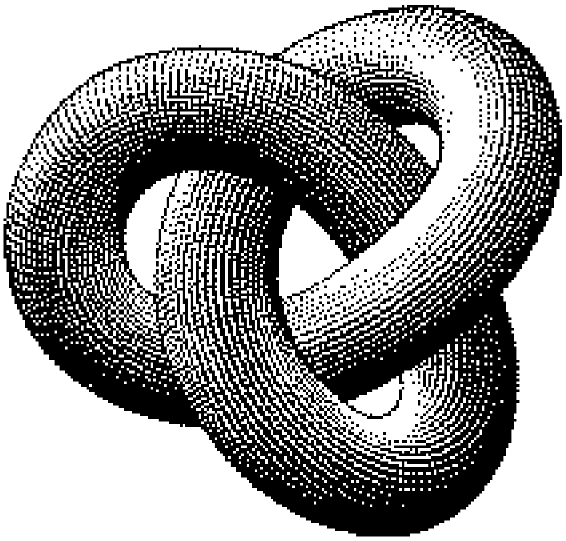


TERMINATION OF REWRITING

—

Extensions, Comparison and Automatic Generation of Simplification Orderings

Vom Fachbereich Informatik der Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation von

Dipl.-Inform. Joachim Steinbach

Berichterstatter:	Prof. Dr. Jürgen Avenhaus
	Prof. Dr. Klaus E. Madlener
Dekan:	Prof. Dr. Otto Mayer

Datum der wissenschaftlichen Aussprache: 28. Januar 1994

D 386

Zusammenfassung Termersetzungssysteme finden immer häufiger Anwendung in der Informatik und der Mathematik. Sie bestehen aus gerichteten Gleichungen, die als nichtdeterministische funktionale Programme verwendet werden. Um effektiv mit ihnen rechnen zu können, müssen sie zumindest terminierend sein.

In dieser Dissertation werden verschiedene Klassen von Simplifikationsordnungen untersucht, die die Terminierung von Termersetzungssystemen nachweisen können. Die Problematik der Anwendbarkeit dieser Terminierungsmethoden in der Praxis wird dabei besonders beachtet. Alle Ordnungen werden mit Hilfe eines neuen Formalismus' transparent dargestellt. Zum einen wird die Mächtigkeit von klassischen Simplifikationsordnungen wie Rekursiven Pfadordnungen, Pfad- und Dekompositionsordnungen, Knuth-Bendix Ordnungen und Polynomordnungen verbessert. Ferner werden für alle diese Ordnungen Einschränkungen angegeben, die kompatibel zu AC-Theorien sind. Dabei werden sowohl bereits existierende Ansätze erweitert als auch neue Methoden entwickelt. Zur automatischen Generierung dieser Ordnungen geben wir auf Heuristiken basierende Algorithmen an. Schließlich werden alle Ordnungen bzgl. ihrer Mächtigkeiten und Zeitkomplexitäten verglichen. Abschließend erfolgt sowohl eine Auswertung einer Statistik über die Terminierung von Beispielen als auch eine kurze Beschreibung einer Softwareumgebung, die die entwickelten Verfahren zur Verfügung stellt.

Abstract More and more, term rewriting systems are applied in computer science as well as in mathematics. They are based on directed equations which may be used as nondeterministic functional programs. Termination is a key property for computing with term rewriting systems.

In this thesis, we deal with different classes of so-called simplification orderings which are able to prove the termination of term rewriting systems. Above all, we focus on the problem of applying these termination methods to examples occurring in practice. We introduce a formalism that allows clear representations of orderings. The power of classical simplification orderings – namely recursive path orderings, path and decomposition orderings, Knuth-Bendix orderings and polynomial orderings – is improved. Further, we restrict these orderings such that they are compatible with underlying AC-theories by extending well-known methods as well as by developing new techniques. For automatically generating all these orderings, heuristic-based algorithms are given. A comparison of these orderings with respect to their powers and their time complexities concludes the theoretical part of this thesis. Finally, not only a detailed statistical evaluation of examples but also a brief introduction into the design of a software tool representing the integration of the specified approaches is given.

Vorwort Zu Beginn möchte ich all denen danken, die mir in vielen Gesprächen über Unfertiges und Unausgegorenes mit Rat und Tat sowie viel Geduld zur Seite standen. Mein besonderer Dank gilt Herrn Prof. Dr. Jürgen Avenhaus und Herrn Prof. Dr. Klaus Madlener, die mich an das Thema Terminierung herangeführt und mir die Gelegenheit zum Erstellen dieser Dissertation gegeben haben. Sie waren stets geduldige Ansprechpartner und trugen in vielen Diskussionen wesentlich zum Gelingen dieser Arbeit bei.

Weiterhin möchte ich mich bei Roland Fettig, Ulrich Kühler, Jochen Nessel, Inger Sonntag und Christof Sprenger bedanken, die große Teile meiner Arbeit sehr intensiv und sorgfältig durchgelesen und wertvolle Verbesserungsvorschläge gemacht haben.

Andrea Sattler-Klein und Bernhard Gramlich danke ich für anregende Diskussionen und zahlreiche Aufmunterungen.

Weiterhin gilt mein Dank sowohl Thomas Deiß, der mich bei der Lösung von LATEX-Problemen tatkräftig unterstützte, als auch Frau Rita Kohl, die mir wesentlich bei der Erstellung von Teilen des Manuskripts geholfen hat.

Robert Eschbach und Christof Sprenger machten es – durch ihre Implementierungen sowie durch die Auswertung von Beispielen – erst möglich, die Statistik in Kapitel 7 zu erstellen.

Durch zahlreiche fruchtbare Gespräche und Hinweise haben viele Kolleginnen und Kollegen zum Entstehen dieser Arbeit beigetragen. Für ihr besonderes Interesse danke ich Mauricio Ayala Rincon, Klaus Becker, Jörg Denzinger, Matthias Fuchs, Carlos Loría Saénz, Birgit Reinert und Claus-Peter Wirth.

Nicht unerwähnt bleiben dürfen auch einige internationale Wissenschaftler des Bereichs Terminierung, die mir u.a. durch ihre konstruktive Kritik behilflich waren. Dazu zählen vor allem Prof. Deepak Kapur (State University of New York, Albany, New York), Prof. Aart Middeldorp (University of Tsukuba, Tsukuba, Japan) und Prof. David Alan Plaisted (University of North Carolina, Chapel Hill, North Carolina).

Besonders herzlich möchte ich meiner Frau Elvira danken, die immer für mich da war. Mit ihrer Liebe, ihrer Geduld und ihrem Verständnis half sie mir über Zeiten des Wartens und Zweifelns hinweg.

Für
Elvira,
Jannik und Rebecca

Contents

Index

List of Figures

Chapter 1

Introduction

> *Nichts ist praktischer als eine gute Theorie.*
>
> Ernst Denert

Term rewriting is a technique that can be applied for reasoning in structures defined by equations. These reasoning abilities are important in mathematics and in many computer applications, including symbolic algebraic computation, automated theorem proving, program specification and verification as well as high-level programming languages. Term rewriting systems (TRSs, for short) contain directed equations (called rules) which may be used as non-deterministic functional programs. These rules are applied by repeatedly replacing subterms of a given formula with equivalent terms until the simplest form possible is obtained. The final result of an inextensible sequence of rule applications is called normal form. TRSs defining exactly one normal form for any input term serve as a procedure for solving the word problem (the equivalence of two terms) for the equational theory given by the rules. *Confluence* (i.e. the result of computing a term by rewriting is unique and therefore independent of the order and choice the rules applied to a term at any time) and *termination* (i.e. all computations are finite) are the key properties for a TRS yielding unique normal forms.

A non-confluent TRS can possibly be transformed into a confluent one which is logically equivalent1) to the first. This can be achieved with the so-called *completion procedure* developed by Knuth and Bendix, which repeatedly adds new rules to the initial TRS ([Knuth and Bendix, 1967])2). Unfortunately, the success of this process crucially depends on the ability to prove the termination of the derived TRS.

In this thesis, we concentrate on *termination*. Thus, features only weakly related to this topic are treated superficially, if at all. For detailed surveys on the theory of term rewriting we refer to [Dershowitz and Jouannaud, 1990], [Avenhaus and Madlener, 1990] and [Huet and Oppen, 1979]. The most essential formal definitions concerning TRSs and relevant in this thesis are listed below3).

---

1) Any equality in the first TRS is an equality in the second one.

2) in that it orients so-called *critical pairs*, which are equations derived from overlapping left-hand sides of already existing rules

3) Unless otherwise specified, we use the notations of [Dershowitz and Jouannaud, 1991].

1.1 Definition (Term Rewriting Systems)

- **Operators:** *Let* $\mathcal{F}$ *be a set of* operators[4], $\mathcal{X}$ *be a set of* variables *with* $\mathcal{F} \cap \mathcal{X} = \emptyset$ *and* $\mathcal{A}r\colon \mathcal{F} \cup \mathcal{X} \mapsto \mathbb{N}$ *be the* arity *of operators and variables, i.e. the number of arguments they are supposed to have*[5]. *Operators of arity zero are called* constants. *A* signature *is a set of operators together with their arities*[6].

- **Terms:** *The set of (first-order)* terms $\mathcal{T}(\mathcal{F},\mathcal{X})$ *over* $\mathcal{F}$ *and* $\mathcal{X}$ *is the smallest set containing* $\mathcal{X}$ *such that* $f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$ *whenever* $f \in \mathcal{F}$, $\mathcal{A}r(f) = n$ *and* $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ *for* $i = 1,\ldots,n$[7]. *An* atom *is either a constant or a variable. Terms in* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *are called* monadic *if* $\forall f \in \mathcal{F}\colon \mathcal{A}r(f) \leq 1$. *Variable-free terms are called* ground terms *and* $\mathcal{T}(\mathcal{F},\emptyset)$ *is denoted by* $\mathcal{G}(\mathcal{F})$. *We assume that there is at least one constant (thus,* $\mathcal{G}(\mathcal{F}) \neq \emptyset$*).*

- **Functions on terms:** *A term* $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ *may be viewed as a finite ordered tree. A* position[8] *within a term may be represented as a sequence of positive integers based on Dewey's decimal notation. The set* $\mathcal{P}os(t)$ *of* positions *of* t *is inductively defined as follows:* $\mathcal{P}os(t) = \{\Lambda\}$ *if* $t \in \mathcal{X}$, *whereas* $\mathcal{P}os(t) = \{\Lambda\} \cup \{i.u \mid 1 \leq i \leq n \wedge u \in \mathcal{P}os(t_i)\}$ *if* $t = f(t_1,\ldots,t_n)$. *If* $u \in \mathcal{P}os(t)$ *then the* subterm $t|_u$ *of* $t = f(t_1,\ldots,t_n)$ *at position* u *is defined by* t *if* $u = \Lambda$ *and by* $t_i|_v$ *if* $u = i.v$. *We write* $s \trianglerighteq t$ *when* t *is a subterm of* s. *The* prefix ordering $>$ *on positions is defined as* $u > v$ *if, and only if,* $u = v.w$ *with* $w \neq \Lambda$. *We write* $t[s]_u$ *(the* context t *with designated position* u*) to denote the term that results from* t *by replacing* $t|_u$ *by* s. *The set* $\mathcal{TP}os(t)$ *of* terminal positions *(i.e. the set of positions of the leaves) of a term* t *is defined by* $\mathcal{TP}os(t) = \{u \mid u \in \mathcal{P}os(t) \wedge \mathcal{A}r(t|_u) = 0\}$. *The* set of variables *of a term* t *is denoted by* $\mathcal{V}ar(t)$. *The* multiset of symbols *(i.e. operators and variables) of a term* t *is denoted by* $\mathcal{S}ym(t)$. *A* substitution σ *is a function from* $\mathcal{X}$ *to* $\mathcal{T}(\mathcal{F},\mathcal{X})$, *extended to a function from* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *to itself satisfying* $f(t_1,\ldots,t_n)\sigma = f(t_1\sigma,\ldots,t_n\sigma)$ *for each* n*-ary operator* f *and for all* $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$, *i.e.* σ *simultaneously replaces all variables of a term by terms. A* ground substitution *replaces all variables by ground terms.*

- **Term rewriting systems:** *A* rule *over a set of terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *is an ordered pair* s, t *of terms, written as* $s \rightarrow t$ *such that* $\mathcal{V}ar(s) \supseteq \mathcal{V}ar(t)$. *Sometimes, a rule will be named by a number* r, *then we write* $r\colon s \rightarrow t$. *A (finite or infinite) set* $\mathcal{R}$ *of rules over* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *is called a* term rewriting system. *For a given TRS* $\mathcal{R}$, $s \in \mathcal{T}(\mathcal{F},\mathcal{X})$ rewrites *to* $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$, *written as* $s \rightarrow_{\mathcal{R}} t$, *if* $s|_u = l\sigma$ *and* $t = s[r\sigma]_u$ *for some* $l \rightarrow r \in \mathcal{R}$, $u \in \mathcal{P}os(s)$ *and* σ. *A* derivation *in* $\mathcal{R}$ *is any (finite or infinite) sequence* $t_0 \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} \cdots$ *of applications of rules in* $\mathcal{R}$. *To indicate the rule* r *which is applied, we write* $\underset{r}{\rightarrow}_{\mathcal{R}}$. *If there is no ambiguity, we use* $\rightarrow$ *instead of* $\rightarrow_{\mathcal{R}}$.

- **Binary relations:** *If* $\rightarrow$ *is a binary relation, then* $\xrightarrow{*}$ *is its* reflexive transitive closure *(*$\rightarrow \circ \cdots \circ \rightarrow$*)*[9] *and* $\xrightarrow{+}$ *is its* transitive closure *(*$\rightarrow \circ \xrightarrow{*}$*). A binary relation is called an* equivalence relation *if, and only if, it is reflexive, symmetric*[10] *and transitive. An*

---

[4] also called *function symbols*

[5] $\mathcal{A}r(x) = 0$ if $x \in \mathcal{X}$. In some cases, operators may possess variable arity (see subsection 1.5.5), i.e. $\mathcal{A}r\colon \mathcal{F} \cup \mathcal{X} \mapsto 2^{\mathbb{N}}$.

[6] Usually, a signature consists of a set of operators, their input sorts and their output sorts. Here, the sorts are redundant since operators are considered which are based on one sort, only.

[7] For simplicity, we sometimes use prefix notation for unary and infix notation for binary operators.

[8] often called *occurrence*

[9] where $\circ$ denotes the *concatenation* of binary relations

[10] A binary relation $\rightarrow$ is *symmetric* if, and only if, $s \rightarrow t$ whenever $t \rightarrow s$ holds.

equivalence relation $\approx$ *on a set* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *of terms describes a* congruence *on* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *if, and only if,* $f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n)$ *whenever* $\forall i \in [1,n]$: $s_i \approx t_i$ *and* $f \in \mathcal{F}$. $\diamond$

We illustrate the technique of term rewriting with an example.

$\boxed{1.2}$ **Example** *Martin Gardner has presented the following problem in the journal 'Scientific American, May 1978':*

> Place six dots in a circle as if to mark the corners of an invisible hexagon and label the dots '*a*' through '*f*'. Regard an isolated dot as a degenerate convex polygon of one corner and two dots joined by a straight line as a degenerate convex polygon of two corners. With a pencil connect the dots in any way to form disjoint convex polygons of one, two, three, four, five or six corners. Disjoint means that no two polygons may have a dot in common. The lines of the same polygon may not cross one another because if they did, the polygon would not be convex; lines of distinct polygons may, however, intersect. If you like, you may draw nothing, so that the pattern will consist of six one-corner polygons. Or you may connect all six dots to make a single hexagon. Or you may produce any mixture of polygons provided they are convex and disjoint. How many different patterns are there?

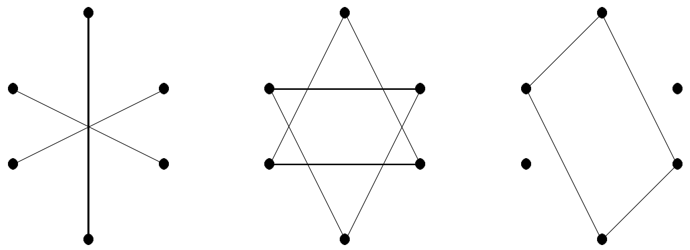


Figure 1.1: Hexagons

Figure 1.1 shows three possible patterns. In more technical terms, the problem is to determine the number of ways a set of six distinct elements can be partitioned into non-empty subsets. You will find there are exactly 203 ways. For two objects O_1, O_2 *the number of ways to partition the set is* 2*:* $\{O_1\}, \{O_2\}$ *and* $\{O_1, O_2\}$. *For three objects* O_1, O_2, O_3 *the number is equal to* 5*:* $\{O_1\}, \{O_2\}, \{O_3\}$ *and* $\{O_1, O_2\}, \{O_3\}$ *and* $\{O_1, O_3\}, \{O_2\}$ *and* $\{O_2, O_3\}, \{O_1\}$ *and* $\{O_1, O_2, O_3\}$. *As the number n of objects increases, the following sequence of numbers is generated:* $1, 2, 5, 15, 52, 203, 877, 4140, \cdots$. *The numbers in this sequence, which are extremely useful in combinatorial theory, are called* Bell numbers[11].

---

[11] in honour of the mathematician Eric Temple Bell. Note that the Bell numbers are used for formalizing e^{e^x} which corresponds to the sequence $e(1 + \frac{\mathbf{1}x}{1!} + \frac{\mathbf{2}x^2}{2!} + \frac{\mathbf{5}x^3}{3!} + \frac{\mathbf{15}x^4}{4!} + \frac{\mathbf{52}x^5}{5!} + \frac{\mathbf{203}x^6}{6!} + \cdots)$. From the Maclaurin expansion it is possible to derive what is called Dobinski's formula for the nth Bell number: $B_n = \frac{1}{e}\sum_{k=0}^{\infty} \frac{k^n}{k!}$.

We are interested in formalizing the sequence of Bell numbers with the help of a TRS. Formulas for the nth Bell number are complicated, but fortunately there exists a simple recursive procedure for computing them. It is best understood by considering the formation of the triangle of numbers (Martin Gardner calls it Bell triangle*) given in figure 1.2. The sequence of Bell numbers appears on two sides of the triangle.*

	0	*1*	*2*	*3*	*4*	*5*	*6*	⋯
0	1							
1	**1**	2						
2	**2**	*3*	5					
3	**5**	*7*	*10*	15				
4	**15**	*20*	*27*	*37*	52			
5	**52**	*67*	*87*	*114*	*151*	203		
6	**203**	*255*	*322*	*409*	*523*	*674*	877	
7	**877**	⋯						
⋮	⋮							

Start with 1 at the top. The last number of each row is the first number of the next row, and all other numbers are obtained by adding the desired number's left neighbour to the number above the neighbour.

Figure 1.2: Bell triangle

In order to use a TRS for representing the above triangle (and thus, the Bell numbers[12])), natural numbers are given as successors of zero, e.g. 5 is represented by $s(s(s(s(s(0)))))$. Following the illustration of the above triangle we generate the rules (the columns of the table given above are represented by the first argument of B whereas the rows are encoded by the second argument)

$$
\begin{array}{rrl}
1: & B(0,0) \rightarrow s(0) & \left.\right\} \; \textit{Initialization} \\
2: & B(0,s(y)) \rightarrow B(y,y) & \left.\right\} \; \begin{array}{l}\textit{The last number of each row is}\\ \textit{the first number of the next row.}\end{array} \\
3: & B(s(x),s(y)) \rightarrow B(x,s(y)) + B(x,y) & \left.\right\} \; \begin{array}{|c|c}\hline a & \\ \hline b & \multicolumn{1}{c|}{c} \\ \hline\end{array} \; \begin{array}{l}\textit{such that}\\ c = a + b\end{array}
\end{array}
$$

We specify the xth Bell number by referring to the first column of the triangle, i.e.

$$4:\; b(x) \rightarrow B(0,x).$$

To complete the specification of the Bell numbers, we need to define the addition '+' on natural numbers (see rule 3):

$$
\begin{array}{rr}
5: & x + 0 \rightarrow x \\
6: & x + s(y) \rightarrow s(x+y)
\end{array}
$$

A computation of the third Bell number (5) with the help of the generated TRS (consisting of the rules 1 to 6) is depicted in figure 1.3[13]). ◇

---

[12]) In the literature, the following formula for the nth Bell number is used (which is more complicated than the one presented here): $b(0) = 1$ and $b(n+1) = \sum_{k=0}^{n} \binom{n}{k} b(k)$.

[13]) We mark the positions where the rules are applied to by underlining the corresponding subterms.

$$\begin{array}{l}
\underline{b(s(s(s(0))))} \xrightarrow[4]{} \underline{B(0,s(s(s(0))))} \xrightarrow[2]{} \underline{B(s(s(0)),s(s(0)))} \xrightarrow[3]{} \underline{B(s(0),s(s(0)))} + \\
B(s(0),s(0)) \xrightarrow[3]{} (\underline{B(0,s(s(0)))} + B(0,s(0))) + B(s(0),s(0)) \xrightarrow[2]{} (\underline{B(s(0),s(0))} + \\
B(0,s(0))) + B(s(0),s(0)) \xrightarrow[3]{} ((\underline{B(0,s(0))} + B(0,0)) + B(0,s(0))) + B(s(0),s(0)) \xrightarrow[2]{} \\
((\underline{B(0,0)} + B(0,0)) + B(0,s(0))) + B(s(0),s(0)) \xrightarrow[1]{} ((s(0) + \underline{B(0,0)}) + B(0,s(0))) + \\
B(s(0),s(0)) \xrightarrow[1]{} ((\underline{s(0) + s(0)}) + B(0,s(0))) + B(s(0),s(0)) \xrightarrow[6]{} (s(\underline{s(0) + 0}) + B(0,s(0))) + \\
B(s(0),s(0)) \xrightarrow[5]{} (s(s(0)) + \underline{B(0,s(0))}) + B(s(0),s(0)) \xrightarrow[2]{} (s(s(0)) + \underline{B(0,0)}) + B(s(0),s(0)) \\
\xrightarrow[1]{} (\underline{s(s(0)) + s(0)}) + B(s(0),s(0)) \xrightarrow[6]{} s(\underline{s(s(0)) + 0}) + B(s(0),s(0)) \xrightarrow[5]{} s(s(s(0))) + \\
\underline{B(s(0),s(0))} \xrightarrow[3]{} s(s(s(0))) + (\underline{B(0,s(0))} + B(0,0)) \xrightarrow[2]{} s(s(s(0))) + (\underline{B(0,0)} + B(0,0)) \xrightarrow[1]{} \\
s(s(s(0))) + (s(0) + \underline{B(0,0)}) \xrightarrow[1]{} s(s(s(0))) + (\underline{s(0) + s(0)}) \xrightarrow[6]{} s(s(s(0))) + s(\underline{s(0) + 0}) \xrightarrow[5]{} \\
\underline{s(s(s(0))) + s(s(0))} \xrightarrow[6]{} s(\underline{s(s(s(0))) + s(0)}) \xrightarrow[6]{} s(s(\underline{s(s(s(0))) + 0})) \xrightarrow[5]{} s(s(s(s(s(0)))))
\end{array}$$

Figure 1.3: Computation of the 3rd Bell number

1.1 Termination of Term Rewriting Systems

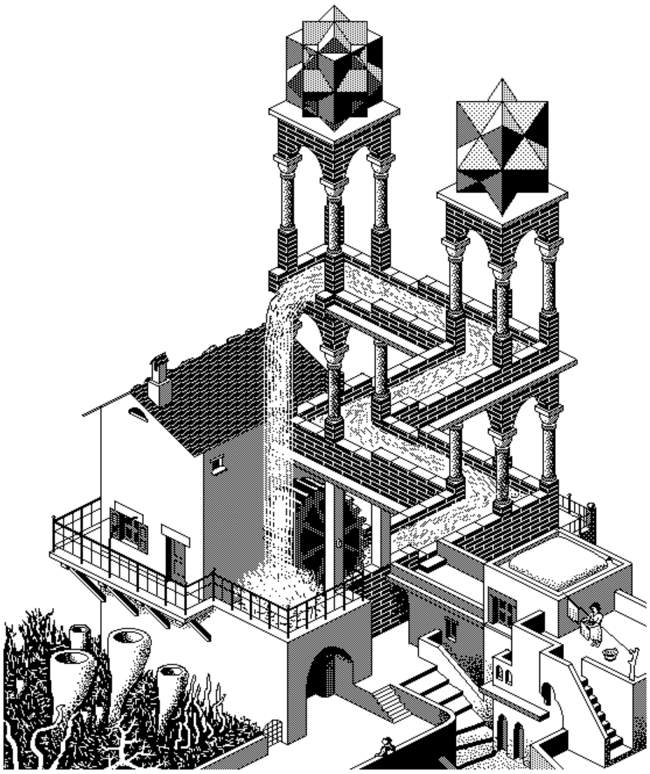
Halting (or termination) is an important, but rather complex, property of algorithms. It is, for example, not obvious that the following succession of integers (see [Sylvester, 1883]) cannot indefinitely be continued:

> Suppose integers to be written down one after the other such that neither one of them is a multiple of any other nor is one of them the sum of a multiple of any other and of a multiple of a specified one.

Furthermore, it is not obvious that the rule (see [Dershowitz, 1987])

$$f(g(x)) \rightarrow g(g(f(f(x))))$$

generates the infinite sequence $\mathbf{f}(\mathbf{f}(\mathbf{g}(\mathbf{x}))) \rightarrow f(g(g(f(f(x))))) \rightarrow g(g(\mathbf{f}(\mathbf{f}(\mathbf{g}(f(f(x))))))) \rightarrow \cdots$.



$\boxed{1.3}$ **Definition (Termination)** *A TRS $\mathcal{R}$ is* terminating *or* Noetherian[14] *for a set $\mathcal{T}(\mathcal{F},\mathcal{X})$ of terms if there exists no infinite (endless) sequence of terms $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ such that $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \cdots$.* ◇

For a finite $\mathcal{R}$, this is the same as saying that the length of derivations starting from any given initial term is finite.

---

[14] after Emily Noether. Terminating systems are also called *finitely* or *strongly* or *uniformly* terminating or *strongly normalizing.*

$\boxed{1.4}$ **Example (Example 1.2 Continued)** *We prove the termination of the TRS given in example 1.2 in an informal manner. First of all, assume the whole TRS to be divided into the parts* $P_1 = \{1, 2, 3\}$*,* $P_2 = \{4\}$ *and* $P_3 = \{5, 6\}$*, where the numbers stand for the corresponding rules. These sets of rules can hierarchically be ordered such that the 'lower' sets call rules from the 'higher' ones:* P_3*,* P_1*,* P_2[15]*. Since there are no indirect recursions between operators of two distinct sets, it is (more or less) sufficient to prove the termination of the rules of each set separately in successive order (from* P_3 *to* P_2*).* **Termination of** P_3**:** *Rule 5 is terminating since the number of symbols decreases. Note that the natural numbers* $\mathbb{N}$ *under the natural ordering* $>$ *are well-founded*[16] *since no sequence of natural numbers can descend beyond* 0*. Rule 6 contains a recursion with respect to (w.r.t., for short) the operator* $+$*. However, since the second argument of the recursion (i.e. of the application of* $+$ *on the right-hand side) is smaller than the corresponding input parameter (i.e. the argument of* $+$ *on the left-hand side) and since the first arguments are identical, rule 6 is terminating.* **Termination of** P_1**:** *The termination of the first rule is obvious since the right-hand side is a ground term where no rule can be applied to. The second rule can be proved to be terminating in analogy with rule 6 (the second argument of the recursive call of operator* B *'decreases'). Rule 3 cannot produce an infinite derivation since* $+$ *(i.e.* P_3*) cannot and the first arguments in the recursion of* B *become smaller.* **Termination of** P_2**:** *Obvious, since* B *is terminating.* ◇

$\boxed{1.5}$ **Theorem ([Huet and Lankford, 1978] and [Dauchet, 1988])** *It is undecidable whether a TRS is terminating, even if it*

- *either consists of one rule*
- *or contains monadic terms,*

only. However, this property is decidable for ground TRSs. ◇

One way of proving termination is to disprove its *non-termination*. It is obvious that a TRS $\mathcal{R}$ is not terminating if the same term repeatedly appears in a derivation: $\cdots \rightarrow_{\mathcal{R}} t_i \rightarrow_{\mathcal{R}} \cdots \rightarrow_{\mathcal{R}} t_k \rightarrow_{\mathcal{R}} \cdots$ with $t_i = t_k$. Furthermore, a TRS $\mathcal{R}$ does not terminate if it is *looping*, which means that there exists a derivation $\cdots \rightarrow_{\mathcal{R}} t_i \rightarrow_{\mathcal{R}} \cdots \rightarrow_{\mathcal{R}} t_k \rightarrow_{\mathcal{R}} \cdots$ with $t_k \trianglerighteq t_i$. But a TRS need not be looping to be non-terminating.

$\boxed{1.6}$ **Example** *The rule*

$$x \star y \;\rightarrow\; (0 + x) \star y$$

is non-looping but has the following infinite derivation, beginning with its left-hand side:

$$\begin{array}{ll} x \star y & \\ \rightarrow & (0 + x) \star y \\ \rightarrow & (0 + (0 + x)) \star y \\ \rightarrow & (0 + (0 + (0 + x))) \star y \\ \rightarrow & \cdots \end{array}$$

◇

Therefore, a notion weaker than looping is needed to characterize non-termination: the *homeomorphic embedding* relation[17] $\succsim_{\mathsf{EMB}}$. It embodies a notion of 'syntactic simplicity': $s \succsim_{\mathsf{EMB}} t$

---

[15] i.e. P_2 needs the definitions of P_1 and the third rule (contained in P_1) needs P_3.

[16] i.e. there exist no infinite descending chains (see definition 1.10)

[17] also called *plunging ordering*

if t can be obtained from s by deleting selected symbols. More formally, the homeomorphic embedding relation can be seen as a means to injectively map the set of symbols of a term s into the set of symbols of a term t, while respecting the topology (i.e. the structure) of s. This mapping is not surjective, in general: If it were surjective, s and t would be identical.

$\boxed{1.7}$ **Definition (Homeomorphic Embedding, Self-Embeddingness)** *Let* $f, g \in \mathcal{F}$, Δ *be an atom and* $s, t, s_i, t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$. *Then*

- *the* homeomorphic embedding relation $\succsim_{\mathsf{EMB}}$ *on* s, t *is defined as follows:*

$$s \succsim_{\mathsf{EMB}} \Delta \quad \Longleftrightarrow \quad \Delta \in \mathcal{S}ym(s)$$

$$s = f(s_1, \ldots, s_m) \succsim_{\mathsf{EMB}} g(t_1, \ldots, t_n) = t \quad \Longleftrightarrow$$
$$\begin{cases} s_i \succsim_{\mathsf{EMB}} t & \text{for some } i \in [1,m] \quad \text{or} \\ f = g \ \wedge \ s_{i_k} \succsim_{\mathsf{EMB}} t_k & \text{for all } k \in [1,n] \ \wedge \ 1 \le i_1 < i_2 < \cdots < i_n \le m \end{cases}$$

$$s \succ_{\mathsf{EMB}} t \quad \Longleftrightarrow \quad s \succsim_{\mathsf{EMB}} t \ \wedge \ s \neq t$$

- *a TRS* $\mathcal{R}$ *is* self-embedding *if it allows a derivation* $\cdots \rightharpoonup_{\mathcal{R}} t_i \rightharpoonup_{\mathcal{R}} \cdots \rightharpoonup_{\mathcal{R}} t_k \rightharpoonup_{\mathcal{R}} \cdots$ *with* $t_k \succsim_{\mathsf{EMB}} t_i$. ◇

Note that the right-hand side of example 1.6 is greater than the corresponding left-hand side w.r.t. $\succ_{\mathsf{EMB}}$.

$\boxed{1.8}$ **Theorem (Tree Theorem, [Kruskal, 1960] and [Higman, 1952])** *Let* $\mathcal{F}$ *be a finite set of operators and* $\mathcal{X}$ *be a finite set of variables. Then any infinite sequence* $t_1, t_2, \cdots$ *of terms in* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *contains two terms* t_i, t_k *with* $i<k$ *and* $t_k \succsim_{\mathsf{EMB}} t_i$. ◇

By definition, a TRS $\mathcal{R}$ is *non-terminating* if there exists at least one infinite derivation $t_1 \rightharpoonup_{\mathcal{R}} t_2 \rightharpoonup_{\mathcal{R}} \cdots$. The following theorem is a consequence of the tree theorem since there can only be a finite number of operators (those of t_1 and of $\mathcal{R}$) appearing in the derivation.

$\boxed{1.9}$ **Theorem ([Kruskal, 1960] and [Higman, 1952])** *If a finite TRS is non-terminating, then it is self-embedding.* ◇

However, self-embeddingness does *not* imply non-termination: $(x^2)^2 \rightarrow (-(x^2))^2$ is both self-embedding and terminating. Note that it is undecidable whether a (finite) TRS is self-embedding ([Plaisted, 1985]). Obviously, self-embeddingness is semi-decidable since one can search through all derivations until an embedding is discovered (see [Plaisted, 1986]). However, this thesis deals with the termination proofs for TRSs based on different ordering strategies. Although termination is undecidable, there exist algorithms allowing a solution of the termination problem in many practical cases. In order to express such termination proofs, we need the following concepts.

$\boxed{1.10}$ Definition (Orderings)

- **Partial Ordering:** *A* partially ordered *set* $(S, \succ)$ *consists of a set* S *and a transitive irreflexive binary relation* $\succ$ *defined on elements of* S.
- **Quasi-Ordering:** *A* quasi-ordered *set* $(S, \succsim)$ *consists of a set* S *and a transitive reflexive binary relation* $\succsim$ *defined on elements of* S. *As usual, a quasi-ordering* $\succsim$ *defines an equivalence relation* $\sim$ *as both* $\succsim$ *and* $\precsim$ *(i.e.* $\succsim \cap \precsim$*), and a partial ordering* $\succ$ *as* $\succsim$ *but not* $\precsim$ *(i.e.* $\succsim \setminus \precsim$*)*[18]. *This requires the property* $a' \sim a \succ b \sim b' \rightsquigarrow a' \succ b'$ *to be fulfilled which is the case for all orderings presented in this thesis.*
- **Total Ordering:** *A quasi-ordered set* $(S, \succsim)$ *(a partially ordered set* $(S, \succ)$*) is said to be* totally ordered *if for any two elements* $a, b \in S$, *either* $a \succsim b$ *or* $b \succsim a$ *(*$a \succeq b$ *or* $b \succeq a$*) holds.*
- **Well-founded Ordering:** *A partially ordered set* $(S, \succ)$ *or a quasi-ordered set* $(S, \succsim)$ *is said to be* well-founded *if there are no infinite descending sequences* $a_1 \succ a_2 \succ \cdots$ *of elements of* S.
- **Extension of an Ordering:** *An* extension *of a partial ordering* $\succ$ *on* S *is a partial ordering* $\succ_E$ *also on* S, *denoted by* $\succ_E \supseteq \succ$, *such that* $a \succ b$ *implies* $a \succ_E b$ *for all* $a, b \in S$. *An* extension *of a quasi-ordering* $\succsim$ *on* S *is a quasi-ordering* $\succsim_E$ *also on* S, *denoted by* $\succsim_E \supseteq \succsim$, *such that* $\succ_E \supseteq \succ$ *and* $a \sim b$ *implies* $a \sim_E b$ *for all* $a, b \in S$. *An extension is* strict *if, and only if, the containment is proper.*
- **Comparability:** *We say that two elements* a *and* b *are* comparable *under an ordering* $\succ$ *(*$\succsim$*, respectively) if, and only if,* $a \succ b$ *(*$a \succsim b$*, respectively) or* $b \succ a$ *(*$b \succsim a$*, respectively). Otherwise, they are* incomparable *and we will write* $a \Bumpeq b$. ◇

An ordered set $(S, \succsim)$ or $(S, \succ)$ is well-founded if, and only if, every non-empty subset of S has (possibly infinitely many) minimal elements. An element x is said to be *minimal* in a subset S' of S if there exists no y in S' such that $x \succ y$. Now, the notion of well-foundedness immediately suggests the pre-reduction ordering method of proving the termination of a TRS $\mathcal{R}$: Show that the rewriting relation $\rightarrow_{\mathcal{R}}$ is included in some well-founded ordering $\succ$ on the set of terms. However, to check the inclusion $\rightarrow_{\mathcal{R}} \subseteq \succ$, all (infinitely many) possible derivations must be tested. The key idea is to restrict this infinite test to a finite one by taking advantage of the structure of terms.

$\boxed{1.11}$ Definition ($\mathcal{T}(\mathcal{F},\mathcal{X})$-Compatibility, Pre-Reduction Ordering)

Let $\succ$ *be a partial ordering on* $\mathcal{T}(\mathcal{F},\mathcal{X})$:

- $\succ$ *is* $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible[19] *if* $s \succ t$ *implies* $f(\cdots, s, \cdots) \succ f(\cdots, t, \cdots)$ *for all terms* s, t *in* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *and all* $f \in \mathcal{F}$.
- $\succ$ *is a* pre-reduction ordering *if it is well-founded and* $\mathcal{T}(\mathcal{F},\mathcal{X})$*-compatible.* ◇

---

[18] In this thesis, 'real' quasi-orderings are used for orderings on operators, only (see subsection 1.5.6). In other respects, we define a quasi-ordering as the union of a partial ordering and an associated congruence (see definition 2.6).

[19] often called *monotonic* or *compatible* w.r.t. the term structure or *replacement* property

The notion of pre-reduction ordering allows the following important characterization of terminating TRSs (going back to Manna & Ness and others).

$\boxed{1.12}$ **Theorem ([Lankford, 1977])** *A TRS $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{X})$ is terminating if, and only if, there exists a pre-reduction ordering $\succ$ on $\mathcal{G}(\mathcal{F})$ such that*

$$l\sigma \succ r\sigma \qquad \textit{for each rule } l \rightarrow r \in \mathcal{R}$$

and for any ground substitution σ for the variables appearing in l. $\diamond$

$\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility of the pre-reduction ordering $\succ$ in the above theorem is necessary since a reduction step $t[l\sigma]_p \rightarrow_{\mathcal{R}} t[r\sigma]_p$ with an application of $l \rightarrow_{\mathcal{R}} r$ must decrease, i.e. $t[l\sigma]_p \succ t[r\sigma]_p$. However, theorem 1.12 only requires $l\sigma \succ r\sigma$. This reveals another dilemma: universal quantification on substitutions. Note that the ordering $\succ$ is defined on ground terms and the theorem requires $l \succ r$ for *all* ground substitutions. Alternatively, the ordering can be required to be *stable* w.r.t. substitutions (*σ-stable*).

$\boxed{1.13}$ **Definition (σ-Stability, Reduction Ordering)** *Let $\rightarrow$ be a binary relation on the set $\mathcal{T}(\mathcal{F},\mathcal{X})$ of terms and $s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})$:*

- ▶ *$\rightarrow$ is* σ-stable *if $s \rightarrow t$ implies $s\sigma \rightarrow t\sigma$, for all ground substitutions σ.*
- ▶ *$\succ$ is a* reduction ordering *if it is a σ-stable pre-reduction ordering on $\mathcal{T}(\mathcal{F},\mathcal{X})$.* $\diamond$

Permitting variables in definitions of reduction orderings, we regard them as additional constants distinct from all other operators in $\mathcal{F}$. Using a reduction ordering, we have the following corollary of theorem 1.12.

$\boxed{1.14}$ **Corollary** *A TRS $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{X})$ is terminating if, and only if, there exists a reduction ordering $\succ$ such that*

$$l \succ r \qquad \textit{for each rule } l \rightarrow r \in \mathcal{R}.$$ $\diamond$

This corollary motivates the following terminology.

$\boxed{1.15}$ **Notation (Ordered w.r.t. an Ordering)** *A TRS $\mathcal{R}$ is* ordered *w.r.t. an ordering $\succ$ if, and only if, $l \succ r$ for each rule $l \rightarrow r \in \mathcal{R}$.* $\diamond$

Thus, a TRS is terminating if, and only if, it is ordered w.r.t. a reduction ordering.

1.2 Simplification Orderings

As we have seen, a termination proof method for TRSs can be a reduction ordering which is a σ-stable pre-reduction ordering (i.e. well-founded and $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible). Guaranteeing the property of well-foundedness is quite difficult in general. This fact leads to the basic idea of identifying certain classes of (reduction) orderings for which there is no need to prove the well-foundedness. One possible solution is represented by the class of *simplification orderings*. These orderings are closely connected with the homeomorphic embedding relation. Thus, the following lemma is a direct implication of theorem 1.9 (stating that a finite TRS is self-embedding if it is non-terminating).

$\boxed{1.16}$ **Lemma ([Kruskal, 1960] and [Higman, 1952])** *Let $\mathcal{F}$ be a finite set of fixed-arity operators. A partial ordering $\succ$ on $\mathcal{T}(\mathcal{F},\mathcal{X})$ is well-founded if it extends the homeomorphic embedding relation (i.e. if $\succ \supseteq \succ_{\mathsf{EMB}}$ holds).* ◇

For applying a partial ordering to TRSs it has to be $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible, at least. The following definition describes $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible extensions of the homeomorphic embedding relation.

$\boxed{1.17}$ **Definition (Simplification Ordering, [Higman, 1952], [Dershowitz, 1979b])** *A $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible partial ordering $\succ$ is a* simplification ordering *for a set $\mathcal{T}(\mathcal{F},\mathcal{X})$ of terms if it possesses the following properties:*

- ▶ *the* subterm property: $s \succ t \quad \textit{if} \quad s \rhd t$
- ▶ *the* deletion property: $f(\cdots, s, \cdots) \succ f(\cdots, \cdots)$

where $s, t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and $f \in \mathcal{F}$. ◇

The subterm property expresses that any term is greater than any of its proper subterms. The deletion condition asserts that deleting subterms of an operator with variable arity reduces the term w.r.t. the ordering. For operators with fixed arity, this property is superfluous[20].

☞ In this thesis[21], each operator is assumed to possess fixed arity with the only exception of associative operators

(see subsection 1.5.5). However, most of the results given also hold for operators with variable arity.

$\boxed{1.18}$ **Lemma (Embedding Lemma, [Dershowitz, 1979b])** *Let $\succ$ be a simplification ordering. Then*

$$\succ \supseteq \succ_{\mathsf{EMB}}.$$ ◇

From this follows that any simplification ordering extends the homeomorphic embedding relation and is well-founded due to lemma 1.16, provided $\mathcal{F}$ is finite. Therefore, the main advantage of introducing simplification orderings lies in the fact that the subterm property (which can easily be tested) provides a sufficient condition for well-foundedness.

$\boxed{1.19}$ **Corollary ([Dershowitz, 1979b])** *Any simplification ordering over a finite $\mathcal{F}$ is a pre-reduction ordering.* ◇

In general, simplification orderings are different from pre-reduction orderings because the former are not necessarily well-founded[22], and the latter do not necessarily have the subterm property. However, Dershowitz ([Dershowitz, 1979b]) proved that, when considering terms over a finite set of operators (such as the terms that comprise a finite TRS) simplification orderings are pre-reduction orderings. Thus,

---

[20] Simplification orderings for fixed-arity operators were investigated in [Higman, 1952]. They are called *divisibility orderings*.

[21] Information which is somewhat important for further reading this thesis, will be marked with ☞.

[22] They will possibly admit infinite derivations if $\mathcal{F}$ is not finite (see the corollary given above).

all simplification orderings referred to in the remaining part of this thesis are based on a *finite* $\mathcal{F}$.

As simplification orderings are a subclass of pre-reduction orderings, theorem 1.12 also holds for simplification orderings. Therefore, any σ-stable simplification ordering is a reduction ordering which implies a general method for termination proofs based on the following theorem.

$\boxed{1.20}$ **Theorem ([Dershowitz, 1979b])** *A TRS over a finite set of operators is terminating, if it is ordered w.r.t. a σ-stable simplification ordering.* ◇

In fact, most reduction orderings used in practice are σ-stable simplification orderings[23].

$\boxed{1.21}$ **Theorem ([Dershowitz, 1979b], [Plaisted, 1978a])** *Any $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible partial ordering which is total on the set $\mathcal{G}(\mathcal{F})$ of ground terms over a finite set $\mathcal{F}$ of fixed-arity operators is well-founded if, and only if, it possesses the subterm property.* ◇

In [Middeldorp and Gramlich, 1993], it is shown that *simple termination* (i.e. the termination which can be proved by means of simplification orderings) of arbitrary TRSs is an undecidable property, even for one-rule systems. It is, however, decidable in polynomial time whether there is a simplification ordering which ensures that a ground TRS terminates ([Plaisted, 1993]).

For more details on simplification orderings, especially their connection to well-quasi-orderings as well as the proof-theoretic relation between Kruskal's theorem and the ordinal Γ_0, see [Gallier, 1991].

1.3 AC-Termination of Term Rewriting Systems

As mentioned in the previous sections, the basic concept of TRSs depends upon the conversion of an equation into a directed rewriting rule by comparing both sides w.r.t. an ordering. Unfortunately, there exist equations which are incomparable in any case. Nevertheless, rewriting techniques can be adapted to handle some of these cases. The simplest example of an equation requiring special treatment is the *commutativity*: Any system containing $x + y \rightarrow y + x$ is non-terminating. Therefore, this axiom can only be used as an equation, and has, consequently, no orientation. If the termination property for a set of axioms is not satisfied, the set can be split into two parts: The axioms causing non-termination are used as equations $\mathcal{E}$ while the others are used as rewriting rules $\mathcal{R}$. Rewriting under those conditions is named *rewriting modulo $\mathcal{E}$*.

$\boxed{1.22}$ **Definition (Equational TRS $\mathcal{R}/\mathcal{E}$, Rewriting Modulo $\mathcal{E}$)** *An* equational TRS $\mathcal{R}/\mathcal{E}$ *is composed of a set $\mathcal{R} = \bigcup\{l_i \rightarrow r_i\}$ of rules and a set $\mathcal{E} = \bigcup\{u_i \leftrightarrow v_i\}$ of equations. An appropriate reduction relation allows reductions modulo the equations in $\mathcal{E}$: A term s* rewrites *to t* modulo $\mathcal{E}$, *denoted by $s \rightarrow_{\mathcal{R}/\mathcal{E}} t$, if*

$$s \overset{*}{\leftrightarrow}_{\mathcal{E}} u[l\sigma]_p \quad \textit{and} \quad u[r\sigma]_p \overset{*}{\leftrightarrow}_{\mathcal{E}} t$$

for some $u \in \mathcal{T}(\mathcal{F},\mathcal{X})$, $p \in \mathcal{P}os(u)$, $l \rightarrow r \in \mathcal{R}$ and σ. ◇

---

[23] Recently, techniques for proving $s \succ t$ if $t \succ_{\mathsf{EMB}} s$ have been developed (see section 7.4).

The effective computation with this relation presumes *$\mathcal{E}$-termination*, i.e. there exists no infinite sequence of terms of the form $t_1 \rightarrow_{\mathcal{R}/\mathcal{E}} t_2 \rightarrow_{\mathcal{R}/\mathcal{E}} \cdots$. As in the classical case, an equational TRS $\mathcal{R}/\mathcal{E}$ terminates if there exists a well-founded ordering $\succ$ which contains the rewriting relation $\xrightarrow{+}_{\mathcal{R}/\mathcal{E}}$. The test for this inclusion requires all derivations of the form $s \rightarrow_{\mathcal{R}/\mathcal{E}} t$ to be checked. This procedure can be simplified using the following property.

$\boxed{1.23}$ **Definition ($\mathcal{E}$-Compatibility)** *An ordering $\succ$ is $\mathcal{E}$-*compatible *if, and only if,*

$$s' \overset{*}{\leftrightarrow}_{\mathcal{E}} s \succ t \overset{*}{\leftrightarrow}_{\mathcal{E}} t' \quad \textit{implies} \quad s' \succ t'. \qquad \diamond$$

$\boxed{1.24}$ **Lemma ([Bachmair and Plaisted, 1985b])** *If $\succ$ is an $\mathcal{E}$-compatible simplification ordering, then*

$$\succ \supseteq \xrightarrow{+}_{\mathcal{R}/\mathcal{E}} \quad \textit{if, and only if,} \quad \succ \supseteq \xrightarrow{+}_{\mathcal{R}}. \qquad \diamond$$

$\boxed{1.25}$ **Theorem ([Bachmair and Plaisted, 1985b])** *An equational TRS $\mathcal{R}/\mathcal{E}$ is terminating, if $\mathcal{R}$ is ordered w.r.t. an $\mathcal{E}$-compatible σ-stable simplification ordering.* ◇

Jouannaud and Muñoz succeeded in weakening the $\mathcal{E}$-compatibility for this theorem (see [Jouannaud and Muñoz, 1984]) by introducing a property called *$\mathcal{E}$-commutation.*

$\boxed{1.26}$ **Definition ($\mathcal{E}$-Commutation)** *An ordering $\succ$ is $\mathcal{E}$-*commuting *if, and only if,*

$$s' \overset{*}{\leftrightarrow}_{\mathcal{E}} s \succ t \quad \textit{implies} \quad \exists t' \colon s' \succ t' \overset{*}{\leftrightarrow}_{\mathcal{E}} t. \qquad \diamond$$

With the help of $\mathcal{E}$-commutation, Jouannaud and Muñoz reduced the $\mathcal{E}$-termination problem to the ordinary termination problem. This allows the application of well-known techniques based on simplification orderings. $\mathcal{E}$-termination and termination of a rewriting relation are identical if the used rewriting relation is $\mathcal{E}$-commuting. If $\mathcal{E}$-commutation does not hold, it can be achieved by systematically adding rules which are generated by orienting critical pairs between rules and equations. The following theorem reveals the main importance of $\mathcal{E}$-commutation for the $\mathcal{E}$-termination problem.

$\boxed{1.27}$ **Theorem ([Jouannaud and Muñoz, 1984])** *An equational TRS $\mathcal{R}/\mathcal{E}$ is terminating, if $\mathcal{R}$ is ordered w.r.t. an $\mathcal{E}$-commuting σ-stable simplification ordering.* ◇

Obviously, $\mathcal{E}$-termination strongly depends on the underlying theory $\mathcal{E}$. For example, $\mathcal{E}$ must satisfy the following two conditions in order to prevent infinite derivations (see [Jouannaud and Muñoz, 1984]):

- If $s \leftrightarrow_{\mathcal{E}} t$, then the set of variables of both terms must be identical to avoid loops. Otherwise, we may instantiate an additional variable with an instance of a left-hand side of a rule of $\mathcal{R}$. By rewriting the term with this rule and by then applying the 'starting equation' twice, the term will be derived from itself.

$\boxed{1.28}$ **Example** *The following systems are given:*

$$\begin{array}{rrcl} \mathcal{R}: & x \star 1 & \to & x \\ \mathcal{E}: & x \star 0 & \leftrightarrow & 0 \end{array}$$

Then the following derivation of $\leftrightarrow_{\mathcal{E}}$*- and* $\rightarrow_{\mathcal{R}}$*-steps will be produced:* $0 \leftrightarrow_{\mathcal{E}} (x \star 1) \star 0 \rightarrow_{\mathcal{R}} x \star 0 \leftrightarrow_{\mathcal{E}} 0 \leftrightarrow_{\mathcal{E}} \cdots$. ◇

- Furthermore, $\mathcal{E}$-termination cannot be satisfied if there is an equation of the form $t \leftrightarrow_{\mathcal{E}} x$ such that x occurs more than once in t. In this case, a left-hand side l of a rule of $\mathcal{R}$ is $\mathcal{E}$-equal to a term with several positions of l. First of all, we instantiate x with l. We can then rewrite the subterm at one of the several positions of l and start the whole process again.

$\boxed{1.29}$ **Example** *The following systems are given:*

$$\begin{array}{rrcl} \mathcal{R}: & \neg\neg x & \to & x \\ \mathcal{E}: & x \wedge x & \leftrightarrow & x \end{array}$$

Then the following derivation of $\leftrightarrow_{\mathcal{E}}$*- and* $\rightarrow_{\mathcal{R}}$*-steps can be produced:* $\neg\neg x \leftrightarrow_{\mathcal{E}} \neg\neg x \wedge \neg\neg x \rightarrow_{\mathcal{R}} x \wedge \neg\neg x \leftrightarrow_{\mathcal{E}} x \wedge (\neg\neg x \wedge \neg\neg x) \rightarrow_{\mathcal{R}} x \wedge (x \wedge \neg\neg x) \leftrightarrow_{\mathcal{E}} \cdots$. ◇

Two important theories satisfying the two conditions above are the following ones: *permutative* congruences[24) and particularly *associative-commutative* (AC, for short) congruences. The first theory was introduced in [Lankford and Ballantyne, 1977a]. Of more practical importance are AC-systems, where $\mathcal{E}$ is an equational TRS consisting of associativity and commutativity axioms for a subset of the binary operators in $\mathcal{F}$. This problem has been examined in [Huet, 1980] for left linear TRSs and for the general case in [Lankford and Ballantyne, 1977b], [Peterson and Stickel, 1981], [Jouannaud and Kirchner, 1986] and [Bachmair and Dershowitz, 1987].

$\boxed{1.30}$ **Definition (AC-Theory)** *Let* $\mathcal{F}$ *be a set of operators,* $f \in \mathcal{F}$ *and* $\mathcal{E}$ *be a set of equations:*

- *A* commutativity axiom *for* f *is an equation of the form* $f(x,y) \leftrightarrow_{\mathcal{E}} f(y,x)$.
- *An* associativity axiom *for* f *is an equation of the form* $f(f(x,y),z) \leftrightarrow_{\mathcal{E}} f(x,f(y,z))$.

The subsets $\mathcal{F}_C$, $\mathcal{F}_A$ *and* $\mathcal{F}_{AC}$ *of* $\mathcal{F}$ *are defined as follows:*

- $f \in \mathcal{F}_C$ *if* $\mathcal{E}$ *contains a commutativity axiom for* f.
- $f \in \mathcal{F}_A$ *if* $\mathcal{E}$ *contains an associativity axiom for* f.
- $f \in \mathcal{F}_{AC}$ *if* $\mathcal{E}$ *contains a commutativity and an associativity axiom for* f.

An equational TRS $\mathcal{R}/\mathcal{E}$ *is an* associative-commutative TRS *if every equation in* $\mathcal{E}$ *is either an associativity or a commutativity axiom.* ◇

[24) *Permutative* congruences are congruences for which each congruence class is finite.

The following example illustrates the problems concerning AC-termination.

$\boxed{1.31}$ **Example ([Porat and Francez, 1986])** *Let* $\mathcal{R}$ *be*

$$\begin{cases} 1: & s(x)+y & \rightarrow & x+s(y) \\ 2: & s(x)+y & \rightarrow & s(x+y) \\ 3: & 0+y & \rightarrow & 0 \\ & + \in \mathcal{F}_{AC} & & \end{cases}$$

Rule 1 is not AC-terminating: $s(0)+0 \rightarrow_{\mathcal{R}} 0+s(0) \leftrightarrow_{AC} s(0)+0$. *However, for a complete definition of* $+$*, it is sufficient to use the second and the last rule, only. This part of the TRS is AC-terminating.* ◇

1.4 Overview and Main Results of this Thesis

This thesis primarily discusses problems related to (practical) applications of simplification orderings in order to provide powerful and user-friendly – i.e. (semi-) automatic – methods of proving termination for large classes of TRSs. Thus, we are dealing with various simplification orderings for classical TRSs as well as for equational TRSs (especially associative-commutative TRSs)[25]. These orderings can be divided into two classes depending on the main principles applied in their construction:

- **Syntactic Orderings** are based on a so-called precedence – satisfying the so-called *variable condition*, i.e. $\forall x,y \in \mathcal{X}$: $x \neq y \hookrightarrow x \not\gtrsim y$ and $\forall x \in \mathcal{X}, \forall f \in \mathcal{F}$: $f \not\gtrsim x$ – which is an ordering on operators, i.e. terms are ordered by a careful analysis of the term structure:

  > A syntactic ordering divides the comparison of two terms s and t into a sequence of comparisons of selected pairs (a_i, b_i) of symbols where $a_i \in \mathcal{S}ym(s)$ and $b_i \in \mathcal{S}ym(t)$. The symbols must be compared w.r.t. a precedence.

- **Semantic Orderings** use a so-called termination function that maps terms of $\mathcal{T}(\mathcal{F},\mathcal{X})$ into a set with an associated well-founded ordering. In [Zantema, 1992a], a particular case of a semantic method based on a monotonic algebra is given.

  First of all, we present a well-known formalism for defining semantic orderings. A termination function is a monotonic morphism from $\mathcal{T}(\mathcal{F},\mathcal{X})$ to a well-founded set.

  $\boxed{1.32}$ **Definition (Termination Function)** *A* termination function ζ *from the set* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *of terms into a partially ordered set* $(S,\succ)$ *is composed of a set of functions* f_ζ: $S^{\mathcal{A}r(f)} \mapsto S$*, one for each operator* $f \in \mathcal{F}$*, and is defined by*

  $$\begin{cases} \zeta(f(t_1,\ldots,t_{\mathcal{A}r(f)})) = f_\zeta(\zeta(t_1),\ldots,\zeta(t_{\mathcal{A}r(f)})) & \forall f(t_1,\ldots,t_{\mathcal{A}r(f)}) \in \mathcal{T}(\mathcal{F},\mathcal{X}) \\ x \succ x' \ \hookrightarrow \ f_\zeta(\cdots x \cdots) \succ f_\zeta(\cdots x' \cdots) & \forall x,x' \in S, \forall f \in \mathcal{F} \end{cases}$$ ◇

  With the help of this definition, we obtain a characterization of terminating TRSs similar to that of theorem 1.12 (going back to Manna & Ness and others).

---

[25] A list of all orderings covered in this thesis is contained in appendix D.

$\boxed{\textbf{1.33}}$ **Theorem ([Lankford, 1975])** *A TRS $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{X})$ is terminating if, and only if, there exists a well-founded set $(S,\succ)$ and a termination function $\zeta\colon \mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto S$ such that*

$$\zeta(l\sigma) \succ \zeta(r\sigma) \quad \textit{for each rule } l \to r \in \mathcal{R}$$

and for any ground substitution σ. ◇

The main contributions of this thesis are

- a new formalism for representing orderings,
- extensions of well-known syntactic and semantic simplification orderings for classical termination proofs as well as for AC-termination proofs,
- a comparison of the strength as well as of the time complexities of these orderings,
- heuristic-based algorithms for (semi-) automatically generating such orderings proving the termination of given TRSs,
- statistics on the practical applicability (based on a large collection of TRSs occurring in the literature) of the orderings and
- the design of a software tool representing the integration of the specified approaches.

Figure 1.4 lists all orderings[26] considered (well-known and new ones). They are divided into syntactic (left-hand side) and semantic (right-hand side) methods. The orderings introduced in this thesis are underlined.

The logical dependencies between the sections of the different chapters of this thesis are depicted in figure 1.5. In the order of the sections presented there we now give a brief description of the contents and the main results of the thesis.

2 A general status function (determining the order according to which the arguments of an operator are compared) for signatures ordered by a quasi-ordering is introduced. Furthermore, we present a formalism for defining orderings.

3.1 Well-known path and decomposition orderings are described. The definitions of the decomposition orderings have been modified in order to clarify the close connection between these two types of orderings. Using these new definitions, the decomposition orderings are easier to understand.

4.1 This section contains the definition of the well-known Knuth-Bendix ordering.

5.1 The original polynomial ordering and the improved polynomial ordering (developed by Manna & Ness and Lankford) are presented.

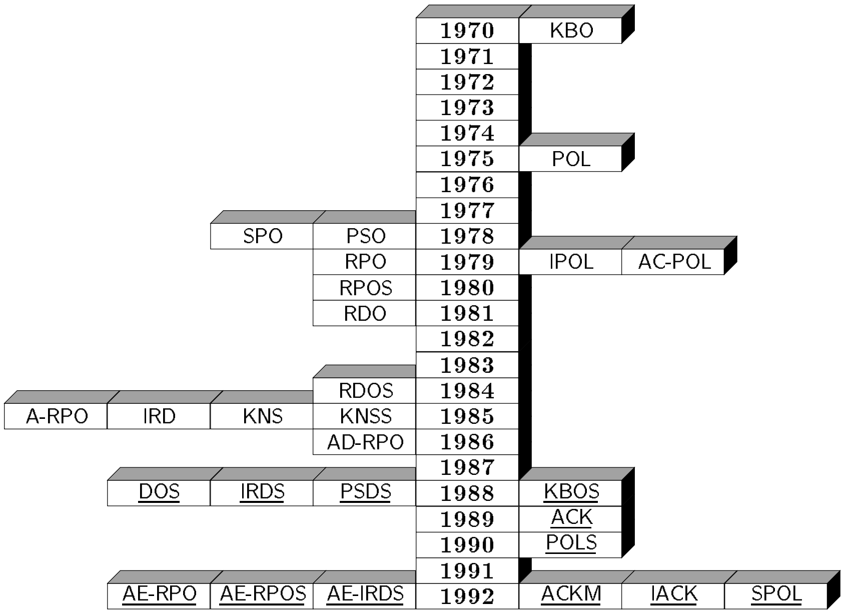
3.2 We introduce three new decomposition orderings. The main novelty lies in the improvement of well-known orderings by incorporating a status function.

4.2 We integrate a status function into the Knuth-Bendix ordering. Moreover, the restriction for the weight function is weakened, especially in the case of unary operators.

5.2 An extension of the improved polynomial ordering is presented by admitting arbitrary lexicographic status as well as multiset status. Another improvement is given which contains the subterm ordering.

---

[26] with the exception of some orderings only briefly mentioned in section 7.4



Name of the Ordering	**Abbrev.**	**Original Source**
Associative-Commutative Knuth-Bendix Ordering	ACK	–
ACK on Memory Flattening	ACKM	–
Associative-Commutative Polynomial Ordering	AC-POL	[Lankford, 1979] [BenCherifa and Lescanne, 1987]
Associative Path Ordering on Distributivity Rules	AD-RPO	[Bachmair and Plaisted, 1985b] [Bachmair and Dershowitz, 1986] [Gnaedig and Lescanne, 1986]
Extended AD-RPO	AE-RPO	–
AE-RPOS based on the IRDS	AE-IRDS	–
AE-RPO with status	AE-RPOS	–
Associative Path Ordering	A-RPO	[Bachmair and Plaisted, 1985b] [Bachmair and Dershowitz, 1986] [Gnaedig and Lescanne, 1986]
Decomposition Ordering with Status	DOS	–
Improved Associative-Commutative Knuth-Bendix Ordering	IACK	–
Improved Polynomial Ordering	IPOL	[Lankford, 1979]
Improved Recursive Decomposition Ordering	IRD	[Rusinowitch, 1987]
Improved Recursive Decomposition Ordering with Status	IRDS	–
Knuth-Bendix Ordering	KBO	[Knuth and Bendix, 1967]
Knuth-Bendix Ordering with Status	KBOS	–
Path Ordering of Kapur, Narendran and Sivakumar	KNS	[Kapur *et al.*, 1985]
Path Ord. with Status of Kapur, Narendran and Sivakumar	KNSS	[Kapur *et al.*, 1985]
Polynomial Ordering	POL	[Manna and Ness, 1970] [Lankford, 1975]
Polynomial Ordering with Status	POLS	–
Path of Subterms Ord. with Status based on Decompositions	PSDS	–
Path of Subterms Ordering	PSO	[Plaisted, 1978a]
Recursive Decomposition Ordering	RDO	[Lescanne, 1981a]
Recursive Decomposition Ordering with Status	RDOS	[Lescanne, 1984]
Recursive Path Ordering	RPO	[Dershowitz, 1979b]
Recursive Path Ordering with Status	RPOS	[Kamin and Lévy, 1980]
Simple Path Ordering	SPO	[Plaisted, 1978b]
Polynomial Ordering with Subterm Property	SPOL	–

Figure 1.4: History of simplification orderings

5.3 We describe some observations on the time complexity of the method of BenCherifa & Lescanne for proving the positiveness of a polynomial. In addition, we provide

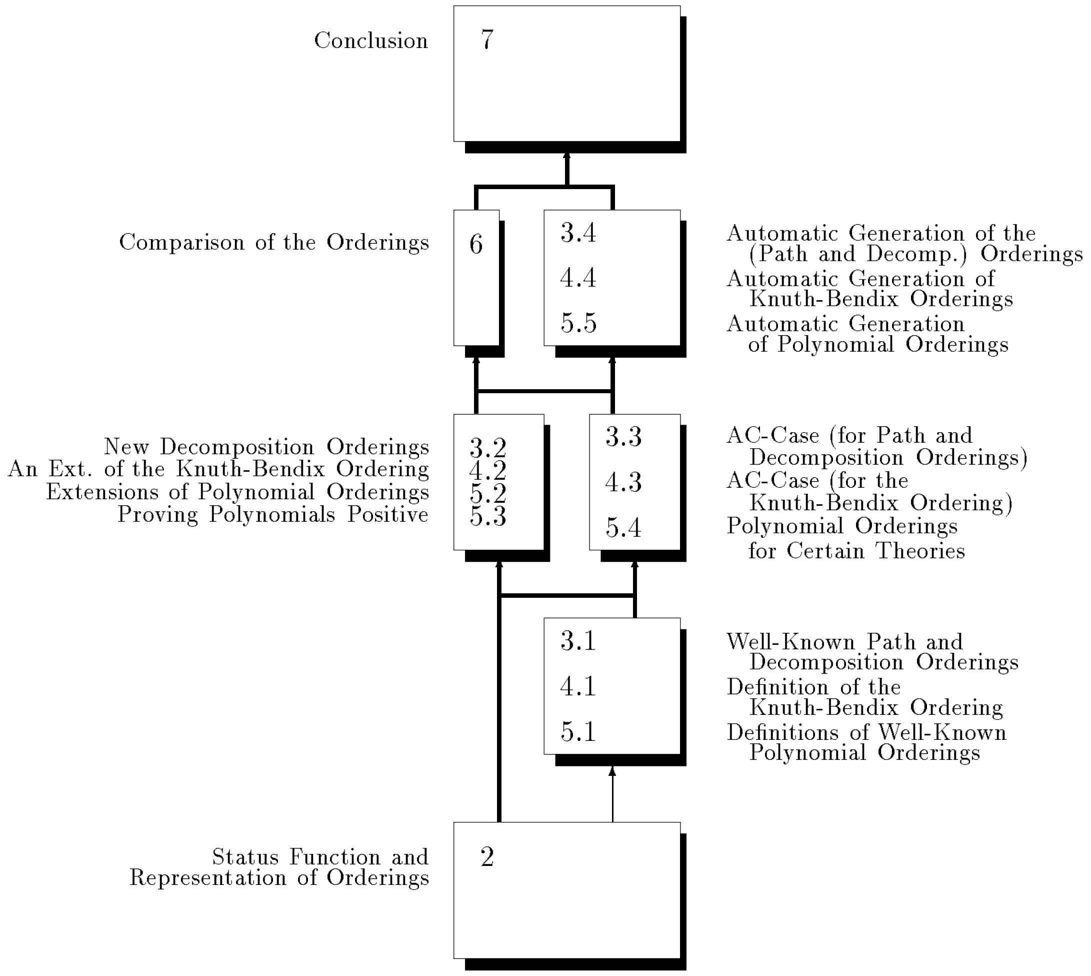


Figure 1.5: Dependence of the sections

an extension of this method which is based on a set of constraints. We also discuss another improvement utilizing the classical arithmetic-mean-geometric-mean inequality. Finally, we deal with a modification for restricted TRSs involving constants in specific positions, only.

3.3 An overview and a classification of the well-known method of establishing AC-termination based on the recursive path ordering is given. Furthermore, we show that this technique (called associative path ordering) cannot use quasi-orderings on AC-operators. Most importantly, an improvement of this ordering is presented which weakens the conditions required for the precedence and which incorporates a status function. Finally, we apply this extended concept to the strongest path and decomposition ordering.

4.3 This section deals with two new orderings applicable to TRSs with underlying AC-theories. Both approaches are based on the Knuth-Bendix ordering. One of them uses the flattening operation, the other one employs a new memory-flattening operation (both operations are defined in subsection 1.5.5) which allow unique representations of AC-equivalent terms.

5.4 In this section, we present conditions for the interpretations of $\mathcal{E}$-operators under which the induced polynomial ordering is $\mathcal{E}$-compatible, where $\mathcal{E}$ contains distributivity or endomorphism axioms, a kind of if-then-else equality and axioms of Boolean rings.

3.4 A summary of the main characteristics of several well-known decision techniques for generating adequate path and decomposition orderings is given.

4.4 There exists a decision procedure for the problem whether or not a Knuth-Bendix ordering can be generated that proves the termination of a given TRS ([Martin, 1987], [Dick *et al.*, 1990]). It is based on the transformation of rewriting rules into linear inequalities that are solved with a classical linear programming method. This section provides a slight extension of this technique which is suitable for the Knuth-Bendix ordering with status (introduced in section 4.2).

5.5 One of the main problems of polynomial orderings lies in the choice of the right interpretations for a given TRS. Here, we present two new heuristic approaches: (i) guidelines for dealing with special classes of TRSs as well as (ii) a sophisticated algorithm for choosing appropriate special polynomials by transforming the rules into linear inequalities and solving them with a classical linear programming technique.

6 This chapter features a comparison of the orderings presented. First of all, the powers of the orderings (i.e. the sets of comparable terms for each combination of two orderings) are compared. Furthermore, the time complexities of the orderings are studied, i.e. the time needed to compare two terms w.r.t. a given ordering is provided.

7 We have tested a significant collection of TRSs occurring in the literature for their termination proofs with simplification orderings. Detailed statistical results about these experiments can be found in the conclusion. Furthermore, a brief description of the TETRES system (an acronym for Termination Expert for Term REwriting Systems) – representing a software tool which integrates the specified approaches – enhances this thesis.

Appendix A consists of a catalogue of selected TRSs including simplification orderings proving their termination. Appendix B contains the proofs of all theorems, lemmata and corollaries introduced in this thesis.

Parts of this thesis have been presented separately in various publications. In sections 3.1 and 3.2 we refer to [Steinbach, 1989a], [Steinbach, 1988b] and [Steinbach, 1988a]. Section 3.3 is an extension of [Steinbach, 1990c] and [Steinbach, 1989b]. Section 4.2 consists of a part of [Steinbach, 1989a] and [Steinbach, 1988b] whereas section 4.3 is an extended version of [Steinbach, 1990a], [Steinbach, 1990b] and [Steinbach, 1989c]. Section 5.3 essentially consists of [Steinbach, 1992b] and [Steinbach, 1991a] whereas sections 5.2 and 5.5 are based on [Steinbach, 1991b], [Steinbach, 1992c] and [Steinbach, 1994]. Chapter 6 represents an improvement of parts of [Steinbach, 1989a], [Steinbach, 1988b], [Steinbach, 1988a] and [Steinbach, 1993a]. Parts of the statistics contained in section 7.3 are published in [Steinbach, 1993b] and [Steinbach, 1991b].

1.5 Notations

In order to define orderings, we need some kind of formalism. Most of the notations needed in the remaining part of this thesis will be collected in this section. This way was chosen in order to clarify connections between similar definitions. The main disadvantage of this kind of representation is the following one: If a reader is interested in some particular information, (s)he has to find out the necessary definitions of this section. For a fast selection of these data, the following table could be helpful:

chapter	**needs notations of subsections**
2	1.5.1, 1.5.6
3	1.5.1, 1.5.2, 1.5.5, 1.5.6
4	1.5.1, 1.5.3, 1.5.5, 1.5.6
5	1.5.1, 1.5.3, 1.5.4, 1.5.6

The major part of this section deals with definitions of operations on terms.

1.5.1 Basic Definitions on Terms and Multisets

$\boxed{1.34}$ **Definition (Simple Operations on Terms)** *Let* $\Delta \in \mathcal{F} \cup \mathcal{X}$ *and* $t = f(t_1,\dots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$:

- **Leading operator:** $\mathcal{H}ead(t) = f$
- **Tuple of arguments:** $\mathcal{A}rgs(t) = (t_1,\dots,t_n)$
- **Size**[27]**:**

$$|t|_\Delta = \begin{cases} \sum_{i=1}^{n} |t_i|_\Delta & \text{if } f \neq \Delta \\ 1 + \sum_{i=1}^{n} |t_i|_\Delta & \text{otherwise} \end{cases}$$

$$|t| = \sum_{\Delta \in \mathcal{F} \cup \mathcal{X}} |t|_\Delta \qquad \diamond$$

Intuitively, a *multiset*[28] M over a set S is an unordered collection of elements of S, with possibly multiple occurrences of elements. Thus, multisets differ from sets by allowing multiple occurrences of identical elements. More formally, a multiset M is a mapping from S to $\mathbb{N}$ associating with each element in S the number of times it occurs in M. It is possible to use an equivalence relation on the elements of S instead of the syntactic identity. In this case, the corresponding equivalence classes will be considered.

To distinguish multisets from sets, each multiset is enclosed in double parentheses: $\{\!\{\dots\}\!\}$[29].

The *union* of multisets is denoted by $\uplus$ whereas $\setminus$ represents the multiset difference.

---

[27] Sometimes, we use $|M|$ to denote the size (i.e. the number of elements) of a (multi-) set M and $|M|_\Delta$ to denote the number of Δ occurring in the (multi-) set M.

[28] also called *bag*

[29] For example, let $S = \{a, b, c\}$. Then $M(a) = 1, M(b) = 2, M(c) = 0$ describe the multiset $M = \{\!\{\, a, b, b \,\}\!\}$.

1.5.2 Paths and Decompositions

A *path* of a term is a sequence of terms starting with the whole term, followed by a path of one of its arguments.

$\boxed{1.35}$ **Definition (Path of a Term)** *Let Δ be an atom, $t = f(t_1,\dots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and let T be a set of terms. Then a* path *of t is defined as*

$$\mathcal{P}ath_u(t) \;=\; \begin{cases} [\Delta] & \text{if } u = \Lambda \text{ and } t = \Delta \\ [t] \cdot \mathcal{P}ath_v(t_i) & \text{if } u = i.v \in \mathcal{TP}os(t) \end{cases}$$

such that $[t_1,\dots,t_{i-1}] \cdot [t_i,\dots,t_n] \;=\; [t_1,\dots,t_n]$ *and*

$$\mathcal{P}ath(T) \;=\; \{\!\{\, \mathcal{P}ath_u(s) \mid s \in T \;\wedge\; u \in \mathcal{TP}os(s) \,\}\!\}$$

is called the (multi-) set of paths *of T.* ◇

A (path-) *decomposition* of a term is an unordered path.

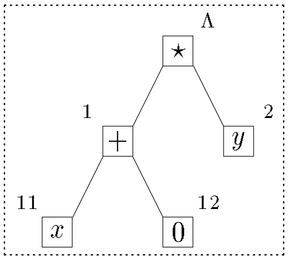
$\boxed{1.36}$ **Definition (Decomposition of a Term)** *Let $t, t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and T be a set of terms. Then*

$$\begin{cases} \mathcal{S}et([t_1,\dots,t_n]) & = \{t_1,\dots,t_n\} & \text{if } [t_1,\dots,t_n] \text{ is a path} \\ \mathcal{D}ec_u(t) & = \mathcal{S}et(\mathcal{P}ath_u(t)) & \text{is called path-decomposition} \end{cases}$$

and the decomposition *of T is defined as*

$$\mathcal{D}ec(T) \;=\; \{\!\{\, \mathcal{D}ec_u(s) \mid s \in T \;\wedge\; u \in \mathcal{TP}os(s) \,\}\!\}.$$ ◇

$\boxed{1.37}$ **Example** *Consider the term $t = (x + 0) \star y$ with its tree representation*



Then

Paths	Decompositions
$\mathcal{P}ath_{11}(t) = [t, x+0, x]$ $\mathcal{P}ath_{12}(t) = [t, x+0, 0]$ $\mathcal{P}ath_{2}(t) = [t, y]$	$\mathcal{D}ec_{11}(t) = \{t, x+0, x\}$ $\mathcal{D}ec_{12}(t) = \{t, x+0, 0\}$ $\mathcal{D}ec_{2}(t) = \{t, y\}$
$\mathcal{P}ath(\{t\}) = \{\!\{\, \mathcal{P}ath_{11}(t), \mathcal{P}ath_{12}(t), \mathcal{P}ath_{2}(t) \,\}\!\}$	$\mathcal{D}ec(\{t\}) = \{\!\{\, \mathcal{D}ec_{11}(t), \mathcal{D}ec_{12}(t), \mathcal{D}ec_{2}(t) \,\}\!\}$

◇

There are two operations on paths and decompositions to be described. They deliver subsequences (subsets, respectively) of the structure they are applied to.

1.38 **Definition (Operations on Paths and Decompositions)**

- *Let* $[t_1,\dots,t_n]$ *be a path. Then the* sequence of subterms *relative to a term* t_i $(i \in [1,n])$ *is defined as* $\mathcal{S}ub(t_i) = [t_{i+1},\dots,t_n]$. *The* sequence of superterms *relative to* t_i *is defined as* $\mathcal{S}up(t_i) = [t_1,\dots,t_{i-1}]$.
- *Let* $\{t_1,\dots,t_n\}$ *be a path-decomposition. Then the* set of subterms *relative to a term* t_i $(i \in [1,n])$ *is defined as* $\mathcal{S}ub(t_i) = \{t_j \mid 1 \leq j \leq n \wedge t_i \rhd t_j\}$. *The* set of superterms *relative to* t_i *is defined as* $\mathcal{S}up(t_i) = \{t_j \mid 1 \leq j \leq n \wedge t_j \rhd t_i\}$. ◇

Note that for $\mathcal{S}ub$ and $\mathcal{S}up$ being unambiguous, the context (i.e. a specific path or path-decomposition) must be given. For the remaining part, this requirement is always fulfilled.

1.39 **Example** *Consider the term* $t = (x+0) \star y$ *of example 1.37 once again. Then*

$$\begin{array}{lcll} \mathcal{S}ub(x+0) & = & [0] & \textit{if the path } [t, x+0, 0] \textit{ is given, and} \\ \mathcal{S}up(t) & = & \emptyset & \textit{for any path-decomposition of } t. \end{array}$$ ◇

1.5.3 Weights

As mentioned previously, we need a well-founded set and a termination function mapping terms into this set for the definition of semantic orderings. Now, we present two such sets and the corresponding termination functions. We start with a termination function using the set $\mathbb{N}$ of natural numbers, i.e. $\mathbb{N} = \{1,2,3,\dots\}$.

In the following, $\mathbb{N}_0$ denotes the set $\mathbb{N} \cup \{0\}$ and $\mathbb{R}_+$ the set of real numbers greater than zero.

1.40 **Definition (Weight Function)** *Let* $\varphi_0 \in \mathbb{N}$ *and* $f \in \mathcal{F}$. *A* weight function φ: $\mathcal{F} \cup \mathcal{X} \mapsto \mathbb{N}_0$ *is defined as follows*[30]*:*

$$\varphi(x) \quad = \quad \varphi_0 \quad \textit{if } x \in \mathcal{X} \qquad \textit{and}$$

$$\varphi(f) \quad \geq \quad \begin{cases} \varphi_0 & \textit{if } \mathcal{A}r(f) = 0 \\ 1 & \textit{if } \mathcal{A}r(f) = 1 \\ 0 & \textit{otherwise} \end{cases}$$

We extend this weight function to φ: $\mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathbb{N}$:

$$\varphi(f(t_1,\dots,t_n)) = \varphi(f) + \sum_{i=1}^{n} \varphi(t_i).$$ ◇

1.41 **Example** *Let* $\mathcal{F} = \{+, \star, s, 0\}$ *be a set of operators such that* $\mathcal{A}r(+) = \mathcal{A}r(\star) = 2$, $\mathcal{A}r(s) = 1$ *and* $\mathcal{A}r(0) = 0$ *hold and* $\varphi_0 = 1$ *is the weight of variables. Then*

f	$+$	$\star$	s	0
$\varphi(f)$	*0*	*2*	*1*	*1*

is a weight function which implies $\varphi(x \star (y + s(0)))$ *equals 6.* ◇

---

[30] It is also possible to use $\mathbb{R}_+ \cup \{0\}$ instead of $\mathbb{N}_0$.

1.5.4 Interpretations

The weight $\varphi(f)$ of an n-ary operator f can be considered as a special linear function $\zeta(f) = x_1+\cdots+x_n+\varphi(f)$. Applying this idea to example 1.41 yields $\zeta(+) = x_1+x_2$, $\zeta(\star) = x_1+x_2+2$, $\zeta(s) = x_1+1$ and $\zeta(0) = 1$. For computing the weight of a term $t = f(t_1,\dots,t_n)$ we use the homomorphism $\zeta(t) = \zeta(f)(\zeta(t_1),\dots,\zeta(t_n))$ which describes the recursive application of ζ to the arguments of t. Note that the sum $x_1+\cdots+x_n$ of the variables of each ζ is responsible for (recursively) adding the arguments of a term. Other (more complex) functions than those of the form $\zeta(f) = x_1+\cdots+x_n+\varphi(f)$ are admissible for $\zeta(f)$. A polynomial ordering even allows polynomials (over natural or possibly real numbers) as the 'weight' of operators.

$\boxed{1.42}$ **Definition (Set of Polynomials)** *Let* $\mathcal{D}$ *be either* $\mathbb{N}$ *or* $\mathbb{R}_+$, $n \in \mathbb{N}_0$. *The set* $\mathcal{D}[x_1,\dots,x_n]$ *of* polynomials *over a set* $\{x_1,\dots,x_n\}$ *of* n *distinct* variables *and with* coefficients *in* $\mathcal{D}$ *is defined as follows:*

$$\mathcal{D}[x_1,\dots,x_n] = \{ \sum_{e_1,\dots,e_n} \alpha_{e_1,\dots,e_n} x_1^{e_1}\cdot\ldots\cdot x_n^{e_n} \mid \alpha_{e_1,\dots,e_n} \in \mathcal{D} \textit{ and } \alpha_{e_1,\dots,e_n} = 0 \textit{ for all but finitely many coefficients}\}.$$

Especially, $\mathcal{D}[\,] = \{c \mid c \in \mathcal{D}\}$ *will be abbreviated to* $\mathcal{D}$. ◇

Each item of the sum forming a polynomial is called a *monomial*. We often use $\alpha_{e_1,\dots,e_n}$ for referring to the exponents of the variables of a monomial.

For simplicity, we omit the commas of the indices,

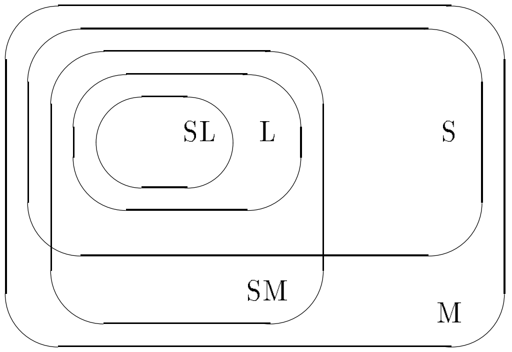
e.g. $\alpha_{210}x^2y + \alpha_{101}xz \in \mathbb{R}_+[x,y,z]$ if $\alpha_{210}, \alpha_{101} \in \mathbb{R}_+$. The next definition represents a possible classification of polynomials. The described classes of polynomials depend on the coefficients as well as the exponents of a polynomial.

$\boxed{1.43}$ **Definition (Classes of Polynomials)** *Let* $\mathcal{D}$ *be either* $\mathbb{N}$ *or* $\mathbb{R}_+$ *and* p *be any polynomial of* $\mathcal{D}[x_1,\dots,x_n]$ *and* $\alpha_i, \alpha_{ij}, \beta \in \mathcal{D}$. *Then*

- p *is* strongly linear *if, and only if,* $p = \sum x_i + \beta$.
- p *is* linear *if, and only if,* $p = \sum \alpha_i x_i + \beta$.
- p *is* separate *if, and only if, each monomial contains at most one variable with arbitrary exponent:* $p = \sum_i(\sum_j \alpha_{ji}x_i^j)$.
- p *is* mixed *if, and only if, it is a general polynomial.*
- p *is* simple-mixed[31)] *if, and only if,* p *is mixed and all exponents are not greater than 1 or* $n = 1$ *and* $p = \alpha_2 x_1^2 + \beta$. ◇

For example, $x+y+5$ is strongly linear, $2x+y+1$ is linear, x^2+2y is separate and the polynomials $2xy+3y$, $5x^2+1$, $3x+2$ are of simple-mixed form. Figure 1.6 illustrates the set-theoretic interdependencies between the different classes of polynomials.

[31)] This kind of polynomials will be used in subsection 5.5.2.



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

Figure 1.6: Classes of polynomials

$\boxed{1.44}$ **Definition (Strictness and Completeness of Polynomials)** *Let* $\mathcal{D}$ *be either* $\mathbb{N}$ *or* $\mathbb{R}_+$ *and* p *be any polynomial of* $\mathcal{D}[x_1,\dots,x_n]$. *Furthermore, let* $\mathcal{C}$ *be any class of definition 1.43. Then*

- p *is called* strict *of a class* $\mathcal{C}$ *if, and only if, it is in* $\mathcal{C}$ *and not contained in any subclass*[32] *of* $\mathcal{C}$.
- p *is called* complete *if, and only if,* $p = \sum \alpha_{e_1\dots e_n} \cdot x_1^{e_1} \cdot \dots \cdot x_n^{e_n}$ *and the following condition holds:*

  If $\alpha_{k_1\dots k_n} > 0$ *then* $\forall i \in [1,n]$: $k_i \geq 1 \hookrightarrow \alpha_{k_1\dots k_{i-1}(k_i-1)k_{i+1}\dots k_n} > 0$. ◇

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

$\boxed{1.45}$ **Example** *The polynomial* $2x^2 + y$ *is strictly separate because it is separate but not linear. Obviously, it is also mixed but not strictly mixed. The polynomial* $2x^2y + 3xy + x^2 + 5x + 5y + 1$ *is complete whereas the polynomial* xy *is not.* ◇

In analogy with the arity function of operators, we define the arity of polynomials.

$\boxed{1.46}$ **Definition (Arity of a Polynomial)** *Let* $\mathcal{D}$ *be either* $\mathbb{N}$ *or* $\mathbb{R}_+$, $n \in \mathbb{N}_0$ *and* $p = \sum_{e_1\dots e_n} \alpha_{e_1\dots e_n} x_1^{e_1} \cdot \dots \cdot x_n^{e_n} \in \mathcal{D}[x_1,\dots,x_n]$. *Then*

$$\mathcal{A}r(p) = |\mathcal{V}ar(p)|$$

where $\mathcal{V}ar(p)$ *denotes the set of variables actually occurring in* p. ◇

Thus, a polynomial $p \in \mathcal{D}[x_1,\dots,x_n]$ possesses the arity m ($\leq n$) if m variables $y_1,\dots,y_m$ (with $\{y_1,\dots,y_m\} \subseteq \{x_1,\dots,x_n\}$) occur in p such that for every y_i there is a monomial in p containing y_i with a non-zero coefficient.

---

[32] see figure 1.6

$\boxed{1.47}$ **Definition ((Polynomial) Interpretation)** *Let* $\mathcal{D}$ *be either* $\mathbb{N}$ *or* $\mathbb{R}_+$. *Then a* polynomial interpretation $[\![.]\!]: \mathcal{F} \cup \mathcal{X} \mapsto \mathcal{D}[\mathcal{V}]$ *over* $\mathcal{D}$ *assigns a polynomial to each* n*-ary operator* f *such that*

$$\begin{array}{l} [\![f]\!](x_1,\ldots,x_n) \in \mathcal{D}[x_1,\ldots,x_n] \quad \wedge \quad \mathcal{A}r([\![f]\!](x_1,\ldots,x_n)) = n \quad \textit{and} \\ [\![x]\!]() \;=\; X \in \mathcal{V} \quad \textit{if } x \in \mathcal{X} \end{array}$$

where $\mathcal{V}$ *is a finite set of (polynomial) variables over* $\mathcal{D}$. *This mapping can be extended to* $[\![.]\!]: \mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{D}[\mathcal{V}]$ *by defining*

$$[\![f(t_1,\ldots,t_n)]\!](x_1,\ldots,x_m) \;=\; [\![f]\!]([\![t_1]\!](x_1,\ldots,x_m),\ldots,[\![t_n]\!](x_1,\ldots,x_m)). \qquad \diamond$$

$\boxed{1.48}$ **Example** *Let* $[\![+]\!](X,Y) \;=\; X + 2Y$, $[\![\star]\!](X,Y) \;=\; XY + Y$, $[\![s]\!](X) \;=\; X^2$ *and* $[\![0]\!]() \;=\; 2$ *be an interpretation over* $\mathbb{N}$ *for* $\mathcal{F} = \{+, \star, s, 0\}$. *Then*

$$\begin{array}{rcl} [\![x \star (y + s(0))]\!](X,Y) & = & X \cdot [\![y + s(0)]\!](X,Y) + [\![y + s(0)]\!](X,Y) \\ & = & X \cdot (Y + 2 \cdot [\![s(0)]\!](X,Y)) + (Y + 2 \cdot [\![s(0)]\!](X,Y)) \\ & = & X \cdot (Y + 2 \cdot [\![0]\!](X,Y)^2) + (Y + 2 \cdot [\![0]\!](X,Y)^2) \\ & = & X \cdot (Y + 2 \cdot 2^2) + (Y + 2 \cdot 2^2) \\ & = & XY + 8X + Y + 8 \end{array} \qquad \diamond$$

In the remaining part of this thesis, we no longer differentiate between capital letters for denoting variables of polynomials and lower case for denoting variables of terms. For simplicity, we adopt the lower case form.

1.5.5 Representations of A-Terms

Usually, terms with associative operators are represented by so-called flattened terms that feature no nested occurrences of identical associative operators, e.g. $+(+(1,+(2,+(1,3))),2) = +(1,2,1,3,2)$. This representation requires the operators to have *variable arity*, i.e. associative operators may possess any positive number (≥ 2) of arguments, whereas non-associative operators have a fixed arity. Keeping this in mind, the flattening operation is defined as follows.

$\boxed{1.49}$ **Definition (Flattening Operation)** *Let* $\mathcal{F}$ *be a set of operators,* $\mathcal{F}_A \subseteq \mathcal{F}$ *and* $t = f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$. *Then*

$$\overline{t} \;=\; \begin{cases} t & \textit{if } t \textit{ is an atom} \\ f(\overline{t_1},\ldots,\overline{t_n}) & \textit{if } f \notin \mathcal{F}_A \\ t' & \textit{otherwise} \end{cases}$$

where t' *results from* t *by replacing* t_i *by* $\overline{t_i}$ *if* $\mathcal{H}ead(t_i) \neq f$, *and replacing* t_i *by* $s_1,\ldots,s_m$ *if* $\overline{t_i} = f(s_1,\ldots,s_m)$. $\diamond$

Note that the flattened form of a term exists and is unique. We now introduce a *new* way of representing terms with associative operators. It somehow generalizes the flattening operation.

1.50 **Definition (Memory-Flattening Operation)**

- **Signature enrichment:** *Let* $\mathcal{F} = \mathcal{F}' \cup \mathcal{F}_A$ *such that* $\mathcal{F}' \cap \mathcal{F}_A = \emptyset$. *Then*

$$\mathcal{F}_A^E = \{ f^i \mid f \in \mathcal{F}_A \ \wedge \ i \geq 1 \}$$

such that $f^1 = f$, $\mathcal{A}r(f^i) = i + 1$ *and*

$$\mathcal{F}^E = \mathcal{F}' \cup \mathcal{F}_A^E.$$

- **Memory-flattening operation:** Memory-flattening *is a mapping from* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *into* $\mathcal{T}(\mathcal{F}^E,\mathcal{X})$. *Let* $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ *and* $\overline{t} = f(t_1,\ldots,t_n)$ *be its flattened version*[33]. *Then*

$$\widetilde{t} = \begin{cases} f^{n-1}(\widetilde{t_1},\ldots,\widetilde{t_n}) & \textit{if } f \in \mathcal{F}_A \\ f(\widetilde{t_1},\ldots,\widetilde{t_n}) & \textit{otherwise} \end{cases} \qquad \diamond$$

Thus, the new memory-flattening operation not only flattens a term in the traditional sense but, at the same time, stores the number of associative operators which have been removed[34].

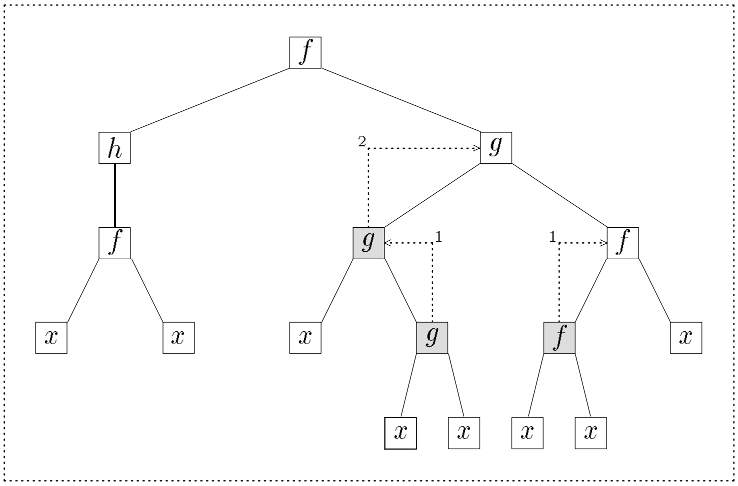
1.51 **Example** *Let* $\mathcal{F} = \{f, g, h\}$ *such that* $\mathcal{F}_{AC} = \{f, g\}$ *and let*

$$t = f(h(f(x,x)), g(g(x, g(x,x)), f(f(x,x),x)))$$

be a term. Then

$$\begin{aligned} \overline{t} &= f(h(f(x,x)), g(x,x,x,f(x,x,x))) \qquad \textit{and} \\ \widetilde{t} &= f^1(h(f^1(x,x)), g^3(x,x,x,f^2(x,x,x))). \end{aligned}$$

This can be illustrated by the following tree:



◇

1.5.6 Miscellaneous

As mentioned in section 1.4, most of the orderings in this thesis are so-called *precedence-based* orderings using a special ordering on operators. More precisely, a *precedence* will either be a

---

[33] For simplicity, the memory-flattened form of a term t is defined using the flattened form $\overline{t}$ of t.

[34] Since each term has finitely many occurrences of operators, it is sufficient for the set $\mathcal{F}_A^E$ to be finite. Thus, corollary 1.19 on page 10 would be applicable to an ordering defined on $\mathcal{F}^E$.

partially ordered set $(\mathcal{F}, \succ)$ or a quasi-ordered set $(\mathcal{F}, \succsim)$ consisting of the set $\mathcal{F}$ of operators and a transitive irreflexive binary relation $\succ$ or a transitive reflexive binary relation $\succsim$ defined on the elements of $\mathcal{F}$. The equivalence relation on operators is denoted by $\sim$. We use $\succ_{\mathsf{ORD}}$ for denoting an ordering ORD using either $\succ$ or $\succsim$.

Each definition of an ordering (contained in chapters 3, 4 and 5) is preceded by an abstract verbal description ('Idea of comparison') of its operational method, i.e. the technique of comparing terms. Furthermore, some helpful and interesting remarks are given.

References to related work can be found in paragraphs marked with huge letters at their beginnings. These boxes also include references to original literature.

Most of the definitions and results will be motivated and/or illustrated by examples (i.e. TRSs). These examples are contained in appendix A. References to the related TRSs are given in the margin[35)].

To specify nomenclature: as often as not, the letters a through i will be used for operators; l, r, s and t will denote arbitrary terms; x, y and z will be reserved for variables, u and v for positions; σ will represent a substitution and p, q will stand for polynomials.

[35)] Examples the references of which are contained in the margins of *definitions* usually provide more detailed information (e.g. traces of the applications of orderings) than examples of other margins.

Chapter 2

Status Function and Representation of Orderings

Throughout the last few years numerous (simplification) orderings have been defined by various authors. Most of them are so-called *precedence-based orderings* and use a special ordering (a precedence) on operators. Accordingly, most of them are so-called *root orderings*: The result of comparing two terms depends on their leading operators (the roots of the terms) as well as on recursive applications of the ordering to the arguments of the terms.

- ▶ **Section 2.1:** In this section, the different ways the recursive application of an ordering to the arguments of terms can be executed will be described.
- ▶ **Section 2.2:** Due to their recursive definitions, some orderings are somewhat complex. In order to reduce redundancies and make the definitions more legible, a formalism for defining orderings will be introduced.

2.1 The Status Function

We are dealing with orderings on terms. Since operators have terms as arguments, we define an extension of a reduction ordering $\succ_{\mathsf{T}}$, called *lexicographic extension* ($\succ_{\mathsf{T}}^{\mathsf{lex}}$) to an ordering, on tuples of terms as follows.

$\boxed{2.1}$ **Definition (Lexicographic Extension)** *Let* $\succ_{\mathsf{T}}$ *be an ordering on terms,* $\sim_{\mathsf{T}}$ *an equivalence relation on terms with* $\succ_{\mathsf{T}} \cap \sim_{\mathsf{T}} = \emptyset$ *and* $s_i, t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$*. Then*

$$(s_1,\ldots,s_m) \succ_{\mathsf{T}}^{\mathsf{lex}} (t_1,\ldots,t_n) \iff$$

$$\begin{cases} m > 0 \;\wedge\; n = 0 & \textit{or} \\ s_1 \succ_{\mathsf{T}} t_1 & \textit{or} \\ s_1 \sim_{\mathsf{T}} t_1 \;\wedge\; (s_2,\ldots,s_m) \succ_{\mathsf{T}}^{\mathsf{lex}} (t_2,\ldots,t_n) \end{cases} \qquad \diamond$$

In definition 2.1, the arguments $s_1,\ldots,s_m$ and $t_1,\ldots,t_n$ of terms s and t, respectively, are compared as sequences. They may also be compared as multisets $\{\!\{ s_1,\ldots,s_m \}\!\}$ and $\{\!\{ t_1,\ldots,t_n \}\!\}$.

