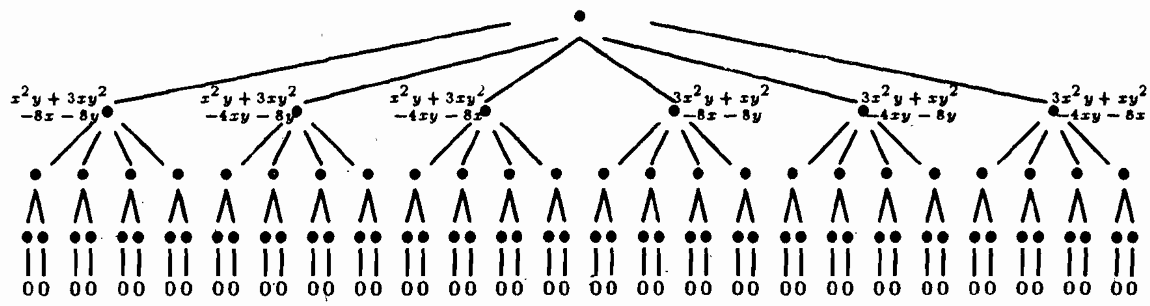


Figure 8: The paths for the polynomial of example 4.3

Remark 5.1 *The monomial x^2y is greater than the monomial $100xy$ since there* **exists** *a μ ($= 101$) such that $(\forall x, y \geq \mu)\, x^2y > 100xy$. Thus, the knowledge of the* **existence** *of such a μ is sufficient to guarantee the positiveness.*

The following definition generalizes the brief comment of remark 5.1.

Definition 5.1 (Ordering on polynomials)

- *Let $m_1 = \alpha_1 x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ and $m_2 = \alpha_2 x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$ be two monomials such that $\alpha_1, \alpha_2 \geq 0$. Then,*

  $m_1 \succ_M m_2$ *iff*
  $(\forall j \in [1,n])\ i_j \geq k_j \ \wedge\ (\exists l \in [1,n])\, i_l > k_l$
  or $(\forall j \in [1,n])\ i_j = k_j \ \wedge\ \alpha_1 > \alpha_2$

  $m_1 =_M m_2$ *iff*
  $(\forall j \in [1,n])\ i_j = k_j \ \wedge\ \alpha_1 = \alpha_2$

- *Let $p_1 = \sum_{j=1}^{l_1} m_{1j}$ and $p_2 = \sum_{j=1}^{l_2} m_{2j}$ with m_{1j}, m_{2j} being monomials with non-negative coefficients. Then*[25],

  $p_1 \sqsupset_P p_2$ *iff*
  $\{m_{11}, \ldots, m_{1l_1}\} \succ\!\succ_M \{m_{21}, \ldots, m_{2l_2}\}$

In order to prove the positiveness of a polynomial p, we consider p of the form $q - r$ where both q and r are two polynomials with positive coefficients, exclusively. This way, p is positive, if $q \sqsupset_P r$.

Example 5.2 *Let $p_1(x,y) = x^2y + xy^2$ and $p_2 = 2x^2 + 3xy + 5y^2$. Then, $p_1 \sqsupset_P p_2$ since $x^2y \succ_M 2x^2, x^2y \succ_M 3xy$ and $xy^2 \succ_M 5y^2$. Note that algorithm $\mathcal{POSITIVE}$ cannot prove the positiveness of $p_1 - p_2$ if $\mu \leq 5$.*

The most important feature of the definition of $\sqsupset_P$ is that it does not use μ, explicitly[26]. However, if there exists just one constant symbol in $\mathcal{R}$, then μ has to be set to a value and we have to check $p > 0$ w.r.t. μ. Thus, $\sqsupset_P$ can only be applied if $\mathcal{R}$ does not contain any constants:

Lemma 5.1 *The ordering $\succ_P$ defined by $s \succ_P t$ iff $[s] \sqsupset_P [t]$ is a reduction ordering on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ if no constant symbols in $\mathcal{F}$ exist.*

$\sqsupset_P$ is a generalization of the ordering realized by algorithm $\mathcal{POSITIVE}$ if no constant symbols exist (see example 5.2). Therefore, *the occurrence of constants reduces the power of polynomial orderings.* With the help of this observation, one could replace all constant symbols by new[27] variables before applying the polynomial ordering. This pre-processing is *always* helpful if the constant symbols occur in the presumably greater sides of the rules, only. Consider the following example.

Example 5.3 *Let $\mathcal{R}$ be composed of the simple rules*

$$\begin{array}{lcll} 0 + y & \rightarrow & y & (*) \\ s(x) + y & \rightarrow & s(x+y) & \end{array}$$

specifying the addition on natural numbers. The interpretations $[+](x,y) = 2x + y$ and $[0]() = 1$ enable

[25] The ordering $\succ\!\succ$ denotes the multiset extension of $\succ$.

[26] Definition 5.1 *implicitly* depends on the *existence* of a μ.

[27] concerning each rule

the orientation of the above rules. The same interpretation (with the same μ, which is equal to 1) can also orient $\mathcal{R}$ if (*) is replaced by $x + y \to y$.

The replacement of constants, occurring on the presumably smaller sides of the rules, by variables[28] can lead to the necessity of changing certain interpretations to more complex ones.

Example 5.4 *Consider the rule*

$$divp(x, y) \to rem(x, y) \equiv 0$$

which can be oriented using $[rem](x, y) = x + y$, $[\equiv](x, y) = x + y$, $[0]() = 1$ and $[divp](x, y) = x + y + 2$. $[divp](x, y)$ has to be set to $x + 2y + 1$ if 0 is replaced by y.

In some cases, the substitution of constants, that occur on the presumably smaller sides of the rules, by variables can generate rules which can no longer be oriented:

Example 5.5 *Let $\mathcal{R}$ be*

$$\{ \begin{array}{lll} f(x,x) & \to & h(x), \\ h(a) & \to & f(a,b) \end{array} \}.$$

With the help of the interpretations $[f](x, y) = xy + 1, [h](x) = x^2, [a]() = 2$ and $[b]() = 1$, the rules can be directed in the desired way. However, in order to replace the constants of the second rule by variables, we have to use the substitution $\sigma = \{a \leftarrow x, b \leftarrow x\}$. Thus, the new rules of $\mathcal{R}$ cannot be oriented in the required directions since $\{f(x, x) \to h(x), h(x) \to f(x, x)\}$ is not terminating[29].

Note that it is always more convenient to apply $\sqsupset_P$ (i.e. definition 5.1) instead of algorithm $\mathcal{POSITIVE}$ to a system $\mathcal{R}$ containing no constants on the presumably smaller sides, since definition 5.1 is more powerful (in this case) than algorithm $\mathcal{POSITIVE}$ (even *with* backtracking) as well as more efficient.

[28] of the presumably greater sides

[29] see also $\{f(a) \to f(b)\}$.

6 A Modification of the Approach of BenCherifa and Lescanne

In this section, we present a deterministic version of the technique contained in [BL87]. The backtracking component will be replaced by a set of constraints. The reasons for modifying the BenCherifa/Lescanne approach are the following ones: Since there exist only heuristics for procedure $\mathcal{CHOOSE}$ we possibly need to backtrack in order to find a successful path (and this considerably extends the time complexity of algorithm $\mathcal{POSITIVE}$, see section 4). Another more important aspect concerns the generation of an interpretation for a given TRS (see [Ste91]). More precisely, given variable interpretations[30] and rules $l_i \to r_i$, we have to choose the right coefficients such that $[l_i] - [r_i] \sqsupset 0$ holds. This problem cannot directly be solved with the help of algorithm $\mathcal{POSITIVE}$ of section 3, but by using the following algorithm. First of all, we illustrate the basic ideas of our derived algorithm.

- One of the main features of our algorithm is identical to the basic idea of the BenCherifa and Lescanne method:

  $x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \geq \mu^{\sum(k_j - i_j)} \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$
  if $(\forall j)\, k_j \geq i_j$

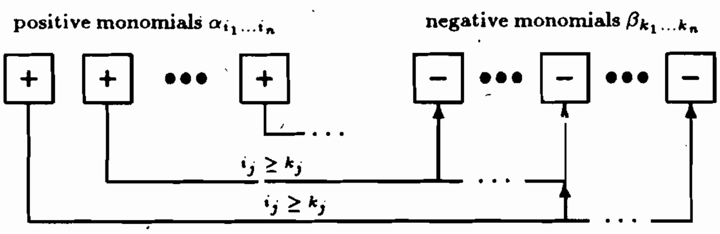
  This inequality is used in ii) of the following item.

- The algorithm transforms a polynomial p (that we want to be proved positive) into a set of linear[31] inequalities. Let $p = \sum(\alpha_{i_1 \ldots i_n} - \beta_{i_1 \ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ such that $\alpha_{i_1 \ldots i_n} \geq 0, \beta_{i_1 \ldots i_n} \geq 0$ and $\alpha_{i_1 \ldots i_n} \cdot \beta_{i_1 \ldots i_n} = 0$. The transformation is based on the following steps:

  i) Dividing: Each positive coefficient $\alpha_{i_1 \ldots i_n}$ will be split into a sum of new variables $\gamma_{i_1 \ldots i_n k_1 \ldots k_n}$ over $\mathbb{N}$ (or $\mathbb{R}$) such that each item of the sum corresponds to a negative coefficient $\beta_{k_1 \ldots k_n}$ if, and only if, $i_j \geq k_j$

[30] i.e. polynomials in which the coefficients are also variables

[31] The inequalities are of linear form if the interpretations are given, i.e. if the coefficients are *not* variables.



