Termination Proofs of Rewriting Systems
–
Heuristics for Generating Polynomial Orderings[1)]

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

December 20, 1991

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

Abstract

Orderings on polynomial interpretations of operators represent a powerful technique for proving the termination of rewriting systems. One of the main problems of polynomial orderings concerns the choice of the right interpretation for a given rewriting system. It is very difficult to develop techniques for solving this problem. Here, we present three new heuristic approaches: (i) guidelines for dealing with special classes of rewriting systems, (ii) an algorithm for choosing appropriate special polynomials as well as (iii) an extension of the original polynomial ordering which supports the generation of suitable interpretations. All these heuristics will be applied to examples in order to illustrate their practical relevance.

Contents

List of Figures

Chapter 1

Introduction and Summary

Term rewriting systems (TRS, for short) provide a powerful tool for expressing non-deterministic computations and as a result they have been widely used in formula-manipulation and theorem-proving systems. Moreover, there exists a potential for their application in many other areas of computer science and mathematics such as abstract data type specifications and program verification. 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 of proving (finite) 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 rewriting systems: *A TRS $\mathcal{R}$ terminates if, and only if, there exists a reduction ordering $\succ$ such that $l \succ r$ for each rule $l \rightarrow r$ 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]). For example, the well-known Knuth-Bendix orderings ([KB70]) are defined using $\mathcal{W} := \mathbb{N}$, $\succ := >$[1] and φ as the 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 $>$). See [Lan75a], [Lan75b], [Lan76], [Sti76], [Lan79], [Der79], [HO80], [Der83], [Ben86], [Lan86], [Les86], [BL87], [Der87], [Rou88], [HL89], [SZ90], [Rou91] and [Ste91] for details about polynomial orderings.

One of the main problems concerning polynomial orderings is the choice of the right interpretation for a given TRS. The object of this paper is to present new insights into the (semi-) automatic generation of termination proofs using polynomial orderings. In the following chapter we briefly recapitulate the most essential notions used in connection with TRS and termination[2]. Chapter 3 contains the definition of polynomial orderings whereas chapter 4 presents some statistics on the orientation of TRS conducted with polynomial orderings. Chapter 5 deals with new guidelines for generating interpretations which can guarantee the termination of special classes of TRS. In chapter 6 we describe an algorithm based on the Simplex method for finding polynomial interpretations for a given rule system provided that this system can be oriented using special polynomials. Moreover, it is possible to extend polynomial orderings similar to the Knuth-Bendix orderings with status (see [Ste89]) by using an additional ordering and a status function on operators. This improvement supports the generation of suitable interpretations and is introduced in chapter 7. Finally, chapter 8 contains some detailed examples, illustrating the features presented. On the margin of the text we refer to related examples given in chapter 8.

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

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

Chapter 2

Notations

We assume familiarity with the standard definitions of the set of *function symbols* (or operators) $\mathcal{F}$ and their *arities*[1] $\mathcal{A}r$, the set of *variables*[2] $\mathcal{X}$, the set of *terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$, the set of *ground terms* $\mathcal{G}(\mathcal{F})$, the set of *occurrences* $\mathcal{P}os(t)$ as well as with the definition of a *substitution* $t\sigma$ of a term t and *rewriting systems* $\mathcal{R} = \{l_i \rightarrow r_i \mid i \in [1,n]\}$.

$t|_u$ stands for the *subterm* of t at position $u \in \mathcal{P}os(t)$. The *replacement* of the term $t|_u$ by s is denoted by $t[s]_u$. $\mathcal{R}$ defines the *rewriting relation* $\Rightarrow_{\mathcal{R}}$ over $\mathcal{T}(\mathcal{F},\mathcal{X})$ such that $s \Rightarrow_{\mathcal{R}} t$ if, and only if, $(\exists l \rightarrow r \in \mathcal{R})(\exists u \in \mathcal{P}os(s))(\exists\sigma)\ s|_u = l\sigma \wedge t = s[r\sigma]_u$. The *transitive reflexive closure* of $\Rightarrow_{\mathcal{R}}$ is referred to by $\Rightarrow^*_{\mathcal{R}}$. $\Rightarrow_{\mathcal{R}}$ is *confluent* if, and only if, ${}^*_{\mathcal{R}}\!\Leftarrow \cdot \Rightarrow^*_{\mathcal{R}} \subseteq \Rightarrow^*_{\mathcal{R}} \cdot {}^*_{\mathcal{R}}\!\Leftarrow$. $\mathcal{R}$ is *canonical* if, and only if, $\Rightarrow_{\mathcal{R}}$ is confluent and $\mathcal{R}$ is terminating. The *Knuth-Bendix completion* procedure ([KB70]) sometimes enables the generation of a canonical rewriting system equivalent to a given one. The Knuth-Bendix algorithm *diverges* if it does not terminate, that means it creates infinitely many rules.

The *leading function symbol* and the tuple of the (direct) *arguments* of a term t are referred to by $\mathcal{H}ead(t)$ and $\mathcal{A}rgs(t)$, respectively. We write '$\Delta \in t$' for depicting the occurrence of the symbol Δ in the term t. The *empty term* is denoted by λ. The *set of variables* of a term t is denoted by $\mathcal{V}ar(t)$. A term is *linear* if no variable occurs more than once. A rule $l \rightarrow r$ is said to be *complicated* if, and only if, both l and r are not linear. A rule $l \rightarrow r$ is *variable-reducing* if, and only if, $\mathcal{V}ar(r) \subset \mathcal{V}ar(l)$. The *size* of a term t reflects the number of symbols occurring in t and the *depth* of t provides the size of the longest path of t, i.e. $depth(\Delta) = 1$ where Δ is a constant or a variable and $depth(f(t_1,\ldots,t_n)) = 1 + max\{depth(t_i)\}$, otherwise.

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 $\mathcal{F}$ is called a *precedence* and 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* (or compatible) w.r.t. the structure of terms ($s \succ t \leadsto f(\ldots,s,\ldots) \succ f(\ldots,t,\ldots)$). With *simplification orderings* we refer to a special class of reduction orderings that require the so-called *subterm property* ($f(\ldots,t,\ldots) \succ t$) which ensures that the so-called *homeomorphic embedding* relation $\succ_{emb}$ is contained in $\succ$ (see [Der79]). The ordering $\succ_{emb}$ is the irreflexive part of $\succeq_{emb}$ which is defined[3] by $s \equiv f(s_1,\ldots,s_m) \succeq_{emb} g(t_1,\ldots,t_n) \equiv t$ iff (1) $f = g \wedge (\forall i \in [1,n])\ s_i \succeq_{emb} t_i$ or (2) $(\exists i \in [1,n])\ s_i \succeq_{emb} t$. A rule $l \rightarrow r$ is said to be *trivial* if, and only if, $l \succ_{emb} r$.

Note that a term ordering $\succ$ is used to compare terms. Since operators have terms as arguments, we define an extension of $\succ$, called *lexicographically greater* ($\succ^{lex}$), on tuples of terms as follows: $(s_1,s_2,\ldots,s_m) \succ^{lex} (t_1,t_2,\ldots,t_n)$ if either $m > 0 \wedge n = 0$ or $s_1 \succ t_1$ or $s_1 = t_1 \wedge (s_2,\ldots,s_m) \succ^{lex} (t_2,\ldots,t_n)$. The extension of $\succ$ to *multisets* of terms is defined as follows: A multiset S is greater than a

[1] An operator with no arguments (i.e. whose arity is zero) is called a *constant* (symbol). A *unary* (*binary*) operator has one (two) arguments.

[2] Note that $\mathcal{A}r(x) = 0$ if $x \in \mathcal{X}$.

[3] We will use this simplified version since we only deal with operators having fixed arity.

multiset T, denoted by $S \succ^{mul} T$ iff $S \neq T \wedge (\forall t \in T \backslash S)(\exists s \in S \backslash T)\, s \succ t$. To combine these two concepts of tuples and multisets, we assign a *status* $\tau(f)$ to each operator $f \in \mathcal{F}$ that determines the order according to which the subterms of f are compared (see [KL80]). Formally, a status is a function which maps the set of operators into the set {mul, left, right}. Thus, a function symbol can have one of the following three types of status: *mul* (the arguments will be compared as multisets), *left* (lexicographical comparison from left to right) and *right* (the arguments will lexicographically be compared from right to left). The result of an application of the function $\mathcal{A}rgs$ to a term $t = f(t_1, \ldots, t_n)$ depends on the status of f: If $\tau(f) = mul$, then $\mathcal{A}rgs(t)$ is the multiset $\{t_1, \ldots, t_n\}$. Otherwise, $\mathcal{A}rgs(t)$ describes the tuple $(t_1, \ldots, t_n)$.

Chapter 3

Polynomial Orderings

Polynomial orderings have been studied by Manna & Ness ([MN70]), Lankford ([Lan75a], [Lan75b], [Lan76], [Lan79]), Dershowitz ([Der79], [Der83], [Der87]), Huet & Oppen ([HO80]) and BenCherifa & Lescanne ([Ben86], [BL87]). 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[1)]. 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*[2)] $\alpha_{r_1 \ldots r_n} \cdot x_1^{r_1} \cdot \ldots \cdot x_n^{r_n}$. A polynomial $\sum_{r_i \in \mathbb{N}} \alpha_{r_1 \ldots r_n} \cdot x_1^{r_1} \cdot \ldots \cdot x_n^{r_n}$ based on n distinct variables is represented by $p(x_1, \ldots, x_n)$. 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 there occur n variables in p that differ by pairs.

Definition 3.1 *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 3.2 ([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*

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

$$\textit{with}^{3)} \quad p \sqsupset q \iff (\forall X_1, \ldots, X_n \in \mathcal{M})\, p(X_1, \ldots, X_n) > q(X_1, \ldots, X_n)$$

Example 3.1 *We prove the termination of* 8.1

8.2

8.5

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(x,y) & \rightarrow & g(x,y) \\ g(h(x),y) & \rightarrow & h(f(x,y)) \end{array} \right.$$

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

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

1) Dershowitz uses an arbitrary set by requiring the functions to possess the subterm property.

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

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

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 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 3.2 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 3.1 *Note that* $\mu = min\{[c]() \mid c \text{ is a constant symbol of } \mathcal{F}\}$ *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,*

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

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.

Chapter 4

Statistics On Some Examples

We have integrated the polynomial orderings in our completion environment *COMTES* ([AMS89]). One of the main problems concerning polynomial orderings lies in the choice of the right interpretations for a given TRS (see chapter 8). A series of 320 experiments considering more than 1700 rules[1)] occurring in the literature has been conducted (see also [SK90]) to gain more insight into the choice of interpretations. In order to infer heuristics for generating adequate interpretations we shall split the examples into various domains and provide some conditions for the interpretations we will need to use. The next definition represents a possible classification of interpretations. The given classes of polynomials are constructed w.r.t. the coefficients as well as the exponents of a polynomial.

Definition 4.1 *Let $p(x_1, \ldots, x_n)$ be any polynomial of $\mathcal{P}ol(\mathbb{N})$ and $\alpha_i, \beta \in \mathbb{N}$.*

- $p(x_1, \ldots, x_n)$ *is* **strongly linear** *iff*[2)] $p(x_1, \ldots, x_n) = \sum x_i + \beta$.
- $p(x_1, \ldots, x_n)$ *is* **linear** *iff* $p(x_1, \ldots, x_n) = \sum \alpha_i x_i + \beta$.
- $p(x_1, \ldots, x_n)$ *is* **separate** *iff each monomial contains at most one variable*[3)]*:* $\sum\limits_i(\sum\limits_j \alpha_{ij} x_i^j)$.
- $p(x_1, \ldots, x_n)$ *is* **mixed** *iff it is a general polynomial.*
- $p(x_1, \ldots, x_n)$ *is* **simple-mixed** *iff* $p(x_1, \ldots, x_n)$ *is mixed and all exponents are not greater than 1 or* $n = 1$ *and* $p(x_1) = \alpha_2 x_1^2 + \alpha_1 x_1 + \beta$ *such that* $\alpha_1 \cdot \alpha_2 = 0$.

For example, $x + y + 5$ is strongly linear, $2x + y + 1$ is linear, $x^2 + 2y$ is separate and the polynomials $2xy + 3y$, $5x^2 + 1$ are of simple-mixed form. Figure 4.1 illustrates the set-theoretic interdependencies between the different classes of polynomials.

Definition 4.2 *Let $p(x_1, \ldots, x_n)$ be any polynomial of $\mathcal{P}ol(\mathbb{N})$. Furthermore, let $\mathcal{C}$ be any class of definition 4.1.*

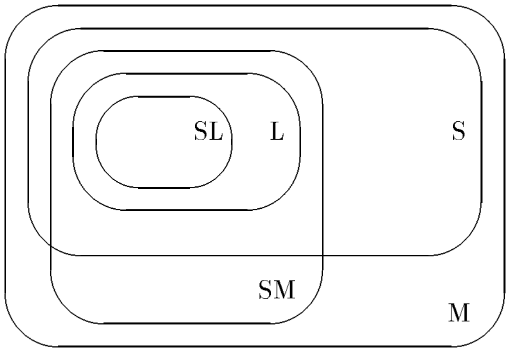
- $p(x_1, \ldots, x_n)$ *of a class $\mathcal{C}$ is called* **strict** *iff it is not contained in any subclass*[4)] *of $\mathcal{C}$.*
- $p(x_1, \ldots, x_n)$ *is called* **complete** *iff* $p(x_1, \ldots, x_n) = \sum \alpha_{r_1 \ldots r_n} \cdot x_1^{r_1} \cdot \ldots \cdot x_n^{r_n}$ *and the following condition holds:*

[1)]The number of rules in an experiment either refers to the canonical or to the initial (if no canonical could be generated) rule system. Note also that we do not count the rules which have been removed by other rules during the normalization process.

[2)]if, and only if,

[3)]with arbitrary exponent

[4)]see figure 4.1



The following abbreviations are used: *SL* (strongly linear), *L* (linear), *S* (separate), *M* (mixed), *SM* (simple-mixed).

Figure 4.1: Classes of polynomials

$$\textit{If} \quad \alpha_{k_1 \ldots k_n} > 0 \quad \textit{then} \quad (\forall i \in [1,n])\, k_i \geq 1 \leadsto \alpha_{k_1 \ldots k_{i-1}(k_i-1)k_{i+1} \ldots k_n} > 0.$$

A polynomial p is complete if, and only if, for each monomial $\alpha_{k_1 \ldots k_n} \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$ (with $\alpha_{k_1 \ldots k_n} > 0$) *every* 'smaller' monomial is also part of p. A monomial $\alpha_{i_1 \ldots i_n} \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ is *smaller* than a monomial $\alpha_{j_1 \ldots j_n} \cdot x_1^{j_1} \cdot \ldots \cdot x_n^{j_n}$ if $(\forall k \in [1,n])\, j_k \geq i_k$.