$\mathcal{M}ult(S)$ denotes the set of all finite multisets with elements taken from S, i.e. the multisets M with finite carrier $\{x \in S \mid M(x) \neq 0\}$. Assume that S represents the set $\mathcal{T}(\mathcal{F},\mathcal{X})$ of all terms which is (partially) ordered by an ordering $\succ_{\mathsf{T}}$ on terms. The extension of $\succ_{\mathsf{T}}$ on multisets of terms is defined as follows.

$\boxed{2.2}$ **Definition (Multiset Extension, [Dershowitz and Manna, 1979])** *Let $\succ_{\mathsf{T}}$ be an ordering on terms, $\sim_{\mathsf{T}}$ an equivalence relation on terms such that $\succ_{\mathsf{T}} \cap \sim_{\mathsf{T}} = \emptyset$ holds and $A, B \in \mathcal{M}ult(\mathcal{T}(\mathcal{F},\mathcal{X}))$. Then*

$$\begin{array}{l} A \succ^{\textcircled{m}}_{\mathsf{T}} B \quad \Longleftrightarrow \\ A \setminus_{\mathsf{T}} B \neq \emptyset \quad \wedge \quad \forall t \in (B \setminus_{\mathsf{T}} A)\colon \exists s \in (A \setminus_{\mathsf{T}} B)\colon s \succ_{\mathsf{T}} t \end{array}$$

where $A \setminus_{\mathsf{T}} B$ is the multiset difference of A and B w.r.t. $\sim_{\mathsf{T}}$, i.e.

$$A \setminus_{\mathsf{T}} B = \begin{cases} A & \text{if } \nexists x \in A, y \in B\colon x \sim_{\mathsf{T}} y \\ (A \setminus \{x\}) \setminus_{\mathsf{T}} (B \setminus \{y\}) & \text{if } x \in A, y \in B \ \wedge \ x \sim_{\mathsf{T}} y \end{cases} \qquad \diamond$$

Thus, $A \succ^{\textcircled{m}}_{\mathsf{T}} B$ if B can be obtained from A by replacing one or more terms in A by any finite number of terms, each of which is smaller (w.r.t. $\succ_{\mathsf{T}}$) than one of the replaced terms. Note that the equality on elements of the multisets A and B is an equivalence relation ($\sim_{\mathsf{T}}$). Remember that, in this case, equivalence classes will be considered[1)].

There exist alternative versions, with equivalent power, of the above definition. Multiset orderings were first proposed by [Dershowitz and Manna, 1979]. Definition 2.2 is taken from [Pettorossi, 1981]. Other equivalent definitions are contained, for example, in [Huet and Oppen, 1979], [Martin, 1986], [Mohan and Srivas, 1986] and [Dershowitz, 1987]. In [Martin, 1989], a classification of multiset orderings is given based on the notion of tame ordering *which, roughly speaking, is an ordering on $\mathcal{M}ult(S)$ preserving both multiset union and an ordering on the set S. For a classification of total multiset orderings for Gröbner basis algorithms, see [Robbiano, 1985].*

In order to combine the concepts of tuples and multisets and thus, to combine the lexicographic with the multiset extension of an ordering on terms, we assign a so-called *status* $\tau(f)$ to each operator $f \in \mathcal{F}$ that determines the order according to which the arguments of f are compared.

The status function goes back to [Kamin and Lévy, 1980] introducing a functional $\gg$ on relations. $\gg$ is a function from $2^{\mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X})}$ into $2^{\mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X})}$. Kamin & Lévy integrated $\gg$ into the recursive path ordering RPO *of Dershowitz and proved that the induced* RPO *is a simplification ordering if $\gg$ possesses the following properties: irreflexivity, transitivity, continuity w.r.t. the subset topology[2)] and a modified $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility[3)]. The expression 'status' was first used by Lescanne in [Lescanne, 1984] (and in [Forgaard, 1984]).*

[1)] For example, $\{\!\{ a, b \}\!\} \setminus_{\mathsf{T}} \{\!\{ a \}\!\}$ is either $\{\!\{ a \}\!\}$ or $\{\!\{ b \}\!\}$ if $a \sim_{\mathsf{T}} b$ holds. However, a and b are elements of the same equivalence class. Thus, $\{\!\{ a \}\!\}$ and $\{\!\{ b \}\!\}$ are equivalent w.r.t. $\sim_{\mathsf{T}}$.

[2)] $s \gg_{\succ_{\mathsf{RPO}}} t$ implies that there is a finite subset $\succ_{\mathsf{SRPO}}$ of $\succ_{\mathsf{RPO}}$ such that $s \gg_{\succ_{\mathsf{SRPO}}} t$. Thus, in order to check $s \gg_{\succ_{\mathsf{RPO}}} t$, one needs to test a finite number of pairs $s' \succ_{\mathsf{RPO}} t'$, only.

[3)] $s \succ_{\mathsf{RPO}} t$ implies $f(\ldots, s, \ldots) \gg_{\succ_{\mathsf{RPO}}} f(\ldots, t, \ldots)$.

$\boxed{2.3}$ **Definition (Status Function, Based on [Kamin and Lévy, 1980])** *Let* $\mathcal{F}$ *be a set of operators and* $\succsim$ *be a quasi-precedence over* $\mathcal{F}$*. Then the* status function τ *assigns to each* $f \in \mathcal{F}$ *one of the following values:*

$$\tau(f) = \begin{cases} \textcircled{m} & \textit{or} \\ (\pi(1),\ldots,\pi(\mathcal{A}r(f))) & \textit{where } \pi \textit{ is a permutation of } \{1,\ldots,\mathcal{A}r(f)\} \end{cases}$$

The following condition for quasi-precedences must be guaranteed:

$$\forall f,g \in \mathcal{F}\colon g \sim f \ \wedge \ \tau(g) = \textcircled{m} \quad \rightsquigarrow \quad \tau(f) = \textcircled{m}$$

The following special cases will be used: $\tau(f) = \textcircled{l}$ *stands for* $\tau(f) = (1,2,\ldots,\mathcal{A}r(f))$ *whereas* $\tau(f) = \textcircled{r}$ *stands for* $\tau(f) = (\mathcal{A}r(f),\mathcal{A}r(f)-1,\ldots,1)$[4]*. Furthermore, if the status of an operator* f *is omitted, then* $\tau(f) = \textcircled{m}$*.* ◇

The status of an operator is irrelevant if its arity is less than two. Note that two operators f and g which are $\sim$-equivalent can have different *lexicographic* types of status. However, lexicographic status and multiset status cannot be mixed. The application of the different types of status to collections of terms is defined as follows.

$\boxed{2.4}$ **Definition (Status Extension of an Ordering)** *Let* $s = f(s_1,\ldots,s_m)$ *and* $t = g(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$*,* $f \sim g$ *and* τ *be a status function over* $\mathcal{F}$*. Furthermore, let* $\succ_{\mathsf{T}}$ *be an ordering on terms. Then*

$$\mathcal{A}rgs(s) \succ_{\mathsf{T}}^{\tau} \mathcal{A}rgs(t) \iff$$

$$\begin{cases} \{\!\{ s_1,\ldots,s_m \}\!\} \succ_{\mathsf{T}}^{\textcircled{m}} \{\!\{ t_1,\ldots,t_n \}\!\} & \textit{if } \tau(f) = \tau(g) = \textcircled{m} \\ (s_{i_1},\ldots,s_{i_m}) \succ_{\mathsf{T}}^{\mathsf{lex}} (t_{j_1},\ldots,t_{j_n}) & \textit{if } \tau(f) = (i_1,\ldots,i_m) \ \wedge \ \tau(g) = (j_1,\ldots,j_n) \end{cases}$$ ◇

The multiset extension as well as the lexicographic extension of an ordering $\succ_{\mathsf{T}}$ need an equivalence relation (denoted by $\sim_{\mathsf{T}}$) which eliminates equivalent terms.

$\boxed{2.5}$ **Definition (Status Extension of an Equivalence Relation)** *Let* $s = f(s_1,\ldots,s_m)$ *and* $t = g(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$*,* $f \sim g$ *and* τ *be a status function over* $\mathcal{F}$*. Furthermore, let* $\sim_{\mathsf{T}}$ *be an equivalence relation on* $\mathcal{T}(\mathcal{F},\mathcal{X})$*. Then*

$$\mathcal{A}rgs(s) \sim_{\mathsf{T}}^{\tau} \mathcal{A}rgs(t) \iff$$

$$m = n \quad \wedge$$

$$\begin{cases} \mathcal{A}rgs(s) \setminus_{\mathsf{T}} \mathcal{A}rgs(t) \ = \ \emptyset & \textit{if } \tau(f) = \tau(g) = \textcircled{m} \\ \forall k \in [1,n]\colon s_{i_k} \sim_{\mathsf{T}} t_{j_k} & \textit{if } \tau(f) = (i_1,\ldots,i_n) \ \wedge \ \tau(g) = (j_1,\ldots,j_n) \end{cases}$$ ◇

Most of the orderings presented in this thesis uniquely define a congruence $\approx$ dependent on $\mathcal{F}$ and τ.

---

[4] $\textcircled{l}$ stands for 'left to right' and $\textcircled{r}$ for 'right to left'.

2.6 **Definition (Congruence $\approx$)** *Let* $s = f(s_1,\dots,s_m)$, $t = g(t_1,\dots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{X})$, $\succsim$ *a quasi-precedence and* τ *a status function over* $\mathcal{F}$. *Then*

$$s \approx t \iff$$

$$f \sim g \quad \wedge \quad \mathcal{A}rgs(s) \approx^{\tau} \mathcal{A}rgs(t) \qquad \diamond$$

The expression $\approx^{\tau}$ describes the extension of $\approx$ to a status function τ (see definition 2.5).

2.7 **Lemma** $\approx$ *is a congruence on* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *and* σ*-stable.* $\diamond$

A precedence and a status function are parameters of an ordering on terms[5].

> Note how we use the term 'ordering' in this thesis. Ordering can either mean a *concrete* ordering, parameterized with specific instances for $\succsim$ and τ, or a *class* of orderings, if $\succsim$ and τ are omitted.

The meaning will be clear from the context.

2.2 Representing Orderings

We introduce a formalism that allows clear representations of orderings. It comprises the following features that often occur in definitions of orderings:

- An ordering can be applied to two terms which are possibly modified by different operations[6].
- An ordering can be defined as a disjunction (or a conjunction) of conditions or orderings.
- An ordering can be defined as the lexicographic concatenation of other orderings.

In the following subsections, we describe how these features are represented in the formalism.

2.2.1 Operations on Terms

The main property of our formalism is given by its abstraction from objects that operations are applied to.

2.8 **Definition** *Let* $\theta, \kappa\colon A \mapsto B$ *be two functions and* $a, b \in A$. *If* ∇ *is a binary operation over* B, *then*

$$\theta(a) \nabla \kappa(b) \quad \textit{will be represented as} \quad \nabla\langle\theta,\kappa\rangle.$$

---

[5] In [Forgaard, 1984], the tuple $(\succ, \tau)$ is called *registry*.

[6] Such operations can be, for example, the identity, $\mathcal{H}ead$ or $\mathcal{A}rgs$.

Moreover, the following abbreviations will be used:

$$\left.\begin{array}{r}\nabla\langle\theta\rangle\\ \succ\\ \succ^{\tau}_{\mathsf{T}}\end{array}\right\}\begin{array}{c}\textit{stands}\\ \textit{for}\end{array}\left\{\begin{array}{l}\nabla\langle\theta,\theta\rangle\\ \succ\langle\mathcal{H}ead\rangle\\ \succ^{\tau}_{\mathsf{T}}\langle\mathcal{A}rgs\rangle\end{array}\right.$$

where $\succ$ *is a precedence,* τ *is a status function,* $\mathcal{H}ead$ *and* $\mathcal{A}rgs$ *are the selector functions on terms defined in subsection 1.5.1 and* $\succ_{\mathsf{T}}$ *is an ordering on terms.* ◇

This formalism simplifies expressions of the form $\theta(a)\ \nabla\ \kappa(b)$ by leaving the input elements a and b out. With regard to definitions of orderings on terms, their essentials – the procedures determining *how* to compare terms – will be emphasized, whereas less important information – concrete data about *what* to compare – will be concealed.

$\boxed{2.9}$ **Example** *Let* $a, b \in \mathbb{R}$, $f, g \in \mathcal{F}$ *and* $s, t \in \mathcal{T}(\mathcal{F},\mathcal{X})$. *Then the two expressions in each row of the following table are equivalent:*

$\geq\langle\lceil\,\rceil\,,\,\lfloor\,\rfloor\rangle$	$\iff$	$\lceil a\rceil\ \geq\ \lfloor b\rfloor$
$\neq\langle\mathcal{A}r\rangle$	$\iff$	$\mathcal{A}r(f)\ \neq\ \mathcal{A}r(g)$
$\cap\langle\mathcal{V}ar\rangle$	$\iff$	$\mathcal{V}ar(s)\ \cap\ \mathcal{V}ar(t)$
$=\langle\mathcal{P}os\,,\,\mathcal{TP}os\rangle$	$\iff$	$\mathcal{P}os(s)\ =\ \mathcal{TP}os(t)$

For the expressions of the second and the third row, abbreviations of the type '$\nabla\langle\theta\rangle$ *instead* $\nabla\langle\theta,\theta\rangle$*' have been used.* ◇

The abbreviations $\nabla\langle\theta\rangle$, $\succ$ and $\succ^{\tau}_{\mathsf{T}}$ of definition 2.8 are very useful in practice since

(i) many orderings compare two objects both of which have previously been modified in the *same* way ($\nabla\langle\theta\rangle \iff \nabla\langle\theta,\theta\rangle$),

(ii) most of the presented orderings are *root* orderings ($\succ \iff \succ\langle\mathcal{H}ead\rangle$ where $\succ$ is a precedence) and

(iii) recursive applications to *arguments* of terms are contained in almost all orderings on terms ($\succ^{\tau}_{\mathsf{T}} \iff \succ^{\tau}_{\mathsf{T}}\langle\mathcal{A}rgs\rangle$ where $\succ_{\mathsf{T}}$ is an ordering on terms).

The functions θ and κ occurring in definition 2.8 can also be composed of other functions. Let θ: $A \mapsto B$ and κ: $B \mapsto C$ be two functions. Then the composition $\kappa \circ \theta$ is defined as $(\kappa \circ \theta)(x) = \kappa(\theta(x))$. Furthermore, parallel evaluation of functions is denoted by $<\kappa, \theta>$. Assuming κ: $A \mapsto B$ and ι: $A \mapsto C$, we define $<\kappa, \iota>(x) := (\kappa(x), \iota(x))$. Moreover,

---

$\mathcal{I}d$ represents the identity function. In most cases, $\mathcal{I}d$ provides the identity as a multiset, e.g. $\mathcal{I}d(t) = \{\!\{\, t \,\}\!\}$.

$\boxed{2.10}$ **Example** *Let* $a, b \in \mathbb{R}$ *and* $s, t \in \mathcal{T}(\mathcal{F},\mathcal{X})$. *Then the two expressions in each row of the following table are equivalent:*

$\geq\langle + \circ < \mathcal{A}r \circ \mathcal{H}ead\,, \lvert . \rvert \circ \mathcal{S}ym > \rangle$	$\Longleftrightarrow$	$\mathcal{A}r(\mathcal{H}ead(s)) + \lvert \mathcal{S}ym(s) \rvert \geq \mathcal{A}r(\mathcal{H}ead(t)) + \lvert \mathcal{S}ym(t) \rvert$
$-\langle \mathcal{I}d\,, {}^2 \circ ! \rangle$	$\Longleftrightarrow$	$a - (b!)^2$
$=\langle \lvert . \rvert \circ \mathcal{V}ar \circ .\vert_1 \rangle$	$\Longleftrightarrow$	$\lvert \mathcal{V}ar(s\vert_1) \rvert = \lvert \mathcal{V}ar(t\vert_1) \rvert$

◇

2.2.2 Disjunction and Conjunction

Disjunctions and conjunctions of conditions or orderings are the main features for constructing orderings. Our formalism for representing orderings only replaces the logical symbols $\wedge$ and $\vee$ since they looks somewhat similar.

$\boxed{2.11}$ **Definition** *Let B_i be Boolean expressions. Then*

$$\left.\begin{array}{l} B_1 \vee \cdots \vee B_n \\ B_1 \wedge \cdots \wedge B_n \end{array}\right\} \begin{array}{l} \textit{will be rep-} \\ \textit{resented as} \end{array} \left\{\begin{array}{l} \{B_1, \ldots, B_n\} \\ B_1 \& \cdots \& B_n \end{array}\right. \qquad \diamond$$

2.2.3 Lexicographic Concatenation

The lexicographic concatenation of orderings is a very powerful technique to construct new orderings:

> Let $\succ_{i}$ be orderings on terms and $\sim_{i}$ be the corresponding equivalence relations.
> Then
> $$\begin{array}{rcl} s \succ_{T} t & \Longleftrightarrow & s \succ_{1} t \quad \text{or} \\ & & s \sim_{1} t \quad \wedge \quad s \succ_{2} t \quad \text{or} \\ & & \vdots \qquad\qquad\qquad\qquad\qquad (*) \\ & & s \sim_{1} t \quad \wedge \quad s \sim_{2} t \quad \wedge \cdots \wedge \quad s \sim_{n-1} t \quad \wedge \quad s \succ_{n} t \end{array}$$
> defines a new ordering $\succ_{T}$. Obviously, the terms s and t may be modified before applying $\succ_{i}$ ($\sim_{i}$).

This technique will be formalized as follows.

$\boxed{2.12}$ **Definition** *Let ω_i, ξ_i: $A_i \mapsto B_i$ be functions, $\gg_i$ be orderings over B_i and $\cong_i$ be the associated equivalence relations over B_i. Then*

$$\bigcup_{i=1}^{n} \{\gg_i\langle\omega_i, \xi_i\rangle \;\&\; \bigwedge_{j=1}^{i-1} \cong_j\langle\omega_j, \xi_j\rangle\}$$

will be represented as

$$(\gg_1\langle\omega_1, \xi_1\rangle, \ldots, \gg_n\langle\omega_n, \xi_n\rangle). \qquad \diamond$$

Using our formalism the definition of $\succ_{T}$ in $(*)$ can be represented as

$$\begin{array}{rcl} s \succ_{T} t & \Longleftrightarrow & \gg_1\langle\omega_1, \xi_1\rangle \quad \text{or} \\ & & \cong_1\langle\omega_1, \xi_1\rangle \quad \wedge \quad \gg_2\langle\omega_2, \xi_2\rangle \quad \text{or} \\ & & \vdots \\ & & \cong_1\langle\omega_1, \xi_1\rangle \quad \wedge \quad \cong_2\langle\omega_2, \xi_2\rangle \quad \wedge \cdots \wedge \quad \cong_{n-1}\langle\omega_{n-1}, \xi_{n-1}\rangle \quad \wedge \quad \gg_n\langle\omega_n, \xi_n\rangle. \end{array}$$

Now, definition 2.12 allows an even more compact notation for $\succ_{\mathrm{T}}$:

$$(\gg_1\langle\omega_1,\xi_1\rangle,\ldots,\gg_n\langle\omega_n,\xi_n\rangle)$$

where the outermost parentheses indicate the lexicographic concatenation.

2.13 **Example** *Let $s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and $\succ$ be a precedence on $\mathcal{F}$. Consider the following ordering on terms:*

$$\begin{array}{lllll} s \succ_{\mathrm{T}} t & \iff & \mathcal{S}ym(s) \supset \mathcal{S}ym(t) & \text{or} & \\ & & \mathcal{S}ym(s) = \mathcal{S}ym(t) & \wedge & \mathcal{H}ead(s) \succ \mathcal{H}ead(t) \end{array}$$

Applying our formalism, $\succ_{\mathrm{T}}$ can be represented by

$$(\supset\langle\mathcal{S}ym\rangle\,,\,\succ).$$

◇

2.2.4 Orderings on Terms

In order to supply an ordering with a specific name, the following definition will be used.

2.14 **Definition** *An ordering $\succ_{\mathrm{T}}$ on terms is represented by*

$$\succ_{\mathrm{T}} \quad\equiv\quad <\!body\!>$$

where <body> is an expression composed of the features defined in definitions 2.8, 2.11 and 2.12. ◇

Note that this definition allows recursive calls in <*body*> of the specified ordering $\succ_{\mathrm{T}}$. We illustrate the above formalism with two examples.

2.15 **Example** *Consider the following reduction ordering on* monadic *terms:*

> *First of all, the set of variables belonging to the presumably smaller term must be a subset of that of the presumably greater term. Furthermore, a term s is greater than a term t if s is longer than t, or if they have the same number of symbols but the leading operator of s is greater than that of t w.r.t. a precedence, or if they are of the same length and their leading operators are identical but the argument of s is (recursively) greater than that of t.*

A formal representation of this ordering follows:

$$\begin{array}{lllllll} s \succ_{\mathrm{T}} t \quad \iff & & & & & & \\ \mathcal{V}ar(s) \supseteq \mathcal{V}ar(t) \ \wedge & & & & & & \\ (i) & |s| > |t| & \text{or} & & & & \\ (ii) & |s| = |t| & \wedge & \mathcal{H}ead(s) \succ \mathcal{H}ead(t) & \text{or} & & \\ (iii) & |s| = |t| & \wedge & s = f(s'),\, t = f(t') & \wedge & s' \succ_{\mathrm{T}} t' & \end{array}$$

The above representation of $\succ_{\mathsf{T}}$ *is realized with the formalism of definition 2.14 as*

$$\succ_{\mathsf{T}} \quad \equiv \quad \supseteq\langle \mathcal{V}ar\rangle \quad \& \quad (>\langle |.| \rangle, \succ, \overset{\textcircled{1}}{\succ_{\mathsf{T}}}) \qquad \diamond$$

$\boxed{\mathbf{2.16}}$ **Example** *Let* $s, t \in \mathcal{T}(\mathcal{F},\mathcal{X})$, $\succsim$ *be a quasi-precedence on* $\mathcal{F}$ *and*

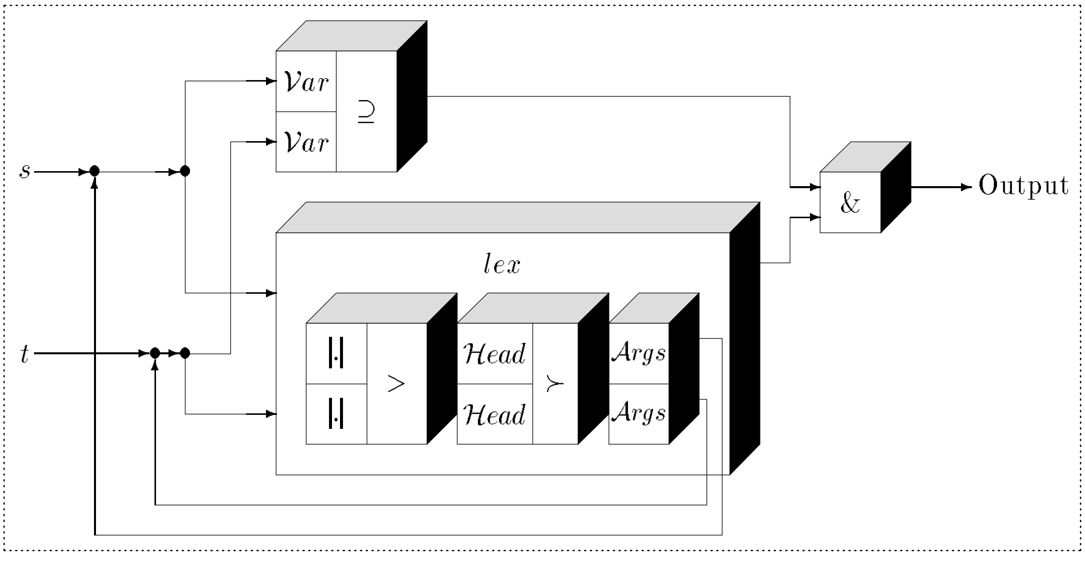
$$\begin{array}{lll} s \succ_{\mathsf{T}} t & \Longleftrightarrow & \mathcal{V}ar(s) \supseteq \mathcal{V}ar(t) \\ & & \textit{and one of the following conditions holds:} \\ & & \begin{array}{ll} (i) & |\mathcal{TP}os(s)| > |\mathcal{TP}os(t)| \\ (ii) & |\mathcal{TP}os(s)| = |\mathcal{TP}os(t)| \ \wedge \ \mathcal{S}ym(s) \supset \mathcal{S}ym(t) \\ (iii) & |\mathcal{TP}os(s)| = |\mathcal{TP}os(t)| \ \wedge \ \mathcal{S}ym(s) = \mathcal{S}ym(t) \ \wedge \ \mathcal{H}ead(s) \succ \mathcal{H}ead(t) \\ (iv) & |\mathcal{TP}os(s)| = |\mathcal{TP}os(t)| \ \wedge \ \mathcal{S}ym(s) = \mathcal{S}ym(t) \ \wedge \ \mathcal{H}ead(s) \sim \mathcal{H}ead(t) \\ & \wedge \ \mathcal{A}rgs(s) \overset{\textcircled{m}}{\succ_{\mathsf{T}}} \mathcal{A}rgs(t) \end{array} \end{array}$$

For example, $f(x,a) \succ_{\mathsf{T}} g^n(x)$ *and* $g(g(f(x,a))) \succ_{\mathsf{T}} f(g(x),g(a))$ *if* $\mathcal{F} = \{f,g,a\}$ *and* $g \succ f$. *With the help of the formalism of definition 2.14,* $\succ_{\mathsf{T}}$ *is defined as*

$$\succ_{\mathsf{T}} \quad \equiv \quad \supseteq\langle \mathcal{V}ar\rangle \ \& \ (>\langle |.| \circ \mathcal{TP}os\rangle \, , \supset\langle \mathcal{S}ym\rangle \, , \succ \, , \overset{\textcircled{m}}{\succ_{\mathsf{T}}}). \qquad \diamond$$

The representation of orderings introduced in this section has a close connection to the methods of [Dershowitz and Hoot, 1993], [Dershowitz, 1983] and has the following properties:

- ▶ It reveals the operational semantics of the orderings, only. Objects of less importance (such as terms) are omitted on purpose.
- ▶ Our formalism supports the (more or less) automatic implementation of orderings. A set of certain basic operations can be provided which, according to the representation of an ordering, can easily be combined to realize the represented ordering (see section 7.2). For instance, the following diagram can be a representation (suitable for an implementation) of the ordering defined in example 2.15:



- Subtle differences in orderings can easily be detected. For instance, consider example 2.15 and example 2.16. It can immediately perceived that the general structures of both orderings are identical: $\supseteq\langle\mathcal{V}ar\rangle \wedge (...)$. Furthermore, the lexicographic concatenation components $(...)$ have two parts in common: $(\ldots, \succ, \succ_{\top}^{\mathcal{T}})$. Finally, the first elements of the $(...)$'s are very similar since both compare sizes $(|.|)$ w.r.t. the ordering on $\mathbb{N}$.
  The more complex orderings are, the more helpful our formalism will be for comparing the structures of orderings. For example, see the definitions of decomposition orderings in section 3.2.

Chapter 3

Path and Decomposition Orderings

This chapter deals with various, well-known as well as new, syntactic simplification orderings. These orderings can be divided into two subclasses, namely *path orderings* and *decomposition orderings.*

- ▶ **Section 3.1:** Subsection 3.1.1 provides a rather complete collection of traditional path orderings, including the path of subterms ordering of Plaisted, the recursive path ordering with status of Dershowitz, Kamin and Lévy and the path ordering with status of Kapur, Narendran and Sivakumar. In subsection 3.1.2, two well-known decomposition orderings developed by Jouannaud, Lescanne, Reinig and Rusinowitch are presented by using the original definition of decompositions. Alternative definitions of these two orderings (extended by the well-known integration of a status function) are treated in subsection 3.1.3. A concept for simplifying the data structure of a decomposition is introduced by taking the definitions of subsection 1.5.2 into account.

- ▶ **Section 3.2:** We develop several new decomposition orderings: (i) an extension of the original recursive decomposition ordering, (ii) an improvement of the path of subterms ordering based on decompositions and a status function and (iii) an extension of the improved recursive decomposition ordering of Rusinowitch by incorporating a status function.

- ▶ **Section 3.3:** A brief classification of the well-known method guaranteeing AC-termination w.r.t. C-termination (subsection 3.3.1), A-termination (subsection 3.3.2) and AC-termination (subsection 3.3.3) based on the recursive path ordering is given. Furthermore, in subsection 3.3.4, we show that this technique (called associative path ordering) cannot use quasi-orderings on AC-operators. Most importantly, an improvement of this ordering is presented in subsection 3.3.5 that weakens the conditions required for the precedence as well as incorporates a status function. Finally, we prove that this extended concept can also be applied to the strongest path and decomposition ordering of sections 3.1 and 3.2.

- ▶ **Section 3.4:** A brief statement of the main characteristics of several well-known techniques for generating adequate path and decomposition orderings is given. This section contains as good as no new results and is supplied for the sake of completeness and consistency with the other chapters as well as for introducing the problem of automatically producing appropriate parameters of orderings for given TRSs.

3.1 Well-Known Path and Decomposition Orderings

3.1.1 Path Orderings

One of first path orderings, called *path of subterms ordering*, was developed by Plaisted in 1978 (see [Plaisted, 1978b], [Plaisted, 1978a]). A slightly modified version (equivalent to the original[1]) due to Rusinowitch is given in definition 3.1.

> **Idea of Comparison** *The path of subterms ordering compares two terms by comparing all their paths. A path p is greater than a path q if the set of terms occurring in p is greater than that of q w.r.t. the ordering comparing the leading operators first, and if they are equal, comparing the paths of their arguments.*

$\boxed{3.1}$ **Definition (Path of Subterms Ordering, [Plaisted, 1978a], [Rusinowitch,**
A.8 **1987])** *Let* $\succ$ *be a total precedence. Then the* path of subterms ordering PSO *is defined as*[2]

$$\begin{array}{lcllcl} \succ_{\mathsf{PSO}} & \equiv & \overset{\textcircled{m}}{\succ}_{\mathsf{PO}}\langle \mathcal{P}ath\rangle & \sim_{\mathsf{PSO}} & \equiv & \approx\langle \mathcal{I}d\rangle \\ \succ_{\mathsf{PO}} & \equiv & \overset{\textcircled{m}}{\succ}_{\mathsf{TO}}\langle \mathcal{S}et\rangle & \sim_{\mathsf{PO}} & \equiv & \overset{\textcircled{m}}{\sim}_{\mathsf{TO}}\langle \mathcal{S}et\rangle \\ \succ_{\mathsf{TO}} & \equiv & (\succ, \overset{\textcircled{m}}{\succ}_{\mathsf{PO}}\langle \mathcal{P}ath \circ \mathcal{A}rgs\rangle) & \sim_{\mathsf{TO}} & \equiv & \approx\langle \mathcal{I}d\rangle \end{array} \qquad \diamond$$

We will illustrate the definition of the PSO in more detail, since for the first time, a concrete ordering is described with our formalism for representing orderings. The topmost expression '$\overset{\textcircled{m}}{\succ}_{\mathsf{PO}}\langle \mathcal{P}ath\rangle$' applies the operation '$\mathcal{P}ath$' to the input terms (s,t) and compares the results w.r.t. the ordering $\overset{\textcircled{m}}{\succ}_{\mathsf{PO}}$: $\mathcal{P}ath(\{s\}) \overset{\textcircled{m}}{\succ}_{\mathsf{PO}} \mathcal{P}ath(\{t\})$. The ordering $\succ_{\mathsf{PO}}$ is defined as $\overset{\textcircled{m}}{\succ}_{\mathsf{TO}}\langle \mathcal{S}et\rangle$, i.e. $[s_1,\ldots,s_m] \succ_{\mathsf{PO}} [t_1,\ldots,t_n]$ if, and only if, $\{s_1,\ldots,s_m\} \overset{\textcircled{m}}{\succ}_{\mathsf{TO}} \{t_1,\ldots,t_n\}$. The definition '$(\succ, \overset{\textcircled{m}}{\succ}_{\mathsf{PO}}\langle \mathcal{P}ath \circ \mathcal{A}rgs\rangle)$' of $\succ_{\mathsf{TO}}$ corresponds to the following lexicographic concatenation of orderings:

$$s \succ_{\mathsf{TO}} t \iff \begin{array}{l} \bullet\ \mathcal{H}ead(s) \succ \mathcal{H}ead(t) \quad \text{or} \\ \bullet\ \mathcal{H}ead(s) \sim \mathcal{H}ead(t) \ \wedge\ \mathcal{P}ath(\mathcal{A}rgs(s)) \overset{\textcircled{m}}{\succ}_{\mathsf{PO}} \mathcal{P}ath(\mathcal{A}rgs(t)). \end{array}$$

The equivalence relations $\sim_{\mathsf{PSO}}$ and $\sim_{\mathsf{TO}}$ are identical: $s \sim_{\mathsf{PSO}} t \iff s \sim_{\mathsf{TO}} t \iff s \approx t$. Finally, $\sim_{\mathsf{PO}} \equiv \overset{\textcircled{m}}{\sim}_{\mathsf{TO}}\langle \mathcal{S}et\rangle$ stands for the ordering scheme

$$[s_1,\ldots,s_m] \sim_{\mathsf{PO}} [t_1,\ldots,t_n] \iff \{s_1,\ldots,s_m\} \overset{\textcircled{m}}{\sim}_{\mathsf{TO}} \{t_1,\ldots,t_n\}.$$

The concept behind the definition of the PSO is that operators have more 'weight' if they
A.8 occur at the top of a more complex subterm[3]. For example, $\star$ in $x \star (y+z)$ would be weighted
A.13 more highly than either $\star$ in $(x \star y) + (x \star z)$. This characteristic is closely related to the
A.14 theoretical fact that a term t of a path P is the largest term (w.r.t. $\succ_{\mathsf{TO}}$) if its leading
A.15 A.20 operator is maximal (w.r.t. any other term in P) according to the precedence, and its size
A.21 is also maximal. The proof of this assertion mainly depends on the precondition of $\succ$ being
A.28 total. However, it is also possible to use a partial precedence (see [Rusinowitch, 1987]).

---

[1] conjectured in [Rusinowitch, 1987] and proved by the author in his masters thesis

[2] The suborderings $\succ_{\mathsf{PO}}$ and $\succ_{\mathsf{TO}}$ are abbreviations for Path Ordering (i.e. ordering on paths) and Term Ordering (i.e. ordering on terms).

[3] see also (ii) on page 47

3.2 Remark

The difference between the original PSO of [Plaisted, 1978a] and the variant given in [Rusinowitch, 1987] is the following one: Plaisted uses subsequences of paths whereas Rusinowitch considers paths as sets. For example, let $t = h(a+b)$. Then the original PSO constructs the sets $\{t.a{+}b.a\,,\, t.a{+}b\,,\, t.a\,,\, a{+}b.a\,,\, t\,,\, a{+}b\,,\, a\}$ and $\{t.a{+}b.b\,,\, t.a{+}b\,,\, t.b\,,\, a{+}b.b\,,\, t\,,\, a{+}b\,,\, b\}$ whereas the modified PSO takes the sets $\{t\,,\, a{+}b\,,\, a\}$ and $\{t\,,\, a{+}b\,,\, b\}$. In most cases, the use of subsequences for comparing paths of subterms yields redundant computations. Considering paths as sets, superfluous information is excluded, simultaneously keeping the necessary data (the term at the beginning of each subsequence is contained in the set and the other terms are treated recursively w.r.t. the definition of the PSO).

Like Plaisted's PSO, Dershowitz's *recursive path ordering* RPO was developed in early 1978 (see [Dershowitz, 1979b]). In 1980, Kamin and Lévy have extended the RPO by integrating a status function ([Kamin and Lévy, 1980]). The definition of this ordering (called RPOS) will be presented.

The RPO is a special case of the RPOS, each operator of the latter having status ⓜ.

Idea of Comparison *The method of comparing two terms w.r.t. the* RPOS *depends on the leading operators. The relationship between these operators w.r.t.* $\succ$ *and the status* τ *is responsible for decreasing one (or both) of the terms in the recursive definition of the* RPOS*. If one of the terms gets 'empty' (i.e. totally decreased), then the other one is greater.*

A.8
A.13
A.14
A.20
A.21
A.28

3.3 Definition (Recursive Path Ordering with Status, [Dershowitz, 1979b], [Kamin and Lévy, 1980])

Let $\succsim$ *be a quasi-precedence and* τ *be a status function. Then the* recursive path ordering RPOS with status *is defined as* A.18

$$\begin{aligned} \succ_{\mathsf{RPOS}} &\equiv \{\succsim^{\textcircled{m}}_{\mathsf{RPOS}}\langle \mathcal{A}rgs, \mathcal{I}d\rangle \;,\; \succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle \mathcal{I}d, \mathcal{A}rgs\rangle \;\&\; (\succ, \succ^{\tau}_{\mathsf{RPOS}})\} \\ \sim_{\mathsf{RPOS}} &\equiv \approx\langle \mathcal{I}d\rangle \end{aligned} \qquad \diamond$$

3.4 Remark

Usually, the RPOS is described by quantifiers (e.g. $\mathcal{H}ead(s) \succ \mathcal{H}ead(t) \wedge \forall t_i \in \mathcal{A}rgs(t)$: $s \succ_{\mathsf{RPOS}} t_i$) instead of multisets (e.g. $\mathcal{H}ead(s) \succ \mathcal{H}ead(t) \wedge \{\!\{\, s \,\}\!\} \succ^{\textcircled{m}}_{\mathsf{RPOS}} \mathcal{A}rgs(t)$). However, it is much easier to represent the multiset terms with our formalism of section 2.2 than the quantified expressions.

From a practical point of view, the condition '$\succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle \mathcal{I}d, \mathcal{A}rgs\rangle$' is only needed if the status (of the leading operators) is not equal to ⓜ (note that $\succ^{\textcircled{m}}_{\mathsf{RPOS}}$ implies this condition because of the subterm property of the RPOS). The necessity of this condition, whenever a lexicographic status is given[4], can be illustrated with the following example. Let $s = f(h(x), x)$ and $t = f(x, f(h(x), x))$ be terms. Then $(h(x), x) \succ^{\textcircled{l}}_{\mathsf{RPOS}} (x, s)$ since $h(x) \succ_{\mathsf{RPOS}} x$. However, s may not be greater than t since the rule $s \to t$ is non-terminating.

---

[4] Because of this condition, in [Kamin and Lévy, 1980] the lexicographic part of the RPOS is called *bounded* lexicographic path ordering.

The RPOS (and the RPO) addresses a problem posed by Lévy. We quote from [Lévy, 1980]:

> Consider a finite rooted oriented tree whose nodes are labeled by arbitrary positive integers. On such a tree we define an operation of expansion as follows: Let v be any node, other than a leaf, whose label l is greater than 1. Let $T_1, T_2, \ldots, T_s$ $(s \geq 1)$ be the subtrees hanging from v (in any order). We can now replace v by an arbitrary nonnull but finite rooted oriented tree. The nonleaf nodes of this new tree must be labeled with any positive integers strictly less than l, and at each of its leaves we must attach a copy of an arbitrary one of the T_i. Show that, starting from any finite tree, no infinite sequence of expansions is possible.

A.18
A.26
A.27
A.31
A.32
A.35
A.36

According to the problem of Lévy, the comparison w.r.t. the RPOS is based on the following idea: A term is decreased by replacing a subterm with any number of smaller terms which are connected by any structure of operators smaller (w.r.t. $\succ$) than the leading operator of the replaced subterm (see [Dershowitz, 1987]).

$\boxed{3.5}$ Remark

The multiset ordering of [Dershowitz and Manna, 1979] and the simple path ordering SPO of [Plaisted, 1978b] are special cases of the RPO, in which the multiset constructor $\{\!\{\ldots\}\!\}$ is greater than any of the other operators (see [Dershowitz, 1987], [Dershowitz, 1982]).
One way of extending the RPO(S) is to allow some component of a term $f(t_1, \ldots, t_n)$ to serve as the operator f. For example, we can consider t_k to be the operator and compare two terms by recursively comparing their kth operands, first: To prove that $s = \mathit{if}(\mathit{if}(x,y,z),u,v) \rightarrow \mathit{if}(x, \mathit{if}(y,u,v), \mathit{if}(z,u,v)) = t$ terminates, we consider the condition (i.e. the term $\mathit{if}(x,y,z)$ which is the first argument of the left-hand side) to be the operator. The term $\mathit{if}(x,y,z)$ of s is greater than the condition x of t. By definition of the RPO(S), we need to show that $\{\!\{\, s \,\}\!\} \succ^{\textcircled{m}}_{\mathsf{RPOS}} \{\!\{\, \mathit{if}(y,u,v), \mathit{if}(z,u,v) \,\}\!\}$. Again, $\mathit{if}(x,y,z)$ is greater than y and z, and thus $\{\!\{\, s \,\}\!\} \succ^{\textcircled{m}}_{\mathsf{RPO}} \{\!\{\, u, v \,\}\!\}$ must hold (which is obviously valid). This kind of extension of the RPO(S) represents a simplification ordering for the same reasons as the original definition does (see [Dershowitz, 1987], [Dershowitz, 1982]).
A more general technique than the previous one for extending the RPO changes terms by replacing their operators with the whole term itself ordered by some *other* well-founded ordering (instead of the RPO itself). This method is called *semantic path ordering* and was developed by Kamin and Lévy ([Kamin and Lévy, 1980], see also [Geser, 1992]). For a brief description of the semantic path ordering, see section 7.4.

A.9
A.10
A.15
A.16

Like the PSO, the path ordering KNSS (KNS with status) is an ordering which compares terms using their paths. It has been devised by Kapur, Narendran and Sivakumar. They have implemented the RPO within their rewrite rule laboratory RRL and have found it weak in handling terms which should intuitively be comparable. The KNSS is a consequence of these experiments and it extends the RPOS.

Idea of Comparison *The first part of the comparison of two terms w.r.t. the* KNSS *is identical to that w.r.t. the* PSO*: Generating the paths and comparing two paths by comparing their elements as multisets w.r.t. an ordering on terms. The ordering on terms compares the leading operators and if they are equal, different substructures (arguments or paths of arguments) will be compared depending on the status of the leading operators.*

$\boxed{3.6}$ **Definition (Path Ordering with Status of [Kapur** *et al.*, **1985])** *Let* $\succsim$ *be a quasi-precedence and* τ *be a status function. Then the* path ordering KNSS with status *is defined as* |A.10

$$
\begin{aligned}
\succ_{\mathsf{KNSS}} &\equiv \overset{\textcircled{m}}{\succ}_{\mathsf{PO}}\langle \mathcal{P}ath\rangle \\
\succ_{\mathsf{PO}} &\equiv \overset{\textcircled{n}}{\succ}_{\mathsf{TO}}\langle \mathcal{S}et\rangle \\
\succ_{\mathsf{TO}} &\equiv (\succ, \{\tau = \textcircled{m} \;\&\; (\succ_{\mathsf{PO}}\langle \mathcal{S}ub\rangle, \overset{\textcircled{m}}{\succ}_{\mathsf{PO}}\langle \mathcal{P}ath \circ \mathcal{A}rgs\rangle, \succ_{\mathsf{PO}}\langle \mathcal{S}up\rangle), \\
&\qquad\quad \tau \neq \textcircled{m} \;\&\; (\overset{\tau}{\succ}_{\mathsf{KNSS}}, \succ_{\mathsf{PO}}\langle \mathcal{S}up\rangle)\}) \\
\sim_{\mathsf{KNSS}} &\equiv \approx\langle \mathcal{I}d\rangle \\
\sim_{\mathsf{PO}} &\equiv \overset{\textcircled{m}}{\sim}_{\mathsf{TO}}\langle \mathcal{S}et\rangle \\
\sim_{\mathsf{TO}} &\equiv \approx\langle \mathcal{I}d\rangle
\end{aligned}
$$

where $\textcircled{n}$ *is a new type of status ('n' stands for not removing equivalent elements from different multisets) defined by*[5)] $A \overset{\textcircled{n}}{\succ}_{\mathsf{T}} B \iff \forall t \in B\colon \exists s \in A\colon s \succ_{\mathsf{T}} t.$ ◇

Path orderings were originally conceived by Plaisted and Dershowitz. Plaisted's work on the PSO *in September 1978 ([Plaisted, 1978a]) and on the* simple path ordering *(*SPO*) in July 1978 ([Plaisted, 1978b]) as well as Dershowitz's work on the* RPO *are the first ones about this class of orderings. The* SPO *maps terms into multisets of sequences of operators (which are simple paths) and compares them w.r.t. an ordering on monadic terms. This ordering lexicographically compares two terms from left to right (i.e. from the root symbol to a leaf). In [Rusinowitch, 1987], a more efficient version of the* PSO *is given, based on the idea described in remark 3.2.*

The well-known and very often used recursive path ordering RPO *of Dershowitz, presented in October 1979 ([Dershowitz, 1979b], and in an internal report in August 1979), is essentially influenced by the work of [Dershowitz and Manna, 1979] on multiset orderings. In [Klop, 1987], an alternative definition of the* RPO *based on a reduction procedure is given. The integration of a status into the* RPO *was developed by Kamin & Lévy (*RPOS*, [Kamin and Lévy, 1980]). Their unpublished (!) letter to Dershowitz, written at the beginning of February 1980, also contains another generalization of the* RPO*, the so-called semantic path ordering (which is not a simplification ordering, in general). In [Sakai, 1984], the lexicographic path ordering (i.e. the* RPOS *such that all operators have status* Ⓛ*) of [Kamin and Lévy, 1980] is also described.*

An improvement of the RPO(S) *was presented by Kapur, Narendran and Sivakumar in 1985 ([Kapur* et al.*, 1985]). The* KNS *was motivated by their unsuccessful attempts to orient equations arising in a simple axiomatic characterization of free groups using the* RPO *within the rewrite rule laboratory RRL (for a description of RRL, see, e.g., [Kapur and Zhang, 1989]). Based on the* KNS*, Rusinowitch developed a decomposition ordering (the* IRD *which is an extension of the* RDO*, see the following pages for a discussion on decomposition orderings)*

[5)] see remark 3.10

equivalent to the KNS*, but more efficient: The ordering* $\succ_{\mathsf{TO}}$ *of the* KNS(S) *will require (irrespective of the status) the comparison of the sequence of superterms (i.e. the comparison w.r.t.* $\succ_{\mathsf{PO}}\langle\mathcal{S}up\rangle$*) if the suffixes of the corresponding paths are equal. Removing the common suffix of both paths, this test can be omitted ([Rusinowitch, 1987], see remark 3.10).*

3.7 Remark

The KNSS can possibly be extended by way of allowing terms from *different* paths to take care of terms along *one* path. Thus, we could remove the restriction that a path must be covered by one *sole* path. Let s and t be the following terms: $s = 2 \star 8$ and $t = 4^2$ with $2 \succ 4$ and $8 \succ {}^2$. The term 8 of the path $[s, 8]$ takes care of the term t of $[t, 4]$, and $2 \in [s, 2]$ is greater than $4 \in [t, 4]$. Note that this new scheme has not been clearly defined (see [Kapur *et al.*, 1985]).

A.18
A.26
A.27
A.31
A.32
A.35
A.36

As noted in remark 3.4, the use of the RPOS with lexicographic status needs to guarantee the condition '$\succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{I}d, \mathcal{A}rgs\rangle$' in order to ensure the well-foundedness. Applying this property to the KNSS, one would have to add the condition '$\succ^{\textcircled{m}}_{\mathsf{KNSS}}\langle\mathcal{I}d, \mathcal{A}rgs\rangle$' to the condition '$\succ^{\mathcal{T}}_{\mathsf{KNSS}}$' of the case $\tau \neq \textcircled{m}$. For the KNSS (as well as for all other path and decomposition orderings of this chapter), this is *not* necessary since the required property holds because of the division of terms into paths: *Each* subterm of the smaller term must be covered by a subterm of the greater term.

3.8 Theorem

- ▶ *The* PSO *is a σ-stable simplification ordering ([Plaisted, 1978a]).*
- ▶ *The* RPOS *is a σ-stable simplification ordering ([Dershowitz, 1979b], [Kamin and Lévy, 1980]).*
- ▶ *The* KNS *is a σ-stable simplification ordering ([Kapur* et al.*, 1985]).* ◇

In [Kapur *et al.*, 1985], the definition of the KNSS is appended to that of the KNS but not proved to be a σ-stable simplification ordering. We will do this in section 3.2. We now want to indicate a problem concerning the use of a status function within the definition of the RPOS and the KNSS.

3.9 Remark

There exist two different versions of the RPOS in the literature:

- ▶ The RPOS of definition 3.3 $\{\succsim^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{A}rgs, \mathcal{I}d\rangle\ ,\ \succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{I}d, \mathcal{A}rgs\rangle\ \&\ (\succ, \succ^{\mathcal{T}}_{\mathsf{RPOS}})\}$ allows to test the first alternative '$\succsim^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{A}rgs, \mathcal{I}d\rangle$' irrespective of the leading operators. For example, if $\mathcal{H}ead(s) \succ \mathcal{H}ead(t)$ holds, the first *or* the second condition could be checked (when comparing two terms s and t).

- ▶ The definition $\{\not\succsim\langle\mathcal{H}ead\rangle\ \&\ \succsim^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{A}rgs, \mathcal{I}d\rangle\ ,\ \succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{I}d, \mathcal{A}rgs\rangle\ \&\ (\succ, \succ^{\mathcal{T}}_{\mathsf{RPOS}})\}$ contains three *disjoint* conditions:

$$\textbf{(i)}\ \not\succsim\langle\mathcal{H}ead\rangle\ \&\ \succsim^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{A}rgs, \mathcal{I}d\rangle \qquad \textbf{(ii)}\ \succ\ \&\ \succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{I}d, \mathcal{A}rgs\rangle$$

$$\textbf{(iii)}\ \sim\langle\mathcal{H}ead\rangle\ \&\ \succ^{\mathcal{T}}_{\mathsf{RPOS}}\ \&\ \succ^{\textcircled{m}}_{\mathsf{RPOS}}\langle\mathcal{I}d, \mathcal{A}rgs\rangle$$

In contrast to the RPO without status for which both versions are equivalent, the powers of these approaches w.r.t. the RPOS differ: The first one is a proper extension of the latter one.

Moreover, the first one is a simplification ordering whereas the latter one does not preserve the subterm property since, for example, $(x \star y) \star z \not\succ_{\mathsf{RPOS}} x \star y$ if $\tau(\star) = \textcircled{r}$.

3.10 Remark

In [Kapur *et al.*, 1985], it has been stated that attaching equivalent paths to the right does not alter the ordering relation, i.e. $P \succ_{\mathsf{PO}} Q$ implies $P{\cdot}P' \succ_{\mathsf{PO}} Q{\cdot}P'$. In [Rusinowitch, 1987], the converse of this fact is stated: $P{\cdot}P' \succ_{\mathsf{PO}} Q{\cdot}P'$ implies $P \succ_{\mathsf{PO}} Q$. Taking both results into account, one can replace the status $\textcircled{n}$ by $\textcircled{m}$ in the definition of $\succ_{\mathsf{PO}}$. This way, the power of the KNSS with multiset status remains unchanged. However, for the sake of the KNSS with lexicographic status being a simplification ordering, one *has* to remove identical suffixes of paths to be compared (i.e. the status $\textcircled{m}$ is necessary for $\succ_{\mathsf{PO}}$). For example (see also remark 3.9), the comparison of the terms $s = (x \star y) \star z$ and $t = x \star y$ with $\tau(\star) = \textcircled{r}$ requires the paths $P = [s, t, x]$ and $Q = [t, x]$ to be comparable. Note that $P \succ_{\mathsf{PO}} Q$ holds only if $[t, x]$ is removed from P and Q (since $\tau(\star) = \textcircled{r}$ and thus, z must be compared with y).

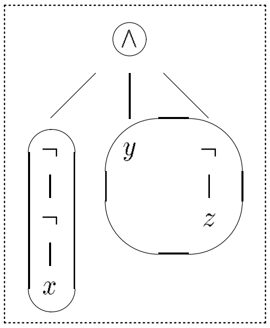
3.1.2 Decomposition Orderings

This subsection provides a summary of the history of (recursive) decomposition orderings. It includes the original ordering of [Jouannaud *et al.*, 1982] in the slightly modified and simplified representation given in [Rusinowitch, 1987] (possessing the same power as that of [Jouannaud *et al.*, 1982]). In this thesis, beginning with the following subsection, another notation of decomposition orderings will be used. Thus, the notation appearing in this subsection is non-recurring. This is why we did not give the notational scenario of this subsection in the first chapter.

As the KNSS, the recursive decomposition ordering (RDO, for short) has been derived from the RPO. One of the important differences from the RPO is the fact that, according to the RDO, a comparison of two incomparable operators need *not* be performed. The reason for this property is that the RDO does not use such a strict structure of terms as the RPO. More precisely, this property of the RDO could conceivably lead to a term to be represented by the *multiset* of all of its subterms and the comparison of two terms is done by comparing the corresponding multisets. Note that the path orderings PSO and KNSS seem to be applied to a more restricted structure of terms since paths are *ordered* sequences of terms. However, the definitions of these two orderings are processing paths as (multi-) sets of terms.

The RDO is based on a preliminary analysis of terms, called *decomposition*, on which comparisons are recursively performed. The decomposition divides a term into three parts:

- the leading operator ($\wedge$),
- any selected argument ($\neg\neg x$) and
- the remaining arguments ($\{y\,,\,\neg z\}$).



In order to describe the decomposition orderings, we use the following formalism.

$\boxed{\textbf{3.11}}$ **Definition (Decomposition)** *Let* $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ *and* $u, v, w, p \in \mathcal{P}os(t)$.

- ▶ **Direct Successor Position:** *For* $u \geq v$, *we define* $\mathcal{S}uc(v,u) = v.i$ *if* $u = v.i.w$ *with* $i \in \mathbb{N}$, *whereas* $\mathcal{S}uc(v,v) = \infty$. *By convention,* $t|_\infty$ *is the empty term.*
- ▶ **Elementary Decomposition:** *The* elementary decomposition $D_u(t,p)$ *of* t *at a position* p *according to a terminal position* $u \in \mathcal{TP}os(t)$, *with* $u \geq p$, *is the triple*
$$D_u(t,p) = \langle\!\langle \mathcal{H}ead(t|_p); t'; T \rangle\!\rangle$$
where $t' = t|_{\mathcal{S}uc(p,u)}$ *and* $T = \mathcal{A}rgs(t|_p) \setminus \{t'\}$.
- ▶ **Path-Decomposition:** *The extension of an elementary decomposition to the set of all prefixes of* $u \in \mathcal{TP}os(t)$ *is called* path-decomposition *and is defined as*
$$D_u(t) = \{D_u(t,p) \mid D_u(t,p) \text{ is an elementary decomposition, } u \geq p\}.$$
- ▶ **Decomposition:** *The extension of a path-decomposition to a multiset w.r.t. all terminal positions of* t *is called* decomposition *and defined as*
$$D(t) = \{\!\{\, D_u(t) \mid D_u(t) \text{ is a path-decomposition, } u \in \mathcal{TP}os(t) \,\}\!\}.$$
- ▶ **Right Quotient Position:** $u/v = w$ *if* $u = v.w$ ◇

$\boxed{\textbf{3.12}}$ **Example** (i) $111 > 1$. (ii) $\mathcal{S}uc(1{,}111) = 11$. *Let* $\mathcal{F} = \{\neg, \wedge\}$ *and* $t = \wedge(\neg\neg x, y, \neg z)$, *the term shown above as a tree.* (iii) *The elementary decomposition* $D_{111}(t,\Lambda)$ *has the form* $\langle\!\langle \wedge; \neg\neg x; \{\!\{\, y, \neg z \,\}\!\} \rangle\!\rangle$. $D_2(t,\Lambda) = \langle\!\langle \wedge; y; \{\!\{\, \neg\neg x, \neg z \,\}\!\} \rangle\!\rangle$ *and* $D_{31}(t,31) = \langle\!\langle z; \ ; \emptyset \rangle\!\rangle$. (iv) $D_2(t) = \{D_2(t,\Lambda), \langle\!\langle y; \ ; \emptyset \rangle\!\rangle\}$. (v) $D(t) = \{\!\{\, D_{111}(t), D_2(t), D_{31}(t) \,\}\!\}$. (vi) $311/3 = 11$. ◇

Idea of Comparison *A term* s *is greater than a term* t *(w.r.t. the* RDO*) if the decomposition of* s *is greater than the decomposition of* t. *The ordering* $(\overset{\textcircled{m}}{\succ}_{\mathsf{D}})^{\textcircled{m}}$ *on these multisets is an extension of the basic ordering* $\succ_{\mathsf{D}}$ *on elementary decompositions to multisets of multisets. The ordering* $\succ_{\mathsf{D}}$ *compares two elementary decompositions by lexicographically comparing their (three) components from left to right.*

$\boxed{\textbf{3.13}}$ **Definition (Recursive Decomposition Ordering, [Jouannaud *et al.*, 1982], [Rusinowitch, 1987])** *Let* $\succ$ *be a precedence. Then the* recursive decomposition ordering RDO *is defined as*

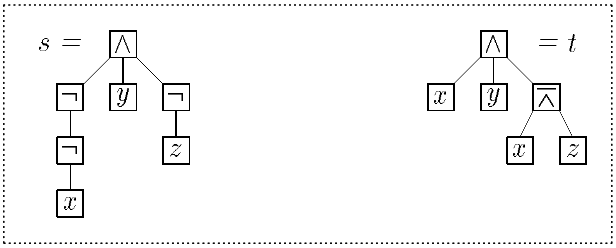
$$s \succ_{\mathsf{RDO}} t \iff D(s)\, (\overset{\textcircled{m}}{\succ}_{\mathsf{D}})^{\textcircled{m}}\, D(t)$$

$$\text{with } D_u(s,p) = \langle\!\langle f; s'; S \rangle\!\rangle \succ_{\mathsf{D}} \langle\!\langle g; t'; T \rangle\!\rangle = D_v(t,q) \iff$$

- ▶ $f \succ g$ *or*
- ▶ $f = g \quad \wedge \quad D_{u/\mathcal{S}uc(p,u)}(s') \overset{\textcircled{m}}{\succ}_{\mathsf{D}} D_{v/\mathcal{S}uc(q,v)}(t')$ *or*
- ▶ $f = g \quad \wedge \quad D_{u/\mathcal{S}uc(p,u)}(s') = D_{v/\mathcal{S}uc(q,v)}(t') \quad \wedge \quad S \overset{\textcircled{m}}{\succ}_{\mathsf{RDO}} T$ ◇

Terms are equal w.r.t. the RDO if they are permutatively congruent. We will illustrate this slightly complicated definition by an example.

$\boxed{3.14}$ **Example** *Let* $s = \wedge(\neg\neg x, y, \neg z)$ *and* $t = \wedge(x, y, x \mathbin{\overline{\wedge}} z)$ *be terms constructed by the Boolean operators* $\neg, \wedge, \overline{\wedge}$. *Moreover, suppose the precedence asserts* $\neg \succ \overline{\wedge}$. *To compare* s *and* t, *we first have to generate their decompositions:*



$$\begin{aligned}
D(s) &= \{\!\{\, D_{111}(s)\,,\, D_2(s)\,,\, D_{31}(s) \,\}\!\} \\
D_{111}(s) &= \{D_{111}(s,\Lambda)\,,\, D_{111}(s,1)\,,\, D_{111}(s,11)\,,\, D_{111}(s,111)\} \\
D_2(s) &= \{D_2(s,\Lambda)\,,\, D_2(s,2)\} \\
D_{31}(s) &= \{D_{31}(s,\Lambda)\,,\, D_{31}(s,3)\,,\, D_{31}(s,31)\}
\end{aligned}$$

$$\begin{aligned}
D(t) &= \{\!\{\, D_1(t)\,,\, D_2(t)\,,\, D_{31}(t)\,,\, D_{32}(t) \,\}\!\} \\
D_1(t) &= \{D_1(t,\Lambda)\,,\, D_1(t,1)\} \\
D_2(t) &= \{D_2(t,\Lambda)\,,\, D_2(t,2)\} \\
D_{31}(t) &= \{D_{31}(t,\Lambda)\,,\, D_{31}(t,3)\,,\, D_{31}(t,31)\} \\
D_{32}(t) &= \{D_{32}(t,\Lambda)\,,\, D_{32}(t,3)\,,\, D_{32}(t,32)\}
\end{aligned}$$

We want to prove that s *is greater than* t *w.r.t. the* RDO. *Consequently, in accordance with the definition of the* RDO, *for every* $D_v(t)$ *we have to find a* $D_u(s)$ *which is greater w.r.t.* $\succ_{\mathsf{D}}^{\textcircled{m}}$. *By reason of the* σ*-stability, our search for* $D_u(s)$ *is restricted by the leaves* $s|_u$ *and* $t|_v$, *respectively. We can verify the following relations:*

- ▶ $D_{111}(s) \succ_{\mathsf{D}}^{\textcircled{m}} D_1(t)$:
  - *(i)* $D_{111}(s,111) = D_1(t,1) = \langle\!\langle x;\ ;\emptyset\rangle\!\rangle$
  - *(ii)* $D_{111}(s,\Lambda) = \langle\!\langle \wedge; \neg\neg x; \{\!\{\, y, \neg z \,\}\!\}\rangle\!\rangle \succ_{\mathsf{D}} \langle\!\langle \wedge; x; \{\!\{\, y, x \mathbin{\overline{\wedge}} z \,\}\!\}\rangle\!\rangle = D_1(t,\Lambda)$
    because $D_{11}(\neg\neg x) \succ_{\mathsf{D}}^{\textcircled{m}} D_\Lambda(x)$ *since* $D_\Lambda(x) \subseteq D_{11}(\neg\neg x)$

- ▶ $D_{111}(s) \succ_{\mathsf{D}}^{\textcircled{m}} D_{31}(t)$:
  - *(i)* $D_{111}(s,111) = D_{31}(t,31) = \langle\!\langle x;\ ;\emptyset\rangle\!\rangle$
  - *(ii)* $D_{111}(s,11) = \langle\!\langle \neg; x; \emptyset\rangle\!\rangle \succ_{\mathsf{D}} \langle\!\langle \overline{\wedge}; x; \{\!\{\, z \,\}\!\}\rangle\!\rangle = D_{31}(t,3)$
    because $\neg \succ \overline{\wedge}$
  - *(iii)* $D_{111}(s,\Lambda) = \langle\!\langle \wedge; \neg\neg x; \{\!\{\, y, \neg z \,\}\!\}\rangle\!\rangle \succ_{\mathsf{D}} \langle\!\langle \wedge; x \mathbin{\overline{\wedge}} z; \{\!\{\, x, y \,\}\!\}\rangle\!\rangle = D_{31}(t,\Lambda)$
    because $D_{11}(\neg\neg x) \succ_{\mathsf{D}}^{\textcircled{m}} D_1(x \mathbin{\overline{\wedge}} z)$ *since* $\neg \succ \overline{\wedge}$

- ▶ $D_{31}(s) \succ_{\mathsf{D}}^{\textcircled{m}} D_{32}(t)$:
  This statement can straightforwardly be proved with the considerations of the previous one.

- *What about $D_2(t)$?*
  The only decomposition applicable is $D_2(s)$, since $s|_2$ is the only leaf consisting of the variable y.

  (i) $D_2(s,2) = D_2(t,2) = \langle\!\langle y;\ ;\emptyset\rangle\!\rangle$

  (ii) $D_2(s,\Lambda) = \langle\!\langle \wedge; y; \{\!\{\neg\neg x, \neg z\}\!\}\rangle\!\rangle \not\succ_{\mathsf{D}} \langle\!\langle \wedge; y; \{\!\{x, x \barwedge z\}\!\}\rangle\!\rangle = D_2(t,\Lambda)$
  because neither $\neg\neg x$ nor $\neg z$ can bound $x \barwedge z$ and they cannot help each other to do that. Instead of comparing the multisets of subterms *consider the* multiset sums of decompositions. *Let $s_1 = \neg\neg x, s_2 = \neg z$ and $t_1 = x$, $t_2 = x \barwedge z$. Consequently, $S = \{s_1, s_2\}$ and $T = \{t_1, t_2\}$. Let us now compare the multisets*
  $D(s_1) \uplus D(s_2) = \{\!\{\, \{D_{11}(s_1,\Lambda)\,, D_{11}(s_1,1)\,, D_{11}(s_1,11)\}\,, \{D_1(s_2,\Lambda)\,, D_1(s_2,1)\}\,\}\!\}$
  and
  $\{\!\{\, \{D_\Lambda(t_1,\Lambda)\}\,, \{D_1(t_2,\Lambda)\,, D_1(t_2,1)\}\,, \{D_2(t_2,\Lambda)\,, D_2(t_2,2)\}\,\}\!\} = D(t_1) \uplus D(t_2)$.
  Now, $D(s_1) \uplus D(s_2)$ $(\overset{\textcircled{m}}{\succ}_{\mathsf{D}})^{\textcircled{m}}$ $D(t_1) \uplus D(t_2)$ follows from the other comparisons already considered. ◇

Summing up, we have seen that the comparison of the third part of an elementary decomposition (i.e. $S \overset{\textcircled{m}}{\succ}_{\mathsf{RDO}} T$), the multisets of subterms, is not ideal. A more successful test is to compare the *multiset sums of the decompositions of these subterms.* With this slight change, a real extension of the RDO (IRD, for short) can be defined[6]. The IRD was developed by Rusinowitch in [Rusinowitch, 1987]. Concluding example 3.14, the following relations hold: $s \not\succ_{\mathsf{RDO}} t$ but $s \succ_{\mathsf{IRD}} t$.

Idea of Comparison *The method of comparing terms w.r.t. the* IRD *is identical to that of the* RDO *with the exception of the last condition: The* IRD *compares multisets of* decompositions *of subterms instead of multisets of subterms.*

$\boxed{3.15}$ **Definition (Improved Recursive Decomposition Ordering, [Rusinowitch, 1987])** *Let $\succ$ be a precedence. Then the* improved recursive decomposition ordering IRD *is defined as*

$s \succ_{\mathsf{IRD}} t \iff D(s)\,(\overset{\textcircled{m}}{\succ}_{\mathsf{D}})^{\textcircled{m}}\,D(t)$
with $D_u(s,p) = \langle\!\langle f; s'; S\rangle\!\rangle \succ_{\mathsf{D}} \langle\!\langle g; t'; T\rangle\!\rangle = D_v(t,q) \iff$

- $f \succ g$ *or*
- $f = g \;\wedge\; D_{u/\mathcal{S}uc(p,u)}(s') \overset{\textcircled{m}}{\succ}_{\mathsf{D}} D_{v/\mathcal{S}uc(q,v)}(t')$ *or*
- $f = g \;\wedge\; D_{u/\mathcal{S}uc(p,u)}(s') = D_{v/\mathcal{S}uc(q,v)}(t') \;\wedge\; \biguplus_{s''\in S} D(s'')\; (\overset{\textcircled{m}}{\succ}_{\mathsf{D}})^{\textcircled{m}}\; \biguplus_{t''\in T} D(t'')$ ◇

In analogy with the RDO, terms are equal w.r.t. the IRD if they are permutatively congruent.

$\boxed{3.16}$ **Theorem**

- *The* RDO *is a σ-stable simplification ordering ([Jouannaud* et al.*, 1982]).*
- *The* IRD *is a σ-stable simplification ordering ([Rusinowitch, 1987]).* ◇

---

[6] For a formal comparison of the RDO and the IRD, see page 54.

3.17 Remark

The *original* decomposition ordering works on decomposition-*quadruples* instead of *triples*. An extra operator $\Box$ is used to mark the place where an elementary decomposition is applied: $d_u(t,p) = D_u(t,p) \cdot \langle\!\langle t[\Box]_p \rangle\!\rangle$. This additional fourth component is called *context*. Rusinowitch proved that it is redundant. The definitions of the RDO and the IRD presented here take this fact into account.

A more efficient RDO without any loss of power is presented in [Reinig and Jouannaud, 1981] and [Jouannaud *et al.*, 1982]. Originally, two terms s and t are compared w.r.t. the RDO by means of constructing the *entire* $D(s)$ and the *entire* $D(t)$. It can be proved that selected (i.e. maximal) subsets of path-decompositions are sufficient. Moreover, a path-decomposition can be restricted to a part of itself. More precisely, one does not have to consider (i) all terminal positions as well as (ii) all prefixes of a terminal position:

(i) The maximal subset of terminal positions of a term $t = f(t_1,\ldots,t_n)$ will be constructed by the following procedure: Mark the root of t. Mark the root of t_i if there exist neither a t_j $(j \neq i)$ with $t_j \succ_{\mathsf{T}} t_i$ ($\succ_{\mathsf{T}}$ is any reduction ordering) nor a marked t_j with $t_i \sim_{\mathsf{T}} t_j$. Apply this process recursively to all t_i which are marked. The maximal subset of terminal positions consists of all positions corresponding to marked leaves.

(ii) A prefix u of a terminal position has to be considered if the corresponding symbol at u is not smaller than or equal to (w.r.t. the precedence $\succ$) the direct preceding symbol and not smaller than the direct succeeding symbol (along the path from the root to the specified leaf).

There exist various publications on the theory of recursive decomposition orderings. Recursive decomposition orderings were originally conceived by Lescanne. His idea has also been taken up and realized by Jouannaud and Reinig. In figure 3.1, a list of references together with a brief description of their contents is given and their dependencies are illustrated in figure 3.2.

3.1.3 Well-Known Decomposition Orderings Revisited

In this subsection, we establish the close connection between the path orderings PSO, KNSS and the decomposition orderings defined in the former subsection. This will be done by modifying the structure of decompositions (as described in subsection 3.1.2). Obviously, this alteration results in a change of the definitions of the decomposition orderings. These modifications are in accordance with the following principle:

- ▶ Simplify the data structure of (elementary) decompositions and shift the thereby lost information into the definition of the orderings.

First of all, an elementary decomposition $D_u(t,p)$ will be re-transformed into its corresponding term, i.e.

$$D_u(t,p) = \langle\!\langle f; t'; T \rangle\!\rangle \quad \Longrightarrow \quad t|_p. \qquad (*)$$

[Lescanne and Reinig, 1980]: Contains the principal idea of decomposition orderings which is based on a proof of the well-foundedness of the PSO.

[Lescanne, 1981a]: Describes an earlier version of the RDO on left-weighted terms in case of total precedences.

[Lescanne, 1981b]: Discusses an efficient implementation of the RPO on monadic terms. Contains the definition of the RDO on monadic terms.

[Reinig, 1981]: Describes the RDO (a generalized version of the 'RDO' in [Lescanne, 1981a] and [Lescanne, 1981b]) and proves its well-foundedness. Moreover, it shows that it is stronger than the RPO.

[Reinig and Jouannaud, 1981]: Contains the same results as [Reinig, 1981] in English and is an earlier version of [Jouannaud *et al.*, 1982].

[Lescanne, 1982]: An earlier version of the RDO (based on the concepts of [Lescanne, 1981a] and [Lescanne, 1981b]) is presented and proved to be equivalent to the RPO.

[Jouannaud *et al.*, 1982]: Contains the definition of the RDO and proves that it is stronger than the RPO.

[Lescanne, 1983b]: It is an earlier version of [Lescanne, 1984].

[Lescanne, 1984]: Presents the recursive decomposition ordering RDOS with status (see subsection 3.1.3).

[Lescanne, 1990]: Redefines the RDOS (see subsection 3.1.3) restricted to left-to-right status and proves some results on it (not proved in [Lescanne, 1984]). Furthermore, it contains proofs of some well-known totality results and the definition of the *closure ordering* (see chapter 6, historical remark on page 142).

Figure 3.1: History of decomposition orderings (Part I)

Each decomposition ordering (i.e. its adequate basic ordering on elementary decompositions) operates upon the triple $D_u(t,p)$ in a lexicographic way by applying specific orderings $\succ_1, \succ_2, \succ_3$ to the three components (possibly changed by the application of specific functions):

$$D_u(s,p) = \langle\!\langle C_1; C_2; C_3 \rangle\!\rangle \ \mathrel{\succ\!\!\!\sim}_{\mathsf{Dec}} \ \langle\!\langle C_1'; C_2'; C_3' \rangle\!\rangle = D_v(t,q) \quad \Longleftrightarrow$$

- ► $\rho_1(C_1) \mathrel{\succ\!\!\!\sim}_1 \rho_1'(C_1')$ or

- ► $\rho_1(C_1) \sim_1 \rho_1'(C_1') \quad \wedge \quad \rho_2(C_2) \mathrel{\succ\!\!\!\sim}_2 \rho_2'(C_2')$ or

- ► $\rho_1(C_1) \sim_1 \rho_1'(C_1') \quad \wedge \quad \rho_2(C_2) \sim_2 \rho_2'(C_2') \quad \wedge \quad \rho_3(C_3) \mathrel{\succ\!\!\!\sim}_3 \rho_3'(C_3')$

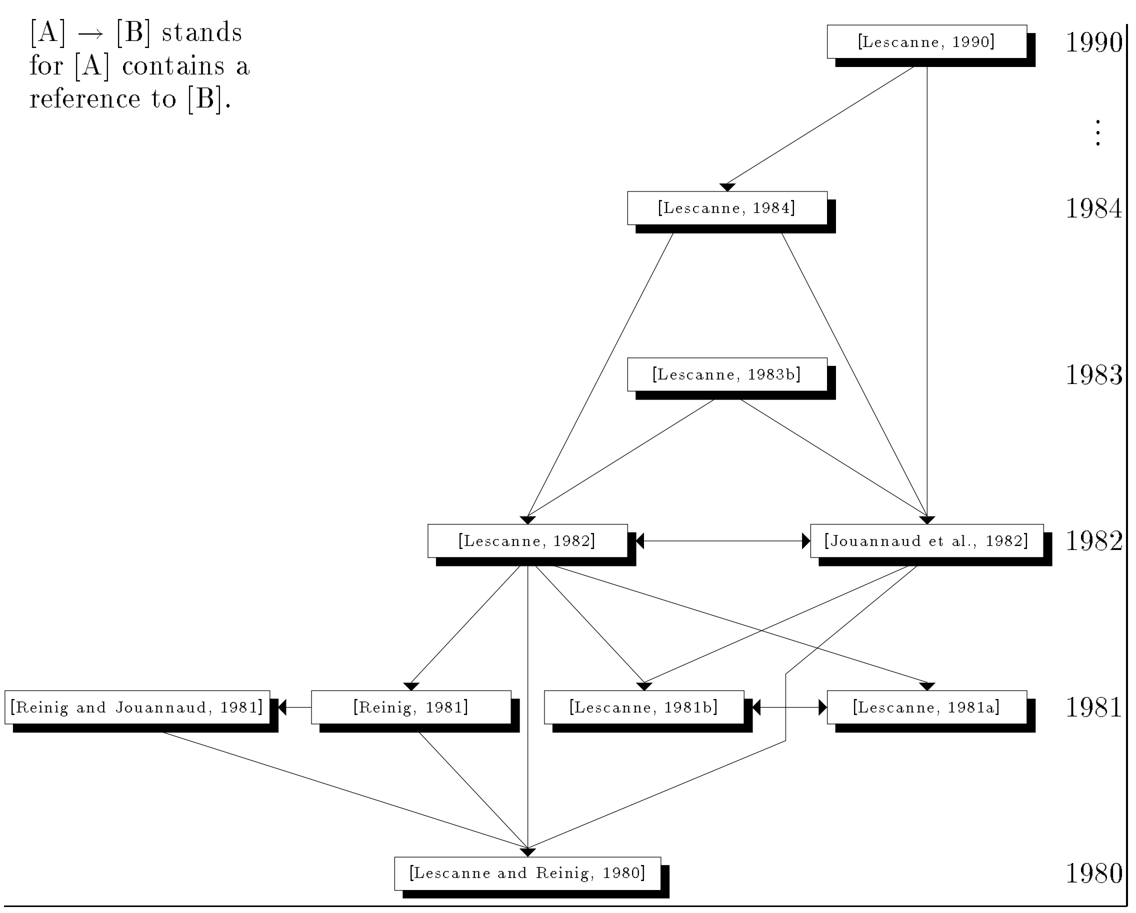


Figure 3.2: History of decomposition orderings (Part II)

such that $C_1 = \mathcal{H}ead(s|_p), C_2 = s|_{\mathcal{S}uc(p,u)}, C_3 = \mathcal{A}rgs(s|_p) \setminus \{C_2\}$ and $C'_1 = \mathcal{H}ead(t|_q), C'_2 = t|_{\mathcal{S}uc(q,v)}, C'_3 = \mathcal{A}rgs(t|_q) \setminus \{C'_2\}$. Now, the ordering $\succ_{\mathsf{Dec}}$ will be changed by integrating the lost information (because of the transformation described in $(*)$). Note that $\succ_{\mathsf{Dec}}$ is now applied to terms instead of elementary decompositions:

$s' \succ_{\mathsf{Dec}} t' \iff$

- $\rho_1(\mathcal{H}ead(s')) \succ_1 \rho'_1(\mathcal{H}ead(t'))$ or
- $\rho_1(\mathcal{H}ead(s')) \sim_1 \rho'_1(\mathcal{H}ead(t')) \quad \wedge \quad \rho_2(s|_{p/\mathcal{S}uc(p,u)}) \succ_2 \rho'_2(t|_{q/\mathcal{S}uc(q,v)})$ or
- $\rho_1(\mathcal{H}ead(s')) \sim_1 \rho'_1(\mathcal{H}ead(t')) \quad \wedge \quad \rho_2(s|_{p/\mathcal{S}uc(p,u)}) \sim_2 \rho'_2(t|_{q/\mathcal{S}uc(q,v)}) \quad \wedge \quad \rho_3(\mathcal{A}rgs(s') \setminus \{s|_{p/\mathcal{S}uc(p,u)}\}) \succ_3 \rho'_3(\mathcal{A}rgs(t') \setminus \{t|_{q/\mathcal{S}uc(q,v)}\})$

with $s' = s|_p$ and $t' = t|_q$.

With the help of this transformation, we are able to change the definitions of subsection 3.1.2 into equivalent ones using *elementary decompositions as terms* and (path-) *decompositions as structures over such terms* (as they are defined in subsection 1.5.2).

$\boxed{3.18}$ **Definition (Recursive Decomposition Ordering, [Jouannaud** *et al.*, **1982], [Rusinowitch, 1987])** *Let $\succ$ be a precedence. Then the* recursive decomposition order- |A.9

ing RDO *is defined as*$^{7)}$

A.10 A.15 A.16

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\succ_{\mathsf{RDO}} & \equiv & (\succ^{\textcircled{m}}_{\mathsf{D}})^{\textcircled{m}}\langle \mathcal{D}ec\rangle & \sim_{\mathsf{RDO}} & \equiv & \approx\langle \mathcal{I}d\rangle \\
\succ_{\mathsf{D}} & \equiv & (\succ\,,\, \succ^{\textcircled{m}}_{\mathsf{D}}\langle \mathcal{S}ub\rangle\,,\, \succ^{\textcircled{m}}_{\mathsf{RDO}}) & \sim_{\mathsf{D}} & \equiv & \approx\langle \mathcal{I}d\rangle
\end{array} \qquad \diamond$$

3.19 Remark

For compactness, we have simplified the definition of the RDO without changing its power. The last part of $\succ_{\mathsf{D}}$ in the original definition is of the form $\succ^{\textcircled{m}}_{\mathsf{RDO}}\langle \backslash \circ < \mathcal{A}rgs, \mathcal{M}ax \circ \mathcal{S}ub > \rangle$ (instead of $\succ^{\textcircled{m}}_{\mathsf{RDO}}$ which is identical to $\succ^{\textcircled{m}}_{\mathsf{RDO}}\langle \mathcal{A}rgs\rangle$) representing the term multiset composed of the arguments of the input term where one argument (specified by $\mathcal{M}ax \circ \mathcal{S}ub$) is removed. Note that \ stands for the multiset difference and $\mathcal{M}ax$ delivers the biggest term (w.r.t. the size) of a set of terms. The power of definition 3.18 is equal to that of the original definition since the arguments specified by $\mathcal{M}ax$ are compared in the second component of $\succ_{\mathsf{D}}$ and they must be equivalent whenever the third component of $\succ_{\mathsf{D}}$ is activated. Obviously, the original definition is more interesting from a practical point of view than the given one since it is more efficient.

In the remaining part of this chapter, we use the simplified version (described in the previous remark) in the definitions of all decomposition orderings.

A.15 A.16 **3.20 Definition (Improved Recursive Decomposition Ordering, [Rusinowitch, 1987])** *Let* $\succ$ *be a precedence. Then the* improved recursive decomposition ordering IRD *is defined as*

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\succ_{\mathsf{IRD}} & \equiv & (\succ^{\textcircled{m}}_{\mathsf{D}})^{\textcircled{m}}\langle \mathcal{D}ec\rangle & & & \\
\succ_{\mathsf{D}} & \equiv & (\succ, \succ^{\textcircled{m}}_{\mathsf{D}}\langle \mathcal{S}ub\rangle, (\succ^{\textcircled{m}}_{\mathsf{D}})^{\textcircled{m}}\langle \mathcal{D}ec \circ \mathcal{A}rgs\rangle) & \sim_{\mathsf{D}} & \equiv & \approx\langle \mathcal{I}d\rangle
\end{array} \qquad \diamond$$

3.21 Remark

Example 3.14 illustrates that the IRD is an extension of the RDO (see also chapter 6). This example is mainly based on the ternary operator $\wedge$. Note that the IRD is stronger than the RDO only if at least one operator with an arity greater than 2 is used. This follows from the last condition of $\succ_{\mathsf{D}}$: '$\succ^{\textcircled{m}}_{\mathsf{RDO}}$' and '$(\succ^{\textcircled{m}}_{\mathsf{D}})^{\textcircled{m}}\langle \mathcal{D}ec \circ \mathcal{A}rgs\rangle$', the only difference between the IRD and the RDO. If each operator has an arity of 2, at most, the relation $\succ^{\textcircled{m}}_{\mathsf{RDO}}$ is identical to $\succ^{\textcircled{m}}_{\mathsf{RDO}}\langle \mathcal{A}rgs\rangle$ and has to compare none or one term on the left-hand side with none or one term on the right-hand side (see remark 3.19). Together with the fact '$\succ_{\mathsf{RDO}} \equiv (\succ^{\textcircled{m}}_{\mathsf{D}})^{\textcircled{m}}\langle \mathcal{D}ec\rangle$', the RDO is equivalent to the IRD whenever the signature contains operators with an arity smaller than 3, only.

In [Lescanne, 1984], a modification of the RDO is presented which integrates a status function. This decomposition ordering (called RDOS) does not use the whole information of the original

---

[7] $\succ_{\mathsf{D}}$ is an abbreviation for Decomposition ordering (i.e. ordering on elementary decompositions).

decomposition data structure. It does not require to select one subterm (see operation $\mathcal{S}ub$) of the set of arguments belonging to a specific operator. Thus, a path-decomposition of the RDOS is not as strictly confined to a leaf as a path-decomposition used in the definition of the RDO. This is why Lescanne provides a simpler elementary decomposition in [Lescanne, 1990] which is only a pair consisting of the leading operator of a term t and the 'list of terms below it' (i.e. the arguments of t) for defining the RDOS restricted to status ①. A.18 A.26 A.31 A.32 A.35 A.36

Idea of Comparison *The decompositions of both terms will be compared w.r.t. an ordering comparing two terms by comparing their leading operators and if they are equal, by recursively comparing their arguments w.r.t. the* RDOS.

3.22 **Definition (Recursive Decomposition Ordering with Status, [Lescanne, 1984])** *Let $\succsim$ be a quasi-precedence and τ be a status function. Then the* recursive decomposition ordering RDOS with status *is defined as* A.27

$$\succ_{\mathsf{RDOS}} \equiv (\succ_{\mathsf{D}}^{ⓜ})^{ⓜ}\langle \mathcal{D}ec\rangle \qquad \sim_{\mathsf{RDOS}} \equiv \approx\langle \mathcal{I}d\rangle$$
$$\succ_{\mathsf{D}} \equiv (\succ, \succ_{\mathsf{RDOS}}^{\tau})$$
◇

3.23 **Theorem ([Lescanne, 1984])** *The* RDOS *is a σ-stable simplification ordering.* ◇

3.2 New Decomposition Orderings

This section deals with the introduction of three new decomposition orderings: (i) a new decomposition ordering DOS with status (which is more powerful than the RDO *as well as* the RDOS), (ii) a variant of the PSO based on decompositions with an incorporated status function and (iii) an improvement of the IRD which includes a status function.

The decomposition ordering DOS, given in the following definition, differs from the RDOS contained in [Lescanne, 1984]. Both orderings are based on the RDO but the incorporations of status are different:

- The RDOS compares the arguments of two terms s and t (whose leading operators are equal) w.r.t. the extension of the RDOS due to the status function, i.e.

  s is greater than t if, and only if, $\mathcal{A}rgs(s) \succ_{\mathsf{RDOS}}^{\tau} \mathcal{A}rgs(t)$.

- In contrast to the RDOS, the DOS differentiates between multiset status and lexicographic status. If the leading operators have status ⓜ, then the original RDO will be applied. However, lexicographic status of the leading operators implies the application of the RDOS.

Idea of Comparison *The operational method of the* DOS *for comparing terms is identical to that of the* RDOS *except when the leading operators are the same and have multiset status. Then the* DOS *is equal to the* RDO.

$\boxed{3.24}$ **Definition (Decomposition Ordering with Status)** *Let* $\gtrsim$ *be a quasi-precedence* A.9
and τ *be a status function. Then the* decomposition ordering DOS with status *is defined as* A.16 A.27

$$
\begin{array}{lcl}
\succ_{\mathsf{DOS}} & \equiv & (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle \mathcal{D}ec\rangle \\
\succ_{\mathsf{D}} & \equiv & (\succ\,, \{\tau = \textcircled{m} \;\&\; (\succ_{\mathsf{D}}^{\textcircled{m}}\langle \mathcal{S}ub\rangle, \succ_{\mathsf{DOS}}^{\textcircled{m}}), \tau \neq \textcircled{m} \;\&\; \succ_{\mathsf{DOS}}^{\tau}\}) \\
\sim_{\mathsf{DOS}} & \equiv & \approx\langle \mathcal{I}d\rangle \\
\sim_{\mathsf{D}} & \equiv & \approx\langle \mathcal{I}d\rangle
\end{array}
\qquad \diamond
$$

$\boxed{3.25}$ **Theorem** *The* DOS *is a* σ*-stable simplification ordering.* $\diamond$

Another ordering based on decompositions results from the PSO. It is remarkable that the PSO is an extremely recursive ordering which takes three suborderings ($\succ_{\mathsf{PO}}$, $\succ_{\mathsf{TO}}$ and $\succ$, see definition 3.1 on page 38) into account. We have succeeded in redefining this path ordering in such a way that the result, called PSD, provides a much simpler method. The PSD, based on decompositions, represents a natural description of the PSO with the same power (see chapter 6). The PSD has another advantage over the PSO: The combination with the concept of status is much easier. The PSD with status (PSDS, for short) as well as the PSD depend on the fact that a path is an ordered path-decomposition.

Idea of Comparison *The comparison of two terms w.r.t. the* PSDS *requires the decompositions of one to be greater than that of the other one w.r.t. the extension of* $\succ_{\mathsf{D}}$ *to multisets of multisets. The semantics of* $\succ_{\mathsf{D}}$ *is identical to that of the corresponding ordering in the* RDOS*, with the exception of the situation describing two identical leading operators with status* $\textcircled{m}$*. Then the decompositions of their arguments will be compared (using the extension of* $\succ_{\mathsf{D}}$ *to multisets of multisets).*

$\boxed{3.26}$ **Definition (Path of Subterms Ordering with Status Based on Decomposi-**
A.8 **tions)** *Let* $\gtrsim$ *be a quasi-precedence and* τ *be a status function. Then the* path of subterms
ordering PSDS with status based on decompositions *is defined as*

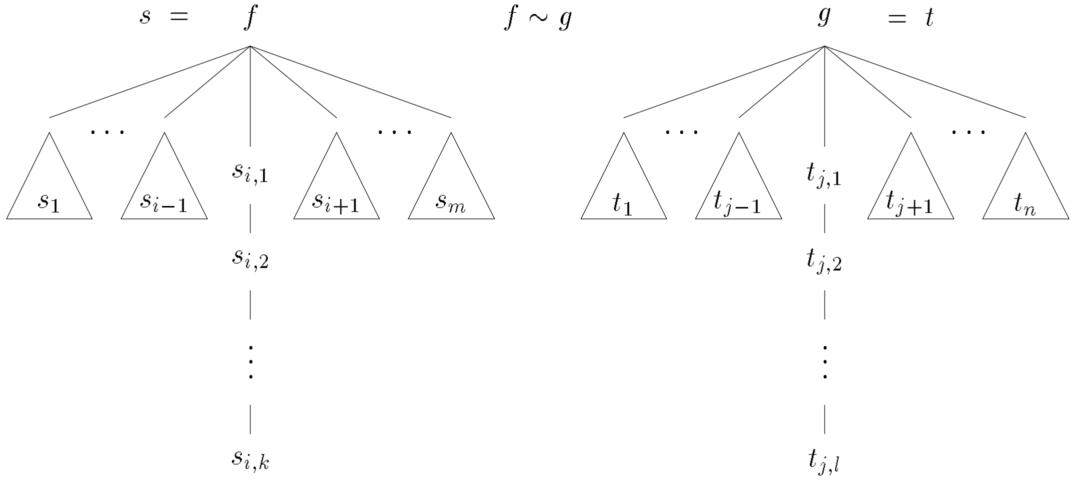
$$
\begin{array}{lcl}
\succ_{\mathsf{PSDS}} & \equiv & (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle \mathcal{D}ec\rangle \\
\succ_{\mathsf{D}} & \equiv & (\succ\,, \{\tau = \textcircled{m} \;\&\; (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle \mathcal{D}ec \circ \mathcal{A}rgs\rangle, \tau \neq \textcircled{m} \;\&\; \succ_{\mathsf{PSDS}}^{\tau}\}) \\
\sim_{\mathsf{PSDS}} & \equiv & \approx\langle \mathcal{I}d\rangle \\
\sim_{\mathsf{D}} & \equiv & \approx\langle \mathcal{I}d\rangle
\end{array}
\qquad \diamond
$$

$\boxed{3.27}$ **Theorem** *The* PSDS *is a* σ*-stable simplification ordering.* $\diamond$

A.15 The part of the PSDS specifying the PSO (i.e. the part associated to $\tau = \textcircled{m}$) has an obvious
A.27 advantage : Comparing the PSO with decomposition orderings is easy. The essential difference is that the PSO works breadth first and the RDO works depth first. The new definition of the PSO clarifies this:

PSO:	$\succ_{\mathsf{D}}$	$\equiv$	$(\succ, (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle \mathcal{D}ec \circ \mathcal{A}rgs\rangle)$
RDO:	$\succ_{\mathsf{D}}$	$\equiv$	$(\succ, \succ_{\mathsf{D}}^{\textcircled{m}}\langle \mathcal{S}ub\rangle, \succ_{\mathsf{RDO}}^{\textcircled{m}})$

If the leading operators of the terms to be compared are identical, the RDO chooses only *one* subterm ($\succ_{\mathsf{D}}^{\textcircled{m}}\langle\mathcal{S}ub\rangle$). On the other hand, the PSO (PSD) considers the decomposition multisets of *all* subterms, simultaneously:

> Let $s = f(s_1,\ldots,s_m)$, $t = g(t_1,\ldots,t_n)$ with $f \sim g$, $u \in \mathcal{TP}os(s)$, $v \in \mathcal{TP}os(t)$ and $\mathcal{D}ec_u(s) = \{s, s_{i,1}, s_{i,2},\ldots, s_{i,k}\}$, $\mathcal{D}ec_v(t) = \{t, t_{j,1}, t_{j,2},\ldots, t_{j,l}\}$ where $s_{i,1} = s_i$ and $t_{j,1} = t_j$:
>
> 
>
>
> Then the definitions of the PSO and the RDO require the following comparisons:
>
> PSO: $\uplus_{p\in[1,m]}\mathcal{D}ec(s_p)\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \uplus_{q\in[1,n]}\mathcal{D}ec(t_q)$
>
> RDO: $\{s_{i,1},\ldots, s_{i,k}\}\ \succ_{\mathsf{D}}^{\textcircled{m}}\ \{t_{j,1},\ldots, t_{j,l}\}$

The IRD of Rusinowitch (described in subsection 3.1.3) reveals the use of the principle of depth first (as the RDO):

IRD:	$\succ_{\mathsf{D}}$	$\equiv$	$(\succ, \succ_{\mathsf{D}}^{\textcircled{m}}\langle\mathcal{S}ub\rangle, (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle\mathcal{D}ec\circ\mathcal{A}rgs\rangle)$
RDO:	$\succ_{\mathsf{D}}$	$\equiv$	$(\succ, \succ_{\mathsf{D}}^{\textcircled{m}}\langle\mathcal{S}ub\rangle, \succ_{\mathsf{RDO}}^{\textcircled{m}})$

However, if the subterms are equal, the RDO compares the arguments of the two terms w.r.t. the RDO itself while the IRD compares the multiset sums of the decompositions of the arguments w.r.t. $\succ_{\mathsf{D}}$:

and $\mathcal{D}ec(s_i) = \{\!\{\,\{\ldots\},\ldots,\{\ldots\}\,\}\!\}$ is a multiset of (multi-) sets of terms ($\mathcal{D}ec(t_i)$ is analogously constructed).

RDO: $\{\!\{\, s_1,\ldots, s_m \,\}\!\}\ \succ_{\mathsf{RDO}}^{\textcircled{m}}\ \{\!\{\, t_1,\ldots, t_n \,\}\!\} \Longleftrightarrow$

$$\{\!\{ \underbrace{\{\!\{\,\{\ldots\},\ldots,\{\ldots\}\,\}\!\}}_{\mathcal{D}ec(s_1)},\ldots,\underbrace{\{\!\{\,\{\ldots\},\ldots,\{\ldots\}\,\}\!\}}_{\mathcal{D}ec(s_m)} \}\!\}\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \{\!\{ \underbrace{\{\!\{\,\{\ldots\},\ldots,\{\ldots\}\,\}\!\}}_{\mathcal{D}ec(t_1)},\ldots,\underbrace{\{\!\{\,\{\ldots\},\ldots,\{\ldots\}\,\}\!\}}_{\mathcal{D}ec(t_n)} \}\!\}$$

$$\textsf{IRD:}\ \{\!\{\ \underbrace{\{\ldots\},\ldots,\{\ldots\}}_{\mathcal{D}ec(s_1)\text{ without }\{\!\{\}\!\}}\ ,\ldots,\ \underbrace{\{\ldots\},\ldots,\{\ldots\}}_{\mathcal{D}ec(s_m)\text{ without }\{\!\{\}\!\}}\ \}\!\}\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \{\!\{\ \underbrace{\{\ldots\},\ldots,\{\ldots\}}_{\mathcal{D}ec(t_1)\text{ without }\{\!\{\}\!\}}\ ,\ldots,\ \underbrace{\{\ldots\},\ldots,\{\ldots\}}_{\mathcal{D}ec(t_n)\text{ without }\{\!\{\}\!\}}\ \}\!\}$$

Thus, in contrast to the RDO, the IRD *removes one level of parentheses*. This way, path-decompositions of *different* terms can help each other to cover a path-decomposition of one term. This is not possible with the RDO since the additional parentheses tie all path-decompositions of a term together, see example 3.14.

Summarizing, there exist four different methods of comparing elementary decompositions (assuming that their leading operators are equal):

- ▶ Multiset status
  - ▶ (1) Breadth first (as in the PSD)
  - ▶ Depth first
    - ▶ (2) Without joining the decompositions of the arguments (as in the RDO)
    - ▶ (3) By joining the decompositions of the arguments (as in the IRD)
- ▶ (4) Lexicographic status

There already exist decomposition orderings combining the breadth first technique with lexicographic status (PSDS) as well as the depth first approach without joining the decompositions of the arguments and lexicographic status (DOS). The concatenation of the depth first technique by joining the decompositions of the arguments with lexicographic status has not yet been considered. The following ordering, called IRDS (IRD with status) combines these two concepts.

Idea of Comparison *The* IRDS *is an improvement of the* IRD *that incorporates an additional status function. Lexicographic types of status are realized as in the* RDOS, *the* PSDS *and the* DOS.

A.16
A.27
3.28 **Definition (Improved Recursive Decomposition Ordering with Status)** *Let* $\succsim$ *be a quasi-precedence and* τ *be a status function. Then the* improved recursive decomposition ordering IRDS with status *is defined as*

$$\begin{array}{lcl}
\succ_{\mathsf{IRDS}} & \equiv & (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle\mathcal{D}ec\rangle \\
\succ_{\mathsf{D}} & \equiv & (\succ\,,\{\tau=\textcircled{m}\ \&\ (\succ_{\mathsf{D}}^{\textcircled{m}}\langle\mathcal{S}ub\rangle,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\langle\mathcal{D}ec\circ\mathcal{A}rgs\rangle),\tau\neq\textcircled{m}\ \&\ \succ_{\mathsf{IRDS}}^{\tau}\}) \\
\sim_{\mathsf{IRDS}} & \equiv & \approx\langle\mathcal{I}d\rangle \\
\sim_{\mathsf{D}} & \equiv & \approx\langle\mathcal{I}d\rangle
\end{array}$$

◇

As the PSDS is an extended version of the PSO based on decompositions and a status function, the IRDS is a decomposition ordering improving the KNS by integrating a status function. More precisely, the IRDS and the KNSS are identical when considering their powers (see chapter 6).