A positive monomial is represented by [+] whereas [−] stands for a negative monomial. Each [+] (i.e. the coefficient of each [+]) is able to cover those [−], of which the exponents (of the corresponding variables) are not greater ($i_j \geq k_j$). Thus, we divide the coefficient of each [+] into variables, where each variable corresponds to a negative monomial [−] (which is coverable by [+]).

Figure 9: Division process of algorithm 6.1

holds for all j. Figure 9 provides a graphical illustration of this process.

ii) **Distributing:** For each negative coefficient $\beta_{k_1\ldots k_n}$ we create an inequality of the form

$$\sum \mu^{\sum(i_j-k_j)} \cdot \gamma_{i_1\ldots i_n k_1\ldots k_n} \geq \beta_{k_1\ldots k_n}.$$

Note that $\gamma_{i_1\ldots i_n k_1\ldots k_n}$ is part of the positive monomial $\alpha_{i_1\ldots i_n}$. It has been generated for covering $\beta_{k_1\ldots k_n}$. Because of the fact that $i_j \geq k_j$ holds (see i)), we may multiply $\gamma_{i_1\ldots i_n k_1\ldots k_n}$ by the difference of the exponents of $\alpha_{i_1\ldots i_n}$ and $\beta_{k_1\ldots k_n}$.

iii) **Solving:** The set of the inequalities generated in i) and ii) can be solved or disproved by applying a decision procedure[32] for linear inequalities. For solving linear inequalities we can use, for example, the first phase of the Simplex method (see, for example, [Mit76], [Thi79], [Chv83]) or any other, more efficient and adequate technique.

The formal description of the above ideas can be found in algorithm 6.1 which is contained in figure 10. A detailed example, illustrating the steps of this algorithm, is presented in figure 11.

The additional condition 'I $I_1 = I_1' \cup \ldots$' of step 1. in algorithm 6.1 refers to the case that a positive monomial can cover only one negative monomial. For example, the polynomial

$$p = 2x^2 + 5y^2 - 3x - 2$$

implies that y^2 covers only -2. Then, the algorithm generates the sets $I_1 := \{2 \geq \gamma_{20_{10}} + \gamma_{20_{00}}, \gamma_{02_{00}} := 5\}$ and $I_2 := \{\mu \cdot \gamma_{20_{10}} \geq 3, \mu^2 \cdot \gamma_{20_{00}} + \mu^2 \cdot 5 \geq 2\}$.

Example 6.2 *We once again consider example 4.4. It has been shown that the polynomial*

$$p = x^4y^2 + 3x^3yz - 8x^3 - 4x^2y^2$$

cannot be proved to be positive w.r.t. $\mu = 2$ with the help of procedure $\mathcal{CHOOSE}$. It is very easy to prove the positiveness of p by using algorithm 6.1:

$$\begin{aligned}
I_1 &:= \{\ 1 \geq \gamma_{420_{300}} + \gamma_{420_{220}}, \\
&\qquad \gamma_{311_{300}} := 3\ \} \\
I_2 &:= \{\ 8\gamma_{420_{300}} + 4\gamma_{311_{300}} \geq 8\ , \\
&\qquad 4\gamma_{420_{220}} \geq 4\ \} \\
I &:= \{\ 1 = \gamma_{420_{300}} + \gamma_{420_{220}} + u_1, \\
&\qquad \gamma_{311_{300}} = 3, \\
&\qquad 2\gamma_{420_{300}} + 3 = 2 + u_2, \\
&\qquad \gamma_{420_{220}} = 1 + u_3, \\
&\qquad u_1 + u_2 + u_3 > 0\ \}
\end{aligned}$$

$\gamma_{420_{300}} = 0, \gamma_{420_{220}} = 1, \gamma_{311_{300}} = 3, u_1 = 0, u_2 = 1, u_3 = 0$ *represents a solution of I.*

In step 3. of algorithm 6.1, a strict inequality ($u_1 + u_2 + \ldots > 0$) must be added to I. Unfortunately, the Simplex method cannot directly deal

[32]This procedure eventually generates values for the variables $\gamma_{i_1\ldots i_n k_1\ldots k_n}$.

[33]see definition 2.2

[34]We do not need to explicitly compute a solution of the set of inequalities. It is sufficient to know that there exists a solution.

Algorithm 6.1 *This algorithm determines whether a polynomial p is positive in the sense of* $\sqsupset$[33]. *Let* $p = \sum(\alpha_{i_1\ldots i_n} - \beta_{i_1\ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ *such that* $\alpha_{i_1\ldots i_n} \geq 0, \beta_{i_1\ldots i_n} \geq 0, \alpha_{i_1\ldots i_n} \cdot \beta_{i_1\ldots i_n} = 0$ *and let* $\gamma_{l_1\ldots l_{n k_1\ldots k_n}}$, u_i *be new variables over* $\mathbb{R}$. *As usual,* μ *denotes the minimum of* $\mathcal{M}$.

1. **Dividing:** *Let* $I_1 := \{\alpha_{i_1\ldots i_n} \geq \sum_{k_j \leq i_j} \gamma_{i_1\ldots i_{n k_1\ldots k_n}} \mid \alpha_{i_1\ldots i_n} > 0, \beta_{k_1\ldots k_n} > 0\}$.
   If $I_1 = I_1' \cup \{\alpha_{i_1\ldots i_n} \geq \gamma_{i_1\ldots i_{n k_1\ldots k_n}}\}$ *then let* $I_1 := I_1' \cup \{\gamma_{i_1\ldots i_{n k_1\ldots k_n}} := \alpha_{i_1\ldots i_n}\}$.
2. **Distributing:** *Let* $I_2 := \{\sum_{k_j \geq i_j} \mu^{\sum(k_j - i_j)} \cdot \gamma_{k_1\ldots k_{n i_1\ldots i_n}} \geq \beta_{i_1\ldots i_n} \mid \beta_{i_1\ldots i_n} > 0, \alpha_{k_1\ldots k_n} > 0\}$
3. **Solving:** *Let* $I := \{l_i = r_i + u_i \mid l_i \geq r_i \in I_1 \cup I_2\} \cup \{\gamma_{i_1\ldots i_{n k_1\ldots k_n}} \geq 0, u_i \geq 0, \sum u_i > 0\}$. *Generating the values for* $\gamma_{i_1\ldots i_{n k_1\ldots k_n}}$, u_i *of I with the help of a decision procedure for linear inequalities*[34]. $p \sqsupset 0$ *w.r.t.* μ *if I has a solution.*

Figure 10: Deterministic version of algorithm $\mathcal{POSITIVE}$

with such inequalities. Thus, an implementation of algorithm 6.1 has to take this fact into consideration, and must

transform	$u_1 + u_2 + \ldots > 0$
into	$u_1 + u_2 + \ldots \geq lpf$

where lpf (i.e. least-positive-float) represents the positive floating-point number closest in value to (but not equal to) zero (provided by the implementation).

Theorem 6.1 *Algorithm 6.1 always terminates. If it does not fail*[35], $p \sqsupset 0$ *holds.*

Our technique enables the generation of a correct sequence $p = p_0 \sqsupseteq p_1 \sqsupseteq \ldots \sqsupseteq p_n \sqsupset 0$ for the BenCherifa & Lescanne approach. The deduced solution $\bigcup\{\gamma_{k_1\ldots k_{n i_1\ldots i_n}}\}$ of algorithm 6.1 will be used in the following way: The part $\gamma_{k_1\ldots k_{n i_1\ldots i_n}}$ of $\alpha_{k_1\ldots k_n}$ will be taken to cover $\beta_{i_1\ldots i_n}$, i.e. each element of the set $\bigcup\{\gamma_{k_1\ldots k_{n i_1\ldots i_n}}\}$ will be considered. Each item $\gamma_{k_1\ldots k_{n i_1\ldots i_n}} > 0$ corresponds to a transition $p_j \rightarrow p_{j+1}$ where the negative monomial $\beta_{i_1\ldots i_n}$ and the positive monomial $\alpha_{k_1\ldots k_n}$ will be added. Figure 12 contains a successful sequence for the polynomial already considered in example 6.1. This sequence is constructed by algorithm 6.1.

[35] Algorithm 6.1 will fail, if there is no solution for the set I of part 3.

[36] This implies splitting $3x^2y$ into $\mathbf{1x^2y} + 2x^2y$.

[37] We skip over $\gamma_{21_{11}}$ since its value is zero.

It is obvious that the more zeros the solution vector of algorithm 6.1 contains[38] the shorter the associated sequence, generated by algorithm $\mathcal{POSITIVE}$, will be.

Note that the technique of BenCherifa and Lescanne *with* backtracking and the algorithm of this section have the same power, i.e. whenever a polynomial can be proved to be positive using procedure $\mathcal{POSITIVE}$ it can also be proved positive with the help of algorithm 6.1, and vice versa. In general, algorithm 6.1 is more powerful than the BenCherifa & Lescanne approach *without* backtracking (see example 6.2).

7 A Modified Simplex Algorithm

An important part of algorithm 6.1 consists of solving linear inequalities. In this section we give an informal description of a special version of the well-known Simplex method to solve this problem. The Simplex method[39] minimizes (or maximizes) a linear expression constricted by linear equalities (or linear inequalities) that are sometimes called constraints. Such problems have come to be known as *Linear Programming*. The following approach is taken from [Thi79], [Mit76] and [Chv83] and the

[38] i.e. $\gamma_{k_1\ldots k_{n i_1\ldots i_n}} = 0$

[39] proposed by Dantzig in 1947

Example 6.1 (Example 3.1 revisited) *In order to prove $p = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$ to be positive w.r.t. $\mu = 3$, we perform the following actions:*

1. **Dividing:**

$$I_1 := \{ \begin{array}{l} 3 \geq \gamma_{21_{20}} + \gamma_{21_{11}} + \gamma_{21_{10}} + \gamma_{21_{01}}, \\ 6 \geq \gamma_{12_{02}} + \gamma_{12_{11}} + \gamma_{12_{10}} + \gamma_{12_{01}} \ \} \end{array}$$

Consider the first inequality of I_1: The coefficient of the positive monomial $3x^2y$ is split into the sum $\gamma_{21_{20}} + \ldots$ because x^2y can cover the negative monomials $2x^2, 12xy, 9x$ and $9y$.