Example 4.1 *(i) The polynomial $2x^2 + y$ is strictly separate because it is separate but not linear. Obviously, it is also mixed but not strictly mixed.*

(ii) The polynomial $2x^2y + 3xy + x^2 + 5x + 5y + 1$ is complete whereas the polynomial xy is not complete.

In addition to the classification of interpretations, we divided the tested examples into domains. This way, we are able to make more subtle statements about the interpretations needed. The following classification for rule systems seems appropriate:

- *Algebraic Structures* (AS, for short): Groups, rings, etc. 8.4
- *Boolean Theories* (BT, for short): This domain is closely connected with that of algebraic structures. 8.5
- *Arithmetic Theories* (AT, for short): Addition, multiplication, etc. 8.3, 8.8
- *Lists* (L, for short): Append, reverse, flatten, etc. 8.6
- *String Rewriting Systems* (SRS, for short): The structure of strings is easier than that of lists since they use unary operators, only. 8.2
- *Other Systems* (OS, for short): Systems that do not fall into one of the five former domains. 8.1

Figure 4.2 and figure 4.3 present detailed statistics on the results of the experiments based on the given domains and the used interpretations. We made a considerable effort to determine 'minimal' interpretations, for example, if a system can either be oriented with the help of a strictly separate or a linear interpretation we preferred the linear one.

The most important results stated in figure 4.2 and figure 4.3 can be summarized as follows:

- First of all, there are some interesting observations concerning the structure of the tested TRS: Each system averagely contains less than five different operators (more unary than binary operators, nearly no function symbols with an arity more than 2). The average number of different variables per rule is less than 2. Surprisingly, about 45% of all rules are trivial ones (whose termination proof is obvious when a simplification ordering is used). Only every 10th rule system contains a complicated rule.

- Canonical systems (including confluent initial systems) generated: L-81%, OS-79%, AT-71%, SRS-71%, AS-68%, BT-64%, Total-73%. Linear (including strongly linear) interpretations used: SRS-97%, OS-82%, AT-72%, BT-68%, L-68%, AS-48%, Total-71%. Strictly mixed interpretations needed: AS-50%, BT-32%, L-32%, AT-15%, OS-14%, SRS-0%, Total-23%. Finally, **96%** of the **orientable** (w.r.t. $\succ_{POL}$) systems can also be oriented using simple-mixed interpretations. Figure 4.4 gives a graphical illustration of these numbers.

- Every 13th rule has a non-linear right-hand side. 35% of these rules cannot be oriented w.r.t. $\succ_{POL}$. 56% of the orientable non right-linear rules require strictly mixed interpretations.

- The number of strictly mixed interpretations needed in the class of algebraic structures is relatively big (50%). In contrast to the other classes, only a small quantity (23%) of initial systems is directly confluent.

- It is significant that 90% of the general systems ('other systems') can be oriented. Furthermore, these orientations were mainly carried out by using linear interpretations, only.

- The proofs of the termination of the examples on lists and strings can often be successfully conducted with polynomial orderings. However, while lists require more than 30% of strictly mixed interpretations, SRS can very often be oriented with the help of linear interpretations. Note that the SRS $\mathcal{R} = \{ab \rightarrow bba\,,\, ca \rightarrow aac\}$ can be oriented[5)] but not with linear interpretations. Linear polynomials even permit SRS to produce derivations of exponential length[6)]. Furthermore, the example $cab \rightarrow acb^2a$ requires exponential functions[7)].

---

5)using $[a](x) = 3x$, $[b](x) = x + 1$ and $[c](x) = x^3$

6)For example, $ab \rightarrow bba$ with $[a](x) = 3x$, $[b](x) = x + 1$ produces $a^n b \xrightarrow{+} b^{2^n} a^n$ using $2^n - 1$ steps.

7)$[a](x) = 3x$, $[b](x) = x + 1$ and $[c](x) = 4^x$

	AS	BT	AT	L	SRS	OS
Number of rule systems tested	57	28	72	36	41	86
Average number of rules per system	6.7	5.5	5.1	6.1	3.5	5.5
Average size of left-hand sides	4.2	3.8	3.7	4.0	3.5	4.0
Average size of right-hand sides	2.8	2.9	2.8	2.9	3.5	3.2
Average depth of left-hand sides	2.8	2.5	2.8	2.7	3.5	2.8
Average depth of right-hand sides	1.9	1.9	2.1	2.0	3.5	2.2
Average number of constant symbols per system	1.0	1.2	1.3	2.3	0	1.9
Average number of unary operators per system	0.8	0.7	1.8	1.4	3.6	2.1
Average number of binary operators per system	1.6	2.0	1.4	2.6	0	1.1
Average number of remaining operators per system	0.0	0.3	0.0	0.2	0	0.1
Average number of variables per rule	3.2	3.3	2.4	3.1	2	2.2
Average number of different variables per rule	1.6	1.3	1.2	1.7	1	1.1
Average number of variable-reducing rules per system	2.1	1.4	0.7	1.6	0	0.7
Average number of trivial rules per system	3.7	2.6	2.4	2.4	0.8	2.2
Average number of non-linear left-hand sides per system	2.0	1.3	0.2	0.3	0	0.7
Average number of non-linear right-hand sides per system	0.5	0.9	0.8	0.2	0	0.3
Average number of complicated rules per system	0.1	0.1	0	0	0	0.1
Confluent initial rule systems	13	15	46	23	20	50
Canonical rule systems generated	26	3	5	6	9	18
Diverging completion processes	7	5	3	2	3	6
Completion processes that stopped with failure	8	2	0	3	1	4
Systems which could not usefully oriented w.r.t. $\succ_{POL}$	3	3	18	2	8	8
Systems oriented using strongly linear polynomials	7	11	13	7	15	40
Systems oriented using strictly linear polynomials	19	6	26	16	17	24
Systems oriented using strictly separate polynomials	1	0	7	0	1	3
Systems oriented using strictly mixed polynomials	27	8	8	11	0	11
Systems oriented using simple-mixed polynomials	48	25	53	30	33	77

Figure 4.2: Statistics on examples w.r.t. special domains

	Total
Number of rule systems tested	320
Average number of rules per system	5.4
Average size of left-hand sides	3.9
Average size of right-hand sides	3.0
Average depth of left-hand sides	2.9
Average depth of right-hand sides	2.3
Average number of constant symbols per system	1.3
Average number of unary operators per system	1.7
Average number of binary operators per system	1.5
Average number of remaining operators per system	0.1
Average number of variables per rule	2.7
Average number of different variables per rule	1.3
Average number of variable-reducing rules per system	1.1
Average number of trivial rules per system	2.4
Average number of non-linear left-hand sides per system	0.8
Average number of non-linear right-hand sides per system	0.5
Average number of complicated rules per system	0.1
Confluent initial rule systems	167
Canonical rule systems generated	67
Diverging completion processes	26
Completion processes that stopped with failure	18
Systems which could not usefully oriented w.r.t. $\succ_{POL}$	42
Systems oriented using strongly linear polynomials	93
Systems oriented using strictly linear polynomials	108
Systems oriented using strictly separate polynomials	12
Systems oriented using strictly mixed polynomials	65
Systems oriented using simple-mixed polynomials	266

Figure 4.3: Statistics on examples without considering domains

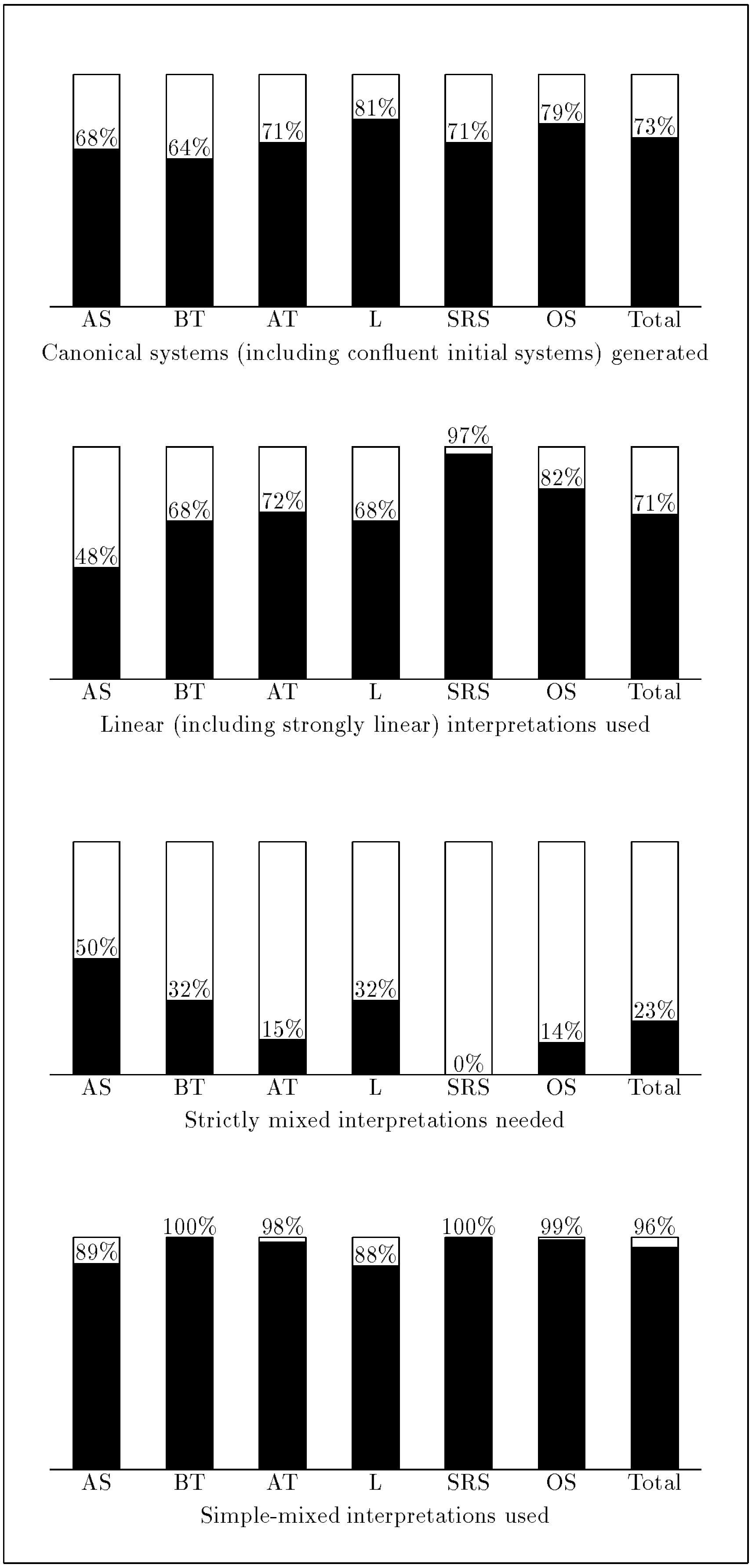


Figure 4.4: Graphical illustration of the most important results of the statistics

Chapter 5

Heuristics for Choosing Interpretations

Based on the experience with examples (see chapter 4), several new heuristics have been developed for the choice of useful (i.e. successful) interpretations. The following two heuristics provide suggestions for proving the termination of arithmetic specifications and algebraic structures.

Heuristic 5.1 *In the case of arithmetic specifications it has proved helpful to choose the complexity hierarchy of the interpretations such that it reflects the complexity hierarchy of the corresponding function symbols.*

For example, $[+](x,y) = xy + x$, $[*](x,y) = x^2y + xy^2$ and $[exp](x,y) = (x+y)^3$. 8.7

Heuristic 5.2 *Let* $*, i, e \in \mathcal{F}$ *and let* $\mathcal{E}$ *be a set of group axioms*[1)]. *Appropriate and relatively simple interpretations for orienting (and completing)* $\mathcal{E}$ *are the following ones:* $[*](x,y) = 2xy + x$ *(or* $2xy + y$*)*, $[i](x) = x^2$ *and* $[e]() = 2$.

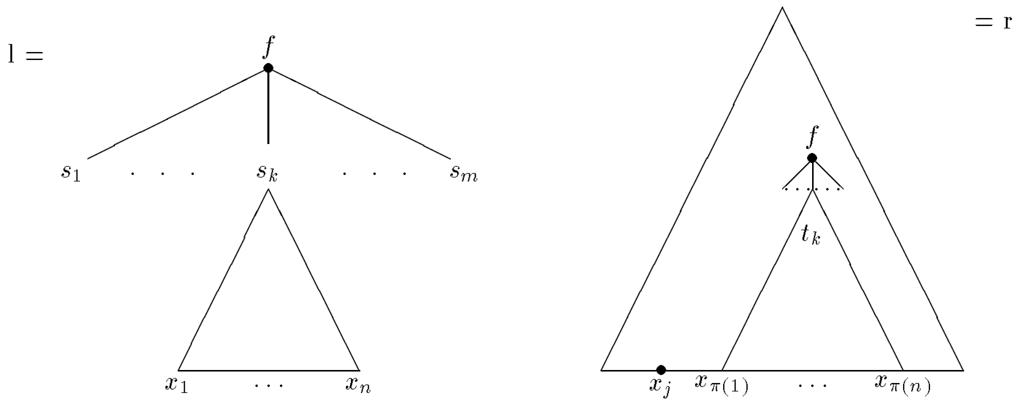
It is obvious that heuristic 5.2 can also be applied to systems representing an *extension* of the set 8.4
$\mathcal{E}$ of group axioms. The following lemmas provide tests for deciding whether a strictly separate interpretation for unary operators as well as a strictly mixed interpretation are necessary for the orientation of a given rule.

Lemma 5.1 *Let* $l \to r$ *be a rule. No linear interpretation for the unary operator* i *exists if one of the following conditions holds:*

- $$\begin{array}{ll} (\exists u \in \mathcal{P}os(l)) & l|_u = i(s) \\ & \wedge\ (\forall h \in \mathcal{F})\, i \neq h \in l \leadsto [h](x_1, \ldots, x_{\mathcal{A}r(h)}) \textit{ is linear} \\ \textit{and} & \\ (\exists v \in \mathcal{P}os(r)) & r|_v = g(t_1, \ldots, t_{\mathcal{A}r(g)}) \\ & \wedge\ [g](x_1, \ldots, x_{\mathcal{A}r(g)}) \textit{ is not linear } \wedge\ \mathcal{V}ar(r|_v) \subseteq \mathcal{V}ar(l|_u) \end{array}$$
- $l = i(f(x,y))$ *and* $r = f(i(y), i(x))$

The main part of the first condition of lemma 5.1 requires that the right-hand side r contains a strictly separate or strictly mixed operator g while only linear operators (except i) appear in l.

