Notes on Transformation Orderings

Joachim STEINBACH

Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
6750 Kaiserslautern
(Germany)
steinba@informatik.uni-kl.de

Abstract. An important property and also a crucial point of a term rewriting system is its termination. Transformation orderings, developed by Bellegarde & Lescanne strongly based on a work of Bachmair & Dershowitz, represent a general technique for extending orderings. The main characteristics of this method are two rewriting relations, one for transforming terms and the other for ensuring the well-foundedness of the ordering. The central problem of this approach concerns the choice of the two relations such that the termination of a given term rewriting system can be proved. In this communication, we present a heuristic-based algorithm that partially solves this problem. Furthermore, we show how to simulate well-known orderings on strings by transformation orderings.

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

1 Introduction and Notations

Term rewriting systems (TRS, for short) are based on directed equations (called *rules*) which may be used as a non-deterministic functional program. These rules are applied by repeatedly replacing subterms of a given formula with equal terms until the simplest possible form is obtained. *Confluence* (i.e. the result of computing a term by rewriting should be unique and therefore independent of the choice of a rule applied to the term at any time) and *termination* (i.e. all computations are finite) are the key properties for a TRS yielding unique normal forms.

Termination is an undecidable property ([Huet and Lankford, 1978], [Dauchet, 1988]). However, there exist various techniques which together are able to cope with many TRS occurring in practice. These *term orderings* are stable ($s \succ t \hookrightarrow s\sigma \succ t\sigma$ for any substitution σ) reduction orderings. *Reduction orderings* are well-founded (there are no infinite descending chains) and compatible with the structure of terms ($s \succ t \hookrightarrow f(\cdots s \cdots) \succ f(\cdots t \cdots)$ for any operator f). *Simplification orderings* are reduction orderings which have the subterm property ($f(\cdots t \cdots) \succ t$). This property does not allow to prove the termination of, for example, a rule of the form $f(f(x)) \rightarrow f(g(f(x)))$ since the left-hand side is homeomorphically embedded in the right-hand side. On the other hand, transformation orderings are able to prove the termination of such a rule. *Transformation orderings*[1] have been developed by Bellegarde & Lescanne ([Bellegarde and Lescanne, 1987], [Bellegarde and Lescanne, 1988], [Bellegarde and Lescanne, 1990]) from a work of Bachmair & Dershowitz ([Bachmair and Dershowitz, 1986], [Bachmair, 1987]). This method is based on the notion of a so-called *termination function* φ mapping terms into a well-founded set $(W, \succ_{\mathsf{W}})$ such that $s \succ t$ iff $\varphi(s) \succ_{\mathsf{W}} \varphi(t)$. For example, the Knuth-Bendix ordering KBO (defined in [Knuth and Bendix, 1970]) and the polynomial ordering ([Manna and Ness, 1970], [Lankford, 1975]) use this feature. The main characteristics of transformation orderings are as follows: We choose $(W, \succ_{\mathsf{W}})$ and φ such that

- W is the set of terms,
- φ is the normalization by a TRS $\mathcal{T}$ ($\mathcal{T}$ stands for transformation), i.e. $\varphi(t)$ is the $\mathcal{T}$-normal form of t,
- $\succ_{\mathsf{W}}$ is induced by a terminating TRS $\mathcal{S}$ and then,
- a TRS $\mathcal{R}$ is terminating if $\varphi(l) \succ_{\mathsf{W}} \varphi(r)$ for all rules $l \rightarrow r$ of $\mathcal{R}$ and both $\mathcal{T}$ and $\mathcal{S}$ satisfy some properties ($\mathcal{T} \cup \mathcal{S}$ is terminating, $\mathcal{T}$ is confluent and $\mathcal{S}$ cooperates with[2] $\mathcal{T}$).

The central problem with transformation orderings is to find adequate TRS $\mathcal{S}$ and $\mathcal{T}$ such that the termination of a given TRS can be proved. In this paper, we introduce a heuristic-driven algorithm which automatically generates $\mathcal{S}$ and $\mathcal{T}$ for many TRS (see section 3). Furthermore, we deal with some simulations of well-known orderings on strings by transformation orderings (see section 4), i.e. for a specific ordering $\succ$ we give $\mathcal{S}$ and $\mathcal{T}$ such that $s \succ t$ iff s is greater than t w.r.t. the induced transformation ordering.

[1] recently called *completion orderings* (see [Bellegarde and Lescanne, 1990])

[2] This is a kind of confluence for two different relations (see section 2).

For a formal description of transformation orderings (given in the following section) we need some indispensable notations. Nevertheless, we assume familiarity with the definitions used in term rewriting and especially in the area of termination of TRS. For good surveys on the theory of term rewriting and termination we refer to [Dershowitz and Jouannaud, 1990], [Avenhaus and Madlener, 1990], [Huet and Oppen, 1980] and [Dershowitz, 1987].

Let $\mathcal{F}$ be a set of *operators* and $\mathcal{X}$ be a set of *variables*. $\mathcal{T}(\mathcal{F},\mathcal{X})$ denotes the set of (first-order) *terms* over $\mathcal{F}$ and $\mathcal{X}$. An *atom* is either a constant (an operator with no arguments) or a variable. The *size* $|t|$ of a term t is the number of symbols (operators and variables) occurring in t[3]. The *set of variables* (*multiset of operators*) of a term t is denoted by $\mathcal{V}ar(t)$ ($\mathcal{F}un(t)$, respectively). A term t is *linear* if each variable of $\mathcal{V}ar(t)$ occurs only once. $\mathcal{P}os(t)$ represents the set of *positions* of a term t. If $u \in \mathcal{P}os(t)$, then $t|_u$ is the *subterm* of t at position u. We write $t[s]_u$ to denote the term that results from t by replacing $t|_u$ by s at position u.