2. **Distributing:**

$$I_2 := \{ \begin{array}{lcl} 3\gamma_{21_{20}} & \geq & 2, \\ 3\gamma_{12_{02}} & \geq & 6, \\ 3\gamma_{21_{11}} + 3\gamma_{12_{11}} & \geq & 12, \\ 9\gamma_{21_{10}} + 9\gamma_{12_{10}} & \geq & 9, \\ 9\gamma_{21_{01}} + 9\gamma_{12_{01}} & \geq & 9 \ \} \end{array}$$

We describe the first inequality of I_2: The variable $\gamma_{21_{20}}$ is part of the coefficient that corresponds to the monomial $3x^2y$ (see I_1). It is responsible for covering the coefficient of $-2x^2$. Since the variable y of $3x^2y$ does not occur in $-2x^2$, $\gamma_{21_{20}}$ can be multiplied by μ $(= 3)$.

3. **Solving:**

$$I := \{ \begin{array}{lcl} 3 = \gamma_{21_{20}} + \gamma_{21_{11}} + \gamma_{21_{10}} + \gamma_{21_{01}} + u_1, & & \\ 6 = \gamma_{12_{02}} + \gamma_{12_{11}} + \gamma_{12_{10}} + \gamma_{12_{01}} + u_2, & & \\ 3\gamma_{21_{20}} & = & 2 + u_3, \\ \gamma_{12_{02}} & = & 2 + u_4, \\ \gamma_{21_{11}} + \gamma_{12_{11}} & = & 4 + u_5, \\ \gamma_{21_{10}} + \gamma_{12_{10}} & = & 1 + u_6, \\ \gamma_{21_{01}} + \gamma_{12_{01}} & = & 1 + u_7, \\ \gamma_{ij_{kl}} & \geq & 0, \\ u_i & \geq & 0, \\ u_1 + u_2 + \ldots + u_7 & > & 0 \ \} \end{array}$$

A decision procedure for linear inequalities solves the union $I_1 \cup I_2$. In order to guarantee $p \sqsupset 0$ (not only $p \sqsupseteq 0$), at least one inequality of $I_1 \cup I_2$ must be a proper one. Thus, each inequality $l_i \geq r_i$ will be transformed into $l_i = r_i + u_i$ such that $\sum u_i > 0$ holds.

A solution of I includes the following values: $u_3 = 1$, $(\forall i \in \{1,2,4,5,6,7\})\, u_i = 0$, $\gamma_{21_{20}} = 1$, $\gamma_{21_{11}} = 0$, $\gamma_{21_{10}} = 1$, $\gamma_{21_{01}} = 1$, $\gamma_{12_{02}} = 2$, $\gamma_{12_{11}} = 4$, $\gamma_{12_{10}} = 0$, $\gamma_{12_{01}} = 0$.

Figure 11: The application of algorithm 6.1 to the example of section 3

Example 6.3 (Example 6.1 revisited) *Let* $p = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$ *and* $\mu = 3$ *as given in the examples 3.1 and 6.1. Furthermore, we use the following solution created in example 6.1:*

$$\gamma_{21_{20}} = 1\,,\ \gamma_{21_{11}} = 0\,,\ \gamma_{21_{10}} = 1\,,\ \gamma_{21_{01}} = 1\,,\ \gamma_{12_{02}} = 2\,,\ \gamma_{12_{11}} = 4\,,\ \gamma_{12_{10}} = 0\,,\ \gamma_{12_{01}} = 0$$

This solution implies the following sequence associated to the BenCherifa/Lescanne approach:

$$p_0 = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$$

$\Downarrow\ \gamma_{21_{20}} = 1$[36]

$$p_0 = \mathbf{x^2y} + 2x^2y + 6xy^2 - \mathbf{2x^2} - 6y^2 - 12xy - 9x - 9y$$
$$p_1 = \tfrac{1}{3}x^2y + 2x^2y + 6xy^2 - 6y^2 - 12xy - 9x - 9y$$

$\Downarrow\ \gamma_{21_{10}} = 1$[37]

$$p_1 = \tfrac{1}{3}x^2y + \mathbf{x^2y} + x^2y + 6xy^2 - 6y^2 - 12xy - \mathbf{9x} - 9y$$
$$p_2 = \tfrac{1}{3}x^2y + x^2y + 6xy^2 - 6y^2 - 12xy - 9y$$
$$\Downarrow\ \gamma_{21_{01}} = 1$$
$$p_2 = \tfrac{1}{3}x^2y + \mathbf{x^2y} + 6xy^2 - 6y^2 - 12xy - \mathbf{9y}$$
$$p_3 = \tfrac{1}{3}x^2y + 6xy^2 - 6y^2 - 12xy$$
$$\Downarrow\ \gamma_{12_{02}} = 2$$
$$p_3 = \tfrac{1}{3}x^2y + \mathbf{2xy^2} + 4xy^2 - \mathbf{6y^2} - 12xy$$
$$p_4 = \tfrac{1}{3}x^2y + 4xy^2 - 12xy$$
$$\Downarrow\ \gamma_{12_{11}} = 4$$
$$p_4 = \tfrac{1}{3}x^2y + \mathbf{4xy^2} - \mathbf{12xy}$$
$$p_5 = \tfrac{1}{3}x^2y$$

Figure 12: Merging algorithm 6.1 and procedure $\mathcal{POSITIVE}$

reader is referred to these references for a more detailed description. The Simplex method can cope with the following problem:

$$\begin{array}{lll} \text{Minimize} & \sum_{i=1}^{n} c_i x_i & \quad (1) \\ \text{Subject to} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i & (i \in [1,m]) \\ & x_j \geq 0 & (j \in [1,n]) \end{array}$$

For simplicity of exposition we shall restrict ourselves to the form (1). It is not difficult to transform a more general form (in which equations of the form $\Sigma a_{ij} x_j = b_i$ as well as inequalities like $\Sigma a_{ij} x_j \geq b_i$ are possible) into the one used in (1). The problem of transforming a strict inequality $u_i < b_i$ into the form of (1) can be solved by adding the arithmetic inaccuracy[40] *lpf* of the used computer: $u_i < b_i \rightsquigarrow u_i + lpf \leq b_i$.

To transform (1) into an equivalent form in which the inequalities are replaced by equalities, m so-called *slack variables* $x_{n+1}, \ldots, x_{n+m}$ are introduced. Note that these have to be distinct from the n so-called *decision variables* in which the problem is defined[41]:

$$\begin{array}{lll} \text{Min.} & z = \sum_{i=1}^{n} c_i x_i & \quad (2) \\ \text{Subj.} & \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i & (i \in [1,m]) \\ & x_j \geq 0 & (j \in [1,n+m]) \end{array}$$

In a linear programming problem, the linear function z to be optimized is called the *objective function*. Any tuple $(x_1, \ldots, x_n)$ with non-negative coordinates that satisfies the system of constraints is called a *feasible solution* to the problem. Thus, the basic problem is to determine, from among the set of all feasible solutions, a tuple that minimizes the objective function. The Simplex method can *decide* whether a problem has, in fact, any feasible solution and, in addition, whether the objective function actually assumes a minimal value. Note, however, that the problem occurring in algorithm 6.1 consists of finding *any* solution of a system of linear equalities, i.e. we shall only study the system

$$\begin{array}{lll} \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i & (i \in [1,m]) & \quad (3) \\ x_j \geq 0 & (j \in [1,n+m]) & \end{array}$$

Solving such systems is no more difficult than solving linear programming problems: To find a solution of (3), or to establish its non-existence, we only need to consider the following problem description:

$$\begin{array}{lll} \text{Min.} & z = x_0 & \quad (4) \\ \text{Subj.} & \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} - x_0 = b_i & (i \in [1,m]) \\ & x_j \geq 0 & (j \in [0,n+m]) \end{array}$$

The basic step of the Simplex method is derived from the familiar *pivot operation* used to solve linear equations. The pivot operation consists of replacing a system of equations with an equivalent system in which a selected variable is eliminated from all but one of the equations.

Definition 7.1 (Pivoting) *Let*

$$\sum_{j=1}^{n} a_{kj} x_j + x_{n+k} - x_0 = b_k$$

be the k-th equality of (4). We choose any x_p *(*$p \in [1,n]$*) and rewrite it in terms of* x_{n+k}*, i.e.*

$$x_p = (b_k - x_{n+k} + x_0 - \sum_{j=1, j \neq p}^{n} a_{kj} x_j) / a_{kp}$$

Substituting x_p *in the other equations, a new set of equations is obtained. This operation represents a change of state and will be denoted by* **pivot(p, k)**.

It is easy to show that the solution set of the system of equations resulting from the pivot operation is identical to that of the original system. In general, repeated use of pivoting can lead to a system of equations whose solution set is obvious. Such a system (called *canonical form*) consists of n equations with n unknowns where each variable appears in one and only one equation, and has in that equation, one as its coefficient. However, in attempting to put the constraint system into canonical form, an arbitrary selection of decision variables could easily

[40] see section 6

[41] Multiply the i-th equation on both sides by -1, if b_i is negative. As a result of this, all the right-hand constants in the equality constraints become non-negative.

lead to a system with negative constant terms and thus to an associated solution that is not even feasible. Therefore, for solving the problem of section 6, it is not sufficient to arbitrarily use pivot operations (like in Gaussian elimination). The Simplex method cleverly applies a convenient pivot operation at the right time.