[1)] $\mathcal{E}$ includes $(x * y) * z = x * (y * z)$ and $i(x * y) = i(y) * i(x)$.



$$\{x_1,\ldots,x_n\} \not\subseteq \mathcal{V}ar(s_i), i \neq k \quad \textit{and} \quad t_k \succeq_{emb} s_k$$

Figure 5.1: Graphical representation of the first part of lemma 5.2

Lemma 5.2 *Let $l \to r$ be a rule. No separate interpretation for the operator f with $\mathcal{A}r(f) \geq 2$ exists if one of the following conditions holds:*

- $l = f(s_1,\ldots,s_m)$ *and*
$$\begin{array}{ll} (\exists k) & \mathcal{V}ar(s_k) \cap \mathcal{V}ar(l[\lambda]_k) = \emptyset \\ & \wedge\ (\exists v \in \mathcal{P}os(r))\ r|_v = f(t_1,\ldots,t_m)\ \wedge\ t_k \succeq_{emb} s_k \\ & \wedge\ \mathcal{V}ar(r[\lambda]_{v.k}) \cap \mathcal{V}ar(s_k) \neq \emptyset \end{array}$$
- $$\begin{array}{ll} (\exists u \in \mathcal{P}os(l)) & l|_u = f(s_1,\ldots,s_m) \\ & \wedge\ (\forall h \in \mathcal{F})\ f \neq h \in l \leadsto [h](x_1,\ldots,x_{\mathcal{A}r(h)}) \textit{ is linear} \\ \textit{and} & \\ (\exists v \in \mathcal{P}os(r)) & r|_v = g(t_1,\ldots,t_{\mathcal{A}r(g)}) \\ & \wedge\ [g](x_1,\ldots,x_{\mathcal{A}r(g)}) \textit{ is strictly mixed } \wedge\ \mathcal{V}ar(r|_v) \subseteq \mathcal{V}ar(l|_u) \end{array}$$
- $l = i(f(x,y)), r = f(i(y),i(x))$ *and f is associative*[2)]

The first condition of lemma 5.2 requires that the variables of s_k do not occur elsewhere in l, at least one variable of $\mathcal{V}ar(s_k)$ occurs in r outside of t_k and t_k is greater than (w.r.t. $\succeq_{emb}$) s_k. A graphical illustration of the technical conditions of the first part of lemma 5.2 is given in figure 5.1. The conditions of the second part of lemma 5.2 are slightly modified versions of the first part of lemma 5.1 (applied to operators with an arity greater than 1).

Example 5.1 *(i) The orientation of the rule $f(i(x),y) \to g(x,y)$ requires a strictly separate interpretation for i if, for example, $[g](x,y) = x^2 + y$ and $[f](x,y) = x + y$, since the first condition of lemma 5.1 is valid.* 8.3

(ii) The orientation of the distributivity axiom $f(x,g(y,z)) \to g(f(x,y),f(x,z))$ requires a strictly mixed interpretation for f since the first condition of lemma 5.2 is valid: $s_k = x$, $v = 1$, $t_k = x$, $t_k \succeq_{emb} s_k$ and $r|_{21} = x$. Furthermore, according to the second condition, the interpretation of f in $h(f(x,f(y,z))) \to h(f(x+y,z))$ must be strictly mixed if $[+](x,y)$ is strictly mixed and $[h](x)$ is linear. 8.8

With the help of lemma 5.1, 91% of the 98 strictly separate interpretations, that were needed for our examples, were automatically detected. Of the 83 operators with strictly mixed interpretations, 75% were found by application of lemma 5.2.[3)]

[2)] f is associative if either $f(f(x,y),z) \to f(x,f(y,z))$ or $f(x,f(y,z)) \to f(f(x,y),z)$ is contained in the TRS.
[3)] 399 unary operators and 467 operators where the arity is greater than 1 have been studied.

During experimentation, the combination of the distributivity and the associativity turned out to be problematic. The following suggestion can sometimes establish the termination of a system containing these two axioms[4)].

Heuristic 5.3 *Let* $*, + \in \mathcal{F}$ *and let*

$$\mathcal{R}_D = \left\{ \begin{array}{lcl} x * (y + z) & \rightarrow & (x * y) + (x * z) \\ (x + y) * z & \rightarrow & (x * z) + (y * z) \end{array} \right.$$

- *The termination of* $\mathcal{R}_{\mathcal{D}}$ *can be proved with the help of* $[+](x, y) = x + y + 1, [*](x, y) = xy$ *and* $\mu \geq 2$. 8.13
- *The system* $\mathcal{R}_D \cup \{x + (y + z) \rightarrow (x + y) + z\}$ *terminates if* $[+](x, y) = x + 2y + 1, [*](x, y) = xy$ *and* $\mu \geq 2$.
- *The rule system* $\mathcal{R}_D \cup \{(x * y) * z \rightarrow x * (y * z)\}$ *can be oriented using the interpretations* $[+](x, y) = x + y + 2, [*](x, y) = xy + x$ *and* $\mu \geq 3$.
- *In order to achieve the termination of* $\mathcal{R}_D \cup \{(x + y) + z \rightarrow x + (y + z), (x * y) * z \rightarrow x * (y * z)\}$, *we use the interpretations* $[+](x, y) = 2x + y + 4, [*](x, y) = xy + x$ *and* $\mu \geq 5$.

[4)]Note that a lot of completion processes including these axioms will diverge if the associativity is not treated as an underlying theory.

Chapter 6

A Method for Simple-Mixed Polynomials

In this chapter, we develop a procedure that attempts to automatically generate interpretations of operators such that a given TRS terminates. This technique is restricted to simple-mixed polynomials because it is very difficult to compare two *general* polynomials. However, since 96% of the interpretations used for the orientation of the tested examples (see chapter 4) are simple-mixed, this is an acceptable restriction.

According to the algorithm of [Mar87][1)] for generating an appropriate weight function (i.e. *strongly linear* interpretations) for the Knuth-Bendix ordering, we transform the set of rules into a set of *linear* inequalities based on the coefficients of the interpretations w.r.t. common variables. We make use of a relatively simple algorithm, the so-called Simplex method for deciding whether a system of *linear* inequalities has a solution. Unfortunately, we do not have *linear* inequalities, initially, because there might be sums of products resulting from strictly simple-mixed interpretations (see the following example). In order to apply the Simplex method we will transform more general inequalities to linear ones by

1. approximating each side to exactly one product (see lemma 6.2) and then
2. applying a logarithmic function to the resulting products.

Because of the technical complexity of the algorithm, we illustrate the important steps by an example.

Example 6.1 *We will prove the termination of the TRS (given in [Mid88])*[2)]

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(x,y) & \rightarrow & x \\ g(a) & \rightarrow & h(a,b,a) \\ i(x) & \rightarrow & f(x,x) \\ h(x,x,y) & \rightarrow & g(x) \end{array} \right.$$

In order to find a polynomial ordering for orienting $\mathcal{R}$, we have to define variable interpretations[3)] for the operators of the used signature. Therefore, we introduce complete[4)] *simple-mixed* polynomials $[f]$ with *variable* coefficients for each $f \in \mathcal{F}$.

[1)] See also appendix of [Lan79].
[2)] Note that the second rule and the fourth rule are the problematic ones.
[3)] i.e. interpretations with *variable* coefficients
[4)] see definition 4.2

Example 6.2 (Example 6.1 continued)

$$\begin{array}{lcl}
[a]() & = & \alpha_0 \\
[b]() & = & \beta_0 \\
[f](x,y) & = & \chi_{11}xy + \chi_{10}x + \chi_{01}y + \chi_{00} \\
[g](x) & = & \delta_2 x^2 + \delta_1 x + \delta_0 \\
[h](x,y,z) & = & \epsilon_{111}xyz + \epsilon_{110}xy + \epsilon_{101}xz + \epsilon_{011}yz + \epsilon_{100}x + \epsilon_{010}y + \epsilon_{001}z + \epsilon_{000} \\
[i](x) & = & \phi_2 x^2 + \phi_1 x + \phi_0
\end{array}$$

Since polynomial orderings are, in general[5)], simplification orderings, we check the rules of $\mathcal{R}$ for being simplifying[6)]. The following definition represents a special homeomorphic embedding relation such that a term s is greater than a term t if, and only if, $s \succeq_{emb} t$ holds and s will be transformed into t by removing at least one operator with an arity greater than one[7)].

Definition 6.1 *The strong homeomorphic embedding relation $\succ_{semb}$ on two terms s and t is defined as*

$$\begin{array}{ll}
s \succ_{semb} t & \\
\Longleftrightarrow & s \succeq_{emb} t \quad \textit{and} \\
 & (\exists f \in s)\ \mathcal{A}r(f) \geq 2 \ \wedge \ f \notin t
\end{array}$$

For example, $f(x,y) \succ_{semb} x$ whereas $f(g(x),y) \not\succ_{semb} f(x,y)$. It is obvious that $s \succ_{POL} t$ if $s \succ_{semb} t$. Note that $[f(g(x),y)](x,y) \not\sqsupset [f(x,y)](x,y)$ if $[g](x) = x$.

Example 6.3 (Example 6.2 continued) *The initial TRS $\mathcal{R}$ will be transformed into*

$$\mathcal{R}' = \left\{ \begin{array}{lcl}
g(a) & \rightarrow & h(a,b,a) \\
i(x) & \rightarrow & f(x,x) \\
h(x,x,y) & \rightarrow & g(x)
\end{array} \right.$$

by using the strong homeomorphic embedding relation since $f(x,y) \succ_{semb} x$.

In order to reduce the number of variable coefficients[8)], we set superfluous coefficients to minimal values. A *superfluous* coefficient is a coefficient which corresponds to an operator which does not occur on the left-hand side of the rules of $\mathcal{R}'$. The interpretation of such an operator will be a strongly linear polynomial (which is the 'smallest' polynomial satisfying the monotonicity condition of definition 3.1).

Definition 6.2 *Let $\mathcal{F} = \bigcup\{f_i\}$ be a set of operators, $\mathcal{R} = \{l_i \rightarrow r_i \mid i \in [1,n]\}$ be a TRS over $\mathcal{T}(\mathcal{F},\mathcal{X})$ and $[.] = \bigcup\{[f_i](x_1,\ldots,x_{\mathcal{A}r(f_i)})\}$ be a set of polynomial interpretations for operators in $\mathcal{F}$. Then, the operation Ψ is defined in the following way:*

$$\Psi([.],\mathcal{R}) = \bigcup [f_i]'(x_1,\ldots,x_{\mathcal{A}r(f_i)}) \textit{ such that}$$

$$[f_i]'(x_1,\ldots,x_{\mathcal{A}r(f_i)}) = \left\{ \begin{array}{ll}
\mu & \textit{if} \ \ \mathcal{A}r(f_i) = 0 \ \wedge \ (\forall k \in [1,n]) f_i \notin l_k \\
\sum_{j=1}^{\mathcal{A}r(f_i)} x_j & \textit{if} \ \ \mathcal{A}r(f_i) > 0 \ \wedge \ (\forall k \in [1,n]) f_i \notin l_k \\
[f_i](x_1,\ldots,x_{\mathcal{A}r(f_i)}) & \textit{otherwise}
\end{array} \right.$$

---

5) if unary operators are not interpreted as identity functions

6) i.e. $l \succ_{POL} r$ independent of the used interpretations.

7) Note that $s \succ_{emb} t$ holds if, and only if, t is identical to s except that at least one (*arbitrary*) operator is removed.

8) and thus the time complexity of our algorithm

Note that the interpretation of an operator which has completely been removed[9], according to the relation $\succ_{semb}$, will be set to a minimal value using the operation Ψ. The function Ψ sets the interpretation of a superfluous constant symbol to the minimal value μ[10]. It is either possible to use μ as a variable or to instantiate μ with a specific value. Whenever μ is treated as a variable, we have to guarantee that μ really represents the minimum of $\mathcal{M}$. Therefore, the inequality $[c_i]() \geq \mu$ must be added for each $c_i \in \mathcal{F}$ for which $\mathcal{A}r(c_i) = 0$ holds. To continue with our example, we set μ to 1.

Example 6.4 (Example 6.3 continued) *Applying the operation Ψ to $\mathcal{R}'$ and $[.]$, we obtain:*

$$\begin{array}{lcl} \beta_0 := 1 & \leadsto & [b]() \quad\;\; = 1 \\ \chi_{11} := 0, \chi_{10} := 1, \chi_{01} := 1, \chi_{00} := 0 & \leadsto & [f](x,y) = x + y \end{array}$$

The remaining coefficients are left unchanged.

The rules of $\mathcal{R}'$ must be transformed into a set of inequalities by

(i) applying the variable interpretations and

(ii) splitting the resulting polynomials into inequalities w.r.t. the coefficients.

The second part will be performed using the following $\mathcal{S}plit$-operation.

Definition 6.3 *The operation $\mathcal{S}plit$ transforms a polynomial p into a set of inequalities. Let*

$$p = \sum(\alpha_{i_1 \dots i_n} - \beta_{i_1 \dots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \quad \text{such that} \quad \alpha_{i_1 \dots i_n} \geq 0 \text{ and } \beta_{i_1 \dots i_n} \geq 0.$$

Then,

$$\mathcal{S}plit(p) := \{\alpha_{i_1 \dots i_n} \geq \beta_{i_1 \dots i_n} \mid i_j \in \mathbb{N}\}.$$

The application of the variable interpretations as well as the splitting of the resulting polynomials into inequalities (w.r.t. the coefficients) will be performed for each rule of $\mathcal{R}'$. Therefore, for each $l_i \to r_i \in \mathcal{R}'$, we firstly generate the polynomial $p_i = [l_i] - [r_i]$ and then compute the set $I_i = \mathcal{S}plit(p_i)$.

Example 6.5 (Example 6.4 continued)

$$\begin{array}{lcl} [g(a)]() & = & \alpha_0^2\delta_2 + \alpha_0\delta_1 + \delta_0 \\ [h(a,b,a)]() & = & \alpha_0^2\epsilon_{111} + \alpha_0\epsilon_{110} + \alpha_0^2\epsilon_{101} + \alpha_0\epsilon_{011} + \alpha_0\epsilon_{100} + \epsilon_{010} + \alpha_0\epsilon_{001} + \epsilon_{000} \\ & & \\ [i(x)](x) & = & \phi_2 x^2 + \phi_1 x + \phi_0 \\ [f(x,x)](x) & = & 2x \\ & & \\ [h(x,x,y)](x,y) & = & \epsilon_{111}x^2y + (\epsilon_{101} + \epsilon_{011})xy + \epsilon_{110}x^2 + (\epsilon_{100} + \epsilon_{010})x + \epsilon_{001}y + \epsilon_{000} \\ [g(x)](x) & = & \delta_2 x^2 + \delta_1 x + \delta_0 \end{array}$$

$I_1 := \{\alpha_0^2\delta_2 + \alpha_0\delta_1 + \delta_0 \geq \alpha_0^2\epsilon_{111} + \alpha_0\epsilon_{110} + \alpha_0^2\epsilon_{101} + \alpha_0\epsilon_{011} + \alpha_0\epsilon_{100} + \epsilon_{010} + \alpha_0\epsilon_{001} + \epsilon_{000}\}$

$I_2 := \{\phi_2 \geq 0\,,\ \phi_1 \geq 2\,,\ \phi_0 \geq 0\}$

$I_3 := \{\epsilon_{111} \geq 0\,,\ \epsilon_{101} + \epsilon_{011} \geq 0\,,\ \epsilon_{110} \geq \delta_2\,,\ \epsilon_{100} + \epsilon_{010} \geq \delta_1\,,\ \epsilon_{001} \geq 0\,,\ \epsilon_{000} \geq \delta_0\}$

[9] i.e. the operator is contained neither in a left-hand side nor in a right-hand side of any rule in $\mathcal{R}'$

[10] see remark 3.1

At this stage, a simplification process removes redundant information from the sets of inequalities (see the following definition 6.4). This process maps an inequality into an easier one, for example, by deleting addends or factors occurring on both sides of the inequality. This transformation is described by inference rules of the form

$$\frac{x \geq y}{x' \geq y'} \quad \text{if} \quad A$$

meaning that $x' \geq y'$ will be generated if $x \geq y$ and condition A hold. If A is omitted, it will be treated as 'true'.