A *rule* over a set of terms $\mathcal{T}(\mathcal{F},\mathcal{X})$ is an ordered pair s,t of terms, written as $s \to t$ such that $\mathcal{V}ar(s) \supseteq \mathcal{V}ar(t)$. Often, a rule will be named (numbered), e.g. by r, and we write $r : s \to t$. A set $\mathcal{R}$ of rules over $\mathcal{T}(\mathcal{F},\mathcal{X})$ is called a *TRS*. A TRS $\mathcal{R}$ is *left-linear* (*variable-preserving*) if the left-hand side of each rule of $\mathcal{R}$ is linear (if $\mathcal{V}ar(l) = \mathcal{V}ar(r)$ for each rule $l \to r \in \mathcal{R}$, respectively). 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 $s \to_{\mathcal{R}} t$, if $s|_u = l\sigma$ and $t = s[r\sigma]_u$ for some $l \to r \in \mathcal{R}, u \in \mathcal{P}os(s)$ and substitution σ. If there is no ambiguity, we use $\to$ instead of $\to_{\mathcal{R}}$. The relation $\leftarrow$ denotes the *inverse* of $\to$; $\xrightarrow{*}$ ($\xrightarrow{+}$) denotes the reflexive and transitive *closure* (the transitive closure, respectively) of $\to$. We write $\to_{\mathcal{R}1} \cdot \to_{\mathcal{R}2}$ for the *composition* of $\to_{\mathcal{R}2}$ and $\to_{\mathcal{R}1}$. Furthermore, $\to_{\mathcal{R}1} \subseteq \to_{\mathcal{R}2}$ describes the fact that $\to_{\mathcal{R}2}$ is an *extension* of $\to_{\mathcal{R}1}$, i.e. $\{(x,y) \mid x \to_{\mathcal{R}1} y\} \subseteq \{(x,y) \mid x \to_{\mathcal{R}2} y\}$. The relation $\to$ is *confluent* if for all terms s,t_1,t_2 such that $s \xrightarrow{*} t_1$ and $s \xrightarrow{*} t_2$, there exists a term s' such that $t_1 \xrightarrow{*} s'$ and $t_2 \xrightarrow{*} s'$, i.e. $\xleftarrow{*} \cdot \xrightarrow{*} \subseteq \xrightarrow{*} \cdot \xleftarrow{*}$. Local confluence is defined by $\leftarrow \cdot \to \subseteq \xrightarrow{*} \cdot \xleftarrow{*}$. If $\to_{\mathcal{R}}$ is *well-founded* (there is no infinite sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$ of terms) and confluent, the *normal form* of a term t, written as $t\downarrow_{\mathcal{R}}$, exists and is unique.

The *subsumption ordering* $\gtrdot$ is defined on terms s,t as usual: $s \gtrdot t$ iff $s \underline{\geq} t$ and $t \not\underline{\geq} s$ where $s \underline{\geq} t$ iff $\exists\sigma$: $s = t\sigma$. Thus, a term is smaller w.r.t. $\gtrdot$ than all of its proper instances.

2 The Transformation Orderings

This section deals with the formal definition of transformation orderings as described in [Bellegarde and Lescanne, 1988]. We give no motivating ideas concerning the concept of transformation orderings. We refer to [Bellegarde and Lescanne, 1988] and [Bellegarde and Lescanne, 1990] for details about the logical background of the approach.

Definition 2.1 (Cooperation)

- $\to_S$ locally cooperates *with* $\to_T$ *iff* ${}_T\!\leftarrow \cdot \to_S \subseteq \xrightarrow{*}_T \cdot \to_S \cdot \xrightarrow{*}_{S \cup T} \cdot {}_T\!\xleftarrow{*}$
- $\to_S$ cooperates *with* $\to_T$ *iff* ${}_T\!\xleftarrow{*} \cdot \to_S \cdot \xrightarrow{*}_{S \cup T} \subseteq \xrightarrow{*}_T \cdot \to_S \cdot \xrightarrow{*}_{S \cup T} \cdot {}_T\!\xleftarrow{*}$ ◇

A critical pair $\langle p,q \rangle$ between a rule of $\mathcal{S}$ and a rule of $\mathcal{T}$ such that $p\ {}_T\!\leftarrow \cdot \to_S q$ is said to be cooperative iff $p \xrightarrow{*}_T \cdot \to_S \cdot \xrightarrow{*}_{S \cup T} \cdot {}_T\!\xleftarrow{*} q$. The test whether a relation cooperates with

[3] $|M|$ also denotes the cardinality of a (multi-) set M.

another one can be reduced to the test of the local cooperation as stated in the following lemma.

Lemma 2.1 ([Bellegarde and Lescanne, 1988]) *Let $\rightarrow_{\mathcal{S}\cup\mathcal{T}}$ be well-founded and $\rightarrow_{\mathcal{T}}$ be locally confluent. Then, $\rightarrow_{\mathcal{S}}$ cooperates with $\rightarrow_{\mathcal{T}}$ if $\rightarrow_{\mathcal{S}}$ locally cooperates with $\rightarrow_{\mathcal{T}}$.* ◇

The cooperation of $\mathcal{S}$ and $\mathcal{T}$ is the basic property for transformation orderings to be term orderings.