For this purpose, a technique for determining an initial feasible solution for an arbitrary system of equations must be developed[42]. The basic idea behind the method used to solve this problem is simple. We introduce a sufficient number of variables, called *artificial variables*, to put the system of constraints into canonical form with these variables as the decision variables. Then, we apply the Simplex method to a new objective function defined in such a way that its minimal value corresponds to a feasible solution of the original problem.

Definition 7.2 (Introducing artificial variables) *The transformation of system (4) into the following system containing the artificial variables* $x_{n+m+1}, \ldots, x_{n+2m}$ *will be denoted by canonical transformation.*

$$\begin{aligned} \sum_{j=1}^{n} a_{ij}x_j + x_{n+i} - x_0 + x_{n+m+i} &= b_i \qquad (5) \\ & (i \in [1,m]) \\ x_0 &= z \\ -\sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} - x_0\right) &= w - \sum_{i=1}^{m} b_i \end{aligned}$$

Note that the system of constraints of (5) is canonical with the artificial variables as decision variables. The new objective function $w = x_{n+m+1} + \ldots + x_{n+2m}$ is transformed into canonical form by subtracting each equation of the system of constraints from w.

If the pivot operations dictated by the problem of minimizing w are simultaneously performed on the equation z, which defines the original objective function, this function will be expressed in each step in terms of variables, which are not decision variables. Thus, if an initial basic feasible solution is found for w the Simplex method can immediately be applied to z.

[42]The procedure for solving this problem is often called *Phase 1* of the Simplex method.

Figure 13 contains algorithm 7.1 representing the steps of the first phase of the Simplex method, starting with a problem in canonical form (see (5)).

The original problem has a basic feasible solution if, and only if, the minimum value of w is zero. Note that p could be any column with a negative c_j term. The smallest c_j can reduce the total number of steps necessary to complete the problem. Furthermore, if the minimum of b_i/a_{ip} is attained in several rows, a simple instruction (such as choosing that row with the smallest index) can be used to determine the pivoting row.

The Simplex method presented in algorithm 7.1 is correct and terminates. However, certain complications can occur during the application of this procedure. For compactness, we would like to refer to the literature for a detailed description of these problems.

8 Integrating the Simplex Method into Algorithm 6.1

In section 7 we have presented the various methods needed for finding a solution of a system of linear equations. Before applying these methods to an example, we will construct an algorithm solving our problem of section 6. This algorithm is contained in figure 14.

Note that in algorithm 8.1, we omit the original objective function $z = x_0$. This function's only use is to justify the employment of the Simplex method for solving systems of linear inequalities. It is irrelevant for producing a basic solution of our problem.

At each step of the Simplex method, it is sufficient to know the coefficients of the variables in the system of equations. In particular, for computation by hand or simple computer implementations it is favourable to record this information, only. A representation known as *Contracted Tableau* or *Tucker-Diagram* is of the following form:

x_1	x_2	$\cdots$	x_n	
a_{11}	a_{12}	$\cdots$	a_{1n}	b_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$
a_{m1}	a_{m2}	$\cdots$	a_{mn}	b_m
c_1	c_2	$\cdots$	c_n	c

Algorithm 7.1 (Phase 1 of the Simplex algorithm)

$\mathcal{SIMPLEX}$ = **proc** *(system in canonical form)* **returns** *(string)*
  stop := **false**
  while (**not** *stop*) **do**
    if $(\forall j \in [1, n+m])\ c_j \geq 0$
    then *stop* := **true**
      return *'success'*
    else **if** $(\exists j \in [1, n+m])(\forall i \in [1, m])\ c_j < 0 \wedge a_{ij} \leq 0$
      then *stop* := **true**
        return *'failure'*
      else *pivot(p, k)*
        such that p represents the column with the smallest negative c_j
        and k is chosen such that $b_k/a_{kp} = min\{b_i/a_{ip} \mid a_{ip} > 0\}$
  end
end

Figure 13: The 1^{st} phase of the Simplex method

Algorithm 8.1

$\mathcal{SOLVE}$ = **proc** $(\sum_{j=1}^{n} a_{ij}x_j = b_i, i = 1, \ldots, m-1$ *: equation set,* $\sum_{j=1}^{n} a_{mj}x_j > 0$ *: inequality)*

Introducing slack variables (see (4)):
$$\sum_{j=1}^{n} a_{ij}x_j - x_0 = b_i \quad \text{for } i \in [1, m-1]$$
$$\sum_{j=1}^{n} a_{mj}x_j + x_1 - x_0 = lpf$$

Canonical transformation (see (5)):
$$\sum_{j=1}^{n} a_{ij}x_j - x_0 + x_{n+m+i} = b_i \quad \text{for } i \in [1, m-1]$$
$$\sum_{j=1}^{n} a_{mj}x_j + x_1 - x_0 + x_{n+2m} = lpf$$
$$-\sum_{i=1}^{m-1}(\sum_{j=1}^{n} a_{ij}x_j - x_0) - (\sum_{j=1}^{n} a_{mj}x_j + x_1 - x_0) = -lpf - \sum_{i=1}^{m-1} b_i$$

Applying algorithm 7.1
end

Figure 14: Connection between algorithm 6.1 and the Simplex method

The first m rows correspond to the system of constraints with the constant terms given in the last column. The last row corresponds to the equation defining the objective function with the constant term (on the right-hand side of that equation) in the last column. The z term of the objective function is suppressed from the tableau as it remains fixed throughout the application of the Simplex method.

Appendix B contains an application of the Simplex method to the set of inequalities of example 6.1, generated by the first two steps of algorithm 6.1.

The time complexity of algorithm 6.1 is mainly influenced by that of the Simplex method (i.e. algorithm 8.1). Obviously, the time complexity of the Simplex method strongly depends on the number of different variables as well as on the number of inequalities. The set of inequalities generated by algorithm 6.1 is of the form presented in figure 15.

As it is widely known, the time behaviour of the Simplex method is, in the worst case, exponential. However, we quote from [Sha87][44]:

Remark 8.1 (Time complexity)
Efficiency is usually measured for the Simplex method as the number of pivot steps (iterations) it requires to solve a problem, expressed as a function of the dimensions of the problem[...]. Here and throughout, we denote the dimensions of the problems differently for each form, in such a way that the following convention holds:

(1) n is the total number of inequalities (including nonnegativity constraints but not including equality constraints).

(2) d is the number of variables in an equivalent presentation of the problem without equality constraints[...].

(3) m := n - d.

We shall usually deal with the LPP *(i.e. Linear Programming Problem)* in the form

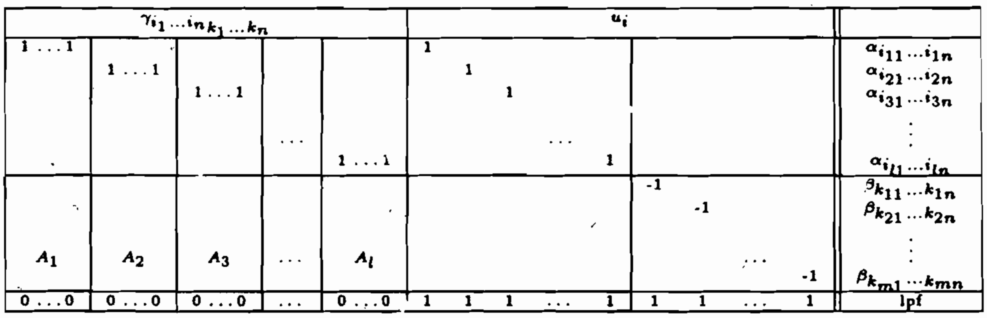
$$\begin{aligned} & \min c^T x \\ \text{such that}\quad & a_i^T x \geq b_i, \quad i = 1, \ldots, m, \\ & x \geq 0 \end{aligned}$$

Let n_p (m_p) be the number of negative (positive) monomials occurring in a given polynomial p. Then, the number m of inequalities generated by algorithm 6.1 is equal to $n_p + m_p + 1$ (I_1 contains m_p inequalities, I_2 contains n_p inequalities and $I = I_1 \cup I_2 \cup \{u_1 + u_2 + \ldots > 0\}$). Therefore, whenever m is used in the remaining part of this section, m is identical to the number of monomials of a polynomial plus 1.

[...]This function *(which usually measures the efficiency of the Simplex method)*, as observed by LP *(i.e. linear programming)* practitioners, is a low degree polynomial and perhaps even linear on most real-life problems[...]. Wolfe and Cutler (1963) experimented with nine 'real' LP problems. The average number of phase I iterations was 1.69m when a full artificial basis was used as a starting base, together with the steepest descent pivoting rule. When better starting bases were chosen, utilizing the sparsity of the data, that average went down to 0.56m. For the total number of iterations required, they obtained an average iterations count varying between 1.71m and 0.98m, depending on phase I and the pivoting rules. They concluded that the rule of '2m iterations' from folklore is fairly good, and that an estimate of between m and 3m iterations will almost be correct[...]. Recently, Ho and Loute (1983) reported on a set of experiments carried out with 30 large-scale problems[...]. So we see that these results, for problem dimensions in the thousands, still conform with the '2m iterations folklore', although there is some indication that the number of variables also influences the number of pivots[...]. We can summarize the observed behaviour of the Simplex method by the following recent words of Dantzig (1979). This appears to be a revision of his earlier summary, in view of the new evidence accumulated in 16 years that passed between the two quotations. *Dantzig:* "The expected number of steps to find a feasible solution to a linear program using phase I of the Simplex method, for moderately sized problems, is conjectured to be, of the order $\alpha \cdot$ m steps where m is the number of equations and α is typically 2 to 3 (or 4 to 6 for an optimal solution using both phase I and phase II). Thousands of linear programs are solved each day using some variant of the Simplex method – a value of $\alpha > 4$ is rarely seen[...]". This can be viewed as a summary on the average performance of the Simplex method on both real-life problems and artificially distributed ones.