Definition 6.4 (Arithmetic laws) *Let x, y, z be arbitrary arithmetical expressions and α, α_i be coefficients of an interpretation. Then, the set of inference rules $\mathcal{AL}$, that describes arithmetic laws, is defined as follows:*

1) Removing common addends:

$$\frac{x+y \geq x+z}{y \geq z}$$

2) Removing common factors:

$$\frac{xy \geq xz}{y \geq z}$$

3) Removing obvious inequalities:

$$\frac{x \geq 0}{}$$ [11)]

4) Splitting trivial products:

$$\frac{(\prod_{i=1}^{n} \alpha_i) \leq 1}{(\forall i \in [1,n])\ \alpha_i := 1}$$ [12)]

5) Dealing with seemingly inconsistent inequalities:

$$\frac{\alpha \leq 0}{\alpha := 0}$$ *if the corresponding interpretation $[f]$ to α is of strict arity $\mathcal{A}r(f)$ and α has not been used as x in inference rule 2), otherwise stop with failure*

Example 6.6 (Example 6.5 continued) $I_2 := \{\phi_1 \geq 2\}$, $I_3 := \{\epsilon_{110} \geq \delta_2\ ,\ \epsilon_{100}+\epsilon_{010} \geq \delta_1\ ,\ \epsilon_{000} \geq \delta_0\}$ *whereas I_1 remains unchanged.*

In order to apply the Simplex method, the inequalities of each I_i must be linear. This condition will be achieved by (i) transforming the inequalities into inequalities each side of which represents only one product and by (ii) applying the binary logarithmic function to these products. The approximation of each side of an inequality to exactly one product is based on the arithmetic-mean-geometric-mean inequality (see, for example, [HLP52]):

$$\left(\frac{\sum_{i=1}^{n} p_i a_i}{\sum_{i=1}^{n} p_i}\right)^{\sum_{i=1}^{n} p_i} \geq \prod_{i=1}^{n} a_i^{p_i} \qquad \text{if } (\forall i \in [1,n])\, p_i > 0 \ \wedge\ a_i \geq 0$$

A detailed proof of this classical inequality is contained in [HLP52]. For our purpose, the case $(\forall i \in [1,n])\, p_i = 1$ is sufficient. Therefore, a slight transformation of the above inequality leads to

11) Sometimes, an inequality $x \geq 0$ can be helpful if $x > 0$ (see definition 6.7).

12) It is also possible to assign $\alpha_j := 0$ for a particular j instead of $\alpha_i := 1$ for all i.

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

The following lemma concludes the theoretical framework for our approximation.

Lemma 6.1 *Let $n \geq 1$, $a_i \geq 1$. Then,*

$$\begin{array}{llcll} i) & \prod_{i=1}^{n} a_i & \geq & \sum_{i=1}^{n} a_i & \quad if \;\; (\forall i \in [1,n]) \quad a_i \geq 2 \\ ii) & n \cdot \prod_{i=1}^{n} a_i & \geq & \sum_{i=1}^{n} a_i & \end{array}$$

The combination of the arithmetic-mean-geometric-mean inequality and lemma 6.1 leads to a method of transforming a general inequality into an inequality of only two products:

Lemma 6.2 *Let $a_i \geq 1$ and $b_i \geq 1$. Then,*

$$\begin{array}{lll} \sum_{i=1}^{n} a_i \geq \sum_{i=1}^{m} b_i \quad if & i) & n^n \prod_{i=1}^{n} a_i \geq \prod_{i=1}^{m} b_i^n \;\wedge\; (\forall i \in [1,m])\, b_i \geq 2 \\ \qquad\qquad\qquad\qquad or & ii) & n^n \prod_{i=1}^{n} a_i \geq m^n \prod_{i=1}^{m} b_i^n \end{array}$$

For example, $2xy + 3z \;\geq\; x + 2y + yz$ if $2^2 \cdot (2xy) \cdot (3z) \;\geq\; x^2 \cdot (2y)^2 \cdot (yz)^2$, $x \;\geq\; 2$, $2y \;\geq\; 2$ and $yz \;\geq\; 2$ by using the first transformation of lemma 6.2.

Example 6.7 (Example 6.6 continued) *In this example, we use the second case of lemma 6.2:*

$I_1 := \{27\alpha_0^3\delta_0\delta_1\delta_2 \;\geq\; 512\alpha_0^{24}\epsilon_{000}^3\epsilon_{001}^3\epsilon_{010}^3\epsilon_{100}^3\epsilon_{011}^3\epsilon_{101}^3\epsilon_{110}^3\epsilon_{111}^3\}$

$I_2 := \{\phi_1 \geq 2\}$

$I_3 := \{\epsilon_{110} \geq \delta_2\,,\, 4\epsilon_{100}\epsilon_{010} \geq \delta_1^2\,,\, \epsilon_{000} \geq \delta_0\}$

The power of our algorithm strongly depends on this transformation and thus on lemma 6.2. Accordingly, the better the approximations are the more powerful the algorithm will be. Note that the more variable coefficients exist the worse the approximation of lemma 6.2 will be because n and m (in lemma 6.2) increase. Moreover, in most cases the right-hand side increases faster than the left-hand side since b_i is transformed into b_i^n (see I_1 in example 6.7). In order to reduce the number of variables, we insert a simplification process (similar to the Ψ operation) before applying lemma 6.2. This simplification is optional and can be forced by the user or by a heuristic. The basic principle underlying this process is to assign zero to the coefficient of a constant monomial in an interpretation $[f]$ if $\mathcal{A}r(f) > 0$.

Definition 6.5 *Let $\mathcal{F} = \bigcup\{f_i\}$ be a set of operators and $[f_i](x_1, \ldots, x_{\mathcal{A}r(f_i)}) = \sum \alpha_{i,r_1 \ldots r_{\mathcal{A}r(f_i)}} \cdot x_1^{r_1} \cdot \ldots \cdot x_{\mathcal{A}r(f_i)}^{r_{\mathcal{A}r(f_i)}}$ be the interpretation of f_i.*

- $\mathcal{C}oe(f_i) \;=\; \bigcup \alpha_{i,r_1\ldots r_{\mathcal{A}r(f_i)}}$ *be the set of all coefficients*[13] *of $[f_i]$.*
- $\nu(\mathcal{C}oe(f_i)) \;=\; \bigcup \alpha_{i,r_1\ldots r_{\mathcal{A}r(f_i)}} \cup \{\alpha_{i,0\ldots 0} := 0\}$

For example, the operation ν sets α_0 to zero in $[f](x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$. Furthermore, the restriction to admit only simple-mixed interpretations can be exploited for unary operators:

[13] variable coefficients and coefficients which have specific values

Let f be a unary operator interpreted as $[f](x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$. Then, we can start the algorithm with $\alpha_2 = 0$. If this proves to be successful, the algorithm stops, otherwise it backtracks to the beginning and sets $\alpha_1 = 0$.

After these assignments, a modified Ψ-operation (denoted by Ψ_I, see the following definition) will be applied to the variable coefficients w.r.t. the actual set of inequalities, alternating with applications of the inference rules in $\mathcal{AL}$. The operation Ψ_I sets superfluous variables (i.e. variables which occur only on the right-hand side of an *inequality*[14)]) to minimal values.

Definition 6.6 *Let $\mathcal{F} = \bigcup\{f_i\}$ be a set of operators, I a set of inequalities and $[.]$ a set of polynomial interpretations for the operators in $\mathcal{F}$. Let $\mathcal{C}oe = \bigcup \mathcal{C}oe(f_i)$ be the set of all coefficients. Then, the operation Ψ_I is defined in the following way:*

$$\Psi_I(\mathcal{C}oe, I) \;=\; \bigcup \mathcal{C}oe'_i \qquad \textit{with} \qquad \mathcal{C}oe'_i \;=\; \bigcup\{\alpha'_{ij}\} \qquad \textit{such that}$$

$$\alpha'_{ij} \;=\; \begin{cases} 0 & \textit{if } \alpha_{ij} \textit{ has no specific value and } \mathcal{A}r(f_i) > 0 \textit{ and } \mathcal{C}oe(f_i) \setminus \{\alpha_{ij}\} \textit{ represents a polynomial of strict arity } \mathcal{A}r(f_i) \textit{ and } (\forall l_k \geq r_k \in I)\, \alpha_{ij} \notin l_k \\ 1 & \textit{if } \alpha_{ij} \textit{ has no specific value and } \mathcal{A}r(f_i) > 0 \textit{ and } \mathcal{C}oe(f_i) \setminus \{\alpha_{ij}\} \textit{ does not describe a polynomial of strict arity } \mathcal{A}r(f_i) \textit{ and } (\forall l_k \geq r_k \in I)\, \alpha_{ij} \notin l_k \\ \mu & \textit{if } \alpha_{ij} \textit{ has no specific value and } \mathcal{A}r(f_i) = 0 \textit{ and } (\forall l_k \geq r_k \in I)\, \alpha_{ij} \notin l_k \\ \alpha_{ij} & \textit{otherwise} \end{cases}$$

where $\alpha_{ij} \in \mathcal{C}oe_i$

Analogous to Ψ, the operation Ψ_I sets the interpretation of a superfluous constant symbol to μ. The minimal value μ of $\mathcal{M}$ can be used as a variable or as an instantiated natural number.

Example 6.8 (Example 6.6 continued) *Starting from example 6.6 with $\delta_0 := 0$, $\epsilon_{000} := 0$[15)] and $\delta_1 := 0$[16)] leads to the following sets:*

$I_1 := \{\alpha_0^2\delta_2 \geq \alpha_0^2\epsilon_{111} + \alpha_0\epsilon_{110} + \alpha_0^2\epsilon_{101} + \alpha_0\epsilon_{011} + \alpha_0\epsilon_{100} + \epsilon_{010} + \alpha_0\epsilon_{001}\}$

$I_2 := \{\phi_1 \geq 2\}$

$I_3 := \{\epsilon_{110} \geq \delta_2 \,,\, \epsilon_{100} + \epsilon_{010} \geq 0\}$

By applying the inference rules of $\mathcal{AL}$ we can remove the last inequality of I_3. The application of Ψ_I leads to $\epsilon_{111} := 0$, $\epsilon_{101} := 0$, $\epsilon_{011} := 0$, $\epsilon_{100} := 0$, $\epsilon_{010} := 0$[17)] and

$I_1 := \{\alpha_0^2\delta_2 \geq \alpha_0\epsilon_{110} + \alpha_0\epsilon_{001}\}$

$I_2 := \{\phi_1 \geq 2\}$

$I_3 := \{\epsilon_{110} \geq \delta_2\}$

The further application of the axioms contained in $\mathcal{AL}$ implies simplifying I_1 (by dividing both sides by α_0):

[14)] Remember that Ψ is operating on rewrite rules instead of inequalities. However, the purpose of Ψ and that of Ψ_I is the same.

[15)] We apply the operation ν to the function symbols g and h.

[16)] $\delta_2 = 0$ does not lead to a successful result. Therefore, we have to backtrack and assign zero to δ_1.

[17)] Note that $\epsilon_{001} \neq 0$ otherwise $[h](x, y, z)$ would not be of strict arity 3. One should try to generate the coefficients (which are zero) of an operator such that the resulting interpretation is minimal. A simple-mixed polynomial is 'smaller' than another one if it contains less strictly mixed monomials. For example, $xy + yz \succ xy + y + z$.

$I_1 := \{\alpha_0\delta_2 \geq \epsilon_{110} + \epsilon_{001}\}$

$I_2 := \{\phi_1 \geq 2\}$

$I_3 := \{\epsilon_{110} \geq \delta_2\}$

The transformation of the sets I_1, I_2 and I_3 into sets with inequalities both sides of which are products (applying the second case of lemma 6.2) leads to

$I_1 := \{\alpha_0\delta_2 \geq 2\epsilon_{110}\epsilon_{001}\}$

$I_2 := \{\phi_1 \geq 2\}$

$I_3 := \{\epsilon_{110} \geq \delta_2\}$

In order to apply the Simplex method, the inequalities must be linear. The transformation of the inequalities of the sets I_1, I_2 and I_3 is based on the following fact: $\prod a_i \geq \prod b_i$ iff $\sum lb(a_i) \geq \sum lb(b_i)$ where lb denotes the binary logarithm. Therefore, we apply lb to each side of the inequalities and rename the variables of the resulting inequalities, i.e. each $lb(\alpha_{j_1 \cdots j_n})$ will be transformed into $\alpha^*_{j_1 \cdots j_n}$.

Notation 6.1 *The transformation according to lemma 6.2 of a general inequality into an inequality where both sides contain only one product will be denoted by red: $red(\sum a_i \geq \sum b_i)$. The transformation of an inequality of two products into a linear inequality (by using lb) is denoted by lin: $lin(\prod a_i \geq \prod b_i)$.*

Example 6.9 (Example 6.8 continued)

$I_1 := \{\alpha_0^* + \delta_2^* \geq 1 + \epsilon_{110}^* + \epsilon_{001}^*\}$

$I_2 := \{\phi_1^* \geq 1\}$

$I_3 := \{\epsilon_{110}^* \geq \delta_2^*\}$

Note that each set I_i must contain at least one strict inequality ($l_i > r_i$) in order to guarantee that the corresponding rule of $\mathcal{R}$ can be oriented. This will be achieved by adding new variables, one variable for each inequality:

Definition 6.7 *Let I be a set of inequalities. Then, the operation $\mathcal{C}om$ is defined as*

$$\mathcal{C}om(I) := \{l_i \geq r_i + \psi_i \mid l_i \geq r_i \in I\} \cup \{\sum_i \psi_i > 0\}$$

such that ψ_i are new variables over $\mathbb{R}$.

After this process, all sets I_i of inequalities will be *collected* in one set I which represents the final set of inequalities, that can serve as the input for the Simplex method.