Definition 2.2 (Transformation Ordering, [Bellegarde and Lescanne, 1988]) *Let $\rightarrow_{\mathcal{S}}$ and $\rightarrow_{\mathcal{T}}$ be two rewriting relations such that $\rightarrow_{\mathcal{S}}$ cooperates with $\rightarrow_{\mathcal{T}}$, $\rightarrow_{\mathcal{S}\cup\mathcal{T}}$ is included in some term ordering $\succ_{\mathsf{T}}$ and $\rightarrow_{\mathcal{T}}$ is confluent. Then, the* transformation ordering TO *on terms s and t is defined as*

$$s \succ_{\mathsf{TO}} t \iff \blacktriangleright\ s\downarrow_{\mathcal{T}} = t\downarrow_{\mathcal{T}} \ \wedge\ s \succ_{\mathsf{T}} t \quad \textit{or} \quad \blacktriangleright\ s\downarrow_{\mathcal{T}} \neq t\downarrow_{\mathcal{T}} \ \wedge\ s\downarrow_{\mathcal{T}} \rightarrow_{\mathcal{S}} \cdot \xrightarrow{*}_{\mathcal{S}\cup\mathcal{T}} t\downarrow_{\mathcal{T}}$$ ◇

Theorem 2.1 ([Bellegarde and Lescanne, 1988]) *The TO is a term ordering.* ◇

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 iff $\mathcal{R}$ terminates. Cooperation of $\mathcal{S}$ and $\mathcal{T}$ is crucial for a transformation ordering to be a term ordering ([Bellegarde and Lescanne, 1988]). In order to use TO as a term ordering, there are three properties to check: local cooperation (see lemma 2.1), confluence and termination. The last two properties can be proved by classical methods. The following theorem provides a means to test local cooperation for a restricted class of transformation systems[4] ($\mathcal{T}$).

Theorem 2.2 ([Bellegarde and Lescanne, 1988]) *Assume $\mathcal{T}$ is a left-linear and variable-preserving TRS. Then, a TRS $\mathcal{S}$ locally cooperates with $\mathcal{T}$ iff all critical pairs between $\mathcal{S}$ and $\mathcal{T}$ are cooperative.* ◇

Example 2.1 ([Dershowitz, 1987], [Bellegarde and Lescanne, 1988]) *Let $\mathcal{R}$ be the TRS composed of the rules*

$$\begin{array}{lcl} f(a) & \rightarrow & f(b) \\ g(b) & \rightarrow & g(a) \end{array}$$

Note that there exists no conventional ordering which is able to orient $\mathcal{R}$ (since either a must be greater than b or vice versa and both alternatives do not work). However, let

$$\mathcal{T}\colon\ f(b) \rightarrow g(b) \qquad\qquad \mathcal{S}\colon\ \begin{array}{lcl} f(a) & \rightarrow & g(b) \\ g(b) & \rightarrow & g(a) \end{array}$$

[4] Geser proves a more general result in [Geser, 1991].

The proof of the termination of $\mathcal{S} \cup \mathcal{T}$ can be performed with a recursive path ordering RPOS *with status (see [Dershowitz, 1987] and [Kamin and Lévy, 1980] for a formal definition of the* RPOS*) based on the precedence $f \succ g \succ b \succ a$. Furthermore, $\mathcal{T}$ is obviously locally confluent and $\mathcal{S}$ locally cooperates with $\mathcal{T}$ because there are no critical pairs. Thus, $\mathcal{S}$ cooperates with $\mathcal{T}$ (due to theorem 2.2). Now, we prove the termination of the rules of $\mathcal{R}$ by the induced* TO*:*

- $f(a)\downarrow_{\mathcal{T}} = f(a) \neq g(b) = f(b)\downarrow_{\mathcal{T}} \wedge f(a) \rightarrow_{\mathcal{S}} g(b)$
- $g(b)\downarrow_{\mathcal{T}} = g(b) \neq g(a) = g(a)\downarrow_{\mathcal{T}} \wedge g(b) \rightarrow_{\mathcal{S}} g(a)$ $\diamond$

We have implemented the TO ([Kandzia, 1990]) and conducted a series of examples. Some of them are listed in appendix A.

3 Generating Transformation Orderings

In this section, we are going to present a new technique for *automatically generating* a transformation ordering (i.e. for automatically generating $\mathcal{S}$ and $\mathcal{T}$) such that the termination of a TRS is ensured. In [Bellegarde and Lescanne, 1990], a completion-like algorithm is suggested for establishing cooperation of $\mathcal{S}$ and $\mathcal{T}$ if this property does not hold. Cooperation is achieved by adding new rules to $\mathcal{S}$. According to this algorithm, the transformation system $\mathcal{T}$ must be given initially, confluent (and terminating) and cannot be changed any more. In contrast to this approach, we are interested in generating the *whole* TO, i.e. $\mathcal{S}$ *and* $\mathcal{T}$. Before giving precise definitions to formally describe our method, let us elucidate the technique more or less informally. Suppose we have to prove the termination of a TRS $\mathcal{R}$ with the help of a TO. The parameters $\mathcal{S}$ and $\mathcal{T}$ of an adequate TO are to be obtained from $\mathcal{R}$. The following abstract inference system provides the general frame for our technique.