3.29 **Theorem** *The* IRDS *is a σ-stable simplification ordering.* $\diamond$

Rusinowitch has mentioned (in [Rusinowitch, 1987]) that it is possible to mix the different kinds of comparing elementary decompositions (referred to above). More precisely, one has a generalized status function that specifies how to compare elementary decompositions (in multiset breadth first order, in lexicographic order, etc.).

3.3 AC-Case

This section deals with restricted versions of the orderings presented in the former sections that are suitable for AC-theories. In the last ten years, one specific ordering – called *associative path ordering* – has been developed and improved more than any other ordering. It is based on the RPO and the concept that simplification orderings have to be AC-compatible to provide termination proofs for AC-rewriting (see theorem 1.25 on page 12). The contents of this section are twofold: (i) The various well-known variants of the associative path ordering are given and their essential features are discussed in order to motivate and clarify (ii) some extensions of the associative path ordering.

In detail, the following main points are treated. The associative path ordering was presented for AC-theories, only. First of all, we will split AC into the concepts of C- (subsection 3.3.1) and A-theories (subsection 3.3.2) before combining them in subsection 3.3.3. In practice, this will enhance the efficiency if either C-theories or A-theories are used. The reason is that the conditions required for the RPO for any of the two cases are not as strict as those for the combined theories.

In subsection 3.3.4, we show that quasi-precedences on AC-operators cannot be used for associative path orderings since they violate the σ-stability.

Another main point of this section is the introduction of several improvements of the associative path ordering (see subsection 3.3.5). (i) It is possible to improve the associative path ordering by weakening the conditions required for the precedence. (ii) Another extension we deal with consists of the incorporation of a status function. (iii) We prove the application of the concept of the associative path ordering – including the former extensions – to the strongest path and decomposition ordering to be correct.

3.3.1 C-Theories

Assume that the underlying theory $\mathcal{E}$ contains only commutative axioms such as $f(x,y) \leftrightarrow_{\mathcal{E}} f(y,x)$. Therefore, a reduction ordering $\succ_{\mathsf{T}}$ is needed which is *C-compatible*: $s' \leftrightarrow_C s \succ_{\mathsf{T}} t \leftrightarrow_C t'$ implies $s' \succ_{\mathsf{T}} t'$. It is not very difficult to detect that the RPO is C-compatible since all operators have multiset status.

Idea of Comparison *The* C-RPO *for C-theories is identical to the* RPO.

3.30 **Definition (Commutative RPO)** *Let $\succ$ be a precedence. Then the* commutative recursive path ordering C-RPO *is defined as*

$$\succ_{\mathsf{C\text{-}RPO}} \equiv \succ_{\mathsf{RPO}}\langle \mathcal{I}d \rangle \qquad \diamond$$

Obviously, the combination of the C-compatibility with the other important properties of the A.14 RPO – simplification ordering and σ-stability – ensures its applicability in practice.

3.31 Lemma *The* C-RPO *is a C-compatible σ-stable simplification ordering.* $\diamond$

3.3.2 A-Theories

Unfortunately, in contrast to C-theories, many reduction orderings are not compatible with A-theories[8]. For example, the RPO ist not A-compatible, i.e. $s' \leftrightarrow_A s \succ_{\mathsf{RPO}} t \leftrightarrow_A t'$ does not imply $s' \succ_{\mathsf{RPO}} t'$, in general.

3.32 Example ([Bachmair and Dershowitz, 1986]) *Let* $+ \in \mathcal{F}_A$ *and* $\succ$ *be a precedence satisfying* $1 \succ 0$. *Then* $1 + (0 + 0) \leftrightarrow_A (1 + 0) + 0 \succ_{\mathsf{RPO}} 1 + (0 + 0)$. *However,* $1 + (0 + 0) \not\succ_{\mathsf{RPO}} 1 + (0 + 0)$. $\diamond$

The essential reason for reduction orderings not being A-compatible is the fact that they are *too powerful.* For instance, in the example given above, $(1 + 0) + 0 \succ_{\mathsf{RPO}} 1 + (0 + 0)$ holds although $(1 + 0) + 0 \leftrightarrow_A 1 + (0 + 0)$. Thus, many reduction orderings are able to orient (ground) associative equations. Therefore, we have to *reduce the power* of the orderings for ensuring their A-compatibility. In the current subsection, this requirement will be satisfied by using so-called *transformation orderings.* As mentioned in section 1.4, the use of well-founded sets is fundamental for termination arguments. Given a TRS $\mathcal{R}$ and a well-founded ordering $\succ_{\mathsf{S}}$ on a set $\mathcal{S}$, the problem is to find a mapping from terms into $\mathcal{S}$ such that the well-foundedness of $\succ_{\mathsf{S}}$ implies the termination of $\mathcal{R}$. A so-called *transform* is a mapping T from terms to terms. For applying T, one has to define conditions on $\mathcal{R}$, T and $\succ_{\mathsf{S}}$ that are sufficient for the termination of $\mathcal{R}$. In [Bachmair and Dershowitz, 1986], T is a canonical TRS. The following lemma is of main interest for A- (as well as AC-) termination.

3.33 Lemma ([Bachmair and Dershowitz, 1986]) *Let* $\mathcal{R}$, S *and* T *be TRSs and* $\mathcal{E}$ *be an equational theory. Suppose that* T *is terminating and confluent modulo* $\mathcal{E}$, $(S/T)/\mathcal{E}$ *is terminating as well as* $\xrightarrow{+}_{S/T}$ *and* $\xrightarrow{*}_T$ *commute. If* $\mathcal{R}$ *is reducing relative to* S/T *and* T, *then* $R/(\mathcal{E} \cup T \cup T^{-1})$ *terminates.* $\diamond$

Two TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ *commute* if ${}_{\mathcal{R}_1}\!\!\leftarrow \circ \rightarrow_{\mathcal{R}_2} \subseteq \rightarrow_{\mathcal{R}_2} \circ {}_{\mathcal{R}_1}\!\!\leftarrow$. A TRS $\mathcal{R}$ is *reducing relative* to S_1 and S_2 if $\rightarrow_{\mathcal{R}} \subseteq \xrightarrow{*}_{S_2} \circ \rightarrow_{S_1} \circ {}_{S_2}\!\!\xleftarrow{*}$.

In [Bachmair and Dershowitz, 1986], S, T and $\mathcal{E}$ are specified such that (i) the conditions of lemma 3.33 are satisfied and (ii) the set of AC-axioms is contained in $\mathcal{E} \cup T \cup T^{-1}$, in order to ensure the termination of $\mathcal{R}/AC$:

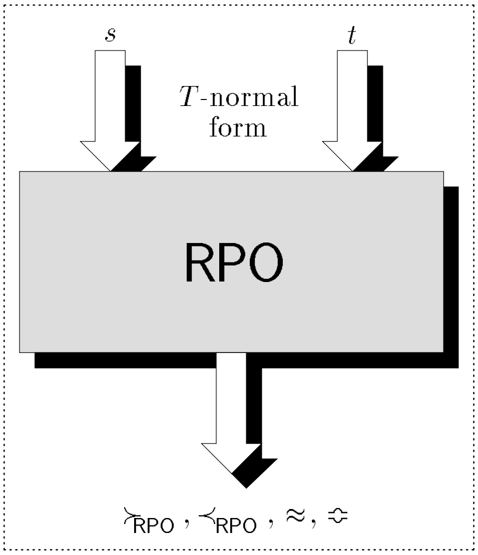
- For the moment, let T consist of all terminating rewriting rules computing flattened normal forms of terms (see, for example, [Bachmair and Dershowitz, 1986]),
- S is a TRS modelling the RPO restricted to terms irreducible w.r.t. T and

---

[8] Note that it is difficult to implement a full Knuth-Bendix completion procedure for associative but *not* commutative operators since there exists no finite unification algorithm for A-operators (see [Plaisted, 1983]).

- $\mathcal{E}$ represents the congruence $\approx$ (which is the congruence of permutation since the RPO is used and thus, all operators have status ⓜ).

The instances of the systems S, T and $\mathcal{E}$ given above satisfy the conditions of lemma 3.33 *with the exception of the commutation property.* This property is guaranteed for *restricted precedences*, only and thus, *T will be refined* during this section. Combining the three features described above, the following technique is established in order to restrict the RPO so that it can prove the A-termination of some TRSs:



- Establishing exactly one representative of each A-equivalence class
- Reducing each term to the representative of its A-equivalence class
- Comparing the representatives w.r.t. the RPO

An ordering constructed using this technique is A-compatible since $rep(s') = rep(s) \succ_{\mathsf{RPO}} rep(t) = rep(t')$ implies $rep(s') \succ_{\mathsf{RPO}} rep(t')$, where $s' \leftrightarrow_A s \wedge t \leftrightarrow_A t'$ and rep denotes the representative (i.e. the normal form w.r.t. T) of a term.

Now, an A-compatible ordering could be defined as $s \succ_{\mathsf{A\text{-}RPO}} t$ if, and only if, $\overline{s} \succ_{\mathsf{RPO}} \overline{t}$. Unfortunately, this A-RPO is not $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible.

$\boxed{3.34}$ **Example** *Let* $f \in \mathcal{F}_A$ *and* $f \succ g$. *Then* $\overline{f(a,b)} \succ_{\mathsf{RPO}} \overline{g(a,b)}$ *but* $\overline{f(f(a,b),c)} = f(a,b,c) \prec_{\mathsf{RPO}} f(g(a,b),c) = \overline{f(g(a,b),c)}$. ◇

Requiring the *A-operators to be minimal* w.r.t. the precedence – $\forall f \in \mathcal{F}_A\colon \nexists g \in \mathcal{F}\colon f \succ g$ – the ordering $\succ_{\mathsf{A\text{-}RPO}}$ can be used for proving A-termination of TRSs.

Idea of Comparison *The* A-RPO *is identical to the* RPO *on flattened normal forms of terms. Furthermore, the precedence must satisfy the minimum condition for A-operators.*

$\boxed{3.35}$ **Definition (Associative** RPO **[Bachmair and Plaisted, 1985b] and [Bachmair and Dershowitz, 1986])** *Let* $\succ$ *be a precedence such that each A-operator is minimal. Then the* associative recursive path ordering A-RPO *is defined as* |A.6

$$\succ_{\mathsf{A\text{-}RPO}} \;\equiv\; \succ_{\mathsf{RPO}}\langle\overline{}\rangle$$ ◇

$\boxed{3.36}$ **Lemma ([Bachmair and Dershowitz, 1986])** *The* A-RPO *is an A-compatible σ-stable simplification ordering.* ◇

A.1 A.2 A.33 Due to lemma 3.36 and theorem 1.25, the A-RPO can be used to prove A-termination of TRSs. However, the restriction of the precedence (each A-operator must be minimal) is somewhat strong.

3.37 Example *Let*

$$\mathcal{R}: \begin{cases} (x+y)\star z & \to & (x\star z)+(y\star z) \\ h(x)\star h(y) & \to & h(x\star y) \end{cases}$$
$$\mathcal{E}: \quad +,\star \in \mathcal{F}_A$$

The TRS $\mathcal{R}$ requires $\star \succ +$ (first rule) and $\star \succ h$ (second rule), i.e. the A-operator $\star$ is only sub-minimal[9]*. Therefore, we are not able to prove the $\mathcal{E}$-termination of $\mathcal{R}$ with the help of the* A-RPO. ◇

Thus, we want to permit *sub-minimal A-operators.* However, as proved in example 3.34, the $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility of the A-RPO is not fulfilled. Consider example 3.34 once again: Note that the term $f(f(a,b),c)$ (which should be greater than $f(g(a,b),c)$) has to be flattened causing a reduction w.r.t. the RPO. On the other side, $f(g(a,b),c)$ – the presumably smaller term – is left unchanged. *The secret now consists of reducing $f(g(a,b),c)$, too.* I.e. we will force the $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility using the intuitive idea that the smaller term has only to be reduced when the bigger one is reduced by flattening.

3.38 Definition (Distributivity Rules, Based on [Bachmair and Dershowitz, 1986] and [Gnaedig and Lescanne, 1986]) *Let $f,g \in \mathcal{F}$ such that $\mathcal{A}r(f), \mathcal{A}r(g) > 0$ and $\succ$ be a precedence satisfying $f \succ g$. A* distributivity rule *for f and g is a rewriting rule of the form*

$$f(X, g(y_1,\ldots,y_n), Z) \to g(f(X,y_1,Z),\ldots,f(X,y_n,Z)).$$

The set of all distributivity rules for $\succ$ is defined as $D(\succ) = \{l \to r \mid (f \in \mathcal{F}_A \vee \mathcal{A}r(f) = 1)$ and $(g \in \mathcal{F}_A \vee \mathcal{A}r(g) = 1)$, $f \succ g$ and $l \to r$ is a distributivity rule for f and $g\}$. ◇

3.39 Remark

The set $D(\succ)$ of all distributivity rules w.r.t. a precedence $\succ$ considers the relation pairs $f \succ g$ where $(f \in \mathcal{F}_A \vee \mathcal{A}r(f) = 1)$ and $(g \in \mathcal{F}_A \vee \mathcal{A}r(g) = 1)$. Note that it is also possible to omit these limitations of f and g. Moreover, $D(\succ)$ can further be restricted by postulating $f \in \mathcal{F}_A$ (which is done in [Bachmair and Dershowitz, 1986] and [Gnaedig and Lescanne, 1986]). Both modifications of $D(\succ)$ can lead to other results when comparing two terms.

For example, $x \star (y+z) \to (x \star y) + (x \star z)$ and $-(x+y) \to (-x)+(-y)$ are distributivity rules. Such sets of distributivity rules are terminating (w.r.t. an RPO) but not confluent, in general.

3.40 Example ([Bachmair and Dershowitz, 1986]) *Let f be a binary and g,h be two unary operators. The precedence $\succ$ satisfies $f \succ g$ and $f \succ h$:*

---

[9] An operator f is *sub-minimal* if, and only if, $\nexists g,h \in \mathcal{F}$: $f \succ g \succ h$, i.e. there exists at most one set of minimal operators which is smaller (w.r.t. $\succ$) than f.

$$D(\succ) \;=\; \begin{cases} f(x,g(y)) & \to \quad g(f(x,y)) \\ f(g(x),y) & \to \quad g(f(x,y)) \\ f(x,h(y)) & \to \quad h(f(x,y)) \\ f(h(x),y) & \to \quad h(f(x,y)) \end{cases}$$

Then

$$\begin{tikzcd} & g(f(x,h(y))) \arrow[r] & g(h(f(x,y))) \\ f(g(x),h(y)) \arrow[ur] \arrow[dr] & & \\ & h(f(g(x),y)) \arrow[r] & h(g(f(x,y))) \end{tikzcd}$$

and thus, $D(\succ)$ is not confluent. ◇

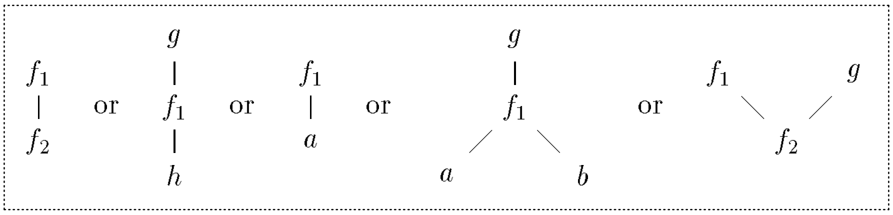
For guaranteeing the confluence of $D(\succ)$ for a particular precedence $\succ$, $\succ$ has to be restricted.

$\boxed{3.41}$ **Definition (Associative Path Condition, [Bachmair and Dershowitz, 1986] and [Gnaedig and Lescanne, 1986])** *Let $\succ$ be a precedence and $\mathcal{F}$ be a set of operators, $\mathcal{F}_A \subseteq \mathcal{F}$, $\mathcal{F}_1$ be a subset of all unary operators of $\mathcal{F}$ and let $\mathcal{F}_D = \mathcal{F}_A \cup \mathcal{F}_1$*10)*. Furthermore, let $\mathcal{F}'$ be $\mathcal{F} \setminus \{a_1,\ldots,a_k\}$ if $a_1,\ldots,a_k$ are minimal w.r.t. $\succ$ among all constants, or $\mathcal{F}$, if there is no such a constant. Then $\succ$ satisfies the* associative path condition *if $\mathcal{F}_D$ can be partitioned into two sets $\{f_1,\ldots,f_m\}$ and $\{g_1,\ldots,g_n\}$ such that $m \leq n$ and the following conditions hold:*

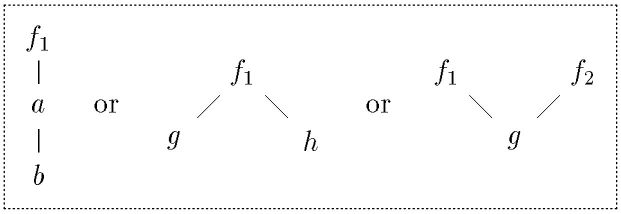
- $\forall i \in [1,m]$: $f_i \succ g_i$
- *Every g_i is minimal w.r.t. $\succ$ in $\mathcal{F}'$, i.e. $\forall i \in [1,n]$: $\nexists h \in \mathcal{F}'$: $g_i \succ h$*
- *Every f_i is minimal w.r.t. $\succ$ in $\mathcal{F}' \setminus \{g_i\}$, i.e. $\forall i \in [1,m]$: $\nexists h \in \mathcal{F}' \setminus \{g_i\}$: $f_i \succ h$* ◇

Figure 3.3 provides a graphical explanation of the slightly complex definition of the associative path condition. Some examples of precedences concerning the associative path condition are given below ($f_1, f_2 \in \mathcal{F}_A$, $\{g,h,a,b\} \subseteq \mathcal{F} \setminus \mathcal{F}_A$, $\mathcal{A}r(g) = \mathcal{A}r(h) = 1$, $\mathcal{A}r(a) = \mathcal{A}r(b) = 0$):

Admissible precedences:



Excluded precedences:



Obviously, for each precedence $\succ$ a rule system $D(\succ)$ can be found. Note that for each term there exists an irreducible form since this system is AC-terminating. Two normal forms of a

---

10) In [Bachmair and Dershowitz, 1986], $\mathcal{F}_D$ 'is a *subset* of $\mathcal{F}$ containing $\mathcal{F}_A$ but no constants'. Thus, $\mathcal{F}_D$ can also include binary operators which are not AC. However, such operators imply $D(\succ)$ to be no longer confluent modulo AC-theories.

To present the relations w.r.t. the precedence $\succ$ in a simple way, we use a kind of the so-called *Hasse diagrams.* If $a \succ b$, then we arrange a above b joining them with a line. Note that $a_1,\ldots,a_k$ are constants.

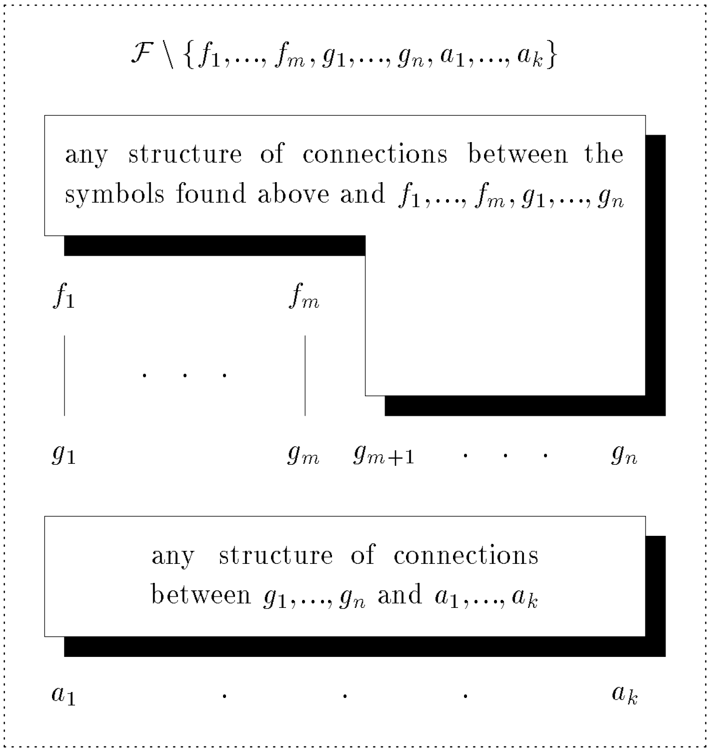


Figure 3.3: The associative path condition

term are AC-equivalent ([Bachmair and Dershowitz, 1986], [Gnaedig and Lescanne, 1986]). This property is sufficient because the flattening operation maps two A-equivalent terms to terms being syntactically equal, i.e. AC-equivalent terms will result in C-equivalent terms. Thus, the flattened normal forms (w.r.t. $D(\succ)$) of two terms are C-equivalent. Due to the C-compatibility of the RPO it is possible to define an A-compatible ordering.

Idea of Comparison *The comparison of two terms w.r.t. the* AD-RPO *requires both terms to be reduced to normal forms (applying the rules of $D(\succ)$), first and then a comparison of the flattened normal forms w.r.t. the* RPO. *The precedence $\succ$ has to satisfy the associative path condition. The problem of proving the termination of a rule consisting of $D(\succ)$-equivalent terms (their normal forms are syntactically equal) is solved by checking whether equivalent terms are $D(\succ)$-equivalent ($s \xrightarrow{+}_{D(\succ)/A} t$ where $\rightarrow_{D(\succ)/A}$ represents the relation $\leftrightarrow_A \circ \rightarrow_{D(\succ)} \circ \leftrightarrow_A$).*

$\boxed{3.42}$ **Definition (Associative RPO Based on Distributivity, [Bachmair and Dershowitz, 1986], [Gnaedig and Lescanne, 1986])** *Let $\succ$ be a precedence satisfying the associative path condition and the operation δ: $\mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{T}(\mathcal{F},\mathcal{X})$ rewrites a term to an irreducible form w.r.t. the TRS $D(\succ)$. Then the* associative recursive path ordering AD-RPO based on distributivity *is defined as* A.23

$$\succ_{\mathsf{AD\text{-}RPO}} \equiv (\succ_{\mathsf{RPO}}\langle^{-} \circ \delta\rangle \,,\, \xrightarrow{+}_{D(\succ)/A}\langle \mathcal{I}d\rangle) \qquad \diamond$$

Note that the associative path condition guarantees that the transformation of distributing and flattening is well-defined and that the AD-RPO is in fact a simplification ordering.

3.43 **Lemma ([Bachmair and Dershowitz, 1986])** *The* AD-RPO *is an A-compatible σ-stable simplification ordering.* ◇ A.4 A.5 A.22

3.44 **Example (Example 3.37 Continued)** *Let*

$$\begin{array}{ll} \mathcal{R}: & \left\{ \begin{array}{lcl} (x+y)\star z & \rightarrow & (x \star z)+(y \star z) \\ h(x) \star h(y) & \rightarrow & h(x \star y) \end{array} \right. \\ \mathcal{E}: & \quad +, \star \in \mathcal{F}_A \end{array}$$

With the help of the AD-RPO *based on $\star \succ +$ the $\mathcal{E}$-termination of the first rule can be proved since $\overline{\delta((x+y)\star z)} = (x{\star}z)+(y{\star}z) = \overline{\delta((x \star z)+(y \star z))}$ and $(x+y){\star}z \rightarrow_{D(\succ)/A} \underline{(x{\star}z)+(y{\star}z)}$ hold. The second rule is ordered w.r.t. the* AD-RPO *based on $\star \succ h$ because $\overline{\delta(h(x) \star h(y))} = h(h(x \star y)) \succ_{\mathsf{RPO}} h(x \star y) = \overline{\delta(h(x \star y))}$. However, the associative path condition does not allow to merge the precedences $\star \succ +$ and $\star \succ h$. Thus, the* AD-RPO *is not able to prove the termination of the entire $\mathcal{R}$.* ◇

3.45 **Remark**

The condition "let $\mathcal{F}'$ be $\mathcal{F} \setminus \{a_1,\ldots,a_k\}$ if $a_1,\ldots,a_k$ are minimal w.r.t. $\succ$ among all constants" of definition 3.41 allows one $\succ$-level of constants below AC-operators, only. If there were two $\succ$-levels of constants (for example, $a \succ b$) and two AC-operators f, g with $f \succ g \succ a$, the AD-RPO would not be $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible: $f(b,b) \succ_{\mathsf{AD\text{-}RPO}} g(b,a)$ holds, but not $f(f(b,b),b) \succ_{\mathsf{AD\text{-}RPO}} f(g(b,a),b)$. However, it is *possible* to weaken the associative path condition in such a way that arbitrarily many $\succ$-levels of constants will be allowed if only *one* AC-operator exists.

In addition to the assertions contained in the literature, we can prove that the AD-RPO without the second condition ($\xrightarrow{\pm}_{D(\succ)/A}\langle \mathcal{I}d \rangle$) is ACD-compatible. The theory ACD is specified by the AC-axioms and the rules of $D(\succ)$ considered as equations.

3.46 **Lemma** *The ordering $\succ_{\mathsf{ACD}}$ defined by $\succ_{\mathsf{RPOS}}\langle \bar{\ } \circ \delta \rangle$ with $\tau(f) = ⓜ$ if $f \in \mathcal{F}_{AC}$ is ACD-compatible.* ◇

3.47 **Remark**

In [Bachmair and Plaisted, 1985b], a method for generalizing the second component (i.e. for $\xrightarrow{\pm}_{D(\succ)/A}\langle \mathcal{I}d \rangle$) of the AD-RPO is presented. If two terms have the same flattened δ-normal forms (i.e. $\overline{\delta(s)} \sim_{\mathsf{RPOS}} \overline{\delta(t)}$) they could still be compared using some reduction ordering $\succ_{\mathsf{T}}$ that is A-compatible and well-founded on every set $[t] = \{s \mid \overline{\delta(s)} = \overline{\delta(t)}\}$. Such an ordering is called *admissible* for the transformation $\bar{\ } \circ \delta$. For example, comparing terms by the inverse of their sizes is an admissible ordering for the transformation used here. More precisely, $\succ_{\mathsf{T}}$ can be defined as $s \succ_{\mathsf{T}} t$ if, and only if, $|s| < |t|$ and no variable appears more often in s than in t.

3.3.3 AC-Theories

Guaranteeing the C-compatibility of the A-compatible orderings presented in the former subsection, we only have to merge these orderings with the C-RPO. Note that the C-RPO is identical to the RPO (i.e. the RPO is already C-compatible). Thus, all orderings described in subsection 3.3.2 are AC-compatible[11].

3.48 Lemma ([Bachmair and Dershowitz, 1986]) *The* A-RPO *and the* AD-RPO *are AC-compatible and σ-stable simplification orderings.* ◇

The list of publications on associative path orderings is relatively long. Thus, in order to give an overview of these articles, we summarize their substantial ideas and differences (see figure 3.4) as well as – as already presented for the decomposition orderings – illustrate their dependencies by a diagram[12]*(see figure 3.5).*

3.49 Remark

In some publications, it has been stated that $\succ_{\mathsf{RPO}}\langle\mathcal{I}d\rangle \subseteq \succ_{\mathsf{RPO}}\langle\overline{\ }\rangle$ holds if each A-operator is minimal w.r.t. the precedence. This is not valid, in general, as shown by an example: Let $f \in \mathcal{F}_A$, $i \in \mathcal{F} \setminus \mathcal{F}_A$ and $i \succ f$. Then $f(f(i(x),x),x) \succ_{\mathsf{RPO}} f(i(x),i(x))$ but $f(i(x),x,x) \not\succ_{\mathsf{RPO}} f(i(x),i(x))$ hold. However, for proving the associative path orderings to be σ-stable simplification orderings, the implication mentioned above is not necessary.

3.3.4 Quasi-Precedences

All orderings presented in this chapter are based on a precedence $\succ$. This ordering on the operators is not clearly defined anywhere. Note that it could be a partial ordering as well as a quasi-ordering. Assuming quasi-precedences, we will consider the following example.

3.50 Example *Let*

$$\begin{array}{ll} \mathcal{R}: & \mathit{false} \vee x \rightarrow \mathit{true} \wedge x \\ \mathcal{E}: & \wedge, \vee \in \mathcal{F}_{AC} \\ \succ: & \wedge \sim \vee \\ & \mathit{false} \succ \mathit{true} \end{array}$$

It is obvious that the rule of $\mathcal{R}$ is ordered w.r.t. the A-RPO *and the* AD-RPO*. However, this orientation is not σ-stable: Let $\sigma = \{x \leftarrow \mathit{false} \vee \mathit{false}\}$. Then*

$$\begin{array}{lcl} \mathit{false} \vee x & & \mathit{false} \vee (\mathit{false} \vee \mathit{false}) \\ \succ_{\mathsf{A\text{-}RPO}} (\succ_{\mathsf{AD\text{-}RPO}}) & \textit{but} & \not\succ_{\mathsf{A\text{-}RPO}} (\not\succ_{\mathsf{AD\text{-}RPO}}) \\ \mathit{true} \wedge x & & \mathit{true} \wedge (\mathit{false} \vee \mathit{false}) \end{array}$$

◇

This example induces that quasi-precedences on AC-operators must be prohibited in order to guarantee σ-stability.

---

[11] Obviously, the definition of the associative path condition must be adapted to AC-theories by replacing '$\mathcal{F}_A$' with '$\mathcal{F}_{AC}$'.

[12] The table is not represented as a graph (as in the case of decomposition orderings) since the graph would feature many crossing lines and would thus be somewhat confusing.

The original associative path ordering is presented in [Dershowitz *et al.*, 1983]. It is based on a relatively complex transformation (given by three functions). Furthermore, the precedence has to satisfy somewhat restricted conditions and it was not possible to lift this ordering to terms containing variables. However, the essential idea of applying distributivity rules was already developed in [Dershowitz *et al.*, 1983].
The main contributions of the articles [Plaisted, 1983], [Bachmair, 1984], [Bachmair and Plaisted, 1985a] and [Bachmair and Plaisted, 1985b] are comparable. The associative path ordering described therein is conceptually simpler than the associative path ordering of [Dershowitz *et al.*, 1983] since any term is transformed into a single term (instead of a multiset of terms). The σ-stability can be guaranteed by checking a finite number of potential (infinitely many) ground substitutions.
In [Bachmair and Dershowitz, 1985a] and [Bachmair and Dershowitz, 1985b], the σ-stability (without any additional tests) of the associative path ordering defined in [Plaisted, 1983], [Bachmair, 1984], [Bachmair and Plaisted, 1985a] and [Bachmair and Plaisted, 1985b] can be proved with the help of (general) transformation orderings. In contrast to [Bachmair and Dershowitz, 1985a], [Bachmair and Dershowitz, 1985b] contains a general scheme of distributivity rules, i.e. AC-operators with **variable** arities are admitted.
In [Gnaedig, 1986] and [Gnaedig and Lescanne, 1986], the associative path ordering of [Plaisted, 1983], [Bachmair, 1984], [Bachmair and Plaisted, 1985a] and [Bachmair and Plaisted, 1985b] is simplified (and called NFLO) by describing the transformation with the help of a reduction relation (such as in [Bachmair and Dershowitz, 1985a] and [Bachmair and Dershowitz, 1985b]). In contrast to the associative path ordering defined in [Bachmair and Dershowitz, 1985a] and [Bachmair and Dershowitz, 1985b] (and that in [Plaisted, 1983], [Bachmair, 1984], [Bachmair and Plaisted, 1985a], [Bachmair and Plaisted, 1985b]), the NFLO strongly differentiates between the flattening and the distributing processes. First of all, distributivity rules have to be applied and then the resulting terms must be flattened. As in [Bachmair and Dershowitz, 1985a] and [Bachmair and Dershowitz, 1985b] (and in contrast to [Plaisted, 1983], [Bachmair, 1984], [Bachmair and Plaisted, 1985a], [Bachmair and Plaisted, 1985b]), Gnaedig and Lescanne gave a proof for the σ-stability based on simple observations on the substitution mechanism. Inspired by the generalized distributivity rules of [Bachmair and Dershowitz, 1985b], they have developed a technique allowing $f \succ h$ where $f \in \mathcal{F}_{AC}$ and h is a unary operator. This case is treated by applying endomorphism rules $f(h(x),y) \to h(f(x,y)), f(x,h(y)) \to h(f(x,y))$ which are distributivity rules with a unary operator (see [Bachmair and Dershowitz, 1985b]).
The most powerful associative path ordering (its definition is recapitulated and discussed in this section) is the one presented in [Bachmair and Dershowitz, 1986] and [Bachmair, 1987]. It is based on [Bachmair and Dershowitz, 1985b] and also generalizes the approach of [Gnaedig, 1986] and [Gnaedig and Lescanne, 1986] by weakening the associative path condition. It was developed concurrently to [Gnaedig, 1986] and [Gnaedig and Lescanne, 1986].
In [Gnaedig, 1988], the definition of the associative path ordering of [Gnaedig, 1986] and [Gnaedig and Lescanne, 1986] is contained. Furthermore, a detailed study of the existence of total orderings for AC-theories is included (see also chapter 6).

Figure 3.4: History of associative path orderings (Part I)

$\boxed{3.51}$ **Lemma** *The* A-RPO *and the* AD-RPO *are not σ-stable, in general, if the precedence on the A(C)-operators is a quasi-ordering.* ◇

It is obvious that quasi-orderings on non-AC-operators can still be used.

3.3.5 Improving the Associative Path Orderings

In this subsection, we deal with the extension of the orderings presented in the former subsections. This will be done considering three different goals: (i) Weakening the requirements

	1	2	3	4	5	6	7	8	9	10		
1											1:	[Dershowitz *et al.*, 1983]
2	•										2:	[Plaisted, 1983]
3	•	•									3:	[Bachmair, 1984]
4	•	•	•								4:	[Bachmair and Plaisted, 1985a]
5				•							5:	[Bachmair and Dershowitz, 1985a]
6	•	•			•						6:	[Bachmair and Plaisted, 1985b]
7						•					7:	[Bachmair and Dershowitz, 1985b]
8	•	•		•			•				8:	[Gnaedig, 1986]
9		•		•			•	•			9:	[Gnaedig and Lescanne, 1986]
10	•	•				•					10:	[Bachmair and Dershowitz, 1986]
11	•	•				•				•	11:	[Bachmair, 1987]
12								•	•	•	12:	[Gnaedig, 1988]

Each item of the table stands for a reference contained in the publication on the left to the article given on the upper side.

Figure 3.5: History of associative path orderings (Part II)

of the associative path condition, (ii) integrating a status function and (iii) applying these two improvements to the strongest decomposition ordering (i.e. to the IRDS).

(i) Weakening the Associative Path Condition

We consider the following example.

3.52 **Example (Boolean Expressions)** *Let*

$$\begin{array}{ll} \mathcal{R}: & \left\{ \begin{array}{rcl} x \oplus x & \rightarrow & \neg true \\ (x \oplus y) \wedge z & \rightarrow & (x \wedge z) \oplus (y \wedge z) \end{array} \right. \\ \mathcal{E}: & \wedge, \oplus \in \mathcal{F}_{AC} \end{array}$$

◇

In order to prove the termination of $\mathcal{R}$ of example 3.52 with the RPO, the precedence must be of the form $\wedge \succ \oplus \succ \neg$, $\oplus \succ true$. Since $\wedge$ and $\oplus$ are AC-operators, either the A-RPO or the AD-RPO have to be used. However, the precedence violates the associative path condition.

We now introduce a technique that is able to prove the termination of $\mathcal{R}$ of example 3.52. It is based on *weakening the associative path condition* by simultaneously *enriching the distributivity rules* used for the reduction of terms to be compared:

- ▶ Divide the precedence $\wedge \succ \oplus \succ \neg$ into the pairs $\wedge \succ \oplus$, $\wedge \succ \neg$ and $\oplus \succ \neg$. This transformation keeps the whole information expressed by the original precedence.
- ▶ Construct $D(\succ)$ w.r.t. the pairs $\wedge \succ \oplus$, $\wedge \succ \neg$ and $\oplus \succ \neg$:

$$\left\{ \begin{array}{rcl} x \wedge (y \oplus z) & \rightarrow & (x \wedge y) \oplus (x \wedge z) \\ (x \oplus y) \wedge z & \rightarrow & (x \wedge z) \oplus (y \wedge z) \\ (\neg x) \wedge y & \rightarrow & \neg(x \wedge y) \\ x \wedge (\neg y) & \rightarrow & \neg(x \wedge y) \\ (\neg x) \oplus y & \rightarrow & \neg(x \oplus y) \\ x \oplus (\neg y) & \rightarrow & \neg(x \oplus y) \end{array} \right. \begin{array}{l} \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{induced by } \wedge \succ \oplus \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{induced by } \wedge \succ \neg \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{induced by } \oplus \succ \neg \end{array}$$

Unfortunately, this system is not confluent and not even confluent modulo AC-theories. However, we have found a somewhat simple enrichment that is confluent modulo AC-theories by adding the rule

$$\neg\neg(x \oplus y) \rightarrow \neg(x \oplus y).$$

The basic idea (described above) of the technique improving associative path orderings can be expanded by admitting an arbitrarily number of unary operators that are smaller (w.r.t. $\succ$) than two AC-operators. The following definitions formalize this concept.

$\boxed{3.53}$ **Definition (Extended Distributivity Rules)** *Let* $\{g, h_1, \ldots, h_n\} \subseteq \mathcal{F}$ *such that* $\mathcal{A}r(g) = m > 0$, $\mathcal{A}r(h_i) = 1$ *and* $\succ$ *be a precedence satisfying* $g \succ h_1 \succ h_2 \succ \cdots \succ h_n \wedge \nexists f \in \mathcal{F}$: $g \succ f \succ h_1$. *The set* $D^E(\succ)$ *of* extended distributivity rules *for* $\succ$ *is defined as*

$$D^E(\succ) = D(\succ) \cup$$

$$\left\{\begin{array}{rcl} h_i(h_i(g(x_1, \ldots, x_m))) & \rightarrow & h_i(g(x_1, \ldots, x_m)) \\ & & 1 \leq i \leq n \\ h_j(h_j(h_{i_1}(\cdots(h_{i_k}(g(x_1, \ldots, x_m)))\cdots))) & \rightarrow & h_j(h_{i_1}(\cdots(h_{i_k}(g(x_1, \ldots, x_m)))\cdots)) \\ & & 1 \leq i_k < \cdots < i_1 < j \leq n, 1 \leq k \leq n-1 \end{array}\right.$$

◇

$\boxed{3.54}$ **Example** *Let* $f, g, h_1, h_2 \in \mathcal{F}$ *with* $\mathcal{A}r(f) = \mathcal{A}r(g) = 2$, $\mathcal{A}r(h_1) = \mathcal{A}r(h_2) = 1$ *and* $f \succ g \succ h_1 \succ h_2$. *Then* $D^E(\succ)$ *is composed of the following rules:*

$$\left\{\begin{array}{l} \left.\begin{array}{rcl} f(x, g(y, z)) & \rightarrow & g(f(x, y), f(x, z)) \\ f(g(x, y), z) & \rightarrow & g(f(x, z), f(y, z)) \\ f(h_1(x), y) & \rightarrow & h_1(f(x, y)) \\ f(x, h_1(y)) & \rightarrow & h_1(f(x, y)) \\ f(h_2(x), y) & \rightarrow & h_2(f(x, y)) \\ f(x, h_2(y)) & \rightarrow & h_2(f(x, y)) \\ g(h_1(x), y) & \rightarrow & h_1(g(x, y)) \\ g(x, h_1(y)) & \rightarrow & h_1(g(x, y)) \\ g(h_2(x), y) & \rightarrow & h_2(g(x, y)) \\ g(x, h_2(y)) & \rightarrow & h_2(g(x, y)) \\ h_1(h_2(x)) & \rightarrow & h_2(h_1(x)) \end{array}\right\} D(\succ) \\ \begin{array}{rcl} h_1(h_1(g(x, y))) & \rightarrow & h_1(g(x, y)) \\ h_2(h_2(g(x, y))) & \rightarrow & h_2(g(x, y)) \\ h_2(h_2(h_1(g(x, y)))) & \rightarrow & h_2(h_1(g(x, y))) \end{array} \end{array}\right.$$

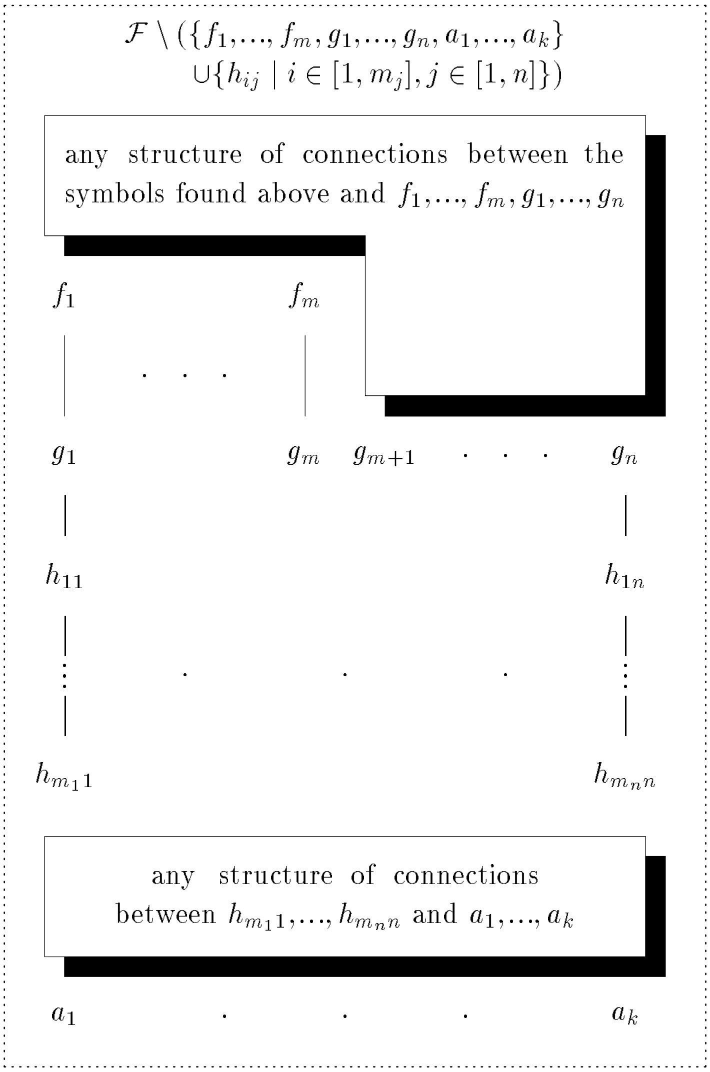
◇

$\boxed{3.55}$ **Definition (Extended Associative Path Condition)** *Let* $\succ$ *be a precedence and* $\mathcal{F}$ *be a set of operators,* $\mathcal{F}_A \subseteq \mathcal{F}$, $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *be two disjoint subsets of all unary operators of* $\mathcal{F}$ *and let* $\mathcal{F}_D = \mathcal{F}_A \cup \mathcal{F}_1$. *Furthermore, let* $\mathcal{F}'$ *be* $\mathcal{F} \setminus \{a_1, \ldots, a_k\}$ *if* $a_1, \ldots, a_k$ *are minimal w.r.t.* $\succ$ *among all constants, or* $\mathcal{F}$, *if there is no such a constant. Then* $\succ$ *satisfies the* extended associative path condition *if* $\mathcal{F}_D$ *can be partitioned into two sets* $\{f_1, \ldots, f_m\}$ *and* $\{g_1, \ldots, g_n\}$ *such that* $m \leq n$ *and the following conditions hold:*

- $\forall i \in [1, m]$: $f_i \succ g_i$
- *Every* g_i *is minimal w.r.t.* $\succ$ *in* $\mathcal{F}' \setminus \mathcal{F}_2$, *i.e.* $\forall i \in [1, n]$: $\nexists h \in \mathcal{F}' \setminus \mathcal{F}_2$: $g_i \succ h$

- ▶ *Every* f_i *is minimal w.r.t.* $\succ$ *in* $\mathcal{F}'\backslash(\{g_i\}\cup\mathcal{F}_2)$*, i.e.* $\forall i \in [1,m]$: $\nexists h \in \mathcal{F}'\backslash(\{g_i\}\cup\mathcal{F}_2)$: $f_i \succ h$
- ▶ $\mathcal{F}_2$ *can be partitioned into* n *sets* $\{h_{11},\ldots,h_{m_1 1}\},\cdots,\{h_{1n},\ldots,h_{m_n n}\}$ *such that* $\forall i \in [1,n]$: $\forall j \in [1, m_i - 1]$: $g_i \succ h_{ji} \succ h_{(j+1)i}$ *and* $\forall i \in [1,m_i], \forall j \in [1,m_j], \forall k,l \in [1,n]$: $k \neq l \hookrightarrow h_{ik} \not\gtrless h_{jl}$. ◇

The last condition somewhat technically demands that there exist n (possibly empty) linear chains w.r.t. the precedence with no connections among each other: $h_{11}\succ\cdots\succ h_{m_1 1},\ldots,h_{1n}\succ\cdots\succ h_{m_n n}$. As for the associative path condition, we provide a graphical explanation of the extended associative path condition using a Hasse diagram (see figure 3.6).



For example, in addition to the associative path condition, the following precedences are allowed by the extended associative path condition: $g\succ h_1\succ\cdots\succ h_n\succ a$ or $f\succ g\succ h_1\succ\cdots\succ h_n\succ a$ where $f,g \in \mathcal{F}_{AC}$, h_i are unary operators and a is a constant.

Figure 3.6: The extended associative path condition

We are now well prepared to define an extension of the AD-RPO which is based on the extended associative path condition and the extended distributivity rules.

Idea of Comparison *The comparison of two terms w.r.t. the* AE-RPO *requires both terms to be reduced to normal forms by applying the rules of $D^E(\succ)$ first, and then a comparison of the flattened normal forms w.r.t. the* RPO. *The precedence $\succ$ has to satisfy the extended associative path condition.*

3.56 Definition (Associative RPO Based on Extended Distributivity) *Let $\succ$ be a precedence satisfying the extended associative path condition and the operation γ: $\mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{T}(\mathcal{F},\mathcal{X})$ rewrites a term to an irreducible form w.r.t. the TRS $D^E(\succ)$. Then the* associative recursive path ordering AE-RPO based on extended distributivity *is defined as*

$$\succ_{\mathsf{AE\text{-}RPO}} \;\equiv\; (\succ_{\mathsf{RPO}}\langle ^{-} \circ \gamma \rangle \,,\, \xrightarrow{+}_{D^E(\succ)/A}\langle \mathcal{I}d \rangle) \qquad \diamond$$

3.57 Example (Example 3.52 Continued) *Let*

$$\begin{array}{ll} \mathcal{R}\colon & \left\{ \begin{array}{rcl} x \oplus x & \rightarrow & \neg true \\ (x \oplus y) \wedge z & \rightarrow & (x \wedge z) \oplus (y \wedge z) \end{array} \right. \\ \mathcal{E}\colon & \wedge, \oplus \in \mathcal{F}_{AC} \end{array}$$

The termination of $\mathcal{R}$ can be proved with the help of the AE-RPO *based on the precedence $\wedge \succ \oplus \succ \neg, \oplus \succ true$. This precedence implies $D^E(\succ)$ to be $\{x \wedge (y \oplus z) \rightarrow (x \wedge y) \oplus (x \wedge z)$, $(x \oplus y) \wedge z \rightarrow (x \wedge z) \oplus (y \wedge z)$, $(\neg x) \wedge y \rightarrow \neg(x \wedge y)$, $x \wedge (\neg y) \rightarrow \neg(x \wedge y)$, $(\neg x) \oplus y \rightarrow \neg(x \oplus y)$, $x \oplus (\neg y) \rightarrow \neg(x \oplus y)$, $\neg\neg(x \oplus y) \rightarrow \neg(x \oplus y)\}$. Therefore, $\overline{\gamma(x \oplus x)} = x \oplus x \succ_{\mathsf{RPO}} \neg true = \overline{\gamma(\neg true)}$ and $\overline{\gamma((x \oplus y) \wedge z)} = (x \wedge z) \oplus (y \wedge z) = \overline{\gamma((x \wedge z) \oplus (y \wedge z))}$, $(x \oplus y) \wedge z \xrightarrow{+}_{D^E(\succ)/A} (x \wedge z) \oplus (y \wedge z)$ hold.* ◇

3.58 Example (Example 3.44 Revisited) *As mentioned in example 3.44, the termination of the TRS*

$$\begin{array}{ll} \mathcal{R}\colon & \left\{ \begin{array}{lcl} (x + y) \star z & \rightarrow & (x \star z) + (y \star z) \\ h(x) \star h(y) & \rightarrow & h(x \star y) \end{array} \right. \\ \mathcal{E}\colon & +, \star \in \mathcal{F}_{A} \end{array}$$

cannot be proved by the AD-RPO. *However, $\mathcal{R}$ is ordered w.r.t. the* AE-RPO *based on the precedence $\star \succ + \succ h$.* ◇

The applicability of the AE-RPO presumes its theoretical foundation. The proof of the theoretical framework is mainly based on lemma 3.33 (on page 57) and thus, on the strategy of [Bachmair and Dershowitz, 1986]. For a formalization of the proof the definitions of some TRSs are helpful.

3.59 Definition *Let $\mathcal{F}_{AC} \subseteq \mathcal{F}$ be a set of AC-operators, $\succ$ be a precedence on $\mathcal{F}$ and $\succ_{\mathsf{ORD}}$ be any reduction ordering based on $\succ$. Then*

- $\mathcal{F}lat(\mathcal{F}_{AC}) = \{f(x, f(y_1,\ldots,y_n), z) \rightarrow f(x, y_1,\ldots,y_n, z) \mid f \in \mathcal{F}_{AC}, n \geq 2, |x|+|z| \geq 1\}$ *represents a system for flattening terms.*
- $\mathcal{T}rans(\succ, \mathcal{F}_{AC}) = D^E(\succ) \cup \mathcal{F}lat(\mathcal{F}_{AC})$ *corresponds to $^{-} \circ \gamma$.*

- *$\mathcal{T}erm(\succ_{\mathsf{ORD}}, \mathcal{F}_{AC}) = \{l \rightarrow r \mid l \succ_{\mathsf{ORD}} r$, l and r irreducible in $\mathcal{T}rans(\succ, \mathcal{F}_{AC})\}$ specifies the reduction ordering $\succ_{\mathsf{ORD}}$.* ◇

The following lemma provides the preconditions for lemma 3.33.

$\boxed{\textbf{3.60}}$ **Lemma** *Let $\mathcal{F}_{AC} \subseteq \mathcal{F}$ be a set of AC-operators and $\succ$ be a precedence on $\mathcal{F}$ satisfying the extended associative path condition. Then*

(i) $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$ is terminating and confluent modulo $\approx$,

(ii) $\mathcal{F}lat(\mathcal{F}_{AC})$ is canonical, $\mathcal{F}lat(\mathcal{F}_{AC})/\approx$ is terminating and $\leftrightarrow_{AC} \subseteq \approx \cup (\rightarrow_{\mathcal{F}lat(\mathcal{F}_{AC})} \cup {}_{\mathcal{F}lat(\mathcal{F}_{AC})}\!\!\leftarrow)^*$,

(iii) $\leftrightarrow_{AC} \subseteq \approx \cup (\rightarrow_{\mathcal{T}rans(\succ, \mathcal{F}_{AC})} \cup {}_{\mathcal{T}rans(\succ, \mathcal{F}_{AC})}\!\!\leftarrow)^*$

(iv) $\rightarrow_{(\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})/\mathcal{T}rans(\succ, \mathcal{F}_{AC}))/\approx}$ is terminating and

(v) $\xrightarrow{+}_{\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})/\mathcal{T}rans(\succ, \mathcal{F}_{AC})}$ and $\xrightarrow{*}_{\mathcal{T}rans(\succ, \mathcal{F}_{AC})}$ commute. ◇

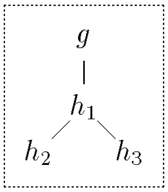
The combination of the above lemma and lemma 3.33 leads to the following description of AC-termination.

$\boxed{\textbf{3.61}}$ **Theorem** *Let $\mathcal{F}_{AC}$ be a set of AC-operators and $\succ$ be a precedence satisfying the extended associative path condition. Then an equational TRS $\mathcal{R}/AC$ terminates if $\mathcal{R}$ is reducing relative to $\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})/\mathcal{T}rans(\succ, \mathcal{F}_{AC})$ and $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$.* ◇

$\boxed{\textbf{3.62}}$ **Theorem** *Let $\mathcal{R}$ be an AC-TRS which is ordered w.r.t. the* AE-RPO. *Then $\mathcal{R}$ is AC-terminating.* ◇

Studying the AE-RPO has led to questions on how to further weaken the extended associative path condition, which still have to be answered.

- Why must the precedence chains w.r.t. the operators of $\mathcal{F}_2$ be linear? For example, consider the precedence



- Is it possible to insert (w.r.t. the precedence) unary operators between two AC-operators f and g, e.g. $f \succ h \succ g$:

$$\left\{\begin{array}{rcl} f(x,h(y)) & \rightarrow & h(f(x,y)) \\ f(h(x),y) & \rightarrow & h(f(x,y)) \\ f(x,g(y,z)) & \rightarrow & g(f(x,y),f(x,z)) \\ f(g(x,y),z) & \rightarrow & g(f(x,z),f(y,z)) \\ h(g(x,y)) & \rightarrow & g(h(x),h(y)) \end{array}\right.$$

  is a canonical system for the above precedence.

- ▶ What about non-AC-operators with an arity greater than 1 which are smaller (w.r.t. the precedence) than AC-operators?

The first question can presumably be solved. Currently, however, we are unable to give a proof. The second problem is solved in [Bachmair, 1987] (on page 89) for one unary operator, at most (and with the transformation system given above). As likely as not, an arbitrary number of unary operators can be involved. However, the third question concerns a somewhat difficult problem (see also historical remark on page 71).

(ii) Integrating a Status Function

Sometimes, a lexicographic status is necessary as illustrated by the following example[13)].

$\boxed{3.63}$ **Example** *Let*

$$\mathcal{R}\colon \begin{cases} x/y & \rightarrow \quad x \star i(y) \\ x/(y/z) & \rightarrow \quad (x/i(z))/y \end{cases}$$
$$\mathcal{E}\colon \quad \star \in \mathcal{F}_{AC} \qquad\qquad \diamond$$

The last rule of $\mathcal{R}$ requires status ⓡ for the division operator /. Thus, the integration of a status function to the AD-RPO (or the AE-RPO) can help proving the termination of $\mathcal{R}$.

In order to compare terms (w.r.t. the RPOS) with operators having variable arity, we do not need to change the definition of the RPOS. However, the deletion property must be satisfied to preserve the property of being a simplification ordering. The RPO (without status) does have the deletion property (see [Dershowitz, 1982]). Unfortunately, the RPOS does not preserve this characteristic. Consider a simple example: Let $\tau(f) = ①, s = f(x,x,y)$ and $t = f(x,y)$. Due to the deletion property, s must be greater than t, but the comparison of the tuples (x,x,y) and (x,y) requires the orientation of $x = y$ which is impossible.

Thus, in order to guarantee the property of a simplification ordering, the status function of the RPOS must satisfy the multiset status of each varyadic operator. Consequently, each AC-operator necessarily has status ⓜ.

> **Idea of Comparison** *The method of comparing terms w.r.t. the* AE-RPOS *is identical to that of the* AE-RPO *presuming status* ⓜ *of each AC-operator.*

$\boxed{3.64}$ **Definition (Associative RPOS Based on Extended Distributivity)** *Let $\succ$ be a precedence satisfying the extended associative path condition, τ a status function such that $\tau(f)$=ⓜ if $f \in \mathcal{F}_{AC}$, γ: $\mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{T}(\mathcal{F},\mathcal{X})$ be an operation which rewrites a term to an irreducible form w.r.t. the TRS $D^E(\succ)$. Then the* associative recursive path ordering AE-RPOS with A.19
status based on extended distributivity *is defined as* A.37

$$\succ_{\mathsf{AE\text{-}RPOS}} \;\equiv\; (\succ_{\mathsf{RPOS}}\langle^{-} \circ \gamma\rangle\, , \xrightarrow{+}_{D^E(\succ)/A}\langle \mathcal{I}d\rangle) \qquad \diamond$$

As in the case of the AE-RPO, the following lemma holds for the AE-RPOS.

[13)] In most cases, lexicographic status is needed for rules representing associativity, only (see section 7.3).

$\boxed{3.65}$ **Lemma** *Let $\mathcal{R}$ be an AC-TRS which is ordered w.r.t. the* AE-RPOS*. Then $\mathcal{R}$ is AC-terminating.* $\diamond$

(iii) Decomposition Orderings Modulo AC-Theories

We are now dealing with the application of the previous concepts to the strongest path and decomposition ordering of this chapter (i.e. to the IRDS, see section 6.1). The AE-RPOS is based on the RPOS. Our goal consists of extending this strategy by strengthening the basic ordering. More precisely, the RPOS will be replaced with a more powerful ordering. In section 6.1, the hierarchy w.r.t. the power of path and decomposition orderings will be given. The IRDS is the strongest one. Therefore, we checked whether it is applicable to the method of the AE-RPOS (and thus also to the AE-RPO, the AD-RPO and the A-RPO). It turned out that there are no additional restrictions to those of the RPOS needed.

We once more have to point out that quasi-precedences on AC-operators are forbidden.

Idea of Comparison *The* AE-IRDS *is identical to the* AE-RPOS *with the exception of the* IRDS *being used instead of the* RPOS *as the basic ordering.*

$\boxed{3.66}$ **Definition (Associative IRDS Based on Extended Distributivity)** *Let $\succ$ be a precedence satisfying the extended associative path condition, τ a status function such that*
A.10 *$\tau(f) = ⓜ$ if $f \in \mathcal{F}_{AC}$, γ: $\mathcal{T}(\mathcal{F},\mathcal{X}) \mapsto \mathcal{T}(\mathcal{F},\mathcal{X})$ be an operation which rewrites a term to an irreducible form w.r.t. the TRS $D^E(\succ)$. Then the* associative improved recursive decomposition ordering AE-IRDS with status based on extended distributivity *is defined as*

$$\succ_{\text{AE-IRDS}} \;\equiv\; (\succ_{\text{IRDS}}\langle^{-} \circ \gamma\rangle \,,\, \xrightarrow{+}_{D^E(\succ)/A}\langle \mathcal{I}d\rangle) \qquad \diamond$$

The following lemma reflects the correctness of the promising manipulation of the AC-compatible orderings based on the RPOS.

$\boxed{3.67}$ **Lemma** *Let $\mathcal{R}$ be an AC-TRS which is ordered w.r.t. the* AE-IRDS*. Then $\mathcal{R}$ is AC-terminating.* $\diamond$

Lemma 3.67 guarantees the correctness of merging the concepts of the various versions of the associative path orderings with the IRDS. In order to complete this study let us point out that it is possible to apply this method to all other path and decomposition orderings of this chapter. Beside their greater power, path and decomposition orderings have another advantage to the RPOS: The construction of a precedence for a given TRS is easier (see the following section).

Delor and Puel provided some generalizations of associative path orderings by further weakening the restrictions on the precedence (see [Delor and Puel, 1993]) by admitting

(i) ▶ $f_1 \succ h_1 \succ h_2 \succ \cdots \succ h_m \succ f_2$ *(see also [Bachmair, 1987])*
▶ $f_1 \succ f_2 \succ h_1$

(ii)
- $f_1 \succ f_2 \succ \cdots \succ f_n$
- $\forall i \in [2, n]$: $f_i \succ f_1$
- $f_1 \succ h_1 \succ h_2 \succ \cdots \succ h_{m-1} \succ f_2 \succ f_3 \succ \cdots \succ f_n \succ h_m$

where f_i are AC-operators and h_i are unary operators. In order to weaken the restrictions on the precedence, Delor and Puel either extend the distributivity rules for ensuring confluence (whenever the precedences are of one of the forms given in (i) above) or if this is not possible, they use a specific strategy by giving priorities to the rules by specifying the order they are to be applied (for the precedences given in (ii) above) which leads to unique normal forms.

3.68 Remark

In [Kapur *et al.*, 1990], a new method for proving AC-termination has been developed. It extends the RPOS by *partitioning* the arguments of AC-operators into disjoint subsets each of which can only be used once for a comparison with other subsets. They introduced two new techniques – *pseudocopying* ($ff(t_1,\ldots,t_n)$ is a pseudo-copy of $f(t_1,\ldots,t_n)$ where the copy is 'very slightly smaller' than the original term) and *elevating* (replacing a term, possessing a leading operator smaller than AC-operators, by one of its arguments) in order to ensure transitivity. This technique imposes no restrictions on the underlying precedence.

In [Bachmair, 1992], a different presentation of the same ordering as in [Kapur *et al.*, 1990] is provided by a schematic TRS while [Kapur *et al.*, 1990] uses a specific algorithm for defining the ordering.

The so-called binary path condition BPC of [Dershowitz and Mitra, 1992] can be seen as a preprocessor for the ordering defined in [Kapur *et al.*, 1990] and [Bachmair, 1992]. The BPC is based on the RPO while comparing the arguments of AC-operators componentwise rather than as multisets (which is similar to the idea of [Kapur *et al.*, 1990]). One can prove that the BPC is well-founded and AC-compatible. However, it is not transitive, in general. Nevertheless, it is easy to implement (much easier than the ordering of [Kapur *et al.*, 1990] or [Bachmair, 1992]) and provides a sufficient condition for the ordering of [Kapur *et al.*, 1990] and [Bachmair, 1992].

Recently, in [Rubio and Nieuwenhuis, 1993], an AC-compatible modification of the RPOS has been presented which is total on non-AC-equivalent ground terms. They use a so-called *interpretation* I which is defined as the normal form of a term w.r.t. the rule set $\mathcal{R}_F \cup \mathcal{R}_I$ under the strategy (i) leftmost-innermost and (ii) rules of $\mathcal{R}_F$ first. $\mathcal{R}_F$ represents the rules describing the flattening operation and $\mathcal{R}_I$ contains rules which transform terms of the form $f(\cdots, g(t_1,\ldots,t_n), \cdots)$ with $f \succ g$ and f associative-commutative into simpler ones. Terms are compared by comparing their interpretations and if they are equal, by comparing the interpreted arguments w.r.t. the status extension of the method. Unfortunately, "in many cases the orientations obtained are different from the usual ones" (quote from [Rubio and Nieuwenhuis, 1993]). For example, $(x \star y) + (x \star z) \succ_{\mathsf{RN}} x \star (y + z)$ holds although $\star \succ +$.

3.4 Automatic Generation of the Orderings

The problem of ensuring that a TRS $\mathcal{R}$ terminates is reduced to the problem of choosing an appropriate reduction (σ-stable simplification) ordering $\succ_{\mathsf{T}}$. In particular, constructing $\mathcal{R}$ in an automatic fashion consists of automatically finding $\succ_{\mathsf{T}}$. Selecting $\succ_{\mathsf{T}}$ based on a static analysis of the equations or rules is not sufficient, because the set of equations or rules grows, for example, during the completion process. The selection of $\succ_{\mathsf{T}}$ must proceed dynamically:

Whenever an equation or a rule is encountered that cannot be oriented using $\succ_{\mathsf{T}}$, $\succ_{\mathsf{T}}$ must be extended (if possible) so that it can orient the new equation or proving the termination of the new rule and also orient all previously oriented equations or rules. As hinted above, there exist two problems:

- A set $\mathcal{E} = \{s_i \leftrightarrow t_i \mid i \in [1,n]\}$ of equations is given. Choose an ordering $\succ_{\mathsf{T}}$ such that $s_i \succ_{\mathsf{T}} t_i$ or $t_i \succ_{\mathsf{T}} s_i$, for all i.
- A set $\mathcal{R} = \{l_i \rightarrow r_i \mid i \in [1,n]\}$ of rules is given. Choose an ordering $\succ_{\mathsf{T}}$ such that $l_i \succ_{\mathsf{T}} r_i$, for all i (i.e. prove the termination of $\mathcal{R}$).

In this section, we summarize the main points of several techniques for generating adequate path and decomposition orderings. These orderings depend on a precedence and a status function. Thus, the choice of an appropriate ordering is reduced to the choice of an appropriate precedence and an appropriate status function. Since we deal with finite sets of operators, there exist decision procedures solving this problem (for example, by enumerating all possible precedences and status functions and then, selecting appropriate combinations). Mostly, one is interested in finding a minimal ordering (i.e. minimal w.r.t. the 'size' of the precedence). All methods contained in the literature follow this sensible[14] restriction.

For the RPO*, Aït-Kaci developed a method in 1984 (published as [Aït-Kaci, 1985]). At the same time, Forgaard has presented his technique for generating an appropriate* RPOS *([Forgaard, 1984], [Detlefs and Forgaard, 1985]). This technique has been implemented in the REVE-system. The same has been done with a procedure for producing an adequate* RDOS*, developed by Lescanne and Choque ([Lescanne, 1983a], [Choque, 1984]).*

> It is essential that the ordering is $\succ$-monotonic (i.e. monotonic w.r.t. the precedence), so that extending the precedence does not change the ordering of previously oriented equations.

A formal definition of the $\succ$-monotonicity can be found in section 6.1. In any case, the RPOS and the RDOS are $\succ$-monotonic (see section 6.1).

We briefly describe the method of [Forgaard, 1984] for the RPOS in an informal way. Forgaard gave an alternative definition of the RPOS (called EPOS, *extensible path ordering with status*). The definition of the EPOS is based on the following two features:

- A new type of status, called ⓤ (for undefined) is introduced. The treatment of ⓤ is the conjunction of the treatment given to ⓜ, ⓛ and ⓡ. This allows to initialize the status of all operators to ⓤ and then to incrementally choose status assignments for operators as needed.
- The two cases $f \succ g$ and $f \sim g$ (where f and g are the leading operators of the terms to be compared) are folding up to $f \succsim g$.

The algorithms of [Forgaard, 1984] and [Aït-Kaci, 1985] are similar (with the exception that the one of [Forgaard, 1984] additionally integrates a status function). The basic concept can be described by the following pseudo-Pascal procedure:

---

[14] This restriction is sensible because the number of extensions of a *minimal* precedence is greatest.

```
$M := \{(\emptyset, \{\tau(f) = \textcircled{u} \mid f \in \mathcal{F}\})\}$
for $l_i \to r_i \in \mathcal{R}$ do
    $M' := \emptyset$
    for $(\succ, \tau) \in M$ do
        $M := M \setminus \{(\succ, \tau)\}$
        $M' := M' \cup \text{gen-ext}(l_i, r_i, (\succ, \tau))$                    $(*)$
    endfor
    $M := M'$
endfor
```

M and M' are sets of tuples $(\succ, \tau)$ consisting of a precedence $\succ$ and a status function τ. At the end of the procedure, M is either empty (i.e. $\mathcal{R}$ cannot be oriented with the help of any EPOS) or contains sets of minimal precedences and status functions for proving the termination of $\mathcal{R}$. The $(*)$ line represents the generation of an extension of the actual precedence and status function such that $l_i \succ_{\mathsf{EPOS}} r_i$. The generation of this extension will be performed according to the definition of the EPOS. The two terms (to be compared) will be processed symbol by symbol (from their roots to their leaves) in the following way. Let $l_i = f(s_1, \ldots, s_m)$ and $r_i = g(t_1, \ldots, t_n)$:

- If f and g have already a connection w.r.t. $\succ$, then continue according to the definition of the EPOS.
- Otherwise, check whether the possible extensions
    - $f \succ g$
    - $f \sim g \wedge (\tau(f) = \tau(g) = \textcircled{u} \vee \tau(f) = \textcircled{m} = \tau(g) \vee \tau(f), \tau(g)$ are lexicographic types of status)
    - $g \succ f$
    - and combinations of the preceding ones

  of $(\succ, \tau)$ lead to the comparison of l_i and r_i (by recursively applying the function *gen-ext* to the arguments of l_i and r_i). Note that for each successful extension, a new set will be constructed and added to M' in the main procedure.

The above description of the technique of [Forgaard, 1984] is an informal one containing only the essential principles and thus, making no pretense to completeness. The differences between the approaches of [Forgaard, 1984] and [Aït-Kaci, 1985] are the following ones:

criterion	[Forgaard, 1984]	[Aït-Kaci, 1985]
input	set of equations	set of rules
status	yes (including $\textcircled{u}$)	no
set of minimal registries	explicit construction	construction of one successful minimal registry (by backtracking)
keeping rejected extensions	no	yes

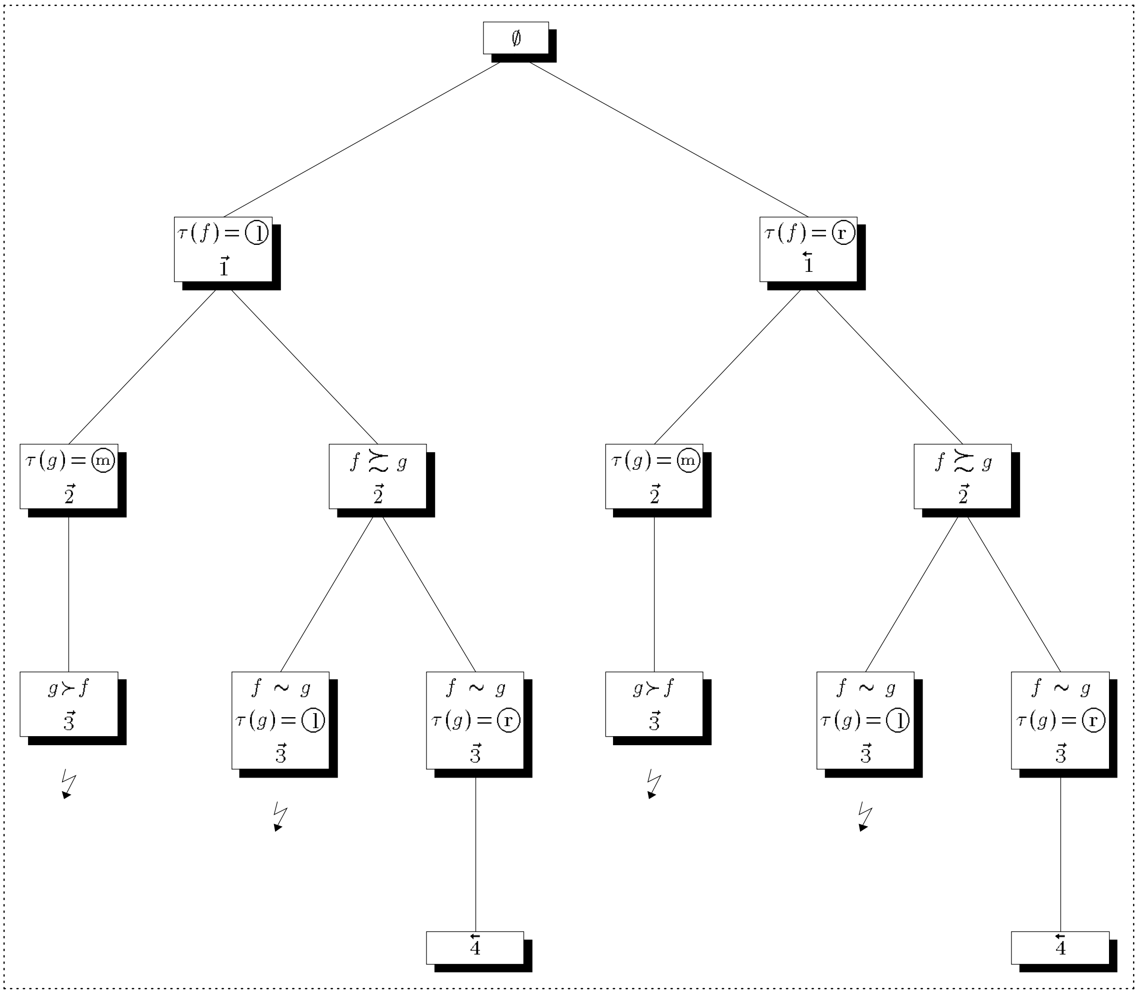
$\boxed{3.69}$ **Example ([Forgaard, 1984])** *We want to generate minimal registries for the* EPOS (RPOS) *in order to orient the equations of the system*

$$\left\{\begin{array}{rrcl} 1: & f(f(x,x),y) & = & f(x,f(x,y)) \\ 2: & f(g(y,x),y) & = & g(x,y) \\ 3: & g(f(y,x),y) & = & f(x,y) \\ 4: & g(g(x,x),y) & = & g(y,g(x,y)) \end{array}\right.$$

We start with an empty precedence and a status function where all operators have undefined status. The different extensions of this empty registry can be illustrated by a tree given below. In this tree, each node together *with its predecessors represents a registry and the corresponding orientations of the equations (the orientation is indicated by an arrow above the number of the equation).*

(i) The first equation is not orientable w.r.t. an empty registry. Thus, we compute the set of minimal extensions (by applying the function gen-ext): $M_1 = \{(\emptyset, \tau(f) = \textcircled{l})\}$ *inducing the orientation of equation 1 from left to right,* $M_2 = \{(\emptyset, \tau(f) = \textcircled{r})\}$ *inducing the orientation of equation 1 from right to left.*

(ii) The second equation is orientable neither w.r.t. M_1 *nor w.r.t.* M_2*. For this equation the following extensions of* M_1 *and* M_2 *exist:* $M_{11} = M_1 \cup \{(\emptyset, \tau(g) = \textcircled{m})\}, M_{12} = M_1 \cup \{(f \succsim g, \emptyset)\}, M_{21} = M_2 \cup \{(\emptyset, \tau(g) = \textcircled{m})\}$ *and* $M_{22} = M_2 \cup \{(f \succsim g, \emptyset)\}$*. All sets imply the equation to be oriented from left to right.*



(iii) *The third equation cannot be oriented w.r.t. any of the previously produced registries. The following extensions are sufficient for orienting equation 3 from left to right:* $M_{111} = M_{11} \cup \{(g \succ f, \emptyset)\}, M_{121} = M_1 \cup \{(f \sim g, \tau(g) = \textcircled{l})\}, M_{122} = M_1 \cup \{(f \sim g, \tau(g) = \textcircled{r})\}, M_{211} = M_{21} \cup \{(g \succ f, \emptyset)\}$, $M_{221} = M_2 \cup \{(f \sim g, \tau(g) = \textcircled{l})\}$ *and* $M_{222} = M_2 \cup \{(f \sim g, \tau(g) = \textcircled{r})\}$.

(iv) *The only possible way to orient equation 4 w.r.t. the* EPOS (RPOS) *is from right to left. In order to do this, the status of the operator* g *must be* $\textcircled{r}$. *Thus, the registries* M_{111}, M_{121}, M_{211} *and* M_{221} *cannot be chosen. Finally, the registries* $M_{1221} = M_{122} = \{(f \sim g, \tau(f) = \textcircled{l}, \tau(g) = \textcircled{r})\}$ *and* $M_{2221} = M_{222} = \{(f \sim g, \tau(f) = \textcircled{r} = \tau(g))\}$ *are the only parameters such that the induced* EPOS (RPOS) *is able to orient the given equations.* ◇

The conceptual idea of the computation of minimal registries for *decomposition orderings* (see [Lescanne, 1983a] and [Choque, 1984]) is nearly identical to that for the RPOS. Additionally, it is easier since particular extensions of the precedence are not as closely related. The reasons for this are the definitions of the two orderings.

> For example, let us compare the term $f(g(x))$ with another one. In case of the RPOS, the position of the operator g within the precedence strongly depends on that of the operator f. Assume we have to compare $f(g(x))$ and $h(i(x))$. Let $g \succ h$ and $f \succ i$. Thus, there is no chance to use f for covering i (although $f \succ i$) since f will be the first operator that is removed from $f(g(x))$. However, if decomposition orderings are used, the operator f does not possess a higher priority than g (only in the $\sim$-case, symbols which are nearer to the root are 'greater').

This benefit of decomposition orderings (in contrast to the RPOS) is exploited by Lescanne in his REVE system. REVE combines the orderings RPOS and RDOS in such a way that the RDOS is called for help when the RPOS fails. The RDOS then provides suggestions that the RPOS needs.

Note that most of the algorithms – implemented in an ad hoc manner – for computing all minimal registries probably require time that is exponential in the number of operators in the terms to be compared. The following lemma underlines this conjecture.

3.70 **Lemma** *The number of all possible (consistent) total precedences of a quasi-ordering* $\succsim$ *for* $n > 1$ *operators is*

$$\sum_{k=1}^{n} k! \cdot F(k,n)$$

$$\text{with } F(k,n) = \begin{cases} 1 & \text{if } k = 1 \ \vee \ k = n \\ k \cdot F(k, n-1) + F(k-1, n-1) & \text{otherwise} \end{cases}$$ ◇

The expression $F(k,n)$ represents the number of partitions of k sets including n different elements, i.e. $M_1 ... M_k$ without any order of succession and $|\bigcup_{i=1}^{k} M_i| = n$. For computing all permutations of k sets, $F(k,n)$ must be multiplied by $k!$ (see the formula of lemma 3.70).

3.71 **Example** *There exist 13 different total quasi-precedences if the signature contains 3 operators* f, g, h:

(i) $f \succ g \succ h \quad f \succ h \succ g \quad g \succ f \succ h \quad g \succ h \succ f \quad h \succ f \succ g \quad h \succ g \succ f \qquad (6 \cdot F(3,3))$

(ii) $f \sim g \succ h \quad h \succ f \sim g$
$f \sim h \succ g \quad g \succ f \sim h$
$g \sim h \succ f \quad f \succ g \sim h \qquad (2 \cdot F(2,3))$

(iii) $f \sim g \sim h \qquad (F(1,3))$ ◇

Note that there exists a close connection between $F(k, n)$ and the Bell numbers of example 1.2. The following table illustrates the complexity of the function given in lemma 3.70.

n	$\sum k! \cdot F(k,n)$
2	3
3	13
4	75
5	541
6	4683
7	47293

Concluding, we point out that the *empirical* time complexity of the presented techniques can slightly be improved in the following way.

3.72 Remark

Since the RPOS (RDOS) is a simplification ordering, the rules of the form $l \rightarrow r$ (the equations of the form $l = r$, respectively) where $l \succ_{\mathsf{EMB}} r$ (or $r \succ_{\mathsf{EMB}} l$) do not need to be considered.

It is obvious that the technique for generating minimal registries for the RPOS (and particularly for the RDOS) can also be used for the other path and decomposition orderings. Further, note that this approach can easily be extended for the AC-case (i.e. for the various types of associative path orderings). This can be realized by additionally testing the constraints, a precedence must satisfy, given by the definitions of these orderings.

Chapter 4

Knuth-Bendix Orderings

This chapter deals with the well-known weight-based ordering of Knuth and Bendix (see [Knuth and Bendix, 1967]). In subsection 1.5.3, the notion of weights has been introduced. It is obvious that each weight function is a special linear interpretation (see subsection 1.5.4). Thus, one may wonder why Knuth-Bendix orderings are not treated together with polynomial orderings. The reason for dedicating an entire chapter to Knuth-Bendix orderings is twofold: (i) The structure and the parameters of Knuth-Bendix orderings are simpler than those of polynomial orderings and thus, Knuth-Bendix orderings suit for motivating and leading to polynomial orderings. (ii) On the other hand, the power of the orderings presented in this chapter overlaps with that of some kinds of polynomial orderings.

The contents of this chapter are the following ones.

- ▶ **Section 4.1:** The definition of the Knuth-Bendix ordering will be given including a brief discussion on its central features.

- ▶ **Section 4.2:** We provide an improvement of the Knuth-Bendix ordering based on weakening the conditions on variables and on unary operators as well as on integrating a status function.

- ▶ **Section 4.3:** This section deals with the application of the Knuth-Bendix ordering to AC-theories. In subsection 4.3.1, we introduce a restricted variant of the Knuth-Bendix ordering based on a weight function and a precedence satisfying specific limitations. An improved version of this AC-commuting Knuth-Bendix ordering, which takes the memory-flattening operation into account, is given in subsection 4.3.2. Finally, subsection 4.3.3 contains a C-commuting Knuth-Bendix ordering.

- ▶ **Section 4.4:** There exists a decision procedure for the generation of the original Knuth-Bendix ordering such that it can prove the termination of a given TRS (see [Martin, 1987], [Dick *et al.*, 1990]). In this section, we give an answer to the question how to apply this technique (especially a slight improvement of it) to the modified Knuth-Bendix orderings of sections 4.2 and 4.3.

4.1 Definition of the Knuth-Bendix Ordering

In the previous chapter, we have investigated, simplified as well as extended various syntactic orderings. The following two chapters deal with semantic orderings. First of all, we consider the well-known Knuth-Bendix ordering based on a weight function introduced in subsection 1.5.3. As shown in [Knuth and Bendix, 1967], this weight function can be slightly generalized in the following way.

$\boxed{4.1}$ **Definition (Extended Weight Function, [Knuth and Bendix, 1967])** *Let $\mathcal{F}$ be a set of operators, $\succ$ be a total precedence and φ be a weight function*[1] *on $\mathcal{F}$. φ can be improved to φ_E – called* extended weight function *– such that*

$$\varphi_E(f) \begin{cases} \geq 0 & \textit{if } \mathcal{A}r(f) = 1 \;\wedge\; \forall g \in \mathcal{F} \setminus \{f\}\colon f \succ g \\ = \varphi(f) & \textit{otherwise} \end{cases}$$

For simplicity, we use φ instead of φ_E. ◇

Idea of Comparison *The Knuth-Bendix ordering* KBO *takes $(\mathbb{N}_0, >)$ as the underlying well-founded set, i.e. it assigns natural (or possibly real) numbers to the operators and then to terms (called weight of a term) by adding the numbers of the operators they contain. Two terms are compared by verifying the variable condition '$\forall x \in \mathcal{X}\colon \geq\langle|.|_x\rangle$' of the* KBO *and by comparing their weights. The identity of the weights implies the leading operators to be compared w.r.t. the precedence. If their weights as well as their leading operators are equal, the arguments are lexicographically compared.*

$\boxed{4.2}$ **Definition (Knuth-Bendix Ordering, [Knuth and Bendix, 1967])** *Let $\succ$ be a*
A.30 *total precedence and φ be an extended weight function. The* Knuth-Bendix ordering KBO *on terms is defined as*

$$\begin{aligned} \succ_{\mathsf{KBO}} \;&\equiv\; \{\forall x \in \mathcal{X}\colon \geq\langle|.|_x\rangle \;\&\; >\langle\varphi\rangle, \\ &\qquad \forall x \in \mathcal{X}\colon =\langle|.|_x\rangle \;\&\; =\langle\varphi\rangle \;\&\; \{\succ_{\mathsf{V}}\langle\mathcal{I}d\rangle\,,\, (\succ, \succ_{\mathsf{KBO}}^{\textcircled{1}})\}\} \\ \sim_{\mathsf{KBO}} \;&\equiv\; =\langle\mathcal{I}d\rangle \end{aligned}$$

A.9
A.14 *where $\succ_{\mathsf{V}}$ is an ordering on terms s, t defined by $s \succ_{\mathsf{V}} t$ if, and only if, $s = f(f(\cdots f(x)\cdots))$*
A.16 *and $t = x$ for $f \in \mathcal{F}, \mathcal{A}r(f) = 1$.* ◇

► The so-called *variable condition* of the KBO, guaranteeing $\forall x \in \mathcal{X}\colon |s|_x \geq |t|_x$ or $|s|_x = |t|_x$ whenever a term s should be greater than a term t (w.r.t. the KBO), is a somewhat strong restriction. Thus, the number of occurrences of a variable must not increase when
A.8 passing from a larger to a smaller term. Note that, for example, the distributivity rule
A.10 $x \star (y + z) \to (x \star y) + (x \star z)$ cannot be oriented in this (usual) direction. However, the variable condition is necessary for ensuring the σ-stability: By using definition 4.2, we can verify that $h(x) \succ_{\mathsf{KBO}} f(x, x)$ if $\varphi_0 = 1$, $\varphi(f) = 0$ and $\varphi(h) = 2$. Due to

---

[1] see definition 1.40 on page 21

the σ-stability, $h(h(y))$ has also to be greater than $f(h(y), h(y))$ using the substitution $\sigma = \{x \leftarrow h(y)\}$. However, this does not hold. Despite of this, the simple structure of the KBO sometimes allows termination proofs when path and decomposition orderings fail. A.11 A.12 A.30

- The weight function of definition 1.40 requires the weight of a unary operator to be greater than or equal to 1. As stated in [Knuth and Bendix, 1967], it is possible to allow *one*[2] unary operator f with weight zero, at most (see definition 4.1). To guarantee the well-foundedness, all other operators have to be smaller than f w.r.t. the precedence: Otherwise, the non-terminating rule $h(x) \to i(h(x))$ can be oriented using $\varphi(i) = 0$ and $h \succ i$. Admitting a unary operator f with weight zero, we have to guarantee that $f^n(x) \succ_{\mathsf{KBO}} x$ holds (see the subordering $\succ_{\mathsf{V}}$).
  By the way, the condition of the weight φ_0 assigned to variables being smaller than or equal to the weight of each constant is also necessary for ensuring the well-foundedness of the KBO: Note that $h(x) \succ_{\mathsf{KBO}} h(a)$ if $\varphi_0 > \varphi(a)$. However, the rule $h(x) \to h(a)$ is not terminating.
  Lastly, the well-foundedness of the KBO would be violated if constants (and thus, also variables) possess weight zero. Then the rule $f(f(x,x),x) \to f(a, f(f(x,x),x))$ – which is obviously non-terminating – can be oriented with the help of the KBO based on $\varphi(a) = 0$, $\varphi_0 = 0$, $\varphi(f) = 0$ and $f \succ a$.

4.3 **Lemma ([Knuth and Bendix, 1967])** *The* KBO *is a σ-stable simplification ordering.* ◇

4.2 An Extension of the Knuth-Bendix Ordering

This section deals with an extension of the KBO. To make the KBO more powerful, we have to realize its short-comings by studying certain examples. In addition, we analyze the definition of the KBO to point out its weaknesses. The following rules are intuitively decreasing but the KBO does not guarantee their termination:

- We want to prove the termination of the rule

  $$s = (x \star y) + y \;\to\; (x+1) \star y = t \tag{4.1}$$

  with the help of the KBO. Therefore, we have to verify that $\varphi(s) \geq \varphi(t)$. This implies that $\varphi_0 \geq \varphi(1)$ must hold. Since the weight of a constant must be greater than or equal to φ_0, the weights of the variables and of the constant 1 have to be the same: $\varphi_0 = \varphi(1)$. On this premise, s and t have the same weight. As the multisets of the variables of the two terms are not identical, s can never be greater than t (according to the definition of the KBO).

  One way to establish the termination of (4.1) is to weaken the variable condition $\forall x \in \mathcal{X}$: $=\langle|.|_x\rangle$ of the KBO by $\forall x \in \mathcal{X}$: $\geq\langle|.|_x\rangle$.

- We would like to show the termination of the rule

  $$s = x \star ((-y) \star y) \;\to\; (-(y \star y)) \star x = t \tag{4.2}$$

---

[2] since the precedence has to be total

with the help of the KBO. The weights and the leading operators of s and t are equal (irrespective of the weight function). This situation demands that $x \succ_{\mathsf{KBO}} -(y \star y)$ which cannot be valid since the variable condition is infringed.

However, $(-y) \star y \succ_{\mathsf{KBO}} -(y \star y)$ if $\star \succ -$. In addition to the fact that x is a subterm of s as well as of t, a possible way to guarantee the termination of (4.2) is to compare the arguments of s and t as multisets instead of lexicographically.

▶ The termination of the rule

$$(x \star y)^2 \rightarrow x^2 \star (-y)^2 \tag{4.3}$$

cannot be proved by any KBO. The reason for this assertion is that two unary operators ($-$ and ^{2}) with weight zero must exist since $\varphi(x) + \varphi(\star) + \varphi(y) + \varphi(^2)$ has to be greater than or equal to $\varphi(x) + \varphi(^2) + \varphi(\star) + \varphi(-) + \varphi(y) + \varphi(^2)$ which is equivalent to the fact that $\varphi(-) + \varphi(^2)$ must be less than or equal to zero. This requirement is not allowed (see definition 4.1).

Using a quasi-ordering on the operators instead of a total ordering without equivalences on different operators, we may allow more than one unary operator with weight zero. On the premise that all these operators are equivalent w.r.t. the precedence, the induced extension of the KBO also is a σ-stable simplification ordering. Then (4.3) can be ordered w.r.t. the parameters $\varphi(-) = \varphi(^2) = 0$ and $- \sim {}^2 \succ \star$.

The different aspects of the analysis of the original KBO lead to a new and *more powerful* definition.

Idea of Comparison *The comparison of two terms w.r.t. the Knuth-Bendix ordering with status is identical to that of the* KBO *with the following three exceptions: (i) A quasi-precedence is used instead of a total one. (ii) The variable conditions '*$\forall x \in \mathcal{X}\colon \geq\langle|\cdot|_x\rangle$*' and '*$\forall x \in \mathcal{X}\colon =\langle|\cdot|_x\rangle$*' have been merged into '*$\forall x \in \mathcal{X}\colon \geq\langle|\cdot|_x\rangle$*'. (iii) A status function has been integrated.*

4.4 Definition (Knuth-Bendix Ordering with Status) *Let $\succsim$ be a quasi-precedence,*
A.18 *τ be a status function and φ be an extended weight function satisfying $h \sim i$ if $h, i \in \mathcal{F} \wedge \mathcal{A}r(h) = \mathcal{A}r(i) = 1 \wedge \varphi(h) = \varphi(i) = 0$. The* Knuth-Bendix ordering KBOS with status *on terms is defined as follows:*

$$\begin{array}{lcl} \succ_{\mathsf{KBOS}} & \equiv & \forall x \in \mathcal{X}\colon \geq\langle|\cdot|_x\rangle \;\;\&\;\; \{\succ_{\mathsf{V}}\langle\mathcal{I}d\rangle\,,\ (>\langle\varphi\rangle, \succ, \succ^{\tau}_{\mathsf{KBOS}})\} \\ \sim_{\mathsf{KBOS}} & \equiv & \approx\langle\mathcal{I}d\rangle \end{array}$$ ◇

A.25
A.31 We conclude this section with the enumeration of important properties of the KBOS.

4.5 Theorem *The* KBOS *is a σ-stable simplification ordering.* ◇

In [Martin, 1993], an extension of the KBO *was developed, providing in general continuum many distinct new orderings with a given choice of Knuth-Bendix weight. This method* *is constructed by a lexicographic concatenation of several orderings. The first component represents the natural ordering* $>$ *on the weights of the terms. In the case of the weights are equal, vectors of numbers of certain patterns (called* zig-zags*) are compared. For example, the zig-zag* $f \xrightarrow{1} f$ *denotes an* f *occurring in a term with a further* f *occurring somewhere in its first argument (thus,* $f(f(a,a),a)$ *has* one *occurrence of the zig-zag and is greater than* $f(a, f(a,a))$ *since it has* none *occurrence of the zig-zag).*

4.3 AC-Case

In this section, we introduce three new classes of orderings, called *associative-commutative Knuth-Bendix orderings*, for proving the termination of associative and (or) commmutative TRSs. The first ordering is an AC-commuting σ-stable simplification ordering such that theorem 1.27 on page 12 can be applied for proving AC-termination. The second ordering is an extension of the first one and it is based on the memory-flattening operation introduced in subsection 1.5.5. Finally, a simple C-commuting KBO is provided.

4.3.1 Associative-Commutative Knuth-Bendix Ordering

The KBOS is a simplification ordering. Unfortunately, it is not AC-commuting. Consider the following examples:

- $$\begin{array}{lcl} s' = (--x) \star x & \leftrightarrow_{AC} & x \star (--x) = s \\ \quad \succ_{\mathsf{KBOS}} & & \\ t' = (-x) \star (-x) & & \end{array}$$

  if $\star \in \mathcal{F}_{AC}$ and $\tau(\star) = ①$. Assuming the AC-commutation of the KBOS, a term t AC-equivalent to t' must exist which is simpler (w.r.t. the KBOS) than s. The only possible term t is the term t' itself (because no other term exists which is AC-equivalent to t'). But s is not greater than t' (w.r.t. the KBOS). On the contrary, $t' \succ_{\mathsf{KBOS}} s$.

  The term s would be greater than t' if the KBOS were used and the status of the multiplication operator $\star$ were of type multiset. All other relations would be left unchanged. Taking this idea ($\tau(f) = \textcircled{m}$ if $f \in \mathcal{F}_C$) into account, let us consider two further examples.

- $$\begin{array}{lcl} s' = (x \vee y) \vee z & \leftrightarrow_{AC} & x \vee (y \vee z) = s \\ \quad \succ_{\mathsf{KBOS}} & & \\ t' = \neg(x \barwedge y) \vee z & & \end{array}$$

  if $\vee \in \mathcal{F}_{AC}$ and $\varphi(\vee) = \varphi(\neg) + \varphi(\barwedge)$, $\vee \succ \neg$ and $\tau(\vee) = \textcircled{m}$. The term t could be either t' or $z \vee \neg(x \barwedge y)$. Unfortunately, $s \not\succ_{\mathsf{KBOS}} t$. The crux of the comparison of s and t is that we have to apply the definition of the KBOS recursively to the arguments of the two terms: There is no subterm of s which is greater than $\neg(x \barwedge y)$.

  Requiring $\varphi(s') > \varphi(t')$ would avoid the recursive application of the ordering. More precisely, $s' \succ_{\mathsf{KBOS}} t'$ and $s \succ_{\mathsf{KBOS}} t$ if $\varphi(\vee) > \varphi(\neg) + \varphi(\barwedge)$. Another solution is to forbid that $s' \succ_{\mathsf{KBOS}} t'$ holds by requiring $\vee$ (all A-operators, respectively) to be minimal w.r.t. the precedence$^{3)}$.

[3)] This requirement allows us to specify a weight function ensuring the condition $\varphi(\vee) > \varphi(\neg) + \varphi(\barwedge)$ which implies $s' \succ_{\mathsf{KBOS}} t'$ and $s \succ_{\mathsf{KBOS}} t$.

▶
$$\begin{array}{lcl} s' = ((-\,-\,x) \star x) \star y & \leftrightarrow_{AC} & (-\,-\,x) \star (x \star y) = s \\ \quad \succ_{\mathsf{KBOS}} & & \\ t' = ((-x) \star (-x)) \star y & & \end{array}$$

if $\star \in \mathcal{F}_{AC}$ and $\tau(\star) = ⓜ$. The term t could either be t' or $(-x)\star((-x)\star y)$ if we consider the multiset status of $\star$. Therefore, we must show that the multiset $\{\!\!\{\, -\,-\,x, x \star y \,\}\!\!\}$ is greater than either $\{\!\!\{\, (-x) \star (-x), y \,\}\!\!\}$ or $\{\!\!\{\, -x, (-x) \star y \,\}\!\!\}$. However, no term exists in $\{\!\!\{\, -\,-\,x, x \star y \,\}\!\!\}$ which is either greater than $(-x) \star (-x)$ or greater than $(-x) \star y$.

A possibility to deal with this problem is to restrict the terms to be compared to their normal forms w.r.t. the associative theory (i.e. to the flattened versions of the terms).

The following definition of a new Knuth-Bendix ordering for AC-theories summarizes the
A.1| substantial ideas of the KBOS illustrated by the examples.

Idea of Comparison *The* ACK *is identical to the* KBOS *on flattened versions of terms. Additionally, the* ACK *requires some parameters to be restricted: (i) The weight of each A-operator has to be zero, (ii) each A-operator must be minimal w.r.t. the precedence and (iii) multiset status is necessary for each C-operator.*

$\boxed{4.6}$ **Definition (Associative-Commutative Knuth-Bendix Ordering)** *Let φ be an*
A.7| *extended weight function such that $\forall f \in \mathcal{F}_A$: $\varphi(f) = 0$. Furthermore, $\succ$ is a precedence such that $\forall f \in \mathcal{F}_A$: $\nexists g$: $f \succ g$. The status function τ fulfils the condition that $\tau(f) = ⓜ$ holds if $f \in \mathcal{F}_C$. The* associative-commutative Knuth-Bendix ordering ACK *on terms is defined as*

$$\begin{array}{lcl} \succ_{\mathsf{ACK}} & \equiv & \succ_{\mathsf{KBOS}}^{\langle \overline{\ } \rangle} \\ \sim_{\mathsf{ACK}} & \equiv & \sim_{\mathsf{KBOS}}^{\langle \overline{\ } \rangle} \end{array}$$
◇

$\boxed{4.7}$ **Theorem ([Steinbach, 1989c])** *The* ACK *is an AC-commuting σ-stable simplification ordering.* ◇

In [Steinbach, 1989c], this theorem is proved. The proof is omitted in this thesis since the following subsection provides an improvement of the ACK. It is not difficult to prove that the ACK is also AC-compatible since $\leftrightarrow_{AC} \subseteq \sim_{\mathsf{ACK}}$ holds and $\sim_{\mathsf{ACK}}$ is an associated congruence for the ACK (i.e. $\sim_{\mathsf{ACK}} \cdot \succ_{\mathsf{ACK}} \subseteq \succ_{\mathsf{ACK}}$ and $\succ_{\mathsf{ACK}} \cdot \sim_{\mathsf{ACK}} \subseteq \succ_{\mathsf{ACK}}$).

4.3.2 Associative-Commutative Knuth-Bendix Ordering Based on Memory-Flattening

One major disadvantage in some practical applications of the ACK is that the weight of an AC-operator must be zero. This subsection deals with an extension of the ACK overcoming this problem.