Example 6.10 (Example 6.9 continued)

$$\begin{aligned} I := \{ \quad & \alpha_0^* + \delta_2^* \geq 1 + \epsilon_{110}^* + \epsilon_{001}^* + \psi_1 \,,\ \psi_1 > 0 \,, \\ & \phi_1^* \geq 1 + \psi_2 \,,\ \psi_2 > 0 \,, \\ & \epsilon_{110}^* \geq \delta_2^* + \psi_3 \,,\ \psi_3 > 0 \quad \} \end{aligned}$$

The last part of our algorithm concerns the computation of a solution of the generated set of inequalities. As noted, we use the Simplex method in order to do this. It is sufficient to apply the first phase of the Simplex method for computing a feasible solution if it exists. A brief description as well as some well-known remarks about its time complexity can be found in [Ste91]. Assume now that the Simplex algorithm provides a solution. For deducing the resulting interpretations of the operators from this solution, we have to retract the renaming of the coefficients (as used with lb) by replacing them by their corresponding initial names, i.e. $\alpha_{j_1 \cdots j_n} = 2^{\alpha^*_{j_1 \cdots j_n}}$.

Example 6.11 (Example 6.10 continued) *The following assignments represent a solution of the set I of inequalities:*

$$\alpha_0^* = 3\,,\ \delta_2^* = 0\,,\ \phi_1^* = 2\,,\ \epsilon_{110}^* = 1\,,\ \epsilon_{001}^* = 0\ (\psi_1 = \psi_2 = \psi_3 = 1).$$

Therefore, the interpretations $[a]() = 8$, $[b]() = 1$, $[g](x) = x^2$, $[i](x) = 4x$, $[f](x,y) = x+y$, $[h](x,y,z) = 2xy + z$ prove the termination of $\mathcal{R}$.

Before presenting the whole algorithm, a modification of the $\mathcal{S}plit$-operation is pointed out which sometimes proves more profitable. It is a slightly modified version of the approach contained in [Ste91].

Definition 6.8 *The operation $\mathcal{S}plit^E$ transforms a polynomial p into a set of inequalities. Let*

$$p = \sum(\alpha_{i_1\ldots i_n} - \beta_{i_1\ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \quad \text{such that} \quad \alpha_{i_1\ldots i_n} \geq 0,\ \beta_{i_1\ldots i_n} \geq 0$$
$$\text{and let } \gamma_{l_1\ldots l_n k_1 \ldots k_n} \text{ be new variables over } \mathbb{R}.$$

$$\begin{aligned}
\mathcal{S}plit^E(p) \ &:= \ J \cup \{ \sum_{k_j \geq i_j} \mu^{\sum(k_j - i_j)} \cdot \gamma_{k_1\ldots k_n i_1 \ldots i_n} \geq \beta_{i_1\ldots i_n} \mid \beta_{i_1\ldots i_n} > 0, \alpha_{k_1\ldots k_n} > 0\} \\
\text{such that} \quad J \ &:= \ \{\alpha_{i_1\ldots i_n} \geq \sum_{k_j \leq i_j} \gamma_{i_1\ldots i_n k_1\ldots k_n} \mid \alpha_{i_1\ldots i_n} > 0\,,\ \beta_{k_1\ldots k_n} > 0\} \\
\text{If} \quad J \ &= \ J' \cup \{\alpha_{i_1\ldots i_n} \geq \gamma_{i_1\ldots i_n k_1\ldots k_n}\} \text{ then let } J \ := \ J' \cup \{\gamma_{i_1\ldots i_n k_1\ldots k_n} := \alpha_{i_1\ldots i_n}\}.
\end{aligned}$$

The operation $\mathcal{S}plit^E(p)$ generates a set of linear inequalities if the coefficients of p are given. For more details about this technique, see [Ste91].

Example 6.12 *The operation $\mathcal{S}plit^E$ is illustrated by applying it to example 6.1. Suppose, we have the interpretations of the terms given at the beginning of example 6.5. Then, since $\beta_0 = 1$, we have to set the minimum μ of $\mathcal{M}$ to 1: $\mu = 1$.*

$$I_1 := \{\alpha_0^2\delta_2 + \alpha_0\delta_1 + \delta_0 \geq \alpha_0^2\epsilon_{111} + \alpha_0\epsilon_{110} + \alpha_0^2\epsilon_{101} + \alpha_0\epsilon_{011} + \alpha_0\epsilon_{100} + \epsilon_{010} + \alpha_0\epsilon_{001} + \epsilon_{000}\}$$

$$I_2 := \{\phi_2 + \phi_1 \geq 2\,,\ \phi_0 \geq 0\}$$

$$\begin{aligned}
I_3 := \{&\epsilon_{111} \geq \epsilon_{111_{110}} + \epsilon_{111_{100}} + \epsilon_{111_{000}}\,,\ \epsilon_{101} + \epsilon_{011} \geq \epsilon_{101_{100}} + \epsilon_{101_{000}}\,,\ \epsilon_{110} \geq \epsilon_{110_{110}} + \epsilon_{110_{100}} + \\
&\epsilon_{110_{000}}\,,\ \epsilon_{100} + \epsilon_{010} \geq \epsilon_{100_{100}} + \epsilon_{100_{000}}\,,\ \epsilon_{111_{110}} + \epsilon_{110_{110}} \geq \delta_2\,,\ \epsilon_{111_{100}} + \epsilon_{101_{100}} + \epsilon_{110_{100}} + \epsilon_{100_{100}} \geq \\
&\delta_1\,,\ \epsilon_{111_{000}} + \epsilon_{101_{000}} + \epsilon_{110_{000}} + \epsilon_{100_{000}} + \epsilon_{001} + \epsilon_{000} \geq \delta_0\}
\end{aligned}$$

The operation $\mathcal{S}plit^E$ can be used instead of $\mathcal{S}plit$. This means, that our algorithm has this operation as an option. All in all, algorithm 6.1, which is given in figure 6.1, is parameterized by the following five features:

- The specification of the mode for splitting the rules into inequalities ($\mathcal{S}plit$ or $\mathcal{S}plit^E$).
- The specification of the method for approximating general inequalities by linear ones (choosing i) or ii) of lemma 6.2).
- The specification of operators with no constant monomial[18)] (i.e. $[f](x_1,\ldots,x_n) = \sum \alpha_{i_1\ldots i_n} x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ such that $\alpha_{0\ldots 0} = 0$).
- The specification of unary operators with square interpretations (i.e. determine whether either $\alpha_1 = 0$ or $\alpha_2 = 0$ for $[f](x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$).

[18)] by applying the operation ν to some operators

- The specification of μ as to whether it is a variable or fixed representing a certain value.

Theorem 6.1 *Algorithm 6.1 always terminates. If it does not fail, $\mathcal{R}$ can be directed by a polynomial ordering (based on simple-mixed interpretations) generated by the algorithm.*

We have implemented a prototype of algorithm 6.1 (see [Spr91]). Appendix B contains some examples run with this implementation. However, we do not have detailed experiences concerning the application of the algorithm to examples, so far. Thus, a thorough investigation of algorithm 6.1 will be part of future plans. Additionally, in order to improve its applicability, we intend to merge algorithm 6.1 and the heuristics developed in chapter 5. In particular, the application of lemma 5.1 for detecting the necessity of the interpretations for a unary operator to be quadratic (instead of linear, only) could lead to interesting results.

[19] Otherwise add the equation $\alpha_{i,2}\alpha_{i,1} = 0$ to the inequalities.

Algorithm 6.1 *This algorithm determines whether a finite TRS* $\mathcal{R}$ *over* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *can be oriented by a polynomial ordering using simple-mixed interpretations for the operators in* $\mathcal{F}$. *Let* $\mathcal{F} = \bigcup\{f_i\}$.

1. **Initialization:**

   1.1 *Defining the interpretations (by using variable coefficients):*
   $[.]$ *is defined to be the union of all* $[f_i](x_1,\ldots,x_{n_i}) = p(x_1,\ldots,x_{n_i})$ *where* $\mathcal{A}r(f_i) = n_i$ *and* p *is a complete simple-mixed polynomial for* n_i *variables.*

   1.2 *Removing trivial rules:*
   $\mathcal{R}' := \mathcal{R} \setminus \{l \to r \mid l \to r \in \mathcal{R} \ \wedge \ l \succ_{semb} r\}$

   1.3 *Setting superfluous interpretations to minimal values:*
   $[.] \ := \ \Psi([.],\mathcal{R}')$

2. **Splitting the rules into inequalities w.r.t. the coefficients:**
   $(\forall l_k \to r_k \in \mathcal{R}')\ I_k := \mathcal{S}plit([l_k]-[r_k])$ *or* $I_k := \mathcal{S}plit^E([l_k]-[r_k])$

3. **Simplification:**

   3.1 *Removing special coefficients (optional):*
   · $\mathcal{C}oe(f_i) := \nu(\mathcal{C}oe(f_i))$ *for some* i, *if* $\mathcal{A}r(f_i) > 0$
   · $\alpha_{i,2} := 0$ *or* $\alpha_{i,1} := 0$ *if* $\mathcal{A}r(f_i) = 1$ *and* $[f_i](x) = \alpha_{i,2}x^2 + \alpha_{i,1}x + \alpha_{i,0}$ [19)]

   3.2 *Applying arithmetic laws:*
   $(\forall k)\ I_k := \{l'_i \geq r'_i \mid l_i \geq r_i \in I_k\,,\ \frac{l_i \geq r_i}{l'_i \geq r'_i} \in \mathcal{AL} \cup \{\frac{l>r}{l \geq r}\}\}$

   3.3 *Applying the* Ψ_I*-operation:*
   $\bigcup_i \mathcal{C}oe(f_i) := \Psi_I(\bigcup_i \mathcal{C}oe(f_i), \bigcup_k I_k)$

   Perform these steps (3.1 - 3.3) once again until no changes occur.

4. **Eliminating addition and reducing a product to a linear polynomial:**
   $(\forall k)\ I'_k := \{lin(red(x \geq y)) \mid x \geq y \in I_k\}$

5. **Completing and collecting:**
   $I := \bigcup_k \mathcal{C}om(I'_k)$

6. **Applying the Simplex method:**
   The Simplex method will possibly generate a solution of the derived set I *of linear inequalities. Afterwards, each* a_i^* *of the solution must be replaced by its corresponding initial name* a_i. *This will be done by reversing the logarithmic mapping:* $\alpha_i := 2^{\alpha_i^*}$.

Figure 6.1: The algorithm of chapter 6

Chapter 7

An Improved Polynomial Ordering

In this chapter we present an extension of the polynomial ordering $\succ_{POL}$ which is also easier to handle than the original polynomial ordering. It is based on the following approach of Lankford.

Definition 7.1 ([Lan79]) *Let $\succ$ be a precedence on $\mathcal{F}$. If there exists a unary operator f interpreted* 8.9
as the identity function[1]*, then no other operator is allowed to be greater than f w.r.t. $\succ$. For each* 8.11
constant $c, [c]() \geq 2$. The polynomial ordering $\succ_{POL^*}$ on two terms s and t is defined as 8.12

$$
\begin{array}{lll}
s \succ_{POL^*} t & & \\
\Longleftrightarrow & & s \equiv f(t) \quad or \\
& & s \succ_{POL} t \quad or \\
& & s =_{POL} t \ \wedge \ \mathcal{H}ead(s) \succ \mathcal{H}ead(t) \quad or \\
& & s =_{POL} t \ \wedge \ \mathcal{H}ead(s) = \mathcal{H}ead(t) \ \wedge \ \mathcal{A}rgs(s) \succ_{POL^*}^{lex} \mathcal{A}rgs(t)
\end{array}
$$

Obviously, $s =_{POL} t$ holds if, and only if, the interpretations of s and t are syntactically identical, i.e. $[s] = [t]$. It is possible to extend the above definition by using a quasi precedence[2] $\succsim$ and replacing $'\mathcal{H}ead(s) = \mathcal{H}ead(t)'$ by $'\mathcal{H}ead(s) \sim \mathcal{H}ead(t)'$.

Note that the conditions (i) $\nexists g \succ f$ if $[f](x) = x$ and (ii) for each constant c, $[c]() \geq 2$ are responsible for guaranteeing $\succ_{POL^*}$ to be well-founded. Consider the following examples:

- The rule $h(x) \rightarrow g(h(x))$ is not terminating but it would be directed (as indicated) by using $\succ_{POL^*}$ (based on $[g](x) = x$ and $h \succ g$) if (i) was not required.
- The orientation of the non-terminating rule $a \rightarrow f(a, a)$ could be achieved with the help of $\succ_{POL^*}$ (based on $[f](x, y) = xy$, $[a]() = 1$ and $a \succ f$) if (ii) was not guaranteed.

The improved polynomial ordering which is an extension of $\succ_{POL^*}$ is based on a status function (see chapter 2).

Definition 7.2 *Let $\succ$ be a precedence and τ a status function on $\mathcal{F}$. If there exists a unary operator* 8.14
f interpreted as the identity function, then no other operator is allowed to be greater than f w.r.t. $\succ$. For each constant $c, [c]() \geq 2$. The improved polynomial ordering $\succ_{IPOL}$ on two terms s and t is defined as

$$
\begin{array}{lll}
s \succ_{IPOL} t & \Longleftrightarrow & s \equiv f(t) \quad or \\
& & s \succ_{POL} t \quad or \\
& & s =_{POL} t \ \wedge \ \mathcal{H}ead(s) \succ \mathcal{H}ead(t) \quad or \\
& & s =_{POL} t \ \wedge \ \mathcal{H}ead(s) = \mathcal{H}ead(t) \ \wedge \ \mathcal{A}rgs(s) \succ_{IPOL, \tau(\mathcal{H}ead(s))} \mathcal{A}rgs(t)
\end{array}
$$

[1] i.e. $[f](x) = x$.

[2] A quasi ordering $\succsim$ is a binary, *reflexive* and transitive relation. $\sim$ denotes the equivalence relation given by $\succsim \cap \precsim$.

The index '$\tau(\mathcal{H}ead(s))$' denotes the extension of $\succ_{IPOL}$ w.r.t. the status of the operator $\mathcal{H}ead(s)$. The idea of this ordering goes back to Lankford (see definition 7.1). His improvement (i.e. $\succ_{POL^*}$) of the original polynomial ordering is a restricted version of $\succ_{IPOL}$ that assigns left-to-right status to all operators. It is influenced by the Knuth-Bendix ordering ([KB70]).

Theorem 7.1 *$\succ_{IPOL}$ is a simplification ordering on $\mathcal{G}(\mathcal{F})$.*

During the computation of a lot of examples (see [SK90]) the generation of appropriate interpretations for $\succ_{IPOL}$ was easier than for the original polynomial ordering $\succ_{POL}$. This result is based on the following technique:

> We use the 'semantics' (if known) of the operators as their interpretations. This will 8.9
> equalize both sides of an equation[3] w.r.t. $=_{POL}$. An additional consideration often 8.12
> provides an appropriate precedence which causes the orientation of the equation.