Orientation:[5] $\dfrac{(\mathcal{S},\{l \rightarrow r\} \cup NO,\mathcal{T},\succ)}{(\{l \rightarrow r\} \cup \mathcal{S},NO,\mathcal{T},\succ)}$ if $l \succ r$ and $\{l \rightarrow r\} \cup \mathcal{S}$ cooperates with $\mathcal{T}$

Simplification: $\dfrac{(\mathcal{S} \cup \mathcal{S}_1,NO \cup \{l' \rightarrow r'\},\mathcal{T},\succ)}{(\mathcal{S},NO \cup \{l'' \rightarrow r''\},\mathcal{T} \cup \mathcal{S}_1,\succ)}$ if $\mathcal{S}_1 = \bigcup\{l_i \rightarrow r_i\}$, $l' \xrightarrow{*}_{\mathcal{S}_1} l''$, $r' \xrightarrow{*}_{\mathcal{S}_1} r''$ and each $l_k \rightarrow r_k \in \mathcal{S}_1$ has been used to compute l'' or r''

Extension: $\dfrac{(\mathcal{S},\{l \rightarrow r\} \cup NO,\mathcal{T},\succ)}{(\mathcal{S}',NO,\mathcal{T}',\succ')}$ if[6] $(\succ',\mathcal{T}') = \mathrm{ext}(\succ,\mathcal{T})$ and $\mathcal{S}'$ (which is a modified $\mathcal{S}$) cooperates with $\mathcal{T}'$, $\mathcal{S}' \cup \mathcal{T}'$ is well-founded and $\mathcal{T}'$ is locally confluent

A derivation begins with $(\emptyset,\mathcal{R},\emptyset,\succ_{\mathsf{T}})$, the initial state. The final state must be of the form $(\mathcal{S},\emptyset,\mathcal{T},\succ)$ which represents the transformation ordering given by $\mathcal{S}$, $\mathcal{T}$ and the basic ordering $\succ$. Applications of the inference rules 'orientation', 'simplification' and 'extension' should be supervised as follows:

- *Put as many rules in $\mathcal{S}$ as possible and keep $\mathcal{T}$ small.* The reason for this strategy is twofold. (i) The restrictions attached to $\mathcal{T}$ are very strong (see theorem 2.2).

[5] '*NO*' stands for 'not orientable'.

[6] The function 'ext' suitably extends the ordering $\succ$ and the transformation system $\mathcal{T}$ (see algorithm 3.1).

(ii) Transformation orderings are strictly more powerful than any other termination proof method if we take $\mathcal{T}$ to be empty and $\mathcal{S}$ to be equal to the system of which the termination has to be shown.

- *The (right-hand sides of the) rules of $\mathcal{T}$ will be constructed with new operators (not occurring in $\mathcal{F}$).* The introduction of (new) operators is a very flexible and simple technique to strengthen the basic ordering of each TO, since these operators initially have no connections (w.r.t. the precedence or any other parameter of the basic ordering) with operators of $\mathcal{F}$.

Let us present our approach in more detail. The input parameters as provided by the initial state $(\emptyset,\mathcal{R},\emptyset,\succ_T)$ are a rule system $\mathcal{R}$ (to be proved to be terminating), a term ordering $\succ_T$ (which will be used as the basic ordering of the TO to be generated) and empty rule systems $\mathcal{S}$ and $\mathcal{T}$. The ordering $\succ_T$ should at least be *improvable in an incremental way* whenever $\mathcal{F}$ is enriched by new operators. For instance, the recursive path ordering RPOS with status possesses this property (it can be started with an empty precedence, i.e. with the homeomorphic embedding). Our method is divided into four steps:

1. Orienting with the basic term ordering: Each rule of $\mathcal{R}$ which can be oriented w.r.t. $\succ_T$ will be part of $\mathcal{S}$ and removed from $\mathcal{R}$ (see inference rule 'orientation').

2. Semantic transformation (normalization): The remaining rules of $\mathcal{R}$ (i.e. each side of them) will be transformed into 'easier' ones by applying the rules of the current $\mathcal{S}$. If these 'easier' rules can be oriented w.r.t. the term ordering $\succ_T$, they will be removed from $\mathcal{R}$ and added to $\mathcal{S}$. Furthermore, each rule of $\mathcal{S}$ which has been part of the reduction process will be moved from $\mathcal{S}$ to $\mathcal{T}$ (see inference rule 'simplification').

3. Generating projection rules: This process will be applied if $\mathcal{R}$ is not empty, i.e. there exists at least one rule which cannot be oriented w.r.t. $\succ_T$. Let $l \to r$ be such a rule. Then, we extract those subterms of r which are 'critical', i.e. for which there is no greater (w.r.t. $\succ_T$) subterm in l. For each such subterm r' of r, we introduce a new rule $r' \to r''$ and include it in $\mathcal{T}$. The term r'' is of the form $f(x_1,\ldots,x_n)$ where f is a *new operator* and $\{x_1,\ldots,x_n\}$ represents the set of all variables occurring in r'. This way, we *project* critical terms of the right-hand side into smaller (w.r.t. $\succ_T$) ones (see inference rule 'extension'). The extended $\mathcal{T}$ will be applied to each side of the rules of $\mathcal{S}$ in order to produce $\mathcal{T}$-normal forms.

4. Checking the required properties of $\mathcal{S}$ and $\mathcal{T}$: This step tests whether $\mathcal{T}$ is variable-preserving and left-linear, $\mathcal{T}$ is confluent and $\mathcal{S}$ cooperates with $\mathcal{T}$[7].

Definition 3.1 (Trivial Term, Set of Non-Trivial Subterms) *Let $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$. Then,*

- *t is a* trivial term *iff* $|\mathcal{F}un(t)| \leq 1$
- $\nabla(t) = \{t|_u \mid u \in \mathcal{P}os(t), t|_u$ *is not trivial*$\}$ *is the set of* non-trivial subterms *of* t. $\diamond$

[7] If the checks for confluence and cooperation fail, it is possible to extend $\mathcal{S}$ and $\mathcal{T}$ such that the properties hold (see subsection 3.5).