[43] Note that all items of each matrix A_i are greater than or equal to zero.

[44] Additional explanations are added in italic type.



The diagram contains only the non-zero numbers. The upper part represents the division operation, whereas the middle sector describes the distribution. Each A_i is a matrix of the following form: Each row contains at most one item which is not equal to zero, whereas each column contains exactly one non-zero item[43]. All other items are equal to zero.

Figure 15: The set of inequalities generated by algorithm 6.1

Note that only the time complexity of the first phase of the Simplex method is relevant for that of algorithm 6.1. However, the time complexity of the Simplex method mainly depends on that of the second phase.

9 Improving the BenCherifa and Lescanne Method

We have implemented the method of [BL87] and integrated it in our completion environment COMTES ([AMS89])[45]. A series of 320 experiments considering more than 1700 rules occurring in the literature has been conducted (see also [SK90]). Most of these rule systems can be proved to be terminating by applying the method of BenCherifa and Lescanne. Certain examples require a more powerful ordering on polynomials. This can be illustrated by a simple TRS.

Example 9.1 *Let $\mathcal{R}$ be*

$$\begin{array}{lll} 1: & (x*y)*z & \to x*(y*z) \\ 2: & x*(y+z) & \to (x*y)+(x*z) \\ 3: & h(x)+h(y) & \to x*y \end{array}$$

In order to orient rule 2, $[*](x,y)$ needs to be mixed and thus $[+](x,y)$ must be of linear form[46] (see suggestion 3.2.10 of [SZ90]). For example, $[*](x,y) = xy + x$ and $[+](x,y) = x + y + 2$ will prove the termination of the first two rules if $\mu = 3$ (by using the method of [BL87]). This way, rule 3 cannot be oriented using the method of [BL87] (independent of $[h](x)$). Assuming the interpretation of h as x^2, the polynomial

$$x^2 - xy - x + y^2 + 2$$

has to be greater than zero. However, this cannot be proved with the help of [BL87] since there is no positive monomial which covers $-xy$.

The proof of $x^2 - xy - x + y^2 + 2 \sqsupset 0$ can be performed with the help of the fact $x^2 + y^2 \geq 2xy$, which is equivalent to $(x-y)^2 \geq 0$. The following lemma generalizes this inequality.

Lemma 9.1 *Let α, β, $x_l > 0$. Then,*

$$\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} + \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} \geq 2\sqrt{\alpha\beta} \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$$
if $(\forall l \in [1,n])\, k_l = \frac{i_l + j_l}{2}$

Example 9.2 *The above lemma can, for example, prove the following inequalities:*

- $x^2 + y^2 \geq 2xy$
- $x^3 + y^3 \geq 2x^{\frac{3}{2}} y^{\frac{3}{2}}$
- $x^2yz^3 + x^2y^3z \geq 2x^2y^2z^2$

[45] An implementation of the approach contained in [Rou88] is also available.

[46] $p(x_1, \ldots, x_n)$ is linear iff $p(x_1, \ldots, x_n) = \sum \alpha_i x_i + \beta$. $p(x_1, \ldots, x_n)$ is mixed iff there exists at least one monomial containing at least two different variables.

- $x^2 + 4y^2 \geq 4xy$

Procedure $\mathcal{POSITIVE}$ cannot show the validity of these inequalities. Note that it is possible to repeatedly apply lemma 9.1:

$$\begin{array}{ccc} \mathbf{x^4 + y^4} + 2z^2 & \quad & x^4 + \mathbf{y^4 + 2z^2} \\ \geq & & \geq \\ \mathbf{2x^2y^2 + 2z^2} & & \mathbf{x^4 + 2\sqrt{2}\cdot y^2z} \\ \geq & & \geq \\ 4xyz & & 2\sqrt{2\sqrt{2}}\cdot x^2yz^{\frac{1}{2}} \end{array}$$

The basic idea of lemma 9.1 can be extended in another way. The following lemma describes a situation where a nearl · arbitrary number of monomials for covering one monomial is admitted.

Lemma 9.2 *Let $n \geq m \geq 1$. Then, for all $x_i > 0$:*

$$\sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \geq m \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$$
$$\text{if} \quad (\forall l \in [1,n])\, k_l = \tfrac{1}{m} \cdot \sum_{j=1}^{m} i_{l_j}$$

The application of lemma 9.2 is based on the following consideration: If we want to cover a monomial $m \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$, a polynomial p consisting of m monomials must exist. Furthermore, the arithmetic mean of the sum of all exponents of p w.r.t. the variable x_i have to be identical to k_i.

Example 9.3 *The following inequalities can be proved by using lemma 9.2:*

- $x^3 + y^3 + z^3 \geq 3xyz$
- $x^3y + y^2z + z^2 \geq 3xyz$
- $x^2yz + y^3zu^2 \geq 2xy^2zu$

The class of polynomials that is successfully managed by the method corresponding to lemma 9.2 overlaps with that of lemma 9.1 (see example 9.2 and example 9.3) as well as with that of the BenCherifa & Lescanne algorithm. However, the combination of these processes is far more powerful than each method by itself. More precisely, the functions $\mathcal{POSITIVE}$, $\mathcal{CHOOSE}$ and $\mathcal{CHANGE}$ should be extended by incorporating a test embodying lemma 9.2 (and/or lemma 9.1). Note that the condition $(\forall l \in [1,n])\, k_l = \frac{1}{m} \cdot \sum_{j=1}^{m} i_{l_j}$ of lemma 9.2 can be achieved by substituting μ for each additional variable and splitting monomials w.r.t. coefficients[47]. For example, let $\mu = 2$. Then, $x^2y + y^2 > 2xy$ since $x^2y + y^2 = \frac{1}{2}x^2y + \frac{1}{2}x^2y + y^2 \geq \frac{1}{2}x^2y + x^2 + y^2, x^2 + y^2 \geq$[48] $2xy$ and $\frac{1}{2}x^2y > 0$.

Example 9.4 *Let $\mu = 2$. We show that $p = 2x^3 + y^2z^2 + yz^2 - x^2 - 3xyz - 2yz \sqsupset 0$ holds.*

$$\begin{array}{lll} p = p_0 & = & \mathbf{2x^3 + y^2z^2 + yz^2} - x^2 - \mathbf{3xyz} - 2yz \\ \sqsupseteq p_1{}^{49)} & = & x^3 + \frac{1}{2}\mathbf{y^2z^2} - x^2 - \mathbf{2yz} \\ \sqsupseteq p_2 & = & \mathbf{x^3 - x^2} \\ \sqsupseteq p_3 & = & \frac{1}{2}x^3 \end{array}$$

The sequence $2x^3 + \mathbf{y^2z^2} + yz^2 - x^2 - 3xyz - \mathbf{2yz}, \mathbf{2x^3} + \frac{1}{2}y^2z^2 + yz^2 - \mathbf{x^2} - 3xyz, \frac{3}{2}x^3 + \frac{1}{2}y^2z^2 + yz^2 - 3xyz, \frac{3}{2}x^3 + y^2z + yz^2 - 3xyz, \frac{1}{2}x^3$ is also successful.

A reduction of the time complexity of the extended procedure $\mathcal{POSITIVE}$ can be achieved by only applying the part corresponding to lemma 9.2 (lemma 9.1) in case the original algorithm fails (see example 9.4).

As we have seen (in example 9.1), lemma 9.2 (as well as lemma 9.1) extends the power of procedure $\mathcal{POSITIVE}$. In addition, its use can sometimes simplify the form of the interpretations of operators needed for a termination proof. Let us consider an example:

Example 9.5 *The TRS*

$$\mathcal{R} = \begin{cases} 1: & (x * y) * z \rightarrow x * (y * z) \\ 2: & i(x * y) \quad\;\; \rightarrow i(y) * i(x) \\ 3: & i(x) + i(y) \rightarrow x * y \end{cases}$$

needs a mixed interpretation for the operator $*$ which implies that $[+](x,y)$ must also be mixed for applying the original algorithm $\mathcal{POSITIVE}$. For example, the interpretations

$$[*](x,y) = 2xy + x, [i](x) = x^2 \textit{ and } [+](x,y) = xy$$

---

47) Analogous considerations are also helpful for lemma 9.1.

48) because of lemma 9.2

49) since $y^2z^2 \geq \frac{1}{2}y^2z^2 + y^2z$ and $x^3 + y^2z + yz^2 \geq 3xyz$.

will orient $\mathcal{R}$ with the help of $\mathcal{POSITIVE}$ if $\mu = 2$. The interpretation of $[+](x, y)$ can be replaced by $2x + y$ (which is simpler than xy) if we use the extended version of $\mathcal{POSITIVE}$.

Lemma 9.1 and lemma 9.2 provide a theoretical framework for extending procedure $\mathcal{POSITIVE}$. A thorough investigation of its effect on practical applications is part of future plans. It is obvious, that this examination presumes detailed considerations about the combination of lemma 9.2 (, lemma 9.1) and procedure $\mathcal{POSITIVE}$ for an efficient implementation.

References

[AMS89] Jürgen Avenhaus, Klaus E. Madlener, and Joachim Steinbach. COMTES – An experimental environment for the completion of term rewriting systems. In N. Dershowitz, editor, *Proc. 3rd RTA (LNCS 355)*, pages 542–546, Chapel Hill (North Carolina), April 1989.

[Ben86] Ahlem BenCherifa. *Preuves de Terminaison des Systèmes de Réécriture – Un Outil fondé sur les Interprétations polynomiales.* PhD thesis, Univ. of Nancy I, Dept. of Computer Science, Nancy (France), 1986.