Example 7.1 *Let*

$$\mathcal{R} = \begin{cases} x+0 & \rightarrow x \\ x+s(y) & \rightarrow s(x+y) \\ f(0) & \rightarrow 0 \\ f(s(x)) & \rightarrow (s(f(x))+x)+x \\ (x+f(y))+z & \rightarrow x+(z+f(y)) \end{cases}$$

The operator + describes the addition on $\mathbb{N}$ and s represents the successor function:

$$[+](x,y) = x+y \ , \ [s](x) = x+1.$$

The recursion appearing in the definition of the operator f can be resolved into $f(x) = x^2$. Therefore, we use

$$[f](x) = x^2.$$

All these interpretations together with $[0]() = 2$ cause both sides of all rules (except the first one) to be equivalent w.r.t. $=_{POL}$. According to the definition of the $\succ_{IPOL}$ we have to compare the leading function symbols: + and s, f and 0, f and +, + and +. An extension of the precedence by

$$+ \succ s, f \succ 0, f \succ +$$

and of the status by $\tau(+) = left$ will cause the desired orientation. Note that the first rule is oriented since $[x+0] = x+2 \sqsupset x = [x]$.

There already exist two other improvements of polynomial orderings. The first one concerns the classes of functions used as interpretations. It is, for example, possible to include exponential functions (see, 8.9
for example, [Der83]). This ordering will be denoted by $\succ_{EXP}$. 8.10

The second improvement of the original polynomial ordering is based on the concept of concatenation of well-founded orderings. In [Der83], a convenient set of basic laws is provided by which well-founded orderings can be constructed. One of these laws implies that a lexicographic ordering of fixed-length tuples is well-founded if the orderings on the components are well-founded. This idea has been applied to the polynomial ordering in such a way that several different polynomial orderings (based on distinct interpretations) can be concatenated. This technique is called polynomial ordering with interpretations taken from a Cartesian product of polynomials (see, for example, [Ben86] and [BL87]). 8.9
In the remaining part of this paper, we refer to this ordering by $\succ_{CPOL}$.

[3] Note that an equation is valid w.r.t. a given theory.

Chapter 8

Examples

This chapter deals with the illustration of polynomial orderings by applying them to examples. All examples will be uniformly presented. They consist of a TRS $\mathcal{R}$ and a polynomial ordering $\succ_{POL}$ (defined in chapter 3), an improved polynomial ordering $\succ_{IPOL}$ or $\succ_{POL^*}$ (both defined in chapter 7), a polynomial ordering $\succ_{CPOL}$ based on Cartesian products or a polynomial ordering $\succ_{EXP}$ using exponential functions (both mentioned in chapter 7) for guaranteeing the termination of $\mathcal{R}$.

Example 8.1

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(f(x)) & \rightarrow & g(g(x)) \\ g(g(f(x))) & \rightarrow & f(g(g(x))) \end{array} \right.$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl} [f](x) & = & 2x+1 \\ [g](x) & = & 2x \end{array}$$

Example 8.2 (Fibonacci group)

$$\mathcal{R}^{1)} = \left\{ \begin{array}{lcl} c & \rightarrow & ab \\ d & \rightarrow & bc \\ e & \rightarrow & cd \\ de & \rightarrow & a \\ ea & \rightarrow & b \end{array} \right.$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl} [a](x) & = & x+1 \\ [b](x) & = & 2x+4 \\ [c](x) & = & 2x+6 \\ [d](x) & = & 4x+17 \\ [e](x) & = & 8x+41 \end{array}$$

[1)] The rule system $\mathcal{R}$ contains only unary operators. Thus, we use strings to describe $\mathcal{R}$ (i.e. $f(x)$ is represented by f).

Example 8.3 (Prime numbers)

$$\mathcal{R}^{2)} = \begin{cases} prime(0) & \rightarrow & false \\ prime(s(0)) & \rightarrow & false \\ prime(s(s(x))) & \rightarrow & prime1(s(s(x)), s(x)) \\ prime1(x, 0) & \rightarrow & false \\ prime1(x, s(0)) & \rightarrow & true \\ prime1(x, s(s(y))) & \rightarrow & \neg divp(s(s(y)), s) \wedge prime1(x, s(y)) \\ divp(x, y) & \rightarrow & rem(x, y) \equiv 0 \end{cases}$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl} [prime](x) & = & x^3 \\ [prime1](x, y) & = & xy^2 \\ [divp](x, y) & = & x + y + 3 \\ [rem](x, y) & = & x + y \\ [\equiv](x, y) & = & x + y \\ [\wedge](x, y) & = & x + y \\ [\neg](x) & = & x + 1 \\ [s](x) & = & 2x \\ [0]() & = & 2 \\ [false]() & = & 2 \\ [true]() & = & 2 \end{array}$$

Example 8.4 (Taussky group)

$$\mathcal{R}^{3)} = \begin{cases} x * (y * z) & \rightarrow & (x * y) * z \\ 1 * 1 & \rightarrow & 1 \\ x * i(x) & \rightarrow & 1 \\ i(x * y) & \rightarrow & i(y) * i(x) \\ g(x * y, y) & \rightarrow & f(x * y, x) \\ f(1, y) & \rightarrow & y \end{cases}$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl} [*](x, y) & = & 2xy + y \\ [f](x, y) & = & x + y \\ [g](x, y) & = & 2xy^2 + x + y^2 + 1 \\ [i](x) & = & x^2 \\ [1]() & = & 2 \end{array}$$

Example 8.5 (Boolean ring)

$$\mathcal{R} = \begin{cases} x \supset y & \rightarrow & (x * y) + (x + 1) \\ x \vee y & \rightarrow & (x * y) + (x + y) \\ x \equiv y & \rightarrow & x + (y + 1) \\ \neg x & \rightarrow & x + 1 \end{cases}$$

and $\succ_{POL}$ *is based on*

---

2) Note that $rem(x, y)$ stands for the remainder when dividing x by y.

3) Note that the group axioms (see heuristic 5.2) are part of the system.

$$\begin{array}{lcl}
[+](x,y) & = & x+y \\
[*](x,y) & = & x+y \\
[\supset](x,y) & = & 2x+y+2 \\
[\vee](x,y) & = & 2x+2y+1 \\
[\equiv](x,y) & = & x+y+2 \\
[\neg](x) & = & x+2 \\
[1]() & = & 1
\end{array}$$

Example 8.6 (Iterative version of the reverse function)

$$\mathcal{R} = \left\{ \begin{array}{lcl}
nil \circ y & \rightarrow & y \\
(x.y) \circ z & \rightarrow & x.(y \circ z) \\
(x \circ y) \circ z & \rightarrow & x \circ (y \circ z) \\
rev(nil) & \rightarrow & nil \\
rev(x.y) & \rightarrow & rev(y) \circ (x.nil) \\
reviter(nil, y) & \rightarrow & y \\
reviter(x.y, z) & \rightarrow & reviter(y, x.z) \\
rev(x) \circ y & \rightarrow & reviter(x, y) \\
rev(x) & \rightarrow & reviter(x, nil)
\end{array} \right.$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl}
[.](x,y) & = & xy+y+1 \\
[\circ](x,y) & = & xy+x \\
[rev](x) & = & 2x^2 \\
[reviter](x,y) & = & xy+x \\
[nil]() & = & 2
\end{array}$$

Example 8.7 (Associativity and Distributivity)

$$\mathcal{R} = \left\{ \begin{array}{lcl}
x + (y+z) & \rightarrow & (x+y)+z \\
x * (y+z) & \rightarrow & (x*y)+(x*z) \\
(x+(y*z))+(y*u) & \rightarrow & x+(y*(z+u))
\end{array} \right.$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl}
[+](x,y) & = & x+2y+1 \\
[*](x,y) & = & xy
\end{array}$$

Example 8.8 (Binomial coefficients)

$$\mathcal{R} = \left\{ \begin{array}{lcl}
bin(x,0) & \rightarrow & s(0) \\
bin(0,s(y)) & \rightarrow & 0 \\
bin(s(x),s(y)) & \rightarrow & bin(x,s(y)) + bin(x,y)
\end{array} \right.$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl}
[bin](x,y) & = & xy+x \\
[s](x) & = & 2x \\
[+](x,y) & = & x+y \\
[0]() & = & 2
\end{array}$$

Example 8.9 (Addition and Multiplication)

$$\mathcal{R} = \begin{cases} x * 0 & \rightarrow & 0 \\ x * s(y) & \rightarrow & x + (x * y) \\ x + 0 & \rightarrow & x \\ x + s(y) & \rightarrow & s(x + y) \end{cases}$$

$\succ_{EXP}$ is based on $[*](x,y) = 2^{xy}, [+](x,y) = x + 2y, [s](x) = x + 1, [0]() = 2$

$\succ_{CPOL}$ is based on *i)* $[*](x,y) = xy, [+](x,y) = x + y, [s](x) = x + 2, [0]() = 2$

ii) $[*](x,y) = xy, [+](x,y) = xy, [s](x) = x + 2, [0]() = 2$

$\succ_{IPOL}$ and $\succ_{POL^*}$ are based on $[*](x,y) = xy, [+](x,y) = x + y, [s](x) = x + 1, [0]() = 2, * \succ + \succ s$

There exists no $\succ_{POL}$ for orienting the above system.

Example 8.10 (Fibonacci function)

$$\mathcal{R} = \begin{cases} f(0) & \rightarrow & s(0) \\ f(s(0)) & \rightarrow & s(0) \\ f(s(s(x))) & \rightarrow & f(s(x)) + f(x) \\ x + 0 & \rightarrow & x \\ x + s(y) & \rightarrow & s(x + y) \end{cases}$$

and $\succ_{EXP}$ is based on

$$\begin{array}{lcl} [+](x,y) & = & x + 2y \\ [f](x) & = & 2^{x+1} \\ [s](x) & = & x + 2 \\ [0]() & = & 1 \end{array}$$

Note that $\succ_{IPOL}$ cannot orient the above system.

Example 8.11 (Division)

$$\mathcal{R} = \begin{cases} x/x & \rightarrow & 1 \\ x/1 & \rightarrow & x \\ i(x/y) & \rightarrow & y/x \\ (x/y)/z & \rightarrow & x/(z/i(y)) \end{cases}$$

and $\succ_{IPOL}$ as well as $\succ_{POL^*}$ are based on

$$\begin{array}{lcl} [/](x,y) & = & x + y \\ [i](x) & = & x \\ [1]() & = & 2 \end{array}$$

and $i \succ /$. Note that there exists no $\succ_{POL}$ for orienting $\mathcal{R}$.

Example 8.12 (Summation)

$$\mathcal{R} = \begin{cases} sum(0) & \rightarrow & 0 \\ sum(s(x)) & \rightarrow & sum(x) + s(x) \\ x + 0 & \rightarrow & x \\ x + s(y) & \rightarrow & s(x + y) \\ s(x) + y & \rightarrow & s(x + y) \\ (x + y) + z & \rightarrow & x + (y + z) \end{cases}$$

and $\succ_{IPOL}$ *as well as* $\succ_{POL^*}$ *are based on*

$$\begin{array}{lcl} [+](x,y) & = & x+y \\ [sum](x) & = & x^2 \\ [s](x) & = & x+1 \\ [0]() & = & 2 \end{array}$$

and $+ \succ s$.

Example 8.13 ([Pau84])

$$\mathcal{R} = \left\{ \begin{array}{lcl} x+0 & \to & x \\ x*1 & \to & x \\ x*0 & \to & 0 \\ x*(y+z) & \to & (x*y)+(x*z) \end{array} \right.$$

and $\succ_{POL}$ *is based on*

$$\begin{array}{lcl} [*](x,y) & = & xy \\ [+](x,y) & = & x+y+1 \\ [1]() & = & 2 \\ [0]() & = & 2 \end{array}$$

Example 8.14

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(h(s(x)),y) & \to & f(y,s(x)) \\ h(s(x)) & \to & s(h(h(x))) \end{array} \right.$$

and $\succ_{IPOL}$ *is based on*

$$\begin{array}{lcl} [f](x,y) & = & x+y \\ [h](x) & = & x \\ [s](x) & = & x+1 \end{array}$$

and $h \succ s$, $\tau(f) = mul$. *Note that* $\succ_{POL^*}$ *cannot orient* $\mathcal{R}$.

Bibliography

[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, October 1987.

[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.

[HL89] Dieter Hofbauer and Clemens Lautemann. Termination proofs and the length of derivations. In N. Dershowitz, editor, *Proc. 3rd RTA (LNCS 355)*, pages 167–177, Chapel Hill (North Carolina), April 1989.

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

[KL80] Sam Kamin and Jean-Jacques Lévy. Attempts for generalizing the recursive path orderings. Urbana (Illinois), February 1980.

[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.

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

[Lan86] Dallas S. Lankford. Some remarks on rewrite rule termination methods which use polynomial interpretations. Unpublished Manuscript, 1986.

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

[Mar87] Ursula Martin. How to choose the weights in the Knuth-Bendix ordering. In P. Lescanne, editor, *Proc. 2nd RTA*, pages 42–52, Bordeaux (France), 1987.

[Mid88] Aart Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. Technical report, Dept. of Mathematics and Computer Science, Vrije Univ., Vrije (The Netherlands), September 1988.

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

[Pau84] Etienne Paul. Proof by induction in equational theories with relations between constructors. In *Proc. 9th Colloquium on Trees in Algebra and Programming (CAAP)*, pages 211–225, Bordeaux (France), March 1984.

[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.

[SK90] Joachim Steinbach and Ulrich Kühler. Check your ordering – Termination proofs and open problems. SEKI-Report SR-90-25, Univ. of Kaiserslautern, Kaiserslautern (Germany), 1990.

[Spr91] Christof Sprenger. Implementierung eines Verfahrens zur Generierung von Polynomordnungen. Projektarbeit, FB Informatik, Universität Kaiserslautern, 1991. to appear.

[Ste89] Joachim Steinbach. Extensions and comparison of simplification orderings. In N. Dershowitz, editor, *Proc. 3rd RTA (LNCS 355)*, pages 434–448, Chapel Hill (North Carolina), April 1989.

[Ste91] Joachim Steinbach. A method to prove the positiveness of polynomials. SEKI-Report, Univ. of Kaiserslautern, Kaiserslautern (Germany), 1991.

[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. SEKI-Report SR-90-03, Univ. of Kaiserslautern, Kaiserslautern (Germany), 1990.

Appendix A

Proofs

Lemma A.1 *Let $l \to r$ be a rule. No linear interpretation for the unary operator i exists if one of the following conditions holds:*

- $(\exists u \in \mathcal{P}os(l)) \quad l|_u = i(s)$
  $\wedge\ (\forall h \in \mathcal{F})\, i \neq h \in l \leadsto [h](x_1, \ldots, x_{\mathcal{A}r(h)})$ *is linear*
  and
  $(\exists v \in \mathcal{P}os(r)) \quad r|_v = g(t_1, \ldots, t_{\mathcal{A}r(g)})$
  $\wedge\ [g](x_1, \ldots, x_{\mathcal{A}r(g)})$ *is not linear* $\wedge\ \mathcal{V}ar(r|_v) \subseteq \mathcal{V}ar(l|_u)$
- $l = i(f(x, y))$ *and* $r = f(i(y), i(x))$

Proof:

i) Let $[.]$ be an interpretation such that $[g](x_1, \ldots, x_{Ar(g)})$ is strictly separate. Assume that $[i](x)$ is linear.