A.3| $\boxed{4.8}$ **Example** *Consider the following rule belonging to a TRS given in [Musser, 1980]:*

$$x \wedge y \;\rightarrow\; \mathit{if}(x, y, \mathit{false})$$

Assume that $\wedge$ *is an AC-operator. Then the rule* cannot *be ordered w.r.t. the* ACK *since* $\varphi(\wedge)$ *must be greater than zero*[4]*.* ◇

In subsection 1.5.5, we have introduced a new flattening operation, called *memory-flattening*. It stores the number of A-operators removed to get the flattened version of a term. This property is closely connected to the reason for the ACK to require weight zero of an A-operator:

> Let $f \in \mathcal{F}_A$, $\varphi(f) > 0$ and $s = f(s_1,\dots,s_n)$, $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ such that $\mathcal{H}ead(t) \neq f$, $|s|_f = |t|_f$ and $s \succ_{\mathsf{ACK}} t$ since $\varphi(\overline{s}) > \varphi(\overline{t})$. Then – according to the $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility – $f(\dots,s,\dots) \succ_{\mathsf{ACK}} f(\dots,t,\dots)$ is required. Note that we have to remove the leading operator f of s whereas t is left unchanged at the top most position. Thus, it is possible that $\varphi(\overline{f(\dots,s,\dots)}) < \varphi(\overline{f(\dots,t,\dots)})$ holds. In order to exclude this situation, the ACK sets the weight of each A-operator to zero. However, by keeping the information on the number of f's (s is composed of) in the leading operator f of $f(\dots,s,\dots)$, $\varphi(f)$ can arbitrarily be chosen.

Idea of Comparison *The* ACKM *is identical to the* KBOS *on memory-flattened terms and is based on the following parameter restrictions: (i)* $\succsim$ *is a quasi-ordering on the enriched signature* $\mathcal{F}^E$ *satisfying* $f^i \sim f$ *for all* $f \in \mathcal{F}$ *and* $i \in \mathbb{N}$*, (ii) each A-operator has to be minimal w.r.t. the precedence, (iii) each A-operator* f^i *must possess* $i{\cdot}\varphi(f)$ *as its weight and (iv) multiset status is necessary for each C-operator.*

$\boxed{4.9}$ **Definition (ACK Based on Memory-Flattening)** *Let* $\succ$ *be a precedence on* $\mathcal{F}$*, extended to a quasi-precedence* $\succsim$ *on* $\mathcal{F}^E$ *such that* $\forall i \in \mathbb{N}$: $f^i \sim f$*. Furthermore,* $\succsim$ *satisfies the condition* $\forall f \in \mathcal{F}_A$: $\nexists g$: $f \succ g$*. Let* φ *be an extended weight function such that* $\forall f^i \in \mathcal{F}^E$: $\varphi(f^i) = i \cdot \varphi(f)$*. The status function* τ *fulfils the condition that* $\tau(f) =$ ⓜ *holds if* $f \in \mathcal{F}_C$*.* |A.3 *The* ACK based on the memory-flattening operation*, called* ACKM*, on terms is defined as*

$$\begin{aligned} \succ_{\mathsf{ACKM}} &\equiv \succ_{\mathsf{KBOS}\langle\sim\rangle} \\ \succsim_{\mathsf{ACKM}} &\equiv \succsim_{\mathsf{KBOS}\langle\sim\rangle} \end{aligned}$$ ◇

$\boxed{4.10}$ **Theorem** *The* ACKM *is an AC-commuting* σ*-stable simplification ordering.* ◇

As we have seen, the ACKM is an ordering which can prove the termination of TRSs modulo an associative and a commutative theory. This theoretical aspect is the foundation for using |A.1 the ACKM in practice. However, from a practical point of view, the ACKM is inefficient since |A.7 the terms to be compared must be flattened. Subsequently, we will present a modified version of the ACKM which improves its applicability.

Comparing two terms w.r.t. the ACKM, we do not always have to flatten them. This expense will only be necessary if the arguments of both terms must be compared. The following definition is mainly based on this fact.

---

[4] Note that $\varphi(\mathit{false}) > 0$ holds since '*false*' is a constant.

Idea of Comparison *The* IACK *works as the* ACK *with the exception that the terms will be 'memory-flattened' only if their weights as well as their leading operators are identical.*

$\boxed{4.11}$ **Definition (Improved Associative-Commutative Knuth-Bendix Ordering)**
Let $\succ$ *be a precedence on* $\mathcal{F}$, *extended to a quasi-precedence* $\succsim$ *on* $\mathcal{F}^E$ *such that* $\forall i \in \mathbb{N}$: $f^i \sim f$.
A.1 *Furthermore,* $\succsim$ *satisfies the condition* $\forall f \in \mathcal{F}_A$: $\nexists g$: $f \succ g$. *Let* φ *be an extended weight func-*
A.3 *tion such that* $\forall f^i \in \mathcal{F}^E$: $\varphi(f^i) = i \cdot \varphi(f)$. *The status function* τ *fulfils the condition that*
A.7 $\tau(f) = ⓜ$ *holds if* $f \in \mathcal{F}_C$. *The* improved ACK, *called* IACK, *on terms is defined as*

$$\begin{array}{lcl} \succ_{\mathsf{IACK}} & \equiv & \forall x \in \mathcal{X}\colon \geq\langle|\cdot|_x\rangle \ \ \& \ \ \{\succ_{\mathsf{V}}\langle\mathcal{I}d\rangle \ , \ (>\langle\varphi\rangle, \succ, \succ^{\tau}_{\mathsf{IACK}}\langle \mathcal{A}rgs \circ \widetilde{\ }\rangle)\} \\ \sim_{\mathsf{IACK}} & \equiv & \sim_{\mathsf{KBOS}}\langle \widetilde{\ } \rangle \end{array}$$ ◇

Obviously, the memory-flattening process of the IACK must be performed only if the leading operators of both terms are A-operators.

$\boxed{4.12}$ **Theorem** *The* IACK *is an AC-commuting* σ*-stable simplification ordering.* ◇

4.3.3 The Knuth-Bendix Ordering with Status for C-Theories

Up to now we admitted commutative *and* associative operators. The exclusion of the latter enables us to simplify the Knuth-Bendix orderings compatible with AC-theories: The (memory-) flattening of terms to be compared is completely redundant. Obviously, we do not need to check the conditions for A-operators.

Idea of Comparison *The commutative Knuth-Bendix ordering* CK *is identical to the* KBOS *additionally requiring multiset status for each commutative operator.*

$\boxed{4.13}$ **Definition (Commutative Knuth-Bendix Ordering)** *Let* $\succsim$ *be a precedence and* φ *be an extended weight function. The status function* τ *fulfils the condition that* $\tau(f) = ⓜ$ *holds if* $f \in \mathcal{F}_C$. *The* commutative Knuth-Bendix ordering CK *on terms is defined as*

$$\begin{array}{lcl} \succ_{\mathsf{CK}} & \equiv & \succ_{\mathsf{KBOS}}\langle\mathcal{I}d\rangle \\ \sim_{\mathsf{CK}} & \equiv & \sim_{\mathsf{KBOS}}\langle\mathcal{I}d\rangle \end{array}$$ ◇

A.14 We want to use this slightly restricted version of the KBOS as an ordering to prove the termination of TRSs modulo commutative theories. Therefore, the CK must be a σ-stable simplification ordering which is C-commuting.

$\boxed{4.14}$ **Lemma** *The* CK *is a C-commuting* σ*-stable simplification ordering.* ◇

4.4 Automatic Generation of Knuth-Bendix Orderings

The problem of automatically choosing an appropriate weight function for a given TRS is not as easy as to leave it to the user. If the KBO(S) fails to orient the TRS, we can try another weight function, but without some help we may fail again and indeed it may be that no KBO(S) exists. However, the problem of choosing an appropriate KBO(S) is *decidable*. This section deals with a brief recapitulation of a decision procedure developed by Martin in [Martin, 1987] and extended in [Dick *et al.*, 1990]. This will be done in two steps: (i) The general skeleton of the method is described in subsection 4.4.1 and (ii) the main component is discussed in 4.4.2. The technique of Martin is able to automatically generate appropriate parameters (i.e. a weight function and a precedence) for the KBO. In subsection 4.4.3, we introduce a slight improvement of it that is suitable for the KBOS and the AC-commuting Knuth-Bendix orderings.

4.4.1 The Method of Martin

We give a brief description of the technique for generating a KBO for proving the termination of a TRS, contained in [Martin, 1987] and [Dick *et al.*, 1990]:

- The weight (which has to be determined) of each operator is represented by a unique variable. Each rule $l \rightarrow r$ of the given TRS $\mathcal{R}$ will be transformed into an inequality $u \geq v$ such that $u = \varphi(l)$ and $v = \varphi(r)$ are linear polynomials over the weight variables. Thus, $\mathcal{R}$ is transformed into a set $\mathcal{I}$ of linear inequalities.
- The application of a decision procedure for linear programming problems to $\mathcal{I}$ provides the set S of all solutions (obviously, S could be empty). If each solution of S leads to the equality of an inequality $u \geq v$, the leading operators of the corresponding rule $l \rightarrow r$ must be checked. If $\mathcal{H}ead(l) \neq \mathcal{H}ead(r)$ then $\mathcal{H}ead(l) \succ \mathcal{H}ead(r)$ has to be merged with the current precedence $\succ$. Otherwise, the inequality $\varphi(s_i) \geq \varphi(t_i)$ must be added to $\mathcal{I}$ where $l = f(s_1,\ldots,s_m)$, $r = f(t_1,\ldots,t_n)$ and $s_1 \Upsilon_{\mathsf{KBO}} t_1 ,\ldots, s_{i-1} \Upsilon_{\mathsf{KBO}} t_{i-1}, s_i \not\Upsilon_{\mathsf{KBO}} t_i$ hold.

$\boxed{4.15}$ **Example ([Dick** *et al.*, **1990])** *We prove the termination of the TRS*

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

by automatically generating an appropriate KBO. *According to Martin's technique, the set of inequalities*

$$\mathcal{I} = \left\{ \begin{array}{lcl} 0 & \geq & 0 \\ 0 & \geq & \varphi(i) \end{array} \right.$$

must be constructed. Since the first inequality is always an equality as well as the leading operators of the corresponding rule are identical, the set $\mathcal{I}$ has to be enriched with the inequality $\varphi(\star) + \varphi_0 \geq 0$ (due to the pair $(x \star y, x)$):

$$\mathcal{I}' = \left\{ \begin{array}{rcl} \varphi(\star) + \varphi_0 & \geq & 0 \\ 0 & \geq & \varphi(i) \end{array} \right.$$

which is true for any values of $\varphi(\star)$ *and* φ_0 *(because* φ_0 *has to be positive). The second inequality has only one solution –* $\varphi(i) = 0$ *– implying both sides to be equal. Thus,* $i \succ \star$ *must be added to the (empty) precedence since* $\mathcal{H}ead(i(x \star y)) = i$ *and* $\star = \mathcal{H}ead(i(y) \star i(x))$*.* ◇

For simplicity, we did not mention that constraints on the correct values of the weights of operators (for example, $\varphi(\star) \geq 0$, required by the definition of the KBO) must be included to the set of inequalities.

The possibility of using such a decision procedure was suggested by Lankford in the appendix of [Lankford, 1979], by Manna & Ness in [Manna and Ness, 1970] as well as by Dershowitz in [Dershowitz, 1982], although no explicit algorithm is given there.
[Martin, 1987] applies the well-known Simplex method whereas [Dick et al.*, 1990] takes the so-called 'method of complete description' for deducing the solution of linear inequalities.*

4.4.2 The Method of Complete Description

The technique of [Dick *et al.*, 1990] transforms a TRS into a set of *homogeneous linear inequalities.* The *method of complete description* (see, for example, [Dick *et al.*, 1990]) is able to solve all linear programming problems which have a solution. Unlike the *Simplex method*, it calculates *all* solutions instead of just the optimal solution. The substantial ideas of the method of complete description are the following ones:

- The method is intrinsically incremental, i.e. adding a new inequality to the existing system does not necessitate a complete recalculation.
- It computes all solutions (if the system of inequalities is solvable) in form of a matrix. The column vectors of this matrix are the extreme vectors of the finite cone of solutions (i.e. each non-negative linear combination of these vectors provides a solution).
- The regulation for constructing the above matrix S_m is $S_m = S_{m-1} \cdot \kappa(a_m^T \cdot S_{m-1})$ where m is the number of initial inequalities, a_m^T represents the vector of coefficients of the mth inequality and S_0 is initialized as the $n \times n$ identity matrix (where the inequalities contain n distinct variables). The function κ provides a simple matrix by combining sparse column vectors according to its input.

Another advantage of this technique (beside its incrementality) is the easy decision whether both sides of a specific inequality $u \geq v$ are equal ($u = v$) under *all* solutions. This can be detected by the following procedure:

> The ith inequality is always an equality if the ith row of the matrix $A \cdot S_m$ (where A is the initial system of inequalities and S_m is the matrix of solutions of A) is composed of zeros, only.

If an inequality is just an equality, then one has either to extend the precedence of the KBO or to apply the KBO recursively to the arguments. In [Dick *et al.*, 1990], an *explicit* appropriate weight function for a given TRS is *not* determined (it is sufficient to know that there exists at least one such a weight function). However, a concrete KBO is (sometimes) desirable from a practical point of view.

There is a problem concerning the explicit construction of an appropriate weight function: The set of solutions possibly *includes* solutions for which an inequality becomes an equality. Thus, these solutions must be excluded. An easy way to select a solution inducing strict inequalities is as follows[5]:

> Let
>
> $$S = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mn} \end{pmatrix}$$
>
> be a solution matrix of a set A of linear inequalities with n distinct variables[6]:
>
> 1. Choose a column of $A \cdot S$ where the number of zeros is minimal. If two such columns exist, take the one where the sum of all elements is minimal.
> 2. Remove all rows in $A \cdot S$ where the corresponding element in the chosen column is not zero.
> 3. If $A \cdot S$ is empty, then return the sum of all chosen columns of S[7]. Otherwise, continue with step 1.
>
> Figure 4.1 contains a procedural description of this heuristic. Note that, for the worst case, this approach computes the sum of all column vectors of the matrix S.

Another problem concerns the generation of *solutions for inhomogeneous inequalities*[8]. A simple trick helps:

> Let $\mathcal{I} = \bigcup_j \{\sum_{i=1}^{n} \alpha_{ij} x_i + \beta_j \geq 0\}$ be a set of inhomogeneous inequalities. We introduce *one new* variable x and apply the method of complete description to $\mathcal{I}_x = \bigcup_j \{\sum_{i=1}^{n} \alpha_{ij} x_i + \beta_j x \geq 0\}$. The set of all solutions for $\mathcal{I}$ is the subset of the solutions for $\mathcal{I}_x$ where $x = 1$ holds.

For the remaining part of this thesis, when referring to the method of complete description, also any other adequate technique for solving linear programming problems can be used.

---

[5] Alternatively, the computation of a basis for the solution matrix S_m (generated by the method of complete description) would help. However, an algorithm is needed which does not produce negative numbers. For example, the well-known Gaussian algorithm computes a basis by *subtracting* columns.

[6] Thus, each column vector of S provides a solution of A.

[7] The ith column of S corresponds to the ith column of $A \cdot S$.

[8] Regarding the KBO, homogeneous inequalities are produced, only. However, in subsection 5.5.2, we are dealing with a generalization of the technique developed by Martin for polynomial orderings where inhomogeneous inequalities must be treated.

The following procedure $\mathcal{SOLUTION}$ computes a solution vector w.r.t. an $m \times n$ solution matrix S. Assume that $A \cdot S$ is composed of n column vectors $(\alpha_{1i},\ldots,\alpha_{mi})$ where $i \in \{1,\ldots,n\}$ (m is the number of different variables and k is the number of inequalities). Procedure $\mathcal{COL}(S,i)$ returns the ith column vector of the matrix S. Procedure $\mathcal{ZER}(V,N)$ returns the number of zeros occurring in the vector V' and procedure $\mathcal{COUNT}(V,N)$ returns the sum of the components of the vector V' with V' being the vector derived from V by removing the components *not* specified in the set N of natural numbers. Furthermore, $\oplus$ describes the usual addition of vectors.

```
$\mathcal{SOLUTION}$ = proc($A : k \times m$ matrix, $S : m \times n$ matrix) returns (vector)
    $SV$: Solution-Vector[9]:= $\mathcal{SELECT}(A, S, \{1,\ldots,m\}, SV)$
    return ($SV$)
end

$\mathcal{SELECT}$ = proc($A : k \times m$ matrix, $S : m \times n$ matrix,
                $N$ : set of natural numbers, $V$ : vector) returns (vector)
if    $N = \emptyset$
then return ($V$)
else  begin
          $MC$: Minimum-Column := $\mathcal{COL}(S, i)$
              such that
              $\mathcal{COUNT}(\mathcal{COL}(A \cdot S, i), N) =$
              $min\{\mathcal{COUNT}(\mathcal{COL}(A \cdot S, k), N) \mid \mathcal{ZER}(\mathcal{COL}(A \cdot S, k), N) =$
                  $min\{\mathcal{ZER}(\mathcal{COL}(A \cdot S, j), N) \mid 1 \le j \le n\}\}$
          $V := V \oplus MC$
          $\mathcal{SELECT}(A, S, N \setminus \{j \mid \alpha_{jl} \neq 0, \alpha_{jl} \in \mathcal{COL}(A \cdot S, i)\}, V)$
      end
end
```

Figure 4.1: Procedure for generating a solution of linear inequalities

4.4.3 Automatic Generation of the Knuth-Bendix Ordering with Status

In order to apply the technique of [Dick *et al.*, 1990] to the KBOS, we have to slightly improve it according to the differences between the KBO and the KBOS.

	KBO	KBOS
variable condition	$\forall x \in \mathcal{X}$: $=\langle \lVert . \rVert_x \rangle$ & $=\langle \varphi \rangle$	$\forall x \in \mathcal{X}$: $\geq\langle \lVert . \rVert_x \rangle$ & $=\langle \varphi \rangle$
precedence	total ordering	quasi-ordering
recursive application	status ① for all operators	arbitrary status

---

[9] The initial value of SV is $(0,\ldots,0)$.

The variable condition is investigated before transforming a TRS into inequalities (as done for the KBO). The test for unary operators is included whenever both sides of an inequality are identical (see the definition of the KBO). The determination of an appropriate status function is executed similarly to the method of [Forgaard, 1984] (see section 3.4):

> We assign undefined status $\textcircled{u} = \{\textcircled{l}, \textcircled{r}, \textcircled{m}\}$ to each operator and reduce it to a maximal subset when necessary. Note that the variable condition (which is checked first) is a good indicator for excluding a specific type of status.

Obviously, this method for generating an appropriate KBOS can also be used for the ACK, the ACKM and the IACK by (i) applying the (memory-) flattening operation before creating the inequalities as well as (ii) adding the further requirements (such as '$\forall f \in \mathcal{F}_A$: $\varphi(f) = 0$' and '$\forall f \in \mathcal{F}_C$: $\tau(f) = \textcircled{m}$' for the ACK and '$\forall i \in \mathbb{N}$: $f^i \sim f$', '$\forall f^i \in \mathcal{F}^E$: $\varphi(f^i) = i \cdot \varphi(f)$' and '$\forall f \in \mathcal{F}_C$: $\tau(f) = \textcircled{m}$' for the ACKM and the IACK) as constraints to the inequalities.

Chapter 5

Polynomial Orderings

Orderings based on so-called *polynomial interpretations* of operators represent a powerful technique of proving the termination of TRSs. The most important potential of polynomial orderings lies in the opportunity to *model the semantics* of the operators as interpretations, and thus make use of the semantics of the given TRS in a termination proof of the TRS. In this chapter, we introduce some features for a suitable adaptation of polynomial orderings to practical requirements.

- **Section 5.1:** We start with two well-known definitions of polynomial orderings (the original one and an extension of it by integrating a precedence and an application to the arguments in lexicographic order).
- **Section 5.2:** This section consists of two extensions of the orderings described in the preceding section (one with a special treatment of, for example, unary operators interpreted as the identity and another one introducing a status function).
- **Section 5.3:** This section deals with techniques of proving the positiveness of a polynomial, one of the main problems caused by polynomial orderings. We investigate and extend a method due to BenCherifa and Lescanne which can partially solve this problem. We describe some observations on the time complexity of this approach. In addition, an extension of this method is provided which is based on a set of constraints. We also discuss another improvement utilizing the arithmetic-mean-geometric-mean inequality. Finally, we deal with a modification for a restricted class of TRSs, namely those involving constants in specific positions, only.
- **Section 5.4:** We introduce polynomial orderings that are suitable to prove the termination of TRSs based on different underlying equational theories. There already exists a restricted class of interpretations for AC-theories. Here, we develop restrictions for polynomial orderings which are compatible with some other interesting theories including distributivity, endomorphism, if-then-else and Boolean rings.
- **Section 5.5:** The last section of this chapter is devoted to another problem with polynomial orderings, namely the choice of the right interpretations for a given TRS. We will be dealing with two new heuristic approaches: (i) guidelines for automatically orienting certain classes of TRSs as well as (ii) an algorithm for choosing appropriate interpretations from restricted classes of polynomials.

5.1 Definitions of Well-Known Polynomial Orderings

First of all, we describe and discuss the definitions of polynomial orderings occurring in the literature. To understand these definitions, some motivating remarks are given.

Polynomial orderings belong to the class of semantic orderings. As mentioned in the introduction, such orderings use a termination function which maps the terms into a well-founded set. Thus, polynomial orderings map the two terms s, t to be compared into polynomials (by applying an interpretation $[\![.]\!]$) and use a well-founded ordering $\succ_{\mathsf{P}}$ on polynomials:

$$\begin{tikzcd} s \arrow[d, "{[\![\cdot]\!]}"'] & & t \arrow[d, "{[\![\cdot]\!]}"] \\ p & \succ_{\mathsf{P}} & q \end{tikzcd}$$

where p and q
are polynomials

Assume that s and t are ground terms. Then p and q are *ground polynomials*, i.e. they are integers or real numbers (depending on the domain of the polynomials). Thus, $\succ_{\mathsf{P}}$ is realized as $>$, the natural ordering on integers (on real numbers, respectively). However, in contrast to the fact that $>$ is total on numbers, the set of ground polynomials cannot be ordered in this way by $>$. For example, $p(2) = 4 = q(1)$ if $p(x) = x^2$ and $q(x) = x+3$. Thus, the relation on ground terms induced by $>$ is a partial ordering. Nevertheless, it is not $\mathcal{T}(\mathcal{F},\emptyset)$-compatible since $[\![1]\!]() > [\![0]\!]()$ but not $[\![0 \star 1]\!]() > [\![0 \star 0]\!]()$ if $[\![0]\!]() = -1, [\![1]\!]() = 2$ and $[\![\star]\!](x, y) = xy$. The reason for this phenomenon is that xy is monotonic in each variable only on $\mathbb{R}_+$ ($\mathbb{N}$) but not on $\mathbb{R}$ ($\mathbb{Z}$). One can easily prove that the monotonicity of the polynomials implies the $\mathcal{T}(\mathcal{F},\emptyset)$-compatibility of the induced polynomial ordering if

(i) the domain of the interpretations is restricted to $\mathbb{R}_+$ or $\mathbb{N}$ and
(ii) the arity of the interpretations is identical to the arity of the corresponding operators.

Note that property (i) can be weakened (see also [Zantema, 1992a]). For instance, let $\mathcal{F} = \{s, 0\}$ and $[\![0]\!]() = 2, [\![s]\!](x) = x^2 - x$, thus, one may use negative coefficients. The induced ordering is also $\mathcal{T}(\mathcal{F},\emptyset)$-compatible because the smallest value of an interpreted term is 2 ($= [\![0]\!]()$) and $\forall a, b \geq 2$: $a > b \hookrightarrow a^2 - a > b^2 - b$. However, the disadvantage of a weakened variant of (i) is the need of an *explicit proof of the monotonicity* of the used interpretations.

Let us now consider the problem of well-foundedness. An ordering possessing the above properties (i) and (ii) is well-founded if it is based on $\mathbb{N}$ since there exist no infinite descending chains in $\mathbb{N}$. On the other hand, polynomial orderings on $\mathbb{R}_+$ need not be well-founded. For example, the infinite sequence $[\![0]\!]() > [\![s(0)]\!]() > [\![s(s(0))]\!]() > \cdots$ exists if $[\![0]\!]() = 1$ and $[\![s]\!](x) = \frac{1}{2}x$. *The use of simplification orderings allows orderings to be extended to domains that are not themselves well-founded.* Therefore, one can associate a monotonic polynomial over the *reals* to each operator, provided the interpretations satisfy a kind of subterm property.

In order to lift polynomial orderings to terms containing variables, σ-stability must be satisfied. A straightforward way for achieving this property is to explicitly require it, as done in the following two definitions.

5.1 Definition ([Lankford, 1975]) *Let $\mathcal{F}$ be a set of operators and $[\![.]\!]$ an associated set of interpretations over $\mathbb{N}$. Then the* polynomial ordering NPOL *on terms is defined as*

$$\succ_{\mathsf{NPOL}} \;\equiv\; \forall \text{ ground } \sigma\text{: } >\langle [\![.]\!] \circ \sigma \rangle \qquad \diamond$$

$\boxed{5.2}$ **Definition ([Dershowitz, 1979a])** *Let $\mathcal{F}$ be a set of operators and $\llbracket . \rrbracket$ an associated set of interpretations over $\mathbb{R}_+$ such that $\forall f \in \mathcal{F}$: $\forall i \in [1, \mathcal{A}r(f)]$: $\forall a_i \in \mathbb{R}_+$: $\llbracket f \rrbracket(x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n) > a_i$. Then the* polynomial ordering RPOL *on terms is defined as*

$$\succ_{\mathsf{RPOL}} \;\equiv\; \forall \textit{ ground } \sigma\colon >\langle \llbracket . \rrbracket \circ \sigma \rangle \qquad \diamond$$

A main disadvantage of these definitions is the composition of ground term substitutions with polynomial interpretations ($\llbracket . \rrbracket \circ \sigma$). There is a close relationship between a ground term substitution, say σ_T, and a ground polynomial substitution (which is a substitution that replaces each polynomial variable with a number of $\mathbb{N}$ and $\mathbb{R}_+$, respectively), say σ_P: $\forall t \in \mathcal{T}(\mathcal{F},\mathcal{X})$: $\forall \sigma_T$: $\exists \sigma_P$: $\llbracket t\sigma_T \rrbracket = \llbracket t \rrbracket \sigma_P$. The proof of this fact is constructive with $\sigma_P = \llbracket \sigma_T \rrbracket$. With this result, it is possible to replace the orderings of definitions 5.1 and 5.2 with $\forall$ ground σ: $>\langle \sigma \circ \llbracket . \rrbracket \rangle$. This ordering is σ-stable by definition. However, it is weaker than its counterparts defined in 5.1 and 5.2. For example, let $\mathcal{F} = \{0, s\}$ and $\llbracket 0 \rrbracket() = 2, \llbracket s \rrbracket(x) = x^2$. Then $s(x)$ would not be greater than x since $1 \not> 1$ for $\sigma_P = \{x \leftarrow 1\}$. This is due to the non-existence of ground terms with an interpretation of 1 (the smallest term is the constant 0)[1]. Therefore, one has to substitute numbers which are the interpretations of ground terms, only, for the polynomial variables. It is quite evident that the set of these numbers strongly depends on the minimal interpretation of the constants. From a practical point of view it is easier to use the set $\{n \mid n \geq \mathit{min}\{\llbracket c \rrbracket() \mid c \text{ is a constant}\}\}$ instead of the set $\{\llbracket t \rrbracket \mid t \in \mathcal{G}(\mathcal{F})\}$. Thus, the following definition of an ordering on polynomials is based on the former set.

Idea of Comparison *A polynomial p is greater than a polynomial q if, and only if, all the corresponding substitutions of numbers greater than or equal to μ (i.e. the minimum of the interpretations of all constants) for the variables can be oriented in the same way.*

$\boxed{5.3}$ **Definition (Ordering on Polynomials)** *Let $\mathcal{F}$ be a set of operators, $\llbracket . \rrbracket$ an associated set of interpretations over $\mathbb{N}$ or over $\mathbb{R}_+$ and $\mu = \mathit{min}\{\llbracket c \rrbracket() \mid c \in \mathcal{F}, \mathcal{A}r(c) = 0\}$. The* ordering $\sqsupset$ on polynomials *p, q over variables $x_1, \ldots, x_n$ is defined as*

$$p \sqsupset q \iff \forall a_i \geq \mu\colon\; p(x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n) > q(x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n) \qquad \diamond$$

It is not necessary for the polynomials p and q to be based on the same set of variables. The set $\{x_1, \ldots, x_n\}$ represents the union of the sets of variables of p and q. Obviously, $p \not\sqsupset q$ if there occurs a variable in q which is not contained in p.

$\boxed{5.4}$ **Remark**

Note that μ has to be (less than or) equal to $\min\{\llbracket c \rrbracket() \mid c \in \mathcal{F}, \mathcal{A}r(c) = 0\}$ if there exists at least one constant in $\mathcal{F}$. If μ is strictly greater than the minimal interpretation of all constants, the induced polynomial ordering might no longer be σ-stable and, as a consequence, does not need to be well-founded. This is illustrated by a simple example: Let $\llbracket g \rrbracket(x) = x^2$, $\llbracket h \rrbracket(x) = x + 1$ and $\llbracket a \rrbracket() = 1$. Then

$$\begin{array}{rcll} \llbracket g(x) \rrbracket(x) & \sqsupset & \llbracket h(x) \rrbracket(x) & \text{if } \mu = 2 \\ \llbracket h(a) \rrbracket() & \sqsupset & \llbracket g(a) \rrbracket() & \end{array}$$

---

[1] From a formal proof-theoretic point of view the reason for this ordering being weaker than the ones in the definitions is that here $\forall t \in \mathcal{T}(\mathcal{F},\mathcal{X})$: $\forall \sigma_P$: $\exists \sigma_T$: $\llbracket t\sigma_T \rrbracket = \llbracket t \rrbracket \sigma_P$ does not hold.

The σ-stability of the induced polynomial ordering must guarantee that $\llbracket g(a)\rrbracket() \sqsupset \llbracket h(a)\rrbracket()$ holds if $\llbracket g(x)\rrbracket(x) \sqsupset \llbracket h(x)\rrbracket(x)$. It is obvious that the system $\{g(x) \rightarrow h(x),\, h(a) \rightarrow g(a)\}$ does not terminate. However, the above condition concerning μ must be guaranteed (which implies that $g(x)$ and $h(x)$ are incomparable w.r.t. the induced polynomial ordering).

The ordering $\sqsupset$ on polynomials forms the basis for the following two definitions of polynomial orderings which are slightly restricted versions of the NPOL and the RPOL, but far *more convenient for practical applications* (as motivated previously).

For the remaining part of this chapter, $\mathbb{R}_{\geq 1}$ and $\mathbb{R}_{>1}$ stand for $\mathbb{R}_{\geq 1} = \{n \mid n \in \mathbb{R}, n \geq 1\}$ and $\mathbb{R}_{>1} = \{n \mid n \in \mathbb{R}, n > 1\}$.

5.5 **Definition (Polynomial Ordering on $\mathbb{N}$, Based on [Lankford, 1975])** *Let $\mathcal{F}$ be a set of operators and $\llbracket . \rrbracket$ an associated set of interpretations over $\mathbb{N}$. Then the* polynomial ordering POLN *on terms is defined as*

$$\succ_{\mathsf{POLN}} \;\equiv\; \sqsupset\langle\llbracket \cdot \rrbracket\rangle \qquad \diamond$$

Idea of Comparison *The polynomial ordering* POLN *transforms the terms to be compared into polynomials over $\mathbb{N}$ by applying $\llbracket . \rrbracket$ and compares the polynomials w.r.t. $\sqsupset$. The following ordering* POLR *is identical to* POLN *except that the domain of the polynomials is $\mathbb{R}_{>1}$ which guarantees the 'subterm' property of each interpretation.*

5.6 **Definition (Polynomial Ordering on $\mathbb{R}_{>1}$, Based on [Dershowitz, 1979a])** *Let $\mathcal{F}$ be a set of operators and $\llbracket . \rrbracket$ an associated set of interpretations over $\mathbb{R}_{>1}$. Then the* polynomial ordering POLR *on terms is defined as*

$$\succ_{\mathsf{POLR}} \;\equiv\; \sqsupset\langle\llbracket \cdot \rrbracket\rangle \qquad \diamond$$

By restricting the domain of the polynomials from $\mathbb{R}_{+}$ to $\mathbb{R}_{>1}$, the 'subterm' property of the interpretations, i.e. $\forall f \in \mathcal{F}$: $\forall i \in [1, \mathcal{A}r(f)]$: $\forall a_i \in \mathbb{R}_{>1}$: $\llbracket f \rrbracket(x_1 \leftarrow a_1, \ldots, x_{\mathcal{A}r(f)} \leftarrow a_{\mathcal{A}r(f)}) > a_i$, is always guaranteed.

5.7 **Remark**

Considering the preceding comments, it is obvious that the coefficient α in the interpretation $\llbracket f \rrbracket(x_1, \ldots, x_n) = \alpha x_1^{e_1} \cdot \ldots \cdot x_n^{e_n}$ for an operator f must be greater than 1 whenever either $(n = 1 \wedge e_1 = 1)$ or $\mu = 1$ holds. The coefficients of all other interpretations (for unary and other types of operators) can be equal to 1.

5.8 **Notation ((General) Polynomial Ordering)** *For the remaining part of this thesis,*
A.29 POL *stands for one of the orderings* NPOL, RPOL, POLN *or* POLR. $\diamond$

We are now going to point out the two main problems arising when polynomial orderings are used in practice. Consider the following rule system which describes two properties of the division on natural or real numbers:

$$\left\{\begin{array}{lrcl} 1: & i(x/y) & \rightarrow & y/x \\ 2: & (x/y)/z & \rightarrow & x/(z/i(y)) \end{array}\right. \tag{5.1}$$

In order to prove the termination of the second rule, the first argument of $[\![/]\!](x, y)$ has to be stronger than the second one and $[\![i]\!](x) = x$ must hold[2]. Therefore, the first rule is not ordered w.r.t. these interpretations and the termination of the system given above cannot be ensured with the POL. Thus, the first problem concerning polynomial orderings is, obviously, their incompleteness. A.35

Another problem with polynomial orderings is related to the generation of appropriate interpretations to prove the termination of a given TRS. Both difficulties can be diminished by making use of the following ordering of Lankford.

A.9
A.10
A.11
A.12
A.15
A.18
A.24
A.26
A.28
A.29
A.30
A.31
A.34
A.36

> **Idea of Comparison** *The improved polynomial ordering* IPOL *is a generalization of the* KBO*, i.e. it works like the* KBO *with the exception that general polynomial interpretations are used instead of weights (which are special linear polynomials, see subsection 1.5.4).*

$\boxed{5.9}$ **Definition (Improved Polynomial Ordering, [Lankford, 1979])** *Let $\mathcal{F}$ be a set of operators, $[\![.]\!]$ be an associated set of interpretations over $\mathbb{N}$ such that $\forall c \in \mathcal{F}, \mathcal{A}r(c) = 0$: $[\![c]\!]() \geq 2$ and $\succ$ be a total precedence satisfying $\forall f \in \mathcal{F}\backslash\{g\}$: $g \succ f$ if $\mathcal{A}r(g) = 1 \wedge [\![g]\!](x) = x$. Then the* improved polynomial ordering IPOL *on terms is defined as* A.17

$$\begin{array}{lcl} \succ_{\mathsf{IPOL}} & \equiv & (\sqsupset\langle[\![.]\!]\rangle\,, \succ\,, \succ^{\textcircled{1}}_{\mathsf{IPOL}}) \\ \succsim_{\mathsf{IPOL}} & \equiv & =\langle[\![.]\!]\rangle \;\; \& \;\; \sim\langle\mathcal{H}ead\rangle \;\; \& \;\; \succsim^{\textcircled{1}}_{\mathsf{IPOL}}\langle\mathcal{A}rgs\rangle \end{array}$$ ◇

Note that the conditions

(i) g is maximal w.r.t. $\succ$ if $[\![g]\!](x) = x$ and

(ii) for each constant c, $[\![c]\!]() \geq 2$

guarantee that the IPOL is well-founded. Consider the following examples:

- The rule $h(x) \rightarrow g(h(x))$ is not terminating but it would be ordered w.r.t. the IPOL (based on $[\![g]\!](x) = x$ and $h \succ g$) if (i) was not required.
- A termination proof for the non-terminating rule $a \rightarrow f(a, a)$ would be possible with the help of the IPOL (based on $[\![f]\!](x, y) = xy$, $[\![a]\!]() = 1$ and $a \succ f$) if (ii) was not guaranteed.

---

[2] Let $l = (x/y)/z$ and $r = x/(z/i(y))$. Then $l|_{12} = y$ and $r|_{22} = i(y)$. Thus, the first argument of $[\![/]\!](x, y)$ must have more 'weight' than the second one. However, this violates the requirements w.r.t. the variable z ($l|_2 = z$ and $r|_{21} = z$). These observations (more or less) imply the interpretation of $[\![/]\!](x, y)$ to be $x + y$. Therefore, $[\![i]\!](x)$ must be x.

$\boxed{5.10}$ Lemma ([Lankford, 1975], [Lankford, 1979], [Dershowitz, 1979a], [Steinbach and Zehnter, 1990])

- *The* NPOL, *the* POLN *and the* IPOL *are reduction orderings.*
- *The* RPOL *and the* POLR *are σ-stable simplification orderings.* ◇

In contrast to the original polynomial orderings, using the IPOL, we are able to prove the
A.13 termination of (5.1). Appropriate interpretations are also easier to generate for the IPOL than
A.17 for the POL. We use the semantics (if known) of the operators as their interpretations. This will equalize both sides of an equation w.r.t. the POL. An additional consideration often provides an appropriate precedence which causes the orientation of the equation. Thus, for orienting the specification of (5.1) we use the interpretations $[\![/]\!](x, y) = x + y$, $[\![i]\!](x) = x$ and $i \succ /$.

The theory of polynomial orderings has been studied in innumerable scientific publications and extended in various ways. Special polynomial orderings have been developed by Manna & Ness in 1970 ([Manna and Ness, 1970]); they use linear and quadratic interpretations and also illustrate how the coefficients of linear interpretations can be chosen by solving inequalities. Another still earlier work has been done by Iturriaga in 1967 where exponential interpretations are applied to left-linear TRSs ([Iturriaga, 1967]). Linear interpretations were used by Knuth & Bendix in 1967 ([Knuth and Bendix, 1967]). However, the best known work is that of Lankford in 1975 ([Lankford, 1975]) and in 1979 (for the improved polynomial ordering, [Lankford, 1979]) who introduced the usage of monotonic polynomial *interpretations over the non-negative integers. Brief sketches and some examples on the original polynomial ordering can be found in [Lankford, 1976], [Dershowitz and Manna, 1979], [Huet and Oppen, 1979] and [Huet, 1980]. The seminal work on simplification orderings of Dershowitz has also influenced the theory of polynomial orderings. In [Dershowitz, 1979a], the domain of interpretations was extended to non-well-founded ones (i.e. from $\mathbb{N}$ to $\mathbb{R}_+$) by requiring a kind of subterm property of the interpretations. As far back as in 1976, Stickel proved ([Stickel, 1976]) that there exist terminating TRSs which cannot be oriented using polynomial interpretations. Even the class of primitive recursive functions is inadequate for this task. In [Dershowitz, 1983], a TRS is given that computes disjunctive normal forms and also requires more than polynomials for its termination proof. In a recent publication ([Lescanne, 1992]) exponential functions are taken care of. In [Zantema, 1992a], a generalization of polynomial orderings has been presented by introducing the notion of* monotonic algebras[3]*. In general, however, this method is not a simplification ordering (see page 177).*

A.8
A.21 In this thesis, we prevalently consider polynomial interpretations (only in appendix A, some
A.22 *exponential functions* will be used for the examples). The reason for this restriction is the fact that

- the practical application of exponential functions is somewhat complex as well as cumbersome and
- the improved polynomial ordering IPOL provides a termination proof method which facilitates the choice of polynomial interpretations and extends the original polynomial orderings in a way such that a respectable number of TRSs can be oriented which require exponential functions instead of polynomials for the POL.

---

[3] We will use this method for proving the termination of example 7.1 on page 177.

$\boxed{5.11}$ **Remark**

In the history of polynomial orderings there have already been two extensions. The first one concerns the classes of functions used as interpretations (see above). The second improvement of the original POL is based on the concept of the lexicographic concatenation of well-founded orderings $\succ_1, \cdots, \succ_n$ (see subsection 2.2.3 on page 32). This concept has been applied to the POL in such a way that several different polynomial orderings are concatenated, i.e. $\succ_1, \cdots, \succ_n$ are polynomial orderings with distinct interpretations. This technique is called polynomial ordering with interpretations by a *Cartesian product* of polynomials (see, for example, [BenCherifa and Lescanne, 1987]). |A.35

5.2 Extensions of Polynomial Orderings

In this section, we develop two extensions of the polynomial orderings discussed in the previous section. The one improves POLN and POLR such that POLN will become a simplification ordering and POLR can be based on interpretations with the domain $\mathbb{R}_{\geq 1}$. The other extension integrates a status function into the IPOL, i.e. the status of the operators will not be restricted any more to status ①.

As stated in section 5.1, (i) the POLN is a reduction ordering but not a simplification ordering and (ii) the POLR is based on $\mathbb{R}_{>1}$ in order to ensure that it has the subterm property. By requiring unary operators not to be interpreted as identity functions[4] and the interpretations of constants to be greater than 1[5] (see also remark 5.7), (i) the POLN would be a simplification ordering and (ii) the domain of the POLR could be $\mathbb{R}_{\geq 1}$. However, this is a restriction which is not necessary. The above properties can also be achieved by checking the homeomorphic embedding relation, first.

Idea of Comparison *Before computing and comparing the interpretations of the two terms to be compared, we test whether one of the terms is homeomorphically embedded in the other one.*

$\boxed{5.12}$ **Definition (Simplification Ordering on Polynomial Interpretations)** *Let $\mathcal{F}$ be a set of operators and $[\![.]\!]$ be an associated set of interpretations over $\mathbb{N}$ or $\mathbb{R}_{\geq 1}$. Then the* simplification ordering SPOL on polynomial interpretations *over terms is defined as*

$$\succ_{\mathsf{SPOL}} \quad \equiv \quad \{\succ_{\mathsf{EMB}}\langle\mathcal{I}d\rangle \ , \ \sqsupset\langle[\![.]\!]\rangle\} \qquad \diamond$$

The integration of the homeomorphic embedding relation into the polynomial orderings is a similar way to ensure the subterm property as done in the definition of the RPOS (by postulating that an argument of the presumably greater term is equal to the presumably

---

[4] Note that identity functions of unary operators can lead to the incomparability of two terms where one of them is a subterm of the other one. For instance, the interpretation $[\![h]\!](x) = x$ implies $h(x) \not\succ_{\mathsf{POLN}} x$.

[5] For example, the terms of the rule $f(a, x) \rightarrow x$ cannot be compared with the POL specified by the interpretations $[\![f]\!](x, y) = xy$ and $[\![a]\!]() = 1$.

smaller one). Note that the SPOL over $\mathbb{R}_{\geq 1}$ is more powerful than the SPOL over $\mathbb{N}$. The difficulty of proving the SPOL to be a σ-stable simplification ordering is closely connected to show a kind of compatibility of the homeomorphic embedding relation with the original polynomial orderings. This compatibility is a direct consequence of the fact that $s \succ_{\mathsf{EMB}} t$ implies $[\![s]\!] \sqsupseteq [\![t]\!]$.

$\boxed{5.13}$ **Lemma** *The* SPOL *is a σ-stable simplification ordering.* $\diamond$

It is also possible to improve the IPOL into two directions. This will be illustrated by the following two examples.

- We want to prove the termination of the rule
  $$(x \star y) + y \rightarrow (x+1) \star y$$
  with the help of the IPOL. Assume that the interpretations $[\![+]\!](x,y) = [\![\star]\!](x,y) = x+y$ and $[\![1]\!]() = 2$ are given. Furthermore, let $\mu = 2$. Thus, $[\![(x \star y) + y]\!] = x + 2y$, $[\![(x+1) \star y]\!] = x+y+2$ and $x+2y \not\sqsupset x+y+2$, $x+2y \neq x+y+2$ hold. However, (i) $x+2y \sqsupset x+y+2$ if $x,y > 2$ and (ii) $x+2y = x+y+2$ if $x,y = 2$. Therefore, the relation $[\![(x \star y) + y]\!] \sqsupseteq [\![(x+1) \star y]\!]$ is valid[6]. Using this fact, the IPOL can be extended by replacing the lexicographic concatenation $(\sqsupset\langle[\![.]\!]\rangle\,,\succ\,,\succ^{①}_{\mathsf{IPOL}})$ with the generalized form $\{\sqsupset\langle[\![.]\!]\rangle\,,\sqsupseteq\langle[\![.]\!]\rangle\,\&\succ\,,\sqsupseteq\langle[\![.]\!]\rangle\,\&\sim\&\succ^{①}_{\mathsf{IPOL}}\}$. The generalized variable condition of the KBOS inspired us to develop this improvement of the IPOL (see also (4.1) on page 81).

- We would like to show the termination of the rule
  $$f(h(s(x)),y) \rightarrow f(y,s(x))$$
  by using the IPOL. Assume that the interpretation of h is the identity function. Therefore, the interpretations of both sides of the rule will be identical, if $[\![f]\!](x,y)$ is commutative[7], and we have to apply the IPOL recursively to the arguments. This employment fails because two different variables cannot be compared. If the IPOL would admit to compare the arguments as multisets, the rule is ordered since $h(s(x)) \succ_{\mathsf{IPOL}} s(x)$ will hold if $h \succ s$ is part of the precedence.

These considerations lead to the following generalization of the IPOL.

> **Idea of Comparison** *The technique of comparing two terms w.r.t. the polynomial ordering* POLS *with status is practically identical to that of the* IPOL *with the following three exceptions: (i) It uses an additional status function which establishes the way of comparing the arguments, (ii) it is based on $\sqsupseteq\langle[\![.]\!]\rangle$ instead of $=\langle[\![.]\!]\rangle$ when $\not\sqsupset\langle[\![.]\!]\rangle$ holds and (iii) the total precedence is replaced by a quasi-precedence.*

$\boxed{5.14}$ **Definition (Polynomial Ordering with Status)** *Let $\mathcal{F}$ be a set of operators, $[\![.]\!]$*
A.25 *be an associated set of interpretations over $\mathbb{N}$ or $\mathbb{R}_{>1}$ such that $\forall c \in \mathcal{F}, \mathcal{A}r(c) = 0$: $[\![c]\!]() \geq 2$, $\succsim$ be a quasi-precedence satisfying $\forall f \in \mathcal{F} \setminus \{g\}$: $g \succ f$ if $\mathcal{A}r(g) = 1 \wedge [\![g]\!](x) = x$ and τ be a status function. Then the* polynomial ordering POLS with status *is defined as*

$$\begin{array}{lcl} \succ_{\mathsf{POLS}} & \equiv & \{\sqsupset\langle[\![.]\!]\rangle\,,\sqsupseteq\langle[\![.]\!]\rangle\,\&\succ\,,\sqsupseteq\langle[\![.]\!]\rangle\,\&\sim\&\succ^{\tau}_{\mathsf{POLS}}\} \\ \succsim_{\mathsf{POLS}} & \equiv & =\langle[\![.]\!]\rangle \;\;\&\;\; \sim\langle\mathcal{H}ead\rangle \;\;\&\;\; \succsim^{\tau}_{\mathsf{POLS}}\langle\mathcal{A}rgs\rangle \end{array}$$ $\diamond$

---

[6] Note that $\sqsupseteq \not\equiv \sqsupset \cup =$.

[7] Otherwise, both sides of the rule are incomparable w.r.t. the IPOL .

5.15 **Lemma** *The* POLS *is a reduction ordering.* ◇

5.16 **Remark**

It is possible to use $\succ_{\mathsf{SPOL}}\langle\mathcal{I}d\rangle$ instead of $\sqsupset\langle[\![.]\!]\rangle$ in the definition of the POLS. Then the conditions '$\forall c \in \mathcal{F}, \mathcal{A}r(c) = 0$: $[\![c]\!]() \geq 2$' and '$\forall f \in \mathcal{F} \setminus \{g\}$: $g \succ f$ if $\mathcal{A}r(g) = 1 \wedge [\![g]\!](x) = x$' are superfluous. However, in order to ensure that $\succ_{\mathsf{POLS}}$ is $\sim_{\mathsf{POLS}}$-compatible (i.e. $\succ_{\mathsf{POLS}} \cdot \sim_{\mathsf{POLS}} \subseteq \succ_{\mathsf{POLS}}$ and $\sim_{\mathsf{POLS}} \cdot \succ_{\mathsf{POLS}} \subseteq \succ_{\mathsf{POLS}}$), $\sim_{\mathsf{POLS}}$ must be the identity on terms ($=\langle\mathcal{I}d\rangle$). Thus, this ordering is not stronger than the IPOL, in general, since $\sim_{\mathsf{IPOL}}$ is a weaker restriction than $=\langle\mathcal{I}d\rangle$.

5.3 Proving Polynomials Positive

The use of polynomial orderings reduces the proof of termination of TRSs to finding appropriate interpretations to orient the given system, on the one hand, and to deciding whether a given polynomial is greater than another one, on the other hand. The first problem will be treated in section 5.5. The topic of this section concerns the second problem. In general, the comparison of two polynomials ($p \sqsupset q$) is reduced to the proof of the positiveness of just *one* polynomial ($p - q \sqsupset 0$). The problem whether a given polynomial in n variables is positive over real numbers is generally decidable, although in exponential time. However, if we restrict the domain of a polynomial to $\mathbb{N}$, the problem is generally undecidable ([Davis, 1973]). The undecidability is a direct consequence of the undecidability of the tenth problem of Hilbert: "There exists no algorithm for testing a polynomial (Diophantine) equation with integer coefficients to determine whether or not it has integer solutions"[8].

In this section, we present some modifications of the technique contained in [BenCherifa and Lescanne, 1987].

In the remaining part of this section, we call it BL-approach.

The following subsection deals with the description of the constructive factors concerning the BL-approach. An examination of this method, related to the time complexity, is given in subsection 5.3.2. An extension of the BL-approach is presented in subsection 5.3.3. It is based on the transformation of a polynomial into a set of constraints (i.e. linear inequalities) which can be solved by applying, for example, the method of complete description mentioned in subsection 4.4.2). In subsection 5.3.4, we present another improvement of the BL-approach which utilizes the arithmetic-mean-geometric-mean inequality. Finally, we provide a modification for restricted TRSs involving constants in specific positions, only.

There are a few other well-known approaches concerning the proof of the positiveness of a polynomial.
For the case of polynomials over $\mathbb{R}$*, (complex) decision procedures of elementary algebra exist – e.g., in [Tarski, 1951] and [Collins, 1975] – since the proof of the positiveness is a logical consequence of multivariate polynomial inequalities over* $\mathbb{R}$*.*
The algorithm of [Lankford, 1976] works as follows: A polynomial p *over the variables* $x_1,\ldots,x_n$ *is eventually positive provided that (i) no minus signs occur in* p *or (ii)* $n = 1$

---

[8] The complement to '$\exists x_1,\ldots,x_n \in \mathbb{N}$: $p(x_1,\ldots,x_n) = 0$' is '$\forall x_1,\ldots,x_n \in \mathbb{N}$: $p(x_1,\ldots,x_n) > 0 \vee \forall x_1,\ldots,x_n \in \mathbb{N}$: $p(x_1,\ldots,x_n) < 0$'. The second item of the disjunction will be decidable if the first one is.

and the leading coefficient is positive or (iii) all the first partial derivatives $dp/dx_1,\ldots,dp/dx_n$ *are eventually positive.*
The method of [Rouyer, 1991] is based on a theorem of Sturm. The basic concept of this method takes into account that a polynomial $p(x)$*, w.r.t. one variable* x*, has the root* x_0 *with multiplicity one if, and only if, it changes its sign at* x_0*. This fact can be used to determine whether a linear polynomial equals zero in the interval* $[a,b]$*. This is true if the sign of* $p(a)$ *is not equal to that of* $p(b)$*. This idea can be extended and applied to the case of general polynomials by using the derivative of a polynomial. Until now, we were engaged in proving the positiveness of a polynomial in one variable. Given a polynomial* $p(x_1,\ldots,x_n)$ *in* n *variables we treat* $x_2,\ldots,x_n$ *as constants and get a polynomial* $p(x_1)$ *in one variable. The positiveness of an expression over* $x_2,\ldots,x_n$ *is proved by recursively applying this strategy.*

5.3.1 The Method of BenCherifa and Lescanne

The main idea of the BL-approach is to prove $p \sqsupset 0$ by finding polynomials $p_1,\ldots,p_n$ such that

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

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

$\boxed{5.17}$ **Algorithm ($\mathcal{POSITIVE}$, [BenCherifa and Lescanne, 1987])** *Assume that a polynomial can be represented as a set of monomials each realized as a tuple* $(\alpha_{e_1\ldots e_n}, e_1\ldots e_n)$ *where* e_i *stands for the exponent of the variable* x_i *and* $\alpha_{e_1\ldots e_n}$ *for the coefficient of the monomial. Figure 5.1 represents algorithm* $\mathcal{POSITIVE}$ *of [BenCherifa and Lescanne, 1987].*

◇

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

Figure 5.1: Procedure $\mathcal{POSITIVE}$ of [BenCherifa and Lescanne, 1987]

As noted previously, the main idea of procedure $\mathcal{POSITIVE}$ is to consider a monomial N with a negative coefficient and try to find a monomial M with a positive coefficient which is an upper bound of it. When such a monomial M is found, we divide it into two parts M_1 and

M_2 with $M_1 + M_2 \sqsubseteq M$ such that M_1 forms an upper bound of N. Thus, the positiveness of the whole polynomial p can be guaranteed by proving the positiveness of a polynomial p', which is derived from p by replacing the monomials N and M by M_2. For example, we prove $2xy - xy - x$ to be positive by transforming $2xy$ to $xy + xy$, replacing $2xy$ and $-xy$ by xy, and proving $xy - x$ to be positive.

We now discuss procedure $\mathcal{CHANGE}$. The realization of $\mathcal{CHANGE}$ strongly depends on the minimum of the interpretations of the constants[9] contained in $\mathcal{F}$.

In [BenCherifa and Lescanne, 1987], the constants are interpreted as natural numbers greater than or equal to 2.

Thus, a monomial M having $\alpha_{e_1 \ldots e_n}$ as positive coefficient forms an upper bound of a monomial N consisting of the coefficient $\alpha_{f_1 \ldots f_n}$ (which is negative) if $\alpha_{e_1 \ldots e_n} \cdot 2^{\Sigma(e_i - f_i)} > |\alpha_{f_1 \ldots f_n}|$[10]. If M is not an upper bound of N, this number can be added to $\alpha_{f_1 \ldots f_n}$ to minimize the negative coefficient[11].

5.18 Algorithm ($\mathcal{CHANGE}$, [BenCherifa and Lescanne, 1987]) *In analogy with algorithm 5.17, let $\alpha_{e_1 \ldots e_n}$ and $\alpha_{f_1 \ldots f_n}$ be representations of monomials. Procedure $\mathcal{CHANGE}$ is contained in figure 5.2.* ◇

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

Figure 5.2: Procedure $\mathcal{CHANGE}$ of [BenCherifa and Lescanne, 1987]

It is obvious that the BL-approach can be adapted to a general minimum μ. Then all the integers '2' of procedure $\mathcal{CHANGE}$ will have to be replaced by μ. By increasing μ, this method becomes more powerful. However, remember that μ is bounded by the minimum of the interpretations of all constants (see definition 5.3).

5.19 Remark

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

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

After merging two monomials, an increase of μ would lead to a greater positive coefficient (a

---

[9] see definition 5.3 and remark 5.4

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

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

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

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

A.28 **5.20** **Example** *Suppose we have to prove that a term s is greater than a term t w.r.t. a polynomial ordering based on a polynomial interpretation* $[\![.]\!]$ *such that*

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

If we want to show $[\![s]\!] \sqsupset [\![t]\!]$*, we need to check the positiveness of the polynomial* $p = [\![s]\!] - [\![t]\!]$*. Consider the following sequence starting with* p[12]*:*

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

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

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

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

Procedure $\mathcal{CHOOSE}$ realizes a heuristic for finding an appropriate positive monomial for each negative monomial. Algorithm 5.21 defines the version of $\mathcal{CHOOSE}$ as it is contained in [BenCherifa, 1986] and implemented in the REVE system. In order to describe $\mathcal{CHOOSE}$, we need the following ordering $\succ_{\mathsf{MON}}$ on two monomials:

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

---

[12] We mark the monomials chosen for merging with bold typed numbers and letters.

$\mathcal{CHOOSE}$ selects the greatest (w.r.t. $\succ_{\mathsf{MON}}$) negative monomial as well as a positive monomial such that the set of variables is minimal w.r.t. the negative monomial. If there exist more than one such positive monomial, the greatest one (w.r.t. $\succ_{\mathsf{MON}}$) will be chosen. For example, consider the polynomial $p = x^3yz + x^2y^3 + x^2y^2 - x^2y - xy^2z$. First of all, we merge the monomials x^2y^3 and $-x^2y$ since x^2y^3, x^2y^2 are minimal w.r.t. the set of variables of $-x^2y$ and $x^2y^3 \succ_{\mathsf{MON}} x^2y^2$.

5.21 Algorithm ($\mathcal{CHOOSE}$, [BenCherifa, 1986]) *In analogy with algorithm 5.17, we assume that a polynomial can be represented as a set of monomials each realized as a tuple* $(\alpha_{e_1\ldots e_n}, e_1\ldots e_n)$ *where* e_i *stands for the exponent of the variable* x_i *and* $\alpha_{e_1\ldots e_n}$ *for the coefficient of the monomial. Procedure* $\mathcal{SORT}$-$\mathcal{MON}$ *receives a list of monomials as input and returns a sorted (w.r.t. the ordering* $\succ_{\mathsf{MON}}$*, in descending order) copy of it. Function* $\mathcal{MIN}$-$\mathcal{MONS}(L, \alpha_{f_1\ldots f_n})$ *provides the list of all monomials of L which are minimal w.r.t. the set of variables specified by the indices of the monomial* $\alpha_{f_1\ldots f_n}$*. Procedure* $\mathcal{CHOOSE}$ *is featured by figure 5.3.* ◇

```
$\mathcal{CHOOSE}$ = proc(p : polynomial) returns (monomials)
    NL : Negative-Monomial-List := ()
    while there exists a negative monomial $\alpha_{f_1\ldots f_n}$ in p do
          NL := NL ∘ $\alpha_{f_1\ldots f_n}$
    end
    NL := $\mathcal{SORT}$-$\mathcal{MON}$(NL)
    $\alpha_{f_1\ldots f_n}$ := NL[1]
    PL : Positive-Monomial-List := ()
    while there exists a positive monomial $\alpha_{e_1\ldots e_n}$ in p do
          if    $\forall i \in [1, n]: e_i \geq f_i$
          then  PL := PL ∘ $\alpha_{e_1\ldots e_n}$
    end
    PL := $\mathcal{SORT}$-$\mathcal{MON}$($\mathcal{MIN}$-$\mathcal{MONS}$(PL , $\alpha_{f_1\ldots f_n}$))
    return (PL[1] , $\alpha_{f_1\ldots f_n}$)
end
```

Figure 5.3: Procedure $\mathcal{CHOOSE}$ of [BenCherifa, 1986]

5.22 Example (Example 5.20 Revisited) *By using procedure* $\mathcal{CHOOSE}$*, the following sequence, starting with polynomial p of example 5.20, will be generated:*

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

Note that the length of such a sequence $p_0, p_1, \cdots$ *strongly depends on procedure* $\mathcal{CHOOSE}$[13]. *For example, the length of the successful sequence presented in example 5.20 is 5 whereas* $\mathcal{CHOOSE}$ *generates a sequence of length 6.* ◇

In [Widiger, 1991], a modification of the BL-approach *was presented. This method assigns a constant* C *to* each *nullary (constant) operator of* $\mathcal{F}$ *by treating* C *as a variable. The proof of the positiveness of a polynomial is performed by using* C *instead of* μ. *A problem*
A.34 *concerning this technique is that the termination proof fails whenever two constants require*
A.35 different *interpretations.*

5.3.2 On the Time Complexity of the Approach of BenCherifa and Lescanne

In this subsection, the empirical time complexity of the BL-approach is discussed. We believe that, in many cases, procedure $\mathcal{CHOOSE}$ will directly generate a successful sequence proving the positiveness of a polynomial p if p is positive. In these cases,

the time complexity of procedure $\mathcal{POSITIVE}$ is $O(k \cdot lg\, k)$

where k represents the number of monomials occurring in p and lg denotes the *binary logarithm*. More precisely, the number of calls of procedure $\mathcal{CHANGE}$ lies between n and $m+n-1$ (where n and m are the numbers of negative and positive monomials, respectively). The case $m+n-1$ will be achieved if $m-1$ positive monomials are covered by a part of the negative ones and the *mth* positive monomial covers the remaining negative monomials. Procedure $\mathcal{SORT\text{-}MON}$[14] has the complexity[15] $O((n \cdot lg\, n \ + \ m \cdot lg\, m) \cdot v)$ where v is the number of different variables occurring in p.

Note that the complexity mentioned above can only be achieved if procedure $\mathcal{CHOOSE}$ *directly* finds a successful path through the corresponding tree of the polynomial. However, it is obvious that there exist positive polynomials (in the sense of $\mathcal{POSITIVE}$ with *another* variant of $\mathcal{CHOOSE}$) that cannot be proved positive by use of $\mathcal{CHOOSE}$ given in algorithm 5.21.

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

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

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

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

ends with a polynomial which is obviously positive. ◇

Thus, procedure $\mathcal{POSITIVE}$ is *incomplete.* Furthermore, it is somewhat difficult to give an exact complexity result since it depends on (the relationships between) the coefficients of the positive and the negative monomials and thus, on the interpretations of the operators.

---

[13] and thus on the values of the coefficients

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

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

For the remaining part of *this subsection*, we assume the BL-approach to be extended by *backtracking* (we call this method the 'extended BL-approach') which will be employed whenever the chosen sequence cannot prove the positiveness of the given polynomial.

As we have seen in the last subsection (example 5.20 and example 5.22), there exist different possibilities to *choose* negative and positive monomials. These manifold choices (produced by the extended BL-approach) can be described by a tree whose nodes represent the polynomials p_i generated from a given polynomial p[16].

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

5.24 **Example (Example 5.20 Revisited)** *Figure 5.4 illustrates the number of polynomials which can be created when starting with polynomial p of example 5.20. Due to the difficulty of graphically displaying all nodes, we present their numbers (beginning at level 4), only. Totally, 768 paths through this tree exist (although p is a relatively simple polynomial) of which 212 cannot be used to show the positiveness of p. Note that* $\Phi(p) = 1967$. ◇

The *best case*[17] for the complexity of the extended BL-approach is $O(n)$, where n represents the number of negative monomials occurring in p. This complexity can only be achieved if p is a polynomial such that each positive monomial can cover all negative monomials, i.e.

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

More formally: Let $p = \sum(\alpha_{i_1\ldots i_n} - \beta_{i_1\ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ such that $\alpha_{i_1\ldots i_n} \geq 0$, $\beta_{i_1\ldots i_n} \geq 0$ and $\alpha_{i_1\ldots i_n} \cdot \beta_{i_1\ldots i_n} = 0$[18]. Then the following two conditions must hold:

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

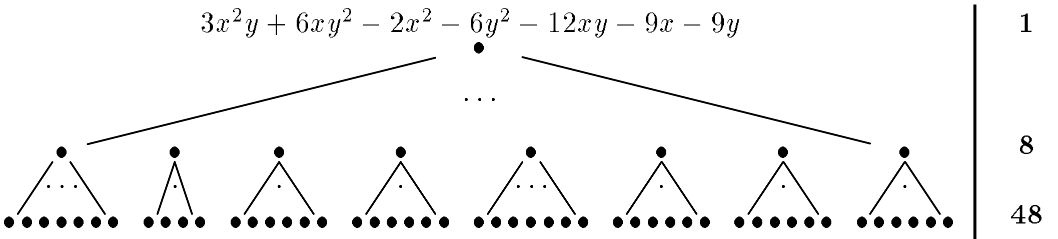
A typical polynomial, fulfilling these conditions, is considered in the following example.

---

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

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

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



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

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

0

1 2 3 3 **4 5 6** 6

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

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

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

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

⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕

Figure 5.4: Number of paths for the polynomial of example 5.24

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

The *worst case* for the extended BL-approach will occur if p is not greater than zero[19] and the corresponding tree is maximal. Furthermore, the exponents of each positive monomial have to cover those of every negative one. The tree will be maximal if it is of the form presented in figure 5.5. Let n be the number of negative monomials, m be the number of positive monomials occurring in a polynomial p and w.l.o.g. let $n = m + k$, $k \geq 0$[20]. Figure 5.5

---

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

[20] see lemma 5.26

represents the number of nodes at each level of the tree which corresponds to p. First of all, the number of negative monomials must be reduced[21] to the number of positive monomials (since $n \geq m$). Then n and m must be alternately decreased by 1[22]. Note that in figure 5.5 we will select m instead of n if $n = m$ holds.

$$1$$

$$nm$$

$$nm(n-1)m$$

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

$$\vdots$$

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

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

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

$$\vdots$$

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

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

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

$$\vdots$$

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

Figure 5.5: The worst case for the extended BL-approach

$\boxed{5.26}$ **Lemma** *Let p be a polynomial with m (≥ 1) positive monomials and n negative monomials. For the worst case of the extended* BL-approach, *$\Pi(p)$ and $\Phi(p)$ are as follows:*

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

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

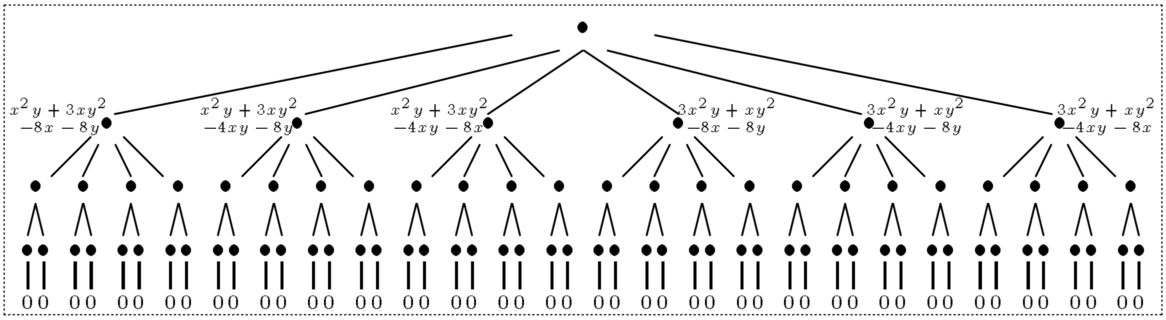
A polynomial composed of 4 negative and 3 positive monomials can correspond to a tree with 5184 leaves and 13729 nodes (for the worst case). A typical example of a tree for representing the worst case of the extended BL-approach is the following one.

$\boxed{5.27}$ **Example** *Let $p = 3x^2y + 3xy^2 - 4xy - 8x - 8y$, $\mu = 2$. All paths end with 0, i.e. p has a root at the position $(2,2)$ and thus $p \not\succ 0$ for $\mu = 2$:*

---

[21] in order to achieve the worst case

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



◇

The following lemma provides a rough lower and a rough upper bound for the number $\Phi(p)$ of nodes determined in lemma 5.26.

$\boxed{5.28}$ **Lemma** *Let $\Phi(p)$ and $\Pi(p)$ be defined as in lemma 5.26. Then*

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

◇

The bounds of lemma 5.28 define the time complexity (for the worst case) of the extended BL-approach as

$$O(m! \cdot n! \cdot m^{n-m}).$$

By using the formula of Stirling[23], the complexity changes to $O((mn)^n)$.

5.3.3 A Modification of the Approach of BenCherifa and Lescanne

In this subsection, we present another extension of the BL-approach. Instead of *backtracking* (as mentioned in the previous subsection), we introduce a *set of constraints* to avoid search without giving up the strength of the backtracking approach. The constraints help covering the entire search space of the BL-approach in just *one* computation. The reasons for modifying the BL-approach are the following ones: Since there exist only heuristics for procedure $\mathcal{CHOOSE}$ we possibly need to backtrack in order to find a successful path (and this considerably extends the time complexity of algorithm $\mathcal{POSITIVE}$, see subsection 5.3.2). Another more important aspect concerns the generation of an interpretation for a given TRS. Unlike the (extended) BL-approach, our algorithm can easily be integrated into a procedure for generating polynomial orderings (see subsection 5.5.2). First of all, we illustrate the basic ideas of our technique.

- One of the main features of our algorithm is identical to the basic idea of the BL-approach:
$$x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \;\geq\; \mu^{\Sigma(k_j - i_j)} \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} \quad \text{if } \forall j\text{: } k_j \geq i_j$$
This inequality is used in (ii) of the following item.
- The algorithm transforms a polynomial p (that we want to be proved positive) into a set of linear[24] inequalities. Let $p = \sum(\alpha_{i_1 \ldots i_n} - \beta_{i_1 \ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ such that

---

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

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

$\alpha_{i_1\ldots i_n} \geq 0, \beta_{i_1\ldots i_n} \geq 0$ and $\alpha_{i_1\ldots i_n} \cdot \beta_{i_1\ldots i_n} = 0$. The transformation is based on the following three steps:

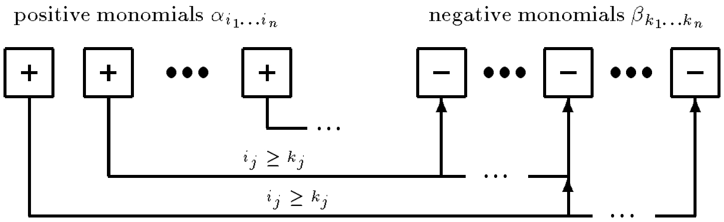
(i) **Dividing:** Each positive coefficient $\alpha_{i_1\ldots i_n}$ will be split into a sum of new variables $\gamma_{i_1\ldots i_n k_1\ldots k_n}$ over $\mathbb{R}$ such that each item of the sum corresponds to a negative coefficient $\beta_{k_1\ldots k_n}$ if, and only if, $i_j \geq k_j$ holds for all j. Figure 5.6 provides a graphical illustration of this process.

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

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

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

(iii) **Solving:** The set of the inequalities generated in (i) and (ii) can be solved or disproved by applying a decision procedure[25)] for linear inequalities. For solving linear inequalities we can use, for example, the method of complete description.



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

Figure 5.6: Division process of algorithm 5.29

The formal description of the above ideas can be found in the following algorithm.

$\boxed{5.29}$ **Algorithm (Constraint-Based Algorithm)** *This algorithm determines whether a polynomial p is positive in the sense of $\sqsupset$. Let $p = \sum(\alpha_{i_1\ldots i_n} - \beta_{i_1\ldots i_n}) \cdot x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ such that $\alpha_{i_1\ldots i_n} \geq 0, \beta_{i_1\ldots i_n} \geq 0, \alpha_{i_1\ldots i_n} \cdot \beta_{i_1\ldots i_n} = 0$ and let $\gamma_{l_1\ldots l_n k_1\ldots k_n}$ be new variables over $\mathbb{R}$. μ is defined as usual.*

1. **Dividing:** *Let* $I_1 := \{\alpha_{i_1\ldots i_n} \geq \sum\limits_{k_j \leq i_j} \gamma_{i_1\ldots i_n k_1\ldots k_n} \mid \alpha_{i_1\ldots i_n} > 0, \beta_{k_1\ldots k_n} > 0\}$.
   If $I_1 = I_1' \cup \{\alpha_{i_1\ldots i_n} \geq \gamma_{i_1\ldots i_n k_1\ldots k_n}\}$
   then let $I_1 := I_1' \cup \{\gamma_{i_1\ldots i_n k_1\ldots k_n} := \alpha_{i_1\ldots i_n}\}$.

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

2. Distributing: *Let* $I_2 := \{ \sum_{k_j \geq i_j} \mu^{\Sigma(k_j - i_j)} \cdot \gamma_{k_1 \ldots k_{n_{i_1 \ldots i_n}}} \geq \beta_{i_1 \ldots i_n} \mid \beta_{i_1 \ldots i_n} > 0, \alpha_{k_1 \ldots k_n} > 0\}$

3. Solving: *Let* $I := \{l_i \geq r_i \mid l_i \geq r_i \in I_1 \cup I_2\} \cup \{\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}} \geq 0\}$. *Generate the values for* $\gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}$ *of* I *with the help of a decision procedure for linear inequalities*[26]. $p \sqsupset 0$ *w.r.t.* μ *if* I *has a solution.* ◇

The additional condition '*If* $I_1 = I_1' \cup \{\alpha_{i_1 \ldots i_n} \geq \gamma_{i_1 \ldots i_{n_{k_1 \ldots k_n}}}\}$' of step 1 in algorithm 5.29 refers to the case that a positive monomial can cover only one negative monomial. For example, the polynomial

$$p = 4x^2 + 5y^2 - 3x - 2$$

implies that y^2 covers only -2. Then the algorithm generates the sets $I_1 := \{4 \geq \gamma_{20_{10}} + \gamma_{20_{00}} , \gamma_{\mathbf{02}_{\mathbf{00}}} := \mathbf{5}\}$ and $I_2 := \{\mu \cdot \gamma_{20_{10}} \geq 3 , \mu^2 \cdot \gamma_{20_{00}} + \mu^2 \cdot \mathbf{5} \geq 2\}$.

A detailed example, illustrating the steps of algorithm 5.29, is presented now.

$\boxed{5.30}$ **Example (Example 5.20 Revisited)** *In order to prove* $p = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$ *to be positive w.r.t.* $\mu = 3$, *we perform the following actions:*

1. Dividing:

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

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

2. Distributing:

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

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

3. Solving:

$$\begin{array}{rlcl} I := \{ & 3 \geq \gamma_{21_{20}} + \gamma_{21_{11}} + \gamma_{21_{10}} + \gamma_{21_{01}}, & & \\ & 6 \geq \gamma_{12_{02}} + \gamma_{12_{11}} + \gamma_{12_{10}} + \gamma_{12_{01}}, & & \\ & 3\gamma_{21_{20}} & \geq & 2, \\ & \gamma_{12_{02}} & \geq & 2, \\ & \gamma_{21_{11}} + \gamma_{12_{11}} & \geq & 4, \\ & \gamma_{21_{10}} + \gamma_{12_{10}} & \geq & 1, \\ & \gamma_{21_{01}} + \gamma_{12_{01}} & \geq & 1, \\ & \gamma_{ij_{kl}} & \geq & 0 \ \} \end{array}$$

A decision procedure for linear inequalities solves the union $I_1 \cup I_2$. In order to guarantee $p \sqsupset 0$ (not only $p \sqsupseteq 0$), at least one inequality of $I_1 \cup I_2$ must be a proper one. This will be guaranteed by the method of complete description (see subsection 4.4.2)

A solution of I *includes the following values:* $\gamma_{21_{20}} = 1$, $\gamma_{21_{11}} = 0$, $\gamma_{21_{10}} = 1$, $\gamma_{21_{01}} = 1$, $\gamma_{12_{02}} = 2$, $\gamma_{12_{11}} = 4$, $\gamma_{12_{10}} = 0$, $\gamma_{12_{01}} = 0$. ◇

---

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

$\boxed{5.31}$ **Example (Example 5.23 Revisited)** *We once again consider example 5.23. It has been shown that the polynomial*

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

cannot be proved to be positive w.r.t. $\mu = 2$ *with the help of procedure* $\mathcal{CHOOSE}$ *described in algorithm 5.21. It is very easy to prove the positiveness of* p *by using algorithm 5.29:*

$$\begin{array}{lcl} I_1 & := & \{ \quad 1 \geq \gamma_{{420}_{300}} + \gamma_{{420}_{220}}, \\ & & \qquad \gamma_{{311}_{300}} := 3 \ \} \\ I_2 & := & \{ \quad 8\gamma_{{420}_{300}} + 4\gamma_{{311}_{300}} \geq 8, \\ & & \qquad 4\gamma_{{420}_{220}} \geq 4 \ \} \\ & & \\ I & := & \{ \quad 1 \geq \gamma_{{420}_{300}} + \gamma_{{420}_{220}}, \\ & & \qquad \gamma_{{311}_{300}} = 3, \\ & & \qquad 2\gamma_{{420}_{300}} + 3 \geq 2, \\ & & \qquad \gamma_{{420}_{220}} \geq 1\} \end{array}$$

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

$\boxed{5.32}$ **Theorem** *Algorithm 5.29 always terminates. If it does not fail*[27]*,* $p \sqsupset 0$ *holds.* ◇

Obviously, algorithm 5.29 is no decision procedure since it deals with polynomials over $\mathbb{N}$. However, it is complete w.r.t. the BL-approach, i.e. all computations of the BL-approach *with backtracking* are covered with one computation.

Our technique enables the generation of a correct sequence $p = p_0 \sqsupseteq p_1 \sqsupseteq \cdots \sqsupseteq p_n \sqsupset 0$ for the BL-approach. The deduced solution $\bigcup\{\gamma_{{k_1\ldots k_n}_{i_1\ldots i_n}}\}$ of algorithm 5.29 will be used in the following way: The part $\gamma_{{k_1\ldots k_n}_{i_1\ldots i_n}}$ of $\alpha_{k_1\ldots k_n}$ will be taken to cover $\beta_{i_1\ldots i_n}$, i.e. each element of the set $\bigcup\{\gamma_{{k_1\ldots k_n}_{i_1\ldots i_n}}\}$ will be considered. Each item $\gamma_{{k_1\ldots k_n}_{i_1\ldots i_n}} > 0$ corresponds to a transition $p_j \rightarrowtail p_{j+1}$ where the negative monomial $\beta_{i_1\ldots i_n}$ and the positive monomial $\alpha_{k_1\ldots k_n}$ will be added (see the following example).

$\boxed{5.33}$ **Example (Example 5.20 Revisited)** *Let* $p = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y$ *and* $\mu = 3$ *as given in examples 5.20 and 5.30. Furthermore, we use the following solution created in example 5.30:*

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

This solution implies the following sequence associated to the BL-approach:

$$\begin{array}{c} p_0 = 3x^2y + 6xy^2 - 2x^2 - 6y^2 - 12xy - 9x - 9y \\ \Downarrow \ \gamma_{{21}_{20}} = 1^{28)} \\ p_0 = \mathbf{x^2y} + 2x^2y + 6xy^2 - \mathbf{2x^2} - 6y^2 - 12xy - 9x - 9y \\ p_1 = \frac{1}{3}x^2y + 2x^2y + 6xy^2 - 6y^2 - 12xy - 9x - 9y \\ \Downarrow \ \gamma_{{21}_{10}} = 1^{29)} \\ p_1 = \frac{1}{3}x^2y + \mathbf{x^2y} + x^2y + 6xy^2 - 6y^2 - 12xy - \mathbf{9x} - 9y \end{array}$$

---

[27] Algorithm 5.29 will fail, if there is no solution for the set I of part 3.

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

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

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

◇

It is obvious that the more zeros the solution vector of algorithm 5.29 contains[30)] the shorter the associated sequence, generated by the BL-approach, will be.

Note that the BL-approach extended by backtracking and the algorithm of this subsection have the same power, i.e. whenever a polynomial can be proved to be positive using the extended BL-approach it can also be proved positive with the help of algorithm 5.29, and vice versa. In general, algorithm 5.29 is more powerful than the original BL-approach (*without* backtracking, see example 5.31).

5.3.4 Improving the BenCherifa and Lescanne Method

Most of the TRSs occurring in the literature are ordered w.r.t. a polynomial ordering based on the BL-approach. Certain examples require a more powerful ordering on polynomials. This can be illustrated by a simple TRS.

$\boxed{5.34}$ **Example** *Let $\mathcal{R}$ be*

$$\begin{cases} 1: & x \wedge (y \vee z) & \rightarrow & (x \wedge y) \vee (x \wedge z) \\ 2: & (\neg x) \vee y & \rightarrow & x \supset y \\ 3: & (\neg x) \supset (\neg y) & \rightarrow & y \supset (x \wedge y) \end{cases}$$

$\mathcal{R}$ cannot be oriented with the help of any polynomial ordering which is realized using the BL-approach. *The reasons are the following ones: In order to orient rule 1, $[\![\wedge]\!](x,y)$ must be mixed and $[\![\vee]\!](x,y)$ linear*[31)]. *Thus, rule 2 implies that $[\![\supset]\!](x,y)$ cannot be mixed. Therefore, irrespective of $[\![\neg]\!](x)$, rule 3 cannot be oriented as demanded, since the left-hand side only contains monomials of the form αx^i or βy^j whereas the right-hand side includes at least one monomial like $\gamma x^m y^n$ (where $m, n \geq 1$).* ◇

In order to prove the termination of TRSs like that in the previous example, we introduce a new technique for proving polynomials to be positive. The example discussed above requires to show that $\alpha x^i + \beta y^j \sqsupset \gamma x^m y^n$ holds for specific α, β, γ and i, j, m, n. The following lemma specifies possible values for these variables. It is a direct consequence of the inequality $(a+b)^2 \geq 4ab$ if $a, b > 0$.

[30)] i.e. $\gamma_{i_1 \ldots i_n k_1 \ldots k_n} = 0$

[31)] see lemma 5.67 on page 125 in subsection 5.5.1

5.35 **Lemma** *Let* $\alpha, \beta, x_l > 0$:

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

5.36 **Example** *Lemma 5.35 is able to prove, for example, the following inequalities:*

- $x^2 + y^2 \sqsupseteq 2xy$
- $x^2yz^6 + x^2y^3 \sqsupseteq 2x^2y^2z^3$
- $x^3 + y^3 \sqsupseteq 2x^{\frac{3}{2}}y^{\frac{3}{2}}$
- $x^2 + 4y^2 \sqsupseteq 4xy$

The BL-approach *cannot show the validity of these inequalities. Note that it is possible to repeatedly apply lemma 5.35:*

$$\begin{array}{c|c}
\mathbf{x^4} + \mathbf{y^4} + 2z^2 & x^4 + \mathbf{y^4} + \mathbf{2z^2} \\
\sqsupseteq & \sqsupseteq \\
\mathbf{2x^2y^2} + \mathbf{2z^2} & \mathbf{x^4} + \mathbf{2\sqrt{2} \cdot y^2z} \\
\sqsupseteq & \sqsupseteq \\
4xyz & 2\sqrt{2\sqrt{2}} \cdot x^2yz^{\frac{1}{2}}
\end{array} \qquad \diamond$$

The basic idea of lemma 5.35 can be extended in another way. The following lemma describes a situation where a nearly arbitrary number of monomials for covering one monomial is admitted.

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

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

5.38 **Remark**

This lemma is derived from the so-called arithmetic-mean-geometric-mean inequality (see, for example, [Hardy *et al.*, 1952]):

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

A detailed proof of this classical inequality is contained in [Hardy *et al.*, 1952]. For our purpose, the case $\forall i \in [1,n]$: $p_i = 1$ is sufficient. Therefore, a slight transformation of the inequality given above leads to

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

Note that lemma 5.35 is an implication of this inequality when assigning 2 to n.

The application of lemma 5.37 is based on the following consideration: If we want to cover a monomial $m \cdot x_1^{k_1} \cdot\ldots\cdot x_n^{k_n}$, a polynomial p consisting of m monomials must exist. Furthermore, the arithmetic mean of the sum of all exponents of p w.r.t. the variable x_i has to be identical to (or greater than) k_i.

5.39 **Example** *The following inequalities can be proved by using lemma 5.37:*

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

The class of polynomials that is successfully managed by the method corresponding to lemma 5.37 overlaps with that of lemma 5.35 (see example 5.36 and example 5.39) as well as with that of the BL-approach. However, *the combination of these processes is far more powerful than each method by itself.* More precisely, the functions $\mathcal{POSITIVE}$, $\mathcal{CHOOSE}$ and $\mathcal{CHANGE}$ should be extended by incorporating a test embodying lemma 5.37 (and/or lemma 5.35). Note that the condition $\forall l \in [1, n]$: $k_l \leq \frac{1}{m} \cdot \sum_{j=1}^{m} i_{l_j}$ of lemma 5.37 can be achieved by substituting μ for each additional variable and splitting monomials w.r.t. coefficients[32]. For example, let $\mu = 2$. Then $x^2z + y^2 \sqsupset 2xy$ since $x^2z + y^2 \sqsupseteq 2x^2 + y^2 = x^2 + x^2 + y^2$, $x^2 + y^2 \sqsupseteq^{33)} 2xy$ and $x^2 \sqsupset 0$.

$\boxed{5.40}$ **Example** *Let* $\mu = 2$. *We show that* $p = 2x^3 + y^2z^2 + yz^2 - x^2 - 3xyz - 2yz \sqsupset 0$ *holds.*

$$\begin{aligned} p = p_0 &= \mathbf{2x^3} + \mathbf{y^2z^2} + \mathbf{yz^2} - x^2 - \mathbf{3xyz} - 2yz \\ \sqsupseteq {p_1}^{34)} &= x^3 + \mathbf{y^2z} - x^2 - \mathbf{2yz} \\ \sqsupseteq p_2 &= \mathbf{x^3} - \mathbf{x^2} \\ \sqsupseteq p_3 &= \tfrac{1}{2}x^3 \end{aligned}$$

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

A reduction of the time complexity of the extended procedure $\mathcal{POSITIVE}$, i.e. the mixture of the original procedure $\mathcal{POSITIVE}$ and lemmata 5.35 and 5.37, can be achieved by only applying the part corresponding to lemma 5.37 (lemma 5.35) in case the original algorithm fails (see example 5.40).

$\boxed{5.41}$ **Example (Example 5.34 Revisited)** *The TRS* $\mathcal{R}$ *in example 5.34 is ordered w.r.t. the* POL *with* $[\![\wedge]\!](x,y) = xy + x$, $[\![\neg]\!](x) = x^2$, $[\![\vee]\!](x,y) = [\![\supset]\!](x,y) = x + y + 2$ *and* $\mu \geq 3$. *The positiveness of the polynomials produced by the first two rules can be proved using the* BL-approach. *Now, consider the last rule:* $[\![(\neg x) \supset (\neg y)]\!](x,y) = x^2 + y^2 + 2$, $[\![y \supset (x \wedge y)]\!](x,y) = xy + x + y + 2$ *and* $x^2 + y^2 - xy - x - y \sqsupset 0$ *since* $x^2 + y^2 \sqsupseteq 2xy$ *and* $xy \sqsupset x + y$ *if* $\mu \geq 3$. ◇

As we have seen (in example 5.34), lemma 5.37 (as well as lemma 5.35) extends the power of procedure $\mathcal{POSITIVE}$. In addition, its use can sometimes simplify the form of the interpretations of operators needed for a termination proof. Let us consider an example.

$\boxed{5.42}$ **Example** *The TRS* $\mathcal{R}$

$$\left\{ \begin{array}{lll} 1: & (x \star y) \star z & \rightarrow x \star (y \star z) \\ 2: & i(x \star y) & \rightarrow i(y) \star i(x) \\ 3: & i(x) + i(y) & \rightarrow x \star y \end{array} \right.$$

needs a mixed interpretation for the operator $\star$[35] *which implies that* $[\![+]\!](x,y)$ *must also be mixed for applying the original* BL-approach. *For example,* $\mathcal{R}$ *will be ordered w.r.t. the interpretations*

---

[32] Analogous considerations are also helpful for lemma 5.35.

[33] because of lemma 5.35 and lemma 5.37

[34] since $y^2z^2 \sqsupseteq 2y^2z$ and $x^3 + y^2z + yz^2 \sqsupseteq 3xyz$.

[35] see the third part of lemma 5.67 on page 125

$$[\![\star]\!](x,y) = 2xy + x, [\![i]\!](x) = x^2 \text{ and } [\![+]\!](x,y) = xy$$

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

The property of simplifying the interpretation of an operator f, when lemma 5.37 or lemma 5.35 is used, will often hold if f is completely defined by *one* rule. This means that $f(x_1,...,x_n) \to t$ is part of $\mathcal{R}$ and $[\![t]\!]$ is any strictly mixed polynomial. Then an easier polynomial for $[\![f]\!](x_1,...,x_n)$ can be helpful for orienting another rule containing f in the presumably smaller side.

5.3.5 The BenCherifa and Lescanne Approach for Restricted Systems

This subsection deals with the effect of restricted TRSs on the BL-approach. We investigate systems with restrictions on the occurrences of constants. As mentioned in section 5.1 and in subsection 5.3.1, the constants of $\mathcal{F}$ are strongly influencing the application of polynomial orderings in general, as well as that of the BL-approach in particular (see remark 5.4 on page 95). For example, increasing the interpretations of constants can lead to the orientation of an equation or to easier interpretations of other operators.

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

$$(x.y) \circ z \;\to\; x.(y \circ z)$$

and $\mu = 1$. With the help of the interpretations $[\![\circ]\!](x,y) = x^2y$ and $[\![.]\!](x,y) = x + 2y$, $\mathcal{R}$ is ordered w.r.t. the POL. *However, setting μ to 2, the interpretations $[\![\circ]\!](x,y) = \mathbf{xy}$ and $[\![.]\!](x,y) = x + 2y$ are sufficient. Note that the latter interpretations cannot prove the termination of $\mathcal{R}$ if $\mu = 1$.* ◇

In the case of $\mathcal{R}$ not containing any constant, the BL-approach requires the lower bound μ[36] to be explicitly set. However, if the BL-approach is changed such that μ is no longer considered, the resulting method is not only more efficient, but also more powerful. The improvement is based on the following observation.

5.44 Remark

The monomial x^2y is greater than the monomial αxy since there *exists* a μ $(= \alpha + 1)$ such that $\forall x, y \geq \mu$: $x^2y > \alpha xy$. Thus, the knowledge of the *existence* of such a μ is sufficient to guarantee the positiveness of $x^2y - \alpha xy$.

The following definition generalizes the brief comment of remark 5.44.

5.45 Definition (Ordering on Polynomials)

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

---

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

$$m_1 \succ_{\mathsf{M}} m_2 \iff \forall j \in [1,n]\colon\ i_j \geq k_j \ \wedge\ \exists l \in [1,n]\colon\ i_l > k_l$$
$$\text{or } \forall j \in [1,n]\colon\ i_j = k_j \ \wedge\ \alpha_1 > \alpha_2$$

$$m_1 \sim_{\mathsf{M}} m_2 \iff \forall j \in [1,n]\colon\ i_j = k_j \ \wedge\ \alpha_1 = \alpha_2$$

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

$$p_1 \sqsupset_P p_2 \iff \{\!\{\, m_{11},\ldots,m_{1l_1} \,\}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{M}} \{\!\{\, m_{21},\ldots,m_{2l_2} \,\}\!\} \qquad \diamond$$

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

$\boxed{5.46}$ **Example** *Let* $p_1(x,y) = x^2y + xy^2$ *and* $p_2(x,y) = 2x^2 + 3xy + 5y^2$*. Then* $p_1(x,y) \sqsupset_P p_2(x,y)$ *since* $x^2y \succ_{\mathsf{M}} 2x^2$*,* $x^2y \succ_{\mathsf{M}} 3xy$ *and* $xy^2 \succ_{\mathsf{M}} 5y^2$*. Note that the* BL-approach *cannot prove the positiveness of* $p_1(x,y) - p_2(x,y)$ *if* $\mu \leq 5$. $\diamond$

A.9
A.11
A.12
A.14
A.16
A.18
A.30
A.31

The most important feature of the definition of $\sqsupset_P$ is the sufficiency of the existence of a μ[37]. However, if there exists just one constant in $\mathcal{R}$, then μ has to be set to a value and we have to check $p \sqsupset 0$ w.r.t. μ. Thus, $\sqsupset_P$ can only be applied if $\mathcal{R}$ does not contain any constants.

$\boxed{5.47}$ **Lemma** *The ordering* $\succ_{\mathsf{P}}$ *defined by* $s \succ_{\mathsf{P}} t$ *if, and only if,* $[\![s]\!] \sqsupset_P [\![t]\!]$ *is a reduction ordering on* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *if no constants in* $\mathcal{F}$ *exist.* $\diamond$

$\sqsupset_P$ is a *generalization* of the ordering realized by the BL-approach if no constants exist (see example 5.46). Therefore,

---

the occurrence of constants reduces the power of polynomial orderings.

With the help of this observation, one could replace all constants by variables before applying a polynomial ordering. This pre-processing has the following consequences:

- Obviously, it is *always* more convenient to apply $\sqsupset_P$ (i.e. definition 5.45) instead of the BL-approach to a system $\mathcal{R}$ containing no constants on the presumably smaller sides since definition 5.45 is more powerful (in this case) than the BL-approach (even *with* backtracking) as well as more efficient. Note that for proving the termination of the 'generalized' TRS, the same interpretations will suffice as those for the 'original' TRS.

  $\boxed{5.48}$ **Example** *Let* $\mathcal{R}$ *be composed of the simple rules*

$$\begin{cases} 1: & 0 + y \rightarrow y \\ 2: & s(x) + y \rightarrow s(x+y) \end{cases}$$

  specifying the addition on natural numbers. These rules are ordered w.r.t. the interpretations $[\![+]\!](x,y) = 2x + y$*,* $[\![s]\!](x) = x + 1$ *and* $[\![0]\!]() = 1$*. The same interpretations (with the same* μ*, which is equal to 1) will also prove the termination of* $\mathcal{R}$ *if rule 1 is replaced by* $x + y \rightarrow y$*.* $\diamond$

---

[37] Definition 5.45 *implicitly* depends on the *existence* of a μ.

- Mostly, $\sqsupset_P$ will be helpful (even when using the same interpretations) if for each rule the set of constants occurring in the presumably smaller side is included in that of the presumably greater one. A.10 A.13 A.15 A.17 A.24 A.25 A.35 A.36

$\boxed{5.49}$ **Example ([Paul, 1984])** *The TRS $\mathcal{R}$*

$$\begin{cases} x+0 & \rightarrow & x \\ s(0) & \rightarrow & 0 \\ s(1) & \rightarrow & 1 \\ s(x+1) & \rightarrow & s(x)+(x+1) \end{cases}$$

as well as its 'generalized' version $\mathcal{R}'$

$$\begin{cases} x+y & \rightarrow & x \\ s(x) & \rightarrow & x \\ s(x+y) & \rightarrow & s(x)+(x+y) \end{cases}$$

can both be oriented with the help of the interpretations $\llbracket + \rrbracket(x,y) = x+y$, $\llbracket s \rrbracket(x) = x^2$ (and $\llbracket 0 \rrbracket() = \llbracket 1 \rrbracket() = 2$ for $\mathcal{R}$). ◇

- The replacement of constants, occurring only on the presumably smaller side of a rule, by variables[38] can lead to the necessity of changing certain interpretations to more complex ones. A.26 A.28 A.29

$\boxed{5.50}$ **Example** *Consider the rule* A.28

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

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

In some cases, the substitution of constants, that occur on the presumably smaller sides of the rules, by variables can generate rules which can no longer be oriented: $a+b \rightarrow b+a$ where $a,b \in \mathcal{F}$. A.34 A.35

5.4 Polynomial Orderings for Certain Theories

As mentioned in the introduction, the use of TRSs based on an additional underlying theory $\mathcal{E}$ presumes a special termination property. Here, we consider some theories and define appropriate polynomial orderings that can be used to prove $\mathcal{E}$-termination. It is well-known that polynomial orderings can also be restricted to AC-theories.

Idea of Comparison *The* AC-POL *is identical to the* POL *on special interpretations of AC-operators.*

---

[38] of the presumably greater sides

A.1 A.2 A.3 A.5 A.6 A.19 A.33 A.37

$\boxed{5.51}$ **Definition (Associative-Commutative Polynomial Ordering, [Lankford, 1979] and [BenCherifa, 1986])** *Let* $[\![.]\!]$ *be a polynomial interpretation such that* $\forall f \in \mathcal{F}_{AC}$: $\exists \alpha, \beta, \gamma \in \mathbb{R}_+ \colon [\![f]\!](x,y) = \alpha xy + \beta(x+y) + \gamma$ *with* $\alpha\gamma + \beta = \beta^2$. *Then the* associative-commutative polynomial ordering AC-POL *on terms is defined as*

$$\succ_{\mathsf{AC\text{-}POL}} \;\equiv\; \succ_{\mathsf{POL}}\langle \mathcal{I}d \rangle \qquad \diamond$$

In [Narendran and Rusinowitch, 1991], a reduction ordering for ground terms is constructed which is AC-compatible and total on the set of congruence classes generated by the AC-axioms. Instead of taking numerical values for the coefficients a, b, c of interpretations for AC-operators (see the definition given above), the ordering takes integer polynomials for them.

In order to guarantee the AC-compatibility, the interpretations of AC-operators must be of a special form. The following lemma generalizes this concept.

$\boxed{5.52}$ **Lemma** *The* POL *is* $\mathcal{E}$*-compatible if*

$$\forall s,t \in \mathcal{T}(\mathcal{F},\mathcal{X}) \colon s \overset{*}{\leftrightarrow}_{\mathcal{E}} t \;\leadsto\; s \sim_{\mathsf{POL}} t. \qquad \diamond$$

This property allows the handling of various theories[39], apart from AC. BenCherifa ([BenCherifa, 1986]) tests a few theories for their compatibility. This section deals with the extension of her catalogue.

In the rest of this section we present the conditions for the interpretations of $\mathcal{E}$-operators under which the induced polynomial ordering is $\mathcal{E}$-compatible

---

where $\mathcal{E}$ contains distributivity or endomorphism axioms[40]. Furthermore, assertions about special theories like a kind of if-then-else equality and Boolean rings are given.

First of all, we deal with distributivity axioms. There exist two different types of distributivity:

Left distributivity: $f(g(x,y),z) \leftrightarrow_{\mathcal{E}} g(f(x,z),f(y,z))$ and
Right distributivity: $f(x,g(y,z)) \leftrightarrow_{\mathcal{E}} g(f(x,y),f(x,z))$

if $f, g \in \mathcal{F}$ and $\mathcal{A}r(f) = \mathcal{A}r(g) = 2$.