An atom and a term of the form $f(x_1, \ldots, x_n)$ are the only possible trivial terms. Based on the notion of trivial terms, the set of non-trivial subterms of a term t is defined. For instance, let $\mathcal{F} = \{f, g, h, a\}$ and $t = h(f(g(x), g(a)))$. Then the set of non-trivial subterms $\nabla(t)$ of t is $\{g(a), f(g(x), g(a)), t\}$.

Definition 3.2 (Set of Projections) *Let $\{t_1, \ldots, t_n\} \subseteq \mathcal{T}(\mathcal{F},\mathcal{X})$ be terms and $\{h_1, \ldots, h_n\}$ new distinct operators (i.e. $\{h_1, \ldots, h_n\} \cap \mathcal{F} = \emptyset$). Then, the set of* projections *for $\{t_1, \ldots, t_n\}$ is defined as*

$$\Pi(\{t_1, \ldots, t_n\}) = \{t_i \to h_i(x_1, \ldots, x_{m_i}) \mid \mathcal{V}ar(t_i) = \{x_1, \ldots, x_{m_i}\}\} \qquad \diamond$$

Thus, the set of projections for some terms consists of a set of rewriting rules, one rule for each term, and maps the terms into trivial terms. For example, $\{f(x,a) \to h_1(x), g(f(a,x),y) \to h_2(x,y)\}$ is the set of projections corresponding to the set $\{f(x,a), g(f(a,x),y)\}$ if $\mathcal{F} = \{f, g, a\}$.

Note that the projection rules can be oriented w.r.t. any reasonable[8] term ordering if the left-hand sides are not variables (since the set of variables of the right-hand sides are identical to the corresponding ones of the left-hand sides and the operators of the right-hand sides have been newly introduced).

Definition 3.3 (Set of Normal Forms) *Let $\mathcal{R}$ be a terminating (not necessarily confluent) TRS over $\mathcal{T}(\mathcal{F},\mathcal{X})$ and $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$. Then, the set $\mathcal{R}(t)$ of* normal forms *of t w.r.t. $\mathcal{R}$ is defined as*

$$\mathcal{R}(t) = \{t' \mid t \xrightarrow{*}_{\mathcal{R}} t' \ \wedge \ \nexists t'' : t' \to_{\mathcal{R}} t''\}.$$

$t{\Downarrow_{\mathcal{R}}}$ denotes any term $t' \in \mathcal{R}(t)$. $\diamond$

Definition 3.4 (Set of Critical Terms) *Let $l, r \in \mathcal{T}(\mathcal{F},\mathcal{X})$ such that r is non-trivial, $l \to r$ be a rewriting rule and $\succ_{\mathsf{T}}$ be a term ordering such that $l \not\succ_{\mathsf{T}} r$. Then, a set of* critical terms *within r w.r.t. $\succ_{\mathsf{T}}$ is defined as*

$$\Upsilon(l \to r, \succ_{\mathsf{T}}) = M \subseteq \nabla(r)$$

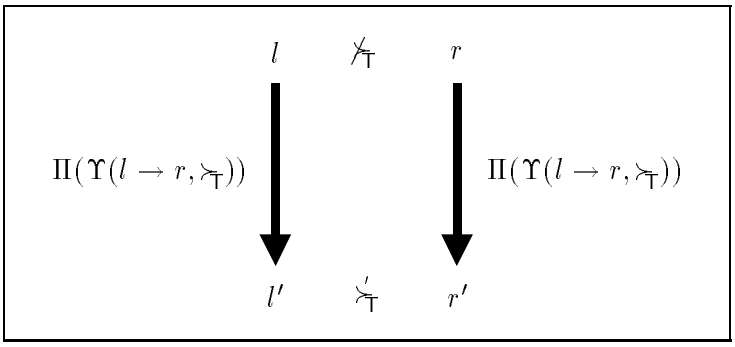
such that $l{\Downarrow_{\Pi(M)}} \not\succ'_{\mathsf{T}} r{\Downarrow_{\Pi(M)}}$ with $\succ'_{\mathsf{T}}$ being a term ordering and $\succ'_{\mathsf{T}} \supseteq \succ_{\mathsf{T}}$. $\diamond$

Note that $\Upsilon(l \to r, \succ_{\mathsf{T}})$ is *not* unique. However, it always exists for any rule $l \to r$ and any $\succ_{\mathsf{T}}$ if $M = \{r\}$ since r is a non-trivial term (i.e. it contains at least *two* operators) and the right-hand side of the projection of r contains *only one* operator which is *new*. As mentioned previously, $\succ_{\mathsf{T}}$ must be improvable in an incremental way.

The technique of the TO is able to extend each term ordering. We use a (classical) term ordering (like the RPOS) for proving termination and if this ordering fails to orient an equation (or a rule), it will be improved by a TO with appropriate TRS $\mathcal{S}$ and $\mathcal{T}$. Inability to prove the termination of a given rule $l \to r$ with the basic term ordering (used in the TO) is the precondition for computing a set of critical terms. Such a set contains all those non-trivial subterms of the right-hand side r of the (non-orientable) rule $l \to r$ which are

[8] i.e. improvable in an incremental way

responsible for the non-orientation of $l \to r$. In other words, if there exists a set M of non-trivial subterms of the right-hand side and the replacement (in l and r) of the elements of M by the right-hand sides of their projections leads to an orientation w.r.t. (an extension $\succ'_{\mathsf{T}}$ of) the basic term ordering $\succ_{\mathsf{T}}$, then M is a set of critical terms $\Upsilon(l \to r, \succ_{\mathsf{T}})$. The following diagram illustrates this procedure.