$\leadsto$
- $[r](y_1, \ldots, y_n) = P + \alpha y_1^{k_1} \cdot \ldots \cdot y_n^{k_n}$ where P is a polynomial and $(\exists i \in [1, n])\; k_i > 1$ since $g \in r$ and $[g]$ is not linear
- $[l](y_1, \ldots, y_n) = \sum_{i=1}^{n} \beta_i y_i$ since $(\forall h \in \mathcal{F})$ $[h](x_1, \ldots, x_{Ar(h)})$ is linear

$\leadsto$ $[l] \not>_{POL} [r]$

ii) Let $[f](x, y) = \Sigma \alpha_{r_1 r_2} x^{r_1} y^{r_2}$. Assume that $[i](x) = \beta x + \gamma$.

$$
\begin{array}{llll}
\leadsto & [l] & = & \beta \cdot \Sigma \alpha_{r_1 r_2} x^{r_1} y^{r_2} + \gamma \\
 & [r] & = & \Sigma \alpha_{r_1 r_2} (\beta y + \gamma)^{r_1} (\beta x + \gamma)^{r_2} \\
\leadsto & \beta & = & 1 \quad \text{and} \\
 & \gamma & = & 0 \\
\leadsto & [l] & = & \Sigma \alpha_{r_1 r_2} x^{r_1} y^{r_2} \\
 & [r] & = & \Sigma \alpha_{r_1 r_2} x^{r_2} y^{r_1} \\
\leadsto & [l] & \not>_{POL} & [r]
\end{array}
$$

□

Lemma A.2 *Let $l \to r$ be a rule. No separate interpretation for the operator f with $\mathcal{A}r(f) \geq 2$ exists if one of the following conditions holds:*

- $l = f(s_1, \ldots, s_m)$ *and*

$$\begin{array}{ll}(\exists k) & \mathcal{V}ar(s_k) \cap \mathcal{V}ar(l[\lambda]_k) = \emptyset \\ & \wedge\ (\exists v \in \mathcal{P}os(r))\ r|_v = f(t_1, \ldots, t_m)\ \wedge\ t_k \succeq_{emb} s_k \\ & \wedge\ \mathcal{V}ar(r[\lambda]_{v.k}) \cap \mathcal{V}ar(s_k) \neq \emptyset\end{array}$$

- $(\exists u \in \mathcal{P}os(l))\quad l|_u = f(s_1, \ldots, s_m)$
  $\qquad \wedge\ (\forall h \in \mathcal{F})\, f \neq h \in l \leadsto [h](x_1, \ldots, x_{\mathcal{A}r(h)})$ *is linear*
  and
  $(\exists v \in \mathcal{P}os(r))\quad r|_v = g(t_1, \ldots, t_{\mathcal{A}r(g)})$
  $\qquad \wedge\ [g](x_1, \ldots, x_{\mathcal{A}r(g)})$ *is strictly mixed* $\wedge\ \mathcal{V}ar(r|_v) \subseteq \mathcal{V}ar(l|_u)$

- $l = i(f(x, y)), r = f(i(y), i(x))$ *and* f *is associative*[1)]

Proof:

i) Assume that $[f](x_1, \ldots, x_m) = \sum_{i=1}^{m} p_i(x_i)$ where $p_i(x_i)$ is the polynomial consisting of x_i, only.
$\leadsto [f(s_1, \ldots, s_m)] = \sum_{j=1}^{m} p_j(s_j)$
Note that $t_k \succeq_{emb} s_k$ which implies that $p_i(t_k) \succeq_{POL} p_i(s_k)$.
$\leadsto$ The degree of the variables of $\mathcal{V}ar(s_k)$ belonging to $[r]$ is greater than or equal to that belonging to $[l]$ because the degree of the variables of $\mathcal{V}ar(s_k)$ belonging to $[f(t_1, \ldots, t_m)]$ is equal to that of $p_i(t_k)$ since $[f]$ is separate.
Note that there is any variable x of $\mathcal{V}ar(s_k)$ occurring in r outside of t_k (by precondition).

$\leadsto$ The degree of x in $[r]$ is greater than that of x in $[l]$

$\leadsto l \not\succ_{POL} r$

ii) Analogous to i) of lemma 5.1.

iii) Let [.] be any interpretation including $[i](x)$ to be proper separate (see lemma 5.1). Assume that $[f](x, y)$ is separate.
$\leadsto$ The monomials corresponding to x must be 'greater' than those corresponding to y if the rule $f(f(x, y), z) \to f(x, f(y, z))$ has to be directed.
$\leadsto\quad l \not\succ_{POL} r$
since the variable x (y) is the first (second) argument in l whereas it is the second (first) argument in r

The case $l \to r$ and $f(x, f(y, z)) \to f(f(x, y), z)$ can be proved in the same way. □

Lemma A.3 *Let* $n \geq 1$, $a_i \geq 1$. *Then,*

$$\begin{array}{lrcll} i) & \prod_{i=1}^{n} a_i & \geq & \sum_{i=1}^{n} a_i & \quad if\ \ (\forall i \in [1, n]) \quad a_i \geq 2 \\ ii) & n \cdot \prod_{i=1}^{n} a_i & \geq & \sum_{i=1}^{n} a_i & \end{array}$$

Proof:

i) We will prove this fact by means of contradiction. Assume that
$a_1 \cdot \ldots \cdot a_n < a_1 + \ldots + a_n$

$\leadsto\quad 1 < \frac{a_1}{a_1 \cdot \ldots \cdot a_n} + \ldots + \frac{a_n}{a_1 \cdot \ldots \cdot a_n}$

[1)] f is associative if either $f(f(x,y),z) \to f(x,f(y,z))$ or $f(x,f(y,z)) \to f(f(x,y),z)$ is contained in the TRS.

$\leadsto \quad 1 < \frac{1}{a_2 \cdot \ldots \cdot a_n} + \ldots + \frac{1}{a_1 \cdot \ldots \cdot a_{n-1}}$

Note that the right-hand side of this inequality is not greater than

$$\frac{1}{(n-1) \cdot 2} \cdot n$$

since $a_i \geq 2$

$\leadsto \quad 1 < \frac{n}{(n-1) \cdot 2}$

$\leadsto$ contradiction

because $\frac{n}{(n-1) \cdot 2} \leq 1 \quad \text{for} \quad n > 1$

The case $n = 1$ is obvious since $a_1 \geq a_1$

ii) We will prove this lemma by induction on n.

- The base case ($n = 1$) is obvious.
- $n \to n + 1$:

$$\begin{array}{rcl} (n+1)a_1 \cdot \ldots \cdot a_n \cdot a_{n+1} & = & n \cdot a_1 \cdot \ldots \cdot a_n \cdot a_{n+1} \; + \; a_1 \cdot \ldots \cdot a_n \cdot a_{n+1} \\ & \geq & (a_1 + \ldots + a_n) \cdot a_{n+1} \; + \; a_1 \cdot \ldots \cdot a_n \cdot a_{n+1} \\ & & \text{by using the induction hypothesis} \\ & = & a_1 \cdot a_{n+1} \; + \ldots + \; a_n \cdot a_{n+1} \; + \; a_1 \cdot \ldots \cdot a_n \cdot a_{n+1} \\ & \geq & a_1 + \ldots + a_n + a_{n+1} \\ & & \text{since } a_i \cdot a_{n+1} \geq a_i \;\; \wedge \;\; a_1 \cdot \ldots \cdot a_n \cdot a_{n+1} \geq a_{n+1} \\ & & (\text{because } a_i \geq 1) \end{array}$$

□

Lemma A.4 *Let $a_i \geq 1$ and $b_i \geq 1$. Then,*

$$\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{m} b_i \quad \textit{if} \quad i) \quad n^n \prod_{i=1}^{n} a_i \geq \prod_{i=1}^{m} b_i^n \;\; \wedge \;\; (\forall i \in [1,m])\, b_i \geq 2$$
$$\textit{or} \quad ii) \quad n^n \prod_{i=1}^{n} a_i \geq m^n \prod_{i=1}^{m} b_i^n$$

Proof: This lemma describes two estimations of the inequality $\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{m} b_i$, i.e. we found a and b such that $\sum_{i=1}^{n} a_i \geq a \geq b \geq \sum_{i=1}^{m} b_i$. It is obvious that $a \geq b$ implies $\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{m} b_i$.

i) $\sum_{i=1}^{n} a_i \overset{(*)}{\geq} n \cdot (\prod_{i=1}^{n} a_i)^{\frac{1}{n}} \geq \prod_{i=1}^{m} b_i \overset{(**)}{\geq} \sum_{i=1}^{m} b_i$

with (*) = arithmetic-mean-geometric-mean inequality and (**) = lemma 6.1 i)

$\leadsto n^n \cdot \prod_{i=1}^{n} a_i \geq \prod_{i=1}^{m} b_i^n$

Note that it must be guaranteed that $(\forall i \in [1,m])\, b_i \geq 2$ because it is a precondition for the use of (**).

ii) $\sum_{i=1}^{n} a_i \overset{(*)}{\geq} n \cdot (\prod_{i=1}^{n} a_i)^{\frac{1}{n}} \geq m \cdot \prod_{i=1}^{m} b_i \overset{(**)}{\geq} \sum_{i=1}^{m} b_i$

with (*) = arithmetic-mean-geometric-mean inequality and (**) = lemma 6.1 ii)

$\leadsto n^n \cdot \prod_{i=1}^{n} a_i \geq m^n \cdot \prod_{i=1}^{m} b_i^n$ □

Theorem A.1 *Algorithm 6.1 always terminates. If it does not fail, $\mathcal{R}$ can be directed by a polynomial ordering (based on simple-mixed interpretations) generated by the algorithm.*

Proof: It is easy to see that the transformation process of each part (1. - 6.) of algorithm 6.1 terminates. Therefore, algorithm 6.1 terminates.

The correctness of the algorithm is mainly guaranteed by lemma 6.1, lemma 6.2 and the correctness of the Simplex method. Furthermore, it is based on the following observations (of which the proofs are obvious):

- $l \succ_{semb} r \rightsquigarrow l >_{POL} r$ independent of $[.]$
- Setting superfluous interpretations to minimal values is correct (see operations ψ and ψ_I).
- The operations $\mathcal{S}plit$ and $\mathcal{S}plit^E$ are correct (see also [St91a]).
- The arithmetic laws ($\mathcal{AL}$) are correct. □

Theorem A.2 *$\succ_{IPOL}$ is a simplification ordering on $\mathcal{G}(\mathcal{F})$.*

Proof: For being a simplification ordering on $\mathcal{G}(\mathcal{F})$ we have to prove the irreflexivity, the transitivity, the subterm property, the monotony and the stability w.r.t. substitutions.

i) $\succ_{IPOL}$ is irreflexive:
We have to show that $(\forall t \in \mathcal{T}(\mathcal{F}, \mathcal{X}))\ t \not\succ_{IPOL} t$ which is true since $\succ_{POL}$ is irreflexive.

ii) $\succ_{IPOL}$ is transitive:
We must prove that $r \succ_{IPOL} s \succ_{IPOL} t \rightsquigarrow r \succ_{IPOL} t$.
Proving this assertion by induction on the size of r, we have to consider three disjoint cases.

- $[r] \sqsupset [s]\ \vee\ [s] \sqsupset [t]$
  $\rightsquigarrow\ [r] \sqsupset [t]$
  since $\sqsupset$ is transitive and the fact that $[s] = [r] \sqsupset [t]$
  $\vee\ [s] \sqsupset [r] = [t]$ implies $[s] \sqsupset [t]$
- $[r] = [s] = [t]\ \wedge\ (\mathcal{H}ead(r) \succ \mathcal{H}ead(s)\ \vee\ \mathcal{H}ead(s) \succ \mathcal{H}ead(t))$
  $\rightsquigarrow\ [r] = [s] = [t]\ \wedge\ \mathcal{H}ead(r) \succ \mathcal{H}ead(t)$
  because $\succ$ is a partial ordering
- $[r] = [s] = [t]\ \wedge\ \mathcal{H}ead(r) = \mathcal{H}ead(s) = \mathcal{H}ead(t)$
  Let $r = f(r_1, \ldots, r_n), s = f(s_1, \ldots, s_n), t = f(t_1, \ldots, t_n)$.

  α) $\tau(\mathcal{H}ead(t)) = mul$
  $\rightsquigarrow \{r_1, \ldots, r_n\} \succ_{IPOL}^{mul} \{s_1, \ldots, s_n\} \succ_{IPOL}^{mul} \{t_1, \ldots, t_n\}$
  $\rightsquigarrow (\forall j)(\exists i)\ s_i \succeq_{IPOL} t_j\ \wedge\ (\forall k)(\exists m) r_m \succeq_{IPOL} s_k$

  · $r_m \succ_{IPOL} s_{k(i)} \succ_{IPOL} t_j$
  $\rightsquigarrow\ r_m >_{IPOL} t_j$
  by induction hypothesis

  · $r_m \succ_{IPOL} s_{k(i)} =_{IPOL} t_j\ \vee\ r_m =_{IPOL} s_{k(i)} \succ_{IPOL} t_j$
  $\rightsquigarrow\ r_m \succ_{IPOL} t_j$
  since $s =_{IPOL} r \succ_{IPOL} t\ \vee\ s \succ_{IPOL} r =_{IPOL} t$ implies $s \succ_{IPOL} t$

  · $r_m =_{IPOL} s_{k(i)} =_{IPOL} t_j$
  $\rightsquigarrow r_m$ will be removed from all multisets (by definition of $\succ^{mul}$)

  β) $\tau(\mathcal{H}ead(t)) = left$
  $\rightsquigarrow (\exists i, j)(\forall k < i)(\forall m < j)\ r_k =_{IPOL} s_k \wedge s_m =_{IPOL} t_m \wedge r_i \succ_{IPOL} s_i \wedge s_j \succ_{IPOL} t_j$

$\cdot\ i \neq j$

$\leadsto\quad r_i \succ_{IPOL} s_i =_{IPOL} t_i \quad (\text{if } i < j)$
$\quad\ \vee\ r_j =_{IPOL} s_j \succ_{IPOL} t_j \quad (\text{if } i > j)$

$\leadsto\quad r_i \succ_{IPOL} t_i \ \vee\ r_j \succ_{IPOL} t_j$
because $s =_{IPOL} r \succ_{IPOL} t \ \vee\ s \succ_{IPOL} r =_{IPOL} t$ implies $s \succ_{IPOL} t$

$\leadsto\quad r \succ_{IPOL} t$
since all predecessors are equivalent w.r.t. $=_{IPOL}$

$\cdot\ i = j$

$\leadsto\quad r_i \succ_{IPOL} s_i \succ_{IPOL} t_i$

$\leadsto\quad r_i \succ_{IPOL} t_i$
by induction hypthesis

$\leadsto\quad r \succ_{IPOL} t$
analogous to the case $i \neq j$

γ) $\tau(\mathcal{H}ead(t)) = right$:
analogous to β)

iii) $\succ_{IPOL}$ has the subterm property:
We have to show that $\Lambda \neq u \in \mathcal{P}os(t) \leadsto t \succ_{IPOL} t|_u$.
This is valid, since the condition referring to unary operators is required and $\mu \geq 2$.

iv) $\succ_{IPOL}$ is monotonic, i.e. $t[r]_u \succ_{IPOL} t[s]_u$ if $r \succ_{IPOL} s$:

i) $[r] \sqsupset [s]$
$\leadsto\quad t[r]_u >_{IPOL} t[s]_u$
since $\sqsupset$ is monotonic

ii) $[r] = [s]$
$\leadsto\quad t[r]_u = t[s]_u$
$\leadsto\quad$ the $t|_u$'s must be compared because the term structures are identical, otherwise
$\leadsto\quad t[r]_u \succ_{IPOL} t[s]_u$
because $r \succ_{IPOL} s$

v) $\succ_{IPOL}$ is stable w.r.t. substitutions:
We must show that $(\forall \sigma, s, t) s \succ_{IPOL} t \leadsto s\sigma \succ_{IPOL} t\sigma$. We will prove this fact by induction on $\mid t \mid$:

i) $[s] \sqsupset [t]$
$\leadsto\quad s\sigma \succ_{IPOL} t\sigma$
since $\sqsupset$ is stable w.r.t. substitutions

ii) $[s] = [t] \wedge \mathcal{H}ead(s) \succ \mathcal{H}ead(t)$
$\leadsto\quad s\sigma = t\sigma$
because $[s] = [t] \leadsto [s\sigma] = [t\sigma]$
$\leadsto\quad s\sigma \succ_{IPOL} t\sigma$
because $(\forall t \in \mathcal{T}(\mathcal{F}, \mathcal{X}))\mathcal{H}ead(t) = \mathcal{H}ead(t\sigma)$

iii) $[s] = [t] \wedge \mathcal{H}ead(s) = \mathcal{H}ead(t)$
Let $s = f(s_1, \ldots, s_n), t = f(t_1, \ldots, t_n)$.