$\boxed{5.53}$ **Lemma (Left Distributivity)** $f(g(x,y),z) \sim_{\mathsf{POL}} g(f(x,z),f(y,z))$ *if one of the following conditions holds:*

- $[\![f]\!](x,y) = \sum_i \alpha_i x y^i$ $\quad [\![g]\!](x,y) = \beta_1 x + \beta_2 y$
- A.4 $[\![f]\!](x,y) = \alpha_1 xy + \alpha_2(x+y) + \alpha_3$ $\quad [\![g]\!](x,y) = x + y + \beta_3$ $\quad \wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1), \alpha_2 = \alpha_1\beta_3$ $\diamond$

Special cases of the interpretations chosen in the lemma given above are the following ones: (*i*) $[\![f]\!](x,y) = xy, [\![g]\!](x,y) = \beta(x+y)$ for arbitrary values of β, (*ii*) $[\![f]\!](x,y) = \alpha x(y+1), [\![g]\!](x,y) = \beta(x+y)$ for arbitrary values of α and β, (*iii*) $[\![f]\!](x,y) = xy+x+y, [\![g]\!](x,y) = x+y+1$.

---

[39] Obviously, lemma 5.52 does not hold for the SPOL when choosing $\mathcal{E}$, for example, to be $f(x) \leftrightarrow_{\mathcal{E}} x$ (since $f(x) \succ_{\mathsf{EMB}} x$).

[40] For simplicity, we use $\sum_i$ instead of $\sum_{i=0}^{n}$ for an arbitrary n.

$\boxed{5.54}$ **Lemma (Right Distributivity)** *$f(x, g(y, z)) \succ_{\mathsf{POL}} g(f(x, y), f(x, z))$ if one of the following conditions holds:*

- $[\![f]\!](x, y) = \sum\limits_i \alpha_i x^i y$ $\quad [\![g]\!](x, y) = \beta_1 x + \beta_2 y$
- $[\![f]\!](x, y) = \alpha_1 xy + \alpha_2(x + y) + \alpha_3$ $\quad [\![g]\!](x, y) = x + y + \beta_3$
  $\wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1), \alpha_2 = \alpha_1\beta_3$ ◇

Some interesting special cases of the interpretations given above are the following ones: (i) $[\![f]\!](x, y) = xy, [\![g]\!](x, y) = \beta(x + y)$ for arbitrary values of β, (ii) $[\![f]\!](x, y) = \alpha y(x + 1), [\![g]\!](x, y) = \beta(x + y)$ for arbitrary values of α and β, (iii) $[\![f]\!](x, y) = xy + x + y, [\![g]\!](x, y) = x + y + 1$.

Note that one cannot combine neither the (left-) associative rule $f(f(x, y), z) \to f(x, f(y, z))$ with the underlying right distributivity nor the (right-) associative rule $f(x, f(y, z)) \to f(f(x, y), z)$ with the underlying left distributivity.

$\boxed{5.55}$ **Lemma (Left and Right Distributivity)** *$f(g(x, y), z) \succ_{\mathsf{POL}} g(f(x, z), f(y, z)) \wedge f(x, g(y, z)) \succ_{\mathsf{POL}} g(f(x, y), f(x, z))$ if one of the following conditions holds:*

- $[\![f]\!](x, y) = \alpha xy$ $\quad [\![g]\!](x, y) = \beta_1 x + \beta_2 y$ |A.22
- $[\![f]\!](x, y) = \alpha_1 xy + \alpha_2(x + y) + \alpha_3$ $\quad [\![g]\!](x, y) = x + y + \beta_3$
  $\wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1), \alpha_2 = \alpha_1\beta_3$ ◇

The interpretations for left and right distributivity represents the intersection of the interpretations for the left distributivity and those for the right distributivity.

$\boxed{5.56}$ **Lemma** *There exists no* POL *which induces the equality of both sides of the following combinations of axioms:*

- $f(g(x, y), z) \leftrightarrow_{\mathcal{E}} g(f(x, z), f(y, z)) \quad \wedge \quad g(x, f(y, z)) \leftrightarrow_{\mathcal{E}} f(g(x, y), g(x, z))$
- $f(g(x, y), z) \leftrightarrow_{\mathcal{E}} g(f(x, z), f(y, z)) \quad \wedge \quad g(f(x, y), z) \leftrightarrow_{\mathcal{E}} f(g(x, z), g(y, z))$
- $f(x, g(y, z)) \leftrightarrow_{\mathcal{E}} g(f(x, y), f(x, z)) \quad \wedge \quad g(x, f(y, z)) \leftrightarrow_{\mathcal{E}} f(g(x, y), g(x, z))$ ◇

Another often occurring axiom is that of the *endomorphism*:

$$h(f(x, y)) \leftrightarrow_{\mathcal{E}} f(h(x), h(y))$$

if $f, h \in \mathcal{F}$ and $\mathcal{A}r(f) = 2, \mathcal{A}r(h) = 1$.

$\boxed{5.57}$ **Lemma (Endomorphism)** *$h(f(x, y)) \succ_{\mathsf{POL}} f(h(x), h(y))$ if one of the following conditions holds:*

- $[\![f]\!](x, y) = \sum\limits_{i,j} \alpha_{ij} x^i y^j$ $\quad [\![h]\!](x) = x$
- $[\![f]\!](x, y) = \alpha_1 x + \alpha_2 y + \alpha_3$ $\quad [\![h]\!](x) = \beta_1 x + \beta_2$
  $\wedge\ \alpha_3(\beta_1 - 1) = \beta_2(\alpha_1 + \alpha_2 - 1)$
- $[\![f]\!](x, y) = \alpha x^i y^j$ $\quad [\![h]\!](x) = \beta x^k$
  $\wedge\ \beta^{i+j-1} = \alpha^{k-1}$ ◇

The following special cases of the interpretations used in lemma 5.57 are the most interesting ones: (i) $[\![f]\!](x,y)=\alpha_1x+\alpha_2y, [\![h]\!](x)=\beta x$ for arbitrary values of α_1, α_2 and β, (ii) $[\![f]\!](x,y)=\alpha_1x+\alpha_2y+\alpha_3, [\![h]\!](x)=x$ for arbitrary values of α_1, α_2 and α_3, (iii) $[\![f]\!](x,y)=x+y+\alpha_3, [\![h]\!](x)=\beta_1x+\alpha_3(\beta_1-1)$ for arbitrary values of α_3 and β_1, (iv) $[\![f]\!](x,y)=x^iy^j, [\![h]\!](x)=x^k$ for arbitrary values of i, j and k.

In some instances a slightly different endomorphism axiom occurs: $h(f(x,y)) \leftrightarrow_{\mathcal{E}} f(h(y),h(x))$ where the arguments of f have been exchanged. If the interpretations of the following lemma are given, both sides of this axiom became identical w.r.t. the POL.

$\boxed{5.58}$ **Lemma** *$h(f(x,y)) \sim_{\mathsf{POL}} f(h(y),h(x))$ if one of the following conditions holds:*

- $[\![f]\!](x,y)=\sum\limits_{i,j}\alpha_{ij}x^iy^j \quad [\![h]\!](x)=x$
  $\wedge\ \forall i,j\colon \alpha_{ij}=\alpha_{ji}$
- $[\![f]\!](x,y)=\alpha_1(x+y)+\alpha_3 \quad [\![h]\!](x)=\beta_1x+\beta_2$
  $\wedge\ \alpha_3(\beta_1-1)=\beta_2(2\alpha_1-1)$
- $[\![f]\!](x,y)=\alpha x^iy^i \quad [\![h]\!](x)=\beta x^j$
  $\wedge\ \beta^{2i-1}=\alpha^{j-1}$ ◇

Some special cases of the interpretations appropriate for the modified endomorphism are listed
A.2| below: (i) $[\![f]\!](x,y)=\alpha(x+y), [\![h]\!](x)=\beta x$ for arbitrary values of α and β, (ii) $[\![f]\!](x,y)=\alpha_1(x+y)+\alpha_3, [\![h]\!](x)=x$ for arbitrary values of α_1 and α_3, (iii) $[\![f]\!](x,y)=x+y+\alpha_3, [\![h]\!](x)=\beta_1x+\alpha_3(\beta_1-1)$ for arbitrary values of α_3 and β_1, (iv) $[\![f]\!](x,y)=x^iy^i, [\![h]\!](x)=x^j$ for arbitrary values of i and j.

We now investigate a *class* of equational theories.

An equation is called *variable-reducing* if the set of variables of one side is properly included in that of the other side.

For example, $x/x=1$ is variable-reducing. It is obvious that there are no interpretations of operators such that both sides of a variable-reducing equation are equivalent w.r.t. a polynomial ordering. A more general result in regard to arbitrary $\mathcal{E}$-termination is the fact that the set of variables of both sides of the axioms must be identical (see example 1.28 on page 13). We now consider axioms where one side is a special subterm of the other one.

$\boxed{5.59}$ **Definition (Subterm-Minimizing Equation)** *An equation $t \leftrightarrow_{\mathcal{E}} t|_u$ is called* sub-term-minimizing *if, and only if,*

- $u \neq \Lambda$ *and*
- $\exists v \in \mathcal{P}os(t), v \not\geq u\colon\ t|_v \in \mathcal{X}$ ◇

For example, $x \vee x \leftrightarrow_{\mathcal{E}} x$ is subterm-minimizing whereas $\neg\neg(x \vee x) = x \vee x$ and $(\neg x) \vee$ *false* $\leftrightarrow_{\mathcal{E}} \neg x$ are not.

$\boxed{5.60}$ **Lemma** *There are no interpretations of operators such that $s \sim_{\mathsf{POL}} t$ if $s \leftrightarrow_{\mathcal{E}} t$ is subterm-minimizing.* ◇

The more general assertion about subterms ($s \leftrightarrow_{\mathcal{E}} t$ where t is a subterm of s) is not true since the use of the identity as an interpretation of unary operators (e.g. $[\![\neg\neg(x \vee x)]\!](x) = [\![x \vee x]\!](x)$ if $[\![\neg]\!](x) = x$) or the interpretations $[\![f]\!](x_1,\dots,x_n) = x_1 \cdot \dots \cdot x_n$ and $[\![c]\!]() = 1$ (e.g. $[\![(\neg x) \vee \mathit{false}]\!](x) = [\![\neg x]\!](x)$ if $[\![\vee]\!](x,y) = xy$ and $[\![\mathit{false}]\!]() = 1$) would imply the equivalence of both terms s and t.

A range of applications for the two general conditions above (variable-reducing and subterm-minimizing) can be found in regard to various axiomatizations of the *if-then-else* construct. The usual specification of this operator consists of the two simple rules $\mathit{if}(\mathit{true},x,y) \rightarrow x, \mathit{if}(\mathit{false},x,y) \rightarrow y$. For example, there are no interpretations of the *if*-operator such that both sides of each of the following equations are equivalent w.r.t. the POL: $\mathit{if}(x,y,y) = y, \mathit{if}(x,x,\mathit{false}) = x, \mathit{if}(x,\mathit{true},x) = x, \mathit{if}(x,\mathit{if}(x,y,z),z) = \mathit{if}(x,y,z)$ and $\mathit{if}(x,y,\mathit{if}(x,y,z)) = \mathit{if}(x,y,z)$. These axioms demonstrate the difficulty of using theorems about *if-then-else* as an underlying theory. Common to all these axioms is the fact that one side contains one more *if*-operator than the other side. An if-then-else axiom will be more suitable for an underlying theory if the number of occurrences of the *if*-operator is equal for both sides. A.3

$\boxed{5.61}$ **Lemma** $\mathit{if}(x,\mathit{if}(y,z,u),\mathit{if}(y,v,w)) \sim_{\mathsf{POL}} \mathit{if}(y,\mathit{if}(x,z,v),\mathit{if}(x,u,w))$
if $[\![\mathit{if}]\!](x,y,z) = \sum\limits_i (\alpha_i x^i y + \beta_i x^i z)$ ◇

For example, $x(\alpha y + \beta z)$ is a special case of the interpretations of lemma 5.61. Note that there exists no separate interpretation for the *if*-operator causing the equivalence of both terms since x occurs once on the one side and twice on the other side of the equation.

Now, we examine the potential of *Boolean rings*, an algebraic structure often occurring in the literature, of being an underlying theory w.r.t. polynomial orderings. A ring is an *Abelian group* together with a multiplicative operation $\star$ such that the elements possess an additional *semigroup* structure (associativity law of $\star$). Moreover, the multiplication is distributive w.r.t. the addition. The Boolean ring is a special ring presented in the following definition.

$\boxed{5.62}$ **Definition (Boolean Ring)** *Let* $\mathcal{F} = \{+, i, \star, 1, 0\}$ *such that* $\mathcal{A}r(+) = \mathcal{A}r(\star) = 2$, $\mathcal{A}r(i) = 1$ *and* $\mathcal{A}r(1) = \mathcal{A}r(0) = 0$. *Then the* Boolean ring *is specified by the following equations:*

$$\left.\begin{array}{l}\left.\begin{array}{rcl} x+0 & \leftrightarrow_{\mathcal{E}} & x \\ (x+y)+z & \leftrightarrow_{\mathcal{E}} & x+(y+z) \\ x+y & \leftrightarrow_{\mathcal{E}} & y+x \\ i(0) & \leftrightarrow_{\mathcal{E}} & 0 \\ i(i(x)) & \leftrightarrow_{\mathcal{E}} & x \\ i(x+y) & \leftrightarrow_{\mathcal{E}} & i(y)+i(x) \\ x+i(x) & \leftrightarrow_{\mathcal{E}} & 0 \end{array}\right\}\begin{array}{c}\mathit{Abelian}\\ \mathit{Group}\end{array} \\ \begin{array}{rcl} x\star(y\star z) & \leftrightarrow_{\mathcal{E}} & (x\star y)\star z \\ x\star(y+z) & \leftrightarrow_{\mathcal{E}} & (x\star y)+(x\star z) \\ (x+y)\star z & \leftrightarrow_{\mathcal{E}} & (x\star z)+(y\star z) \end{array}\end{array}\right\}\mathit{Ring} \qquad \begin{array}{rcl} x\star y & \leftrightarrow_{\mathcal{E}} & y\star x \\ x+x & \leftrightarrow_{\mathcal{E}} & x \\ x\star x & \leftrightarrow_{\mathcal{E}} & x \\ x+(x\star y) & \leftrightarrow_{\mathcal{E}} & x \\ x\star(x+y) & \leftrightarrow_{\mathcal{E}} & x \\ x+(y\star z) & \leftrightarrow_{\mathcal{E}} & (x+y)\star(x+z) \\ (x\star y)+z & \leftrightarrow_{\mathcal{E}} & (x+z)\star(y+z) \\ i(x+y) & \leftrightarrow_{\mathcal{E}} & i(x)\star i(y) \\ i(x\star y) & \leftrightarrow_{\mathcal{E}} & i(x)+i(y) \\ x\star i(x) & \leftrightarrow_{\mathcal{E}} & 1 \end{array}$$
◇

In order to use (a part of) the Boolean ring as an underlying theory, the axioms that are either variable-reducing or subterm-minimizing can immediately be excluded (see lemma 5.60). These axioms are: The *idempotency* ($x + x \leftrightarrow_{\mathcal{E}} x$ and $x \star x \leftrightarrow_{\mathcal{E}} x$), the *adsorption* ($x + (x \star y) \leftrightarrow_{\mathcal{E}} x$ and $x \star (x + y) \leftrightarrow_{\mathcal{E}} x$) and the *complement* ($x + i(x) \leftrightarrow_{\mathcal{E}} 0$ and $x \star i(x) \leftrightarrow_{\mathcal{E}} 1$).

The axioms about the distributivity are the most difficult to handle. One can show that (see the proof of the following lemma) *all* distributivity axioms have to be part of the reducing system. Unfortunately, these equations cannot be oriented in the usual direction with any polynomial ordering. This leads to the following lemma.

$\boxed{5.63}$ **Lemma** *It will be impossible to use any part of the Boolean ring as an underlying theory (and the remaining part as reduction rules), if the termination is to be proved by a* POL. $\diamond$

Based on the assertion above, it is also hopeless to orient all axioms of the Boolean ring in the usual way (see definition 5.62) w.r.t. a POL.

5.5 Automatic Generation of 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 section is to present new insights into the (semi-) automatic generation of termination proofs using polynomial orderings. The following subsection deals with new guidelines for generating interpretations which can guarantee the termination of special classes of TRSs. In subsection 5.5.2, we describe an algorithm based on the method of complete description (see subsection 4.4.2) for finding polynomial interpretations for a given TRS provided that this system can be oriented using special polynomials.

5.5.1 Heuristics for Choosing Interpretations

Based on the experience with examples[41], 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.

$\boxed{5.64}$ **Heuristic (Arithmetic Specifications)** *In the case of* arithmetic specifications *it*
A.13 *has proved helpful to choose the complexity hierarchy of the interpretations such that it reflects*
A.17 *the complexity hierarchy of the corresponding operators.* $\diamond$
A.19
A.22
A.25 For example, $[\![+]\!](x,y) = xy + x$, $[\![\star]\!](x,y) = x^2y + xy^2$ and $[\![exp]\!](x,y) = (x+y)^3$.

$\boxed{5.65}$ **Heuristic (Groups)** *Let* $\star, i, e \in \mathcal{F}$ *and let* $\mathcal{E}$ *be a set of* group axioms[42]. *Appropriate and relatively simple interpretations for orienting (and completing)* $\mathcal{E}$ *are the following ones:* $[\![\star]\!](x,y) = 2xy + x$ *(or* $2xy + y$*)*, $[\![i]\!](x) = x^2$ *and* $[\![e]\!]() = 2$. $\diamond$

It is obvious that heuristic 5.65 can also be applied to systems representing an *extension* of the set $\mathcal{E}$ of group axioms.

The following lemmata provide tests for deciding whether a strictly separate interpretation for unary operators as well as a strictly mixed interpretation are necessary for a given rule to be ordered w.r.t. the POL.

---

[41] For an overview of the statistical results on a series of examples, see section 7.3.

[42] $\mathcal{E}$ includes $(x \star y) \star z \leftrightarrow_{\mathcal{E}} x \star (y \star z)$ and $i(x \star y) \leftrightarrow_{\mathcal{E}} i(y) \star i(x)$.

$\boxed{5.66}$ **Lemma** *Let $l \rightarrow r$ be a rule. No linear interpretation for the unary operator i exists if one of the following conditions holds:*

- $$\begin{array}{ll} \exists u \in \mathcal{P}os(l)\colon & l|_u = i(s) \\ & \wedge\ \forall h \in \mathcal{F}\colon\ i \neq h \in \mathcal{S}ym(l) \ \hookrightarrow\ [\![h]\!](x_1,\ldots,x_{\mathcal{A}r(h)}) \text{ is linear} \end{array}$$ A.28 A.29
  and
  $$\begin{array}{ll} \exists v \in \mathcal{P}os(r)\colon & r|_v = g(t_1,\ldots,t_{\mathcal{A}r(g)}) \\ & \wedge\ [\![g]\!](x_1,\ldots,x_{\mathcal{A}r(g)}) \text{ is not linear } \wedge\ \emptyset \neq \mathcal{V}ar(r|_v) \subseteq \mathcal{V}ar(l|_u) \end{array}$$
- $l = i(f(x,y))$, $r = f(i(y), i(x))$ *and* $[\![f]\!](x,y)$ *is not linear*[43] ◇

The main part of the first condition of lemma 5.66 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.

$\boxed{5.67}$ **Lemma** *Let $l \rightarrow r$ 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:*

- $$\begin{array}{ll} l = f(s_1,\ldots,s_m) \text{ and } \exists k\colon & \mathcal{V}ar(s_k) \cap \mathcal{V}ar(l[a]_k) = \emptyset \\ & \wedge\ \exists v \in \mathcal{P}os(r)\colon\ r|_v = f(t_1,\ldots,t_m)\ \wedge\ t_k \succsim_{\mathsf{EMB}} s_k \\ & \wedge\ \mathcal{V}ar(r[a]_{v.k}) \cap \mathcal{V}ar(s_k) \neq \emptyset \end{array}$$ A.13 A.19 A.28 A.36
  where 'a' is any constant
- $$\begin{array}{ll} \exists u \in \mathcal{P}os(l)\colon & l|_u = f(s_1,\ldots,s_m) \\ & \wedge\ \forall h \in \mathcal{F}\colon\ f \neq h \in \mathcal{S}ym(l) \ \hookrightarrow\ [\![h]\!](x_1,\ldots,x_{\mathcal{A}r(h)}) \text{ is linear} \end{array}$$ A.26
  and
  $$\begin{array}{ll} \exists v \in \mathcal{P}os(r)\colon & r|_v = g(t_1,\ldots,t_{\mathcal{A}r(g)}) \\ & \wedge\ [\![g]\!](x_1,\ldots,x_{\mathcal{A}r(g)}) \text{ is strictly mixed } \wedge\ \emptyset \neq \mathcal{V}ar(r|_v) \subseteq \mathcal{V}ar(l|_u) \end{array}$$
- $l = i(f(x,y)), r = f(i(y), i(x))$ *and f is associative*[44] ◇

The first condition of lemma 5.67 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. $\succsim_{\mathsf{EMB}}$) s_k. A graphical illustration of the technical conditions of the first part of lemma 5.67 is given in figure 5.7. The conditions of the second part of lemma 5.67 are slightly modified versions of the first part of lemma 5.66 (applied to operators with an arity greater than 1).

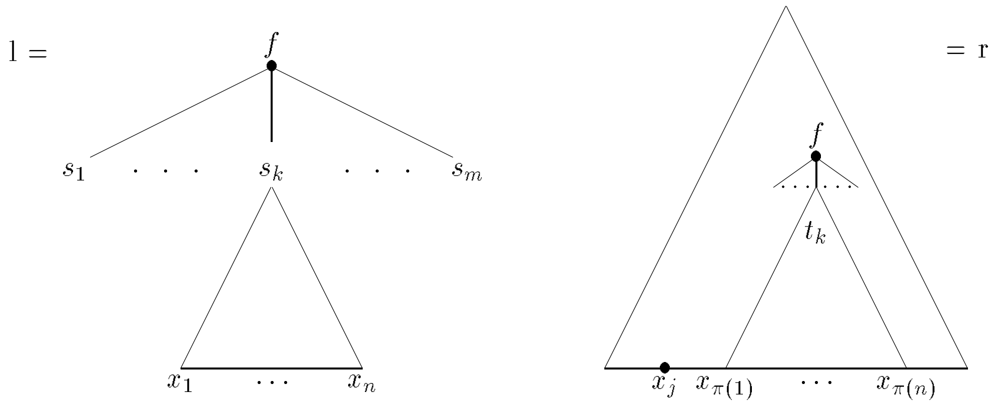
$\boxed{5.68}$ **Example** *(i) The termination proof (w.r.t. the* POL*) of the rule $f(i(x),y) \rightarrow g(x,y)$ requires a strictly separate interpretation for i if, for example, $[\![g]\!](x,y) = x^2 + y$ and $[\![f]\!](x,y) = x + y$, since the first condition of lemma 5.66 is valid.*

(ii) The termination proof (w.r.t. the POL*) of the distributivity axiom $f(x, g(y,z)) \rightarrow g(f(x,y), f(x,z))$ requires a strictly mixed interpretation for f since the first condition of lemma 5.67 is valid: $s_k = x$, $v = 1$, $t_k = x$, $t_k \succsim_{\mathsf{EMB}} s_k$ and $r|_{21} = x$. Furthermore, according to the second condition, the interpretation of f in $h(f(x, f(y,z))) \rightarrow h(f(x+y,z))$ must be strictly mixed if $[\![+]\!](x,y)$ is strictly mixed and $[\![h]\!](x)$ is linear.* ◇

---

[43] These conditions can be applied, for example, to Abelian groups.

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



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

Figure 5.7: Graphical representation of the first part of lemma 5.67

We have tested various examples including 399 unary operators and 467 operators where the arity is greater than 1. With the help of lemma 5.66, 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 5.67.

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[45].

$\boxed{5.69}$ **Heuristic (Distributivity and Associativity)** *Let* $+, \star \in \mathcal{F}$ *and let* $\mathcal{R}_D$ *be*

$$\begin{cases} x \star (y+z) & \rightarrow \quad (x \star y) + (x \star z) \\ (x+y) \star z & \rightarrow \quad (x \star z) + (y \star z) \end{cases}$$

A.4 A.5 A.35

- ► *The termination of* $\mathcal{R}_D$ *can be proved with the help of* $[\![+]\!](x,y) = x+y+1, [\![\star]\!](x,y) = xy$ *and* $\mu \geq 2$.
- ► *The TRS* $\mathcal{R}_D \cup \{x + (y+z) \rightarrow (x+y)+z\}$ *terminates if* $[\![+]\!](x,y) = x + 2y + 1, [\![\star]\!](x,y) = xy$ *and* $\mu \geq 2$.
- ► *The TRS* $\mathcal{R}_D \cup \{(x \star y) \star z \rightarrow x \star (y \star z)\}$ *is ordered w.r.t. the interpretations* $[\![+]\!](x,y) = x+y+2, [\![\star]\!](x,y) = xy + x$ *and* $\mu \geq 3$.
- ► *In order to achieve the termination of* $\mathcal{R}_D \cup \{(x+y)+z \rightarrow x+(y+z), (x \star y) \star z \rightarrow x \star (y \star z)\}$, *we use the interpretations* $[\![+]\!](x,y) = 2x+y+4, [\![\star]\!](x,y) = xy+x$ *and* $\mu \geq 3$. ◇

In [BenCherifa, 1986], other heuristics for determining interpretations for special rewriting rules are given. The set of rules includes (i) $f(x, f(y,z)) \rightarrow f(f(x,y),z)$, *(ii)* $f(h(t_1), t_2) \rightarrow$

---

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

$h(f(t_1,t_2))$, (iii) $f(s_1,\ldots,s_n) \rightarrow t$ where $f \notin \mathcal{S}ym(t)$, (iv) $h(f(t_1,\ldots,t_n)) \rightarrow f(h(t_1),\ldots,h(t_n))$. Furthermore, BenCherifa introduced a heuristic for expediently utilizing the 'Cartesian product' of polynomial orderings (see remark 5.11 on page 99).

5.5.2 A Method for Simple-Mixed Polynomials

We have implemented polynomial orderings (based on [BenCherifa and Lescanne, 1987]) and conducted a series of experiments considering 320 TRSs with more than 1700 rules *occurring in the literature* to gain more insight into the choice of interpretations[46]. According to these experiments, 96% of the TRSs ordered w.r.t. polynomial orderings are also ordered w.r.t. polynomial orderings using *simple-mixed* interpretations[47], only. In this subsection, we develop a procedure that attempts to automatically generate simple-mixed interpretations of operators (as the basis for the SPOL) in order to prove that a given TRS terminates.

Based on the algorithm of [Martin, 1987] (and [Dick *et al.*, 1990], see also [Lankford, 1979]) for generating an appropriate weight function (i.e. *strongly linear* interpretations) for the KBO (see section 4.4), 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 the method of complete description 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 method of complete description, we will transform more general inequalities to linear ones by

- ▶ *approximating* each side to exactly one product (see lemma 5.78) and then
- ▶ applying a *logarithmic function* to the resulting products.

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

A Motivating Example

We will prove the termination of the following TRS.

5.70 **Example ([Middeldorp, 1988])** *Consider the TRS $\mathcal{R}$* |A.34

$$\left\{\begin{array}{rrcl} 1: & f(x,y) & \rightarrow & x \\ 2: & g(a) & \rightarrow & h(a,b,a) \\ 3: & i(x) & \rightarrow & f(x,x) \\ 4: & h(x,x,y) & \rightarrow & g(x) \end{array}\right.$$

Note that the second rule and the fourth rule are the problematic ones since g and h are mutually recursive. Neither path nor decomposition orderings are able to prove the termination of $\mathcal{R}$. ◇

---

[46] The interpretations have been generated by the user.

[47] Simple-mixed interpretations have been introduced in definition 1.43 on page 22.

(i)
$$\begin{aligned}
\llbracket a\rrbracket() &= \alpha_0\\
\llbracket b\rrbracket() &= \beta_0\\
\llbracket f\rrbracket(x,y) &= \chi_{11}xy+\chi_{10}x+\chi_{01}y+\chi_{00}\\
\llbracket g\rrbracket(x) &= \delta_2x^2+\delta_1x+\delta_0\\
\llbracket h\rrbracket(x,y,z) &= \epsilon_{111}xyz+\epsilon_{110}xy+\epsilon_{101}xz+\epsilon_{011}yz+\epsilon_{100}x+\epsilon_{010}y+\epsilon_{001}z+\epsilon_{000}\\
\llbracket i\rrbracket(x) &= \phi_2x^2+\phi_1x+\phi_0
\end{aligned}$$

(ii)
$$\begin{aligned}
\llbracket a\rrbracket() &= \alpha_0\\
\llbracket b\rrbracket() &= \beta_0\\
\llbracket f\rrbracket(x,y) &= \chi_{11}xy+\chi_{10}x+\chi_{01}y+\chi_{00}\\
\llbracket g\rrbracket(x) &= \delta_2x^2\\
\llbracket h\rrbracket(x,y,z) &= \epsilon_{111}xyz+\epsilon_{110}xy+\epsilon_{101}xz+\epsilon_{011}yz+\epsilon_{100}x+\epsilon_{010}y+\epsilon_{001}z\\
\llbracket i\rrbracket(x) &= \phi_1x+\phi_0
\end{aligned}$$

(iii) $\mathcal{R}'=\mathcal{R}\setminus\{f(x,y)\to x\}$

(iv) $\beta_0=1$ ($\rightsquigarrow \llbracket b\rrbracket()=1$) and
$\chi_{11}=0,\chi_{10}=1,\chi_{01}=1,\chi_{00}=0$ ($\rightsquigarrow \llbracket f\rrbracket(x,y)=x+y$)
whereas the other coefficients remain unchanged.

(v) $\llbracket g(a)\rrbracket = \alpha_0^2\delta_2$
and $\llbracket h(a,b,a)\rrbracket = \alpha_0^2\epsilon_{111}+\alpha_0\epsilon_{110}+\alpha_0^2\epsilon_{101}+\alpha_0\epsilon_{011}+\alpha_0\epsilon_{100}+\epsilon_{010}+\alpha_0\epsilon_{001}$,
$\llbracket i(x)\rrbracket = \phi_1x+\phi_0$
and $\llbracket f(x,x)\rrbracket = 2x$,
$\llbracket h(x,x,y)\rrbracket = \epsilon_{111}x^2y+(\epsilon_{101}+\epsilon_{011})xy+\epsilon_{110}x^2+(\epsilon_{100}+\epsilon_{010})x+\epsilon_{001}y$
and $\llbracket g(x)\rrbracket = \delta_2x^2$.

$$\begin{aligned}
&I_1=\{\alpha_0^2\delta_2\geq\alpha_0^2\epsilon_{111}+\alpha_0\epsilon_{110}+\alpha_0^2\epsilon_{101}+\alpha_0\epsilon_{011}+\alpha_0\epsilon_{100}+\epsilon_{010}+\alpha_0\epsilon_{001}\}\\
&I_2=\{\phi_1\geq 2\,,\ \phi_0\geq 0\}\\
&I_3=\{\epsilon_{111}\geq 0\,,\ \epsilon_{101}+\epsilon_{011}\geq 0\,,\ \epsilon_{110}\geq\delta_2\,,\ \epsilon_{100}+\epsilon_{010}\geq 0\,,\ \epsilon_{001}\geq 0\}
\end{aligned}$$

(vi) $I_2=\{\phi_1\geq 2\}$, $I_3=\{\epsilon_{110}\geq\delta_2\}$ whereas I_1 remains unchanged.

(vii)
$$\begin{aligned}
&I_1=\{\alpha_0\delta_2\geq\epsilon_{110}+\epsilon_{001}\}\\
&I_2=\{\phi_1\geq 2\}\\
&I_3=\{\epsilon_{110}\geq\delta_2\}
\end{aligned}$$

(viii)
$$\begin{array}{lcl}
I_1=\{\alpha_0\delta_2\geq 2\epsilon_{110}\epsilon_{001}\} & & I_1=\{\alpha_0^*+\delta_2^*\geq 1+\epsilon_{110}^*+\epsilon_{001}^*\}\\
I_2=\{\phi_1\geq 2\} & \rightsquigarrow & I_2=\{\phi_1^*\geq 1\}\\
I_3=\{\epsilon_{110}\geq\delta_2\} & & I_3=\{\epsilon_{110}^*\geq\delta_2^*\}
\end{array}$$

(ix)
$$\begin{aligned}
I=\{\ \ &\alpha_0^*+\delta_2^*\geq 1+\epsilon_{110}^*+\epsilon_{001}^*\,,\\
&\phi_1^*\geq 1\,,\\
&\epsilon_{110}^*\geq\delta_2^*\ \ \}
\end{aligned}$$

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

Figure 5.8: Generating interpretations for example 5.70

(i) In order to find a polynomial ordering for proving the termination of $\mathcal{R}$, we have to define interpretations with *variable* coefficients[48] for the operators. Thus, we introduce *complete simple-mixed* polynomials $\llbracket f\rrbracket$ with *variable* coefficients for each $f\in\mathcal{F}$ (see (i) of figure 5.8).

---

[48] i.e. unknown coefficients

(ii) In order to reduce the number of *variable coefficients* (which increases the power of our algorithm, see lemma 5.78 and remark 5.79), we insert a simplification process. The basic principle underlying this process is to assign zero to the coefficient of the constant monomial in *some* (possibly none) interpretations $[\![f]\!]$ if $\mathcal{A}r(f) > 0$. Continuing the example with $\delta_0 = 0$, $\epsilon_{000} = 0$ and $\delta_1 = 0$, $\phi_2 = 0$[49], we obtain the interpretations given in (ii) of figure 5.8.

(iii) We now remove all rules whose right-hand side is smaller than (w.r.t. the SPOL) the corresponding left-hand side w.r.t. any interpretation. Thus, the initial TRS $\mathcal{R}$ will be transformed into $\mathcal{R}'$ given in (iii) of figure 5.8 since $f(x,y) \succ_{\mathsf{SPOL}} x$ for all $[\![.]\!]$.

In the remaining part of this subsection, $\mathcal{R}'$ is used instead of $\mathcal{R}$.

(iv) In order to further reduce the number of *variable coefficients*, we set the interpretations of so-called *uninduced* operators to minimal values. An *uninduced operator* does not occur on the left-hand side of the rules of $\mathcal{R}'$. The interpretation of such an operator f will be a strongly linear polynomial without the constant monomial (i.e. $[\![f]\!](x_1,\ldots,x_n) = \sum_{i=1}^{n} x_i$) because it is the 'smallest' polynomial satisfying the conditions of definition 1.47 on page 24. This operation, called Ψ, sets the interpretation of an uninduced constant to the minimal value μ.

To continue with our example, we set μ to 1

(see (iv) of figure 5.8).

(v) Now, the rules of $\mathcal{R}'$ must be transformed into a set of inequalities by (1) applying the variable interpretations and (2) splitting the resulting polynomials into inequalities w.r.t. common monomials. Part (2) will be performed using the so-called *Split*-operation. This splitting process transforms one polynomial (i.e. $[\![l]\!] - [\![r]\!]$ for a rule $l \to r$) into a set of inequalities in the following way: We collect the positive and the negative coefficients w.r.t. common tuples of exponents. Then for each tuple of exponents, an inequality will be generated with the positive coefficients on the left-hand side and the negative ones on the right-hand side (see (v) of figure 5.8).

(vi) At this stage, another simplification process removes redundant information by removing common addends, common factors, obvious inequalities ($x \geq 0$) and so on (see (vi) of figure 5.8).

(vii) A modified Ψ-operation, denoted by Ψ_I, will be applied to the interpretations w.r.t. the actual set of inequalities, alternating with applications of simplification rules as described in (vi). The operation Ψ_I sets *uninduced variables* (i.e. variable coefficients which do not occur on the left-hand sides of the *inequalities*[50]) to minimal values. The application of Ψ_I leads to $\epsilon_{111} = 0$, $\epsilon_{101} = 0$, $\epsilon_{011} = 0$, $\epsilon_{100} = 0$, $\epsilon_{010} = 0$ and

---

[49] According to the definition of simple-mixed polynomials, a unary operator g can either be interpreted as $\delta_2 x^2 + \delta_0$ or as $\delta_1 x + \delta_0$. In this example, $\delta_2 = 0$ does not lead to a successful result. Thus, we backtrack and assign zero to δ_1 (see also ϕ_2).

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

$I_1 = \{\alpha_0^2\delta_2 \geq \alpha_0\epsilon_{110} + \alpha_0\epsilon_{001}\}$[51]. The application of a simplification rule of (vi) simplifies I_1 (by dividing both sides by α_0). See (vii) of figure 5.8.

(viii) In order to apply the method of complete description, the inequalities of each I_i must be linear. This condition will be achieved by (1) transforming the inequalities into inequalities, each side of which represents only one product, by (2) applying the binary logarithmic function lg to these products and by (3) renaming the variables of the resulting inequalities, i.e. each $lg(\alpha_{r_1 \ldots r_n})$ will be transformed into $\alpha^*_{r_1 \ldots r_n}$. The approximation of each side of an inequality to exactly one product (i.e. part (1)) is performed according to lemma 5.78 (see (viii) of figure 5.8).

(ix) Note that each set I_i must contain at least one strict inequality ($p_i > q_i$) in order to guarantee that the corresponding rule of $\mathcal{R}'$ can be oriented. This will be achieved by the procedure described in subsection 4.4.2 on page 88. All sets of inequalities will be *collected* in one set I which represents the final set of inequalities, that can serve as the input for the method of complete description (see (ix) of figure 5.8).

(x) The assignments in (x) of figure 5.8 represent a solution of the set I of inequalities. 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 $\mathcal{R}$ with the help of the SPOL.

The Algorithm

Here we deal with the formal description of the algorithm motivated previously. It is divided into ten paragraphs according to both the previous paragraph and the steps of algorithm 5.86 given in figure 5.9 on page 138.

(i) ★Defining the interpretations★

For each operator in $\mathcal{F}$ a complete simple-mixed polynomial with variable coefficients will be introduced. Note that one has to ensure that the interpretation of a unary operator has one of the forms $\alpha_2 x^2 + \alpha_0$ or $\alpha_1 x + \alpha_0$ by setting $\alpha_1 = 0$ or $\alpha_2 = 0$.

$\boxed{5.71}$ **Definition (Operation Υ)** *Let $\mathcal{F} = \bigcup\{f_i\}$ be a set of operators and assume that $\mathcal{A}r(f_i) = n_i$. $\Upsilon(\mathcal{F})$ is defined to be the union of all $[\![f_i]\!](x_1,\ldots,x_{n_i}) = p \in \mathbb{R}_1[x_1,\ldots,x_{n_i}]$ where p is the complete simple-mixed polynomial for the variables $x_1,\ldots,x_{n_i}$.* ◇

(ii) ★Removing special constant monomials★

$\boxed{5.72}$ **Definition (Operation ν)** *Let $\mathcal{F}$ and $\mathcal{F}' \subseteq \mathcal{F}$ be two sets of operators and $[\![.]\!]$ be a set of polynomial interpretations for operators in $\mathcal{F}$. Then the* operation ν: $[\![.]\!] \mapsto [\![.]\!]$ *is defined as*

$$\begin{aligned}\nu([\![.]\!]) \;=\; \{[\![f]\!](x_1,\ldots,x_n) - \alpha_{0\ldots0} \mid\; & f \in \mathcal{F}', \mathcal{A}r(f) = n > 0,\\ & [\![f]\!](x_1,\ldots,x_n) \;=\; \textstyle\sum \alpha_{r_1\ldots r_n} \cdot x_1^{r_1} \cdot \ldots \cdot x_n^{r_n}\}\end{aligned}$$ ◇

---

[51] Note that $\epsilon_{001} \neq 0$ must hold, otherwise $[\![h]\!](x,y,z)$ would not be a polynomial depending on three distinct variables. One should try to generate interpretations such that the number of non-zero coefficients of monomials containing at least two different variables is minimal. For example, $xy + y + z$ has a higher priority than $xy + yz$.

For example, the operation ν sets α_0 to zero in $[\![f]\!](x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$. Note that the success of our algorithm does in many cases necessitate this operation.

(iii) ★Using the SPOL for simplifying the rules★

Since the SPOL is a simplification ordering, we check the rules of $\mathcal{R}$ for being simplifying, i.e. $l \succ_{\mathsf{SPOL}} r$ must hold independent of the used interpretations. Thus, we remove all rules whose right-hand side is homeomorphically embedded in the corresponding left-hand side.

(iv) ★Uninduced operators★

The interpretations of *uninduced* operators – i.e. operators not occurring on the left-hand side of any rule – will be set to minimal values.

$\boxed{5.73}$ **Definition (Operation Ψ)** *Let $\mathcal{F} = \bigcup\{f_i\}$ be a set of operators, $\mathcal{R}' = \bigcup\{l_i \to r_i\}$ 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 *(computing an interpretation) is defined in the following way:*

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

$$[\![f_i]\!]'(x_1,\ldots,x_{\mathcal{A}r(f_i)}) = \begin{cases} \mu & \text{if } \mathcal{A}r(f_i) = 0 \ \wedge \ f_i \text{ is uninduced} \\ \sum\limits_{j=1}^{\mathcal{A}r(f_i)} x_j & \text{if } \mathcal{A}r(f_i) > 0 \ \wedge \ f_i \text{ is uninduced} \\ [\![f_i]\!](x_1,\ldots,x_{\mathcal{A}r(f_i)}) & \text{otherwise} \end{cases} \qquad \diamond$$

Note that the interpretation of an operator which has completely been removed, according to the relation $\succ_{\mathsf{EMB}}$, will be set to a minimal value using the operation Ψ. More formally, $[\![f]\!](x_1,\ldots,x_n)$ will be minimal if the operator f is contained neither in a left-hand side nor in a right-hand side of any rule in $\mathcal{R}'$. The operation Ψ sets the interpretation of an uninduced constant to the minimal value μ. It is either possible to use μ as a variable or to instantiate μ with a specific value[52]. Whenever μ is treated as a variable, we have to guarantee that μ really represents the minimum. Therefore, the inequality $[\![c]\!]() \geq \mu$ must be added for each $c \in \mathcal{F}$ for which $\mathcal{A}r(c) = 0$ holds.

(v) ★Splitting the rules into inequalities★

$\boxed{5.74}$ **Definition ($\mathcal{S}plit$-Operation)** *The operation $\mathcal{S}plit$ transforms a polynomial p into a set of inequalities. Let $p = \sum(\alpha_{i_1\ldots i_n} - \beta_{i_1\ldots i_n}) \cdot x_1^{i_1} \cdot\ldots\cdot x_n^{i_n}$ such that $\alpha_{i_1\ldots i_n} \geq 0$ and $\beta_{i_1\ldots i_n} \geq 0$. Then*

$$\mathcal{S}plit \ = \ \{\alpha_{i_1\ldots i_n} \geq \beta_{i_1\ldots i_n} \mid i_1\ldots i_n \in \mathbb{N}^n\}. \qquad \diamond$$

---

[52] Note that the instantiation of μ with a specific value should be done at the beginning of the algorithm.

To split the rules into inequalities we first generate the polynomial $p_i = [\![l_i]\!] - [\![r_i]\!]$ and then compute the set $I_i = \mathcal{S}plit(p_i)$ for each $l_i \to r_i \in \mathcal{R}'$. The $\mathcal{S}plit$-operation provides a sufficient criterion for l_i being greater (w.r.t. the (S)POL) than r_i.

(vi) ★Applying arithmetic laws★

A *simplification process* removes redundant information from the sets of inequalities (see the following definition 5.75). 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 $x' \geq y'$ can be generated if $x \geq y$ and condition A hold. If A is omitted, it will be treated as 'true'.

$\boxed{5.75}$ **Definition (Arithmetic Laws $\mathcal{AL}$)** *Let x, y, z be arbitrary non-negative 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 (the inference rules apply to AC-variants of the addition $+$ and the multiplication $\cdot$):*

1) Removing common addends:

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

2) Removing common factors:

$$\frac{xy \geq xz}{y \geq z} \quad \textit{if } x \neq 0$$

3) Removing obvious inequalities:

$$\frac{x \geq 0}{}$$ [53]

4) Splitting trivial products:

$$\frac{\left(\prod_{i=1}^{n} \alpha_i\right) \leq 1}{\forall i \in [1,n]:\ \alpha_i := 1}$$ [54]

5) Dealing with seemingly inconsistent inequalities:

$$\frac{\alpha \leq 0}{\alpha := 0} \quad \begin{array}{l} \textit{if the corresponding interpretation } [\![f]\!](x_1, \ldots, x_{\mathcal{A}r(f)}) \textit{ to } \alpha \textit{ is of arity } \mathcal{A}r(f), \\ \textit{otherwise stop with failure} \end{array}$$ ◇

(vii) ★Uninduced variables★

The interpretations of *uninduced* variables – i.e. variables which do not occur on the left-hand side of any inequality – will be set to minimal values.

---

[53] Sometimes, an inequality $x \geq 0$ can be helpful if $x > 0$ (see the method of complete description).

[54] It is also possible to assign $\alpha_j := 0$ for at least one particular j instead of $\alpha_i := 1$ for all i.

5.76 **Definition (Operation Ψ_I)** *Let $\mathcal{F}$ be a set of operators, I a set of inequalities and $[\![.]\!]$ a set of polynomial interpretations for the operators in $\mathcal{F}$. Then the* operation Ψ_I *(computing a set of assignments) is defined in the following way:*

$$\Psi_I([\![.]\!], I) \;=\; \{\alpha_{r_1\ldots r_n} = \beta \mid [\![f]\!](x_1,\ldots,x_n) = \sum \alpha_{r_1\ldots r_n}\cdot x_1^{r_1}\cdot\ldots\cdot x_n^{r_n}, f\in\mathcal{F}, \mathcal{A}r(f)=n\}$$

such that β is determined as follows:

```
if      $\alpha_{r_1\ldots r_n}$ is uninduced and has no specific value
then if    $\mathcal{A}r(f) > 0$
     then $\beta := 0$
          if    $[\![f]\!](x_1,\ldots,x_n)$ is not a (correct) polynomial interpretation
                (after this assignment)
          then $\beta := 1$
          endif
     else $\beta := \mu$
     endif
else $\beta := \alpha_{r_1\ldots r_n}$
endif
```

◇

For example, $[\![f]\!](x,y) = \alpha_{11}xy + \alpha_{10}x + \alpha_{01}y + \alpha_{00}$, α_{10} is uninduced and has no specific value: if $\alpha_{11} \neq 0$ then $\alpha_{10} := 0$ else $\alpha_{10} := 1$. Note that the operation Ψ_I is non-deterministic (because of the second then-case of definition 5.76). Moreover, Ψ_I sets the interpretation of an uninduced constant to μ.

(viii) ★Creating linear inequalities★

In order to apply the method of complete description, the inequalities of each I_i must be linear. This condition will be achieved by (1) transforming the inequalities into inequalities each side of which represents only one product and by (2) applying the binary logarithm 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 remark 5.38 on page 115). The following lemma concludes the theoretical framework for our approximation.

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

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

◇

The combination of the arithmetic-mean-geometric-mean inequality and lemma 5.77 leads to a method of transforming a general inequality into an inequality of only two products.

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

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

◇

For example, the problem of showing $2ab + 3c \geq a + 2b + bc$ is reduced to that of showing $2^2 \cdot (2ab) \cdot (3c) \geq a^2 \cdot (2b)^2 \cdot (bc)^2$, $a \geq 2$, $2b \geq 2$ and $bc \geq 2$ using the first transformation of lemma 5.78. Whenever $n = m = 1$ holds, the second alternative of lemma 5.78 should be applied since the (input) sums remain unchanged.

$\boxed{5.79}$ Remark

The power of our algorithm strongly depends on this transformation and thus on lemma 5.78. *The better the approximations are the more powerful the algorithm will be.* Note that the more variable coefficients exist the worse the approximation with lemma 5.78 will be, because n and m in lemma 5.78 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.

With lemma 5.78, an inequality $\sum a_i \geq \sum b_i$ can be transformed into an inequality $\prod c_i \geq \prod d_i$. A linear inequality will be generated from $\prod c_i \geq \prod d_i$ in a stage as follows: $\prod c_i \geq \prod d_i$ if, and only if, $\sum lg(c_i) \geq \sum lg(d_i)$ where lg denotes the *binary* logarithm. We use lg since it has the smallest base. Therefore, we apply lg to each side of the inequalities and rename the variables of the resulting inequalities, i.e. each $lg(\alpha_{r_1 \ldots r_n})$ will be transformed into $\alpha^*_{r_1 \ldots r_n}$.

$\boxed{5.80}$ Notation *The transformation according to lemma 5.78 of a general inequality into an inequality where both sides contain only one product will be denoted by $\mathcal{R}ed$: $\mathcal{R}ed(\sum a_i \geq \sum b_i)$. The transformation of an inequality of two products into a linear inequality (by using lg) is denoted by $\mathcal{L}in$: $\mathcal{L}in(\prod a_i \geq \prod b_i)$. The operation $\mathcal{R}en$ renames the variables lg is applied to, i.e. $lg(\alpha_{r_1 \ldots r_n})$ will be replaced with $\alpha^*_{r_1 \ldots r_n}$.* ◇

(ix) ★Collecting all inequalities★

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 $\mathcal{R}$ is ordered w.r.t. the resulting SPOL. The algorithm of [Martin, 1987] generates just one inequality for each rule whereas our algorithm produces a *set* of inequalities for each rule. Thus, Martin's approach computes a solution (with the help of the method of complete description) for the set of all inequalities provided that *each* inequality is a proper one (and if not, the current precedence must be extended). However, our technique for the SPOL requires the input of the method of complete description to be a set of sets of inequalities. Each of the inner sets must contain at least one proper inequality.

☞ This requirement will implicitly be guaranteed when collecting the sets I_i in just one set I (see subsection 4.4.2).

(x) ★Applying the method of complete description★

The last part of our algorithm concerns the computation of a solution of the generated set of inequalities. As noted before, we use the method of complete description for this.

☞ The input for the method of complete description (i.e. the set I of inequalities) also includes inequalities of the form $\alpha^*_{r_1 \ldots r_n} \geq 0$ (for each variable $\alpha^*_{r_1 \ldots r_n}$) in order to ensure the coefficients to be non-negative (for simplicity, these inequalities are not specified here).

Assume now that the method of complete description provides a solution. For deducing the resulting interpretations of the operators from this solution, we have to undo the renaming of the coefficients (as used with lg) by replacing them by their corresponding initial names, i.e. $\alpha_{r_1\ldots r_n} = 2^{\alpha^*_{r_1\ldots r_n}}$.

Before presenting the whole algorithm, a modification of the *Split*-operation is pointed out which sometimes proves more profitable. It is a slightly modified version of the approach contained in subsection 5.3.3 (see pages 110–114)[55].

$\boxed{5.81}$ **Definition (Operation *Split*^{M})** *The* operation *Split*^{M} *transforms a polynomial* p *into a set of inequalities. Let*

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

and let $\gamma_{l_1\ldots l_{n_{k_1\ldots k_n}}}$ *be new variables over* $\mathbb{R}$.

$$\begin{array}{rl} \mathit{Split}^M(p) & := J \cup \{ \sum\limits_{k_j \geq i_j} \mu^{\Sigma(k_j - i_j)} \cdot \gamma_{k_1\ldots k_{n_{i_1\ldots i_n}}} \geq \beta_{i_1\ldots i_n} \mid \beta_{i_1\ldots i_n} > 0, \alpha_{k_1\ldots k_n} > 0\} \\ \textit{such that} \quad J & := \{\alpha_{i_1\ldots i_n} \geq \sum\limits_{k_j \leq i_j} \gamma_{i_1\ldots i_{n_{k_1\ldots k_n}}} \mid \alpha_{i_1\ldots i_n} > 0\,,\, \beta_{k_1\ldots k_n} > 0\} \\ & \textit{If } J = J' \cup \{\alpha_{i_1\ldots i_n} \geq \gamma_{i_1\ldots i_{n_{k_1\ldots k_n}}}\} \\ & \textit{then let } J := J' \cup \{\gamma_{i_1\ldots i_{n_{k_1\ldots k_n}}} := \alpha_{i_1\ldots i_n}\}. \end{array} \qquad \diamond$$

The operation *Split*$^M(p)$[56] generates a set of linear inequalities if the coefficients of p are given (see subsection 5.3.3).

$\boxed{5.82}$ **Example** *The operation Split*^{M} *will be illustrated by applying it to the TRS*

$$\begin{cases} 1: & f(x, h(y)) & \to & g(f(x, y), x) \\ 2: & i(x) & \to & f(x, x) \\ 3: & k(i(x), y) & \to & k(k(x, y), 0) \end{cases}$$

(i)

$$\begin{array}{rcll} [\![f]\!](x, y) & = & \alpha_{11}xy + \alpha_{10}x + \alpha_{01}y + \alpha_{00} & \\ [\![g]\!](x, y) & = & \beta_{11}xy + \beta_{10}x + \beta_{01}y + \beta_{00} & \\ [\![h]\!](x) & = & \gamma_1 x + \gamma_0 & (\gamma_2 = 0) \\ [\![i]\!](x) & = & \delta_2 x^2 + \delta_0 & (\delta_1 = 0) \\ [\![k]\!](x, y) & = & \epsilon_{11}xy + \epsilon_{10}x + \epsilon_{01}y + \epsilon_{00} & \\ [\![0]\!]() & = & 2 & (\mu = 2) \end{array}$$

(ii) $\alpha_{00} = 0$, $\epsilon_{00} = 0$

(iii) –

(iv) $[\![g]\!](x, y) = x + y$

(v) The set of inequalities for the first rule contains

$$\begin{array}{rcl} \alpha_{11}\gamma_1 & \geq & A_{11_{11}} + A_{11_{10}} + A_{11_{01}} \\ \alpha_{01}\gamma_0 & \geq & 0 \\ A_{11_{11}} & \geq & \alpha_{11} \\ \alpha_{01}\gamma_1 + 2A_{11_{10}} & \geq & \alpha_{01} \\ \alpha_{11}\gamma_0 + 2A_{11_{01}} + \alpha_{10} & \geq & \alpha_{10} + 1 \end{array}$$

---

[55] $\alpha_{i_1\ldots i_n} \cdot \beta_{i_1\ldots i_n} = 0$ cannot be required since $\alpha_{i_1\ldots i_n}$ and $\beta_{i_1\ldots i_n}$ are variables.

[56] *Split*^{M} stands for ‘modified’ *Split*.

The set of inequalities for the second rule contains

$$\begin{array}{rcl} \delta_2 & \geq & B_{2_2} + B_{2_1} \\ \delta_0 & \geq & 0 \\ B_{2_2} & \geq & \alpha_{11} \\ 2B_{2_1} & \geq & \alpha_{01} + \alpha_{10} \end{array}$$

The set of inequalities for the third rule contains

$$\begin{array}{rcl} \epsilon_{11}\delta_2 & \geq & C_{21_{11}} + C_{21_{10}} + C_{21_{01}} + C_{21_{00}} \\ \epsilon_{10}\delta_2 & \geq & C_{20_{10}} + C_{20_{00}} \\ \epsilon_{11}\delta_0 + \epsilon_{01} & \geq & C_{01_{01}} + C_{01_{00}} \\ 2C_{21_{11}} & \geq & \epsilon_{10}\epsilon_{11} + 2\epsilon_{11}^2 \\ 4C_{21_{10}} + 2C_{20_{10}} & \geq & \epsilon_{10}^2 + 2\epsilon_{10}\epsilon_{11} \\ 4C_{21_{01}} + C_{01_{01}} & \geq & \epsilon_{01}\epsilon_{10} + 2\epsilon_{01}\epsilon_{11} \\ \epsilon_{10}\delta_0 + 8C_{21_{00}} + 4C_{20_{00}} + 2C_{01_{00}} & \geq & 2\epsilon_{01} \end{array}$$

(vi) - (x) *This example illustrates the use of the operation* $\mathcal{S}plit^M$. *Thus, a detailed presentation of the remaining parts is omitted. The (implemented version of our) algorithm generates the following interpretations:*

$$\begin{array}{rcl} [\![f]\!](x,y) & = & xy + 2.598x + 2502.619y \\ [\![g]\!](x,y) & = & x + y \\ [\![h]\!](x) & = & 43.324x + 1 \\ [\![i]\!](x) & = & 26007.979x^2 + 1 \\ [\![k]\!](x,y) & = & xy + 1.26x + 10.079y \\ [\![0]\!]() & = & 2 \end{array}$$

◇

5.83 Remark

The power of $\mathcal{S}plit$ and that of $\mathcal{S}plit^M$ are incomparable. An interpretation for example 5.70 can be found with the help of $\mathcal{S}plit$ but not with $\mathcal{S}plit^M$. $\mathcal{S}plit^M$ computes an SPOL for the TRS given in example 5.82 whereas $\mathcal{S}plit$ fails. However, for the most part of the examples tested where at least one of the operations is successful, the $\mathcal{S}plit$-operation succeeds. Sometimes, $\mathcal{S}plit^M$ generates simpler interpretations than $\mathcal{S}plit$. The main problem concerning the $\mathcal{S}plit^M$-operation is that a lot of new variables are introduced affecting prejudicially lemma 5.78 (see remark 5.79 on page 134).

The operation $\mathcal{S}plit^M$ can be used instead of $\mathcal{S}plit$. This means, that our algorithm has this operation as an option. Summarizing, algorithm 5.86 which is given in figure 5.9, is parameterized by the following five features:

- The specification of the mode for splitting the rules into inequalities ($\mathcal{S}plit$ or $\mathcal{S}plit^M$).
- The specification of the method of approximating general inequalities by linear ones (choosing (i) or (ii) of lemma 5.78).
- The specification of operators with no constant monomials occurring in their interpretations (i.e. $[\![f]\!](x_1,\dots,x_n) = \sum \alpha_{i_1\dots i_n} x_1^{i_1} \cdot\ldots\cdot x_n^{i_n}$ such that $\alpha_{0\dots0} = 0$) by applying the operation ν to a subset $\mathcal{F}'$ of $\mathcal{F}$.
- The specification of unary operators with square interpretations (i.e. determine whether either $\alpha_1 = 0$ or $\alpha_2 = 0$ for $[\![f]\!](x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$).
- The specification of μ as either a variable or a constant.

We have implemented algorithm 5.86 (see also section 7.2 on page 169). The implementation does not require any user interactions since all possible values of the parameters described will be set one after the other (including backtracking) until the algorithm stops.

5.84 **Theorem** *Algorithm 5.86 always terminates. If it does not fail, $\mathcal{R}$ is ordered w.r.t. the* SPOL *(based on simple-mixed interpretations) generated by the algorithm.* ◇

5.85 **Remark**

Algorithm 5.86 can be applied to TRSs with underlying AC-theories to create adequate interpretations for the AC-POL. The special condition, the coefficients must satisfy, will be encoded as a constraint. Note that interpretations for AC-operators are simple-mixed polynomials.

For generating an appropriate POLS (or IPOL), we make use of an improved version of algorithm 5.86. In case of the IPOL, the precedence is created as in the method of [Martin, 1987] ([Dick *et al.*, 1990]). Subsection 4.4.3, dealing with the automatic generation of the KBOS, also provides a technique for generating a precedence and a status function for the POLS. However, step (iii) of algorithm 5.86 ('Remove trivial rules') must be modified since neither the IPOL nor the POLS is a simplification ordering, in general.

Note that the presented technique is not a decision procedure (because of lemma 5.78). However, it is very useful in practice as confirmed by the following statistics: We have applied algorithm 5.86 to 242 TRSs (which are all orientable with the help of simple-mixed polynomials). For 228 TRSs (94.2%) the method was successful. Note that 35.6% of the examples require the algorithm to backtrack with the following initial parameter values: $\mu = 1$, $\nu(\llbracket.\rrbracket) = \llbracket.\rrbracket$, $\alpha_2 = 0$ for each $\llbracket f\rrbracket(x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$ and (ii) of lemma 5.78. The simplification processes (Ψ and Ψ_I) are very often necessary for deducing a set of interpretations to orient a given TRS.

$\boxed{5.86}$ **Algorithm** *This algorithm determines whether a finite TRS $\mathcal{R}$ over $\mathcal{T}(\mathcal{F},\mathcal{X})$ can be oriented by an* SPOL *using simple-mixed interpretations for the operators in $\mathcal{F}$. Let $\mathcal{F} = \bigcup\{f_i\}$.*

Initialization:

(i) *Define the interpretations (by using variable coefficients):* $\llbracket.\rrbracket := \Upsilon(\mathcal{F})$

Setting parameters:

- *Specify μ: $\mu := n$ with $n \in \mathbb{N}\backslash\{0\}$ arbitrary but fixed*
- *Specify $\llbracket f\rrbracket(x)$: $\alpha_1 = 0$ or $\alpha_2 = 0$ for each $\llbracket f\rrbracket(x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$*

(ii) *Remove special constant monomials:* $\llbracket.\rrbracket := \nu(\llbracket.\rrbracket)$

- *Specify $\mathcal{R}ed$: Choose (i) or (ii) in lemma 5.78*

Simplification

(iii) *Remove trivial rules:* $\mathcal{R}' := \mathcal{R} \setminus \{l \rightarrow r \in \mathcal{R} \mid l \succ_{\mathsf{EMB}} r\}$

(iv) *Set uninduced operators to minimal values:* $\llbracket.\rrbracket := \Psi(\llbracket.\rrbracket, \mathcal{R}')$

Splitting the rules into inequalities w.r.t. the coefficients:

(v) $\forall l_k \rightarrow r_k \in \mathcal{R}'$: $I_k := \mathcal{S}plit(\llbracket l_k\rrbracket - \llbracket r_k\rrbracket)$ *or* $I_k := \mathcal{S}plit^M(\llbracket l_k\rrbracket - \llbracket r_k\rrbracket)$

Simplification:

Repeat

(vi) *Apply arithmetic laws:*

$$\begin{aligned}\forall k\colon\ I_k := {} & \{p_i' \geq q_i' \mid p_i \geq q_i \in I_k \wedge \tfrac{p_i \geq q_i}{p_i' \geq q_i'} \in \mathcal{AL}\} \\ & \cup\ \{p_i \geq q_i \in I_k \mid \nexists \tfrac{p_i \geq q_i}{p_i' \geq q_i'} \in \mathcal{AL}\}\end{aligned}$$

(vii) *Set uninduced variables to minimal values: Apply Ψ_I to $\llbracket.\rrbracket$ and each I_k*

Until no changes occur

Eliminating addition and reducing a product to a linear polynomial:

(viii) $\forall k$: $I_k' := \{\mathcal{R}en(\mathcal{L}in(\mathcal{R}ed(p \geq q))) \mid p \geq q \in I_k\}$

Collecting:

(ix) $I := \bigcup_k I_k'$

Applying the method of complete description:

(x) *The method of complete description will possibly generate a solution of the derived set I of linear inequalities. Afterwards, $\mathcal{R}en^{-1}$ must be applied to the values of the solution in order to deduce the $\alpha_{r_1 \ldots r_n}$ from the $\alpha^*_{r_1 \ldots r_n}$. If I has no solution, the parameters will be set to other values by backtracking to step 'Setting parameters'.*

Figure 5.9: Algorithm for generating simple-mixed interpretations

Chapter 6

Comparison of the Orderings

In chapters 3, 4 and 5, various (simplification) orderings have either been recapitulated or introduced for the first time. We quote from [Dick *et al.*, 1990]:

> TERMINATION OF A SET OF RULES IS UNDECIDABLE IN GENERAL [HUET AND LANKFORD, 1978] SO THE BEST WE CAN HOPE FOR IS TO HAVE SEVERAL SIMPLIFICATION ORDERINGS AVAILABLE WHICH BETWEEN THEM COPE WITH MANY OF THE KINDS OF RULES THAT OCCUR IN PRACTICE. THUS NO ONE FAMILY OF SIMPLIFICATION ORDERINGS IS NECESSARILY 'BETTER' THAN ANOTHER. EACH MAY ORDER SOME RULES THAT OTHERS CANNOT, ALTHOUGH SOME MAY BE EASIER TO IMPLEMENT, FASTER TO EXECUTE OR SIMPLER TO WRITE DOWN THAN OTHERS. [...]

However, there exist several criteria for comparing orderings:

Kind of parameters	The orderings of chapter 3 are syntactic orderings because they are based on precedences. The orderings of chapter 4 and chapter 5 are semantic orderings since they use weight functions and polynomial interpretations.
Number of different parameters	For the syntactic orderings, only a finite number of parameters exists (for finite $\mathcal{F}$). In contrast, infinitely many different semantic orderings (of the same class, e.g. KBO) can be generated.
How to implement?	We have experienced that the path and decomposition orderings as well as the Knuth-Bendix orderings can easily be implemented by following the formal definitions of the orderings. Obviously, it is more difficult to implement, for example, the KNSS or the IRDS than the RPOS. Due to the complexity of $\sqsupset$, it is still more time-consuming to implement one of the polynomial orderings.
Automatic generation	As we have seen, it is decidable whether there is a syntactic ordering or a Knuth-Bendix ordering for orienting a given set of equations (or proving the termination of a given TRS). Furthermore, there exist procedures for explicitly generating an appropriate ordering of these two classes if it exists. However, polynomial orderings can only be generated by heuristic-based approaches. This is a direct consequence of the remarks given above in 'Number of different parameters'.

Totality on ground terms	Considering ground terms $\mathcal{G}(\mathcal{F})$ instead of $\mathcal{T}(\mathcal{F},\mathcal{X})$ and presuming a total precedence, path and decomposition orderings are total on $\mathcal{G}(\mathcal{F})/\approx$. The KBOS is also total on $\mathcal{G}(\mathcal{F})/\approx$ whereas the KBO is total on $\mathcal{G}(\mathcal{F})$. However, polynomial orderings are not total on ground terms. In [Narendran and Rusinowitch, 1991], an AC-compatible polynomial ordering has been developed which is total on $\mathcal{G}(\mathcal{F})/AC$ (see historical remark on page 120). In [Gnaedig, 1988], conditions under which the associative path orderings can be total on ground terms are investigated. Assuming total precedence, the AD-RPO is total on $\mathcal{G}(\mathcal{F})/ACD$ where $\mathcal{G}(\mathcal{F})/ACD$ is the algebra $\mathcal{G}(\mathcal{F})$ modulo AC-axioms and distributivity axioms. It is obvious that there exists no AC-compatible total ordering on $\mathcal{G}(\mathcal{F})$. The AC-compatible ordering of [Rubio and Nieuwenhuis, 1993] is total on $\mathcal{G}(\mathcal{F})/AC$.
Length of derivations	The derivation height of a term t, relative to a TRS $\mathcal{R}$, $dh_{\mathcal{R}}(t)$, is the length of a longest derivation from t. If termination of $\mathcal{R}$ can be proved by the POL, then $dh_{\mathcal{R}}$ is bounded from above by a doubly exponential function, whereas termination proofs by Knuth-Bendix orderings are possible even for systems where $dh_{\mathcal{R}}$ cannot be bounded by any primitive recursive function[1)] (see, for example, [Hofbauer and Lautemann, 1989]). In [Hofbauer, 1990], it has been shown that a termination proof for a TRS using the RPO *without* lexicographic status yields a primitive recursive bound on the length of derivations (see also [Hofbauer, 1991]). This result holds for a great variety of the path and decomposition orderings of chapter 3 if lexicographic status is not incorporated.

Additionally, the following two criteria can be applied to orderings on terms.

- **Section 6.1:** Mostly, the power of an ordering is measured against another ordering by examining the *relation* between their sets of comparable pairs of terms. This section provides a detailed comparison of the power of the presented orderings.
- **Section 6.2:** In this section, we deal with theoretical time complexities (for the worst case) of most of the orderings described in chapters 3 and 4.

6.1 Towards the Power of the Orderings

In this section, we compare the power of the orderings presented in chapters 3, 4 and 5.

The *power* of an ordering is the set of comparable pairs of terms. We do not compare the size of these sets but *examine the relation between them.*

There are three possible relations:

[1)] This fact is mainly influenced by unary operators with weight zero.

- Two orderings $\succ_{\mathsf{T1}}$, $\succ_{\mathsf{T2}}$ can be *equivalent*:

$$\succ_{\mathsf{T1}} = \succ_{\mathsf{T2}} \quad \text{if, and only if, } \forall s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})\text{: } s \succ_{\mathsf{T1}} t \iff s \succ_{\mathsf{T2}} t$$

- An ordering $\succ_{\mathsf{T1}}$ can be a *proper extension* of an ordering $\succ_{\mathsf{T2}}$ ($\succ_{\mathsf{T2}}$ is *properly included* in $\succ_{\mathsf{T1}}$ or $\succ_{\mathsf{T1}}$ is *stronger* than $\succ_{\mathsf{T2}}$):

$$\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}} \quad \text{if, and only if, } \succ_{\mathsf{T1}} \supseteq \succ_{\mathsf{T2}} \quad \text{and} \quad \succ_{\mathsf{T1}} \neq \succ_{\mathsf{T2}}$$

  For $\supseteq$, see definition 1.10 on page 8.

- Two orderings $\succ_{\mathsf{T1}}$, $\succ_{\mathsf{T2}}$ are *incomparable*[2]:

$$\succ_{\mathsf{T1}} \Updownarrow \succ_{\mathsf{T2}} \quad \text{if, and only if,} \quad \begin{array}{l} \exists s_1, s_2, t_1, t_2 \in \mathcal{T}(\mathcal{F},\mathcal{X})\text{:} \\ s_1 \succ_{\mathsf{T1}} t_1 \;\wedge\; s_1 \not\succ_{\mathsf{T2}} t_1 \;\wedge\; s_2 \succ_{\mathsf{T2}} t_2 \;\wedge\; s_2 \not\succ_{\mathsf{T1}} t_2 \end{array}$$

  Consequently, the proof of incomparability of two orderings is composed of specifying two counter-examples.

The orderings presented in the former chapters depend on different subsets of the following parameters: a precedence $\succ$, a status function τ, an extended weight function φ and a polynomial interpretation $\llbracket . \rrbracket$. For example, the PSO is based only on a precedence whereas the POLS uses an interpretation, a precedence *and* a status function. The relations between orderings (as described above) are existentially quantified with these parameters. Thus, we have to define the relations between orderings more precisely.

$\boxed{6.1}$ **Definition (Parameterized Ordering, Relations Between Orderings)** *Let* $\succ$ *be a precedence,* τ *be a status function,* φ *be an extended weight function and* $\llbracket . \rrbracket$ *be a polynomial interpretation.*

- *Parameterized ordering: A reduction ordering* $\succ_{\mathsf{T}}$ *based on an arbitrary subset of the parameters* $\succ, \tau, \varphi$ *and* $\llbracket . \rrbracket$ *is denoted by* $\succ_{\mathsf{T}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh$.
- *Let* $\succ_{\mathsf{T1}}$, $\succ_{\mathsf{T2}}$ *be two reduction orderings and* $s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})$*:*

$$\begin{array}{llcl}
\textit{(i)} & \succ_{\mathsf{T1}} = \succ_{\mathsf{T2}} & \iff & \bullet\; s \succ_{\mathsf{T1}} t \leadsto \exists \succ, \tau, \varphi, \llbracket . \rrbracket\text{: } s \succ_{\mathsf{T2}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh\; t \;\wedge \\
 & & & \bullet\; s \succ_{\mathsf{T2}} t \leadsto \exists \succ, \tau, \varphi, \llbracket . \rrbracket\text{: } s \succ_{\mathsf{T1}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh\; t \\
 & & & \\
\textit{(ii)} & \succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}} & \iff & \bullet\; s \succ_{\mathsf{T2}} t \leadsto \exists \succ, \tau, \varphi, \llbracket . \rrbracket\text{: } s \succ_{\mathsf{T1}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh\; t \;\wedge \\
 & & & \bullet\; \exists s,t\text{: } s \succ_{\mathsf{T1}} t \;\wedge\; \nexists \succ, \tau, \varphi, \llbracket . \rrbracket\text{: } s \succ_{\mathsf{T2}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh\; t \\
 & & & \\
\textit{(iii)} & \succ_{\mathsf{T1}} \Updownarrow \succ_{\mathsf{T2}} & \iff & \bullet\; \exists s,t\text{: } s \succ_{\mathsf{T1}} t \;\wedge\; \nexists \succ, \tau, \varphi, \llbracket . \rrbracket\text{: } s \succ_{\mathsf{T2}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh\; t \;\wedge \\
 & & & \bullet\; \exists s,t\text{: } s \succ_{\mathsf{T2}} t \;\wedge\; \nexists \succ, \tau, \varphi, \llbracket . \rrbracket\text{: } s \succ_{\mathsf{T1}}\Rsh\succ, \tau, \varphi, \llbracket . \rrbracket\Lsh\; t \quad \diamond
\end{array}$$

Henceforth, the expression 'syntactic orderings' ('semantic orderings') only stands for the syntactic orderings (semantic orderings, respectively) *contained in this thesis.*

---

[2] We deliberately borrow the sign $\Updownarrow$ for denoting incomparability of two elements (see definition 1.10).

As it will be proved later on, semantic and syntactic orderings are incomparable. All syntactic orderings and some semantic orderings are based on a precedence. This parameter may be either partial or total. For the following reason we are only interested in the latter: Our results would be more general if we could give some information about the comparisons of orderings *unrelated in the precedence.* Unfortunately, the proofs of such comparisons are somewhat complex. Therefore, we are only interested in total precedences since this kind of comparison in conjunction with the property '$\succ$-monotonicity' and lemma 6.4 leads to the desired result.

$\boxed{6.2}$ **Definition ($\succ$-Monotonicity)** *Let* $\succ_{\mathsf{T}}$ *be a precedence-based reduction ordering and* $\succsim_1$, $\succsim_2$ *be two quasi-precedences. Then* $\succ_{\mathsf{T}}$ *is said to be* $\succ$-monotonic[3)] *if, and only if,*

$$\forall \succsim_1, \succsim_2, \tau, \varphi, [\![.]\!]\colon \succsim_1 \supseteq \succsim_2 \ \leadsto\ [s \succ_{\mathsf{T}} \Rsh \succsim_2, \tau, \varphi, [\![.]\!] \Lsh\ t \ \leadsto\ s \succ_{\mathsf{T}} \Rsh \succsim_1, \tau, \varphi, [\![.]\!] \Lsh\ t] \qquad \diamond$$

In other words, if the precedence is augmented (while simultaneously keeping the other parameters unchanged), the reduction ordering becomes stronger.

$\boxed{6.3}$ **Lemma** *The orderings*

- ▶ DOS, IRD, IRDS, KNSS, PSDS, PSO, RDO, RDOS, RPO, RPOS
- ▶ KBO, KBOS
- ▶ IPOL, POLS

are $\succ$-monotonic. ◇

The $\succ$-monotonicity of the RPOS (the RDO and the RDOS) is proved in [Lescanne, 1983c] ([Reinig and Jouannaud, 1981] and [Lescanne, 1983b], respectively).

Note that there exists an extension (if the precedence is strictly partial) for each precedence-based and $\succ$-monotonic ordering. Given a partial quasi-precedence $\succsim$ and any $\succ$-monotonic reduction ordering $\succ_{\mathsf{TO}}$, then $\succ_{\mathsf{TTO}}$ is defined as

$$s \succ_{\mathsf{TTO}} t \iff \forall \succsim'\colon [\succsim' \supseteq \succsim \ \wedge \ \succsim' \ \textit{total}] \leadsto s \succ'_{\mathsf{TO}} t$$

where all possible total extensions $\succsim'$ of the given precedence $\succsim$ are considered and the terms s, t have to be compared in the same way (for each $\succsim'$). Such ordering extensions are called closure orderings. *They were suggested by Plaisted, and presented for the restricted cases of the* RPOS *and the* RDOS *by Forgaard ([Forgaard, 1984]) as well as for the general case by Lescanne ([Lescanne, 1990]). For example, $h(h(x) \star y)$ is greater (in the sense of the closure of the* RPOS*) than $x \star h(x \star y)$ if the precedence is empty and the status function satisfies $\tau(\star) =$ ① (see [Lescanne, 1990]).*

An extension of a precedence can often be used to enable the comparison of two terms which cannot be compared with the original precedence. Due to the $\succ$-monotonicity, comparisons without the extension are still valid (see also section 3.4).

$\boxed{6.4}$ **Lemma** *Let $\succ$ be a precedence and $\succ_{\mathsf{T1}}$, $\succ_{\mathsf{T2}}$ be $\succ$-monotonic reduction orderings. Then $\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}}$ w.r.t.* arbitrary *precedences if $\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}}$ w.r.t.* total *precedences.* ◇

[3)] also called *monotonic w.r.t. the precedence*

Thus, whenever a $\succ$-monotonic reduction ordering $\succ_{\mathsf{T1}}$ is stronger than another $\succ$-monotonic reduction ordering $\succ_{\mathsf{T2}}$ w.r.t. *total* precedences, then $\succ_{\mathsf{T1}}$ is also stronger than $\succ_{\mathsf{T2}}$ for arbitrary precedences. This fact leads to simpler proofs of inclusion assertions since only total precedences (in case of $\succ$-monotonic orderings) must be considered. Taking this into account, the relations between the orderings will be checked on additional premises in form of

- total precedences and *arbitrary* terms (see subsection 6.1.1),
- total precedences and *ground* terms (see subsection 6.1.2) and
- total precedences and *monadic* terms (see subsection 6.1.3).

In order to retain the relations proved in the three following subsections and to find one of them easily we use a kind of Hasse diagrams. If an ordering $\succ_{\mathsf{T2}}$ is included in another ordering $\succ_{\mathsf{T1}}$ (i.e. $\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}}$), then we arrange $\succ_{\mathsf{T1}}$ above $\succ_{\mathsf{T2}}$ and connect them with a line. An ordering or equivalent orderings are represented by a cube. The Hasse diagrams illustrating the results of the current section are given in figures 6.2, 6.4, 6.6 and 6.8.

These illustrations are supplemented by counter-examples proving strictness of the relations between orderings. More precisely, one example is given whenever $\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}}$ holds, and two examples are presented whenever $\succ_{\mathsf{T1}} \Bumpeq \succ_{\mathsf{T2}}$ is valid. The counter-examples are consecutively numbered and listed in figure 6.1[4]. The contents of the tabulars of figures 6.3, 6.5, 6.7 and 6.9 (corresponding to figures 6.2, 6.4, 6.6 and 6.8, respectively) relate to these numbers. The tabulars provide a counter-example for a relation $\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}}$ in the following way[5]: Take the example contained in figure 6.1 which corresponds to the number being both in the row of $\succ_{\mathsf{T1}}$ and the column of $\succ_{\mathsf{T2}}$. We have tried to find minimal counter-examples (i.e. TRSs) w.r.t. the number of rules, the number of different operators as well as the length of the rules. In case of two examples with similar sizes, one of them having pathological significance and the other one having practical relevance, we have chosen the latter one.

Unfortunately, certain relations between orderings have not been determined yet. For these relations we use '?' in the diagrams of counter-examples and dashed lines in the Hasse diagrams. Fortunately, most of the unknown relations are not very important. However, at the end of each subsection, some ideas and conjectures about possible relations can be found.

6.1.1 Total Precedences and Arbitrary Terms

This subsection deals with the most general of the cases listed above: We compare orderings w.r.t. *arbitrary terms* and total precedences.

We have to point out that a small part of the lemmata of this subsection *is contained in [Rusinowitch, 1987]. Rusinowitch was the first one (a preliminary version of [Rusinowitch, 1987] has been published at the first RTA in 1985) who gave a comparison of several path and decomposition orderings. Some of the other publications on path and decomposition orderings provide comparisons with the* RPO.

Obviously, an ordering will become more powerful if it is extended by a status function (with the exception of the RDO and the RDOS).

---

[4] Orderings proving the termination of these examples are given at the end of appendix B on pages 258–267.

[5] Note that the counter-example for $\succ_{\mathsf{T1}} \supset \succ_{\mathsf{T2}}$ and that for $\succ_{\mathsf{T2}} \supset \succ_{\mathsf{T1}}$ ensures the validity of the relation $\succ_{\mathsf{T1}} \Bumpeq \succ_{\mathsf{T2}}$.

1	$(x \star y) \star z$	$\rightarrow$	$x \star (y \star z)$
2	$f(x, g(y, a))$	$\rightarrow$	$g(f(x, y), f(x, y))$
3	$((\neg x) \supset y) \vee z$	$\rightarrow$	$(\neg((\neg y) \wedge (\neg z))) \vee x$
4	$f(g(x))$	$\rightarrow$	$g(f(f(x)))$
5	$f(g(a))$	$\rightarrow$	$g(g(f(a)))$
6	$f(a)$ $g(g(a))$	$\rightarrow$ $\rightarrow$	$g(a)$ $f(a)$
7	$(\neg x) \supset (y \supset z)$	$\rightarrow$	$y \supset (x \vee z)$
8	$((-x) - (-x)) - ((-y) - (-y))$	$\rightarrow$	$(x - y) - (x - y)$
9	$\wedge(\neg\neg x, y, \neg z)$	$\rightarrow$	$\wedge(x, y, x \overline{\wedge} z)$
10	$x \star ((-y) \star y)$	$\rightarrow$	$(-(y \star y)) \star x$
11	$f(g(s(x)), y)$ $g(s(x))$	$\rightarrow$ $\rightarrow$	$f(y, s(x))$ $s(g(g(x)))$
12	$0 + 1$	$\rightarrow$	$1 + 0$
13	$0 + s(s(0))$	$\rightarrow$	$s(0) + s(0)$
14	$f(g(x), g(y))$	$\rightarrow$	$f(h(x, y), x)$
15	$f(g(x))$ $h(f(x))$ $i(h(x))$	$\rightarrow$ $\rightarrow$ $\rightarrow$	$g(g(f(x)))$ $f(f(h(x)))$ $h(h(i(x)))$
16	$(0 + 1) \star 2$ $+, \star \in \mathcal{F}_{AC}$	$\rightarrow$	$(0 \star 2) + (1 \star 2)$
17	$true \wedge false$ $\wedge \in \mathcal{F}_{AC}$	$\rightarrow$	$if(true, false, false)$
18	$true \oplus true$ $(true \oplus true) \wedge false$ $\wedge, \oplus \in \mathcal{F}_{AC}$	$\rightarrow$ $\rightarrow$	$\neg true$ $(true \wedge false) \oplus (true \wedge false)$
19	$f(a, b + c)$ $+ \in \mathcal{F}_{AC}$	$\rightarrow$	$f(b + c, a)$
20	$(\neg x) \supset (\neg y)$ $\wedge \in \mathcal{F}_{AC}$	$\rightarrow$	$y \supset (x \wedge y)$
21	$f(a) + a$ $g(g(a)) + a$ $+ \in \mathcal{F}_{AC}$	$\rightarrow$ $\rightarrow$	$a + g(a)$ $a + f(a)$
22	$f(g(a)) + a$ $+ \in \mathcal{F}_{AC}$	$\rightarrow$	$g(g(f(a))) + a$
23	$(true \wedge false) \wedge (true \wedge false)$ $\wedge \in \mathcal{F}_{AC}$	$\rightarrow$	$if(true, false, false)$
24	$f(g(f(x, y), z), x)$ $f, g \in \mathcal{F}_{AC}$	$\rightarrow$	$f(f(g(x, y), z), f(x, g(y, z)))$

Figure 6.1: Counter-examples for the comparison of orderings

6.5 Lemma *Let $\succ$ be a total precedence. Then*

(i) RPOS $\supset$ RPO   *(iv)* KBOS $\supset$ KBO   *(v)* POLS $\supset$ IPOL
(ii) PSDS $\supset$ PSO
(iii) IRDS $\supset$ IRD ◇

The following assertions have been proved in [Rusinowitch, 1987]. The RPO is identical to the RPOS if all operators have status ⓜ.

6.6 Lemma ([Rusinowitch, 1987]) *Let $\succ$ be a precedence*[6]*. Then*

$$\begin{array}{rlcl} (i) & \mathsf{PSO} & \supset & \mathsf{RPO} \\ (ii) & \mathsf{RDO} & \supset & \mathsf{RPO} \\ (iii) & \mathsf{IRD} & \supset & \mathsf{RDO} \end{array} \qquad \diamond$$

In [Rusinowitch, 1987], it was shown that the IRD and the KNSS *without* status are equivalent. The following lemma states the corresponding assertions of this fact and the previous lemma w.r.t. the integration of a status function.

6.7 Lemma *Let $\succ$ be a total precedence. Then*

$$\begin{array}{rlcl} (i) & \mathsf{RDOS} & \supset & \mathsf{RPOS} \\ (ii) & \mathsf{IRDS} & = & \mathsf{KNSS} \end{array} \qquad \diamond$$

Regarding decomposition orderings, the depth first status is more powerful than the breadth first status. Furthermore, the DOS is stronger than the RDOS.

6.8 Lemma *Let $\succ$ be a total precedence. Then*

$$\begin{array}{rlcl} (i) & \mathsf{IRD} & \supset & \mathsf{PSO} \\ (ii) & \mathsf{IRDS} & \supset & \mathsf{PSDS} \\ (iii) & \mathsf{DOS} & \supset & \mathsf{RDOS} \end{array} \qquad \diamond$$

It is not difficult to show the following inclusions.

6.9 Lemma *Let $\succ$ be a total precedence. Then*

$$\begin{array}{rlcl} (i) & \mathsf{PSDS} & \supset & \mathsf{RDOS} \\ (ii) & \mathsf{IRDS} & \supset & \mathsf{DOS} \\ (iii) & \mathsf{DOS} & \supset & \mathsf{RDO} \end{array} \qquad \diamond$$

The following inclusion holds for the polynomial orderings: IPOL $\supset$ POL if $\forall c \in \mathcal{F}$: $\mathcal{A}r(c) = 0 \;\leadsto\; [\![c]\!] \geq 2$. Furthermore, the two classes – Knuth-Bendix orderings and polynomial orderings – of semantic orderings have a close connection.

6.10 Lemma *Let $\succ$ be a total precedence. Then*

$$\begin{array}{rlcl} (i) & \mathsf{IPOL} & \supset & \mathsf{KBO} \\ (ii) & \mathsf{POLS} & \supset & \mathsf{KBOS} \end{array} \qquad \diamond$$

To conclude the comparisons of the classical (non-AC) orderings, counter-examples for the following incomparabilities must be given.

---

[6] Note that the precedence may not necessarily be total.

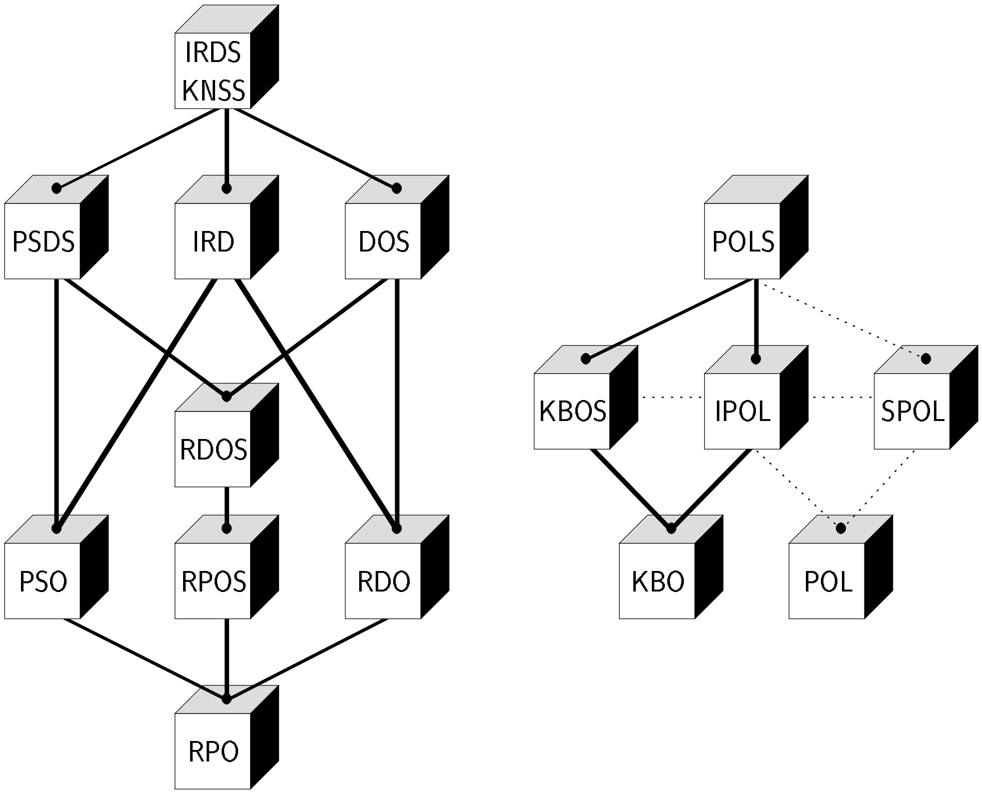


Figure 6.2: Total precedences and arbitrary terms

6.11 Lemma

(i) RPOS $\Bumpeq$ PSO, RDO
(ii) PSO $\Bumpeq$ RDO, RDOS, DOS
(iii) RDO $\Bumpeq$ RDOS, PSDS
(iv) PSDS $\Bumpeq$ IRD, DOS
(v) IRD $\Bumpeq$ RPOS, RDOS, DOS
(vi) KBO $\Bumpeq$ POL, SPOL
(vii) KBOS $\Bumpeq$ POL, SPOL ◇

6.12 Lemma

(i) KBO $\Bumpeq$ RPO, RPOS, PSO, PSDS, RDO, RDOS, DOS, IRD, IRDS
(ii) KBOS $\Bumpeq$ RPO, RPOS, PSO, PSDS, RDO, RDOS, DOS, IRD, IRDS
(iii) POL $\Bumpeq$ RPO, RPOS, PSO, PSDS, RDO, RDOS, DOS, IRD, IRDS
(iv) SPOL $\Bumpeq$ RPO, RPOS, PSO, PSDS, RDO, RDOS, DOS, IRD, IRDS
(v) IPOL $\Bumpeq$ RPO, RPOS, PSO, PSDS, RDO, RDOS, DOS, IRD, IRDS
(vi) POLS $\Bumpeq$ RPO, RPOS, PSO, PSDS, RDO, RDOS, DOS, IRD, IRDS ◇

The comparison of the orderings suitable for TRSs modulo AC-theories yields two linear chains described by the following two lemmata.

6.13 Lemma *Let $\succ$ be a total precedence. Then*

$$\text{AE-IRDS} \supset \text{AE-RPOS} \supset \text{AE-RPO} \supset \text{AD-RPO} \supset \text{A-RPO}$$ ◇

	RPO	RPOS	RDO	RDOS	PSO	PSDS	DOS	IRD	IRDS	KBO	KBOS	POL	SPOL	IPOL	POLS
RPO	■	■	■	■	■	■	■	■	■	2	2	2	2	2	2
RPOS	1	■	1	■	1	■	■	1	■	2	2	2	2	2	2
RDO	7	7	■	7	8	8	■	■	■	2	2	2	2	2	2
RDOS	1	14	1	■	1	■	■	1	■	2	2	2	2	2	2
PSO	7	7	9	9	■	■	9	■	■	2	2	2	2	2	2
PSDS	1	7	1	9	1	■	9	1	■	2	2	2	2	2	2
DOS	1	7	1	7	1	8	■	1	■	2	2	2	2	2	2
IRD	7	7	9	9	8	8	9	■	■	2	2	2	2	2	2
IRDS	1	7	1	9	1	8	9	1	■	2	2	2	2	2	2
KBO	3	3	3	3	3	3	3	3	3	■	■	4	4	■	■
KBOS	4	4	4	4	4	4	4	4	4	10	■	4	4	?	■
POL	3	3	3	3	3	3	3	3	3	5	5	■	■	?	?
SPOL	3	3	3	3	3	3	3	3	3	5	5	?	■	?	?
IPOL	4	4	4	4	4	4	4	4	4	5	5	4	4	■	■
POLS	4	4	4	4	4	4	4	4	4	5	5	4	4	11	■

This diagram takes the equivalence 'IRDS =KNSS' (see lemma 6.7) into account.

Figure 6.3: Counter-examples for arbitrary terms

6.14 **Lemma** *Let $\succ$ be a total precedence. Then*

IACK = ACKM ⊃ ACK ◇

Obviously, the orderings C-RPO and CK for C-theories are incomparable with the orderings contained in the two previous lemmata. Furthermore, the incomparabilities in the following lemma complete the comparison of AC-orderings w.r.t. total precedences and arbitrary terms.

6.15 **Lemma**
(i) ACK ≎ A-RPO,AD-RPO,AE-RPO,AE-RPOS,AE-IRDS
(ii) ACKM ≎ A-RPO,AD-RPO,AE-RPO,AE-RPOS,AE-IRDS
(iii) AC-POL ≎ ACK,ACKM,A-RPO,AD-RPO,AE-RPO,AE-RPOS,AE-IRDS ◇

The following relations remain open:

- IPOL ⊃ (S)POL or IPOL ≎ (S)POL: The termination of example 4 can be proved by the IPOL but not by the (S)POL. If there exists a set of equations which can be oriented with the $\mathsf{(S)POL}$ but not with the IPOL, the relation $\mathsf{IPOL} \Bumpeq \mathsf{(S)POL}$ will hold. This counter-example must either satisfy the properties described in remark 5.7 on page 96 or contain a constant necessarily interpreted as 1 (which is not allowed by the definition of the IPOL). It is implausible that an example with such conditions exists.

- ▶ $\mathsf{POLS} \supset \mathsf{(S)POL}$ or $\mathsf{POLS} \Bumpeq \mathsf{(S)POL}$: Example 4 is ordered w.r.t. the POLS but not w.r.t. the $\mathsf{(S)POL}$. We conjecture that a counter-example for the relation $\mathsf{(S)POL} \supset \mathsf{POLS}$ does not exist (see the previous item).
- ▶ $\mathsf{SPOL} \supset \mathsf{POL}$ or $\mathsf{SPOL} = \mathsf{POL}$: It is obvious (according to the definitions) that the POL cannot be more powerful than the SPOL. For proving the relation $\mathsf{SPOL} \supset \mathsf{POL}$, an example similar to that for the relation $\mathsf{IPOL} \Bumpeq \mathsf{SPOL}$ (see above) is necessary. We conjecture the SPOL and the POL to be identical.
- ▶ $\mathsf{IPOL} \supset \mathsf{KBOS}$ or $\mathsf{IPOL} \Bumpeq \mathsf{KBOS}$: Example 5 is ordered w.r.t. the IPOL but not w.r.t. the KBOS. A counter-example (which is ordered w.r.t. the KBOS) including a rule $f(s_1, s_2) \rightarrow f(t_1, t_2)$ such that $[\![f(s_1, s_2)]\!] = [\![f(t_1, t_2)]\!]$ and $\mathcal{V}ar(t_1) \setminus \mathcal{V}ar(s_1) \neq \emptyset$ would imply the relation $\mathsf{IPOL} \Bumpeq \mathsf{KBOS}$.

6.1.2 Total Precedences and Ground Terms

Arbitrary terms provided, we have compared the orderings. It is a question now whether the relations among the orderings will endure if this precondition is restricted. Regarding syntactic orderings, most of the proper inclusions holding for arbitrary terms will become equalities if we restrict the set of arbitrary terms to ground terms. If we consider $\mathcal{G}(\mathcal{F})$ and take total precedences, then the syntactic orderings with multiset status are equivalent and included in the syntactic orderings with arbitrary status which themselves are equivalent.

6.16 Lemma *Assuming total precedences and ground terms, the following relations hold:*

$$\begin{array}{rl} \textit{(i)} & \mathsf{RPO} = \mathsf{PSO} = \mathsf{RDO} = \mathsf{IRD} \\ \textit{(ii)} & \mathsf{RPOS} = \mathsf{PSDS} = \mathsf{RDOS} = \mathsf{DOS} = \mathsf{IRDS} \\ \textit{(iii)} & \mathsf{RPOS} \supset \mathsf{RPO} \end{array}$$

◇

The following relations between semantic orderings remain the same when restricting arbitrary terms to ground terms.

6.17 Lemma *Assuming total precedences and ground terms, the following relations hold:*

$$\begin{array}{rl} \textit{(i)} & \mathsf{POLS} \supset \mathsf{KBOS} \supset \mathsf{KBO} \\ \textit{(ii)} & \mathsf{IPOL} \supset \mathsf{KBO} \end{array}$$

◇

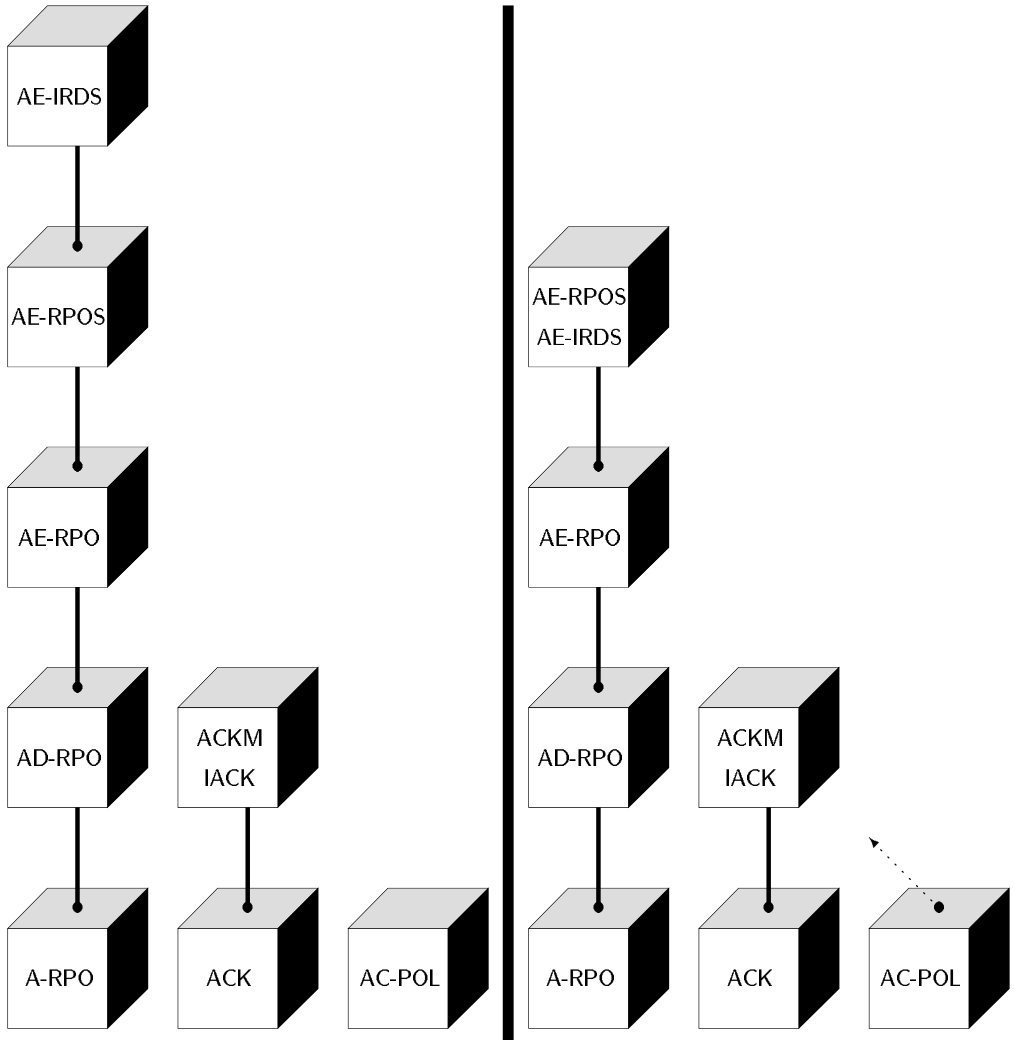
The following lemma describes the detected incomparabilities between syntactic and semantic orderings.