Example 3.1 *Let* $\mathcal{F} = \{f, g, h, a, b\}, l = h(f(g(x), g(b)))$ *and* $r = g(f(h(x), h(a)))$. *We use the* RPOS *based on the precedence* $f \succ h \succ g$ *and* $b \succ a$. *Note that* $l \not\succ_{\mathsf{RPOS}} r$.

The set of non-trivial subterms of r *is* $\nabla(r) = \{h(a), f(h(x), h(a)), r\}$. *The set of projections of this set is defined as* $\Pi(\nabla(r)) = \{h(a) \to h_1, f(h(x), h(a)) \to h_2(x), r \to h_3(x)\}$. *We enumerate all possible values a set* $\Upsilon(l \to r, \succ_{\mathsf{RPOS}})$ *of critical terms can have, together with appropriate extensions of the precedence:*

- $\{f(h(x), h(a))\}$, $h \succ h_2$
- $\{r\}$, $h \succ h_3$
- $\{f(h(x), h(a)), r\}$, $h \succ h_2 \wedge h \succ h_3$
- $\nabla(r)$, $h \succ h_2 \wedge h \succ h_3$ *i.e.* $h(a)$ *is superfluous* ◇

Our algorithm uses the projection rules w.r.t. the set of critical terms as part of the system $\mathcal{T}$ of the TO. Note that the set of critical terms may include non-linear terms. Since these terms could become left-hand sides of rules in $\mathcal{T}$, we have to generalize them (since $\mathcal{T}$ must contain left-linear rules, only). We do this with the help of the subsumption ordering and define the so-called *LL-generalization* (left-linear generalization) of a projection rule.

Definition 3.5 (LL-Generalization of Projection Rules) *Let* $l \to r$ *be a projection rule. Then, an* LL-generalization Γ *of* $l \to r$ *is defined as*

$$\Gamma(l \to r) = \begin{cases} l \to r & \text{if } l \text{ is linear} \\ l' \to r' & \text{otherwise} \end{cases}$$

where $l \trianglerighteq l' \notin \mathcal{X}$, l' *is linear and* $\Pi(\{l'\}) = \{l' \to r'\}$. ◇

For example, $\Gamma(f(a, x) \to h(x)) = f(a, x) \to h(x)$ whereas $\Gamma(f(x, g(y, x)) \to h(x, y)) = f(x, g(y, z)) \to h_1(x, y, z)$. Note that LL-generalization is *not* unique.

Algorithm 3.1 ($\mathcal{S}$-$\mathcal{T}$-Generation Technique) *Let $\mathcal{R}$ be a TRS for which a termination proof is needed. Furthermore, let $\succ_\mathsf{T}$ be a term ordering. We start with empty sets $\mathcal{S}$ and $\mathcal{T}$. The following procedure possibly generates $\mathcal{S}$ and $\mathcal{T}$.*

1. Orient with the basic term ordering:

$$\begin{array}{lcl} \mathcal{S} & := & \{l \to r \in \mathcal{R} \mid l \succ_\mathsf{T} r\} \\ \mathcal{R}' & := & \mathcal{R} \setminus \mathcal{S} \end{array}$$

2. Semantic transformation (normalization):

$$\begin{array}{lcl} \mathcal{S}' & := & \{l{\Downarrow_\mathcal{S}} \to r{\Downarrow_\mathcal{S}} \mid l \to r \in \mathcal{R}', l{\Downarrow_\mathcal{S}} \succ_\mathsf{T} r{\Downarrow_\mathcal{S}}\} \\ \mathcal{T} & := & \{l \to r \in \mathcal{S} \mid l \to r \text{ was used to compute } u{\Downarrow_\mathcal{S}} \text{ or } v{\Downarrow_\mathcal{S}} \text{ for } u \to v \in \mathcal{R}'\} \\ \mathcal{R}' & := & \mathcal{R}' \setminus \{l \to r \in \mathcal{R} \mid l{\Downarrow_\mathcal{S}} \to r{\Downarrow_\mathcal{S}} \in \mathcal{S}'\} \\ \mathcal{S} & := & (\mathcal{S} \setminus \mathcal{T}) \cup \mathcal{S}' \end{array}$$

3. Generate projection rules:

$$\begin{array}{lcl} \mathcal{T} & := & \mathcal{T} \cup \{l' \to r' \mid l' \to r' = \Gamma(l'' \to r''), l'' \to r'' \in \Pi(\Upsilon(l \to r, \succ_\mathsf{T})), l \to r \in \mathcal{R}'\} \\ \mathcal{S} & := & \mathcal{S} \cup \{l{\Downarrow_\mathcal{T}} \to r{\Downarrow_\mathcal{T}} \mid l \to r \in \mathcal{R}'\} \\ \mathcal{R}' & := & \emptyset \end{array}$$

   $\succ_\mathsf{T}$ is extended by Π and the orientation of $l{\Downarrow_\mathcal{T}} \to r{\Downarrow_\mathcal{T}}$

4. Check the required properties of $\mathcal{S}$ and $\mathcal{T}$:

   If $\mathcal{T}$ is confluent, variable-preserving and left-linear, $\mathcal{S}$ cooperates with $\mathcal{T}$ and $\forall l \to r \in \mathcal{R}\colon l \succ_{\mathsf{TO}} r$
   then return $\mathcal{S}$ and $\mathcal{T}$
   else fail ◇

Note that $\mathcal{R}'$ contains non-orientable rules of $\mathcal{R}$ while $\mathcal{S}$ contains orientable rules of $\mathcal{R}$. All rules of $\mathcal{R}'$ which can be oriented after reducing them are part of $\mathcal{S}'$. The tests in step 4 for variable-preservation and left-linearity of $\mathcal{T}$ are necessary since in step 2 arbitrary rules are added to $\mathcal{T}$. Furthermore, we have to check (in step 4) whether $\mathcal{R}$ can be oriented with the help of the induced TO because $\mathcal{T}$ is dynamically constructed[9]. However, there is no need to test the termination of $\mathcal{S} \cup \mathcal{T}$ since this property holds by construction of $\succ_\mathsf{T}$ (which contains $\xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}}$).