[BL87] Ahlem BenCherifa and Pierre Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *SCP*, 9(2):137–160, Oct. 1987.

[Chv83] Vašek Chvátal. *Linear Programming.* W.H. Freeman and Company, New York/San Francisco (ISBN 0-7167-1195-8), 1983.

[Col75] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conference on Automata and Formal Languages*, pages 134–183, 1975.

[Dav73] Martin Davis. Hilbert's tenth problem is unsolvable. *Amer. Math. Monthly*, 80(3):233–269, March 1973

[Der79] Nachum Dershowitz. A note on simplification orderings. *IPL*, 9(5):212–215, 1979.

[Der83] Nachum Dershowitz. Well-founded orderings. Technical Report ATR-83(8478)-3, Office of Information Sciences Research, The Aerospace Corporation, El Segundo (California), 1983.

[Der87] Nachum Dershowitz. Termination of rewriting. *JSC*, 3:69–116, 1987.

[HLP52] G. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities (Second Edition).* Cambridge Univ. Press, 1952.

[HO80] Gérard Huet and Derek C. Oppen. Equations and rewrite rules: A survey. In R. Book, editor, *Formal Languages – Perspectives and Open Problems*, pages 349–405. Academic Press, 1980.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[Lan75a] Dallas S. Lankford. Canonical algebraic simplification in computational logic. Memo ATP-25, Univ. of Texas, Austin (Texas), 1975.

[Lan75b] Dallas S. Lankford. Canonical inference. Memo ATP-32, Univ. of Texas, Austin (Texas), 1975.

[Lan76] Dallas S. Lankford. A finite termination algorithm. Technical report, Southwestern Univ. of Georgetown, Georgetown (Texas), 1976.

[Lan77] Dallas S. Lankford. Some approaches to equality for computational logic: A survey and assessment. Report ATP-36, Dept. of Mathematics and Computer Science, Univ. of Texas, Austin (Texas), Spring 1977.

[Lan79] Dallas S. Lankford. On proving term rewriting systems are noetherian. Memo MTP-3, Louisiana Technical Univ., Dept. of Mathematics, Ruston (Louisiana), May 1979.

[Les86] Pierre Lescanne. Divergence of the Knuth-Bendix completion procedure and termination orderings. *Bulletin of the EATCS*, 30:80–83, 1986.

[Mit76] Gautam Mitra. *Theory and application of mathematical programming.* Academic Press Inc. (London) Ltd., 1976.

[MN70] Zohar Manna and Stephen Ness. On the termination of Markov algorithms. In *Proc. 3rd Int. Conf. System Science*, pages 789–792, Honolulu (Hawaii), 1970.

[Rou88] Jocelyne Rouyer. Preuves de Terminaison des Systèmes de Réécriture fondées sur les Interprétations polynomiales – Une Méthode basée sur le Théorème de Sturm. Internal Report, Univ. of Nancy, Nancy (France), 1988.

[Rou91] Jocelyne Rouyer. *Calcul formel en Géometrie algébrique réélle appliqué à la Terminaison des Systèmes de Réécriture.* PhD thesis, Univ. of Nancy I, Nancy (France), 1991.

[Sha87] Ron Shamir. The efficiency of the Simplex method: A survey. *Management Science*, 33(3), March 1987.

[SK90] Joachim Steinbach and Ulrich Kühler. Check your ordering – Termination proofs and open problems. SEKI-Report, Univ. of Kaiserslautern, Kaiserslautern (Germany), 1990.

[Ste91] Joachim Steinbach. Termination proofs of rewriting systems – Heuristics for generating polynomial orderings. SEKI-Report, Univ. of Kaiserslautern, Kaiserslautern (Germany), 1991. to appear. see also 'Towards the generation of polynomial orderings', 2nd Int. Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale (Florida), Jan. 1992.

[Sti76] Mark E. Stickel. The inadequacy of primitive recursive complexity measures for determining finite termination of sets of reductions. Unpublished Memo, 1976.

[SZ90] Joachim Steinbach and Michael Zehnter. Vade-mecum of polynomial orderings. Technical Report SR-90-03, Univ. of Kaiserslautern, Kaiserslautern (Germany), 1990.

[Tar51] Alfred Tarski. A decision method for elementary algebra and geometry. Univ. of California Press, Berkeley, 1951.

[Thi79] Paul R. Thie. *An introduction to linear programming and game theory.* John Wiley & Sons Inc. (ISBN 0-471-04248-X), 1979.

A Appendix : Proofs

Lemma A.1 *Let p be a polynomial with m positive monomials and n negative monomials. For the worst case of algorithm $\mathcal{POSITIVE}$, $\Pi(p)$ and $\Phi(p)$ are defined as follows:*

$$\begin{aligned}\Pi(p) &= (m!)^3 \cdot n! \cdot m^{n-m-1}\\ \Phi(p) &= m \cdot n! \cdot \sum_{j=0}^{n-m} \frac{m^j}{(n-j-1)!} + \frac{n! \cdot m^{n-m+1}}{(m-1)!} \cdot \sum_{j=1}^{2m-2} \prod_{i=1}^{j} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor)\end{aligned}$$

These two expressions are valid if $n \geq m$ holds. In order to get $\Pi(p)$ and $\Phi(p)$ for $m > n$, we have to interchange the values of m and n, only.

Proof A.1 *The number $\Pi(p)$ of paths in the tree corresponding to p is equivalent to the number of leaves in the tree. Therefore, according to figure 7,*

$$\begin{aligned}\Pi(p) &= l \cdot (\mathrm{m}-1)m(\mathrm{m}-1)(m-1) \cdot \ldots \cdot (\mathrm{m}-\mathrm{i})(m-i)(\mathrm{m}-\mathrm{i}-1)(m-i)(\mathrm{m}-\mathrm{i}-1)(m-i-1) \cdot 1 \cdot 1 \text{ with}\\ l &= \mathrm{n}m(\mathrm{n}-1)m(\mathrm{n}-2)m \cdot \ldots \cdot (\mathrm{n}-\mathrm{k})m \text{ and } k = n - m\end{aligned}$$

The non-italic expressions represent the number w.r.t. n[50]*. Thus, there are two sequences (i.e. products) of numbers (one sequence w.r.t. n and one sequence w.r.t. m):*

$$\begin{aligned}P1 &:= n(n-1)(n-2) \cdot \ldots \cdot (n-k)(n-k-1)(n-k-1) \cdot \ldots \cdot (n-k-i)(n-k-i) \cdot \ldots \cdot 1 \text{ and}\\ P2 &:= m^{k+1} m(m-1)(m-1) \cdot \ldots \cdot (m-i)(m-i) \cdot \ldots \cdot 1\end{aligned}$$

$$\begin{aligned}\leadsto\ & P1 = n! \cdot (n-k-1)! = n! \cdot (m-1)!\\ & P2 = m^{k+1} \cdot m! \cdot (m-1)! = m^{n-m+1} \cdot m! \cdot (m-1)!\\ \leadsto\ & \Pi(p) = P1 \cdot P2 = n! \cdot (m-1)! \cdot m^{n-m+1} \cdot m! \cdot (m-1)! = n! \cdot (m!)^3 \cdot m^{n-m+1-2}\end{aligned}$$

The number $\Phi(p)$ of nodes corresponds to the sum of all levels (except the first one) of figure 7. We split the sum (as well as the diagram) into two parts:

$$\begin{aligned}S1 &:= nm + nm(n-1)m + \ldots + nm(n-1)m(n-2)m \cdot \ldots \cdot (n-k)m\\ S2 &:= l(m-1)m + l(m-1)m(m-1)(m-1) + \ldots + l(m-1)m(m-1)(m-1) \cdot \ldots \cdot 1 \cdot 1\\ \leadsto\ & S1 = \sum_{j=0}^{k}((\prod_{i=0}^{j}(n-i)) \cdot m^{j+1}) = \sum_{j=0}^{k}(\frac{n!}{(n-j-1)!} \cdot m^{j+1}) = m \cdot n! \cdot \sum_{j=0}^{n-m} \frac{m^j}{(n-j-1)!}\end{aligned}$$

In order to compute $S2$ we set it to $S2 := l \cdot S2'$ and further split each item of $S2'$ into two products:

i	1^{st} *Part of S2'*	2^{nd} *Part of S2'*
1	**(m-1)**	**m**
2	**(m-1)** *(m-1)*	**(m-1)** *m*
3	**(m-2)** *(m-1)(m-1)*	**(m-1)** *(m-1)m*
4	**(m-2)** *(m-2)(m-1)(m-1)*	**(m-2)** *(m-1)(m-1)m*
5	**(m-3)** *(m-2)(m-2)(m-1)(m-1)*	**(m-2)** *(m-2)(m-1)(m-1)m*
⋮	⋮	⋮

The bold expressions are of the forms $(m - \lceil \frac{i}{2} \rceil)$ and $(m - \lfloor \frac{i}{2} \rfloor)$ and therefore

$$S2' = \sum_{j=1}^{2m-2} (\prod_{i=1}^{j} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor))$$

[50] Note that $m - i = n - k - i$.