6.18 Lemma *Assuming ground terms, the following relations hold:*

$$\begin{array}{rl} \textit{(i)} & \mathsf{RPO} \Bumpeq \mathsf{KBO}, \mathsf{KBOS} \\ \textit{(ii)} & \mathsf{RPOS} \Bumpeq \mathsf{KBO}, \mathsf{KBOS} \end{array}$$

◇

Regarding AC-orderings, there is only one change when considering ground terms instead of arbitrary terms: The AE-IRDS is no more stronger than the AE-RPOS.



Total precedences and arbitrary terms

Total precedences and ground terms[7)]

Figure 6.4: AC-theories

6.19 **Lemma** *Assuming total precedences and ground terms, the following relations hold:*

(i) AE-IRDS $=$ AE-RPOS $\supset$ AE-RPO $\supset$ AD-RPO $\supset$ A-RPO
(ii) IACK $=$ ACKM $\supset$ ACK

(iii) ACK $\Bumpeq$ A-RPO,AD-RPO,AE-RPO,AE-RPOS
(iv) ACKM $\Bumpeq$ A-RPO,AD-RPO,AE-RPO,AE-RPOS ◇

[7)] Note that the AC-POL is not proved to be incomparable with the other orderings (as indicated in the diagram).

	ACK	ACKM	AC-POL	A-RPO	AD-RPO	AE-RPO	AE-RPOS	AE-IRDS
ACK			4[8]	23	23	23	23	23
ACKM	17		4[8]	17	17	17	17	17
AC-POL	22	22		21	21	21	21	21
A-RPO	22	22	2[9]					
AD-RPO	16	16	24[10]	16				
AE-RPO	16	16	24[10]	16	18			
AE-RPOS	16	16	24[10]	16	18	19		
AE-IRDS	16	16	24[10]	16	18	19	20[11]	

This diagram takes the equivalence 'ACKM = IACK' (see lemma 6.14) into account.

Figure 6.5: Counter-examples for AC-orderings

The following relations remain open:

- POLS ⊃ (S)POL or POLS ≎ (S)POL: Currently, we cannot verify either relation.
- POLS ⊃ IPOL or POLS = IPOL: According to the definitions, the IPOL cannot be stronger than the POLS. An example proving the strict inclusion is hardly conceivable.
- IPOL ≎ POL or IPOL = POL: A proof of one of the relations in the first item will probably clarify this problem.
- (S)POL ⊃ KBO(S) or (S)POL ≎ KBO(S), IPOL ⊃ KBOS or IPOL ≎ KBOS: Example 5 is ordered w.r.t. the polynomial orderings POL, SPOL and IPOL but not w.r.t. the Knuth-Bendix orderings KBO and KBOS. In order to verify the relations KBO(S) ≎ (S)POL and KBOS ≎ IPOL, a TRS is needed which is ordered w.r.t. the KBO(S) but not w.r.t. the polynomial orderings. We did not find such a TRS containing ground terms, only.
- IPOL ⊃ SPOL or IPOL ≎ SPOL or SPOL ⊃ IPOL or SPOL = IPOL: Each of the possible relations between the IPOL and the SPOL is conceivable.
- SPOL ⊃ POL or SPOL = POL: Obviously, the relation SPOL ⊇ POL holds. The existence of a TRS including a rule of the form $h(t) \to t$ with $[\![h]\!](x) = x$ would prove the strict inclusion. We doubt finding such an example (see also the remark on the relation 'IPOL ⊃ SPOL or IPOL ≎ SPOL' on page 148).
- POL,SPOL,IPOL,POLS ⊃ RPO(S) or POL,SPOL,IPOL,POLS ≎ RPO(S): Example 6 is responsible for the relation RPO(S) ⊇ POL,SPOL,IPOL,POLS being impossible. It is somewhat

---

[8] The KBOS is able to prove the termination of counter-example 4 whereas the POL is not. Note that the ACK(M) and the AC-POL are AC-extensions of the KBOS and the POL, respectively. Unfortunately, for the case of ground terms, this example is no consideration.

[9] The whole argumentation of the previous footnote can be applied, mutatis mutandis, to this case.

[10] Note that, for the case of ground terms, this example is no consideration.

[11] This example is responsible for the inclusion w.r.t. arbitrary terms. Note that AE-IRDS = AE-RPOS holds w.r.t. ground terms.

hard to find a ground TRS which is ordered w.r.t. the RPO(S) but not w.r.t. polynomial orderings.

- AC-POL $\supset$ A(D)-RPO,AE-RPO(S),ACK,ACKM or AC-POL $\Bumpeq$ A(D)-RPO,AE-RPO(S),ACK,ACKM: As in the case of the corresponding relations for non-AC-orderings, it is difficult to establish either (especially the latter) of these relations.

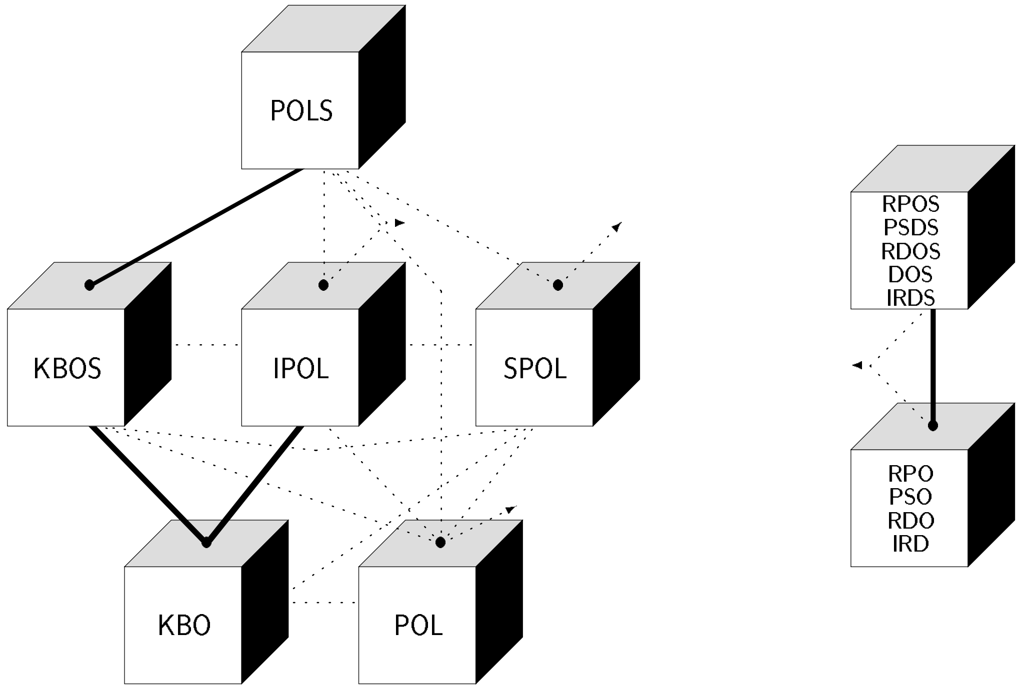


Figure 6.6: Total precedences and ground terms

6.1.3 Total Precedences and Monadic Terms

Remember that a *monadic term* only contains unary operators and either a constant or a variable. The subset of the monadic terms without constants can unequivocally be transformed into *strings* and vice versa. Therefore, the subsequent results refer to the corresponding orderings on *string rewriting systems*. Note that, on monadic terms, an ordering with status and the version without status coincide.

$\boxed{6.20}$ **Lemma** *Assuming total precedences and monadic terms, the following relations hold:*

(i) RPO = PSO = RDO = IRD = RPOS = PSDS = RDOS = DOS = IRDS
(ii) KBO = KBOS

(iii) IPOL $\supset$ KBO

(iv) RPO $\Bumpeq$ KBO,POL,SPOL,IPOL,POLS
(v) KBO $\Bumpeq$ POL,SPOL ◇

	RPO	RPOS	KBO	KBOS	POL	SPOL	IPOL	POLS
RPO	■	■	5	5	?	?	?	?
RPOS	12	■	5	5	?	?	?	?
KBO	6	6	■	■	?	?	■	■
KBOS	6	6	13	■	?	?	?	■
POL	6	6	5	5	■	■	?	?
SPOL	6	6	5	5	?	■	?	?
IPOL	6	6	5	5	?	?	■	■
POLS	6	6	5	5	?	?	?	■

This diagram takes the following equivalences of orderings into account: RPO = IRD = PSO = RDO and RPOS = DOS = IRDS = KNSS = PSDS = RDOS (see lemma 6.16).

Figure 6.7: Counter-examples for ground terms

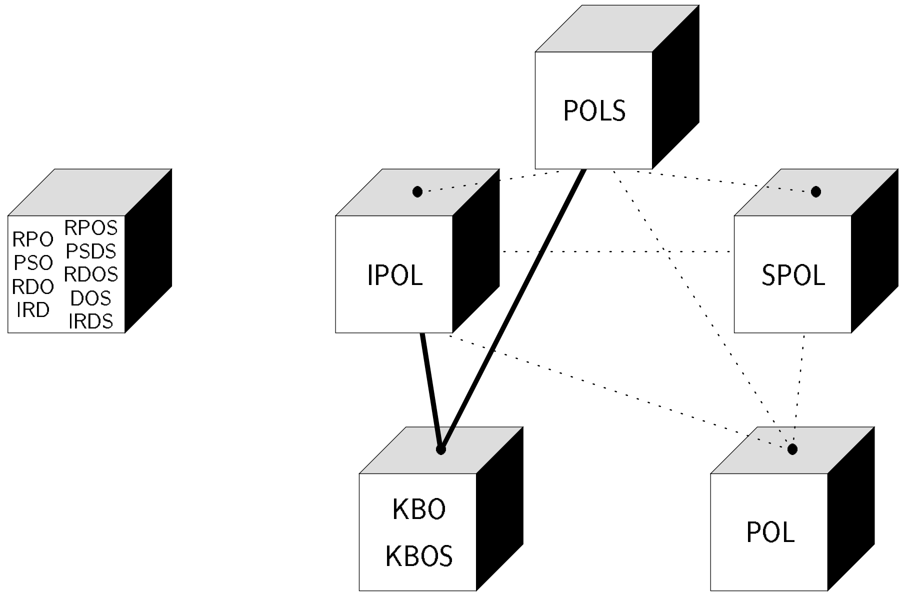


Figure 6.8: Total precedences and monadic terms

Obviously, AC-orderings are not applicable to TRSs over exclusively monadic terms since AC-operators have an arity of two, at least.

The following relations remain open:

- ▶ POLS ⊃ IPOL or POLS = IPOL: According to the definitions of both orderings, the relation POLS ⊂ IPOL and thus, the relation POLS ≎ IPOL is not possible. To verify that the POLS is strictly stronger than the IPOL, we have to exploit the relation $\sqsupseteq$ as described in the example '$(x \star y) + y \to (x+1) \star y$' on page 100.

- IPOL $\supset$ (S)POL or IPOL ≎ (S)POL: Example 4 implies that the relation (S)POL $\supseteq$ IPOL does not hold. However, in order to verify the strict inclusion, a TRS must be found which includes a rule of the form $h(t) \to t$ such that $[\![h]\!](x) = x$ (in case of the SPOL) and the IPOL's restriction '$\forall c \in \mathcal{F}, \mathcal{A}r(c) = 0$: $[\![c]\!]() \geq 2$' must be exploited (in case of the POL). Most probably, these properties cannot be guaranteed.
- POLS $\supset$ (S)POL or POLS ≎ (S)POL, SPOL $\supset$ POL or SPOL = POL: See the corresponding items for arbitrary terms on page 148.

	RPO	KBO	POL	SPOL	IPOL	POLS
RPO	■	5	15	15	15	15
KBO	6	■	4	4	■	■
POL	6	5	■	■	?	?
SPOL	6	5	?	■	?	?
IPOL	6	5	4	4	■	■
POLS	6	5	4	4	?	■

This diagram takes the following equivalences of orderings into account: RPO = DOS = IRD = IRDS = KNSS = PSDS = PSO = RDO = RDOS = RPOS and KBO = KBOS (see lemma 6.20).

Figure 6.9: Counter-examples for monadic terms

6.2 Towards the Time Complexities

In this section, *theoretical time complexities* of the worst cases for most orderings contained in this thesis will be presented. More formally, the time for comparing two terms w.r.t. a *given* ordering[12)] will be determined. The following worst case complexities will be proved (s and t are terms to be compared):

RPO	$O(\|s\| \cdot \|t\|)$	PSO	$O(\|s\|^3 \cdot \|t\|^3)$	PSDS	$O(\|s\|^4 \cdot \|t\|^4)$
KBOS	$O(\|s\| \cdot \|t\|)$	KNSS	$O(\|s\|^3 \cdot \|t\|^3)$	IRDS	$O(\|s\|^4 \cdot \|t\|^4)$

These expressions represent upper bounds which are not necessarily strict.

Obviously, these time complexities depend on the formal definitions of the orderings. Thus, two equivalent (w.r.t. the power) orderings can possess different complexities (see the time

---

[12)] i.e. the ordering is completely defined by its parameters (precedence, status function, extended weight function, polynomial interpretations) complexities of the KNSS and the IRDS). One of the most interesting results is the fact that decomposition orderings are more time-consuming than path orderings. This has been confirmed by various examples. To prove any assertion on the time complexity of polynomial orderings is somewhat difficult. It depends (i) on the method for proving the positiveness of polynomials and (ii) on the interpretations of the operators. Thus, for example, the time complexity of the approach of BenCherifa & Lescanne (described in subsection 5.3.1) cannot be determined since it is no decision procedure. However, the polynomial ordering's *empirical* time complexity is higher than that of the other orderings.

What follows are the lemmata corresponding to the tabular given above.

$\boxed{\textbf{6.21}}$ **Lemma** (RPO, **[Krishnamoorthy and Narendran, 1985]**) *Given two terms* s, t, *a precedence* $\succ$ *and a status function* τ, $s \succ_{\mathsf{RPO}} t$ *can be determined in time* $O(|s| \cdot |t|)$. $\diamond$

In [Snyder, 1993], it has been shown that although the straightforward implementation of the recursive definition of the RPOS *can result in exponential behaviour in the general case, it is possible to compare two ground terms w.r.t. the* RPOS *with total precedences in* $O(n \cdot lg\, n)$, *where* n *is the combined size of the two terms to be compared. The algorithm is based on the following concepts: (i) sorting the set of all subterms of the two terms to be compared, (ii) using a priority queue to order the subterms (i.e. doing a Heap sort) and (iii) proceeding bottom-up to insert the subterms into the queue. Unfortunately, this approach probably cannot be generalized for comparing* non-ground *terms with the* RPOS *based on* non-total *precedences.*

$\boxed{\textbf{6.22}}$ **Lemma** (KBOS) *Given two terms* s, t, *a precedence* $\succsim$, *an extended weight function* φ *and a status function* τ, $s \succ_{\mathsf{KBOS}} t$ *can be determined in time* $O(|s| \cdot |t|)$. $\diamond$

$\boxed{\textbf{6.23}}$ **Lemma** (PSO) *Given two terms* s, t *and a precedence* $\succ$, $s \succ_{\mathsf{PSO}} t$ *can be determined in time* $O(|s|^3 \cdot |t|^3)$. $\diamond$

$\boxed{\textbf{6.24}}$ **Lemma** (KNSS) *Given two terms* s, t, *a precedence* $\succsim$ *and a status function* τ, $s \succ_{\mathsf{KNSS}} t$ *can be determined in time* $O(|s|^3 \cdot |t|^3)$. $\diamond$

Note that in [Kapur *et al.*, 1985] an upper bound of $O(|s|^5 \cdot |t|^5)$ is proved for the KNSS. The previous lemma is based on the possibility to remove identical suffixes of paths (see remark 3.10 on page 43).

$\boxed{\textbf{6.25}}$ **Lemma** (IRDS) *Given two terms* s, t, *a precedence* $\succsim$ *and a status function* τ, $s \succ_{\mathsf{IRDS}} t$ *can be determined in time* $O(|s|^4 \cdot |t|^4)$. $\diamond$

In [Rusinowitch, 1987], an upper bound of $O(|s|^4 \cdot |t|^4)$ is stated for the IRD (i.e. the IRDS *without* status).

6.26 **Lemma** (PSDS) *Given two terms* s, t*, a precedence* $\succsim$ *and a status function* τ*,* $s \succ_{\mathsf{PSDS}} t$ *can be determined in time* $O(|s|^4 \cdot |t|^4)$*.* ◇

In [Kapur et al.*, 1985], a method based on* dynamic programming *for analyzing the complexity of syntactic orderings was introduced which was subsequently used by other researchers.*

Chapter 7

Conclusion

In this thesis, essential characteristics of various simplification orderings have been investigated. In this final chapter, we recapitulate the *most* important results and describe how to combine them and make them available for practical applications.

- **Section 7.1:** The essential new results of this thesis will be summarized in this section.
- **Section 7.2:** Here, an abstract specification of the TETRES system is given, a software tool which integrates the approaches given in this thesis.
- **Section 7.3:** Statistical results about the examination of the termination proofs of TRSs can be found in this section.
- **Section 7.4:** Finally, we describe possible future points of interest concerning the field of termination by suggesting extensions of our work, on the one hand, and by listing challenging TRSs whose termination proofs need semantic arguments, on the other.

7.1 Main Results

The most important results *introduced* in this thesis are the following ones.

- **A Formalism for Representing Orderings**
  In order to reduce redundancies and to make definitions of orderings on terms more legible, a formalism for defining orderings has been introduced. The main property of this formalism is its abstraction from the objects that operations are applied to.
- **Path and Decomposition Orderings**

  (i) *Modification of decomposition orderings and comparison with path orderings*: The close connection between the path orderings PSO, KNSS and the various decomposition orderings has been emphasized by modifying the structure of decomposition orderings. To do this we have simplified the data structure of (elementary) decompositions and shifted the thereby lost information into the definition of the orderings.

(ii) *New decomposition orderings*: Three new decomposition orderings have been developed. The DOS combines the good qualities of the RDO with those of the RDOS. The PSDS is an extension of the PSO. The IRDS improves the IRD by incorporating an additional status function.

(iii) *Quasi-precedences for associative path orderings*: We have shown that quasi-precedences on AC-operators are prohibited in order to guarantee the σ-stability of associative path orderings.

(iv) *Extensions of associative path orderings*: The requirements of the associative path condition have been weakened by additionally allowing precedence chains of the form $f \succ g \succ h_1 \succ \cdots \succ h_n$ where f, g are AC-operators and h_i are unary operators (note that the distributivity rules must be enriched). Moreover, this method has further been improved by integrating a status function into the basic ordering. Finally, both extensions have been applied to the strongest path and decomposition ordering, i.e. the basic ordering RPOS has been replaced by the IRDS.

▶ Knuth-Bendix Orderings

(i) *An extension of the Knuth-Bendix ordering*: We have provided an improvement of the KBO in that we weakened the conditions on variables ($\forall x \in \mathcal{X}$: $\geq\langle|.|\rangle$ is allowed instead of $\forall x \in \mathcal{X}$: $=\langle|.|\rangle$ whenever the weights of two terms are identical) and on unary operators (using a quasi-precedence, more than one unary operator with weight zero is allowed on the premise that all these operators are equivalent w.r.t. the precedence). We have also strengthened the KBO by integrating a status function.

(ii) *AC-commuting modifications of the Knuth-Bendix ordering*: First of all, we have introduced a restricted variant of the KBOS based on a weight function, a precedence and a status function satisfying specific limitations (the weight of each A-operator has to be zero, each A-operator must be minimal w.r.t. the precedence and multiset status is necessary for each C-operator). Furthermore, an improvement of this ordering has been possible using the new memory-flattening operation (which allows to assign arbitrary weights to A-operators).

▶ Polynomial Orderings

(i) *Two extensions of polynomial orderings*: We have shown that the original polynomial ordering can be improved such that it will even become a simplification ordering for, for example, identity interpretations of unary operators. This can be achieved by testing whether one of the terms is homeomorphically embedded in the other one, before computing and comparing their interpretations. Another extension of polynomial orderings follows from the integration of a status into the IPOL which requires no more conditions than the IPOL.

(ii) *Methods for proving polynomials to be positive*: We have extended the method of BenCherifa & Lescanne for proving the positiveness of a polynomial. The technique of BenCherifa & Lescanne is incomplete since it is based on a heuristic for choosing appropriate positive and negative monomials. Thus, backtracking is needed whenever the chosen sequence cannot prove the positiveness of the given polynomial. Instead of backtracking, we have introduced a set of constraints to avoid search without giving up the strength of the backtracking approach. Furthermore, another improvement of BenCherifa & Lescanne's technique has been presented which utilizes the well-known arithmetic-mean-geometric-mean inequality. With the help of this method, a polynomial of the form $x^3 + y^3 + z^3 - 3xyz$, for example, can be proved to be positive. Finally, we have provided a modification of the approach for restricted TRSs involving constants in specific positions, only.

(iii) *Polynomial orderings for certain theories*: We have developed restrictions of polynomial orderings which are compatible with some interesting theories including special kinds of distributivity, endomorphism and if-then-else. Furthermore, it has been shown that using any part of the Boolean ring as an underlying theory (and the remaining part as reduction rules) is impossible, if the termination is to be proved with a POL.

(iv) *Automatic generation of polynomial orderings*: An algorithm for choosing appropriate simple-mixed polynomials for a given TRS has been developed. It is based on a transformation of the set of rules into a set of inequalities (by applying complete simple-mixed polynomials with variable coefficients for each operator) each side of which is approximated to a linear polynomial (using a variant of the arithmetic-mean-geometric-mean inequality). Then the method of complete description for deciding whether a system of linear inequalities has a solution possibly helps generating an appropriate polynomial ordering. Note that, for producing a polynomial ordering, it is often necessary to integrate some other (simplification) processes into this algorithm.

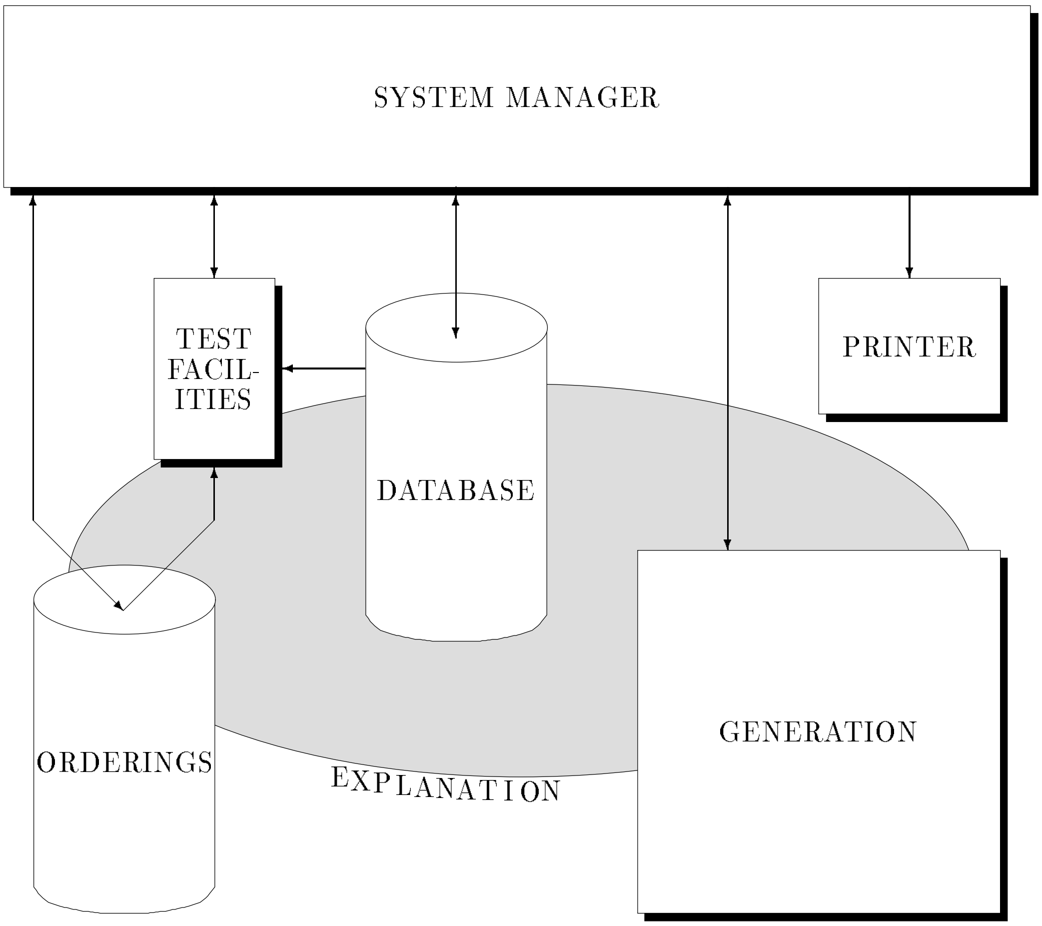
▶ Comparison of the Orderings

(i) *The power of the orderings*: We have compared the power of the orderings presented in this thesis. The power of an ordering is the set of comparable pairs of terms. We did not compare the size of these sets but examined the relation between them. Most of the orderings are based on a precedence $\succ$ and are $\succ$-monotonic. We have shown that one only has to consider total precedences since the results on arbitrary precedences can be reduced to total ones (whenever the ordering is $\succ$-monotonic). The relations between the orderings have been checked on additional premises, namely for the cases of arbitrary terms, ground terms and monadic terms. It has turned out that the IRDS (= KNSS) is the most powerful ordering in the class of path and decomposition orderings and that the POLS is (more or less) stronger than all other polynomial and Knuth-Bendix orderings (for the case of total precedences and arbitrary terms). Since the IRDS and the POLS are incomparable, only two of the nineteen orderings need to be considered: IRDS = KNSS and POLS.

(ii) *The time complexities*: Theoretical time complexities of the worst cases of the most orderings have been investigated. The time for comparing two terms w.r.t. a given ordering has been determined. The following worst case complexities have been proved (s and t are terms to be compared): $O(|s| \cdot |t|)$ for the KBOS (and the RPO), $O(|s|^3 \cdot |t|^3)$ for the PSO and the KNSS, $O(|s|^4 \cdot |t|^4)$ for the PSDS and the IRDS.

7.2 Design of TETRES

In this thesis, we have dealt with various aspects of reduction (simplification) orderings. Our goal is to make this knowledge on orderings available to anybody not familiar with this area. More precisely, the design of a system for proving termination of TRSs will be presented. The system is called TETRES which is an acronym for 'termination expert for term rewriting systems'. A high-level description of the components of TETRES will follow together with references for those parts of the system which have already been implemented. First of all, the overall structure of TETRES will be introduced.



Architecture of TETRES

The **system manager** of TETRES establishes the connection between the system and a user or another system. Essentially, it acquires the necessary (input) data and initializes TETRES. The kernel of TETRES is the so-called **generation** module. It tries to automatically find an appropriate ordering for a given TRS (or a set of equations). A control component in the generation module is used to select the class of orderings (RPOS, KBOS, etc.). It then activates the corresponding technique for generating appropriate parameters (precedence, status function, etc.). The various ordering methods are represented in the **orderings** module which is organized as a database in order to allow the acquisition of new orderings by the user. TETRES also includes some **test facilities** for orderings (time measurements, data for proving the correctness of an implemented ordering and a structure test providing information on the structure of TRSs and orderings) and an **explanation** tool. A **printer** displays the information provided by the **explanation** tool. Finally, examples already dealt with, i.e. TRSs and appropriate orderings, are kept in a **database**.

System Manager

The **system manager** controls each consultation of TETRES by another system or a user. The input variable `kind_of_interaction` specifies whether an (external) system or a user wants to consult TETRES. It may adopt the values 'user-driven' or 'system-driven'. If another system calls TETRES, an appropriate *data structure transformation* may have to be performed. If `kind_of_interaction` is set to 'user-driven', the following actions can be executed:

- ▶ Input a TRS (by hand or from the **database**) for proving its termination: The *input requirements* are a `signature` (i.e. operators and their arities), an `AC-operator_set` (containing all operators which should be associative and commutative), a set of `pairs_of_terms` and an `orientation_flag` (which determines how to consider the set of `pairs_of_terms`, either as equations or as rules[1]), an `ordering` (consisting of specific parameters that are possibly 'empty'), a `generation_flag` (indicating whether or not an automatic termination proof is required), an `exception_string` (determining the kind of action that must be performed when the termination proof fails or the orientation can be done in several ways) and a value for the variable `trace_level` (i.e. controlling the amount of explanation output). Values for the variables `signature`, `AC-operator_set`, `pairs_of_terms` and `ordering` can be loaded from the **database**. A *parser* transforms the database format into an adequate internal representation. Another part of the *parser* can generate a database entry for an input by the user or from an external system. If the value of the `generation_flag` is 'automatic', the **generation** module will be called. Otherwise, the given ordering of the **orderings** database is used to orient the `pairs_of_terms` or to try proving their termination (depending on the variable `orientation_flag`).
- ▶ Call one of the **test facilities**: It is possible to check some properties of TRSs and orderings.
- ▶ Extend the **orderings** database: All orderings available are stored in the **orderings** database. The *implementation* submodule is part of the **system manager** and provides the user with the opportunity to increase the **orderings** database.

  In section 2.2, a formalism for defining orderings is given. With its simple components this formalism allow us to easily develop a compiler for translating an ordering into internal code. In case of functional programming languages such as LISP, this compiler can be implemented straightforwardly in the following way:

  - The basic elements of the formalism (such as $\mathcal{H}ead, \mathcal{A}rgs, \langle x,y\rangle, \langle x\rangle$, lexicographic evaluation, multiset and lexicographic extension, etc.) are implemented as functions with appropriate input arguments. For example, $\mathcal{H}ead$ is transformed into '(defun head (term) (car term))' if a term is represented as a list where the first element specifies the $\mathcal{H}ead$ and the other elements are the arguments of the term.

---

[1] `orientation_flag` may possess the values 'no-matter' and 'left-to-right'.

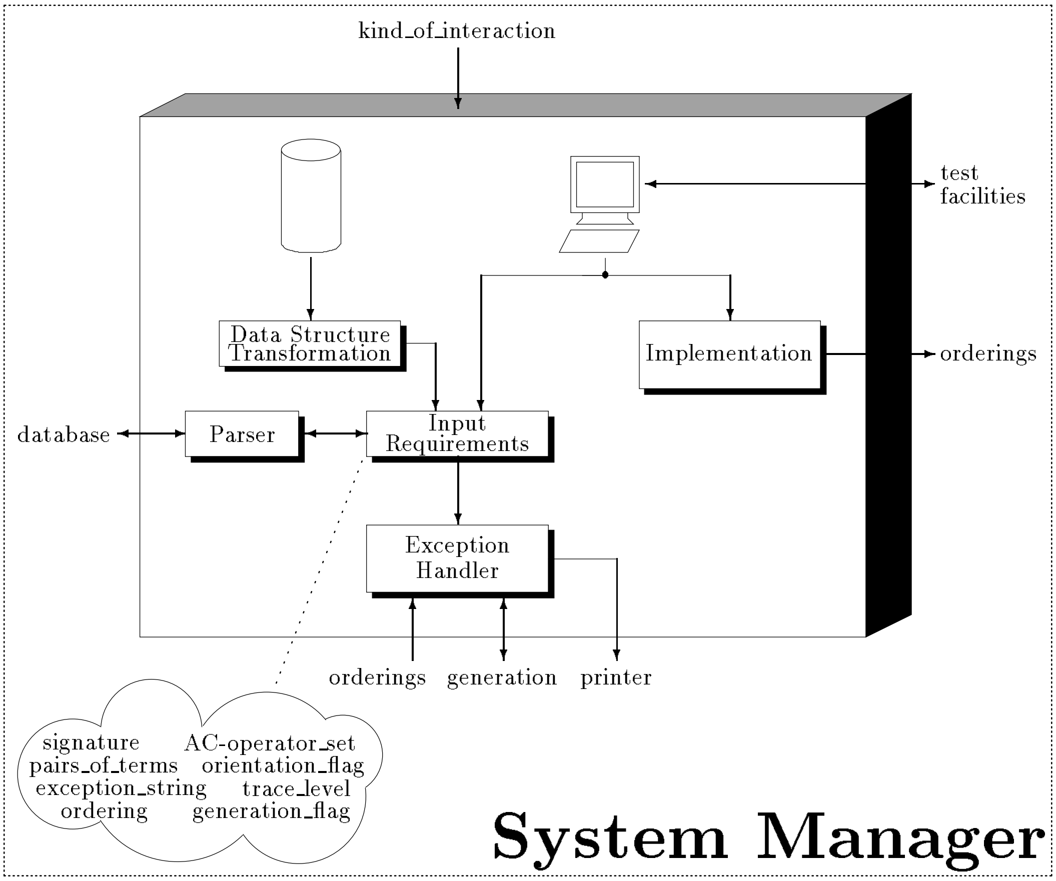
- A complete definition of an ordering is translated by applying the basic functions to corresponding arguments. For instance, consider example 2.15 on page 33:

  $$\succ_{\mathrm{T}} \quad \equiv \quad \supseteq\langle \mathcal{V}ar \rangle \quad \& \quad (>\langle|.|\rangle, \succ, \succ_{\mathrm{T}}^{\textcircled{1}})$$

  This definition corresponds to the following LISP code:

  ```
  (defun ordT (x y)
         (and (supersetp (vars x) (vars y))
              (lex-eval (> (len x) (len y))
                        (root-order x y)
                        (l-ext 'ordT 'ordT-eq (args x) (args y)))))
  ```

  Note that additional operations such as $\supseteq$ (i.e. supersetp), $\mathcal{V}ar$ (i.e. vars) and $|.|$ (i.e. len) must be available. The functions lex-eval, root-order and l-ext describe the lexicographic evaluation, the root ordering ($\mathcal{H}ead(x) \succ \mathcal{H}ead(y)$) and the lexicographic extension of orderings, respectively. The expression 'ordT-eq represents $\sim_{\mathrm{T}}$ which is defined (in an abstract form) by the user.

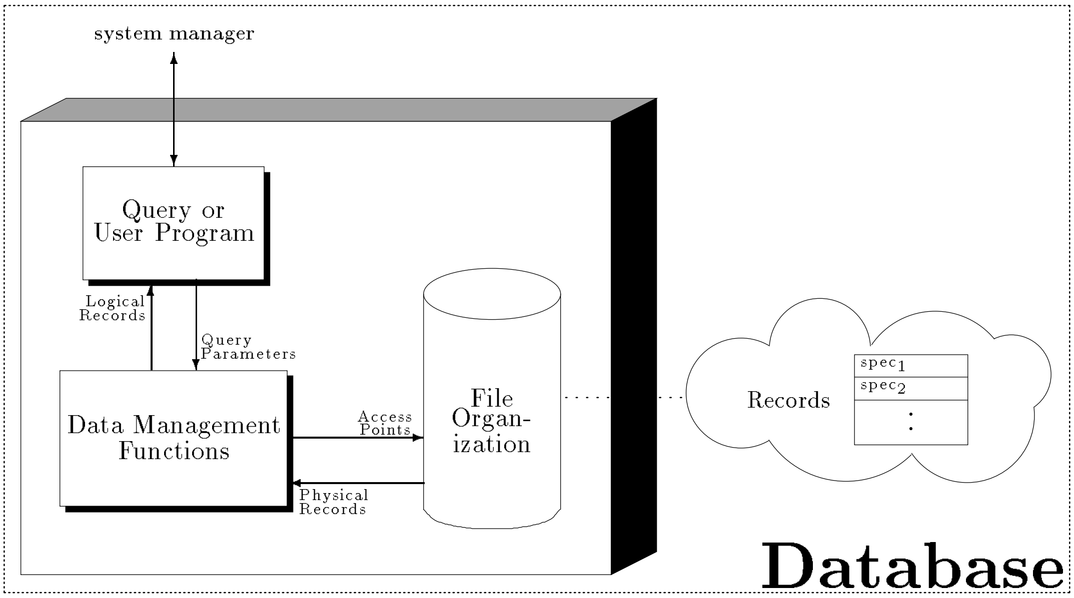


In case a termination proof is requested (independent of the variables `kind_of_interaction`, `orientation_flag` and `generation_flag`), an *exception handler* will be called. This tool is in charge of situations arising from a failed termination proof of a TRS or from a set of equations that can be oriented in several ways. If the variable `exception_string` is set to 'user', the user is allowed to make suggestions (to guarantee the termination or to treat the TRS as a system of equations respectively to select an oriented equation set). If the value of both the `exception_string` and the `orientation_flag` is 'don't care', the *exception handler* chooses any of the oriented equation sets. Other assignments are possible.

■ **Implemented:** There exists an implementation of the *parser*. For sane information on the grammar the *parser* accepts, see the description of the **database**. ■

Database

The **database** is the repository for all data of interest to the user. The queries, which are stored as *query programs*, represent the user of the system and invoke the desired actions. *File organization* refers to the physical and logical relationships of data. The *data management functions* represent the programs in the system necessary for addition, deletion, update, retrieval and space management.



Our **database** contains specifications together with a lot of other information. Above all, different orderings are allowed for the same specification. The syntax of a **database** entry is omitted here (see [Eschbach, 1993] for details). Instead, the following example should illustrate the *most* important parts of a **database** entry.

```
•  specification illustration
•  begin
•
•      sorts GROUP
•      functions
•        0 :  -> GROUP
•        f :  GROUP -> GROUP
•        _ + _ :  GROUP # GROUP -> GROUP
•
•      equations
•        f((x + 0)) == f(x) ;
•        ((x + y) + z) == (x + (y + z))
•
•      ordering Lankford-Manna-Ness !
•      interpretation
•        [+](x,y) = x + 2y
```

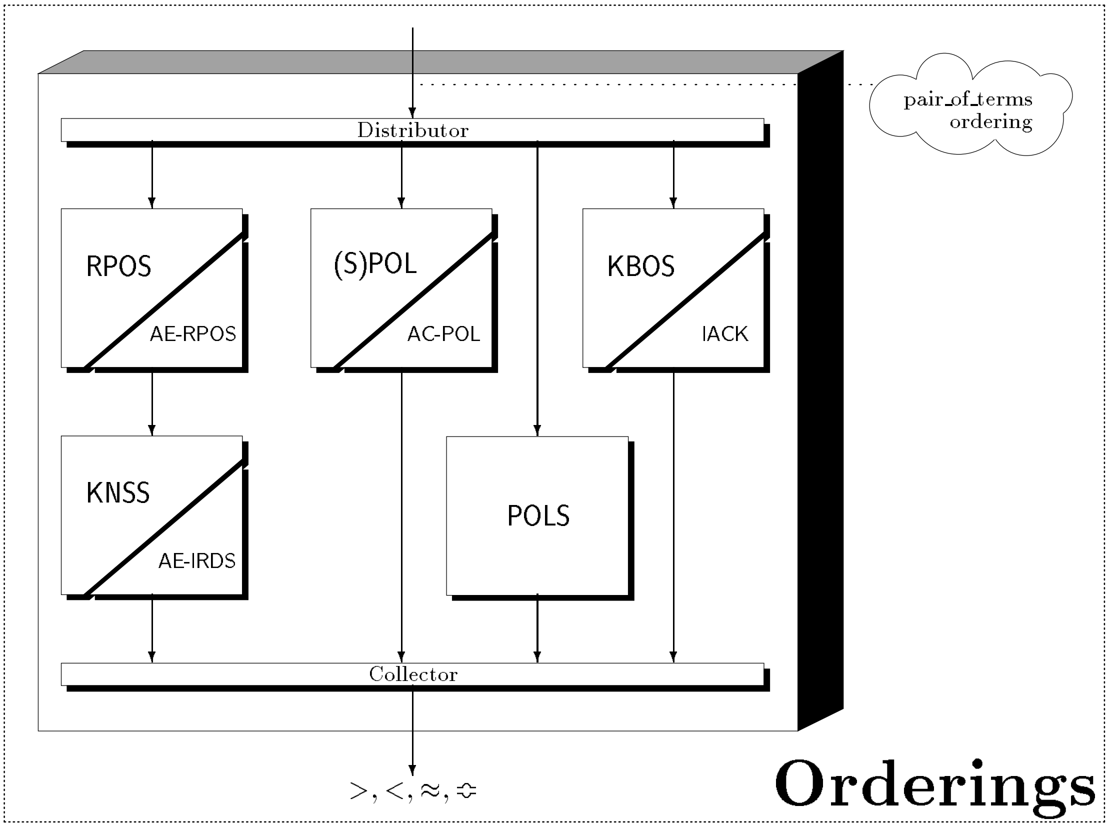
```
•        [f](x) = x
•        [0]() = 1
•        variable bound :  1
•        approach :  Rouyer
•
•      ordering Knuth-Bendix
•      status
•        0 f + :  left
•      weight
•        0 f + :  1
•
•      ordering Dershowitz
•      status
•        0 f + :  left
•
•  end
```

Note that the symbol '!' in the specification given above selects an ordering. Furthermore, 'Rouyer' stands for the approach of [Rouyer, 1991] for proving the positiveness of a polynomial (see historical remark on page 102). We have provided an additional slot named 'rules' for each ordering. Currently, it contains the canonical TRS (if it exists), generated by the corresponding ordering and equivalent to the initial set of equations.

■ **Implemented:** We have installed the relational database management system POSTGRES (which is a successor of INGRES) and used it to some extent for storing specifications and their statistics[2)]. The major design goals – from our point of view – have been (i) to support complex objects each of which is stored in one record so that only one query is required to retrieve it, (ii) to allow new data types, new operators and new access methods to be included in the database management system and (iii) to make as few changes to the relational model as possible, i.e. rather than building a system that is based on a large, complex data model, a new system is built on a small, simple model that is extendible. The parser is described in [Eschbach, 1993]. ■

Orderings

The **orderings** database contains all ordering techniques used in TETRES. It would suffice to provide TETRES only with the strongest orderings. Due to practical considerations, however, we have also made weaker, but more efficient versions of them available. The diagram given below adheres to this strategy: the KNSS and the POLS are (more or less) the strongest orderings. Provided with the same parameters, the RPOS is more efficient than – as well as included in – the KNSS. Therefore, the RPOS serves as a pre-processor for the KNSS. The SPOL and the KBOS are weaker than the



[2)] The contents of the database are derived from the internal representation of the specifications and the statistics files generated by the test facilities.

POLS; contrary to the POLS, however, they are more efficient and there exist AC-versions of it. Obviously, TETRES will include an implementation of an AC-variant of each ordering as soon as one will be available.

Each ordering technique only checks whether a term s is greater than a term t. Consequently, if the test $s > t$ returns a negative result, we still have to check whether $t > s$ holds[3]. This is also done in this module.

The *distributor* calls the right ordering depending on the input which is a `pair_of_terms` and an `ordering` (defined by its global scheme and its parameters)[4]. The *collector* receives the result $(>, <, \approx, \Bumpeq)$ from the activated ordering and returns it to the calling environment.

■ **Implemented:** All orderings depicted in the above diagram are implemented. ■

Generation

The **generation** module makes up the heart of TETRES. Its task is to automatically generate an ordering that proves the termination of an input TRS. The **system manager** provides a set of `pairs_of_terms` and the `orientation_flag`. If the `orientation_flag` has the value 'left-to-right', an ordering must be found which proves the termination of the set of `pairs_of_terms` where each pair (s,t) is regarded as a rule $s \to t$. Otherwise, an ordering has to be generated which orients the set of `pairs_of_terms` regardless of the order in which the terms of a pair are given. Obviously, the latter problem can be reduced to the former one. We will therefore assume that we have to show that a TRS $\mathcal{R}$ is terminating.

The *simplification* submodule applies techniques for reducing the complexity of the structure of a TRS to the given TRS $\mathcal{R}$. With the effect that both the process of specifying an appropriate ordering and the structure of the ordering are possibly simpler than without this simplification. Currently, the *simplification* submodule will remove every rule $l \to r$ from $\mathcal{R}$ if $l \succ_{\mathsf{EMB}} r$ holds. However, it is also possible to use transformation orderings[5], semantic labelling (see (iii) on page 177) or similar techniques.

The simplified version of $\mathcal{R}$ serves as the input for the *control* unit. This component manages the (local) *generation* techniques for the path orderings, the KBOS and the POL. More precisely, the *control* unit has to select one of these methods. So-called sensors help solving this complex problem. We use two different kinds of sensors:

- ▶ Negative sensors, also called *critical rules*: They consist of single rules or sets of rules the termination of which *cannot* be proved with a specific ordering. We give some examples using capital letters to denote higher-order variables (i.e. X represents an operator or a first-order variable):

---

[3] Obviously, the orderings can also be implemented such that they directly return $>, <, \approx$ or $\Bumpeq$. However, the automatic translation of orderings given by the user into internal code would be too complex.

[4] The distributor is necessary since the orderings depend on different parameters.

[5] In this case, the TRS $S \cup T$ represents the simplified system (see (ii) on page 176).

- *General critical rules*:

No.	rule scheme	conditions
1[6]	$s \to t$	$t \succ_{\mathsf{EMB}} s$
2	$s \to t$	$\mathcal{V}ar(t) \supset \mathcal{V}ar(s)$

- *Critical rules for the* RPOS:

No.	rule scheme	conditions
3	$X(X(x,y),z) \to X(x,X(y,z))$ $X(s_1,s_2) \to X(t_1,t_2)$	$\exists i$: $\mathcal{V}ar(t_i) \supset \mathcal{V}ar(s_i)$, $\forall j < i$: $s_j \approx t_j$
4	$X(s_1,\dots,s_n) \to Y(t_1,\dots,t_m)$ $Y(Z_1,\dots,Z_m) \to X(r_1,\dots,r_n)$	$\exists i$: $r_i = W(w_1,\dots,w_k) \wedge \mathcal{V}ar(r_i) \neq \emptyset \wedge \{f \mid f \in \mathcal{S}ym(s_j), f \notin \mathcal{X}\} \subseteq \{f \mid f \in \mathcal{S}ym(r_j), f \notin \mathcal{X}\}$

- *Critical rules for the* KBOS:

No.	rule scheme	conditions
5	$s \to t$	$\exists x \in \mathcal{X}$: $\lvert s\rvert_x < \lvert t\rvert_x$
6	$s \to t$	$\mathcal{S}ym(s) \subset \mathcal{S}ym(t) \wedge$ $\{\!\{ \Delta \mid \Delta \in \mathcal{S}ym(s), \mathcal{A}r(\Delta) = 0 \}\!\} \subset$ $\{\!\{ \Delta \mid \Delta \in \mathcal{S}ym(t), \mathcal{A}r(\Delta) = 0 \}\!\}$
7	$s = Y(s_1,\dots,s_n) \to X(t_1) = t$	$n > 1$, $\mathcal{S}ym(s) \subset \mathcal{S}ym(t)$, $\lvert s\rvert_X < \lvert t\rvert_X$

- *Critical rules for the* POL:

No.	rule scheme	conditions
8	$X(s_1,s_2) \to X(t_1,X(t_2,t_3))$	• $s_1 = t_1 \wedge \mathcal{V}ar(X(t_2,t_3)) \setminus \mathcal{V}ar(s_2) \neq \emptyset$ or • $s_1 = t_2 \wedge \mathcal{V}ar(X(t_1,t_3)) \setminus \mathcal{V}ar(s_2) \neq \emptyset$ or • $s_2 = t_2 \wedge X(t_1,t_3) \succ_{\mathsf{EMB}} s_1$ $\wedge\, \mathcal{V}ar(X(t_1,t_3)) \cap \mathcal{V}ar(s_2) = \emptyset$
9	$s = X(s_1,\dots,s_k) \to t$	$\lvert s\rvert_X = 1 \quad \wedge$ $\forall g \in \mathcal{S}ym(s) \setminus \{X\}$: $[\![g]\!](x_1,\dots,x_m)$ is superlinear $\wedge$ $\exists u,v \in \mathcal{P}os(t), \exists x \in \mathcal{X}, \forall y \in \mathcal{X} \setminus \{x\}$: $\mathcal{H}ead(t\vert_u) = X \wedge [\![\mathcal{H}ead(t\vert_v)]\!](x_1,\dots,x_n)$ is strictly mixed (or strictly separate if $n = 1$) $\wedge\ w = min_{>}(u,v) \wedge \lvert s\rvert_x < \lvert t\vert_w\rvert_x \wedge \lvert s\rvert_y \leq \lvert t\vert_w\rvert_y$

In order to illustrate the practical relevance of the negative sensors, some rules will be given each of which is 'detected' by one of the negative sensors. For example, the associativity and endomorphism TRS $\{(x+y)+z \to x+(y+z), f(x)+(f(y)+z) \to f(x+y)+z\}$ is an instance of no. 3, and due to no. 4, the termination of the TRS $\{del(x.(y.z)) \to f(x \equiv y,x,y,z), f(true,x,y,z) \to del(y.z)\}$ of [Mohan and Srivas, 1986] cannot be proved with an RPOS either. The A.12 A.29

---

[6] Whenever orderings are part of the **orderings** database and are able to prove the termination of the rule $s \to t$, this critical rule (no. 1) must be removed.

distributivity rule $x \star (y+z) \rightarrow (x \star y) + (x \star z)$ fits no. 5, $rev(x.y) \rightarrow rev(y) \circ (x.nil)$ is an instance of no. 6 and $f(x) + y \rightarrow f(x + f(y))$ an instance of no. 7. The second condition of rule scheme no. 8 applies to the Ackermann rule $ack(s(x), s(y)) \rightarrow ack(x, ack(s(x), y))$. The last condition of rule scheme no. 8 fits the rule $(x/y)/z \rightarrow x/(z/i(y))$. The ninth critical rule is somewhat technical. Therefore, we provide a detailed 'proof' of the fact that it 'matches' with, for example, the rule $s = exp(x, s(y)) \rightarrow x \star exp(x, y) = t$, the recursive case in the definition of the exponential function. The leading operator *exp* appears only once in the left-hand side: $|s|_{exp} = 1$. All operators in s except the leading operator *exp* must be interpreted as super-linear polynomials: $[\![s]\!](x)$ has to be super-linear since it occurs in the addition rule $x + s(y) \rightarrow s(x+y)$[7]. Let $u = 2$, $v = \Lambda$ and $w = \Lambda$. Then $\mathcal{H}ead(t|_u) = exp$, $\mathcal{H}ead(t|_\Lambda) = \star$ and $[\![\star]\!](x, y)$ must be strictly mixed[8]. Furthermore, $|s|_x = 1 < 2 = |t|_\Lambda|_x \wedge |s|_y = 1 \leq 1 = |t|_\Lambda|_y$. This concludes the proof.

The collection of negative sensors can automatically be extended whenever the system detects the non-orientability of an equation, i.e. a rule will be added to the critical rules if its termination cannot be proved with a specific ordering[9]. Note that it would be best to add more general critical rules to the system in order to reduce the search space.

- Positive sensors, also called *heuristics*: These conditions indicate that a certain (class of) ordering(s) should be preferred to another. We give some examples:

  - *Choosing the* RPOS:
    Let $Sl_i = \{f \mid f \in \mathcal{S}ym(l_i), f \notin \mathcal{S}ym(r_i), f \notin \mathcal{X}\}$ and $Sr_i = \{f \mid f \in \mathcal{S}ym(r_i), f \notin \mathcal{S}ym(l_i), f \notin \mathcal{X}\}$ for each rule $l_i \rightarrow r_i$. An appropriate condition for trying to generate an RPOS is the following one:

$$\begin{array}{rl} & \exists i, j, f, g\colon i \neq j \,,\, f \not\succ g \,,\, f \in Sr_i \cap Sl_j \,,\, g \in Sl_i \cap Sr_j \\ \leadsto & [\exists h \in Sl_i\colon h \notin Sr_j \ \wedge\ \nexists k\colon f \in Sl_k \ \wedge\ h \in Sr_k] \ \ \vee \\ & [\exists h \in Sl_j\colon h \notin Sr_i \ \wedge\ \nexists k\colon g \in Sl_k \ \wedge\ h \in Sr_k] \end{array}$$

    This condition checks whether there exist mutually recursive operators. If two rules include such operators, another operator must occur on the (presumably) greater side that covers one of the recursive operators. For example, the operators f and g are mutually recursive w.r.t. the TRS $\{g(h(x)) \rightarrow f(x)\,,\, f(x) \rightarrow g(i(x))\}$. The operator h helps proving the termination of the first rule if $h \succ f$ belongs to the precedence. Thus, the relation $f \succ g$ (and $f \succ i$) should be added to the precedence in order to show the termination of the second rule.
    Note that it is very easy to develop a simple strategy for generating an appropriate RPOS for rule systems defining inductive theories ($\mathcal{R}$ is then confluent and totally defined).

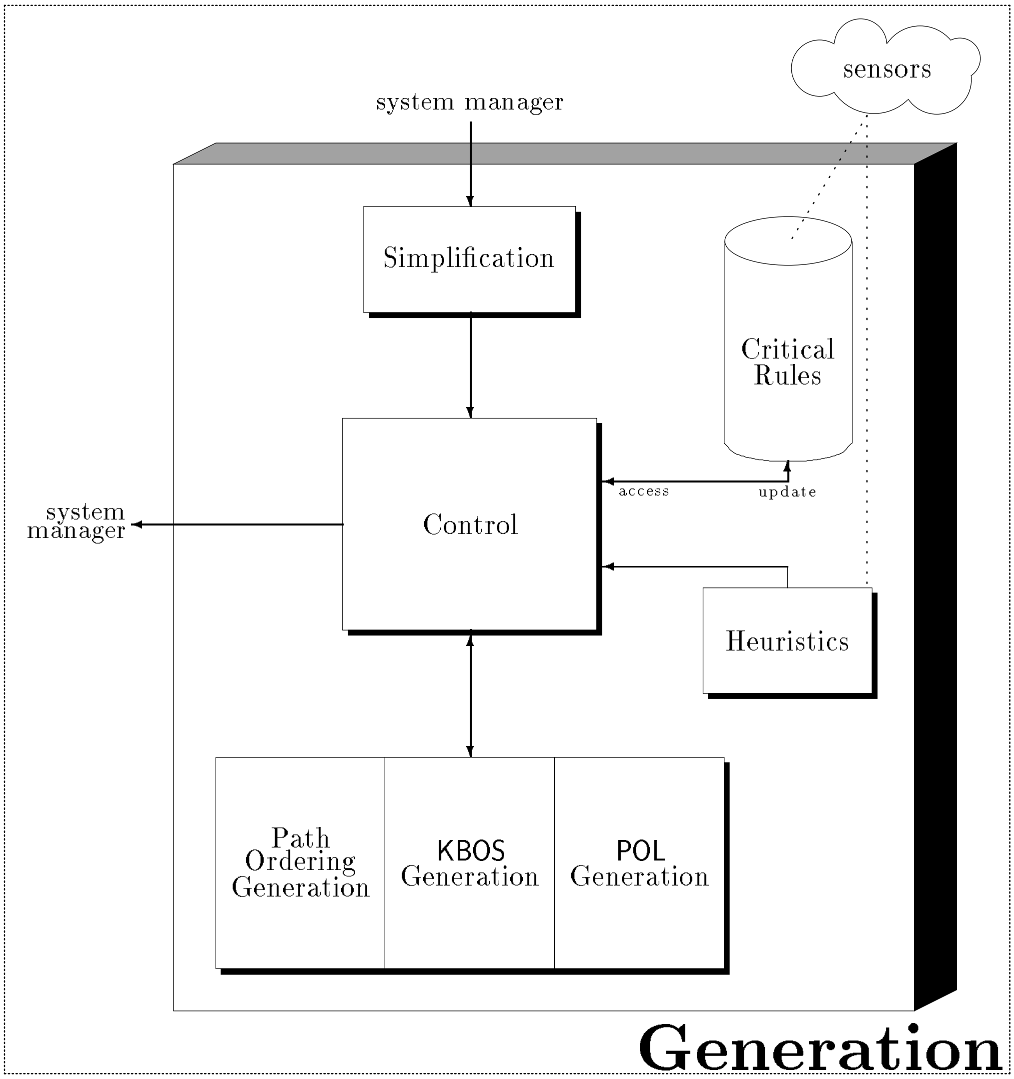
---

[7] A simple heuristic like '$[\![i]\!](x)$ must be super-linear if, and only if, there exists a rule $l \rightarrow i(r)$ such that $\mathcal{S}ym(l) \subseteq \mathcal{S}ym(i(r))$, $\mathcal{V}ar(l) \neq \emptyset$ and $\mathcal{A}r(\mathcal{H}ead(l)) \geq 2$' is of help.

[8] since the multiplication rule $x \star s(y) \rightarrow (x \star y) + x$ is part of the specification and a similar heuristic as that for super-linear polynomials can be generated

[9] This feature is closely connected with learning theory.

- *Choosing the* KBOS:
  $\forall l \to r$: $\forall x \in \mathcal{X}$: $|l|_x \geq |r|_x$, i.e. the variable condition of the KBOS is fulfilled for each rule.
- *Choosing the* POL:
  $\exists l \to r, l' \to r'$: $\mathcal{S}ym(r') \ni \mathcal{H}ead(l) \neq \mathcal{H}ead(l') \in \mathcal{S}ym(r)$, i.e. the operators $\mathcal{H}ead(l)$ and $\mathcal{H}ead(l')$ are mutually recursive.



If (both negative and positive) sensors do not suggest exactly one generation method, one has to apply all techniques one after another until an ordering has been generated or all approaches have failed. We propose the following order of succession: 1. KBOS generation, 2. path ordering generation and 3. POL generation. This is motivated by the following two facts: (i) the KBOS often fails because of the variable condition (and there exists a sensor that checks this property), (ii) in contrast to the POL generation, the path ordering generation is a decision procedure.

Obviously, it is possible to use data from already oriented systems. In that case the *control* unit needs direct access to the database consisting of special reduction rules including information on their orientation w.r.t. the several orderings.

■ **Implemented:** The generation techniques for path orderings, Knuth-Bendix orderings and polynomial orderings have already been implemented. Documentation can be found in [Nessel, 1993] (for path orderings) and in [Sprenger, 1992] (for Knuth-Bendix and polynomial orderings). ■

Test Facilities

The module **test facilities** consists of three independent test environments which the user can call.

- ► **I-test**: The *implementation test* tries to find out whether an implementation of a new ordering or a new implementation of an already existing ordering is 'correct' in the sense that various comparisons are performed and these new results checked against the correct results previously stored in the **database**. The user has to select an ordering to be checked.

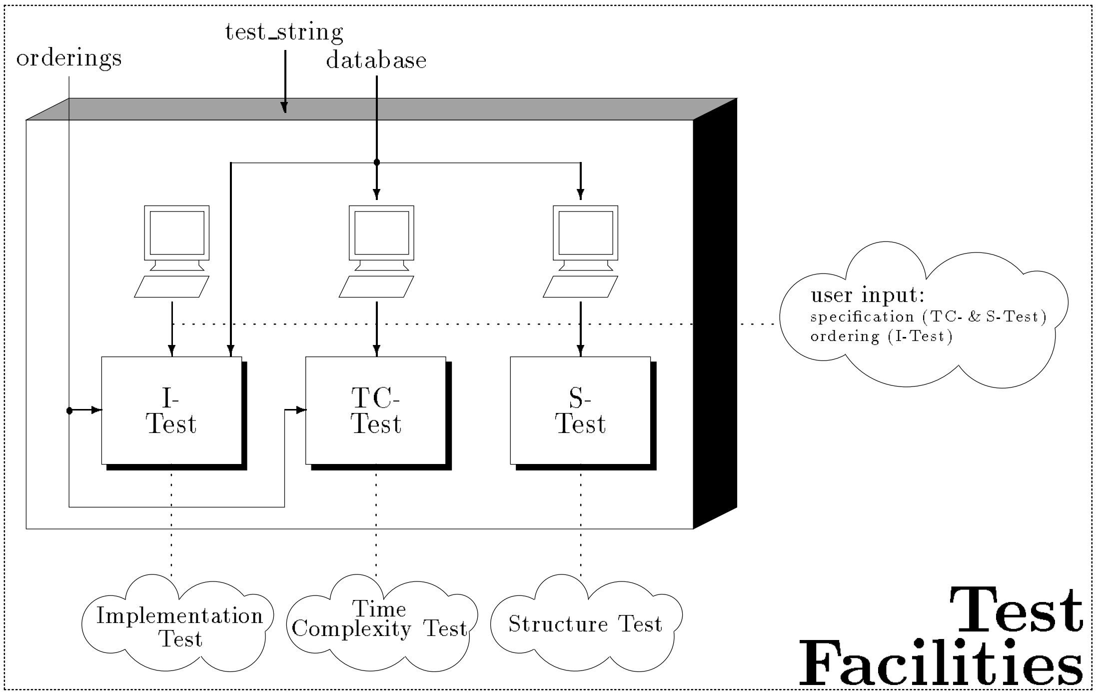
- ► **TC-test**: The *time complexity test* computes the empirical time complexity of a series of comparisons of two terms w.r.t. the orderings implemented. The most interesting question concerns the comparison of the time complexities of different orderings. Thus, the user has to select a set of orderings and a set of equations that can be oriented in the same way w.r.t. each ordering. It is also possible to leave the choice of the set of equations to the system, i.e. the system extracts all specifications from the **database** which are appropriate for the specific orderings.

  The test includes the positive test ($s > t$ has been conjectured and really $s > t$ holds), the negative test, both tests and the time for the positive test divided by the time for the negative test. A graphical representation of these four results (by transforming the results into figures which are described in LaTeX) enhances the implementation test.

- ► **S-test**: The *structure test* checks various properties of TRSs and orderings and is applied to a set of database entries selected by the user. A few of the more interesting properties that can be tested are the following ones:

  - Average size and depth of left- and right-hand sides
  - Average number of different variables per rule
  - Average number of trivial rules ($l \rightarrow r$ such that $l \succ_{\mathsf{EMB}} r$) per system
  - Average number of non-linear left- and right-hand sides per system
  - Average number of directly (mutually) recursive operators per system
  - Systems oriented using strictly linear (separate, mixed) polynomials
  - Average number of unary operators with weight zero (in case of the KBOS)
  - Average number of operators with more than one argument with lexicographic status which do not occur in an associative rule (in case of path and decomposition orderings)

  For a test, the user may choose any subset of the set of all (currently about 50) properties (see also section 7.3).



■ **Implemented:** The structure test is implemented and described in [Kirchberg, 1992]. This system is also able to produce an output in form of a LaTeX file. A prototype of the time complexity test (without LaTeX output) also exists. ■

Explanation

One important aspect of the dialogue between a user and TETRES is the system's explanation to a user of its steps taken during a specific consultation. A clear and understandable explanation can be a valuable means of justifying the recommendations of the system and of indicating its limitations to the user.

The user should be able to comprehend the system by asking specific questions like (i) Why does TETRES choose this ordering? and (ii) Which rule (equation) is responsible for the exclusion of this ordering?

A system utility that allows the user to obtain a detailed description of the behaviour of the system is the trace facility. This facility is of considerable value as it enables the user to visualize all steps carried out, at various levels of detail. The level of detail can usually be adjusted to specific needs.

The explanation component can be invoked by the user both during and at the end of a consultation. It is active during a consultation if the user has replied positively to a certain system request. However, it can also be activated at the end of a consultation, provided the user requested explanations to be given later at the beginning of the session. The explanation tool collects and saves all information produced during a consultation. The **printer** displays (parts of) this information.

Obviously, the information saved is tool-dependent, for example,

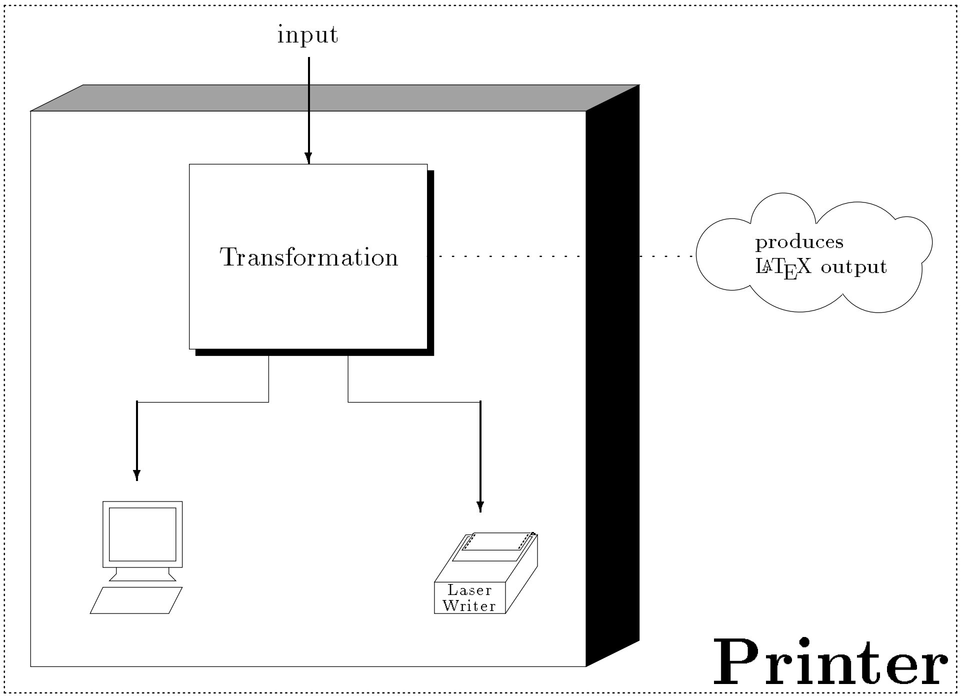
- in case of **orderings**, the explanation component keeps the sequence of term comparisons needed for comparing two terms (when recursive path orderings are used) or the interpretations of terms and the sequence of polynomials generated by the BL-approach (when polynomial orderings are applied),

- in case of **generation**, the progressive refinement of an initially empty registry (for path and decomposition orderings) or the simplification steps and the set of inequalities which lead to the final interpretations (for polynomial orderings) are important among others and
- in case of **test facilities** and **database**, the corresponding necessary information to save is obvious.

■ **Implemented:** An implementation of an explanation tool for the automatic generation of polynomial orderings, including a trace facility with six levels, is described in [Sprenger, 1992]. ■

Printer

In order to display the data collected during a consultation, a special output format is used: The *transformation* submodule transforms the data into a LaTeX document. The output can then be displayed using 'xdvi' or sent to a laserwriter.



Obviously, the internal formats of the data collected by the **explanation** tool are different. Thus, the **printer** must be able to handle these formats.

■ **Implemented:** There already exists an implementation of a transformation part generating a LaTeX file from the structure test module ([Kirchberg, 1992]). It also includes a graphical illustration of specific statistical results. ■

7.3 Statistics on Examples

Various methods for proving the termination of TRSs have been discussed in this thesis. The preceding section contains an abstract description of TETRES, a system comprising these methods. To evaluate the system's performance, i.e. to examine the strength of the orderings implemented, they should be applied to a significant number of examples often

Summary Statistic	
Number of rule systems tested	257
Number of constant operators	1.42
Number of unary operators	2.02
Number of binary operators	1.53
Number of remaining operators	.14
Number of rules	4.62
Number of variable-reducing rules	.927
Number of non-linear left-hand sides	.698
Number of non-linear right-hand sides	.232
Number of trivial rules	2.1
Number of complicated rules	.038
Number of directly recursive rules	1.2
Number of directly recursive operators	1.89
Number of indirectly recursive operators	.253
Number of quasi-trivial rules	.736
Number of associative rules	.228
Number of endomorphism rules	.083
Number of distributivity rules	.025
Average size of left-hand side	3.88
Average size of right-hand side	2.9
Average depth of left-hand side	3.13
Average depth of right-hand side	2.41
Average number of variables per rule	2.83
Average number of different variables per rule	1.38
Canonical rule systems generated	200
Confluent initial rule systems	100
Diverging completion processes	25
Initial orientable rule systems	225
Completion processes that stopped with failure	32
Non-simplifying rule systems with no appropriate ordering	11
Number of appropriate **polynomial orderings** given	68.9%
Number of orderings based on strictly super-linear polynomials	15.8%
Number of orderings based on strictly linear polynomials	52.0%
Number of orderings based on strictly separate polynomials	6.2%
Number of orderings based on strictly mixed polynomials	5.1%
Number of orderings based on simple-mixed polynomials	93.8%
Quota of mixed interpretations for directly recursive operators	1.1%
Quota of mixed interpretations for indirectly recursive operators	1.6%
Quota of super-linear interpretations for operators in quasi-trivial rules	44.8%
Quota of linear interpretations for operators in quasi-trivial rules	82.8%
Quota of operators without a constant monomial	48.7%
Number of appropriate **Knuth-Bendix orderings** given	43.6%
Number of orderings which require a precedence	44.6%
Quota of unary operators with weight zero	21.6%
Quota of unary operators with weight zero included in the precedence	74.1%
Quota of operators with weight zero	13.7%
Quota of operators with weight greater than 2	5.6%
Number of appropriate **path and decomposition orderings** given	71.6%
Quota of operators with lexicographic status not in an associative rule	32%
Quota of operators which are greater than an operator $\mathcal{F}_1$	3.7%

Figure 7.1: Statistics on Examples (Part I)

found in practice. Therefore, we have collected and investigated more than 250 examples. They belong to several different classes including TRSs for arithmetic theories and lists as well as string rewriting systems. We have used two criteria for selecting these systems:

- We have examined systems which often appear in the literature.

number of specifications ordered w.r.t.			
polynomial orderings	knuth-bendix orderings	path and dec. orderings	
×	×	×	98
	×	×	1
×		×	65
×	×		11
		×	19
	×		2
×			2
			59

Figure 7.2: Statistics on Examples (Part II)

- The remaining systems are interesting from a termination point of view, i.e. the orderings cannot be straightforwardly applied in order to prove the termination.

Parts of these examples can be found in appendix A and in [Steinbach and Kühler, 1990].

Since the implementation of TETRES has not been completed yet, we have tested the TRSs with our completion environment COMTES (see [Avenhaus *et al.*, 1989]) in which all presented orderings have been integrated. Detailed statistical results of the tests are given in tabular form in figure 7.1. This table has been generated by the S-module (described on page 169). Since it includes a great amount of data with complex interrelations between the different information, an interpretation of the table is omitted. However, some brief information for understanding the criteria will be given:

> The *depth* $||t||$ of a term $t = f(t_1,...,t_n)$ is 1 if t is an atom and $1 + max\{||t_i|| \mid i \in [1,n]\}$ otherwise. An operator f is *directly recursive* if $\mathcal{A}r(f) > 0$ and $\exists l \to r$: $l \not\succ_{\mathsf{EMB}} r \wedge f \in \mathcal{S}ym(l) \cap \mathcal{S}ym(r)$. An operator f is *indirectly*[10] *recursive* if $\mathcal{A}r(f) > 0$ and $\exists \{g_1,...,g_n\} \subseteq \mathcal{F}$: $g_1 = f = g_n \wedge \forall i \in [1, n-1]$: $\exists l \to r$: $g_i \in \mathcal{S}ym(l) \wedge g_{i+1} \in \mathcal{S}ym(r)$. A rule $l \to r$ is called *directly recursive* if $l \not\succ_{\mathsf{EMB}} r$ and $\exists f \in \mathcal{F}$: $\mathcal{A}r(f) > 0 \wedge f \in \mathcal{S}ym(l) \cap \mathcal{S}ym(r)$. A rule $l \to r$ is called *trivial* if $l \succ_{\mathsf{EMB}} r$ holds. A rule $l \to r$ is called *complicated* if l and r are non-linear. A rule $l \to r$ is *quasi-trivial* if $l \not\succ_{\mathsf{EMB}} r$ holds and there are no (in-) directly recursive operators in l. A *non-simplifying* TRS contains at least one rule $l \to r$ such that $r \succ_{\mathsf{EMB}} l$. The restricted set $\{f \mid \mathcal{A}r(f) \geq i\}$ of operators is denoted by $F_{\geq i}$.
>
> The first row of each ordering statistics in figure 7.1 contains the number of successful termination proofs w.r.t. the ordering (in %). The remaining rows are related to the first row.

In figure 7.2, the number of ordered TRSs w.r.t. unions of ordering classes are given. For example, the third row means that there exist 66 TRSs which can be ordered either w.r.t. polynomial orderings or w.r.t. path and decomposition orderings but *not* w.r.t. Knuth-Bendix orderings.

---

[10] also called *mutually*

7.4 Future Prospects

This section deals with an obviously incomplete list of possible future work in the field of termination.

▶ **Direct extensions (applications) of the results given in this thesis:**

(i) A *generalization of the status function* such that equivalent (w.r.t. the precedence) but syntactically different operators may possess arbitrary types of status[11] renders possible the orientation of interesting examples. However, it is somewhat tedious to specify how to then compare the arguments of two terms with equivalent leading operators (which have lexicographic and multiset status, respectively). Furthermore, the ordering resulting from an integration of this kind of status into a simplification ordering must no longer be a simplification ordering.

(ii) The development of an *AC-compatible ground-total and precedence-based ordering* could be achieved by weakening the precedence conditions of the associative path orderings[12].

(iii) In order to improve the *Knuth-Bendix orderings suitable for AC-theories*, the restrictions on the precedence must be weakened.

(iv) An interesting problem concerns a *combination of the orderings* presented in this thesis. In [Dershowitz, 1991] (see also [Dershowitz and Hoot, 1993]), one way of composing orderings is provided in form of a general framework based on different termination functions. The shape or surface of this so-called *general path ordering* is a restricted recursive path ordering (without the case of two operators being equivalent). The recursive application of this ordering to arguments is replaced by a lexicographic comparison of two vectors of (different termination) functions applied to the terms to be compared. Possible values of such functions are, for example, a precedence (applied to the leading operators), a polynomial interpretation, a weight function or the extraction of a specified argument. Thus, the general path ordering is able to merge different orderings (by guaranteeing some minor properties of the termination functions). In [Dershowitz and Hoot, 1993], parameters for the general path ordering are given proving the termination of the following rules specifying *insertion sort* over $\mathbb{N}$:

$$\begin{array}{rcl}
sort(nil) & \rightarrow & nil \\
sort(x.y) & \rightarrow & ins(x, sort(y)) \\
ins(x, nil) & \rightarrow & x.nil \\
ins(x, u.v) & \rightarrow & choose(x, u.v, x, u) \\
choose(x, u.v, 0, 0) & \rightarrow & x.(u.v) \\
choose(x, u.v, s(p), 0) & \rightarrow & x.(u.v) \\
choose(x, u.v, 0, s(q)) & \rightarrow & u.ins(x, v) \\
choose(x, u.v, s(p), s(q)) & \rightarrow & choose(x, u.v, p, q)
\end{array}$$

(v) In chapter 6, some *relations between orderings* remain open. An interesting part of these problems concerns the comparison of path (and decomposition) orderings with polynomial orderings w.r.t. ground terms and monadic terms. We conjecture

---

[11] Obviously, the case where an operator f has a lexicographic status and an operator g has multiset status (assuming $f \sim g$) is new.

[12] see [Rubio and Nieuwenhuis, 1993] and remark 3.68 on page 71

that, in many cases, the polynomial orderings are stronger than the path (and decomposition) orderings.

(vi) There are still some components (and their connections) of the *termination expert* TETRES that have to be precisely formalized and then implemented. Currently, we have just begun with this work.

▶ **Results on Termination at RTA '93:**

The international conference on *rewriting techniques and applications* provides an academic computer science forum for the presentation of new results and current research in the field of rewriting. In 1993, a great variety of articles on termination have been published. The most important (w.r.t. the topics of this thesis) publications will be mentioned subsequently.

[Dershowitz and Hoot, 1993] generalizes path orderings and polynomial orderings (see (iv) of the previous item).

[Ferreira and Zantema, 1993] describes (among others) work on well-founded monotonic $\mathcal{F}$-algebras (see historical remark on page 98 and example 7.1).

[Middeldorp and Gramlich, 1993] shows that simple termination is an undecidable property.

[Rubio and Nieuwenhuis, 1993] defines a simplification path ordering which is AC-compatible as well as total on non-AC-equivalent ground terms (see remark 3.68 on page 71).

[Delor and Puel, 1993] provides a generalization of associative path orderings (see historical remark on page 71).

[Plaisted, 1993] proves that it is decidable in polynomial time whether there exists a simplification ordering which shows that a ground TRS terminates.

[Martin, 1993] contains an extension of the KBO (see historical remark on page 82).

▶ **Semantic orderings:**

As pointed out before, simplification orderings are pre-reduction orderings which have the subterm property. This property does not allow to prove the termination of, for example, the rule $f(f(x)) \to f(g(f(x)))$ since the left-hand side is homeomorphically embedded in the right-hand side. Sometimes homeomorphic embeddings appear in recursive definitions of functions. A way of (partially) dealing with these problems is the use of semantic orderings. Research in the field of semantic orderings (excluding polynomial orderings) is relatively young. However, more and more scientific publications on ordering techniques using semantics have been published recently. Some classical references should be briefly discussed at least:

(i) Semantic path orderings due to Kamin and Lévy ([Kamin and Lévy, 1980], [Geser, 1992]): The semantic path ordering SPO is a generalized version of the RPO(S) since it uses some reduction ordering instead of the precedence. We quote from [Dershowitz, 1987]: MORE GENERALLY, TERMS MAY BE MAPPED BY REPLACING THEIR OPERATORS WITH THE WHOLE TERM ITSELF, WHERE THE NEW OPERATOR IS THE WHOLE TERM ITSELF ORDERED BY SOME OTHER WELL-FOUNDED ORDERING. We insert the definition of the SPO: Let τ be a status function and $\succsim_{\mathsf{T}}$ be a quasi-ordering where $\succ_{\mathsf{T}}$ is σ-stable and well-founded:

$$\succsim_{\mathsf{SPO}} \;\equiv\; \{\succsim_{\mathsf{SPO}}^{\text{ⓜ}}\langle \mathcal{A}rgs\,,\, \mathcal{I}d\rangle\;,\; \succ_{\mathsf{SPO}}^{\text{ⓜ}}\langle \mathcal{I}d\,,\, \mathcal{A}rgs\rangle \;\&\; (\succ_{\mathsf{T}}\,,\, \succsim_{\mathsf{SPO}}^{\tau})\}$$

In order to apply the SPO to TRSs, some specific properties must be required. Following [Geser, 1992], a *semantic quasi-ordering* $\succsim_{\mathsf{T}}$ is a quasi-ordering such that $\sim_{\mathsf{T}}$ is σ-stable and $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible and $\succ_{\mathsf{T}}$ is σ-stable and well-founded. The *monotonic semantic path quasi-ordering* MSPO of [Geser, 1992] is then defined as

$$\succsim_{\mathsf{MSPO}} \;\equiv\; \succsim_{\mathsf{SPO}}\langle \mathcal{I}d\rangle \;\;\&\;\; \forall i \in [1,n]\colon \succsim_{\mathsf{T}}\langle f(x_1,\ldots,x_n)[.]_i\rangle$$

where $\succsim_{\mathsf{T}}$ is a semantic quasi-ordering. The strict part of the MSPO is a reduction ordering. The question now is what suitable semantic quasi-orderings are. [Geser, 1992] provides the following ones:

- Let $\succsim$ be a quasi-precedence. The root ordering
  $$\succsim_{\mathsf{F}} \;\equiv\; \{\succ\,,\,\sim\}$$
  is a semantic quasi-ordering. Note that, if $\succsim_{\mathsf{T}}$ is $\succsim_{\mathsf{F}}$ then the SPO is identical to the RPOS.
- A monotonic interpretation method $\succsim_{[.]}$ is a semantic quasi-ordering.
- Lexicographic combination of two semantic quasi-orderings yields a semantic quasi-ordering.

The following specification of quick-sort (see [Gramlich, 1990]) can be proved to be terminating with the MSPO where $\succsim_{\mathsf{T}} \;\equiv\; (\succsim_{\mathsf{F}}\,,\succsim_{[.]})$ with appropriate $\succsim_{\mathsf{F}}$ and $\succsim_{[.]}$:

$$\begin{array}{rcl}
qsort(nil) & \rightarrow & nil \\
qsort(x.y) & \rightarrow & qsort(lowers(x,y)) \circ (x.qsort(greaters(x,y))) \\
lowers(x,nil) & \rightarrow & nil \\
lowers(x,y.z) & \rightarrow & if(y \leq x, y.lowers(x,z), lowers(x,z)) \\
greaters(x,nil) & \rightarrow & nil \\
greaters(x,y.z) & \rightarrow & if(y \leq x, greaters(x,z), y.greaters(x,z))
\end{array}$$

The ordering $\succ_{\mathsf{T}}$ used to prove the termination of the quick-sort system illustrates that it is possible to mix syntactic orderings ($\succsim_{\mathsf{F}}$) and semantic orderings ($\succsim_{[.]}$). In [Walther, 1988], a very similar approach is presented (see also [Zehnter, 1992]). This method is able to prove termination automatically.

(ii) **Transformation orderings** due to Bellegarde and Lescanne ([Bellegarde and Lescanne, 1990], [Bachmair and Dershowitz, 1986]): Transformation orderings are based on a termination function ξ mapping terms into a well-founded set $(S, \succ_{\mathsf{S}})$ such that $s \succ_{\mathsf{S}} t$ if, and only if, $\xi(s) \succ_{\mathsf{S}} \xi(t)$ (see also definition 1.32 on page 14). The main characteristics of transformation orderings are as follows: We choose $(S, \succ_{\mathsf{S}})$ and ξ such that (i) S is the set $\mathcal{T}(\mathcal{F},\mathcal{X})$ of terms, (ii) ξ is the normalization by a TRS $\mathcal{T}$ ($\mathcal{T}$ stands for transformation), i.e. $\xi(t)$ is the $\mathcal{T}$-normal form of t and (iii) $\succ_{\mathsf{S}}$ is induced by a terminating TRS $\mathcal{S}$. Then a TRS $\mathcal{R}$ is terminating if $\xi(l) \succ_{\mathsf{S}} \xi(r)$ for all rules $l \rightarrow r \in \mathcal{R}$ and both $\mathcal{T}$ and $\mathcal{S}$ satisfy some properties:

> Let $\rightarrow_{\mathcal{S}}$ and $\rightarrow_{\mathcal{T}}$ be two rewriting relations such that $\rightarrow_{\mathcal{S}}$ locally cooperates with $\rightarrow_{\mathcal{T}}$, $\rightarrow_{\mathcal{S}\cup\mathcal{T}}$ is included in some reduction ordering $\succ_{\mathsf{T}}$, and $\rightarrow_{\mathcal{T}}$ is confluent:
>
> $$\succ_{\mathsf{TO}} \;\equiv\; \{=\langle\downarrow_{\mathcal{T}}\rangle \;\&\; \succ_{\mathsf{T}}\langle\mathcal{I}d\rangle\;,\; \neq\langle\downarrow_{\mathcal{T}}\rangle \;\&\; (\rightarrow_{\mathcal{S}} \cdot \xrightarrow{*}_{\mathcal{S}\cup\mathcal{T}})\langle\downarrow_{\mathcal{T}}\rangle\}$$
>
> where $t\!\downarrow_{\mathcal{R}}$ stands for the normal form of a term t w.r.t. a TRS $\mathcal{R}^{13)}$. $\rightarrow_{\mathcal{S}}$ locally cooperates with $\rightarrow_{\mathcal{T}}$ if, and only if, ${}_{\mathcal{T}}\xleftarrow{*} \cdot \rightarrow_{\mathcal{S}} \subseteq \xrightarrow{*}_{\mathcal{T}} \cdot \rightarrow_{\mathcal{S}} \cdot \xrightarrow{*}_{\mathcal{S}\cup\mathcal{T}} \cdot {}_{\mathcal{T}}\xleftarrow{*}$.

---

13) Note that, if $\rightarrow_{\mathcal{R}}$ is well-founded and confluent, the normal form of a term t exists and is unique.

Transformation orderings are more powerful than any other ordering: Take $\mathcal{T} = \emptyset$ and $\mathcal{S} = \mathcal{R}$, then $l{\downarrow_{\mathcal{T}}} \neq r{\downarrow_{\mathcal{T}}} \;\wedge\; l{\downarrow_{\mathcal{T}}} \rightarrow_{\mathcal{S}} r{\downarrow_{\mathcal{T}}}$ for all $l \rightarrow r \in \mathcal{R}$ and $\mathcal{S}$ terminates if, and only if, $\mathcal{R}$ terminates. In order to use the TO as a reduction ordering, there are three properties to be checked: local cooperation, confluence and termination. The last two properties can be proved with classical methods. There exists a method to test local cooperation for a restricted class of transformation systems $\mathcal{T}$. The central problem with transformation orderings is to find adequate TRSs $\mathcal{S}$ and $\mathcal{T}$ such that the termination of a given TRS can be proved. In [Steinbach, 1992a], a heuristic-driven algorithm is given which automatically generates $\mathcal{S}$ and $\mathcal{T}$ for many TRSs. For example, this algorithm automatically proves the termination of the TRS

$$\begin{array}{rcl}
int(0,0) & \rightarrow & 0.nil \\
int(0,s(y)) & \rightarrow & 0.int(s(0),s(y)) \\
int(s(x),0) & \rightarrow & nil \\
intlist(nil) & \rightarrow & nil \\
intlist(x.y) & \rightarrow & s(x).intlist(y) \\
int(s(x),s(y)) & \rightarrow & intlist(int(x,y))
\end{array}$$

specifying intervals of natural numbers, i.e. the binary function '*int*' is defined as $int(s^m(0), s^n(0)) = s^m(0).(s^{m+1}(0).(s^{m+2}(0).(\ldots(s^n(0).nil)\ldots)))$ where 's' represents the successor on natural numbers.

(iii) **Semantic labelling** due to Zantema: To conclude our brief discussion of contributions to semantic orderings, the work of [Zantema, 1992b] should be mentioned. We quote from [Zantema, 1992b]: A new kind of transformation of TRSs is proposed, depending on a choice for a model for the TRS. The labelled TRS is obtained from the original one by labelling operator symbols, possibly creating extra copies of some rules. This construction has the remarkable property that the labelled TRS is terminating if and only if the original TRS is terminating. Although the labelled version has more operation symbols and may have more rules (sometimes infinitely many), termination is often easier to prove for the labelled TRS than for the original one. This provides a new technique for proving termination, making classical techniques like RPO and polynomial interpretations applicable for non-simplifying TRSs.

We will now give some (amusing) examples requiring 'extremely semantic' techniques for their termination proofs. These TRSs also illustrate the limits of simplification orderings.

$\boxed{7.1}$ Example (A Chaotic Sequence ([Hofstadter, 1979]))

```
          f(s(0)) → s(0)
       f(s(s(0))) → s(0)
 f(s(s(s(x)))) → f(s(s(s(x))) - f(s(s(x)))) + f(s(s(s(x))) - f(s(x)))
            x + 0 → x
         x + s(y) → s(x + y)
            x - 0 → x
            0 - y → 0
      s(x) - s(y) → x - y
```

We quote from [Hofstadter, 1979]:

> This example of recursion in number theory leads to a small mystery. It is reminiscent of the Fibonacci definition in that each new value is a sum of the two previous values – but not of the *immediately* previous two values. Instead, the two immediately previous values tell *how far to count back* to obtain the numbers to be added to make the new value! The first 17 `f`-numbers run as follows:
>
> $$1, 1, 2, 3, 3, 4, 5, 5, 6, 6, 6, 8, 8, 8, 10, 9, 10.$$
>
> [...] The resulting sequence is, to put it mildly, erratic. The further out you go, the less sense it seems to make. This is one of those very peculiar cases where what seems to be a somewhat natural definition leads to extremely puzzling behaviour: chaos produced in a very orderly manner. One is naturally led to wonder whether the apparent chaos conceals some subtle regularity. Of course, by definition, there is regularity, but what is of interest is whether there is another way of characterizing this sequence – and with luck, a nonrecursive way.

We use a generalization of polynomial orderings for proving the termination of the TRS given above. This technique maps terms in a so-called well-founded monotonic $\mathcal{F}$-algebra $(A, \succ)$. For details, see [Zantema, 1992a].

Let $(A, \succ)$ be $(\{0,1\} \times \mathbb{N} \setminus \{0,1\}, \succ)$ such that $\succ$ is defined as $(a,b) \succ (c,d)$ if, and only if, $a \leq c \wedge b > d$. For each operator we have to provide a function that is strictly monotonic in all of its arguments:

$$\begin{aligned}
\llbracket \mathtt{f} \rrbracket((x,y)) &= \begin{cases} (0, 6y) & \textit{if } x = 0 \\ (0, y+1) & \textit{otherwise} \end{cases} \\
\llbracket \mathtt{-} \rrbracket((x,y),(u,v)) &= (1, y+v+1) \\
\llbracket \mathtt{+} \rrbracket((x,y),(u,v)) &= \begin{cases} (1, y+2v+1) & \textit{if } u = 0 \\ (1, y+v+1) & \textit{otherwise} \end{cases} \\
\llbracket \mathtt{s} \rrbracket((x,y)) &= \begin{cases} (0, 4y) & \textit{if } x = 0 \\ (0, y+1) & \textit{otherwise} \end{cases} \\
\llbracket \mathtt{0} \rrbracket() &= (1, 2)
\end{aligned}$$

We prove the monotonicity of the interpretation for the operator `s`. *Note that each variable x has to be interpreted as (x_1, x_2) where $x_1 \in \{0,1\}$ and $x_2 \in (\mathbb{N} \setminus \{0,1\})$. Thus, $(a,b) \succ (a',b') \rightsquigarrow \llbracket \mathtt{s}((a,b)) \rrbracket \succ \llbracket \mathtt{s}((a',b')) \rrbracket$ must hold.*

$$\begin{array}{ll}
a=1 \rightsquigarrow a'=1: & (0, b+1) \succ (0, b'+1) \\
a=0, a'=0: & (0, 4b) \succ (0, 4b') \\
a=0, a'=1: & (0, 4b) \succ (0, b'+1)
\end{array}$$

The $\succ$-relations hold since $b > b'$.

It is possible to show, for each rule, that the interpretation of the left-hand side is greater (w.r.t. $\succ$) than that of its corresponding right-hand side. For example, consider the third rule `f(s(s(s(x))))` $\rightarrow$ `f(s(s(x)) - f(s(s(x)))) + f(s(s(s(x))) - f(s(x)))`.

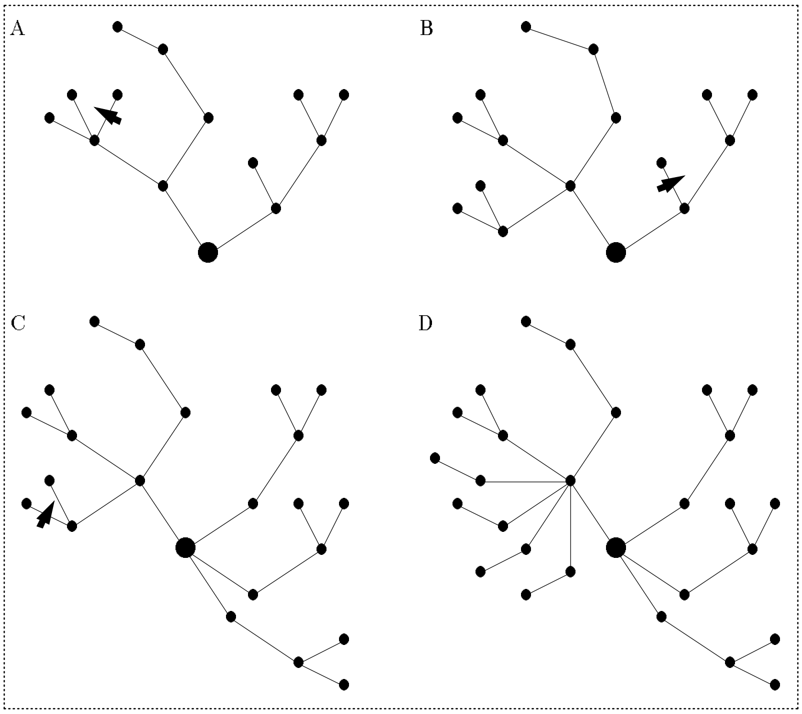
$$[\![\texttt{f(s(s(s((}x_1\texttt{,}x_2\texttt{)))))}]\!] = \begin{cases} (0,384x_2) & \textit{if } x_1 = 0 \\ (0,96x_2+96) & \textit{otherwise} \end{cases}$$

$$\begin{aligned}&[\![\texttt{f(s(s(s((}x_1,x_2\texttt{)))) - f(s(s((}x_1,x_2\texttt{)))))}\\ &\quad\texttt{+ f(s(s(s((}x_1,x_2\texttt{)))) - f(s((}x_1,x_2\texttt{))))}]\!]\end{aligned} = \begin{cases} (1,336x_2+7) & \textit{if } x_1 = 0 \\ (1,84x_2+91) & \textit{otherwise} \end{cases}$$

The left-hand side of the third rule is greater than its right-hand side because $(0,384x_2) \succ (1,336x_2+7)$ *and* $(0,96x_2+96) \succ (1,84x_2+91)$ *hold.* ◇

7.2 Example (Hercules and Hydra) *The battle of Hercules and a hydra was developed in [Kirby and Paris, 1982]. We quote from [Gardner, 1983a]:*

> A [...] complicated way of chopping down a tree was proposed by Laurie Kirby and Jeff Paris [...]. They call their tree graph a hydra. Its endpoints are the hydra's heads, and Hercules wants to destroy the monster by total decapitation. When a head is severed, its attached segment goes with it. Unfortunately after the first chop the hydra acquires one or more new heads by growing a new branch from a point (call it k) that is one step below the lost segment. This new branch is an exact replica of the part of the hydra that extends up from k. The figure at the top right in the *following* illustration [...] shows the hydra after Hercules has chopped off the head indicated by the sword in the figure at the top left.



> The situation for Hercules becomes increasingly desperate because when he makes his second chop, two replicas grow just below the severed segment [figure at bottom left]. And three replicas grow after the third chop [figure at bottom right], and so on. In general,

> n replicas sprout at each nth chop. [...] Kirby and Paris are able to show, utilizing an argument based on a remarkable number theorem found by the British logician R.L. Goodstein, that no matter what sequence Hercules follows in cutting off heads, the hydra is eventually reduced to a set of heads (there may be millions of them even if the starting form of the beast is simple) that are all joined directly to the root. They are then eliminated one by one until the hydra expires from lack of heads.
>
> A useful way to approach the hydra game is to think of the tree as modelling a set of nested boxes. Each box contains all the boxes reached by moving upward on the tree, and it is labeled with the maximum number of levels of nesting that it contains. Thus in the first figure of the hydra the root is a box of rank 4. Immediately above it on the left is a 3 box and on the right is a 2 box, and so on. All endpoints are empty boxes of rank 0. Each time a 0 box (hydra head) is removed the box immediately below gets duplicated (along with all its contents), but each of the duplicates as well as the original box now contains one less empty box. Eventually you are forced to start reducing ranks of boxes [...]. An inductive argument [...] will show that ultimately all boxes become empty, after which they are removed one at a time.
>
> I owe this approach to Dershowitz, who pointed out that it is not even necessary for the hydra to limit its growth to a consecutively increasing number of new branches. After each chop as many finite duplicates as you like may be allowed to sprout. It may take Hercules much longer to slay the monster, but there is no way he can permanently avoid doing so if he keeps hacking away. Note that the hydra never gets taller as it widens. [...]