Theorem 3.1 *Algorithm 3.1 always terminates. If it does not fail, $\mathcal{R}$ is proved to be terminating with the help of the TO defined by $\mathcal{S}$ and $\mathcal{T}$.* ◇

Example 3.2 (Difference) *Let $\mathcal{R}$ be the TRS containing the rules*

$$\begin{array}{lrcl} 1\text{:} & 0 - y & \to & 0 \\ 2\text{:} & x - 0 & \to & x \\ 3\text{:} & x - s(y) & \to & \mathit{if}(x > s(y), s(x - p(s(y))), 0) \\ 4\text{:} & p(0) & \to & 0 \\ 5\text{:} & p(s(x)) & \to & x \end{array}$$

[9] As various examples show, the condition '$\forall l \to r \in \mathcal{R}\colon l \succ_{\mathsf{TO}} r$' of algorithm 3.1 is usually fulfilled.

partially specifying the difference operation on natural numbers. Algorithm 3.1 performs the following steps if we use the RPOS *with an empty precedence as the basic term ordering* $\succ_{\mathsf{T}}$*:*

1. $\mathcal{S} = \{1,2,4,5\}$
   $\mathcal{R}' = \{3\}$

2. $\mathcal{S}' = \{x - s(y) \to \mathit{if}(x > s(y), s(x-y), 0)\}$
   if $\succ$ *is extended by* $- \succ \mathit{if}, - \succ >, - \succ s, - \succ 0$
   $\mathcal{T} = \{5\}$
   $\mathcal{R}' = \emptyset$
   $\mathcal{S} = \{1,2,4\} \cup \mathcal{S}'$

3. *This step is not performed since* $\mathcal{R}'$ *is empty.*

4. $\mathcal{T} = \{5\}$
   $\mathcal{S} = \{1,2,4\} \cup \mathcal{S}'$
   Note that both the confluence and the cooperation are satisfied and for each rule $l \to r \in \mathcal{R}, l \succ_{\mathsf{TO}} r$ *holds using the above transformation systems* $\mathcal{S}$ *and* $\mathcal{T}$*.* ◇

We consider another TRS specifying intervals of natural numbers, i.e. the binary function int will be 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.

Example 3.3 (Intervals of Natural Numbers) *The following TRS* $\mathcal{R}$ *specifies intervals of natural numbers:*

1:	$int(0,0)$	$\to$	$0.nil$
2:	$int(0,s(y))$	$\to$	$0.int(s(0),s(y))$
3:	$int(s(x),0)$	$\to$	nil
4:	$intlist(nil)$	$\to$	nil
5:	$intlist(x.y)$	$\to$	$s(x).intlist(y)$
6:	$int(s(x),s(y))$	$\to$	$intlist(int(x,y))$

Assume that the RPOS *with the precedence* $int \succ intlist \succ . \succ s \succ nil$ *is given. Algorithm 3.1 performs the following steps:*

1. $\mathcal{S} = \{1,3,4,5,6\}$
   $\mathcal{R}' = \{2\}$

2. $\mathcal{S}' = \{int(0,s(y)) \to 0.intlist(int(0,y))\}$
   $\mathcal{T} = \{6\}$
   $\mathcal{R}' = \emptyset$
   $\mathcal{S} = \{1,3,4,5\} \cup \mathcal{S}'$

3. *This step is not performed since* $\mathcal{R}'$ *is empty.*

4. $\mathcal{S}$ *and* $\mathcal{T}$ *have the desired properties.* ◇

Example 3.4 (Greatest Common Divisor) *Let $\mathcal{R}$ be the TRS including the rules*

$$\begin{array}{lrcl} 1: & gcd(x,0) & \to & x \\ 2: & gcd(0,y) & \to & y \\ 3: & gcd(s(x),s(y)) & \to & if(x<y, gcd(s(x),y-x), gcd(x-y,s(y))) \end{array}$$

representing the greatest common divisor of two natural numbers. We apply our algorithm to $\mathcal{R}$ for generating an appropriate TO*. We start with the* RPOS *based on the empty precedence.*

1. $\mathcal{S} = \{1,2\}$
   $\mathcal{R}' = \{3\}$
2. *It is impossible to apply even one rule of $\mathcal{S}$ to rule 3.*
3. $\mathcal{T} = \{gcd(s(x),y-z) \to h_1(x,y,z),\ gcd(x-y,s(z)) \to h_2(x,y,z)\}$
   Note that the smallest (w.r.t. the size) critical subterms of the right-hand side are $gcd(s(x),y-x)$ and $gcd(x-y,s(y))$. Thus, after generalizing, $\mathcal{T}$ consists of the projection of these terms. In order to prove the termination of the rules of $\mathcal{T}$,

   we extend the precedence by $gcd \succ h_1, gcd \succ h_2$.

   $\mathcal{S} = \{1,2\} \cup \{gcd(s(x),s(y)) \to if(x<y, h_1(x,y,x), h_2(x,y,y))\}$
   which can be oriented by

   extending the precedence by $gcd \succ if$, $gcd \succ <$.