It is obvious that $l = m^{k+1} \cdot n!/(n-k-1)! = m^{n-m+1} \cdot n!/(m-1)!$ *which leads to the final result*

$$\begin{aligned} S2 &= l \cdot S2' = \tfrac{m^{n-m+1} \cdot n!}{(m-1)!} \cdot \sum_{j=1}^{2m-2} \left(\prod_{i=1}^{j} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) \right) \quad \text{and} \\ \Phi(p) &= S1 + S2 \end{aligned}$$

□

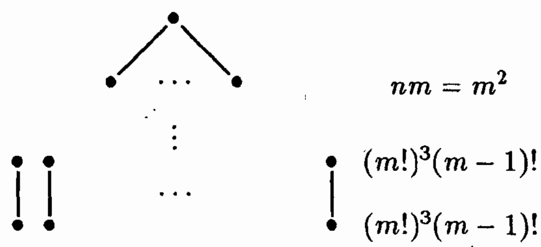
Lemma A.2 *Let* $\Phi(p)$ *be defined as in lemma 4.1. Then,* $\Phi(p) = 1$ *if* $n = m = 1$ *or* $2\Pi(p) \leq \Phi(p) \leq 3\Pi(p)$*, otherwise.*

Proof A.2 *The case* $n = m = 1$ *is obvious. Therefore, let* $n = m + k$, $k \geq 0$ *and not both* $m = 1$ *and* $k = 0$*. Furthermore, let*

$$\begin{aligned} \Pi(p) &= (m!)^3 \cdot n! \cdot m^{n-m-1} \\ \Phi(p) &= m \cdot n! \cdot \sum_{j=0}^{n-m} \tfrac{m^j}{(n-j-1)!} + \tfrac{n! \cdot m^{n-m+1}}{(m-1)!} \cdot \sum_{j=1}^{2m-2} \prod_{i=1}^{j} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) \end{aligned}$$

1. *We show that* $2\Pi(p) \leq \Phi(p)$*:*

   - $k = 0$, $m > 1$*: Note that this implies* $n = m$*.*



     The last two levels of the tree contain

$$\begin{aligned} \tfrac{n! \cdot m^{n-m+1}}{(m-1)!} \cdot \prod_{i=1}^{2m-2} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) &= (m!)^3 \cdot (m-1)! \quad \text{and} \\ \tfrac{n! \cdot m^{n-m+1}}{(m-1)!} \cdot \prod_{i=1}^{2m-3} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) &= (m!)^3 \cdot (m-1)! \end{aligned}$$

     nodes (see the expressions for $\Pi(p)$ *and* $\Phi(p)$ *of lemma 4.1).*
     $\leadsto \Phi(p) \geq 2\Pi(p)$

   - $k > 0$*: Assume that* $m = 1$*.*

$$\begin{aligned} \leadsto \quad \Pi(p) &= n! \quad \text{and} \\ \Phi(p) &= n! \cdot \sum_{j=0}^{n-1} \tfrac{1}{(n-j-1)!} = n! \cdot \left(\tfrac{1}{(n-1)!} + \tfrac{1}{(n-2)!} + \ldots + \tfrac{1}{1!} + \tfrac{1}{0!} \right) \\ \leadsto \quad \Phi(p) &\geq 2n! = 2\Pi(p) \end{aligned}$$

     The proof of the case "$m > 1$*" can be performed analogous to that of the case "*$k = 0$, $m > 1$*" (see figure).*

2. *We prove that* $\Phi(p) \leq 3\Pi(p)$*. This inequality is equivalent to the following one:*

$$\frac{\sum_{j=0}^{n-m} \frac{m^j}{(n-j-1)!}}{(m!)^3 \cdot m^{n-m-2}} + \frac{1}{((m-1)!)^3 \cdot m!} \cdot \sum_{j=1}^{2m-2} \prod_{i=1}^{j} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) \overset{!}{\leq} 3$$

With the help of the fact that

$$\prod_{i=1}^{2m-2} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) = \prod_{i=1}^{2m-3} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) = ((m-1)!)^3 \cdot m!$$

we have to prove that

$$\sum_{j=0}^{n-m} \frac{1}{(n-j-1)! \cdot ((m-1)!)^3 \cdot m^{n-m-j+1}} + \frac{1}{((m-1)!)^3 \cdot m!} \cdot \sum_{j=1}^{2m-4} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) \overset{!}{\leq} 1$$

Note that

- $\prod_{i=1}^{2m-4} (m - \lceil \tfrac{i}{2} \rceil)(m - \lfloor \tfrac{i}{2} \rfloor) = \frac{((m-1)!)^3 \cdot m!}{2}$
- $\frac{\prod_{i=1}^{j} (m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor)}{\prod_{i=1}^{j+1} (m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor)} \leq \frac{1}{2}$
- $\frac{1}{(n-j-1)! \cdot ((m-1)!)^3 \cdot m^{n-m-j+1}} : \frac{1}{(n-j-2)! \cdot ((m-1)!)^3 \cdot m^{n-m-j}} = \frac{1}{(n-j-1) \cdot m} \leq \frac{1}{2}$
- *The division of the greatest item of the sum* $\sum_{j=0}^{n-m} \frac{1}{(n-j-1)! \cdot ((m-1)!)^3 \cdot m^{n-m-j+1}}$ *and the smallest item of the sum* $\frac{1}{((m-1)!)^3 \cdot m!} \cdot \sum_{j=1}^{2m-4} \prod_{i=1}^{j} (m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor)$ *has the value*

$$\frac{1}{((m-1)!)^4 \cdot m} : \frac{(m-1) \cdot m}{((m-1)!)^3 \cdot m!} = \frac{1}{(m-1) \cdot m} \leq \frac{1}{2}$$ [51]

These four results directly imply the following fact:

$$\Phi(p) \leq 2 \cdot \frac{1}{((m-1)!)^3 \cdot m!} \cdot \frac{((m-1)!)^3 \cdot m!}{2} = 1$$

□

Lemma A.3 *The ordering $\succ_P$ defined by $s \succ_P t$ iff $[s] \sqsupset_P [t]$ is a reduction ordering on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ if no constant symbols in $\mathcal{F}$ exist.*

Proof A.3 *By the definition of $\succ_M$, for each $m_i \succeq_M m_j$ there exists a μ_{ij} such that $m_i \sqsupseteq m_j$ w.r.t. $\mathcal{M} = \{k | k \geq \mu_{ij}\}$. Let $\mu_i = \sum_{m_i \succeq_M m_j} \mu_{ij}$. This implies that $m_i' \sqsupseteq \sum_{j=1}^{l} m_j$ holds if $(\forall j \in [1, l])\, m_i' \sqsupseteq m_j$. Let $\mu = 1 + \max_i \mu_i$. Then, $\sum m_i' \sqsupset \sum m_j$ if $\bigcup^{mult} m_i' \succ\!\succ_M \bigcup^{mult} m_j$.* □

Theorem A.1 *Algorithm 6.1 always terminates. If it does not fail*[52]*, $p \sqsupset 0$ holds.*

---

[51] if $m \geq 2$. The case $m = 1$ is obvious.

[52] Algorithm 6.1 will fail, if there is no solution for the set I of part 3.

Proof A.4 *The proof of the termination is obvious. The correctness of the algorithm is guaranteed since the inequality* $x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \geq \mu^{\sum(k_j - i_j)} x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ *if* $(\forall j)\, k_j \geq i_j$ *is valid and as the decision procedure for linear inequalities provides a correct solution.* □

Lemma A.4 *Let* $\alpha, \beta, x_l > 0$*. Then,*

$$\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} + \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} \geq 2\sqrt{\alpha\beta} \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$$
$$\text{if } (\forall l \in [1,n])\, k_l = \tfrac{i_l + j_l}{2}$$

Proof A.5 $\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} + \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} \geq 2\sqrt{\alpha\beta} \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$

$\iff$

$\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} + \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} \geq 2\sqrt{\alpha\beta} \cdot x_1^{\frac{i_1+j_1}{2}} \cdot \ldots \cdot x_n^{\frac{i_n+j_n}{2}}$

since $(\forall l \in [1,n])\, k_l = \frac{i_l+j_l}{2}$

$\iff$

$\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} + \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} \geq 2 \cdot (\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n})^{\frac{1}{2}} (\beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n})^{\frac{1}{2}}$

Let $a_1 = \alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$, $a_2 = \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n}$*. Then, we have to show:*

$a_1 + a_2 \geq 2 \cdot a_1^{\frac{1}{2}} a_2^{\frac{1}{2}}$

$\iff$

$\frac{1}{2}(a_1 + a_2) \geq (a_1 a_2)^{\frac{1}{2}}$

This relation is valid since it is a special case ($r = 2$*) of the arithmetic-mean-geometric-mean inequality for* r *non-negative numbers* $a_1, \ldots, a_r$

$$\frac{1}{r} \cdot \sum_{i=1}^{r} a_i \geq (\prod_{i=1}^{r} a_i)^{\frac{1}{r}}$$

which is proved, for example, in [HLP52]. □

Lemma A.5 *Let* $n \geq m \geq 1$*. Then, for all* $x_i > 0$*:*

$$\sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \geq m \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$$

if $(\forall l \in [1,n])\, k_l = \frac{1}{m} \cdot \sum_{j=1}^{m} i_{l_j}$

Proof A.6 *The proof is based on the arithmetic-mean-geometric- nean inequality:*
For any non-negative numbers $a_1, \ldots, a_r$,

$$\frac{1}{r} \cdot \sum_{i=1}^{r} a_i \geq (\prod_{i=1}^{r} a_i)^{\frac{1}{r}} \quad \text{holds.} \qquad (*)$$

Let $n \geq m$.

$\leadsto \sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \geq m \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$

$\iff$

$\sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} + \sum_{j=m+1}^{n} x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \geq m \cdot x_n^{k_1} \cdot \ldots \cdot x_n^{k_n} + \sum_{j=m+1}^{n} x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$

$\iff$

$$\sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} + \sum_{j=m+1}^{n} x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \geq n \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \qquad (**)$$

Let $(\forall j \in [m+1,n])\,(\forall l \in [1,n])\ i_{l_j} = k_l$.