In [Dershowitz and Jouannaud, 1990], the following specification of the battle of Hercules and hydra is given.

$$\begin{aligned}
\texttt{h(z,e(x))} &\to \texttt{h(c(z),d(z,x))}\\
\texttt{d(z,g(0,0))} &\to \texttt{e(0)}\\
\texttt{d(z,g(x,y))} &\to \texttt{g(e(x),d(z,y))}\\
\texttt{d(c(z),g(g(x,y),0))} &\to \texttt{g(d(c(z),g(x,y)),d(z,g(x,y)))}\\
\texttt{g(e(x),e(y))} &\to \texttt{e(g(x,y))}
\end{aligned}$$

Dershowitz and Jouannaud use ordinals and transfinite induction for proving the termination of this TRS. Note that the first rule is crucial w.r.t. termination (all other rules are ordered w.r.t. the RPO *based on the precedence* $\texttt{d} \succ \texttt{g} \succ \texttt{e}$*).* ◇

$\boxed{7.3}$ **Example ($3x+1$-Problem)** *The $3x+1$-problem, as it is now commonly called, is still resisting solution. It was first proposed before 1940 by Lothar Collatz:*

> *Start with any positive integer. Halve it if it is even; triple it and add 1 if it is odd. Keep repeating this procedure until the series loops in the cycle* $2, 1, 4, 2, 1, 4, \ldots$

But does the series always enter the $2, 1, 4$ loop? No one has proved that it does, nor has a counter-example been found. False proofs have even been published. A counter-example *would be a number that either keeps generating larger and larger numbers forever, without ever repeating a number, or one that enters a loop higher than the* $4, 2, 1$. *If a counterexample exists it would have to be extremely large because the conjecture has been tested for all numbers less than* 7×10^{11}. *Among numbers smaller than* 50, *the worst is* 27. *After* 77 *steps it reaches its peak of* 9232, *then* 34 *steps are required to take it down to* 1. *For more details, see [Gardner, 1983b]. The following TRS represents one possible specification of the* $3x+1$*-problem.*

```
              x + 0 → x
           x + s(y) → s(x + y)

              x ⋆ 0 → 0
           x ⋆ s(y) → (x ⋆ y) + x

       if(true,x,y) → x
      if(false,x,y) → y

            even(0) → true
         even(s(0)) → false
      even(s(s(x))) → even(x)

            half(0) → 0
         half(s(0)) → 0
      half(s(s(x))) → s(half(x))

             add(x) → s(s(s(s(s(s(s(s(0))) ⋆ x)))))))[14]

         problem(0) → 0
      problem(s(0)) → s(0)
   problem(s(s(x))) → if(even(x),problem(s(half(x))),problem(add(x)))
```

Obviously, the last rule is the crucial point in the termination proof w.r.t. an innermost reduction strategy checking the condition of the `if`*-operator first*[15]. *Especially, the term* `problem(add(x))` *on the right-hand side cannot be covered by* `problem(s(s(x)))` *since* `add(x)` *is a larger number than* `s(s(x))`. *It is possible to decrease* `add(x)` *by replacing it by* `half(add(x))` *because* `add(x)` *is always even*[16]. *However,* `half(add(x))` *is still greater than* `s(s(x))`. *Another way to decrease* `add(x)` *is based on a case distinction of the arguments of the operator* `problem`*: (i) If the argument is of the form* $4y+1$ *then the result is* $3y+1$ *(after two reductions), (ii) if the argument is of the form* $2y$ *then the result is* y *and (iii) the case* $4y-1$ *is not solved yet. Note that the results in the cases (i) and (ii) are smaller than their corresponding input values.* ◇

7.4 Example (Bulgarian Solitaire) *The task involves any triangular number*[17] n *of playing cards. Form a pile of* n *cards, then divide it into as many piles as you like, with an arbitrary number of cards in each pile. You may leave it as a single pile of* n, *or cut it into two, three or more piles, cutting anywhere you want, including* $n-1$ *cuts to make* n *piles*

---

[14] The operator *add* is needed on the right-hand side of the last rule, only. The term $add(x)$ is identical to $3x+7$ since the left-hand side of the last rule equals $3(x+2)+1$ whenever x is odd.

[15] Note that the TRS is *not* uniformly terminating since the following cycle exists: `problem(4)` → `problem(13)` → `problem(6)` → `problem(19)` → `problem(9)` → `problem(4)`.

[16] This fact holds since `add(x)` is called in the else-case on the right-hand side of the last rule, only. Thus, x is odd which implies $3x+7$ to be even.

[17] Partial sums of the series $1+2+3+\cdots$ are known as triangular numbers because they correspond to triangular arrays such as the 10 bowling pins or the 15 pool balls.

of one card each. Now keep repeating the following procedure. Take one card from each pile and place all the removed cards on the table to make a new pile. The piles need not be in a row. Just put them anywhere. Repeat the procedure to form another pile, and keep doing it.

As the structure of the piles keeps changing in irregular ways it seems unlikely you will reach a state where there will be just one pile with one card, one pile with two cards, one with three and so on. If you should reach this improbable state, without getting trapped in loops that keep returning the game to a previous state, the game must end, because now the state cannot change. Repeating the procedure leaves the cards in exactly the same consecutive state as before. It turns out, surprisingly, that regardless of the initial state of the game, you are sure to reach the consecutive state in a finite number of moves.

Bulgarian solitaire is a way of modelling some problems in partition theory that are far from trivial. The partitions of a counting number n are all the ways a positive integer can be expressed as the sum of positive integers without regard to their order. Bulgarian solitaire is a way of changing one partition to another by subtracting 1 from each number in the partition, then adding a number equal to the number of subtracted 1's.

Bulgarian solitaire for any triangular number of cards can be diagrammed as a tree with the consecutive partition labelling its root and all other partitions represented by the tree's points. The left-hand side of figure 7.3 shows the tree for the six-card game. The (directed[18]) graph *on the right-hand side of figure 7.3 represents the partitions of the prohibited seven-card game.*

For more details, see [Gardner, 1983a].

The following TRS represents a specification of the Bulgarian solitaire. The piles of cards are collected in a list where each pile is given as the number of cards it contains. The arguments of the top-level operator `problem` *are x and y where y is the list of piles totally consisting of* $\frac{1}{2}x(x+1)$ *cards (which is checked by the operator oklist).*

```
          x + 0 → x
       x + s(y) → s(x + y)

         sum(0) → 0
      sum(s(x)) → sum(x) + s(x)

   if(true,x,y) → x
  if(false,x,y) → y

          0 = 0 → true
       0 = s(y) → false
       s(x) = 0 → false
    s(x) = s(y) → x = y

    rem1(x,nil) → nil
    rem1(x,y.z) → if(x = y,z,y.rem1(x,z))

     rem(nil,y) → y
     rem(x.y,z) → rem(y,rem1(x,z))

 emptylist(nil) → true
 emptylist(x.y) → false

        list(0) → nil
     list(s(x)) → s(x).list(x)
```

---

[18] The representation of the six-card game does not need any *directed* graph since there exist no cycles.

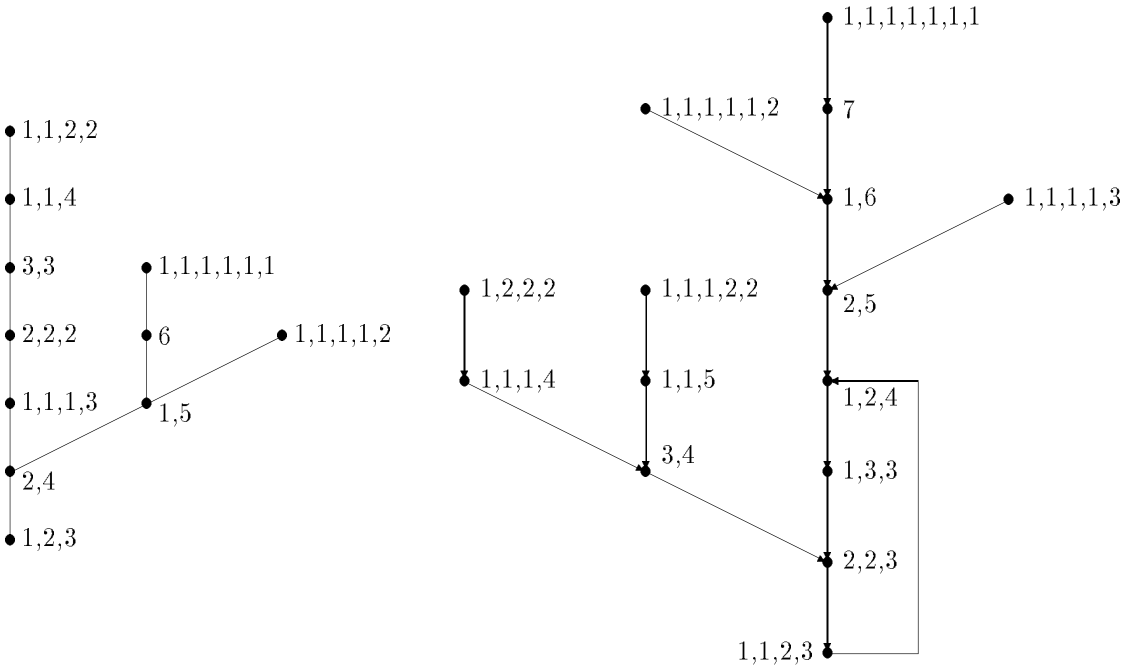


Figure 7.3: Bulgarian solitaire: Six-card and seven-card game

```
        end(x,y) → emptylist(rem(list(x),y))

    listsum(nil) → 0
    listsum(x.y) → x + listsum(y)

     oklist(0,y) → false
  oklist(s(x),y) → sum(s(x)) = listsum(y)

        h(nil,y) → y.nil
     h(s(0).x,y) → h(x,s(y))
  h(s(s(x)).y,z) → s(x).h(y,s(z))

     divide(x,y) → if(end(x,y),y,divide(x,h(y,0)))

    problem(x,y) → if(oklist(x,y),divide(x,y),y)
```

The proof of the innermost termination of the rule before last is difficult since the left-hand side is homeomorphically embedded in the right-hand side. What could help is a well-founded ordering for sorting the lists of piles in the order they are generated. ◇

$\boxed{7.5}$ **Example (Egyptian Fractions)** *The ancient Egyptians had a peculiarly hobbled approach to fractions. With the sole exception of* $2/3$*, for which there was a special hieroglyph, they had symbols only for unit fractions, that is, fractions that are the reciprocals of positive integers, with 1 above the line and any positive integer below. To manipulate fractions with numerators higher than 1 the Egyptians expressed such fractions as sums of distinct unit fractions. For example, instead of writing* $5/6$ *they wrote* $1/2 + 1/3$*. The term 'Egyptian* *fraction' has come to refer to any expression of a rational number as a sum of distinct unit fractions, traditionally arranged in decreasing order of size*[19]*.*

It is obvious that any proper fraction can be expressed as the sum of unit fractions if a repetition of terms is allowed. For example, $3/7$ *equals* $1/7 + 1/7 + 1/7$*. It is not obvious, however, that every proper fraction can be expressed as the sum of unit fractions even if a repetition is forbidden. One proof of this fact is the existence of a famous algorithm for writing any proper fraction as the sum of a finite number of distinct Egyptian fractions. The algorithm was first published in 1202 by Leonardo of Pisa, better known as Fibonacci.*

Fibonacci's method is simple. Call the proper fraction a/b*. The first term of the expansion is the largest unit fraction not greater than* a/b*. Now subtract the unit fraction from* a/b *to obtain another proper fraction. The second term of the expansion is the largest unit fraction not greater than this remainder. Continue in this manner, each time putting down the largest usable unit fraction as the next term in the expansion and then subtracting and repeating the process with the remainder. It is obvious that the fractions obtained in this way will grow steadily smaller. It can be proved that the process always terminates. Hence the algorithm always works.*[20] *For more details, see [Gardner, 1978].*

One way of specifying Fibonacci's algorithm with a TRS is given below. A sum of unit fractions will be represented as a list of its denominators. For example, the sum $1/2 + 1/4$ *has the representation* $s(s(0)) \,.\, (s(s(s(s(0)))) \,.\, nil)$*. Note that the division operator / is used as a constructor.*

```
                 x + 0 → x
              x + s(y) → s(x + y)

                 x ⋆ 0 → 0
              x ⋆ s(y) → (x ⋆ y) + x

                 x - 0 → x
                 0 - y → 0
           s(x) - s(y) → x - y

           (x/y)⋈(u/v) → ((x ⋆ v) - (y ⋆ u))/(y ⋆ v)

           h(s(0),y,z) → s(0)
      h(s(s(x)),s(0),z) → s(h(s(x),z,z))
   h(s(s(x)),s(s(y)),z) → h(s(x),s(y),z)

              div↾(x,y) → h(x,y,y)

           problem(0/y) → nil
        problem(s(x)/y) → div↾(y,s(x)).problem((s(x)/y)⋈(s(0)/div↾(y,s(x))))
```

The proof of the termination of the last rule is difficult since the left-hand side is homeomorphically embedded in the right-hand side. Note that it is easy to show that `s(x)/y` *– the argument of* `problem` *on the left-hand side – is greater than* `(s(x)/y)⋈(s(0)/div↾(y,s(x)))` *which is the problematic argument of* `problem` *on the right-hand side. However, this is no termination proof because there exist infinitely many fractions which are smaller than*

---

[19] Investigating the properties of Egyptian fractions is now a small but challenging task in number theory. For example, consider the following old Arabian brainteaser: Three sons share ownership of seven camels. The first son claims half the herd; the second son wants a quarter, the third demands an eighth. Can they each take their share without chopping up a camel (see [Stewart, 1992])?

[20] In today's terminology, Fibonacci's method is known as a greedy algorithm, because at each step in the process the largest fraction possible is chosen.

a specific one. A semantic termination argument could *be the value of the numerator after cancellation (using the idea for proving the termination of Fibonacci's algorithm), e.g.* $\frac{11}{13} \to \frac{1}{2} + \frac{9}{26} \to \frac{1}{2} + \frac{1}{3} + \frac{1}{78}$. ◇

$\boxed{7.6}$ **Example (The Rope Made of Rubber, see [Gardner, 1987])** *A worm is sitting at one end of a rope made of rubber. The length of the rope amounts to one kilometre.*

The worm is crawling along the rope with a steady speed of one centimetre per second. After the first second, the rope stretches like an elastic band to two kilometres, after a further second to three kilometres and so on. The question now is whether the worm will ever reach the end of the rope? It seems to be impossible for the worm to reach the end of the rope. However, he will succeed (after about $2^{200\,000}$ *seconds!).*

The reason is that the rope is stretching like an elastic band. This is why the worm will be carried along the rope with each expansion.

A way to solve this termination problem is to represent the worm's covered distance after each second as a fraction of the rope's length after this second. The worm will have reached the end of the rope if the sum of these fractions amounts to 1. After k seconds the worm has covered the distance (represented as fractions of the whole length of the rope)

$$\tfrac{1}{100\,000}\left(\tfrac{1}{1} + \tfrac{1}{2} + \tfrac{1}{3} + \cdots + \tfrac{1}{k}\right).$$

The sequence within the parentheses is known as the harmonic progression. The worm will reach the end of the rope since the harmonic progression is divergent.

In order to specify this example by a TRS we have to model the inequality $\frac{1}{100\,000}(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}) \geq 1$. *Obviously, dealing with fractions is more difficult than with positive integers. Thus, we transform the given inequality into an inequality over positive integers:*

$$\tfrac{k!}{1} + \tfrac{k!}{2} + \tfrac{k!}{3} + \cdots + \tfrac{k!}{k} \geq 100\,000 \cdot k!.$$

The following TRS is a specification of this inequality where the integer '100 000' is treated as a constant 'a'.

$$
\begin{aligned}
\mathtt{x + 0} &\to \mathtt{x}\\
\mathtt{x + s(y)} &\to \mathtt{s(x + y)}\\[1ex]
\mathtt{x} \star \mathtt{0} &\to \mathtt{0}\\
\mathtt{x} \star \mathtt{s(y)} &\to \mathtt{(x} \star \mathtt{y) + x}\\[1ex]
\mathtt{fac(0)} &\to \mathtt{s(0)}\\
\mathtt{fac(s(x))} &\to \mathtt{s(x)} \star \mathtt{fac(x)}\\[1ex]
\mathtt{f(s(0))} &\to \mathtt{s(0)}\\
\mathtt{f(s(s(x)))} &\to \mathtt{(s(s(x))} \star \mathtt{f(s(x))) + fac(s(x))}\\[1ex]
\mathtt{if(true,x,y)} &\to \mathtt{x}\\
\mathtt{if(false,x,y)} &\to \mathtt{y}\\[1ex]
\mathtt{0} \geq \mathtt{0} &\to \mathtt{true}\\
\mathtt{0} \geq \mathtt{s(y)} &\to \mathtt{false}\\
\mathtt{s(x)} \geq \mathtt{0} &\to \mathtt{true}\\
\mathtt{s(x)} \geq \mathtt{s(y)} &\to \mathtt{x} \geq \mathtt{y}\\[1ex]
\mathtt{problem(0)} &\to \mathtt{0}\\
\mathtt{problem(s(x))} &\to \mathtt{if(f(s(x))} \geq \mathtt{a} \star \mathtt{fac(s(x)),s(x),problem(s(s(x))))}
\end{aligned}
$$

This TRS is terminating w.r.t. an innermost reduction strategy checking the condition of the `if`*-operator first.* ◇

We conclude the list of amusing examples with the presentation of another example interesting from a termination point of view. We quote from [Gardner, 1983a]:

> Our [...] example of a task that looks as if it could go on forever when it really cannot is known as the 18-point problem. You begin with a line segment. Place a point anywhere you like on it. Now place a second point so that each of the two points is within a different half of the line segment. (The halves are taken to be '*open* intervals', which means that the endpoints are not considered 'inside' the interval.) Place a third point so that each of the three is in a different third of the line. At this stage it becomes clear that the first two points cannot be just anywhere. They cannot, for example, be close together in the middle of the line or close together at one end. They must be carefully placed so that when the third point is added, each will be in a different third of the line. You proceed in this way, placing every nth point so that the first n points always occupy different $1/n$th parts of the line. If you choose locations carefully, how many spots can you put on the line?
>
> Intuitively it seems as if the number should be endless. A line segment obviously can be divided into as many equal parts as you like and each may contain a point. The catch is that the points must be serially numbered to meet the task's conditions. It turns out, astonishingly, that you cannot get beyond 17 points! Regardless of how clever you are at placing 17 points, the 18th will violate the rules and the game ends. In fact, it is not even easy to place 10 points. [...]

As in the case of example 7.1 on page 177, it could be possible to prove the termination of (some of) the examples presented in this section with well-founded monotonic $\mathcal{F}$-algebras. A collection of other TRSs – which cannot naturally be oriented using simplification orderings – is currently being investigated in this direction. Good results can already be observed. For instance, for the following TRSs there exist monotonic functions proving termination:

Takeuchi Function:

```
f(s(x),s(y),s(z)) → if(s(x) ≤ s(y),
                        s(y),
                        f(f(x,s(y),s(z)),f(y,s(z),s(x)),f(z,s(x),s(y))))
```

Perfect Numbers:

```
       perfectp(0) → false
    perfectp(s(x)) → f(x,s(0),s(x),s(x))
        f(0,y,0,u) → true
     f(0,y,s(z),u) → false
     f(s(x),0,z,u) → f(x,u,z-s(x),u)
  f(s(x),s(y),z,u) → if(x ≤ y,f(s(x),y-x,z,u),f(x,u,z,u))
```

Exponential Function:

$$\begin{aligned} \mathtt{x}^{\mathtt{y}} &\to \mathtt{f(x,y,s(0))} \\ \mathtt{f(x,0,z)} &\to \mathtt{z} \\ \mathtt{f(x,s(y),z)} &\to \mathtt{if(odd(s(y)),f(x,y,x} \star \mathtt{z),f(x} \star \mathtt{x,half(s(y)),z))} \end{aligned}$$

For compactness, the specifications of the basic operations $\leq$, `if`, `-`, $\star$, `half` and `odd` have been omitted. However, the mentioned termination proofs cover the entire specifications.

Since the technique of well-founded monotonic $\mathcal{F}$-algebras is a generalization of polynomial orderings, we will try to develop a modification of the automatic generation method for polynomial orderings which is suitable for well-founded monotonic $\mathcal{F}$-algebras.

Die Zeit ist reif für neue Perspektiven.

IBM Werbeslogan

Appendix A

Examples

This chapter deals with the power as well as the limits of the practical applicability of the orderings presented. We collected some examples of TRSs, sometimes together with underlying theories. All examples will be presented in the following way:

<table>
<tr><td rowspan="5">$\mathcal{R}$</td><td>ops</td><td>ord1</td><td>ord2</td><td>$\ldots$</td></tr>
<tr><td>f_1</td><td></td><td></td><td></td></tr>
<tr><td>f_2</td><td></td><td></td><td></td></tr>
<tr><td>$\vdots$</td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">prec</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>$\mathcal{E}$</td></tr>
</table>

The TRS $\mathcal{R}$ based on a possibly empty underlying theory $\mathcal{E}$ is decreasing w.r.t. the orderings ord1, ord2, etc. The parameters of the orderings are given below their names. Status functions, extended weight functions and interpretations are associated to the operators $f_1, f_2, \ldots$ of $\mathcal{R}$ whereas precedences are represented as Hasse diagrams. We tried to find minimal parameters. If (certain parts of) parameters have been omitted, they either can possess *any* possible value or do not belong to the specified ordering.

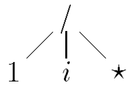
For representing the TRSs we use x, y, z and u, v as variables. Operators will be denoted by f, g (binary operators), h, i (unary operators), a, b (constants) and $k, k1$ (operators with three or four arguments). We also employ special operators to point to certain models. These symbols together with their meanings will be listed below:

$\equiv$	equality predicate		
$0, 1$	the natural numbers 0 and 1, respectively	p	predecessor on $\mathbb{N}$: $p(n) = n \dot{-} 1$
s	successor on $\mathbb{N}$: $s(n) = n + 1$	$+$	addition on $\mathbb{N}$
$\star$	multiplication on $\mathbb{N}$	$-$	unary or binary subtraction on $\mathbb{N}$
$/$	division on $\mathbb{N}$	fac $floop$	factorial function: $fac(n) = n!$ (for $floop$, see example A.19)
fib	Fibonacci function: $fib(n+2) = fib(n+1) + fib(n)$ where $fib(1) = fib(0) = 1$	exp	exponential function: $exp(n) = e^n$

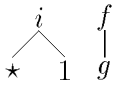
bin	binomial function: $bin(n,k) = \binom{n}{k}$	rem	remainder on $\mathbb{N}$: $rem(m,n)$ provides the remainder after dividing m by n. If the divisor is 0, then the remainder is m.
$divp$	the predicate whether the remainder of the first argument divided by the second argument is 0: $divp(m,n) = rem(m,n) \equiv 0$.	$prime$ $prime1$	predicates for testing prime numbers: $prime(n) = true$ if n is greater than 1 and its only divisors are itself and 1, provided n is an integer (for $prime1$, see example A.28)
$isum$	inverse sum: $isum(n) = \max\{i \mid \sum_{j=1}^{i} j \leq n\}$		
$false$ $true$	Boolean false and true, respectively	$\neg$	Boolean not
$\wedge$	Boolean and	$\vee$	Boolean or
$\oplus$	exclusive or	$\supset$	Boolean implication
if	if-then-else operator		
nil	empty list	$.$	cons operator on lists
del	removes duplicates in a list of elements (see example A.29)	$pair$	pairing constructor: $pair(m,n)$ represents a list of two elements m and n
1^{st} 2^{nd}	projection functions for $pair$: $1^{st}(pair(m,n)) = m$ and $2^{nd}(pair(m,n)) = n$		

Sometimes, the operator i is used as the reverse mapping ρ^{-1} of a mapping ρ (for example, $i(x) = -x$ or $i(x) = \frac{1}{x}$). To enhance readability we will often use infix notation.

$\boxed{\text{A.1}}$ Example (Group Theory)

	ops	ACK	AC-POL	A-RPO
$(x/x)/((y/y)/y) \rightarrow y$ $(x/y)/(z/y) \rightarrow x/z$ $x/x \rightarrow 1$ $1/x \rightarrow i(x)$ $x/i(y) \rightarrow x \star y$	$\star$	0, ⓜ	$x + y$	
	$/$	1	$x + y + 1$	
	i	1	$x + 1$	
	1	1	1	
$\star \in \mathcal{F}_{AC}$	**prec**			

$\boxed{\text{A.2}}$ Example (Taussky Group)

	ops	AC-POL	A-RPO
$1 \star 1 \rightarrow 1$ $x \star i(x) \rightarrow 1$ $f(x \star y, y) \rightarrow g(x \star y, x)$ $g(1, y) \rightarrow y$ $i(x \star y) \rightarrow i(y) \star i(x)$	$\star$	$x + y$	
	f	$2x + y$	
	g	$x + y$	
	i	x^2	
	1	1	
$\star \in \mathcal{F}_{AC}$	**prec**		

The interpretations $[\![\star]\!](x,y) = x+y$, $[\![f]\!](x,y) = 2x+y$, $[\![g]\!](x,y) = x+y$, $[\![i]\!](x) = x$ and $[\![1]\!]() = 1$ are compatible with the special endomorphism axiom $i(x \star y) \leftrightarrow_{\mathcal{E}} i(y) \star i(x)$. Thus, these interpretations are able to prove the termination of the equational (w.r.t. the last rule) TRS without the last rule (see lemma 5.58 on page 122).

$\boxed{\text{A.3}}$ Example (Logical Values, [Musser, 1980])

$$\begin{aligned} x \wedge y &\rightarrow \mathit{if}(x, y, \mathit{false}) \\ x \vee y &\rightarrow \mathit{if}(x, \mathit{true}, y) \\ x \supset y &\rightarrow \mathit{if}(x, y, \mathit{true}) \\ \neg x &\rightarrow \mathit{if}(x, \mathit{false}, \mathit{true}) \\ \mathit{if}(\mathit{true}, x, y) &\rightarrow x \\ \mathit{if}(\mathit{false}, x, y) &\rightarrow y \end{aligned}$$

$\wedge, \vee \in \mathcal{F}_{AC}$

ops	ACKM	AC-POL
$\wedge$	2,ⓜ	$x+y+2$
$\vee$	2,ⓜ	$x+y+2$
$\supset$	2	$x+y+2$
$\neg$	3	$x+3$
if	0	$x+y+z$
true	1	1
false	1	1

We prove the termination of the first rule by using the ACKM based on the given parameters: $s = x \wedge y \succ_{\mathsf{ACKM}} \mathit{if}(x,y,\mathit{false}) = t \iff \tilde{s} \succ_{\mathsf{KBOS}} \tilde{t}$. The memory-flattened forms are as follows: $\tilde{s} = x \wedge^1 y$, $\tilde{t} = \mathit{if}(x,y,\mathit{false})$. The relation $x \wedge^1 y \succ_{\mathsf{KBOS}} \mathit{if}(x,y,\mathit{false})$ is valid since $\varphi(x \wedge^1 y) = 4 > 3 = \varphi(\mathit{if}(x,y,\mathit{false}))$ holds and the variable condition is fulfilled.

Note that the parameters of the ACKM and that of the AC-POL are absolutely identical since the constant monomial of each interpretation is equal to the weight of the corresponding operator (see the paragraph below example 1.41 on page 21).

$\boxed{\text{A.4}}$ Example (Boolean Algebra)

$$\begin{aligned} x \oplus \mathit{false} &\rightarrow x \\ x \oplus x &\rightarrow \mathit{false} \\ x \wedge \mathit{false} &\rightarrow \mathit{false} \\ x \wedge \mathit{true} &\rightarrow x \\ x \wedge x &\rightarrow x \\ (x \oplus y) \wedge z &\rightarrow (x \wedge z) \oplus (y \wedge z) \end{aligned}$$

$\wedge, \oplus \in \mathcal{F}_{AC}$

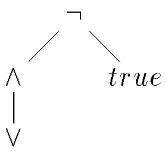
ops	AC-POL	AD-RPO
$\wedge$	xy	
$\oplus$	$x+y+1$	
false	2	
true	2	
prec		$\wedge$ | $\oplus$ | false

If the last rule is used as the underlying theory (as distributivity axiom) instead of the AC-theory, the POL based on the interpretations $[\![\wedge]\!](x,y) = xy + x$, $[\![\oplus]\!](x,y) = x+y$, $[\![\mathit{true}]\!]() = [\![\mathit{false}]\!]() = 1$ is able to prove the termination (see lemma 5.53 on page 120).

A.5 Example (Boolean Functions)

$$\begin{aligned} x \vee false &\rightarrow x \\ x \wedge true &\rightarrow x \\ \neg false &\rightarrow true \\ \neg(x \vee y) &\rightarrow (\neg x) \wedge (\neg y) \\ (x \vee y) \wedge z &\rightarrow (x \wedge z) \vee (y \wedge z) \end{aligned}$$

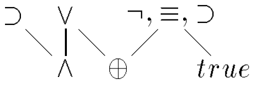
$\wedge, \vee \in \mathcal{F}_{AC}$

ops	AC-POL	AD-RPO
$\wedge$	xy	
$\vee$	$x+y+1$	
$\neg$	$2x$	
$false$	1	
$true$	1	
prec		

A.6 Example (Boolean Ring)

$$\begin{aligned} \neg x &\rightarrow x \oplus true \\ x \supset y &\rightarrow (x \wedge y) \oplus (x \oplus true) \\ x \vee y &\rightarrow (x \wedge y) \oplus (x \oplus y) \\ x \equiv y &\rightarrow x \oplus (y \oplus true) \end{aligned}$$

$\wedge, \oplus \in \mathcal{F}_{AC}$

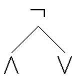
ops	AC-POL	A-RPO
$\wedge$	$x+y$	
$\vee$	$2x+2y+1$	
$\supset$	$2x+y+2$	
$\oplus$	$x+y$	
$\equiv$	$x+y+2$	
$\neg$	$x+2$	
$true$	1	
prec		

In order to compare the second rule w.r.t. the A-RPO, $\overline{x \supset y} = x \supset y \succ_{\mathsf{RPO}} \oplus(x \wedge y, x, true) = \overline{(x \wedge y) \oplus (x \oplus true)}$ must hold. This is true since the relations $\supset \succ \oplus$, $\supset \succ \wedge$ and $\supset \succ true$ are parts of the precedence.

A.7 Example (Disjunctive Normal Form, [Dershowitz, 1982])

$$\begin{aligned} \neg\neg x &\rightarrow x \\ \neg(x \vee y) &\rightarrow (\neg\neg\neg x) \wedge (\neg\neg\neg y) \\ \neg(x \wedge y) &\rightarrow (\neg\neg\neg x) \vee (\neg\neg\neg y) \\ x \wedge x &\rightarrow x \\ x \vee x &\rightarrow x \end{aligned}$$

$\wedge, \vee \in \mathcal{F}_{AC}$

ops	ACK
$\wedge$	0,ⓜ
$\vee$	0,ⓜ
$\neg$	0
prec	

The parameters of this ACK satisfy the conditions required by the definition of the ACK, for example, the A-operators $\wedge$ and $\vee$ are minimal w.r.t. the precedence. Note that the second rule and the third rule decrease only if $\varphi(\neg) = 0$. Furthermore, the application of the flattening operation is superfluous since all terms are in normal form w.r.t. $\overline{\ }$.

Note that there does not exist an AC-path or decomposition ordering which guarantees the termination of $\mathcal{R}$ modulo AC since even the classical termination of $\mathcal{R}$ cannot be shown with any path or decomposition ordering. In [Dershowitz, 1982], other specifications for computing disjunctive normal forms are given which can be proved to be AC-terminating with AC-path and decomposition orderings (see also the following example).

$\boxed{\text{A.8}}$ Example (Disjunctive Normal Form, [Dershowitz, 1983])

		RPO
$\neg(x \wedge y) \rightarrow (\neg x) \vee (\neg y)$ $\neg(x \vee y) \rightarrow (\neg x) \wedge (\neg y)$ $x \wedge (y \vee z) \rightarrow (x \wedge y) \vee (x \wedge z)$	**prec**	$\neg$ $\mid$ $\wedge$ $\mid$ $\vee$

As shown in [Dershowitz, 1983], the POL is not able to prove the termination of this TRS. However, the following functions are sufficient: $[\![\neg]\!](x) = 2^x$, $[\![\wedge]\!](x,y) = xy$ and $[\![\vee]\!](x,y) = x + y + 1$ and $\mu \geq 2$[1].

We prove the termination of the distributivity rule $l = x \wedge (y \vee z) \rightarrow (x \wedge y) \vee (x \wedge z) = r$ with the help of the PSO based on the precedence $\wedge \succ \vee$. The set of paths of both terms are as follows:

$$
\begin{aligned}
\mathcal{P}ath(\{l\}) &= \{\!\{\, [l,x]\,, [l, y \vee z, y]\,, [l, y \vee z, z] \,\}\!\} \\
\mathcal{P}ath(\{r\}) &= \{\!\{\, [r, x \wedge y, x]\,, [r, x \wedge y, y]\,, [r, x \wedge z, x]\,, [r, x \wedge z, z] \,\}\!\}
\end{aligned}
$$

$$
\begin{array}{lllll}
l \succ_{\mathsf{PSO}} r & \Longleftrightarrow & \multicolumn{3}{l}{\mathcal{P}ath(\{l\}) \overset{\textcircled{m}}{\succ}_{\mathsf{PO}} \mathcal{P}ath(\{r\})} \\
 & \curvearrowleft & 1. \quad [l,x] \succ_{\mathsf{PO}} [r, x \wedge y, x] & & \\
 & & \Longleftrightarrow \quad \{l,x\} \overset{\textcircled{m}}{\succ}_{\mathsf{TO}} \{r, x \wedge y, x\} & & \\
 & & \curvearrowleft \qquad \bullet\ l \succ_{\mathsf{TO}} r & & \\
 & & \qquad\qquad \text{since } \wedge \succ \vee & & \\
 & & \qquad\qquad \bullet\ l \succ_{\mathsf{TO}} x \wedge y & & \\
 & & \qquad\qquad\quad \Longleftrightarrow \quad \{\!\{\, [x]\,, [y \vee z, y]\,, [y \vee z, z] \,\}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{PO}} \{\!\{\, [x]\,, [y] \,\}\!\} & & \\
 & & \qquad\qquad\quad \curvearrowleft \quad [y \vee z, y] \succ_{\mathsf{PO}} [y] & & \\
 & & \qquad\qquad\qquad\quad \Longleftrightarrow \quad \{y \vee z, y\} \overset{\textcircled{m}}{\succ}_{\mathsf{TO}} \{y\} & & \\
 & & 2. \quad [l, y \vee z, y] \succ_{\mathsf{PO}} [r, x \wedge y, y]\text{: analogous to 1.} & & \\
 & & 3. \quad [l, x] \succ_{\mathsf{PO}} [r, x \wedge z, x]\text{: analogous to 1.} & & \\
 & & 4. \quad [l, y \vee z, z] \succ_{\mathsf{PO}} [r, x \wedge z, z]\text{: analogous to 1.} & &
\end{array}
$$

The method of comparing terms w.r.t. the PSDS is identical to that of the PSO when all operators have status ⓜ.

---

[1] For example, $[\![\neg(x \wedge y)]\!](x,y) = 2^{xy} \sqsupset 2^x + 2^y + 1 = [\![(\neg x) \vee (\neg y)]\!](x,y)$ since $2^{xy} \sqsupseteq 2^x \cdot 2^y$ (because $xy \sqsupseteq x + y$ if $\mu \geq 2$) and $2^x \cdot 2^y \sqsupset 2^x + 2^y + 1 \Longleftrightarrow 2^x(2^y - 1) \sqsupset 2^y + 1 \curvearrowleft 4(2^y - 1) \sqsupset 2^y + 1 \Longleftrightarrow 3 \cdot 2^y - 5 \sqsupset 0$ which is true for all $y \geq 1$.

A.9 Example (Boolean Implication)

<table>
<tr><td rowspan="5">$(\neg x) \supset y \;\rightarrow\; x \vee y$
$(\neg x) \supset (y \supset z) \;\rightarrow\; y \supset (x \vee z)$
$x \supset (y \vee z) \;\rightarrow\; y \vee (x \supset z)$</td><td>ops</td><td>KBO</td><td>POL</td><td>RDO</td></tr>
<tr><td>$\vee$</td><td>1</td><td>$x+y$</td><td></td></tr>
<tr><td>$\supset$</td><td>1</td><td>$x+2y$</td><td></td></tr>
<tr><td>$\neg$</td><td>1</td><td>$2x$</td><td></td></tr>
<tr><td>prec</td><td>$\supset$
$\mid$
$\vee$</td><td></td><td>$\neg$
$\mid$
$\supset$
$\mid$
$\vee$</td></tr>
</table>

We show that the second rule $l = (\neg x) \supset (y \supset z) \;\rightarrow\; y \supset (x \vee z) = r$ is decreasing w.r.t. the RDO based on the precedence $\neg \succ \supset \succ \vee$:

$$\begin{aligned} \mathcal{D}ec(\{l\}) &= \{\!\{\, \{l, \neg x, x\}\,,\ \{l, y \supset z, y\}\,,\ \{l, y \supset z, z\} \,\}\!\} \\ \mathcal{D}ec(\{r\}) &= \{\!\{\, \{r, y\}\,,\ \{r, x \vee z, x\}\,,\ \{r, x \vee z, z\} \,\}\!\} \end{aligned}$$

$$\begin{array}{lll}
l \succ_{\mathsf{RDO}} r & \Longleftrightarrow & \mathcal{D}ec(\{l\})\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \mathcal{D}ec(\{r\}) \\
 & & \curvearrowleft\ 1.\ \{l, y \supset z, y\} \succ_{\mathsf{D}}^{\textcircled{m}} \{r, y\} \\
 & & \qquad \curvearrowleft\ l \succ_{\mathsf{D}} r \\
 & & \qquad\qquad \Longleftrightarrow\ \{y \supset z, y\} \succ_{\mathsf{D}}^{\textcircled{m}} \{y\} \\
 & & \quad\ \ 2.\ \{l, \neg x, x\} \succ_{\mathsf{D}}^{\textcircled{m}} \{r, x \vee z, x\} \\
 & & \qquad\ \text{since } \neg \succ \supset \text{ and } \neg \succ \vee \\
 & & \quad\ \ 3.\ \{l, y \supset z, z\} \succ_{\mathsf{D}}^{\textcircled{m}} \{r, x \vee z, z\} \\
 & & \qquad \curvearrowleft\ \bullet\ l \succ_{\mathsf{D}} r \\
 & & \qquad\qquad\quad \curvearrowleft\ \{y \supset z, z\} \succ_{\mathsf{D}}^{\textcircled{m}} \{x \vee z, z\} \\
 & & \qquad\qquad\qquad \text{since } \supset \succ \vee \\
 & & \qquad\qquad \bullet\ l \succ_{\mathsf{D}} x \vee z \\
 & & \qquad\qquad\quad \text{since } \supset \succ \vee
\end{array}$$

The comparison of l and r w.r.t. the DOS must be performed in the same way.

A.10 Example (Boolean Implication)

<table>
<tr><td rowspan="6">$x \wedge \mathit{false} \;\rightarrow\; \mathit{false}$
$x \wedge (\neg \mathit{false}) \;\rightarrow\; x$
$\neg\neg x \;\rightarrow\; x$
$\mathit{false} \supset y \;\rightarrow\; \neg \mathit{false}$
$x \supset \mathit{false} \;\rightarrow\; \neg x$
$(\neg x) \supset (\neg y) \;\rightarrow\; y \supset (x \wedge y)$</td><td>ops</td><td>POL</td><td>RDO</td></tr>
<tr><td>$\wedge$</td><td>$x+y$</td><td></td></tr>
<tr><td>$\supset$</td><td>$2x+2y$</td><td></td></tr>
<tr><td>$\neg$</td><td>$2x$</td><td></td></tr>
<tr><td>false</td><td>1</td><td></td></tr>
<tr><td>prec</td><td></td><td>$\supset$
$\mid$
$\neg$
$\mid$
$\wedge$</td></tr>
</table>

Assuming we have two boolean constructors *false* and $\neg$. Furthermore, there is a complete definition of the boolean operator $\wedge$ (see the first two rules) in addition to the rule $\neg\neg x \rightarrow x$ (see the third rule). Then the last three rules completely define the boolean *implication* $\supset$.

We prove the termination of the last rule $l = (\neg x) \supset (\neg y) \; \rightarrow \; y \supset (x \wedge y) = r$ with the help of the KNSS based on the precedence $\neg \succ \wedge$:

$$\begin{array}{rcl} \mathcal{P}ath(\{l\}) & = & \{\!\{ [l, \neg x, x], [l, \neg y, y] \}\!\} \\ \mathcal{P}ath(\{r\}) & = & \{\!\{ [r, y], [r, x \wedge y, x], [r, x \wedge y, y] \}\!\} \end{array}$$

$$\begin{array}{lll} l \succ_{\mathsf{KNSS}} r & \Longleftrightarrow & \mathcal{P}ath(\{l\}) \overset{\textcircled{m}}{\succ}_{\mathsf{PO}} \mathcal{P}ath(\{r\}) \\ & & \curvearrowleft \; 1. \; [l, \neg x, x] \succ_{\mathsf{PO}} [r, x \wedge y, x] \\ & & \qquad \Longleftrightarrow \; \{l, \neg x, x\} \overset{\textcircled{n}}{\succ}_{\mathsf{TO}} \{r, x \wedge y, x\} \\ & & \qquad\quad \curvearrowleft \; \bullet \; l \succ_{\mathsf{TO}} r \\ & & \qquad\qquad\quad \curvearrowleft \; [\neg x, x] \succ_{\mathsf{PO}} [x \wedge y, x] \\ & & \qquad\qquad\qquad \text{since } \neg \succ \wedge \\ & & \qquad\qquad \bullet \; \neg x \succ_{\mathsf{TO}} x \wedge y \\ & & \qquad\qquad\quad \text{since } \neg \succ \wedge \\ & & \quad 2. \; [l, \neg y, y] \succ_{\mathsf{PO}} [r, x \wedge y, y]\text{: analogous to 1.} \\ & & \quad 3. \; [l, \neg y, y] \succ_{\mathsf{PO}} [r, y]\text{: analogous to 1.} \end{array}$$

If $\wedge$ is treated as an AC-operator, the TRS is decreasing w.r.t. the AE-IRDS with the precedence given above. Note that the termination of the last rule cannot be proved by the AE-RPOS.

A.11 Example (Associativity and Distributivity, [Lescanne, 1986])

	ops	KBO	POL[2]
$(x+y)+z \rightarrow x+(y+z)$	$+$		$xy+x$
$(x \star y)+(x \star z) \rightarrow x \star (y+z)$	$\star$	1	xy
$(x \star y)+((x \star z)+u) \rightarrow (x \star (y+z))+u$			

A.12 Example (Associativity and Endomorphism, [Bellegarde and Lescanne, 1988])

	ops	KBO	POL
$(x+y)+z \rightarrow x+(y+z)$	$+$		$xy+x$
$h(x)+h(y) \rightarrow h(x+y)$	h	1	$2x$
$h(x)+(h(y)+z) \rightarrow h(x+y)+z$			

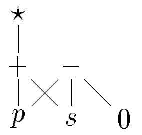
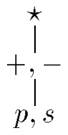
If $+$ is treated as an AC-operator (and the first rule is removed), the ACK based on the parameters $\varphi(+) = 0$, $\varphi(h) = 1$ is able to prove the AC-termination of this TRS.

---

[2] $\mu \geq 2$

A.13 Example (Integers, [Nutt *et al.*, 1989])

$$\begin{aligned}
p(s(x)) &\to x\\
s(p(x)) &\to x\\
0+y &\to y\\
s(x)+y &\to s(x+y)\\
p(x)+y &\to p(x+y)\\
-0 &\to 0\\
-s(x) &\to p(-x)\\
-p(x) &\to s(-x)\\
0\star y &\to 0\\
s(x)\star y &\to (x\star y)+y\\
p(x)\star y &\to (x\star y)+(-y)
\end{aligned}$$

ops	IPOL	RPO
$+$	$x+y$	
$\star$	xy	
$-$	x	
p	$x+1$	
s	$x+1$	
0	2	
prec		

This is a specification of the integers which is confluent on ground terms. Note that the interpretations specifying the IPOL model the semantics of the operators.

A.14 Example (Difference)

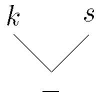
$$\begin{aligned}
i(x)+i(y) &\to i(x)-y\\
i(x)-i(y) &\to i(x)+y\\
i(x)-i(y) &\to y-x
\end{aligned}$$

ops	KBO	POL	RPO
$+$	0	$x+y$	
$-$	0	$x+y$	
i	1	$x+1$	
prec			$+ \sim -$

The operator $+$ can be used as a C-operator while applying the KBOS with the *same* parameters as given for the KBO (i.e. the CK with status ⓜ for C-operators), the POL (i.e. the AC-POL) and the RPO (i.e. the C-RPO with status ⓜ for C-operators) for proving the termination of the TRS modulo C-theories.

A.15 Example (Absolute Value)

$$\begin{aligned}
x-0 &\to x\\
0-y &\to 0\\
s(x)-s(y) &\to x-y\\
k(0,y,z) &\to z\\
k(s(x),y,0) &\to s(x)-y\\
k(s(x),0,s(z)) &\to k(z,0,x)\\
k(s(x),s(y),s(z)) &\to k(x-y,y-x,s(z))
\end{aligned}$$

ops	POL	PSO
k	$x+y+z$	
$-$	$x+y$	
s	$3x$	
0	1	
prec		

The operator k specifies the function $|(x \dot{-} y)-z|$ on natural numbers x, y, z. The termination of the last rule can be proved with the help of the IRD similar to example 3.14 on page 45.

$\boxed{\text{A.16}}$ Example (Unary and Binary Subtraction)

	ops	KBO	POL	RDO
$((-x)-(-x))-((-y)-(-y)) \;\to\; (x-y)-(x-y)$	$-_{bin}$		$x+y$	
	$-_{un}$	1	$x+1$	

Note that this TRS is not decreasing w.r.t. the PSO. We will show how the IRD proves the termination of the rule $l = ((-x)-(-x))-((-y)-(-y)) \;\to\; (x-y)-(x-y) = r$:

$$\begin{array}{rcl}
\mathcal{D}ec(\{l\}) & = & \{\!\{\, \{l,(-x)-(-x),-x,x\}\,,\, \{l,(-x)-(-x),-x,x\}\,, \\
 & & \{l,(-y)-(-y),-y,y\}\,,\, \{l,(-y)-(-y),-y,y\} \,\}\!\} \\
\mathcal{D}ec(\{r\}) & = & \{\!\{\, \{r,x-y,x\}\,,\, \{r,x-y,y\}\,,\, \{r,x-y,x\}\,,\, \{r,x-y,y\} \,\}\!\}
\end{array}$$

$$\begin{array}{l}
l \succ_{\mathsf{IRD}} r \iff \mathcal{D}ec(\{l\}) \; (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}} \; \mathcal{D}ec(\{r\}) \\
\qquad \curvearrowleft \;\; 1.\;\; \{l,(-x)-(-x),-x,x\} \succ_{\mathsf{D}}^{\textcircled{m}} \{r,x-y,x\} \\
\qquad\qquad \curvearrowleft \;\; \bullet\; l \succ_{\mathsf{D}} r \\
\qquad\qquad\qquad \iff \{(-x)-(-x),-x,x\} \succ_{\mathsf{D}}^{\textcircled{m}} \{x-y,x\} \\
\qquad\qquad\qquad\quad \curvearrowleft \;\; (-x)-(-x) \succ_{\mathsf{D}} x-y \\
\qquad\qquad\qquad\qquad \curvearrowleft \;\; \{-x,x\} \succ_{\mathsf{D}}^{\textcircled{m}} \{x\} \\
\qquad\qquad\quad \bullet\; (-x)-(-x) \succ_{\mathsf{D}} x-y \\
\qquad\qquad\qquad \iff \{-x,x\} \succ_{\mathsf{D}} \{x\} \\
\qquad\quad\;\; 2.\;\; \{l,(-y)-(-y),-y,y\} \succ_{\mathsf{D}}^{\textcircled{m}} \{r,x-y,y\} \\
\qquad\qquad \text{analogous to 1.}
\end{array}$$

$\boxed{\text{A.17}}$ Example (Square Function)

	ops	IPOL
$x+0 \to x$ $x+s(y) \to s(x+y)$ $h(0) \to 0$ $h(s(x)) \to (s(h(x))+x)+x$ $(x+h(y))+z \to x+(z+h(y))$	$+$	$x+y$
	h	x^2
	s	$x+1$
	0	2
	prec	h $\vert$ $+$ $\vert$ s

In this example, the IPOL uses the semantics of the operators. The operator $+$ describes the addition on $\mathbb{N}$ and s represents the successor function. The recursion appearing in the definition of the operator h can be resolved into $h(x) = x^2$. The TRS without the last rule represents a complete specification of the square function and decreases w.r.t. the POL based on the interpretations $[\![h]\!](x) = x^2$, $[\![+]\!](x,y) = x + 2y$, $[\![s]\!](x) = x+3$ and $[\![0]\!]() = 2$.

We prove the termination of the fourth rule w.r.t. the IPOL given above:

$$\begin{array}{rcl}
[\![h(s(x))]\!](x) & = & (x+1)^2 \;=\; x^2+2x+1 \\
[\![(s(h(x))+x)+x]\!](x) & = & ((x^2+1)+x)+x \;=\; x^2+2x+1
\end{array}$$

The equality of the polynomials of both sides of the rule requires the comparison of the leading operators. Since h (the leading operator of the left-hand side) is greater than $+$ (the leading operator of the right-hand side), the rule is decreasing w.r.t. the IPOL given above.

Note that there exists no path or decomposition ordering that is able to prove the termination of the last rule since for covering the arguments of the top most occurrence of $+$ on the right-hand side, $\tau(+)$ must be ① and for covering the other occurrence of $+$ on the right-hand side, $\tau(+)$ must be ⓜ.

$\boxed{\text{A.18}}$ Example (Division)

	ops	KBOS	POL	RPOS
$i(x/y) \rightarrow y/x$ $x/(y/z) \rightarrow (x/i(z))/y$	$/$	ⓡ	$x + 2y + 1$	ⓡ
	i	0	$2x$	ⓡ
	prec	i $\mid$ $/$		$i \sim /$

We prove that the second rule $l = x/(y/z) \rightarrow (x/i(z))/y = r$ is decreasing w.r.t. the RPOS based on the parameters given above:

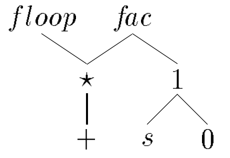
$$\begin{array}{l}
l \succ_{\mathsf{RPOS}} r \curvearrowleft 1.\ \{\!\{ l \}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{RPOS}} \{\!\{ x/i(z), y \}\!\} \\
\qquad \Longleftrightarrow\ l \succ_{\mathsf{RPOS}} x/i(z) \\
\qquad\qquad \curvearrowleft\ 1.1\ \{\!\{ l \}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{RPOS}} \{\!\{ x, i(z) \}\!\} \\
\qquad\qquad\qquad \Longleftrightarrow\ \bullet\ l \succ_{\mathsf{RPOS}} x \\
\qquad\qquad\qquad\qquad \text{since } x \in \mathcal{S}ym(l) \\
\qquad\qquad\qquad\qquad \bullet\ l \succ_{\mathsf{RPOS}} i(z) \\
\qquad\qquad\qquad\qquad\quad \curvearrowleft\ \bullet\ \{\!\{ l \}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{RPOS}} \{\!\{ z \}\!\} \\
\qquad\qquad\qquad\qquad\qquad\quad \text{since } z \in \mathcal{S}ym(l) \\
\qquad\qquad\qquad\qquad\qquad\quad \bullet\ (x, y/z) \overset{\textcircled{r}}{\succ}_{\mathsf{RPOS}} (z) \\
\qquad\qquad\qquad\qquad\qquad\quad \text{since } z \in \mathcal{S}ym(y/z) \\
\qquad\qquad 1.2\ (x, y/z) \overset{\textcircled{r}}{\succ}_{\mathsf{RPOS}} (x, i(z)) \\
\qquad\qquad\quad \curvearrowleft\ y/z \succ_{\mathsf{RPOS}} i(z) \\
\qquad\qquad\qquad \curvearrowleft\ \bullet\ \{\!\{ y/z \}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{RPOS}} \{\!\{ z \}\!\} \\
\qquad\qquad\qquad\quad \text{since } z \in \mathcal{S}ym(y/z) \\
\qquad\qquad\qquad\quad \bullet\ (y, z) \overset{\textcircled{r}}{\succ}_{\mathsf{RPOS}} (z) \\
\quad 2.\ (x, y/z) \overset{\textcircled{r}}{\succ}_{\mathsf{RPOS}} (x/i(z), y) \\
\qquad \curvearrowleft\ y/z \succ_{\mathsf{RPOS}} y \\
\qquad\quad \curvearrowleft\ \{\!\{ y \}\!\} \overset{\textcircled{m}}{\succsim}_{\mathsf{RPOS}} \{\!\{ y \}\!\}
\end{array}$$

The application of the KBOS requires $\varphi(i)$ to be zero otherwise the second rule is decreasing in the reverse direction. This weight function makes the weights of both terms of the second rule equal. This is why we need status ⓡ for the operator $/$.

A.19 Example (Factorial Function, [Padawitz, 1989])

$$\begin{aligned} x \star 0 &\rightarrow 0 \\ x \star s(y) &\rightarrow (x \star y) + x \\ 1 &\rightarrow s(0) \\ fac(0) &\rightarrow 1 \\ fac(s(x)) &\rightarrow s(x) \star fac(x) \\ floop(0, y) &\rightarrow y \\ floop(s(x), y) &\rightarrow floop(x, s(x) \star y) \end{aligned}$$

$+, \star \in \mathcal{F}_{AC}$

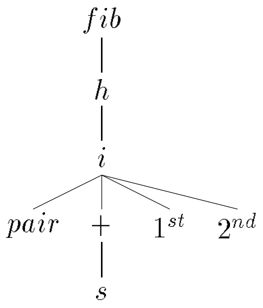
ops	AC-POL	AE-RPOS
$+$	$x + y$	
$\star$	xy	
fac	x^3	
$floop$	$x^2y + x$	①
s	$x^2 + 1$	
0	2	
1	6	
prec		

floop is an iterative program for the factorial function. Its first argument is the argument of *fac*, while the second argument serves as an accumulator for the result of *fac*: $floop(x, y) = y \star fac(x)$.

Note that the first condition of lemma 5.67 on page 125 is able to detect a strictly mixed interpretation of the operator $\star$ (see the second rule).

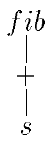
A.20 Example (Fibonacci Function, [Toyama, 1986])

$$\begin{aligned} fib(0) &\rightarrow s(0) \\ fib(s(0)) &\rightarrow s(0) \\ fib(s(s(x))) &\rightarrow 1^{st}(i(h(x))) \\ h(0) &\rightarrow pair(s(0), s(0)) \\ h(s(x)) &\rightarrow i(h(x)) \\ i(x) &\rightarrow pair(1^{st}(x) + 2^{nd}(x), 1^{st}(x)) \\ 1^{st}(pair(x, y)) &\rightarrow x \\ 2^{nd}(pair(x, y)) &\rightarrow y \\ x + 0 &\rightarrow x \\ x + s(y) &\rightarrow s(x + y) \end{aligned}$$

	RPO
prec	

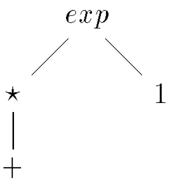
A.21 Example (Fibonacci Function)

$$\begin{aligned} x + 0 &\rightarrow x \\ x + s(y) &\rightarrow s(x + y) \\ fib(0) &\rightarrow s(0) \\ fib(s(0)) &\rightarrow s(0) \\ fib(s(s(x))) &\rightarrow fib(s(x)) + fib(x) \end{aligned}$$

	RPO
prec	

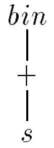
Polynomial orderings are not able to prove the termination of this TRS. However, the following functions are sufficient: $[\![fib]\!](x) = 2^{x+1}$, $[\![+]\!](x, y) = x + 2y$, $[\![s]\!](x) = x + 2$ and $[\![0]\!]() = 1$. The last three rules of the TRS given above are decreasing w.r.t. the POL based on the interpretations $[\![fib]\!](x) = 3x$, $[\![+]\!](x, y) = x + y$, $[\![s]\!](x) = 2x$ and $[\![0]\!]() = 1$.

$\boxed{\text{A.22}}$ Example (Exponential Function, [Jouannaud and Muñoz, 1984])

		AD-RPO
$x+0 \to x$ $x \star 1 \to x$ $(x+y)\star z \to (x\star z)+(y\star z)$ $x\star(y+z) \to (x\star y)+(x\star z)$ $exp(0) \to 1$ $exp(x+y) \to exp(x)\star exp(y)$	**prec**	
$+,\star \in \mathcal{F}_{AC}$		

The TRS modulo the underlying theory is decreasing w.r.t. the following functions: $[\![exp]\!](x) = 2^x$, $[\![\star]\!](x,y) = xy$, $[\![+]\!](x,y) = x+y+1$, $[\![0]\!]() = [\![1]\!]() = 2$. Furthermore, using both distributivity rules as the underlying theory (instead of $+,\star \in \mathcal{F}_{AC}$) the interpretations $[\![exp]\!](x) = 2^x$, $[\![\star]\!](x,y) = xy$, $[\![+]\!](x,y) = 2x+2y$, $[\![0]\!]() = [\![1]\!]() = 2$ prove the termination of the equational TRS (see lemma 5.55 on page 121).

$\boxed{\text{A.23}}$ Example (Binomial Coefficients)

		AD-RPO
$x+0 \to x$ $x+s(y) \to s(x+y)$ $bin(0,s(y)) \to 0$ $bin(x,0) \to s(0)$ $bin(s(x),s(y)) \to bin(x,s(y))+bin(x,y)$	**prec**	
$+ \in \mathcal{F}_{AC}$		

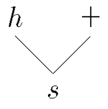
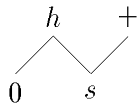
In order to prove the termination with the AD-RPO, the TRS $D(\succ)$ must be computed, first: $D(\succ) = D(+\succ s) = \{x+s(y) \to s(x+y)\,,\, s(x)+y \to s(x+y)\}$. Consider the second rule $l = x+s(y) \to s(x+y) = r$. For comparing l and r, we apply δ and $^{-}$ to both sides and then use the RPO for comparing the generated normal forms: $\overline{\delta(l)} = s(x+y) = \overline{\delta(r)}$. Since the normal forms are identical, $l \xrightarrow{+}_{D(\succ)/A} r$ must hold: $l = x+s(y) \to_{D(\succ)} s(x+y) = r$.

$\boxed{\text{A.24}}$ Example (Constant Function)

	ops	POL
$h(0) \to s(0)$ $h(s(0)) \to s(0)$ $h(s(s(x))) \to h(h(s(x)))$	h	$x+2$
	s	x^2+1
	0	1

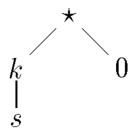
The function h always returns the constant '$s(0)$'. This example was proposed by Christoph Walther as a challenge.

$\boxed{\text{A.25}}$ Example (Identity Function)

	ops	KBOS	POLS
$x+0 \to x$ $x+s(y) \to s(x+y)$ $h(0) \to 0$ $h(s(x)) \to s(h(h(x)))$ $h(s(x))+y \to y+s(x)$	h	0	x
	$+$	ⓜ	$x+y$,ⓜ
	s		$x+1$
	0		1
	prec		

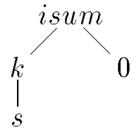
For proving the termination of the fourth rule, $\varphi(h)$ must be zero and $[\![h]\!](x)$ has to be the identity, respectively. Thus, both sides of the last rule are equal w.r.t. their weights and their interpretations, respectively. Both the KBOS and the POLS have to require multiset status for the operator $+$ since the terms are commutatively identical when removing the operator h on the left-hand side. This is why the KBO as well as the IPOL fail to show the termination of this TRS.

$\boxed{\text{A.26}}$ Example (Multiplication)

	ops	POL	RPOS
$x \star y \to k(x,y,x,0)$ $k(0,y,u,v) \to 0$ $k(s(x),0,u,v) \to 0$ $k(s(x),s(0),0,v) \to v$ $k(s(x),s(0),s(u),v) \to k(s(x),s(0),u,s(v))$ $k(s(x),s(s(y)),0,v) \to k(s(x),s(y),s(x),v)$ $k(s(x),s(s(y)),s(u),v) \to k(s(x),s(s(y)),u,s(v))$	$\star$	$x^2y + x^2 + 2$	
	k	$x^2y + z^2 + u$	①
	s	$x + 2$	
	0	1	
	prec		

This TRS specifies the multiplication on natural numbers without explicitly using the addition. The last argument of the operator k provides the result. Consider the first rule. The third argument (x) will be added to the fourth one y times (where y is the second argument).

$\boxed{\text{A.27}}$ Example (Inverse Sum)

	ops	RPOS
$isum(x) \to k(x,s(0),s(0))$ $k(0,y,z) \to 0$ $k(s(x),s(0),z) \to s(k(x,s(z),s(z)))$ $k(s(x),s(s(y)),z) \to k(x,s(y),z)$	$isum$	
	k	①
	s	
	0	
	prec	

The operator $isum$ with input x computes the maximum of all numbers y_i satisfying the inequality $\sum_{j=1}^{y_i} j \leq x$.

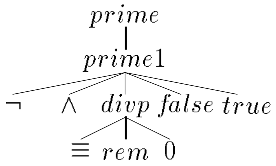
We show that the third rule $l = k(s(x),s(0),z) \to s(k(x,s(z),s(z))) = r$ is also terminating w.r.t. the RDOS (since the RDOS is an extension of the RPOS) based on the parameters $k \succ s$ and $\tau(k) = ①$:

$$\mathcal{D}ec(\{l\}) = \{\!\{\, \{l, s(x), x\}\,, \{l, s(0), 0\}\,, \{l, z\} \,\}\!\}$$
$$\mathcal{D}ec(\{r\}) = \{\!\{\, \{r, k(x,s(z),s(z)), x\}\,, \{r, k(x,s(z),s(z)), s(z), z\}\,, \{r, k(x,s(z),s(z)), s(z), z\} \,\}\!\}$$

$$
\begin{array}{llll}
l \succ_{\mathsf{RDOS}} r & \Longleftrightarrow & \multicolumn{2}{l}{\mathcal{D}ec(\{l\}) \; (\overset{\text{ⓜ}}{\succ}_{\mathsf{D}})^{\text{ⓜ}} \; \mathcal{D}ec(\{r\})} \\
 & \curvearrowleft\ 1. & \multicolumn{2}{l}{\{l, s(x), x\} \overset{\text{ⓜ}}{\succ}_{\mathsf{D}} \{r, k(x, s(z), s(z)), x\}} \\
 & & \curvearrowleft\ \bullet & l \succ_{\mathsf{D}} r \\
 & & & \text{since } k \succ s \\
 & & \bullet & l \succ_{\mathsf{D}} k(x, s(z), s(z)) \\
 & & & \Longleftrightarrow \quad (s(x), s(0), z) \overset{\text{①}}{\succ}_{\mathsf{RDOS}} (x, s(z), s(z)) \\
 & & & \qquad \text{since } s(x) \succ_{\mathsf{RDOS}} x \\
 & 2. & \multicolumn{2}{l}{\{l, z\} \overset{\text{ⓜ}}{\succ}_{\mathsf{D}} \{r, k(x, s(z), s(z)), s(z), z\}} \\
 & & \curvearrowleft\ \bullet & l \succ_{\mathsf{D}} r \\
 & & & \text{since } k \succ s \\
 & & \bullet & l \succ_{\mathsf{D}} k(x, s(z), s(z)) \\
 & & & \text{see 1.} \\
 & & \bullet & l \succ_{\mathsf{D}} s(z) \\
 & & & \text{since } k \succ s
\end{array}
$$

The method of comparing terms w.r.t. the DOS, the PSDS and the IRDS is identical to that of the RDOS whenever the operators possess lexicographic status.

A.28 **Example (Prime Numbers, [Boyer and Moore, 1979])**

<table>
<tr><td rowspan="14">$$\begin{array}{rcl} prime(0) & \rightarrow & false \\ prime(s(0)) & \rightarrow & false \\ prime(s(s(x))) & \rightarrow & prime1(s(s(x)), s(x)) \\ prime1(x, 0) & \rightarrow & false \\ prime1(x, s(0)) & \rightarrow & true \\ prime1(x, s(s(y))) & \rightarrow & (\neg divp(s(s(y)), x)) \\ & & \wedge\ prime1(x, s(y)) \\ divp(x, y) & \rightarrow & rem(x, y) \equiv 0 \end{array}$$</td><th>ops</th><th>POL</th><th>RPO</th></tr>
<tr><td>$prime$</td><td>x^3</td><td></td></tr>
<tr><td>$prime1$</td><td>xy^2</td><td></td></tr>
<tr><td>$\wedge$</td><td>$x + y$</td><td></td></tr>
<tr><td>$\neg$</td><td>$x + 1$</td><td></td></tr>
<tr><td>$\equiv$</td><td>$x + y$</td><td></td></tr>
<tr><td>rem</td><td>$x + y$</td><td></td></tr>
<tr><td>$divp$</td><td>$x+y+3$</td><td></td></tr>
<tr><td>s</td><td>$2x$</td><td></td></tr>
<tr><td>0</td><td>2</td><td></td></tr>
<tr><td>$false$</td><td>2</td><td></td></tr>
<tr><td>$true$</td><td>2</td><td></td></tr>
<tr><td>**prec**</td><td></td><td> </td></tr>
</table>

We illustrate the BL-approach for proving a polynomial to be positive by applying it to the sixth rule (note that $\mu = 2$):

$$
\begin{array}{rcl}
[\![prime1(x, s(s(y)))]\!](x, y) & = & x \cdot (2(2y))^2 \;=\; 16xy^2 \\
[\![(\neg divp(s(s(y)), x)) \wedge prime1(x, s(y))]\!](x, y) & = & ((2(2y) + x + 3) + 1) + x \cdot (2y)^2 \\
 & = & 4xy^2 + x + 4y + 4
\end{array}
$$

$$\begin{aligned}16xy^2-(4xy^2+x+4y+4) &= 12xy^2-x-4y-4\\ &\sqsupseteq \tfrac{47}{4}xy^2-4y-4\\ &\sqsupseteq \tfrac{43}{4}xy^2-4\\ &\sqsupseteq \tfrac{41}{4}xy^2\\ &\sqsupset 0\end{aligned}$$

Note that the interpretation of the operator *prime* must be separate since $[\![prime1]\!](x,y)$ is strictly mixed. The necessity of a strictly separate interpretation for *prime*1 can be deduced independently of the interpretation of *prime*. Lemma 5.66 on page 125 can be applied to the operator *prime* and lemma 5.67 on page 125 can be applied to the operator *prime*1.

$\boxed{\textbf{A.29}}$ Example ([Mohan and Srivas, 1986])

	ops	POL
$del(x.(y.z)) \rightarrow k(x \equiv y, x, y, z)$	del	x^2
$k(true, x, y, z) \rightarrow del(y.z)$	$.$	$x+y$
$k(false, x, y, z) \rightarrow x.del(y.z)$	$\wedge$	$x+y$
$nil \equiv nil \rightarrow true$	$\equiv$	xy
$(x.y) \equiv nil \rightarrow false$	k	$x+y+z^2+2zu+u^2$
$nil \equiv (u.v) \rightarrow false$	nil	2
$(x.y) \equiv (u.v) \rightarrow (x \equiv u) \wedge (y \equiv v)$	$false$	2
	$true$	2

The unary function *del* on lists returns a list obtained by deleting consecutive repetitions of members of the argument of *del*.

The first rule is decreasing w.r.t. the POL given above:

$$\begin{aligned}[\![del(x.(y.z))]\!](x,y,z) &= (x+y+z)^2 = x^2+y^2+z^2+2xy+2xz+2yz\\ [\![k(x\equiv y,x,y,z)]\!](x,y,z) &= xy+x+y^2+2yz+z^2\end{aligned}$$

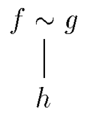
It is obvious that $x^2+y^2+z^2+2xy+2xz+2yz \sqsupset y^2+z^2+x+xy+2yz$ holds since it is equivalent to $x^2+xy+2xz \sqsupset x$ which is true because $x^2 \sqsupseteq x$ (and even $x^2 \sqsupset x$ if $\mu = 2$).

$\boxed{\textbf{A.30}}$ Example (Mutual Recursion)

	ops	KBO	POL
	f	0	$2x+y+1$
	g	0	$2x+y$
$f(x,y) \rightarrow g(x,y)$	h	1	$x+2$
$g(h(x),y) \rightarrow h(f(x,y))$	**prec**	f $\mid$ g $\mid$ h	

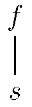
This TRS is decreasing w.r.t. the KBO since both sides of each rule have the same weights (provided that $\varphi(f)$ and $\varphi(g)$ are equal). The precedence is chosen in a way such that the leading operators of the left-hand sides are greater than the leading operators of the corresponding right-hand sides. The validity of the variable condition concludes the termination proof.

$\boxed{\text{A.31}}$ Example (Mutual Recursion)

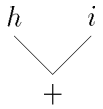
	ops	KBOS	POL	RPOS
$f(h(x),y) \rightarrow g(x,h(y))$ $g(h(x),y) \rightarrow f(x,h(y))$	f	0,①	$2x+y$	①
	g	0,①	$2x+y$	①
	h	1	$x+1$	
	prec	$f \sim g$		

The KBO is unable to prove the termination of this TRS since a quasi-precedence is necessary.

$\boxed{\text{A.32}}$ Example (Péter Function)

	ops	RPOS
$f(x,0) \rightarrow s(x)$ $f(0,s(y)) \rightarrow f(s(0),y)$ $f(s(x),s(y)) \rightarrow f(f(x,s(y)),y)$	f	ⓡ
	s	
	0	
	prec	

$\boxed{\text{A.33}}$ Example (Non-Deterministic Machines, [Hullot, 1980])

	ops	AC-POL	A-RPO
$x+x \rightarrow x$ $x+nil \rightarrow x$ $h(x)+x \rightarrow h(x)$ $h(x+h(y)) \rightarrow h(x+y)+h(y)$ $i(x+h(y)) \rightarrow i(x+y)+i(y)$ $h(x+y)+x \rightarrow h(x+y)$ $h(h(x)) \rightarrow h(x)$ $i(h(x)) \rightarrow i(x)$	$+$	$x+y$	
	h	x^2	
	i	x^2	
	nil	2	
$+ \in \mathcal{F}_{AC}$	**prec**		

This is the theory considered by R. Milner for axiomatizing the behaviour of finite non-deterministic machines.

$\boxed{\text{A.34}}$ Example ([Middeldorp, 1988])

	ops	POL
$f(x,y) \rightarrow x$ $h(a) \rightarrow k(a,b,a)$ $i(x) \rightarrow f(x,x)$ $k(x,x,y) \rightarrow h(x)$	f	$x+y$
	h	x^2
	i	$2x+1$
	k	$xy+z$
	a	3
	b	1

$\boxed{\text{A.35}}$ Example ([Kapur *et al.*, 1986])

$$\begin{aligned}
a+b &\rightarrow b+a\\
a+(b+z) &\rightarrow b+(a+z)\\
(x+y)+z &\rightarrow x+(y+z)\\
f(a,y) &\rightarrow a\\
f(b,y) &\rightarrow b\\
f(x+y,z) &\rightarrow f(x,z)+f(y,z)
\end{aligned}$$

ops	IPOL	RPOS
$+$	$x+y+1$	①
f	xy	
a	3	
b	2	
prec		a f $\vert$ $\vert$ b $+$

The termination of this TRS can also be proved with the lexicographic concatenation of two POLs ('Cartesian product', see remark 5.11 on page 99):

(i) $[\![+]\!](x,y) = x+y+1,\ [\![f]\!](x,y) = xy,\ [\![a]\!]() = 3,\ [\![b]\!]() = 2$:

R1: $[\![a+b]\!] = [\![b+a]\!] = 6$
R2: $[\![a+(b+z)]\!] = [\![b+(a+z)]\!] = z+7$
R3: $[\![(x+y)+z]\!] = [\![x+(y+z)]\!] = x+y+z+2$
R4: $[\![f(a,y)]\!] = 3y \sqsupset 3 = [\![a]\!]$
R5: $[\![f(b,y)]\!] = 2y \sqsupset 2 = [\![b]\!]$
R6: $[\![f(x+y,z)]\!] = xz+yz+z \sqsupset xz+yz+1 = [\![f(x,z)+f(y,z)]\!]$

(ii) $[\![+]\!](x,y) = xy+x,\ [\![a]\!]() = 3,\ [\![b]\!]() = 2$:

R1: $[\![a+b]\!] = 9 \sqsupset 8 = [\![b+a]\!]$
R2: $[\![a+(b+z)]\!] = 6z+9 \sqsupset 6z+8 = [\![b+(a+z)]\!]$
R3: $[\![(x+y)+z]\!] = xyz+xy+xz+x \sqsupset xyz+xy+x = [\![x+(y+z)]\!]$

$\boxed{\text{A.36}}$ Example (SEND+MORE=MONEY, [Narain, 1989])

$$\begin{aligned}
f(x,nil) &\rightarrow nil\\
f(x,y.z.u.v) &\rightarrow g(k1(x,y,z)\equiv u,\\
&\qquad y.z.u.f(k(x,y,z),v))
\end{aligned}$$

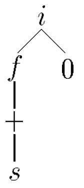
ops	POL	RPOS
f	xy^2	ⓡ
$.$	xy	
$\equiv$	$x+y$	
k	$x+y+z$	
$k1$	$x+y+z$	
g	$x+y$	
nil	2	
prec		f $\vert$ $g, \equiv, k1, ., k$

The problem is to assign the variables D, E, M, N, O, R, S, Y to distinct values in $\{0, 1, \ldots, 9\}$ so that $SEND + MORE = MONEY$ is a correct equation, where for example, $SEND$ is interpreted as $1000 \cdot S + 100 \cdot E + 10 \cdot N + D$. The TRS given above represents the most important part (w.r.t. termination) of the TRS specifying this problem. For more information, we refer to [Narain, 1989], pages 337–338. Note that, for simplicity, the parentheses for the cons operator '.' have been omitted.

A.37 Example ([Jantke and Thomas, 1988])

$$
\begin{aligned}
x + 0 &\rightarrow x \\
x + s(y) &\rightarrow s(x + y) \\
h(i(h(x))) &\rightarrow h(f(s(0), x)) \\
h(i(f(x, y))) &\rightarrow h(f(s(x), y)) \\
h(f(x, f(y, z))) &\rightarrow h(f(x + y, z))
\end{aligned}
$$

$+ \in \mathcal{F}_{AC}$

ops	AC-POL	AE-RPOS
f	$xy + y$	ⓡ
h	x	
i	$4x + 1$	
$+$	xy	
s	$x + 1$	
0	2	
prec		

Appendix B

Proofs

This appendix consists of the proofs of the theorems and the lemmata contained in the sections 2, 3, 4, 5 and 6. We use the following abbreviations: $\hookrightarrow$ for therefore and $\because$ for because.

B.1 Proofs of Chapter 2

$\boxed{\text{B.1}}$ **Lemma (2.7)** *$\approx$ is a congruence on $\mathcal{T}(\mathcal{F},\mathcal{X})$ and σ-stable.* $\diamond$

Proof: We have to prove that $\approx$ is reflexive, symmetric, transitive, σ-stable and that $\forall i \in [1,n]$: $s_i \approx t_i \hookrightarrow f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n)$ holds:

i) $t \approx t$: This can easily be proved by induction on $|t|$ (analogous to iii)).

ii) $s \approx t \hookrightarrow t \approx s$: This is true by definition of $\approx$.

iii) $r \approx s \approx t \hookrightarrow r \approx t$: Induction on $|r| + |t|$:
Let $r = f(r_1,\ldots,r_k)$, $s = g(s_1,\ldots,s_m)$, $t = h(t_1,\ldots,t_n)$.
$\hookrightarrow$ $f \sim g$, $g \sim h$, $k = m$, $m = n$
$\because$ Definition of $\approx$
$\hookrightarrow$ $f \sim h$, $k = n$
$\because$ Transitivity of $\sim$ and $=$

(1) $\tau(f) = \tau(g) = \tau(h) =$ ⓜ
$\hookrightarrow$ $\exists \pi_1$: $\forall k \in [1,n]$: $r_k \approx s_{\pi_1(k)} \wedge \exists \pi_2$: $\forall k \in [1,n]$: $s_k \approx t_{\pi_2(k)}$
$\because$ Definition of $\approx$
$\hookrightarrow$ $\exists \pi_3$: $\forall k \in [1,n]$: $r_k \approx t_{\pi_3(k)}$
$\because$ Induction hypothesis
$\hookrightarrow$ $r \approx t$
$\because$ Definition of $\approx$

(2) $\tau(f) = (o_1,\ldots,o_n)$, $\tau(g) = (p_1,\ldots,p_n)$, $\tau(h) = (q_1,\ldots,q_n)$
$\hookrightarrow$ $\forall k \in [1,n]$: $r_{o_k} \approx s_{p_k} \wedge \forall k \in [1,n]$: $s_{p_k} \approx t_{q_k}$
$\because$ Definition of $\approx$
$\hookrightarrow$ $\forall k \in [1,n]$: $r_{o_k} \approx t_{q_k}$
$\because$ Induction hypothesis

iv) $s \approx t \hookrightarrow s\sigma \approx t\sigma$: Induction on $|s| + |t|$ (analogous to iii))

v) $\forall i \in [1, n]$: $s_i \approx t_i \hookrightarrow f(s_1,\dots,s_n) \approx f(t_1,\dots,t_n)$: This is obviously true by definition of $\approx$ (irrespective of $\tau(f)$). ◆

B.2 Proofs of Chapter 3

$\boxed{\textbf{B.2}}$ **Theorem (3.25)** *The* DOS *is a* σ*-stable simplification ordering.* ◇

Proof: The proof of the DOS being a simplification ordering is very similar to that of the IRDS (see B.4). Therefore, it is omitted. ◆

$\boxed{\textbf{B.3}}$ **Theorem (3.27)** *The* PSDS *is a* σ*-stable simplification ordering.* ◇

Proof: The proof of the PSDS being a simplification ordering is very similar to that of the IRDS (see B.4). Therefore, it is omitted. ◆

$\boxed{\textbf{B.4}}$ **Theorem (3.29)** *The* IRDS *is a* σ*-stable simplification ordering.* ◇

Proof:

i) The IRDS is irreflexive:
We have to show that $\forall t \in \mathcal{T}(\mathcal{F},\mathcal{X})$: $t \not\succ_{\mathsf{IRDS}} t$ which is valid since $\succ_{\mathsf{IRDS}} \equiv (\succ_{\mathsf{IRDS}}^{\textcircled{m}})^{\textcircled{m}}\langle\mathcal{D}ec\rangle$ and $\sim_{\mathsf{D}} \equiv \approx\langle\mathcal{I}d\rangle$.

ii) $\succ_{\mathsf{D}}$ is $\sim_{\mathsf{D}}$-compatible:
We prove that $(r \succ_{\mathsf{D}} s \sim_{\mathsf{D}} t \vee r \sim_{\mathsf{D}} s \succ_{\mathsf{D}} t) \hookrightarrow r \succ_{\mathsf{D}} t$ holds by induction on $|r| + |s| + |t|$.

▶ $r \succ_{\mathsf{D}} s \sim_{\mathsf{D}} t$: Let $r = f(r_1,\dots,r_l)$, $s = g(s_1,\dots,s_m)$, $t = h(t_1,\dots,t_n)$ such that $g \sim h$ (since $s \sim_{\mathsf{D}} t$).

1) $f \succ g \sim h$
$\hookrightarrow$ $f \succ h$
$\because$ $\succsim$ is a quasi-precedence
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$
$\because$ Definition of $\succ_{\mathsf{D}}$

2) $f \sim g \sim h$

2.1) $\tau(f) = \tau(g) = \tau(h) = \textcircled{m}$

2.1.1) $\mathcal{S}ub(r) \succ_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(s)$
$\hookrightarrow$ $\mathcal{S}ub(r) \succ_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t)$
$\because$ $\mathcal{S}ub(s) \approx^{\textcircled{m}} \mathcal{S}ub(t)$ (since $s \sim_{\mathsf{D}} t$) and induction hypothesis
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$
$\because$ Definition of $\succ_{\mathsf{D}}$

2.1.2) $\mathcal{S}ub(r) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(s) \;\wedge\; \mathcal{D}ec(\mathcal{A}rgs(r))\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\mathcal{A}rgs(s))$
$\hookrightarrow$ $\mathcal{S}ub(r) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t) \;\wedge\; \mathcal{D}ec(\mathcal{A}rgs(r))\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\mathcal{A}rgs(t))$
$\because$ $\mathcal{S}ub(s) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t)$, $\mathcal{D}ec(\mathcal{A}rgs(s))\,(\approx^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\mathcal{A}rgs(t))$ and induction hypothesis
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$
$\because$ Definition of $\succ_{\mathsf{D}}$

2.2) $\tau(f) = (i_1,\ldots,i_l),\ \tau(g) = (j_1,\ldots,j_m),\ \tau(h) = (k_1,\ldots,k_n)$

2.2.1) $\mathcal{A}rgs(r) \succ_{\mathsf{IRDS}}^{\tau} \mathcal{A}rgs(s)$:
W.l.o.g. let $r_{i_1} \sim_{\mathsf{IRDS}} s_{j_1},\ldots,r_{i_{p-1}} \sim_{\mathsf{IRDS}} s_{j_{p-1}}$
$\hookrightarrow$ $\mathcal{D}ec(\{r_{i_p}\})\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\{s_{j_p}\})$
$\because$ Definition of the IRDS
$\hookrightarrow$ $\mathcal{D}ec(\{r_{i_p}\})\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\{t_{k_p}\})$
$\because$ $\mathcal{D}ec(\{s_{j_p}\})\,(\approx^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\{t_{k_p}\})$ (since $s_{j_p} \approx t_{k_p}$) and induction hypothesis
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$
$\because$ Definition of $\succ_{\mathsf{D}}$ and the fact that $\forall q \in [1, p-1]$: $s_{j_q} \approx t_{k_q}$ holds (since $s \sim_{\mathsf{D}} t$)

- ▶ $r \sim_{\mathsf{D}} s \succ_{\mathsf{D}} t$: analogous to the previous case

iii) The IRDS is transitive:
It is obvious that if an ordering is transitive and compatible with its equivalence relation then its extension to multisets preserves this property $(*)$ (see [Dershowitz and Manna, 1979]). Thus, it is sufficient to show that $\succ_{\mathsf{D}}$ is transitive: $r \succ_{\mathsf{D}} s \succ_{\mathsf{D}} t \;\hookrightarrow\; r \succ_{\mathsf{D}} t$. We prove it by induction on $|r| + |s| + |t|$. Let $r = f(r_1,\ldots,r_l)$, $s = g(s_1,\ldots,s_m)$ and $t = h(t_1,\ldots,t_n)$ with $r \in \mathcal{D}ec_u(r')$, $s \in \mathcal{D}ec_v(s')$ and $t \in \mathcal{D}ec_w(t')$. We have to consider the following disjoint cases:

▶ $f \succ g \sim h \;\vee\; f \sim g \succ h$
$\hookrightarrow$ $f \succ h$
$\because$ $\succsim$ is a quasi-precedence
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$
$\because$ Definition of $\succ_{\mathsf{D}}$

▶ $f \sim g \sim h \;\wedge\; \tau(f) = \tau(g) = \tau(h) = \textcircled{m} \;\wedge\; (\mathcal{S}ub(r) \succ_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(s) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t) \;\vee\; \mathcal{S}ub(r) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(s) \succ_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t))$
$\hookrightarrow$ $\mathcal{S}ub(r) \succ_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t)$
$\because$ $\succ_{\mathsf{D}}$ is $\sim_{\mathsf{D}}$-compatible, induction hypothesis (since $\forall \alpha \in \mathcal{S}ub(\beta)$: $|\alpha| < |\beta|$) and $(*)$
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$
$\because$ Definition of $\succ_{\mathsf{D}}$

▶ $f \sim g \sim h \;\wedge\; \tau(f) = \tau(g) = \tau(h) = \textcircled{m} \;\wedge\; \mathcal{S}ub(r) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(s) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t) \;\wedge$
$\mathcal{D}ec(\mathcal{A}rgs(r))\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\mathcal{A}rgs(s))\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\mathcal{A}rgs(t))$
$\hookrightarrow$ $\wedge\; \mathcal{D}ec(\mathcal{A}rgs(r))\,(\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\,\mathcal{D}ec(\mathcal{A}rgs(t))$
$\because$ $\succ_{\mathsf{D}}$ is $\sim_{\mathsf{D}}$-compatible, induction hypothesis (since $\forall \alpha \in \mathcal{D}ec(\mathcal{A}rgs(\beta))$: $|\alpha| < |\beta|$) and $(*)$
$\hookrightarrow$ $r \succ_{\mathsf{D}} t$

$\because$ $\mathcal{S}ub(r) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t)$ and the definition of $\succ_{\mathsf{D}}$

- ▶ $f \sim g \sim h \;\wedge\; \tau(f) = (i_1,\dots,i_l),\ \tau(g) = (j_1,\dots,j_m),\ \tau(h) = (k_1,\dots,k_n) \;\wedge$ $\mathcal{A}rgs(r) \succ^{\mathcal{T}}_{\mathsf{IRDS}} \mathcal{A}rgs(s) \succ^{\mathcal{T}}_{\mathsf{IRDS}} \mathcal{A}rgs(t)$

  $\hookrightarrow$ $\mathcal{A}rgs(r) \succ^{\mathcal{T}}_{\mathsf{IRDS}} \mathcal{A}rgs(t)$

  $\because$ $\succ_{\mathsf{IRDS}} \;\equiv\; (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}$, the lexicographic extension of an ordering is transitive if the ordering is, $\succ_{\mathsf{D}}$ is $\sim_{\mathsf{D}}$-compatible, induction hypothesis (since $\forall \alpha \in \mathcal{D}ec(\mathcal{A}rgs(\beta))$: $|\alpha| < |\beta|$) and $(*)$

  $\hookrightarrow$ $r \succ_{\mathsf{D}} t$

  $\because$ Definition of $\succ_{\mathsf{D}}$

iv) The IRDS has the subterm property:
We have to show that $t \succ_{\mathsf{IRDS}} t|_u$ if $u \in \mathcal{P}os(t) \;\wedge\; u \neq \Lambda$. Let $u \in \mathcal{P}os(t) \;\wedge\; u \neq \Lambda$. Then $t \succ_{\mathsf{RPOS}} t|_u$ (see theorem 3.62). This fact implies that $t \succ_{\mathsf{IRDS}} t|_u$ because IRDS $\supset$ RPOS (see lemma 6.4, lemma 6.8 (ii), lemma 6.9 (i) and lemma 6.7 (i)).

v) The IRDS is $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible:
The following has to be shown:

$$\forall i \in [1,n]\colon\ s \succ_{\mathsf{IRDS}} s' \;\hookrightarrow\; t := f(t_1,\dots,t_i[s]_\Lambda,\dots,t_n) \succ_{\mathsf{IRDS}} f(t_1,\dots,t_i[s']_\Lambda,\dots,t_n) =: t'.$$

Note that $t \succ_{\mathsf{IRDS}} t'$ if, and only if, $\mathcal{D}ec(\{t\})\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \mathcal{D}ec(\{t'\})$.
$\mathcal{D}ec(\{t\}) = \{\!\{\, D \cup \{t\} \mid D \in \mathcal{D}ec(\mathcal{A}rgs(t_j)), j \in [1,n], j \neq i \,\}\!\} \uplus \{\!\{\, D \cup \{t\} \mid D \in \mathcal{D}ec(\{s\}) \,\}\!\}$
and
$\mathcal{D}ec(\{t'\}) = \{\!\{\, D \cup \{t'\} \mid D \in \mathcal{D}ec(\mathcal{A}rgs(t_j)), j \in [1,n], j \neq i \,\}\!\} \uplus \{\!\{\, D \cup \{t'\} \mid D \in \mathcal{D}ec(\{s'\}) \,\}\!\}$.
Thus, we have to prove the following two assertions:

- ▶ $\mathcal{D}ec(\{s\})\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \mathcal{D}ec(\{s'\})$:
  This relation directly follows from the precondition that $s \succ_{\mathsf{IRDS}} s'$ holds and the definition of the IRDS.
- ▶ $\forall u \in \mathcal{TP}os(t) \setminus \{i.v \mid v \in \mathcal{TP}os(s)\}$: $\mathcal{D}ec_u(t) \ni t \succ_{\mathsf{D}} t' \in \mathcal{D}ec_u(t')$:
  Note that if this statement is valid, the path-decompositions of
  $$\{\!\{\, D \cup \{t\} \mid D \in \mathcal{D}ec(\mathcal{A}rgs(t_j)), j \in [1,n], j \neq i \,\}\!\}$$
  are greater than those of
  $$\{\!\{\, D \cup \{t'\} \mid D \in \mathcal{D}ec(\mathcal{A}rgs(t_j)), j \in [1,n], j \neq i \,\}\!\}.$$
  The proof is divided into two parts depending on the status of the leading operator f of t and t':

  1) $\tau(f) = \textcircled{m}$

     $\hookrightarrow$ $\mathcal{D}ec(\mathcal{A}rgs(t))\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \mathcal{D}ec(\mathcal{A}rgs(t'))$

     $\because$ $\mathcal{D}ec(\mathcal{A}rgs(t))\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \mathcal{D}ec(\mathcal{A}rgs(t'))$

     $\iff$ $\mathcal{D}ec(\{s\})\ (\succ_{\mathsf{D}}^{\textcircled{m}})^{\textcircled{m}}\ \mathcal{D}ec(\{s'\})$

     $\hookrightarrow$ $t \succ_{\mathsf{IRDS}} t'$

     $\because$ $\mathcal{S}ub(t) \sim_{\mathsf{D}}^{\textcircled{m}} \mathcal{S}ub(t')$ and the definition of the IRDS

  2) $\tau(f) = (j_1,\dots,j_n)$

     $\hookrightarrow$ $\mathcal{A}rgs(t) \succ^{\mathcal{T}}_{\mathsf{IRDS}} \mathcal{A}rgs(t')$

     $\because$ $\forall k < i$: $\mathcal{A}rgs(t) \ni t_j \approx t_j \in \mathcal{A}rgs(t')$ and
     $\mathcal{A}rgs(t) \ni t_i = s \succ_{\mathsf{IRDS}} s' = t_i \in \mathcal{A}rgs(t')$ by precondition

vi) The IRDS is σ-stable:

We have to show that $\forall\sigma$: $s \succ_{\mathsf{IRDS}} t \hookrightarrow s\sigma \succ_{\mathsf{IRDS}} t\sigma$ holds. Strictly speaking, we must prove: $(\forall v \in \mathcal{TP}os(t)\colon \exists u \in \mathcal{TP}os(s)\colon \mathcal{D}ec_u(s) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t)) \hookrightarrow (\forall v' \in \mathcal{TP}os(t\sigma)\colon \exists u' \in \mathcal{TP}os(s\sigma)\colon \mathcal{D}ec_{u'}(s\sigma) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_{v'}(t\sigma))$.

Let $v' \in \mathcal{TP}os(t\sigma)$, then $\exists i, v \in \mathbb{N}^*\colon\ v' = v.i \ \wedge\ v \in \mathcal{TP}os(t)$. Since $s \succ_{\mathsf{IRDS}} t$ there exists $u \in \mathcal{TP}os(s)$ such that $\mathcal{D}ec_u(s) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t)$. To prove that $\exists\psi \in \mathcal{TP}os(s\sigma)$: $\mathcal{D}ec_\psi(s\sigma) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_{v'}(t\sigma)$ we have to distinguish two disjoint cases:

▶ $t|_v \notin \mathcal{X}$
$\hookrightarrow\ v' = v$

1) $s|_u \notin \mathcal{X}$
$\hookrightarrow\ u \in \mathcal{TP}os(s\sigma)$
$\hookrightarrow\ \mathcal{D}ec_u(s\sigma) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t\sigma)$
$\because\ \mathcal{D}ec_u(s) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t)$
$\hookrightarrow\ \psi = u$

2) $s|_u = x \in \mathcal{X}$
$\hookrightarrow\ \forall w \in \mathcal{TP}os(x\sigma)\colon \mathcal{D}ec_{u.w}(s\sigma) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t\sigma)$
$\because\ \mathcal{D}ec_u(s) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t)$
$\hookrightarrow\ \psi = u.w'$ with $w' \in \mathcal{TP}os(x\sigma)$

▶ $t|_v = x \in \mathcal{X}$
$\hookrightarrow\ s|_u = x$
$\because$ Otherwise, $\mathcal{D}ec_u(s) \not\succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t)$
$\hookrightarrow\ v' = v.i$
$\hookrightarrow\ \mathcal{D}ec_{u.i}(s\sigma) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_{v.i}(t\sigma)$
$\because\ \mathcal{D}ec_u(s) \succ^{\textcircled{m}}_{\mathsf{D}} \mathcal{D}ec_v(t)$ and $s\sigma|_u = t\sigma|_v = x\sigma$
$\hookrightarrow\ \psi = u.i$ ♦

$\boxed{\textbf{B.5}}$ **Lemma (3.31)** *The* C-RPO *is a C-compatible σ-stable simplification ordering.* ◇

Proof: We have to show that $s \overset{*}{\leftrightarrow}_C s' \succ_{\mathsf{RPO}} t' \overset{*}{\leftrightarrow}_C t$ implies $s \succ_{\mathsf{RPO}} t$. It is sufficient to prove that $s \overset{*}{\leftrightarrow}_C s' \succ_{\mathsf{RPO}} t'$ implies $s \succ_{\mathsf{RPO}} t'$ as well as $s' \succ_{\mathsf{RPO}} t' \overset{*}{\leftrightarrow}_C t$ implies $s' \succ_{\mathsf{RPO}} t$. We will prove the first assertion by induction on $|s'| + |t'|$ (the proof of the second assertion can be performed in analogy with that of the first one):

i) $\mathcal{H}ead(s') \succ \mathcal{H}ead(t') \ \wedge\ \{s'\} \succ^{\textcircled{m}}_{\mathsf{RPO}} \mathcal{A}rgs(t')$
$\hookrightarrow$

▶ $\mathcal{H}ead(s) \succ \mathcal{H}ead(t')$
$\because\ \mathcal{H}ead(s) = \mathcal{H}ead(s')$

▶ $\{s\} \succ^{\textcircled{m}}_{\mathsf{RPO}} \mathcal{A}rgs(t')$
$\because$ Induction hypothesis

$\hookrightarrow\ s \succ_{\mathsf{RPO}} t'$
$\because$ Definition of the RPO

ii) $\mathcal{H}ead(s') \sim \mathcal{H}ead(t') \ \wedge \ \mathcal{A}rgs(s') \succ_{\mathsf{RPO}}^{\textcircled{m}} \mathcal{A}rgs(t')$
$\hookrightarrow \quad \mathcal{A}rgs(s) \succ_{\mathsf{RPO}}^{\textcircled{m}} \mathcal{A}rgs(t')$
$\because \quad \exists\pi$: $s'_i \overset{*}{\leftrightarrow}_C s_{\pi(i)}$ and induction hypothesis
$\hookrightarrow \quad s \succ_{\mathsf{RPO}} t'$
$\because \quad \mathcal{H}ead(s) = \mathcal{H}ead(s')$ and the definition of the RPO

iii) $\mathcal{A}rgs(s') \succsim_{\mathsf{RPO}}^{\textcircled{m}} \{t'\}$
$\hookrightarrow \quad \exists s'_i$: $s'_i \succsim_{\mathsf{RPO}} t'$

- ► $s'_i \sim_{\mathsf{RPO}} t'$
$\hookrightarrow \quad \exists s_j$: $s_j \sim_{\mathsf{RPO}} t'$
$\because \quad s_j \overset{*}{\leftrightarrow}_C s_i$ and $\overset{*}{\leftrightarrow}_C \subseteq \sim_{\mathsf{RPO}}$
$\hookrightarrow \quad s \succ_{\mathsf{RPO}} t'$
$\because$ Subterm property of the RPO

- ► $s'_i \succ_{\mathsf{RPO}} t'$
$\hookrightarrow \quad \exists s_j$: $s_j \overset{*}{\leftrightarrow}_C s'_i$
$\because \quad s' \overset{*}{\leftrightarrow}_C s$
$\hookrightarrow \quad s_j \succ_{\mathsf{RPO}} t'$
$\because$ Induction hypothesis
$\hookrightarrow \quad s \succ_{\mathsf{RPO}} t'$
$\because$ Definition of the RPO ♦

$\boxed{\textbf{B.6}}$ **Lemma (3.46)** *The ordering* $\succ_{\mathsf{ACD}}$ *defined by* $\succ_{\mathsf{RPOS}}\langle^{-} \circ \delta\rangle$ *with* $\tau(f) = \textcircled{m}$ *if* $f \in \mathcal{F}_{AC}$ *is ACD-compatible.* ◇

Proof: We have to prove that $s' \overset{*}{\leftrightarrow}_{ACD} s \succ_{\mathsf{ACD}} t \overset{*}{\leftrightarrow}_{ACD} t' \hookrightarrow s' \succ_{\mathsf{ACD}} t'$ holds: The two implications $s' \overset{*}{\leftrightarrow}_{ACD} s \succ_{\mathsf{ACD}} t \hookrightarrow s' \succ_{\mathsf{ACD}} t$ and $s \succ_{\mathsf{ACD}} t \overset{*}{\leftrightarrow}_{ACD} t' \hookrightarrow s \succ_{\mathsf{ACD}} t'$ suffice to prove the assertion. Since the second one is symmetrical to the first one (the proofs are very similar) we will only show the former:

$s \succ_{\mathsf{ACD}} t$
$\hookrightarrow \quad \overline{\delta(s)} \succ_{\mathsf{RPOS}} \overline{\delta(t)}$
$\because$ Definition of $\succ_{\mathsf{ACD}}$
$\hookrightarrow \quad \overline{\delta(s')} \succ_{\mathsf{RPOS}} \overline{\delta(t)}$
$\because \quad \overline{\delta(s')} \overset{*}{\leftrightarrow}_C \overline{\delta(s)}$ (since $s \overset{*}{\leftrightarrow}_D t \hookrightarrow \delta(s) \overset{*}{\leftrightarrow}_{AC} \delta(t)$, see [Gnaedig and Lescanne, 1986] and $s \overset{*}{\leftrightarrow}_A t \iff \overline{s} = \overline{t}$) and the RPOS with multiset status of AC-operators is C-compatible
$\hookrightarrow \quad s' \succ_{\mathsf{ACD}} t$
$\because$ Definition of $\succ_{\mathsf{ACD}}$ ♦

$\boxed{\textbf{B.7}}$ **Lemma (3.51)** *The* A-RPO *and the* AD-RPO *are not* σ*-stable, in general, if the precedence on the A(C)-operators is a quasi-ordering.* ◇

Proof: The proof of this lemma is performed by giving a counter-example: see example 3.50 on page 64.