α) $\tau(\mathcal{H}ead(t)) = mul$

$\leadsto$ $\{s_1, \ldots, s_n\} \succ_{IPOL}^{mul} \{t_1, \ldots, t_n\}$
by definition of $\succ_{IPOL}$

$\leadsto$ $(\forall j)(\exists i) s_i \succeq_{IPOL} t_j$
by definition of $\succ^{mul}$

$\leadsto$ $(\forall j)(\exists i) s_i\sigma \succeq_{IPOL} t_j\sigma$
by induction hypothesis and the fact that $s =_{IPOL} t \leadsto s\sigma =_{IPOL} t\sigma$

$\leadsto$ $\{s_1\sigma, \ldots, s_n\sigma\} \succ_{IPOL}^{mul} \{t_1\sigma, \ldots, t_n\sigma\}$
by definition of $\succ_{IPOL}$

$\leadsto$ $s\sigma \succ_{IPOL} t\sigma$

β) $\tau(\mathcal{H}ead(t)) = left$

$\leadsto$ $(\exists i)(\forall j < i) s_j =_{IPOL} t_j \wedge s_i \succ_{IPOL} t_i$

$\leadsto$ $(\exists i)(\forall j < i) s_j\sigma =_{IPOL} t_j\sigma \wedge s_i\sigma \succ_{IPOL} t_i\sigma$
by induction hypothesis and the fact that $s =_{IPOL} t \leadsto s\sigma =_{IPOL} t\sigma$

$\leadsto$ $s\sigma \succ_{IPOL} t\sigma$
by definition of $\succ_{IPOL}$

γ) $\tau(\mathcal{H}ead(t)) = right$:
analogous to β) $\square$

Appendix B

Exemplary Sessions

The first session concerns the generation of a polynomial ordering for the TRS given in example 6.1. It is performed by using the operation $\mathcal{S}plit^E$.

```
**********************************************
* Generation System for Polynomial Orderings *
*                  MENU                      *
**********************************************
 0 : exit.
 1 : load a new specification.
 2 : set the pathname for the specifications
     ( actual: #P"//node_19669/data/ulrich/system/inout/intern/*.intern" ).
( 0-2 ) : 1

Please enter the filename of the specification : p-226

**********************************************
* Generation System for Polynomial Orderings *
*                  MENU                      *
**********************************************
 0 : exit.
 1 : load a new specification.
 2 : set the pathname for the specifications
     ( actual: #P"//node_19669/data/ulrich/system/inout/intern/*.intern" ).
 3 : show current rule system.
 4 : change the level of trace (actual : 5).
 5 : toggle the mode for splitting the rules (actual : SPLIT-E).
 6 : specify the operators with no constant monomials.
 7 : specify the unary operators with quadratic interpretations.
 8 : specify the lower bound of the variables (actual : 1).
 9 : generate the interpretations.
( 0-9 ) : 7

Do you want a linear interpretation for every, some or none unary operator (E, S or N): s

Answer Y if you want a quadratic interpretation of the given operator.
G          (Y or N): y
E          (Y or N): n
```

```
( 0-9 ) : 6

Do you want to set every, some or none constant monomial to zero (E, S or N): s

Answer Y if you want no constant monomial in the interpretation off the given operator.
G          (Y or N): y
E          (Y or N): n
H          (Y or N): y

( 0-9 ) : 9

**********************************************
* Generation System for Polynomial Orderings *
**********************************************
Parameters :
The trace-level is 5.
The lower bound of interpretations of constant operators is 1.
The function for splitting the polynomials is SPLIT-E.

rule-system :

1: H(X,X,Y) --> G(X)
2: G(C) --> H(C,D,C)
3: E(X) --> F(X,X)
4: F(X,Y) --> X

Setting special coefficients to minimum values : H-000 := 0, E-2 := 0,
G-0 := 0, G-1 := 0

Deleting trivial rules :
4: F(X,Y) --> X

Superfluous operators are : D and F.

The operators are interpreted as follows:
[C]() = C-0
[F](X Y) = Y + X
[G](X) = G-2 X^2
[E](X) = E-1 X + E-0
[H](X Y Z) = H-111 Z Y X + H-011 Z Y + H-101 Z X + H-110 Y X + H-001 Z + H-010 Y + H-100 X
[D]() = 1

Splitting the rules into inequalities :
 1: { H-011 + H-101  >=  0 ,
      H-001  >=  0 ,
      H-010 + H-100  >=  0 ,
      H-111 + H-110  >=  G-2 }
 2: { C-0^2 G-2  >=  C-0^2 H-101 + C-0^2 H-111 + C-0 H-001 + C-0 H-011 +
        C-0 H-100 + C-0 H-110 + H-010 }
 3: { E-0  >=  0 ,
      E-1  >=  2 }

Applying arithmetic laws :
```

```
 1: { H-111 + H-110  >=  G-2 }
 2: { C-0^2 G-2  >=  C-0^2 H-101 + C-0^2 H-111 + C-0 H-001 + C-0 H-011 +
       C-0 H-100 + C-0 H-110 + H-010 }
 3: { E-1  >=  2 }

Setting superfluous coefficients to minimum value : E-0 := 0, H-001 := 0,
H-010 := 0, H-100 := 0, H-011 := 0, H-101 := 0
 1: { H-110 + H-111  >=  G-2 }
 2: { C-0^2 G-2  >=  C-0^2 H-111 + C-0 H-110 }
 3: { E-1  >=  2 }

Applying arithmetic laws :
 1: { H-110 + H-111  >=  G-2 }
 2: { C-0 G-2  >=  C-0 H-111 + H-110 }
 3: { E-1  >=  2 }

Result of simplification loop :
 1: { H-110 + H-111  >=  G-2 }
 2: { C-0 G-2  >=  C-0 H-111 + H-110 }
 3: { E-1  >=  2 }


Remaining variable coefficients with no values : H-111, H-110, E-1, G-2 and C-0.


Result of eliminating the addition by using lemma 2.
 1: { 2.0 + H-111* + H-110* >= 2 G-2* }
 2: { G-2* + C-0* >= 1.0 + H-111* + H-110* + C-0* }
 3: { E-1* >= 1.0 }


Result of collecting :
C-0* >=   1
U-1* >=   1
- U-1* + H-111* + H-110* - 2 G-2* >= -2.0
U-2* >=   1
- U-2* - H-111* - H-110* + G-2* >= 1.0
U-3* >=   1
- U-3* + E-1* >= 1.0


Result of eliminating the addition by using lemma 1.
 1: { 2.0 + H-111* + H-110* >= 2 G-2*,
      1 + G-2* >= 2 }
 2: { G-2* + C-0* >= H-111* + H-110* + C-0*,
      1 + H-110* >= 2,
      1 + H-111* + C-0* >= 2 }
 3: { E-1* >= 1.0,
      2 >= 2 }


Result of collecting :
C-0* >=   1
U-1* + U-2* >=   1
```

```
- U-1* + H-111* + H-110* - 2 G-2* >= -2.0
- U-2* + G-2* >=   1
U-3* + U-4* + U-5* >=   1
- U-3* - H-111* - H-110* + G-2* >= -0.0
- U-4* + H-110* >=   1
- U-5* + H-111* + C-0* >=   1
U-6* + U-7* >=   1
- U-6* + E-1* >= 1.0
- U-7* >=   0

The simplex algorithm returns : H-111* = 1.0, H-110* = 1.0, E-1* = 2.0,
G-2* = 2.0, C-0* = 1.0.

i.e. : H-111 = 2.0, H-110 = 2.0, E-1 = 4.0, G-2 = 4.0, C-0 = 2.0.

The algorithm terminated successfully.

The operators are interpreted as follows:
[C]() = 2.0
[F](X Y) = Y + X
[G](X) = 4.0 X^2
[E](X) = 4.0 X
[H](X Y Z) = 2.0 Z Y X + 2.0 Y X
[D]() = 1

The interpreted rules are:
[H(X,X,Y)] = 2.0 Y X^2 + 2.0 X^2              >              4.0 X^2  = [G(X)]
[G(C)] = 16.0                       >                          12.0  = [H(C,D,C)]
[E(X)] = 4.0 X                        >                          2 X  = [F(X,X)]
[F(X,Y)] = Y + X                          >                           X  = [X]
( 0-9 ) :
```

The second session considers the same rule system as the first one. However, instead of using the operation $\mathcal{S}plit^E$ it applies $\mathcal{S}plit$.

```
**********************************************
* Generation System for Polynomial Orderings *
*                  MENU                      *
**********************************************
 0 : exit.
 1 : load a new specification.
 2 : set the pathname for the specifications
     ( actual: #P"//node_19669/data/ulrich/system/inout/intern/*.intern" ).
 3 : show current rule system.
 4 : change the level of trace (actual : 5).
 5 : toggle the mode for splitting the rules (actual : SPLIT-E).
 6 : specify the operators with no constant monomials
     (actual: H G).
 7 : specify the unary operators with quadratic interpretations
     (actual: G).
 8 : specify the lower bound of the variables (actual : 1).
 9 : generate the interpretations.
( 0-9 ) : 5

The function for splitting the polynomials is now SPLIT.
( 0-9 ) : 9

**********************************************
* Generation System for Polynomial Orderings *
**********************************************
Parameters :
The trace-level is 5.
The lower bound of interpretations of constant operators is 1.
The function for splitting the polynomials is SPLIT.

rule-system :

1: H(X,X,Y) --> G(X)
2: G(C) --> H(C,D,C)
3: E(X) --> F(X,X)
4: F(X,Y) --> X

Setting special coefficients to minimum values : H-000 := 0, E-2 := 0,
G-0 := 0, G-1 := 0

Deleting trivial rules :
4: F(X,Y) --> X

Superfluous operators are : D and F.

The operators are interpreted as follows:
[C]() = C-0
[F](X Y) = Y + X
[G](X) = G-2 X^2
```

```
[E](X) = E-1 X + E-0
[H](X Y Z) = H-111 Z Y X + H-011 Z Y + H-101 Z X + H-110 Y X + H-001 Z + H-010 Y + H-100 X
[D]() = 1

Splitting the rules into inequalities :
 1: { H-111  >=  0 ,
      H-011 + H-101  >=  0 ,
      H-001  >=  0 ,
      H-110  >=  G-2 ,
      H-010 + H-100  >=  0 }
 2: { C-0^2 G-2  >=  C-0^2 H-101 + C-0^2 H-111 + C-0 H-001 + C-0 H-011 +
        C-0 H-100 + C-0 H-110 + H-010 }
 3: { E-1  >=  2 ,
      E-0  >=  0 }

Applying arithmetic laws :
 1: { H-110  >=  G-2 }
 2: { C-0^2 G-2  >=  C-0^2 H-101 + C-0^2 H-111 + C-0 H-001 + C-0 H-011 +
        C-0 H-100 + C-0 H-110 + H-010 }
 3: { E-1  >=  2 }

Setting superfluous coefficients to minimum value : E-0 := 0, H-001 := 1,
H-010 := 0, H-100 := 0, H-011 := 0, H-101 := 0, H-111 := 0
 1: { H-110  >=  G-2 }
 2: { C-0^2 G-2  >=  C-0 H-110 + C-0 }
 3: { E-1  >=  2 }

Applying arithmetic laws :
 1: { H-110  >=  G-2 }
 2: { C-0 G-2  >=  H-110 + 1 }
 3: { E-1  >=  2 }

Result of simplification loop :
 1: { H-110  >=  G-2 }
 2: { C-0 G-2  >=  H-110 + 1 }
 3: { E-1  >=  2 }


Remaining variable coefficients with no values : H-110, E-1, G-2 and C-0.


Result of eliminating the addition by using lemma 2.
 1: { H-110* >= G-2* }
 2: { G-2* + C-0* >= 1.0 + H-110* }
 3: { E-1* >= 1.0 }


Result of collecting :
C-0* >=   1
U-1* >=   1
- U-1* + H-110* - G-2* >= -0.0
U-2* >=   1
- U-2* - H-110* + G-2* + C-0* >= 1.0
U-3* >=   1
```

```
- U-3* + E-1* >= 1.0

The simplex algorithm returns : H-110* = 1.0, E-1* = 2.0, G-2* = 0, C-0* = 3.0.

i.e. : H-110 = 2.0, E-1 = 4.0, G-2 = 1, C-0 = 7.999999999999998.

The algorithm terminated successfully.

The operators are interpreted as follows:
[C]() = 8.0
[F](X Y) = Y + X
[G](X) = X^2
[E](X) = 4.0 X
[H](X Y Z) = 2.0 Y X + Z
[D]() = 1

The interpreted rules are:
[H(X,X,Y)] = 2.0 X^2 + Y                      >                      X^2  = [G(X)]
[G(C)] = 64.0                        >                        24.0  = [H(C,D,C)]
[E(X)] = 4.0 X                         >                         2 X  = [F(X,X)]
[F(X,Y)] = Y + X                           >                           X  = [X]
( 0-9 ) : 0
```