$\leadsto$ (**)

$\Longleftrightarrow$

$$\sum_{j=1}^{n} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \geq n \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \quad (***)$$

Let $(\forall l \in [1,n])\ a_l = x_1^{i_{1_l}} \cdot \ldots \cdot x_n^{i_{n_l}}$.

$\leadsto$ (***)

$\Longleftrightarrow$

$$\sum_{j=1}^{n} a_j \geq n \cdot x_1^{\frac{1}{n}\sum_{j=1}^{n} i_{1_j}} \cdot x_2^{\frac{1}{n}\sum_{j=1}^{n} i_{2_j}} \cdot \ldots \cdot x_n^{\frac{1}{n}\sum_{j=1}^{n} i_{n_j}}$$

since $(\forall l \in [1,n])\ k_l = \frac{1}{n}\sum_{j=1}^{n} i_{l_j}$ *(because* $m \cdot k_l = \sum_{j=1}^{m} i_{l_j}$

implies $n \cdot k_l = \sum_{j=1}^{n} i_{l_j}$, *since* $(\forall l \in [m+1,n])\,(\forall l \in [1,n])\ i_{l_j} = k_l)$

$\Longleftrightarrow$

$$\sum_{j=1}^{n} a_j \geq n \cdot \left(\prod_{j=1}^{n} a_j\right)^{\frac{1}{n}}$$

since $(\forall l \in [1,n])\ a_l = x_1^{i_{1_l}} \cdot \ldots \cdot x_n^{i_{n_l}}$

This inequality is identical to (*) which is proved to be valid. □

B An Example

This section deals with the proof of the positiveness of the polynomial $p_f = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$ given in examples 3.1, 3.2 and 6.1. We apply the Simplex method to the result generated in example 6.1. The following transformations will be carried out:

Input for algorithm 8.1:

$$\left(\begin{array}{lcl}
\gamma_{21_{20}} + \gamma_{21_{11}} + \gamma_{21_{10}} + \gamma_{21_{01}} + u_1 & = & 3 \\
\gamma_{12_{02}} + \gamma_{12_{11}} + \gamma_{12_{10}} + \gamma_{12_{01}} + u_2 & = & 6 \\
3\gamma_{21_{20}} - u_3 & = & 2 \\
\gamma_{12_{02}} - u_4 & = & 2 \\
\gamma_{21_{11}} + \gamma_{12_{11}} - u_5 & = & 4 \\
\gamma_{21_{10}} + \gamma_{12_{10}} - u_6 & = & 1 \\
\gamma_{21_{01}} + \gamma_{12_{01}} - u_7 & = & 1 \\
u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 & > & 0
\end{array}\right)$$

Introducing slack variables (without x_0):

$\gamma_{i_1 \ldots i_n k_1 \ldots k_n}$								u_i							x_1	b_i
1	1	1	1					1								3
				1	1	1	1		1							6
3										-1						2
				1							-1					2
	1				1							-1				4
		1				1							-1			1
			1				1							-1		1
								1	1	1	1		1	1	1	1[53]

Canonical transformation:

$\gamma_{i_1 \ldots i_n k_1 \ldots k_n}$								u_i							x_1	x_{n+m+i}								b_i
1	1	1	1					1								1								3
				1	1	1	1		1								1							6
3										-1								1						2
				1							-1								1					2
	1				1							-1								1				4
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
-4	-2	-2	-2	-2	-2	-2	-2	-2	-2						-1									-20

Pivoting:

$p = 1$, $k = 3$:

$\gamma_{i_1 \ldots i_n k_1 \ldots k_n}$								u_i							x_1	x_{n+m+i}								b_i
	1	1	1					1		$\frac{1}{3}$						1		$-\frac{1}{3}$						$\frac{7}{3}$
				1	1	1	1		1								1							6
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
	1				1							-1								1				4
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
	-2	-2	-2	-2	-2	-2	-2	-2	-2	$-\frac{4}{3}$					-1			$\frac{4}{3}$						$-\frac{52}{3}$

$p = 2$, $k = 1$:

[53]In this example we have chosen 1 instead of *lpf*.

$\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}$								u_i							x_1	x_{n+m+i}								b_i
	1	1	1					1		$\frac{1}{3}$						1		$-\frac{1}{3}$						$\frac{7}{3}$
				1	1	1	1		1								1							6
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
		-1	-1		1			-1		$-\frac{1}{3}$		-1				-1		1		1				$\frac{5}{3}$
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
				-2	-2	-2	-2		-2	$-\frac{2}{3}$					-1	2		$\frac{2}{3}$						$-\frac{38}{3}$

$p = 5,\ k = 4$:

$\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}$								u_i							x_1	x_{n+m+i}								b_i
	1	1	1					1		$\frac{1}{3}$						1		$-\frac{1}{3}$						$\frac{7}{3}$
					1	1	1		1		1						1		-1					4
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
		-1	-1		1			-1		$-\frac{1}{3}$		-1				-1		$\frac{1}{3}$		1				$\frac{5}{3}$
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
					-2	-2	-2		-2	$-\frac{2}{3}$	-2				-1	2		$\frac{2}{3}$	2					$-\frac{26}{3}$

$p = 6,\ k = 5$:

$\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}$								u_i							x_1	x_{n+m+i}								b_i
	1	1	1					1		$\frac{1}{3}$						1		$-\frac{1}{3}$						$\frac{7}{3}$
		1	1			1	1	1	1	$\frac{1}{3}$	1	1				1	1	$-\frac{1}{3}$	-1	-1				$\frac{7}{3}$
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
		-1	-1		1			-1		$-\frac{1}{3}$		-1				-1		$\frac{1}{3}$		1				$\frac{5}{3}$
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
		-2	-2			-2	-2	-2	-2	$-\frac{4}{3}$	-2	-2			-1			$\frac{4}{3}$	2	2				$-\frac{16}{3}$

$p = 3,\ k = 6$:

$\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}$								u_i							x_1	x_{n+m+i}								b_i
	1		1			-1		1		$\frac{1}{3}$			1			1		$-\frac{1}{3}$			-1			$\frac{4}{3}$
			1				1	1	1	$\frac{1}{3}$	1	1	1			1	1	$-\frac{1}{3}$	-1	-1	-1			$\frac{4}{3}$
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
			-1		1	1		-1		$-\frac{1}{3}$		-1	-1			-1		$\frac{1}{3}$		1	1			$\frac{8}{3}$
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
			-2				-2	-2	-2	$-\frac{4}{3}$	-2	-2	-2		-1			$\frac{4}{3}$	2	2	2			$-\frac{10}{3}$

$p = 4,\ k = 7$:

$\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}$								u_i							x_1	x_{n+m+i}								b_i
	1					-1	-1	1		$\frac{1}{3}$			1	1		1		$-\frac{1}{3}$			-1	-1		$\frac{4}{3}$
								1	1	$\frac{1}{3}$	1	1	1	1		1	1	$-\frac{1}{3}$	-1	-1	-1	-1		$\frac{4}{3}$
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
					1	1	1	-1		$-\frac{1}{3}$		-1	-1	-1		-1		$\frac{1}{3}$		1	1	1		$\frac{11}{3}$
		1				1							-1								1			1
			1				1							-1								1		1
								1	1	1	1	1	1	1	1								1	1
								-2	-2	$-\frac{4}{3}$	-2	-2	-2	-2	-1			$\frac{4}{3}$	2	2	2	2		$-\frac{4}{3}$

$p = 9,\ k = 2$:

$\gamma_{i_1\ldots i_n k_1\ldots k_n}$								u_i							x_1	x_{n+m+i}								b_i
	1					-1	-1		-1		-1	-1					-1		1	1				0
								1	1	$\frac{1}{3}$	1	1	1	1		1	1	$-\frac{1}{3}$	-1	-1	-1	-1		$\frac{1}{3}$
1										$-\frac{1}{3}$								$\frac{1}{3}$						$\frac{2}{3}$
				1							-1								1					2
					1	1	1	-1		$-\frac{1}{3}$		-1	-1	-1				$\frac{1}{3}$		1	1	1		$\frac{11}{3}$
		1				1							-1								1			1
			1				1							-1								1		1
										$\frac{2}{3}$					1	-1	-1	$\frac{1}{3}$	1	1	1	1	1	$\frac{2}{3}$
										$-\frac{2}{3}$					-1	2	2	$\frac{4}{3}$						$-\frac{2}{3}$

$p = 11$, $k = 8$:

$\gamma_{i_1\ldots i_n k_1\ldots k_n}$								u_i							x_1	x_{n+m+i}								b_i
	1					-1	-1		-1		-1	-1					-1		1	1				0
								1	1		1	1	1	1	$-\frac{1}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	$-\frac{1}{2}$	$-\frac{3}{2}$	$-\frac{3}{2}$	$-\frac{3}{2}$	$-\frac{3}{2}$	$-\frac{1}{2}$	0
1															$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
				1							-1								1					2
					1	1	1	-1				-1	-1	-1	$\frac{1}{2}$	$-\frac{3}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{1}{2}$	4
		1				1							-1								1			1
			1				1							-1								1		1
										1					$\frac{3}{2}$	$-\frac{3}{2}$	$-\frac{3}{2}$	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	$\frac{3}{2}$	1
																1	1	1	1	1	1	1	1	0

A feasible solution is of the form $(1,0,1,1,2,4,0,0,0,0,1,0,0,0,0)$. This vector leads to the following values:
$\gamma_{21_{20}} = 1$, $\gamma_{21_{11}} = 0$, $\gamma_{21_{10}} = 1$, $\gamma_{21_{01}} = 1$, $\gamma_{12_{02}} = 2$, $\gamma_{12_{11}} = 4$, $\gamma_{12_{10}} = 0$, $\gamma_{12_{01}} = 0$, $u_3 = 1$ and $(\forall i \in \{1,2,4,5,6,7\})\, u_i = 0$.