Furthermore, it is easy (by studying the corresponding proofs) to show that the restriction 'no quasi-ordering on AC-operators' is sufficient for guaranteeing the stability w.r.t. substitutions. Note that the other (non-AC-) operators can be quasi-ordered. ◆

$\boxed{\textbf{B.8}}$ **Lemma (3.60)** *Let $\mathcal{F}_{AC} \subseteq \mathcal{F}$ be a set of AC-operators and $\succ$ be a precedence on $\mathcal{F}$ satisfying the extended associative path condition. Then*

(i) $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$ *is terminating and confluent modulo* $\approx$*,*

(ii) $\mathcal{F}lat(\mathcal{F}_{AC})$ *is canonical,* $\mathcal{F}lat(\mathcal{F}_{AC})/\approx$ *is terminating and* $\leftrightarrow_{AC} \subseteq \approx \cup (\rightarrow_{\mathcal{F}lat(\mathcal{F}_{AC})} \cup {}_{\mathcal{F}lat(\mathcal{F}_{AC})}\!\leftarrow)^*$*,*

(iii) $\leftrightarrow_{AC} \subseteq \approx \cup (\rightarrow_{\mathcal{T}rans(\succ,\mathcal{F}_{AC})} \cup {}_{\mathcal{T}rans(\succ,\mathcal{F}_{AC})}\!\leftarrow)^*$

(iv) $\rightarrow_{(\mathcal{T}erm(\succ_{\mathsf{RPO}},\mathcal{F}_{AC})/\mathcal{T}rans(\succ,\mathcal{F}_{AC}))/\approx}$ *is terminating and*

(v) $\xrightarrow{+}_{\mathcal{T}erm(\succ_{\mathsf{RPO}},\mathcal{F}_{AC})/\mathcal{T}rans(\succ,\mathcal{F}_{AC})}$ *and* $\xrightarrow{*}_{\mathcal{T}rans(\succ,\mathcal{F}_{AC})}$ *commute.* ◇

Proof:

(i) We have to prove that $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$ is terminating and confluent modulo $\approx$: Let $\succ$ be a precedence including the chain $f \succ g \succ h_1 \succ h_2 \succ \cdots \succ h_n$ where $f, g \in \mathcal{F}_{AC}$, $\mathcal{A}r(h_i) = 1$. Then $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$ is defined as

$$\begin{array}{lrcl}
(1) & f(x,g(y,z)) & \rightarrow & g(f(x,y),f(x,z)) \\
(2) & f(g(x,y),z) & \rightarrow & g(f(x,z),f(y,z))
\end{array}$$

$$\left.\begin{array}{lrcl}
(3) & f(x,h_i(y)) & \rightarrow & h_i(f(x,y)) \\
(4) & f(h_i(x),y) & \rightarrow & h_i(f(x,y)) \\
(5) & g(x,h_i(y)) & \rightarrow & h_i(g(x,y)) \\
(6) & g(h_i(x),y) & \rightarrow & h_i(g(x,y)) \\
(7) & h_i(h_i(g(x,y))) & \rightarrow & h_i(g(x,y))
\end{array}\right\} \forall i \in [1,n]$$

$$\begin{array}{lrcl}
(8) & h_i(h_j(x)) & \rightarrow & h_j(h_i(x)) \quad \forall i \in [1,n-1]\colon \forall j \in [i+1,n] \\
(9) & h_j(h_j(h_{i_1}(\cdots(h_{i_k}(g(x,y)))\cdots))) & \rightarrow & h_j(h_{i_1}(\cdots(h_{i_k}(g(x,y)))\cdots))
\end{array}$$

$$\left.\begin{array}{lrcl}
(10) & f(x,f(y_1,\ldots,y_k),z) & \rightarrow & f(x,y_1,\ldots,y_k,z) \\
(11) & g(x,g(y_1,\ldots,y_k),z) & \rightarrow & g(x,y_1,\ldots,y_k,z)
\end{array}\right\} x,z \in \mathcal{X},\ |x|+|z| \geq 1$$

Rule (9) holds for $1 \leq i_k < \cdots < i_1 < j \leq n, 1 \leq k \leq n-1$

- ▶ Termination modulo $\approx$:

  We take a $\approx$-compatible POL based on the lexicographic concatenation of the following three classes of interpretations:

$$\begin{array}{lrcl}
\text{(a)} & [\![f]\!](x,y) & = & xy \\
 & [\![g]\!](x,y) & = & x+y+2 \\
 & [\![h_i]\!](x) & = & x \\
\text{(b)} & [\![f]\!](x,y) & = & xy+2(x+y)+2 \\
 & [\![g]\!](x,y) & = & x+y \\
 & [\![h_i]\!](x) & = & x \\
\text{(c)} & [\![f]\!](x,y) & = & xy \\
 & [\![g]\!](x,y) & = & xy \\
 & [\![h_i]\!](x) & = & (i{+}1)x+i+1
\end{array}$$

and $\mu = 2$.
Note that the interpretations of (a) prove the termination of (1), (2) and (11) (simultaneously proving both sides of the remaining rules to be equivalent). Rule (10) is ordered w.r.t. the interpretations of (b). Finally, the interpretations of (c) prove the termination of the remaining rules.

- ▶ Confluence modulo $\approx$:
  Let $T = \mathcal{T}rans(\succ, \mathcal{F}_{AC})$. Then it is sufficient to show that

$$\begin{tikzcd}
 & & \cdot \arrow[dll, "T"'] \arrow[drr, "T"] & & \\
s \arrow[dr, "*"', "T"] & & & & t \arrow[dl, "*", "T"'] \\
 & s' & \approx & t' &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \cdot \arrow[dl, "T"'] & \approx & t \arrow[ddl, "*", "T"'] \\
s \arrow[dr, "*"', "T"] & & & \\
 & s' \approx & t' &
\end{tikzcd}$$

holds for all $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Our completion environment COMTES proves the confluence of T. The rules (7) and (9) join the critical pairs.

(ii) ▶ $\mathcal{F}lat(\mathcal{F}_{AC})$ is canonical:
Note that $\mathcal{F}lat(\mathcal{F}_{AC})$ is an infinite set of rewrite rules where the left-hand sides are of the form $f(x, f(y_1, \ldots, y_n), z)$ for all $n \geq 2$ and $|x| + |z| \geq 1$. Thus, all possible patterns are covered.
Let $f(x_1^k, f_{k+1}^m, x_{m+1}^n)$ be the term $f(x_1, \ldots, x_k, f(x_{k+1}, \ldots, x_m), x_{m+1}, \ldots, x_n)$. Since

$$f(x_1^{k_1}, f_{k_1+1}^{m_1}, x_{m_1+1}^n) \rightarrow f(x_1^n) \leftarrow f(x_1^{k_2}, f_{k_2+1}^{m_2}, x_{m_2+1}^n)$$

for all $1 \leq k_1 \leq m_1 \leq n$ and $1 \leq k_2 \leq m_2 \leq n$, $\mathcal{F}lat(\mathcal{F}_{AC})$ is confluent. The termination of $\mathcal{F}lat(\mathcal{F}_{AC})$ is obviously satisfied.

- ▶ $\mathcal{F}lat(\mathcal{F}_{AC})/\approx$ is terminating:
  The A-RPO with the empty precedence proves the termination.
- ▶ $\leftrightarrow_{AC} \subseteq\ \approx \cup (\rightarrow_{\mathcal{F}lat(\mathcal{F}_{AC})} \cup {}_{\mathcal{F}lat(\mathcal{F}_{AC})}\!\leftarrow)^*$:
  This inclusion holds since $\leftrightarrow_C \subseteq \approx$ and $\leftrightarrow_A \subseteq (\rightarrow_{\mathcal{F}lat(\mathcal{F}_{AC})} \cup {}_{\mathcal{F}lat(\mathcal{F}_{AC})}\!\leftarrow)^*$.

(iii) $\leftrightarrow_{AC} \subseteq\ \approx \cup (\rightarrow_{\mathcal{T}rans(\succ, \mathcal{F}_{AC})} \cup {}_{\mathcal{T}rans(\succ, \mathcal{F}_{AC})}\!\leftarrow)^*$:
In (ii) it is proved that $\leftrightarrow_{AC} \subseteq\ \approx \cup (\rightarrow_{\mathcal{F}lat(\mathcal{F}_{AC})} \cup {}_{\mathcal{F}lat(\mathcal{F}_{AC})}\!\leftarrow)^*$. This fact directly leads to the above inclusion since $\mathcal{F}lat(\mathcal{F}_{AC}) \subseteq \mathcal{T}rans(\succ, \mathcal{F}_{AC})$.

(iv) $\rightarrow_{(\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})/\mathcal{T}rans(\succ, \mathcal{F}_{AC}))/\approx}$ is terminating:
The RPO contains

- $\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})$
  ∵ $\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC}) \subseteq \{l \rightarrow r \mid l \succ_{\mathsf{RPO}} r\}$
- $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$
  ∵ Using the precedence $f \succ g \succ h_1 \succ \cdots \succ h_n$

and is compatible with $\approx$ (which is the permutation congruence).
Furthermore, $\mathcal{T}rans(\succ, \mathcal{F}_{AC})/\approx$ is confluent (see (i)).

(v) $\xrightarrow{+}_{\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})/\mathcal{T}rans(\succ, \mathcal{F}_{AC})}$ and $\xrightarrow{*}_{\mathcal{T}rans(\succ, \mathcal{F}_{AC})}$ commute:
For proving this assertion, we make use of the following lemma (see lemma 5 on page 11 of [Bachmair and Dershowitz, 1986]):

> Suppose that T is canonical and S/T terminates. Then $(S/T)^+$ and T^* commute, if ${}_T\!\leftarrow \cdot \rightarrow_S$ is contained in $\xrightarrow{+}_{S/T} \cdot {}_T\!\xleftarrow{*}$.

Let $S := \mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})$ and $T := \mathcal{T}rans(\succ, \mathcal{F}_{AC})$. Then[1)]

- If $l \rightarrow r \in S$ then $l' \xrightarrow{+}_S r'$ where l' and r' are T-normal forms of $f(\ldots, l, \ldots)$ and $f(\ldots, r, \ldots)$, respectively. This property holds for all $f \in \mathcal{F}$. The proof is performed by the following case distinctions:
  - $\cdot\ f \notin \mathcal{F}_{AC}$ and $f \notin \{h_i \mid i \in [1, n]\}$
  - $\cdot\ f \in \mathcal{F}_{AC}$ and f minimal
  - $\cdot\ f = h_i$ and $\exists f', g' \in \mathcal{F}_{AC}$: $f' \succ g' \succ f$
  - $\cdot\ f \in \mathcal{F}_{AC}$ and f non-minimal (proof by induction on $|f(\ldots,\ldots)| + |s| + |t|$)
- If $l \rightarrow r \in S$ then $l' \xrightarrow{+}_S r'$ where l' and r' are T-normal forms of $l\sigma$ and $r\sigma$, respectively. This property holds for all σ. The proof is performed by case distinctions on $\mathcal{H}ead(l)$ and $\mathcal{H}ead(r)$ and by induction on $|l| + |r|$. ♦

$\boxed{\textbf{B.9}}$ **Theorem (3.61)** *Let $\mathcal{F}_{AC}$ be a set of AC-operators and $\succ$ be a precedence satisfying the extended associative path condition. Then an equational TRS $\mathcal{R}/AC$ terminates if $\mathcal{R}$ is reducing relative to $\mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})/\mathcal{T}rans(\succ, \mathcal{F}_{AC})$ and $\mathcal{T}rans(\succ, \mathcal{F}_{AC})$.* ◇

Proof: Let $T := \mathcal{T}rans(\succ, \mathcal{F}_{AC})$, $S := \mathcal{T}erm(\succ_{\mathsf{RPO}}, \mathcal{F}_{AC})$ and $\mathcal{E} := \approx$. Then

- T is terminating and confluent modulo $\mathcal{E}$
  $\because$ (i) of lemma 3.60
- $(S/T)/\mathcal{E}$ is terminating
  $\because$ (iv) of lemma 3.60
- $\xrightarrow{+}_{S/T}$ and $\xrightarrow{*}_T$ commute
  $\because$ (v) of lemma 3.60

$\hookrightarrow$ $\mathcal{R}/(\mathcal{E} \cup T \cup T^{-1})$ terminates
$\because$ $\mathcal{R}$ is reducing relative to S/T and T (precondition) and lemma 3.33
$\hookrightarrow$ $\mathcal{R}/AC$ terminates
$\because$ $AC \subseteq \mathcal{E} \cup T \cup T^{-1}$ ($A \subseteq T \cup T^{-1}$, $C \subseteq \mathcal{E}$) ♦

$\boxed{\textbf{B.10}}$ **Theorem (3.62)** *Let $\mathcal{R}$ be an AC-TRS which is ordered w.r.t. the* AE-RPO. *Then $\mathcal{R}$ is AC-terminating.* ◇

Proof: This is a direct consequence of theorem 3.61 by taking the $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility and the σ-stability of the RPO into account. ♦

$\boxed{\textbf{B.11}}$ **Lemma (3.65)** *Let $\mathcal{R}$ be an AC-TRS which is ordered w.r.t. the* AE-RPOS. *Then $\mathcal{R}$ is AC-terminating.* ◇

[1)] Many thanks to Jochen Nessel and also Alfons Geser for mainly supporting me to perform this proof.

Proof: Lemma 3.60 and theorem 3.61 can also be proved whenever the RPO is replaced with the RPOS. The proofs for the RPOS are identical to those for the RPO since they are independent of the status of non-AC-operators. ♦

B.12 **Lemma (3.67)** *Let $\mathcal{R}$ be an AC-TRS which is ordered w.r.t. the* AE-IRDS. *Then $\mathcal{R}$ is AC-terminating.* ◇

Proof: In analogy with the proof of lemma 3.65, one can easily show the facts of lemma 3.60 and theorem 3.61 where the strongest path and decomposition ordering IRDS is used instead of the RPO(S). ♦

B.13 **Lemma (3.70)** *The number of all possible (consistent) total precedences of a quasi-ordering $\succsim$ for $n > 1$ operators is*

$$\sum_{k=1}^{n} k! \cdot F(k,n)$$

$$\textit{with } F(k,n) = \begin{cases} 1 & \textit{if } k = 1 \vee k = n \\ k \cdot F(k,n-1) + F(k-1,n-1) & \textit{otherwise} \end{cases}$$ ◇

Proof: Let $\mathcal{F} = \{f_i \mid i \in [1,n]\}$. Furthermore, let $f_{\{i_1,\ldots,i_k\}}$ = “$f_{i_1} \sim f_{i_2} \sim \cdots \sim f_{i_k}$” be the string consisting of *all* elements of an equivalence class w.r.t. $\sim$ such that $\forall j \in [1,n]$: $j \in \{i_1,\ldots,i_k\} \vee f_j \not\sim f_{i_1}$. We call such an $f_{\{i_1,\ldots,i_k\}}$ a *record*. Note that $f_{\{i\}}$ = “f_i”. For example, $f_1 \sim f_4 \succ f_2 \succ f_3 \sim f_5 \sim f_6$ will be represented as $f_{\{1,4\}} \succ f_{\{2\}} \succ f_{\{3,5,6\}}$.
We consider a total precedence including k records, i.e.

$$f_{M_1} \succ f_{M_2} \succ \cdots \succ f_{M_k}$$

where the M_i are sets with the properties $i \neq j \hookrightarrow M_i \cap M_j = \emptyset$ and $\bigcup M_i = \{1,\ldots,n\}$. Let $F(k,n)$ represent the number of partitions of k sets including n different elements. Then

$$F(k,n) = \begin{cases} 1 & \text{if } k = 1 \vee k = n \\ k \cdot F(k,n-1) + F(k-1,n-1) & \text{otherwise} \end{cases}$$

It is not difficult to verify this expression if the following two facts are taken into account. Let $M_1 \ldots M_k$ be a partition including n elements and let x be a new element, i.e. $x \notin \bigcup M_i$.

- There exist k different possibilities to extend the partition $M_1 \ldots M_k$:

$$M_1 \cup \{x\} M_2 \ldots M_k \text{ , \ldots , } M_1 \ldots M_{k-1} M_k \cup \{x\}.$$

- The number of partitions, where $M_1 = \{x\}$ remains unchanged, is $F(k-1,n)$.

In order to determine the number of all total quasi-precedences for n operators we have to add all *permutations* of k ($1 \leq k \leq n$) records: $\sum_{k=1}^{n} k! \cdot F(k,n)$. Note that $k!$ is responsible for computing the permutations. ♦

B.3 Proofs of Chapter 4

$\boxed{\text{B.14}}$ **Theorem (4.5)** *The* KBOS *is a σ-stable simplification ordering.* ◇

Proof: The proof of this theorem will be split into the proofs of the lemmata B.15 - B.20. ◆

$\boxed{\text{B.15}}$ **Lemma** $\succ_{\mathsf{KBOS}}$ *is* $\sim_{\mathsf{KBOS}}$*-compatible.* ◇

Proof: We have to show

$$\begin{array}{rll} & \text{i)}\ r \succ_{\mathsf{KBOS}} s \sim_{\mathsf{KBOS}} t & \hookrightarrow \quad r \succ_{\mathsf{KBOS}} t \\ \text{and} & \text{ii)}\ r \sim_{\mathsf{KBOS}} s \succ_{\mathsf{KBOS}} t & \hookrightarrow \quad r \succ_{\mathsf{KBOS}} t \end{array}$$

It is obvious that $\forall x \in \mathcal{X}$: $|r|_x \geq |t|_x$ and

$$\forall s', t' \in \mathcal{T}(\mathcal{F},\mathcal{X}):\ s' \sim_{\mathsf{KBOS}} t' \hookrightarrow \varphi(s') = \varphi(t'). \tag{$*$}$$

i) We have to consider the following disjoint cases which will be proved by induction on $|r| + |s|$.

- ▶ $\varphi(r) > \varphi(s)$
  $\hookrightarrow\ \varphi(r) > \varphi(t)$
  $\because\ (*)$
  $\hookrightarrow\ r \succ_{\mathsf{KBOS}} t$
  $\because$ Definition of the KBOS
- ▶ $\varphi(r) = \varphi(s)\ \wedge\ \mathcal{H}ead(r) \succ \mathcal{H}ead(s)$
  $\hookrightarrow\ \varphi(r) = \varphi(t)\ \wedge\ \mathcal{H}ead(r) \succ \mathcal{H}ead(t)$
  $\because$ $(*)$ and $\mathcal{H}ead(s) \sim \mathcal{H}ead(t)$
  $\hookrightarrow\ r \succ_{\mathsf{KBOS}} t$
  $\because$ Definition of the KBOS
- ▶ $\varphi(r) = \varphi(s)\ \wedge\ \mathcal{H}ead(r) \sim \mathcal{H}ead(s)$:
  Let $r = f(r_1,\ldots,r_m), s = g(s_1,\ldots,s_n)$ such that $f \sim g$. Furthermore, let $t = h(t_1,\ldots,t_n)$.
  1. $\tau(f) = (i_1,\ldots,i_m), \tau(g) = (j_1,\ldots,j_n), \tau(h) = (k_1,\ldots,k_n)$
     $\hookrightarrow\ \exists a\colon \forall b < a\colon r_{i_b} \sim_{\mathsf{KBOS}} s_{j_b}\ \wedge\ r_{i_a} \succ_{\mathsf{KBOS}} s_{j_a}$
     $\because$ Definition of $\succ^{\mathcal{T}}_{\mathsf{KBOS}}$
     $\hookrightarrow\ \forall b < a\colon r_{i_b} \sim_{\mathsf{KBOS}} t_{k_b}\ \wedge\ r_{i_a} \succ_{\mathsf{KBOS}} t_{k_a}$
     $\because$ $\sim_{\mathsf{KBOS}}$ is an equivalence relation (lemma 2.7) and induction hypothesis
     $\hookrightarrow\ r \succ_{\mathsf{KBOS}} t$
     $\because$ Definition of the KBOS
  2. $\tau(f) = \tau(g) = \tau(h) = ⓜ$
     $\hookrightarrow\ \{\!\{ r_1,\ldots,r_m \}\!\} \overset{ⓜ}{\succ}_{\mathsf{KBOS}} \{\!\{ s_1,\ldots,s_n \}\!\} \overset{ⓜ}{\sim}_{\mathsf{KBOS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
     $\because$ Definition of the KBOS and $\sim_{\mathsf{KBOS}}$
     $\hookrightarrow\ \{\!\{ r_1,\ldots,r_m \}\!\} \overset{ⓜ}{\succ}_{\mathsf{KBOS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
     $\because$ $\sim_{\mathsf{KBOS}}$ is an equivalence relation (lemma 2.7) and induction hypothesis

ii) analogous to i) ◆

$\boxed{\textbf{B.16}}$ **Lemma** *The* KBOS *is a partial ordering.* ◇

Proof: We must show

$$
\begin{array}{lllll}
\text{i)} & t \not\succ_{\mathsf{KBOS}} t & & & \text{(irreflexivity)} \\
\text{ii)} & r \succ_{\mathsf{KBOS}} s \succ_{\mathsf{KBOS}} t & \hookrightarrow & r \succ_{\mathsf{KBOS}} t & \text{(transitivity)}
\end{array}
$$

i) Assume that $t \succ_{\mathsf{KBOS}} t$

$\hookrightarrow$ $\mathcal{A}rgs(t) \succ^{\tau}_{\mathsf{KBOS}} \mathcal{A}rgs(t)$

$\because$ Definition of the KBOS

$\hookrightarrow$ $\emptyset \succ^{\tau}_{\mathsf{KBOS}} \emptyset$ irrespective of τ

$\hookrightarrow$ Contradiction

ii) It is obvious that $\forall x \in \mathcal{X}$: $|r|_x \geq |s|_x \geq |t|_x \hookrightarrow |r|_x \geq |t|_x$ since $>$ is a partial ordering on $\mathbb{N}$. We have to consider three disjoint cases which will be proved by induction on $|r|+|s|+|t|$.

- $\varphi(r) > \varphi(s) \ \vee\ \varphi(s) > \varphi(t)$

  $\hookrightarrow$ $\varphi(r) > \varphi(t)$

  $\because$ $>$ is a partial ordering on $\mathbb{N}$

  $\hookrightarrow$ $r \succ_{\mathsf{KBOS}} t$

  $\because$ Definition of the KBOS

- $\varphi(r) = \varphi(s) = \varphi(t) \ \wedge\ (\mathcal{H}ead(r) \succ \mathcal{H}ead(s) \ \vee\ \mathcal{H}ead(s) \succ \mathcal{H}ead(t))$

  $\hookrightarrow$ $\varphi(r) = \varphi(t) \ \wedge\ \mathcal{H}ead(r) \succ \mathcal{H}ead(t)$

  $\because$ $\succsim$ is a quasi-ordering on $\mathcal{F}$

  $\hookrightarrow$ $r \succ_{\mathsf{KBOS}} t$

  $\because$ Definition of the KBOS

- $\varphi(r) = \varphi(s) = \varphi(t) \ \wedge\ \mathcal{H}ead(r) \sim \mathcal{H}ead(s) \sim \mathcal{H}ead(t)$:

  Let $r = f(r_1,\ldots,r_l), s = g(s_1,\ldots,s_m), t = h(t_1,\ldots,t_n)$ such that $f \sim g \sim h$.

  1. $\tau(f) = (i_1,\ldots,i_l)$, $\tau(g) = (j_1,\ldots,j_m)$, $\tau(k) = (k_1,\ldots,k_n)$

     $\hookrightarrow$ $\exists a, c$: $\forall b < a, d < c$: $r_{i_b} \sim_{\mathsf{KBOS}} s_{j_b} \wedge s_{j_d} \sim_{\mathsf{KBOS}} t_{k_d} \wedge r_{i_a} \succ_{\mathsf{KBOS}} s_{j_a} \wedge s_{j_c} \succ_{\mathsf{KBOS}} t_{k_c}$

     α) $a \neq c$

     $\hookrightarrow$ $r_{i_a} \succ_{\mathsf{KBOS}} s_{j_a} \sim_{\mathsf{KBOS}} t_{k_a} \ \vee\ r_{i_c} \sim_{\mathsf{KBOS}} s_{j_c} \succ_{\mathsf{KBOS}} t_{k_c}$

     $\hookrightarrow$ $r_{i_a} \succ_{\mathsf{KBOS}} t_{k_a} \ \vee\ r_{i_c} \succ_{\mathsf{KBOS}} t_{k_c}$

     $\because$ $\succ_{\mathsf{KBOS}}$ is $\sim_{\mathsf{KBOS}}$-compatible (lemma B.15)

     $\hookrightarrow$ $r \succ_{\mathsf{KBOS}} t$

     $\because$ Definition of the KBOS

     β) $a = c$

     $\hookrightarrow$ $r_{i_a} \succ_{\mathsf{KBOS}} s_{j_a} \succ_{\mathsf{KBOS}} t_{k_a}$

     $\hookrightarrow$ $r_{i_a} \succ_{\mathsf{KBOS}} t_{k_a}$

     $\because$ Induction hypothesis

     $\hookrightarrow$ $r \succ_{\mathsf{KBOS}} t$

     $\because$ Definition of the KBOS

  2. $\tau(f) = \tau(g) = \tau(h) = \textcircled{m}$

     $\hookrightarrow$ $\{\!\{ r_1,\ldots,r_l \}\!\} \succ^{\textcircled{m}}_{\mathsf{KBOS}} \{\!\{ s_1,\ldots,s_m \}\!\} \succ^{\textcircled{m}}_{\mathsf{KBOS}} \{\!\{ t_1,\ldots,t_n \}\!\}$

$\because$ Definition of the KBOS
$\hookrightarrow$ $\{\!\{ r_1,\ldots,r_l \}\!\} \overset{\textcircled{m}}{\succ}_{\mathsf{KBOS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
$\because$ Induction hypothesis and $\succ_{\mathsf{KBOS}}$ is $\sim_{\mathsf{KBOS}}$-compatible (lemma B.15)
$\hookrightarrow$ $r \succ_{\mathsf{KBOS}} t$
$\because$ Definition of the KBOS ◆

$\boxed{\textbf{B.17}}$ **Lemma** *The* KBOS *has the subterm property.* ◇

Proof: We have to show that $t \succ_{\mathsf{KBOS}} t|_u$ if $u \in \mathcal{P}os(t) \wedge u \neq \Lambda$. Let us consider a term t and an occurrence $u \in \mathcal{P}os(t), \mathcal{H}ead(t) = f$ and $\mathcal{H}ead(t|_u) = g$. It is obvious that $\forall x \in \mathcal{X}$: $|t|_x \geq |t|_u|_x$. We must distinguish two cases which will be proved by induction on $|t|$:

- ▶ $\varphi(t) > \varphi(t|_u)$
  $\hookrightarrow$ $t \succ_{\mathsf{KBOS}} t|_u$
  $\because$ Definition of the KBOS

- ▶ $\varphi(t) = \varphi(t|_u)$
  $\hookrightarrow$ $\varphi(f) = 0 \wedge \mathcal{A}r(f) = 1$
  $\because$ Definition of the KBOS

  α) $f \not\sim g$
  $\hookrightarrow$ $f \succ g$
  $\because$ Definition of φ and $\succ$
  $\hookrightarrow$ $t \succ_{\mathsf{KBOS}} t|_u$
  $\because$ Definition of the KBOS

  β) $f \sim g$:
  Let $t = f(t')$ and $t|_u = g(t'')$.
  $\hookrightarrow$ $t' \succ_{\mathsf{KBOS}} t''$
  $\because$ Induction hypothesis since t' (t'') is a proper subterm of t (t', resp.)
  $\hookrightarrow$ $t \succ_{\mathsf{KBOS}} t|_u$
  $\because$ Definition of the KBOS ◆

$\boxed{\textbf{B.18}}$ **Lemma** *The* KBOS *is* $\mathcal{T}(\mathcal{F},\mathcal{X})$*-compatible.* ◇

Proof: We have to show:

$$\forall f \in \mathcal{F}\colon \forall i \in [1,n]\colon s \succ_{\mathsf{KBOS}} s' \hookrightarrow t := f(t_1,\ldots,t_i[s]_\Lambda,\ldots,t_n) \succ_{\mathsf{KBOS}} f(t_1,\ldots,t_i[s']_\Lambda,\ldots,t_n) =: t'.$$

Obviously, $\forall x \in \mathcal{X}$: $|s|_x \geq |s'|_x \hookrightarrow |t|_x \geq |t'|_x$. We have to examine two disjoint cases:

- ▶ $\varphi(s) > \varphi(s')$
  $\hookrightarrow$ $\varphi(t) > \varphi(t')$
  $\hookrightarrow$ $t \succ_{\mathsf{KBOS}} t'$
  $\because$ Definition of the KBOS

- ▶ $\varphi(s) = \varphi(s')$
  $\hookrightarrow$ $\varphi(t) = \varphi(t')$
  $\hookrightarrow$ $\mathcal{A}rgs(t) \overset{\mathcal{T}}{\succ}_{\mathsf{KBOS}} \mathcal{A}rgs(t')$

$\because$ The arguments of t and t' are equal except the t_i's (thus, this fact holds irrespective of τ)

$\hookrightarrow$ $t \succ_{\mathsf{KBOS}} t'$

$\because$ Definition of the KBOS ◆

$\boxed{\textbf{B.19}}$ **Lemma** *Let* $s, t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ *and* $\forall x \in \mathcal{X}$: $|s|_x \geq |t|_x$. *Then*

$$\forall \sigma: \quad \begin{array}{rl} \textit{(i)} & \forall x \in \mathcal{X}: |s\sigma|_x \geq |t\sigma|_x \\ \textit{(ii)} & \varphi(s) > \varphi(t) \hookrightarrow \varphi(s\sigma) > \varphi(t\sigma) \\ \textit{(iii)} & \varphi(s) = \varphi(t) \hookrightarrow \varphi(s\sigma) \geq \varphi(t\sigma) \end{array}$$ ◇

Proof:

(i) Let $x \in \mathcal{X}$: We have to show that $|s\sigma|_x - |t\sigma|_x \geq 0$. Note that $|s\sigma|_x = \sum\limits_{y \in \mathcal{X}} |s|_y \cdot |y\sigma|_x$.

$$\begin{array}{lrcl} \hookrightarrow & |s\sigma|_x - |t\sigma|_x & = & \sum\limits_{y \in \mathcal{X}} |s|_y \cdot |y\sigma|_x - \sum\limits_{y \in \mathcal{X}} |t|_y \cdot |y\sigma|_x \\ & & = & \sum\limits_{y \in \mathcal{X}} (|s|_y - |t|_y) \cdot |y\sigma|_x \\ & & \geq & 0 \end{array}$$

$\because$ $\forall x \in \mathcal{X}$: $|s|_x \geq |t|_x$

(ii) We must prove that $\varphi(s\sigma) - \varphi(t\sigma) > 0$. Note that $\varphi(s\sigma) = \varphi(s) + \sum\limits_{x \in \mathcal{X}} |s|_x \cdot (\varphi(x\sigma) - \varphi_0)$.

$$\begin{array}{lrcl} \hookrightarrow & \varphi(s\sigma) - \varphi(t\sigma) & = & \varphi(s) + \sum\limits_{x \in \mathcal{X}} |s|_x \cdot (\varphi(x\sigma) - \varphi_0) - \varphi(t) - \sum\limits_{x \in \mathcal{X}} |t|_x \cdot (\varphi(x\sigma) - \varphi_0) \\ & & = & \varphi(s) - \varphi(t) + \sum\limits_{x \in \mathcal{X}} (\varphi(x\sigma) - \varphi_0) \cdot (|s|_x - |t|_x) \\ & & > & 0 \end{array}$$

$\because$ $\varphi(s) - \varphi(t) > 0$ (precondition), $\varphi(x\sigma) - \varphi_0 \geq 0$ (since $\varphi(x) = \varphi_0$) and $|s|_x - |t|_x \geq 0$ (precondition)

(iii) analogous to (ii) ◆

$\boxed{\textbf{B.20}}$ **Lemma** *The* KBOS *is* σ*-stable.* ◇

Proof: We have to show that $\forall \sigma$: $s \succ_{\mathsf{KBOS}} t \hookrightarrow s\sigma \succ_{\mathsf{KBOS}} t\sigma$ holds. This will be accomplished by induction on $|s| + |t|$. Let $s = f(s_1, \dots, s_m)$ and $t = g(t_1, \dots, t_n)$. Note that the variable condition is fulfilled: $|s\sigma|_x \geq |t\sigma|_x$ (see (i) of lemma B.19). We have to examine the following four disjoint cases:

▶ $\varphi(s) > \varphi(t)$

$\hookrightarrow$ $\varphi(s\sigma) > \varphi(t\sigma)$

$\because$ (ii) of lemma B.19

$\hookrightarrow$ $s\sigma \succ_{\mathsf{KBOS}} t\sigma$

$\because$ Definition of the KBOS

▶ $\varphi(s) = \varphi(t) \quad \wedge \quad f \succ g$

$\hookrightarrow$ $\varphi(s\sigma) \geq \varphi(t\sigma) \wedge \mathcal{H}ead(s\sigma) = f \succ g = \mathcal{H}ead(t\sigma)$

$\because$ (iii) of lemma B.19

$\hookrightarrow$ $s\sigma \succ_{\mathsf{KBOS}} t\sigma$

$\because$ Definition of the KBOS

- $\varphi(s) = \varphi(t) \;\wedge\; f \sim g \;\wedge\; \tau(f) = (i_1, \ldots, i_m), \tau(g) = (j_1, \ldots, j_n)$
  $\hookrightarrow \quad \exists k\colon \forall l < k\colon s_{i_l} \sim_{\mathsf{KBOS}} t_{j_l} \;\wedge\; s_{i_k} \succ_{\mathsf{KBOS}} t_{j_k}$
  $\because$ Definition of $\succ^{\tau}_{\mathsf{KBOS}}$
  $\hookrightarrow \quad \forall l < k\colon s_{i_l}\sigma \sim_{\mathsf{KBOS}} t_{j_l}\sigma \;\wedge\; s_{i_k}\sigma \succ_{\mathsf{KBOS}} t_{j_k}\sigma$
  $\because$ Lemma 2.7 and induction hypothesis
  $\hookrightarrow \quad s\sigma \succ_{\mathsf{KBOS}} t\sigma$
  $\because$ Definition of the KBOS

- $\varphi(s) = \varphi(t) \;\wedge\; f \sim g \;\wedge\; \tau(f) = \tau(g) = \textcircled{m}$
  $\hookrightarrow \quad \{\!\{ s_1, \ldots, s_m \}\!\} \succ^{\textcircled{m}}_{\mathsf{KBOS}} \{\!\{ t_1, \ldots, t_n \}\!\}$
  $\because$ Definition of $\succ^{\tau}_{\mathsf{KBOS}}$
  $\hookrightarrow \quad \{\!\{ s_1\sigma, \ldots, s_m\sigma \}\!\} \succ^{\textcircled{m}}_{\mathsf{KBOS}} \{\!\{ t_1\sigma, \ldots, t_n\sigma \}\!\}$
  $\because$ Lemma 2.7 and induction hypothesis
  $\hookrightarrow \quad s\sigma \succ_{\mathsf{KBOS}} t\sigma$
  $\because$ Definition of the KBOS ♦

$\boxed{\textbf{B.21}}$ **Theorem (4.10)** *The* ACKM *is an AC-commuting σ-stable simplification ordering.* ◇

Proof: The proof of the ACKM being a simplification ordering is performed in analogy with the KBOS being a simplification ordering since $\succ_{\mathsf{ACKM}} \equiv \succ_{\mathsf{KBOS}}\langle\tilde{\ }\rangle$.

- The ACKM has the deletion property, i.e. $s := f(\ldots, r, \ldots) \succ_{\mathsf{ACKM}} f(\ldots, \ldots) =: t$:
  It is obvious that $\forall x \in \mathcal{X}\colon |s|_x \geq |t|_x$ holds.
  $\hookrightarrow \quad \forall x \in \mathcal{X}\colon |\tilde{s}|_x \geq |\tilde{t}|_x \qquad (*)$
  $\because$ $\tilde{\ }$ does not remove leaves of terms
  Note that $\varphi(s) > \varphi(t)$.
  $\because$ t results from s by removing at least one leaf (and the constants and variables have a positive weight).
  $\varphi(\tilde{s}) > \varphi(\tilde{t})$
  $\because$ $\forall t\colon \varphi(t) = \varphi(\tilde{t})$ by definition of φ
  $\hookrightarrow \quad \tilde{s} \succ_{\mathsf{KBOS}} \tilde{t}$
  $\because$ Definition of the KBOS and $(*)$
  $\hookrightarrow \quad s \succ_{\mathsf{ACKM}} t$
  $\because$ Definition of the ACKM

- The ACKM is σ-stable, i.e. $\forall\sigma\colon s \succ_{\mathsf{ACKM}} t \hookrightarrow s\sigma \succ_{\mathsf{ACKM}} t\sigma$:
  Note that the variable condition is fulfilled, i.e. $\forall x \in \mathcal{X}\colon |\tilde{s}|_x \geq |\tilde{t}|_x \hookrightarrow |\widetilde{s\sigma}|_x \geq |\widetilde{t\sigma}|_x$.
  The remaining proof, i.e.

$$\tilde{s} \succ_{\mathsf{KBOS}} \tilde{t} \quad \hookrightarrow \quad \widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$$

  will be performed by induction on $|\tilde{s}| + |\tilde{t}|$:

  1. $\varphi(\tilde{s}) > \varphi(\tilde{t})$
     $\hookrightarrow \quad \varphi(s) > \varphi(t)$
     $\because$ Definition of $\tilde{\ }$
     $\hookrightarrow \quad \varphi(s\sigma) > \varphi(t\sigma)$

$\because$ Lemma B.19 (ii)

$\hookrightarrow$ $\varphi(\widetilde{s\sigma}) > \varphi(\widetilde{t\sigma})$

$\because$ $\forall t$: $\varphi(t) = \varphi(\widetilde{t})$ by definition of $\widetilde{\ }$

$\hookrightarrow$ $\widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$

$\because$ Definition of the KBOS

2. $\varphi(\widetilde{s}) = \varphi(\widetilde{t}) \quad \wedge \quad \mathcal{H}ead(\widetilde{s}) \succ \mathcal{H}ead(\widetilde{t})$

$\hookrightarrow$ $\varphi(s) = \varphi(t)$

$\because$ Definition of $\widetilde{\ }$

$\hookrightarrow$ $\varphi(s\sigma) \geq \varphi(t\sigma)$

$\because$ Lemma B.19 (iii)

- $\varphi(s\sigma) > \varphi(t\sigma)$

  $\hookrightarrow$ $\widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$

  $\because$ see 1.

- $\varphi(s\sigma) = \varphi(t\sigma)$

  $\hookrightarrow$ $\varphi(\widetilde{s\sigma}) = \varphi(\widetilde{t\sigma})$

  $\because$ $\forall t$: $\varphi(t) = \varphi(\widetilde{t})$ by definition of $\widetilde{\ }$

  $\hookrightarrow$ $\widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$

  $\because$ $\mathcal{H}ead(\widetilde{s\sigma}) \succ \mathcal{H}ead(\widetilde{t\sigma})$ since $\mathcal{H}ead(\widetilde{s}) \succ \mathcal{H}ead(\widetilde{t})$

3. $\varphi(\widetilde{s}) = \varphi(\widetilde{t}) \quad \wedge \quad \mathcal{H}ead(\widetilde{s}) \sim \mathcal{H}ead(\widetilde{t})$

$\hookrightarrow$ $\varphi(s\sigma) \geq \varphi(t\sigma)$

$\because$ see 2.

- $\varphi(s\sigma) > \varphi(t\sigma)$

  $\hookrightarrow$ $\widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$

  $\because$ see 1.

- $\varphi(s\sigma) = \varphi(t\sigma)$:

  α) $\mathcal{H}ead(\widetilde{s}) \notin \mathcal{F}_{AC}$

  $\hookrightarrow$ $\mathcal{A}rgs(\widetilde{s}) \succ^{\mathcal{T}}_{\mathsf{KBOS}} \mathcal{A}rgs(\widetilde{t})$

  $\because$ $\widetilde{s} \succ_{\mathsf{KBOS}} \widetilde{t}$

  $\hookrightarrow$ $\mathcal{A}rgs(\widetilde{s\sigma}) \succ^{\mathcal{T}}_{\mathsf{KBOS}} \mathcal{A}rgs(\widetilde{t\sigma})$

  $\because$ Induction hypothesis and the fact that $\widetilde{s} \sim_{\mathsf{KBOS}} \widetilde{t} \hookrightarrow \widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$ holds

  β) $\mathcal{H}ead(\widetilde{s}) \in \mathcal{F}_{AC}$:

  Let $s = f(s_1,\ldots,s_n)$, $t = f(t_1,\ldots,t_n)$. Note that $\mathcal{H}ead(s)$ and $\mathcal{H}ead(t)$ must be syntactically identical by definition of the ACKM. Let $\widetilde{s} = f^i(s'_1,\ldots,s'_k)$ and $\widetilde{t} = f^j(t'_1,\ldots,t'_l)$. Assume that $t'_p = x \in \mathcal{X}$ (for some $p \in [1,l]$).

  $\hookrightarrow$ $\exists s'_j$: $x \in \mathcal{V}ar(s'_j)$

  $\because$ Otherwise, $\widetilde{s} \not\succ_{\mathsf{KBOS}} \widetilde{t}$

  1) $s'_j = x$

  $\hookrightarrow$ $s'_j\sigma = t'_p\sigma$

  $\hookrightarrow$ Both terms will be removed when comparing the arguments of $\widetilde{s\sigma}$ and $\widetilde{t\sigma}$

  2) $s'_j \succ_{\mathsf{EMB}} x$:

  – Let $x \leftarrow f(\widetilde{r}_1, \widetilde{r}_2)$.

  $\hookrightarrow$ $\widetilde{s'_j\sigma} \succ_{\mathsf{KBOS}} \widetilde{r_1\sigma} \quad \wedge \quad \widetilde{s'_j\sigma} \succ_{\mathsf{KBOS}} \widetilde{r_2\sigma}$

  $\because$ $x \in \mathcal{V}ar(s'_j)$

– Let $x \leftarrow g(\widetilde{r_1},\ldots,\widetilde{r_m})$, $f \not\succ g$.
$\hookrightarrow$ $s'_j\sigma \succ_{\mathsf{KBOS}} g(\widetilde{r_1\sigma},\ldots,\widetilde{r_m\sigma})$
$\because$ σ-stability of the KBOS

$\hookrightarrow$ W.l.o.g. let $t'_p \notin \mathcal{X}$ for all $p \in [1,l]$.
$\hookrightarrow$ $\widetilde{t\sigma} = f^j(\widetilde{t'_1\sigma},\ldots,\widetilde{t'_l\sigma})$
$\because$ $\widetilde{t} = f^j(t'_1,\ldots,t'_l)$ and thus, $\forall p \in [1,l]$: $\mathcal{H}ead(t'_p) \neq f$
$\hookrightarrow$ $\mathcal{A}rgs(\widetilde{s\sigma}) \overset{\textcircled{m}}{\succ}_{\mathsf{KBOS}} \{\!\{\, \widetilde{t'_1\sigma},\ldots,\widetilde{t'_l\sigma} \,\}\!\}$
$\because$ $\mathcal{A}rgs(\widetilde{s}) \overset{\textcircled{m}}{\succ}_{\mathsf{KBOS}} \{\!\{\, t'_1,\ldots,t'_l \,\}\!\}$ and induction hypothesis
$\hookrightarrow$ $\widetilde{s\sigma} \succ_{\mathsf{KBOS}} \widetilde{t\sigma}$
$\because$ Definition of $\widetilde{t\sigma}$ and the definition of the KBOS

- The ACKM is AC-commuting, i.e. $r \overset{*}{\leftrightarrow}_{AC} s \succ_{\mathsf{ACKM}} t \;\hookrightarrow\; \exists t'$: $r \succ_{\mathsf{ACKM}} t' \overset{*}{\leftrightarrow}_{AC} t$:
$r \overset{*}{\leftrightarrow}_{AC} s \succ_{\mathsf{ACKM}} t$
$\hookrightarrow$ $r \sim_{\mathsf{ACKM}} s \succ_{\mathsf{ACKM}} t$
$\because$ $r \overset{*}{\leftrightarrow}_{AC} s \iff \widetilde{r} \approx \widetilde{s} \iff \widetilde{r} \sim_{\mathsf{KBOS}} \widetilde{s} \iff r \sim_{\mathsf{ACKM}} s$
$\hookrightarrow$ $\widetilde{r} \sim_{\mathsf{KBOS}} \widetilde{s} \succ_{\mathsf{KBOS}} \widetilde{t}$
$\because$ Definition of the ACKM
$\hookrightarrow$ $\widetilde{r} \succ_{\mathsf{KBOS}} \widetilde{t}$
$\because$ Lemma B.15
$\hookrightarrow$ $r \succ_{\mathsf{ACKM}} t$
$\because$ Definition of the ACKM
$\hookrightarrow$ $t' = t$
$\because$ $t' = t \;\hookrightarrow\; t' \overset{*}{\leftrightarrow}_{AC} t$ ◆

B.22 **Theorem (4.12)** *The* IACK *is an AC-commuting σ-stable simplification ordering.* ◇

Proof: The proof can easily be performed by using the following facts:

- $\forall x \in \mathcal{X}$: $|\widetilde{s}|_x \geq |\widetilde{t}|_x \iff |s|_x \geq |t|_x$
$\because$ Definition of $\widetilde{\ }$
- $\varphi(\widetilde{s}) > \varphi(\widetilde{t}) \iff \varphi(s) > \varphi(t)$ and
$\varphi(\widetilde{s}) = \varphi(\widetilde{t}) \iff \varphi(s) = \varphi(t)$
$\because$ Definition of $\widetilde{\ }$
- $\mathcal{H}ead(\widetilde{t}) = \mathcal{H}ead(t) \;\wedge\; \succsim$ is a quasi-ordering on $\mathcal{F}^E$ ◆

B.23 **Lemma (4.14)** *The* CK *is a C-commuting σ-stable simplification ordering.* ◇

Proof: The CK is a σ-stable simplification ordering since $\succ_{\mathsf{CK}} \equiv \succ_{\mathsf{KBOS}}\langle\mathcal{I}d\rangle$ and the KBOS is a σ-stable simplification ordering (see lemma 4.5). Thus, we have to prove the C-commutation of the CK. We do it by induction on $|t|$:

$$s' \overset{*}{\leftrightarrow}_C s \succ_{\mathsf{CK}} t \;\hookrightarrow\; \exists t'\colon s' \succ_{\mathsf{CK}} t' \overset{*}{\leftrightarrow}_C t$$

i) $\varphi(s) > \varphi(t)$

$\hookrightarrow \quad \varphi(s') > \varphi(t)$
$\quad \because \quad \varphi(s') = \varphi(s)$ since $s' \overset{*}{\leftrightarrow}_C s$
$\hookrightarrow \quad t' = t$
$\quad \because$ Definition of the KBOS

ii) $\varphi(s) = \varphi(t) \quad \wedge \quad \mathcal{H}ead(s) \succ \mathcal{H}ead(t)$
$\hookrightarrow \quad \varphi(s') = \varphi(t) \quad \wedge \quad \mathcal{H}ead(s') \succ \mathcal{H}ead(t)$
$\quad \because \quad \varphi(s') = \varphi(s) \quad \wedge \quad \mathcal{H}ead(s) \sim \mathcal{H}ead(s')$ since $s' \overset{*}{\leftrightarrow}_C s$
$\hookrightarrow \quad t' = t$
$\quad \because$ Definition of the KBOS

iii) $\varphi(s) = \varphi(t) \quad \wedge \quad \mathcal{H}ead(s) \sim \mathcal{H}ead(t)$:

- $\mathcal{H}ead(s) \notin \mathcal{F}_C$
  $\hookrightarrow \quad \mathcal{A}rgs(s) \succ^{\mathcal{T}}_{\mathsf{KBOS}} \mathcal{A}rgs(t)$
  $\quad \because$ Definition of the KBOS
  $\hookrightarrow \quad \mathcal{A}rgs(s') \succ^{\mathcal{T}}_{\mathsf{KBOS}} \mathcal{A}rgs(t)$
  $\quad \because$ Induction hypothesis
  $\hookrightarrow \quad t' = t$
  $\quad \because$ Definition of the KBOS
- $\mathcal{H}ead(s) \in \mathcal{F}_C$
  $\hookrightarrow \quad \tau(\mathcal{H}ead(s)) = \text{ⓜ}$
  $\quad \because$ Definition of the CK
  $\hookrightarrow \quad \mathcal{A}rgs(s) \succ^{\text{ⓜ}}_{\mathsf{KBOS}} \mathcal{A}rgs(t)$
  $\quad \because$ Definition of the KBOS
  $\hookrightarrow \quad \mathcal{A}rgs(s') \succ^{\text{ⓜ}}_{\mathsf{KBOS}} \mathcal{A}rgs(t)$
  $\quad \because$ Induction hypothesis and the fact that $s' \overset{*}{\leftrightarrow}_C s$ holds
  $\hookrightarrow \quad t' = t$
  $\quad \because$ Definition of the KBOS ◆

B.4 Proofs of Chapter 5

$\boxed{\text{B.24}}$ **Lemma (5.13)** *The* SPOL *is a σ-stable simplification ordering.* ◇

Proof:

- The SPOL is irreflexive, i.e. $t \not\succ_{\mathsf{SPOL}} t$.
  Assume that $t \succ_{\mathsf{SPOL}} t$ holds.
  $\hookrightarrow \quad [\![t]\!] \sqsupset [\![t]\!]$
  $\quad \because \quad t \not\succ_{\mathsf{EMB}} t$
  $\hookrightarrow$ Contradiction
  $\quad \because \quad [\![t]\!] = [\![t]\!]$
- The SPOL is transitive, i.e. $r \succ_{\mathsf{SPOL}} s \succ_{\mathsf{SPOL}} t \hookrightarrow r \succ_{\mathsf{SPOL}} t$.

  i) $r \succ_{\mathsf{EMB}} s \succ_{\mathsf{EMB}} t$
  $\hookrightarrow \quad r \succ_{\mathsf{EMB}} t$
  $\quad \because$ Transitivity of the EMB

ii) $r \succ_{\mathsf{EMB}} s \;\wedge\; \llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
$\hookrightarrow\; \llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
$\because\; r \succ_{\mathsf{EMB}} s \;\hookrightarrow\; \llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket$
$\hookrightarrow\; \llbracket r \rrbracket \sqsupset \llbracket t \rrbracket$

iii) $\llbracket r \rrbracket \sqsupset \llbracket s \rrbracket \;\wedge\; s \succ_{\mathsf{EMB}} t$: analogous to ii)

iv) $\llbracket r \rrbracket \sqsupset \llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
$\hookrightarrow\; \llbracket r \rrbracket \sqsupset \llbracket t \rrbracket$
$\because\; \sqsupset$ is transitive

- The SPOL has the subterm property since the EMB is a simplification ordering.

- The SPOL is $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible, i.e. $\forall f \in \mathcal{F}, \forall s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})$: $s \succ_{\mathsf{SPOL}} t \;\hookrightarrow\; f(\ldots,s,\ldots) \succ_{\mathsf{SPOL}} f(\ldots,t,\ldots)$.

i) $s \succ_{\mathsf{EMB}} t$
$\hookrightarrow\; f(\ldots,s,\ldots) \succ_{\mathsf{EMB}} f(\ldots,t,\ldots)$
$\because$ Definition of the EMB

ii) $\llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
$\hookrightarrow\; \llbracket f(\ldots,s,\ldots) \rrbracket \sqsupset \llbracket f(\ldots,t,\ldots) \rrbracket$
$\because$ Definition of $\sqsupset$

- The SPOL is σ-stable, i.e. $\forall \sigma$: $s\sigma \succ_{\mathsf{SPOL}} t\sigma$ if $s \succ_{\mathsf{SPOL}} t$.

i) $s \succ_{\mathsf{EMB}} t$
$\hookrightarrow\; s\sigma \succ_{\mathsf{EMB}} t\sigma$
$\because$ σ-stability of the EMB

ii) $\llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
$\hookrightarrow\; \llbracket s\sigma \rrbracket \sqsupset \llbracket t\sigma \rrbracket$
$\because$ Definition of the σ-stability of $\sqsupset$ ◆

B.25 **Lemma (5.15)** *The* POLS *is a reduction ordering.* ◇

Proof: For being a reduction ordering on $\mathcal{G}(\mathcal{F})$ we have to prove the irreflexivity, the transitivity, the $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatibility, the σ-stability and the well-foundedness.

i) The POLS is irreflexive:
We have to show that $\forall t \in \mathcal{T}(\mathcal{F},\mathcal{X})$: $t \not\succ_{\mathsf{POLS}} t$ holds which is true since $\sqsupset$ and $\succ$ are irreflexive.

ii) $\succ_{\mathsf{POLS}}$ is $\sim_{\mathsf{POLS}}$-compatible:
We prove that $(r \succ_{\mathsf{POLS}} s \sim_{\mathsf{POLS}} t \;\vee\; r \sim_{\mathsf{POLS}} s \succ_{\mathsf{POLS}} t) \;\hookrightarrow\; r \succ_{\mathsf{POLS}} t$ holds by induction on $|r|$.

- $r \succ_{\mathsf{POLS}} s \sim_{\mathsf{POLS}} t$

α) $\llbracket r \rrbracket \sqsupset \llbracket s \rrbracket$
$\hookrightarrow\; \llbracket r \rrbracket \sqsupset \llbracket t \rrbracket$
$\because\; \llbracket s \rrbracket = \llbracket t \rrbracket$ by definition of $\sim_{\mathsf{POLS}}$
$\hookrightarrow\; r \succ_{\mathsf{POLS}} t$
$\because$ Definition of the POLS

$\beta)$ $\llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket \;\wedge\; \mathcal{H}ead(r) \succ \mathcal{H}ead(s)$
  $\hookrightarrow$ $\llbracket r \rrbracket \sqsupseteq \llbracket t \rrbracket \;\wedge\; \mathcal{H}ead(r) \succ \mathcal{H}ead(t)$
    $\because$ $\llbracket s \rrbracket = \llbracket t \rrbracket \;\wedge\; \mathcal{H}ead(s) \sim \mathcal{H}ead(t)$ by definition of $\sim_{\mathsf{POLS}}$
  $\hookrightarrow$ $r \succ_{\mathsf{POLS}} t$
    $\because$ Definition of the POLS

$\gamma)$ $\llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket \;\wedge\; \mathcal{H}ead(r) \sim \mathcal{H}ead(s) \;\wedge\; \mathcal{A}rgs(r) \succ^{\mathcal{T}}_{\mathsf{POLS}} \mathcal{A}rgs(s)$:
Let $r = f(r_1,\ldots,r_m)$, $s = g(s_1,\ldots,s_n)$, $t = h(t_1,\ldots,t_n)$ such that $f \sim g \sim h$.

- $\tau(f) = \tau(g) = \tau(h) = \textcircled{m}$
  $\hookrightarrow$ $\{\!\{ r_1,\ldots,r_m \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ s_1,\ldots,s_n \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
  $\hookrightarrow$ $\{\!\{ r_1,\ldots,r_m \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
    $\because$ Induction hypothesis
  $\hookrightarrow$ $r \succ_{\mathsf{POLS}} t$
    $\because$ Definition of the POLS
- $\tau(f) = (i_1,\ldots,i_m)$, $\tau(g) = (j_1,\ldots,j_n)$, $\tau(h) = (k_1,\ldots,k_n)$
  $\hookrightarrow$ $\exists p\colon \forall q<p\colon r_{i_q} \sim_{\mathsf{POLS}} s_{j_q} \;\wedge\; r_{i_p} \succ_{\mathsf{POLS}} s_{j_p}$ and $\forall q \leq p\colon s_{j_q} \sim_{\mathsf{POLS}} t_{k_q}$
    $\because$ Definitions of $\succ^{\mathcal{T}}_{\mathsf{POLS}}$ and $\sim^{\mathcal{T}}_{\mathsf{POLS}}$
  $\hookrightarrow$ $\exists p\colon \forall q<p\colon r_{i_q} \sim_{\mathsf{POLS}} t_{k_q} \;\wedge\; r_{i_p} \succ_{\mathsf{POLS}} t_{k_p}$
    $\because$ $\sim_{\mathsf{POLS}}$ is an equivalence relation and induction hypothesis
  $\hookrightarrow$ $r \succ_{\mathsf{POLS}} t$
    $\because$ Definition of the POLS

► $r \sim_{\mathsf{POLS}} s \succ_{\mathsf{POLS}} t$: analogous to the previous case

iii) The POLS is transitive:
We must prove that $r \succ_{\mathsf{POLS}} s \succ_{\mathsf{POLS}} t \hookrightarrow r \succ_{\mathsf{POLS}} t$ holds. Proving this assertion by induction on $|r|$, we have to consider three disjoint cases.

► $\llbracket r \rrbracket \sqsupset \llbracket s \rrbracket \;\vee\; \llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
  $\hookrightarrow$ $\llbracket r \rrbracket \sqsupset \llbracket t \rrbracket$
    $\because$ $\sqsupset$ is transitive and the fact that $\llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket \sqsupset \llbracket t \rrbracket \vee \llbracket r \rrbracket \sqsupset \llbracket s \rrbracket \sqsupseteq \llbracket t \rrbracket$ implies $\llbracket r \rrbracket \sqsupset \llbracket t \rrbracket$

► $\llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket \sqsupseteq \llbracket t \rrbracket \;\wedge\; (\mathcal{H}ead(r) \succ \mathcal{H}ead(s) \vee \mathcal{H}ead(s) \succ \mathcal{H}ead(t))$
  $\hookrightarrow$ $\llbracket r \rrbracket \sqsupseteq \llbracket t \rrbracket \;\wedge\; \mathcal{H}ead(r) \succ \mathcal{H}ead(t)$
    $\because$ $\sqsupseteq$ and $\succsim$ are transitive

► $\llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket \sqsupseteq \llbracket t \rrbracket \;\wedge\; \mathcal{H}ead(r) \sim \mathcal{H}ead(s) \sim \mathcal{H}ead(t)$:
Let $r = f(r_1,\ldots,r_l)$, $s = g(s_1,\ldots,s_m)$, $t = h(t_1,\ldots,t_n)$ such that $f \sim g \sim h$.

$\alpha)$ $\tau(f) = \tau(g) = \tau(h) = \textcircled{m}$
  $\hookrightarrow$ $\{\!\{ r_1,\ldots,r_l \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ s_1,\ldots,s_m \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
  $\hookrightarrow$ $\forall j\colon \exists i\colon s_i \succsim_{\mathsf{POLS}} t_j \wedge \forall k\colon \exists p\colon r_p \succsim_{\mathsf{POLS}} s_k$

- $r_p \succ_{\mathsf{POLS}} s_i \succ_{\mathsf{POLS}} t_j$
  $\hookrightarrow$ $r_p \succ_{\mathsf{POLS}} t_j$
    $\because$ Induction hypothesis
- $r_p \succ_{\mathsf{POLS}} s_i \sim_{\mathsf{POLS}} t_j \;\vee\; r_p \sim_{\mathsf{POLS}} s_i \succ_{\mathsf{POLS}} t_j$
  $\hookrightarrow$ $r_p \succ_{\mathsf{POLS}} t_j$
    $\because$ ii)
- $r_p \sim_{\mathsf{POLS}} s_i \sim_{\mathsf{POLS}} t_j$
  $\hookrightarrow$ r_p will be removed from all multisets (by definition of $\succ^{\textcircled{m}}_{\mathsf{POLS}}$)

β) $\tau(f) = (i_1,\ldots,i_l)$, $\tau(g) = (j_1,\ldots,j_m)$, $\tau(h) = (k_1,\ldots,k_n)$
$\hookrightarrow$ $\exists p, q\colon\ \forall u < p\,, \forall v < q\colon\ r_{i_u} \sim_{\mathsf{POLS}} s_{j_u} \ \wedge\ s_{j_v} \sim_{\mathsf{POLS}} t_{k_v} \ \wedge\ r_{i_p} \succ_{\mathsf{POLS}} s_{j_p} \ \wedge\ s_{j_q} \succ_{\mathsf{POLS}} t_{k_q}$

- $p \neq q$
  $\hookrightarrow$ $r_{i_p} \succ_{\mathsf{POLS}} s_{j_p} \succsim_{\mathsf{POLS}} t_{k_p}$ (if $p<q$) $\quad\vee$
  $r_{i_q} \succsim_{\mathsf{POLS}} s_{j_q} \succ_{\mathsf{POLS}} t_{k_q}$ (if $p>q$)
  $\hookrightarrow$ $r_{i_p} \succ_{\mathsf{POLS}} t_{k_p} \ \vee\ r_{i_q} \succ_{\mathsf{POLS}} t_{k_q}$
  $\because$ ii)
  $\hookrightarrow$ $r \succ_{\mathsf{POLS}} t$
  $\because$ All predecessors are equivalent w.r.t. $\succsim_{\mathsf{POLS}}$

- $p = q$
  $\hookrightarrow$ $r_{i_p} \succ_{\mathsf{POLS}} s_{j_p} \succ_{\mathsf{POLS}} t_{k_p}$
  $\hookrightarrow$ $r_{i_p} \succ_{\mathsf{POLS}} t_{k_p}$
  $\because$ Induction hypothesis
  $\hookrightarrow$ $r \succ_{\mathsf{POLS}} t$
  $\because$ Analogous to the case $p \neq q$

iv) The POLS is $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible, i.e. $f(\ldots,r,\ldots) \succ_{\mathsf{POLS}} f(\ldots,s,\ldots)$ if $r \succ_{\mathsf{POLS}} s$:

▶ $\llbracket r \rrbracket \sqsupset \llbracket s \rrbracket$
$\hookrightarrow$ $f(\ldots,r,\ldots) \succ_{\mathsf{POLS}} f(\ldots,s,\ldots)$
$\because$ $\sqsupset$ is $\mathcal{T}(\mathcal{F},\mathcal{X})$-compatible

▶ $\llbracket r \rrbracket \sqsupseteq \llbracket s \rrbracket$
$\hookrightarrow$ $\llbracket f(\ldots,r,\ldots) \rrbracket \sqsupseteq \llbracket f(\ldots,s,\ldots) \rrbracket$
$\because$ The term structures are identical, otherwise
$\hookrightarrow$ $f(\ldots,r,\ldots) \succ_{\mathsf{POLS}} f(\ldots,s,\ldots)$
$\because$ Definition of the POLS and $r \succ_{\mathsf{POLS}} s$

v) The POLS is σ-stable:
We must show that $\forall \sigma, s, t\colon\ s \succ_{\mathsf{POLS}} t \ \hookrightarrow\ s\sigma \succ_{\mathsf{POLS}} t\sigma$. We will prove this fact by induction on $|t|$:

▶ $\llbracket s \rrbracket \sqsupset \llbracket t \rrbracket$
$\hookrightarrow$ $s\sigma \succ_{\mathsf{POLS}} t\sigma$
$\because$ $\sqsupset$ is σ-stable

▶ $\llbracket s \rrbracket \sqsupseteq \llbracket t \rrbracket \ \wedge\ \mathcal{H}ead(s) \succ \mathcal{H}ead(t)$
$\hookrightarrow$ $\llbracket s\sigma \rrbracket \sqsupseteq \llbracket t\sigma \rrbracket$
$\because$ $\llbracket s \rrbracket = \llbracket t \rrbracket \hookrightarrow \llbracket s\sigma \rrbracket = \llbracket t\sigma \rrbracket$ and $\sqsupset$ is σ-stable
$\hookrightarrow$ $s\sigma \succ_{\mathsf{POLS}} t\sigma$
$\because$ $\forall t \in \mathcal{T}(\mathcal{F},\mathcal{X})\colon\ t \notin \mathcal{X} \ \hookrightarrow\ \mathcal{H}ead(t) = \mathcal{H}ead(t\sigma)$

▶ $\llbracket s \rrbracket \sqsupseteq \llbracket t \rrbracket \ \wedge\ \mathcal{H}ead(s) \sim \mathcal{H}ead(t)$
Let $s = f(s_1,\ldots,s_m)$, $t = g(t_1,\ldots,t_n)$ such that $f \sim g$.

α) $\tau(f) = \tau(g) = \textcircled{m}$

$\hookrightarrow$ $\{\!\{ s_1,\ldots,s_m \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ t_1,\ldots,t_n \}\!\}$
$\because$ Definition of the POLS

$\hookrightarrow$ $\forall j$: $\exists i$: $s_i \succsim_{\mathsf{POLS}} t_j$
$\because$ Definition of $\succ^{\textcircled{m}}_{\mathsf{POLS}}$

$\hookrightarrow$ $\forall j$: $\exists i$: $s_i\sigma \succsim_{\mathsf{POLS}} t_j\sigma$
$\because$ Induction hypothesis and the fact that $s \sim_{\mathsf{POLS}} t \hookrightarrow s\sigma \sim_{\mathsf{POLS}} t\sigma$ holds

$\hookrightarrow$ $\{\!\{ s_1\sigma,\ldots,s_m\sigma \}\!\} \succ^{\textcircled{m}}_{\mathsf{POLS}} \{\!\{ t_1\sigma,\ldots,t_n\sigma \}\!\}$
$\because$ Definition of the POLS

$\hookrightarrow$ $s\sigma \succ_{\mathsf{POLS}} t\sigma$

β) $\tau(f) = (i_1,\ldots,i_m)$, $\tau(g) = (j_1,\ldots,j_n)$

$\hookrightarrow$ $\exists k$: $\forall l < k$: $s_l \sim_{\mathsf{POLS}} t_l \;\wedge\; s_k \succ_{\mathsf{POLS}} t_k$

$\hookrightarrow$ $\exists k$: $\forall l < k$: $s_l\sigma \sim_{\mathsf{POLS}} t_l\sigma \;\wedge\; s_k\sigma \succ_{\mathsf{POLS}} t_k\sigma$
$\because$ Induction hypothesis and the fact that $s \sim_{\mathsf{POLS}} t \hookrightarrow s\sigma \sim_{\mathsf{POLS}} t\sigma$ holds

$\hookrightarrow$ $s\sigma \succ_{\mathsf{POLS}} t\sigma$
$\because$ Definition of the POLS

vi) The POLS is well-founded:
We have to prove that there does not exist an infinite chain $t_1 \succ_{\mathsf{POLS}} t_2 \succ_{\mathsf{POLS}} \cdots$. This is a direct implication of the facts that the orderings $\sqsupset$ and $\succ$ are well-founded.

◆

$\boxed{\textbf{B.26}}$ **Lemma (5.26)** *Let p be a polynomial with m (≥ 1) positive monomials and n negative monomials. For the worst case of the extended* BL-approach, *$\Pi(p)$ and $\Phi(p)$ are as follows:*

$$\Pi(p) = (m!)^3 \cdot n! \cdot m^{n-m-1}$$

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

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

Proof: The number $\Pi(p)$ of paths in the tree corresponding to p is equivalent to the number of leaves in the tree. Therefore, according to figure 5.5, given below once again ($k = n - m$, $n \geq m$)

$$\begin{array}{c}
1 \\
nm \\
nm(n-1)m \\
nm(n-1)m(n-2)m \\
\vdots \\
nm(n-1)m(n-2)m\cdot\ldots\cdot(n-k)m =: l \\
l\cdot(m-1)m \\
l\cdot(m-1)m(m-1)(m-1) \\
\vdots \\
l\cdot(m-1)m(m-1)(m-1)\cdot\ldots\cdot(m-i)(m-i) \\
l\cdot(m-1)m(m-1)(m-1)\cdot\ldots\cdot(m-i)(m-i)(m-i-1)(m-i) \\
l\cdot(m-1)m(m-1)(m-1)\cdot\ldots\cdot(m-i)(m-i)(m-i-1)(m-i)(m-i-1)(m-i-1) \\
\vdots \\
l\cdot(m-1)m(m-1)(m-1)\cdot\ldots\cdot(m-i)(m-i)(m-i-1)(m-i)(m-i-1)(m-i-1)\cdot\ldots\cdot 1\cdot 1
\end{array}$$

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

The non-italic expressions represent the number w.r.t. n[2]. Thus, there are two sequences (i.e. products) of numbers (one sequence w.r.t. n and one sequence w.r.t. m):

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

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

The number $\Phi(p)$ of nodes corresponds to the sum of all levels (except the first one) of figure 5.5. We split the sum (as well as the diagram) into two parts:

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

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

---

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

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

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

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

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

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

◆

$\boxed{\textbf{B.27}}$ **Lemma (5.28)** *Let* $\Phi(p)$ *and* $\Pi(p)$ *be defined as in lemma 5.26. Then*

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

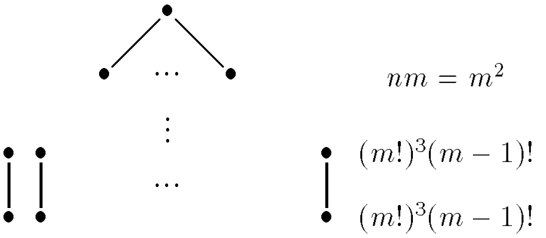
◇

Proof: The case $n = m = 1$ is obvious. Therefore, let $n = m + k$, $k \ge 0$ and not both $m = 1$ and $k = 0$. Furthermore, let

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

▶ We show that $2\Pi(p) \le \Phi(p)$:

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



The last two levels of the tree contain

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

nodes (see the expressions for $\Pi(p)$ and $\Phi(p)$ of lemma 5.26).

$\hookrightarrow \quad \Phi(p) \ge 2\Pi(p)$

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

  $$\hookrightarrow \quad \Pi(p) \;=\; n! \quad \textit{and}$$
  $$\Phi(p) \;=\; n! \cdot \sum_{j=0}^{n-1} \frac{1}{(n-j-1)!} \;=\; n! \cdot \left(\frac{1}{(n-1)!} + \frac{1}{(n-2)!} + \cdots + \frac{1}{1!} + \frac{1}{0!}\right)$$
  $$\hookrightarrow \quad \Phi(p) \;\geq\; 2n! \;=\; 2\Pi(p)$$

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

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

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

α) Let $m = 1$ and $n > 1$:

Then we have to prove the following inequality:

$$\sum_{j=0}^{n-1} \frac{1}{(n-j-1)!} \;\leq\; 3$$

This is valid since $\sum_{j=0}^{n-1} \frac{1}{(n-j-1)!} = \frac{1}{(n-1)!} + \frac{1}{(n-2)!} + \cdots + \frac{1}{1!} + \frac{1}{0!} = \frac{1}{(n-1)!} + \frac{1}{(n-2)!} + \cdots + \frac{1}{2!} + 2$ and $\frac{1}{(n-1)!} + \frac{1}{(n-2)!} + \cdots + \frac{1}{2!} \leq \frac{1}{2} + \frac{1}{6} + \frac{1}{18} + \cdots + \frac{1}{2 \cdot 3^{n-3}} =^{3)} \frac{3}{2}(\frac{1}{2} - \frac{1}{2 \cdot 3^{n-2}}) \leq 1$.

β) Let $m = 2$ and $n \geq 2$4):

Then we have to prove the following inequality:

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

This is valid since $\frac{1}{2} \cdot \sum_{j=1}^{2} \prod_{i=1}^{j} (2 - \lceil \frac{i}{2} \rceil)(2 - \lfloor \frac{i}{2} \rfloor) = 2$ and $\sum_{j=0}^{n-2} (\frac{2^j}{(n-j-1)!} \cdot \frac{1}{2^3 \cdot 2^{n-4}}) = \sum_{j=0}^{n-2} \frac{1}{(n-j-1)! \cdot 2^{n-j-1}} = \frac{1}{1! \cdot 2^1} + \frac{1}{2! \cdot 2^2} + \frac{1}{3! \cdot 2^3} + \cdots + \frac{1}{(n-1)! \cdot 2^{n-1}} \leq \frac{1}{2} + \frac{1}{2 \cdot 4^1} + \frac{1}{2 \cdot 4^2} + \cdots + \frac{1}{2 \cdot 4^{n-2}} = \frac{4}{3}(\frac{1}{2} - \frac{1}{2 \cdot 4^{n-1}}) \leq 1$.

γ) Let $n \geq m \geq 3$:

With the help of the fact that

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

we have to prove that

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

Note that the following two facts hold:

- $\sum\limits_{j=0}^{n-m} \frac{1}{(n-j-1)! \cdot ((m-1)!)^3 \cdot m^{n-m-j+1}} \leq (\frac{1}{2})^{n-m} \cdot a + (\frac{1}{2})^{n-m-1} \cdot a + \cdots + (\frac{1}{2})^1 \cdot a + a = 2a(1 - (\frac{1}{2})^{n-m+1})$ where $a = \frac{1}{((m-1)!)^4 \cdot m}$

  $\because \frac{1}{(n-(n-m)-1)! \cdot ((m-1)!)^3 \cdot m^{n-m-(n-m)+1}} = \frac{1}{((m-1)!)^4 \cdot m}$ and

  $\frac{1}{(n-j-1)! \cdot ((m-1)!)^3 \cdot m^{n-m-j+1}} \Big/ \frac{1}{(n-j-2)! \cdot ((m-1)!)^3 \cdot m^{n-m-j}} = \frac{1}{(n-j-1) \cdot m} \leq \frac{1}{2}$

---

3) by using the sum formula $\frac{x_1 - x_n q}{1-q}$ for geometric sequences where $x_1 = \frac{1}{2}$, $x_n = \frac{1}{2 \cdot 3^{n-3}}$ and $q = \frac{1}{3}$

4) Note that $n \geq m$ must hold due to the precondition of this lemma.

- $\sum_{j=1}^{2m-4} \prod_{i=1}^{j} (m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor) \leq (\frac{1}{2})^{2m-5} \cdot b + (\frac{1}{2})^{2m-6} \cdot b + \cdots + (\frac{1}{2})^{1} \cdot b + b =$
  $2b(1 - (\frac{1}{2})^{2m-4})$ where $b = \frac{((m-1)!)^3 \cdot m!}{2}$
  $\because \prod_{i=1}^{2m-4}(m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor) = b$ and
  $\prod_{i=1}^{j}(m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor) \leq \frac{1}{2} \cdot \prod_{i=1}^{j+1}(m - \lceil \frac{i}{2} \rceil)(m - \lfloor \frac{i}{2} \rfloor)$ if $j \leq 2m-5$

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

This expression must be less than or equal to 1 which is identical to the inequality

$$\frac{2}{((m-1)!)^4 \cdot m}(1 - (\tfrac{1}{2})^{n-m+1}) \ \leq \ (\tfrac{1}{2})^{2m-4}.$$

Since $1 - (\frac{1}{2})^{n-m+1} \ \leq \ 1$, it is sufficient to prove the following inequality:

$$2^{2m-3} \ \leq \ ((m-1)!)^4 \cdot m$$

The proof is done by induction on m:

- $m = 3$: $2^3 \ \leq \ 2^4 \cdot 3$
- $m \mapsto m+1$: $2^{2(m+1)-3} \ \leq \ (m!)^4 \cdot (m+1)$
  $\Longleftrightarrow \ 4 \cdot 2^{2m-3} \ \leq \ m^4 \cdot ((m-1)!)^4 \cdot m + (m!)^4$
  $\because$ Induction hypothesis and the fact that $4 \ \leq \ m^4$ holds $\blacklozenge$

$\boxed{\textbf{B.28}}$ **Theorem (5.32)** *Algorithm 5.29 always terminates. If it does not fail,* $p \sqsupset 0$ *holds.* $\diamond$

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

$\boxed{\textbf{B.29}}$ **Lemma (5.35)** *Let* α, β, $x_l > 0$:

$$\alpha x_1^{i_1} \cdot \ldots \cdot x_n^{i_n} + \beta x_1^{j_1} \cdot \ldots \cdot x_n^{j_n} \ \sqsupseteq \ 2\sqrt{\alpha\beta} \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \quad \textit{if} \quad \forall l \in [1, n]\text{: } k_l = \frac{i_l + j_l}{2} \qquad \diamond$$

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

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

$$a_1 + a_2 \ \sqsupseteq \ 2 \cdot a_1^{\frac{1}{2}} a_2^{\frac{1}{2}} \qquad \Longleftrightarrow \qquad \tfrac{1}{2}(a_1 + a_2) \ \sqsupseteq \ (a_1 a_2)^{\frac{1}{2}}$$

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

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

which is proved, for example, in [Hardy *et al.*, 1952] (see also remark 5.38 on page 115). ◆

$\boxed{\textbf{B.30}}$ **Lemma (5.37)** *Let* $n \geq m \geq 1$. *Then for all* $x_i > 0$:

$$\sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \sqsupseteq m \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \quad \textit{if} \quad \forall l \in [1,n]\text{: } k_l \leq \tfrac{1}{m} \cdot \sum_{j=1}^{m} i_{l_j} \qquad \diamond$$

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

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

Let $n \geq m$.

$$\hookrightarrow \quad \sum_{j=1}^{m} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \;\sqsupseteq\; m \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$$

$$\Longleftrightarrow$$

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

$$\Longleftrightarrow$$

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

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

$$\hookrightarrow \quad (**)$$

$$\Longleftrightarrow$$

$$\sum_{j=1}^{n} x_1^{i_{1_j}} \cdot \ldots \cdot x_n^{i_{n_j}} \;\sqsupseteq\; n \cdot x_1^{k_1} \cdot \ldots \cdot x_n^{k_n} \tag{$***$}$$

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

$$\hookrightarrow \quad (***)$$

$$\hookleftarrow$$

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

$$\because \forall l \in [1,n]\text{: } k_l \leq \tfrac{1}{n} \sum_{j=1}^{n} i_{l_j}$$

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

$$\Longleftrightarrow$$

$$\sum_{j=1}^{n} a_j \;\sqsupseteq\; n \cdot (\prod_{j=1}^{n} a_j)^{\frac{1}{n}}$$

$$\because \quad \forall l \in [1,n]\text{: } a_l = x_1^{i_{1_l}} \cdot \ldots \cdot x_n^{i_{n_l}}$$

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

$\boxed{\textbf{B.31}}$ **Lemma (5.47)** *The ordering* $\succ_{\mathsf{P}}$ *defined by* $s \succ_{\mathsf{P}} t$ *if, and only if,* $[\![s]\!] \sqsupset_P [\![t]\!]$ *is a reduction ordering on* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *if no constants in* $\mathcal{F}$ *exist.* $\diamond$

Proof: By the definition of $\succ_{\mathsf{M}}$, for each $m_i \succsim_{\mathsf{M}} m_j$ there exists a μ_{ij} such that $m_i \sqsupseteq m_j$ w.r.t. $\mathcal{M} = \{k \mid k \geq \mu_{ij}\}$. Let $\mu_i = \sum_{m_i \succsim_{\mathsf{M}} m_j} \mu_{ij}$. This implies that $m_i' \sqsupseteq \sum_{j=1}^{l} m_j$ holds if $\forall j \in [1,l]$: $m'_i \sqsupseteq m_j$. Let $\mu = 1 + \max_i \mu_i$. Then $\sum m'_i \sqsupset \sum m_j$ if $\Cup\{m'_i\} \overset{\textcircled{m}}{\succ}_{\mathsf{M}} \Cup\{m_j\}$. ◆

$\boxed{\textbf{B.32}}$ **Lemma (5.52)** *The* POL *is $\mathcal{E}$-compatible if*

$$\forall s,t \in \mathcal{T}(\mathcal{F},\mathcal{X})\colon\ s \overset{*}{\leftrightarrow}_{\mathcal{E}} t \quad \hookrightarrow \quad s \sim_{\mathsf{POL}} t. \qquad \diamond$$

Proof: We have to prove

$$s' \overset{*}{\leftrightarrow}_{\mathcal{E}} s \succ_{\mathsf{POL}} t \overset{*}{\leftrightarrow}_{\mathcal{E}} t' \quad \hookrightarrow \quad s' \succ_{\mathsf{POL}} t'.$$

Note that

$$s' \overset{*}{\leftrightarrow}_{\mathcal{E}} s \ \wedge\ t \overset{*}{\leftrightarrow}_{\mathcal{E}} t' \quad \hookrightarrow \quad s' \sim_{\mathsf{POL}} s \ \wedge\ t \sim_{\mathsf{POL}} t'$$

which implies

$$s' \sim_{\mathsf{POL}} s \succ_{\mathsf{POL}} t \sim_{\mathsf{POL}} t'.$$

Thus, the assertion holds since $t \sim_{\mathsf{POL}} t'$ implies $[\![t]\!] = [\![t']\!]$. ◆

$\boxed{\textbf{B.33}}$ **Lemma (Left Distributivity, 5.53)** $f(g(x,y),z) \sim_{\mathsf{POL}} g(f(x,z),f(y,z))$ *if one of the following conditions holds:*

▶ $[\![f]\!](x,y) = \sum\limits_i \alpha_i x y^i$ $\quad [\![g]\!](x,y) = \beta_1 x + \beta_2 y$

▶ $[\![f]\!](x,y) = \alpha_1 xy + \alpha_2(x+y) + \alpha_3$ $\quad [\![g]\!](x,y) = x + y + \beta_3$
$\wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1), \alpha_2 = \alpha_1\beta_3$ ◇

Proof:

i) Let $[\![f]\!](x,y) = \sum\limits_i \alpha_i x y^i$, $[\![g]\!](x,y) = \beta_1 x + \beta_2 y$

$$\begin{aligned}\hookrightarrow\quad [\![f(g(x,y),z)]\!] &= \sum_i \alpha_i(\beta_1 x + \beta_2 y) z^i \\ &= \sum_i \beta_1 \alpha_i x z^i + \sum_i \beta_2 \alpha_i y z^i \\ &= \beta_1 \sum_i \alpha_i x z^i + \beta_2 \sum_i \alpha_i y z^i = [\![g(f(x,z),f(y,z))]\!]\end{aligned}$$

ii) Let $[\![f]\!](x,y) = \alpha_1 xy + \alpha_1\beta_3(x+y) + \beta_3(\alpha_1\beta_3 - 1)$, $[\![g]\!](x,y) = x + y + \beta_3$

$$\begin{aligned}\hookrightarrow\quad [\![f(g(x,y),z)]\!] &= \alpha_1(x+y+\beta_3)z + \alpha_1\beta_3(x+y+\beta_3+z) + \beta_3(\alpha_1\beta_3 - 1) \\ &= \alpha_1 xz + \alpha_1 yz + \alpha_1\beta_3 x + \alpha_1\beta_3 y + 2\alpha_1\beta_3 z + 2\alpha_1\beta_3^2 - \beta_3 \\ &= (\alpha_1 xz + \alpha_1\beta_3(x+z) + \beta_3(\alpha_1\beta_3 - 1)) + (\alpha_1 yz + \alpha_1\beta_3(y+z) \\ &\quad + \beta_3(\alpha_1\beta_3 - 1)) + \beta_3 = [\![g(f(x,z),f(y,z))]\!]\end{aligned}$$

◆

$\boxed{\textbf{B.34}}$ **Lemma (Right Distributivity, 5.54)** $f(x,g(y,z)) \sim_{\mathsf{POL}} g(f(x,y),f(x,z))$ *if one of the following conditions holds:*

▶ $[\![f]\!](x,y) = \sum\limits_i \alpha_i x^i y$ $\quad [\![g]\!](x,y) = \beta_1 x + \beta_2 y$

▶ $[\![f]\!](x,y) = \alpha_1 xy + \alpha_2(x+y) + \alpha_3$ $\quad [\![g]\!](x,y) = x + y + \beta_3$
$\wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1), \alpha_2 = \alpha_1\beta_3$ ◇

Proof: analogous to the proof of lemma 5.53 ◆

$\boxed{\textbf{B.35}}$ **Lemma (Left and Right Distributivity, 5.55)** $f(g(x,y),z) \sim_{\mathsf{POL}} g(f(x,z),f(y,z)) \ \wedge \ f(x,g(y,z)) \sim_{\mathsf{POL}} g(f(x,y),f(x,z))$ *if one of the following conditions holds:*

- ▶ $[\![f]\!](x,y) = \alpha xy$ $\quad [\![g]\!](x,y) = \beta_1 x + \beta_2 y$
- ▶ $[\![f]\!](x,y) = \alpha_1 xy + \alpha_2(x+y) + \alpha_3$ $\quad [\![g]\!](x,y) = x + y + \beta_3$
  $\wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1), \alpha_2 = \alpha_1\beta_3$ ◇

Proof:

i) $[\![f]\!](x,y) = \alpha xy$, $[\![g]\!](x,y) = \beta_1 x + \beta_2 y$:

- $[\![f(g(x,y),z)]\!] \ = \ [\![g(f(x,z),f(y,z))]\!]$
  $\because$ $[\![f]\!](x,y)$ is a special case of $[\![f]\!](x,y)$ in lemma 5.53 and $[\![g]\!](x,y)$ is identical to that of lemma 5.53
- $[\![f(x,g(y,z))]\!] \ = \ [\![g(f(x,y),f(x,z))]\!]$
  $\because$ $[\![f]\!](x,y)$ is a special case of $[\![f]\!](x,y)$ in lemma 5.54 and $[\![g]\!](x,y)$ is identical to that of lemma 5.54

ii) $[\![f]\!](x,y) = \alpha_1 xy + \alpha_2(x+y) + \alpha_3$, $[\![g]\!](x,y) = x + y + \beta_3$
$\wedge\ \alpha_3 = \beta_3(\alpha_1\beta_3 - 1)$, $\alpha_2 = \alpha_1\beta_3$:
These interpretations are identical to that of lemma 5.53 and lemma 5.54. ⧫

$\boxed{\textbf{B.36}}$ **Lemma (5.56)** *There exists no* POL *which induces the equality of both sides of the following combinations of axioms:*

- ▶ $f(g(x,y),z) \leftrightarrow_{\mathcal{E}} g(f(x,z),f(y,z)) \ \wedge \ g(x,f(y,z)) \leftrightarrow_{\mathcal{E}} f(g(x,y),g(x,z))$
- ▶ $f(g(x,y),z) \leftrightarrow_{\mathcal{E}} g(f(x,z),f(y,z)) \ \wedge \ g(f(x,y),z) \leftrightarrow_{\mathcal{E}} f(g(x,z),g(y,z))$
- ▶ $f(x,g(y,z)) \leftrightarrow_{\mathcal{E}} g(f(x,y),f(x,z)) \ \wedge \ g(x,f(y,z)) \leftrightarrow_{\mathcal{E}} f(g(x,y),g(x,z))$ ◇

Proof:

i) The equalities

$$[\![f(g(x,y),z)]\!] = [\![g(f(x,z),f(y,z))]\!] \quad \text{and} \quad [\![g(x,f(y,z))]\!] = [\![f(g(x,y),g(x,z))]\!]$$

imply the equality

$$[\![f(g(x,y),z)]\!] \ = \ [\![f(g(f(x,z),y),g(f(x,z),z))]\!]$$

gained by unifying $g(f(x,z),f(y,z))$ and $g(x,f(y,z))$. Thus,

$$[\![f(g(x,x),x)]\!] \ = \ [\![f(g(f(x,x),x),g(f(x,x),x))]\!]$$

must hold. By a relatively simple analysis one can prove that there are no polynomials interpreting f and g such that the above equality is valid.

ii) In analogy with i), the equality

$$[\![f(g(x,x),x)]\!] \;=\; [\![f(g(x,f(x,x)),g(x,f(x,x)))]\!]$$

has to be fulfilled which is impossible with the help of the POL.

iii) analogous to ii) ◆

$\boxed{\textbf{B.37}}$ **Lemma (Endomorphism, 5.57)** $h(f(x,y)) \sim_{\mathsf{POL}} f(h(x),h(y))$ *if one of the following conditions holds:*

- ▶ $[\![f]\!](x,y) = \sum_{i,j} \alpha_{ij} x^i y^j$ $\quad [\![h]\!](x) = x$
- ▶ $[\![f]\!](x,y) = \alpha_1 x + \alpha_2 y + \alpha_3$ $\quad [\![h]\!](x) = \beta_1 x + \beta_2$
  $\wedge\ \alpha_3(\beta_1 - 1) = \beta_2(\alpha_1 + \alpha_2 - 1)$
- ▶ $[\![f]\!](x,y) = \alpha x^i y^j$ $\quad [\![h]\!](x) = \beta x^k$
  $\wedge\ \beta^{i+j-1} = \alpha^{k-1}$ ◇

Proof:

i) Let $[\![f]\!](x,y) = \sum_{i,j} \alpha_{ij} x^i y^j$ and $[\![h]\!](x) = x$.

$$\hookrightarrow \quad [\![h(f(x,y))]\!] = \sum_{i,j} \alpha_{ij} x^i y^j = [\![f(h(x),h(y))]\!]$$

ii) Let $[\![f]\!](x,y) = \alpha_1 x + \alpha_2 y + \alpha_3$ and $[\![h]\!](x) = \beta_1 x + \beta_2$ such that $\alpha_3(\beta_1 - 1) = \beta_2(\alpha_1 + \alpha_2 - 1)$.

$$\begin{aligned} \hookrightarrow \quad [\![h(f(x,y))]\!] &= \alpha_1\beta_1 x + \alpha_2\beta_1 y + \alpha_3\beta_1 + \beta_2 \\ [\![f(h(x),h(y))]\!] &= \alpha_1\beta_1 x + \alpha_2\beta_1 y + \alpha_1\beta_2 + \alpha_2\beta_2 + \alpha_3 \end{aligned}$$

$\hookrightarrow \quad \alpha_3\beta_1 + \beta_2 = \alpha_1\beta_2 + \alpha_2\beta_2 + \alpha_3$ must be true which is equivalent to the precondition

iii) Let $[\![f]\!](x,y) = \alpha x^i y^j$ and $[\![h]\!](x) = \beta x^k$ such that $\beta^{i+j-1} = \alpha^{k-1}$.

$$\begin{aligned} \hookrightarrow \quad [\![h(f(x,y))]\!] &= \alpha^k \beta x^{ik} y^{jk} \\ [\![f(h(x),h(y))]\!] &= \alpha\beta^{i+j} x^{ik} y^{jk} \end{aligned}$$

$\hookrightarrow \quad \alpha^k\beta = \alpha\beta^{i+j}$ must be true which is equivalent to the precondition ◆

$\boxed{\textbf{B.38}}$ **Lemma (5.58)** $h(f(x,y)) \sim_{\mathsf{POL}} f(h(y),h(x))$ *if one of the following conditions holds:*

- ▶ $[\![f]\!](x,y) = \sum_{i,j} \alpha_{ij} x^i y^j$ $\quad [\![h]\!](x) = x$
  $\wedge\ \forall i,j\colon \alpha_{ij} = \alpha_{ji}$
- ▶ $[\![f]\!](x,y) = \alpha_1(x+y) + \alpha_3$ $\quad [\![h]\!](x) = \beta_1 x + \beta_2$
  $\wedge\ \alpha_3(\beta_1 - 1) = \beta_2(2\alpha_1 - 1)$
- ▶ $[\![f]\!](x,y) = \alpha x^i y^i$ $\quad [\![h]\!](x) = \beta x^j$
  $\wedge\ \beta^{2i-1} = \alpha^{j-1}$ ◇

Proof: analogous to the proof of lemma 5.57 ◆

$\boxed{\textbf{B.39}}$ **Lemma (5.60)** *There are no interpretations of operators such that* $s \sim_{\mathsf{POL}} t$ *if* $s \leftrightarrow_{\mathcal{E}} t$ *is subterm-minimizing.* ◇

Proof: W.l.o.g. let $t = f(t_1,\ldots,t_n)$, $n > 1$, $u \in \{1,\ldots,n\}$ and $v \in \{1,\ldots,n\} \setminus \{u\}$. Assume that there exists an interpretation for f such that $[\![t]\!] = [\![t_u]\!]$.

$\hookrightarrow$ The coefficients of the monomials of $[\![f]\!](x_1,\ldots,x_n)$ containing x_v must be set to zero.
  $\because$ $[\![t]\!] \neq [\![t_u]\!]$, otherwise
$\hookrightarrow$ $\mathcal{A}r([\![f]\!](x_1,\ldots,x_n)) < n$
$\hookrightarrow$ $[\![f]\!](x_1,\ldots,x_n)$ is not a correct interpretation.
  $\because$ Definition of an interpretation ◆

$\boxed{\textbf{B.40}}$ **Lemma (5.61)** *$\mathit{if}(x, \mathit{if}(y, z, u), \mathit{if}(y, v, w)) \nsim_{\mathsf{POL}} \mathit{if}(y, \mathit{if}(x, z, v), \mathit{if}(x, u, w))$ if $[\![\mathit{if}]\!](x, y, z) = \sum\limits_i(\alpha_i x^i y + \beta_i x^i z)$* ◇

Proof: Let $[\![\mathit{if}]\!](x, y, z) = \sum\limits_i(\alpha_i x^i y + \beta_i x^i z)$.

$$
\begin{aligned}
\hookrightarrow\ & [\![\mathit{if}(x, \mathit{if}(y, z, u), \mathit{if}(y, v, w))]\!] \\
&= \sum_i \alpha_i x^i \cdot (\sum_i \alpha_i y^i z + \sum_i \beta_i y^i u) + \sum_i \beta_i x^i \cdot (\sum_i \alpha_i y^i v + \sum_i \beta_i y^i w) \\
&= z \sum_i(\alpha_i x^i \sum_i \alpha_i y^i) + u \sum_i(\alpha_i x^i \sum_i \beta_i y^i) + v \sum_i(\beta_i x^i \sum_i \alpha_i y^i) + w \sum_i(\beta_i x^i \sum_i \beta_i y^i) \\
&= \sum_i \alpha_i y^i \cdot (\sum_i \alpha_i x^i z + \sum_i \beta_i x^i v) + \sum_i \beta_i y^i \cdot (\sum_i \alpha_i x^i u + \sum_i \beta_i x^i w) \\
&= [\![\mathit{if}(y, \mathit{if}(x, z, v), \mathit{if}(x, u, w))]\!] \\
&\because \sum_i(\alpha_i x^i \cdot \sum_i \beta_i y^i) \; = \; \sum_i(\alpha_i y^i \cdot \sum_i \beta_i x^i)
\end{aligned}
$$

◆

$\boxed{\textbf{B.41}}$ **Lemma (5.63)** *It will be impossible to use any part of the Boolean ring as an underlying theory (and the remaining part as reduction rules), if the termination is to be proved by a* POL. ◇

Proof: For simplicity, we use the following notations:

- ▶ $D_L(f, g)$ stands for the equality $f(g(x, y), z) = g(f(x, z), f(y, z))$
- ▶ $D_R(f, g)$ stands for the equality $f(x, g(y, z)) = g(f(x, y), f(x, z))$

According to definition 5.62 on page 123, the Boolean ring includes

$$D_L(+, \star),\ D_R(+, \star),\ D_L(\star, +) \text{ and } D_R(\star, +)$$

as axioms or rules.

Due to lemma 5.56 the following combinations are not allowed as underlying theories:

$$
\begin{array}{lcl}
D_L(+, \star) & \wedge & D_R(\star, +) \\
D_L(\star, +) & \wedge & D_R(+, \star) \\
D_L(+, \star) & \wedge & D_L(\star, +) \\
D_R(+, \star) & \wedge & D_R(\star, +)
\end{array}
$$

Therefore, either $D_L(+, \star) \wedge D_R(+, \star)$ or $D_L(\star, +) \wedge D_R(\star, +)$ must be part of the reduction rule set. However, both requirements are impossible:

i)

$$\begin{array}{rcl} (x \star y) + z & \succ_{\mathsf{POL}} & (x + z) \star (y + z) \\ x + (y \star z) & \succ_{\mathsf{POL}} & (x + y) \star (x + z) \\ [\![(x + y) \star z]\!] & = & [\![(x \star z) + (y \star z)]\!] \\ [\![x \star (y + z)]\!] & = & [\![(x \star y) + (x \star z)]\!] \end{array}$$

In order to prove the termination of the two rules with the POL, we have to require $[\![+]\!](x, y)$ to be a mixed interpretation and $[\![\star]\!](x, y)$ to be a separate one (due to lemma 5.67). However, the validity of the equalities cannot be ensured since there exists a monomial on the right-hand side of the first equality containing the variables x and y (y and z in the last equality, respectively) which cannot be covered by the left-hand side because $[\![+]\!](x, y)$ is strictly mixed.

ii)

$$\begin{array}{rcl} [\![(x \star y) + z]\!] & = & [\![(x + z) \star (y + z)]\!] \\ [\![x + (y \star z)]\!] & = & [\![(x + y) \star (x + z)]\!] \\ (x + y) \star z & \succ_{\mathsf{POL}} & (x \star z) + (y \star z) \\ x \star (y + z) & \succ_{\mathsf{POL}} & (x \star y) + (x \star z) \end{array}$$

The considerations of i) also apply, mutatis mutandis, to this case.

iii)

$$\begin{array}{rcl} (x \star y) + z & \succ_{\mathsf{POL}} & (x + z) \star (y + z) \\ x + (y \star z) & \succ_{\mathsf{POL}} & (x + y) \star (x + z) \\ (x + y) \star z & \succ_{\mathsf{POL}} & (x \star z) + (y \star z) \\ x \star (y + z) & \succ_{\mathsf{POL}} & (x \star y) + (x \star z) \end{array}$$

The first two rules require a strictly mixed interpretation for $+$ and a separate one for $\star$ whereas the last two rules require a strictly mixed interpretation for $\star$ and a separate one for $+$ (due to lemma 5.67). This is a contradiction. ♦

B.42 **Lemma (5.66)** *Let $l \to r$ be a rule. No linear interpretation for the unary operator i exists if one of the following conditions holds:*

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

Proof:

i) Let $[\![.]\!]$