   Now, $\mathcal{R}'$ is empty.
4. *$\mathcal{S}$ and $\mathcal{T}$ have the demanded properties.* ◇

Appendix A contains further examples concerning the problem of finding a TO for proving termination of TRS. For some of these TRS, our algorithm automatically generates an appropriate TO. Obviously, the technique introduced is (and *can only be*) a heuristic and not a decision procedure. We conclude this section by discussing some remarks, uncertainties and extensions of algorithm 3.1.

3.1 Termination of Oriented Equations

Algorithm 3.1 is designed to prove the termination of a given TRS. It is not difficult (but more time consuming) to generalize this technique such that a given set of *equations* can be transformed into a provably terminating TRS.

3.2 Towards the Basic Ordering

It is obvious that the basic ordering $\succ_\mathsf{T}$ can be refined during algorithm 3.1. Thus, one can start with a weak (or even empty) ordering and incrementally strengthen it. For example, the precedence of an RPOS or a Knuth-Bendix ordering KBO (see [Knuth and Bendix, 1970]) may be extended. The basic ordering $\succ_\mathsf{T}$ *needs* to be improved in step 3 of the algorithm ('generating projection rules') since $\mathcal{F}$ will be enriched.

A problem concerning the basic ordering is that there sometimes exists the possibility of selecting different sets of incomparable rules w.r.t. $\succ_\mathsf{T}$.

Example 3.5 *Consider the following TRS $\mathcal{R}$:*

$$\begin{array}{lrcl}
1: & p(s(x)) & \to & x \\
2: & p(0) & \to & 0 \\
3: & f(x,y,z) & \to & g(x \leq y, x, y, z) \\
4: & g(true,x,y,z) & \to & z \\
5: & g(false,x,y,z) & \to & f(f(p(x),y,z),f(p(y),z,x),f(p(z),x,y))
\end{array}$$

Using the RPOS*, we are able to orient rule 3 in the desired direction by including the relations $f \succ g$ and $f \succ \leq$. This decision implies incomparability of the terms in rule 5. Thus, $\mathcal{T}$ must contain a subterm of the right-hand side of this rule which leads to non-cooperation of $\mathcal{S}$ and $\mathcal{T}$. However, by orienting rule 5 with the* RPOS *based on $g \succ f$ and $g \succ p$, algorithm 3.1 will be successful. It generates the systems*

$$\begin{array}{l}
\mathcal{T} = \{g(x \leq y, z, u, v) \to h_1(x, y, z, u, v)\} \quad \textit{and} \\
\mathcal{S} = \{1, 2, 4, 5\} \cup \{f(x, y, z) \to h_1(x, y, x, y, z)\}
\end{array}$$

and extends the precedence by $f \succ h_1$. ◇

3.3 Choosing an Appropriate Normal Form

Another subtle point of our algorithm, requiring careful treatment, is the use of the operations $t{\Downarrow_{\mathcal{S}}}$ and $t{\Downarrow_{\mathcal{T}}}$ for choosing a term from the set of normal forms derived from a term t.

In step 2 and step 3 of our approach, *sets* of normal forms of terms will be computed. It is obvious that we may replace *none* of these operations ($t{\Downarrow_{\mathcal{S}}}$ and $t{\Downarrow_{\mathcal{T}}}$) by the computation of *the* normal form ($t{\downarrow_{\mathcal{S}}}$ and $t{\downarrow_{\mathcal{T}}}$) since the systems $\mathcal{S}$ and $\mathcal{T}$ are not necessarily confluent (at the time of their application). We assume that, at least in the case of $\mathcal{T}$, the normal forms are often unique (since the systems in most cases contain very few rules, only). However, we need a good heuristic for choosing a convenient normal form (because these normal forms are important for step 4). Such a heuristic is always applied to a rule $l \to r$ (which has to be transformed into a rule $l{\Downarrow_{\mathcal{R}}} \to r{\Downarrow_{\mathcal{R}}}$ for any $\mathcal{R}$) and should possess (at least) the following properties:

(i) The chosen normal form of l must be greater than that of r w.r.t. $\succ_{\mathsf{T}}$.

(ii) The term $l{\Downarrow_{\mathcal{T}}}$ ($l{\Downarrow_{\mathcal{S}}}$) should not overlap with left-hand sides of $\mathcal{S}$ ($\mathcal{T}$, respectively).

Property (ii) is also important for LL-generalization of projection rules occurring in step 3 of the algorithm.

3.4 On Generalizing Terms

LL-generalization is necessary since the projection rules are part of $\mathcal{T}$ and $\mathcal{T}$ must be left-linear in order to apply theorem 2.2. As mentioned, LL-generalization is not uniquely determined (for example, $f(g(f(x,x)),y)$ can be either generalized to $f(g(f(x,z)),y)$ or to $f(g(x),y)$ or to $f(x,y)$). It is obvious that the more general the left-hand sides of $\mathcal{T}$-rules are the more possibilities of overlappings (w.r.t. $\mathcal{S}$ and $\mathcal{T}$) exist and the more difficult it will be to guarantee the properties required in step 4. Thus, LL-generalization should compute the greatest (w.r.t. $\succcurlyeq$) generalization of a projection rule which leads to the

left-linearity. Note that the greatest generalization is identical to the replacement of all non-linear variables by new and unique ones.

Obviously, it is possible to generalize also left-linear rules (this case is not covered in definition 3.5).

Another more general technique for generalizing terms should be available if non-left-linear rules (which are part of the reduction process for generating $\mathcal{S}'$) are added to $\mathcal{T}$ in step 2.

3.5 Integration of a Completion Procedure for Cooperation

An obvious extension of our algorithm concerns step 4. Here, we only check the required properties of $\mathcal{S}$ and $\mathcal{T}$. The application of the completion procedure for eventually extending $\mathcal{T}$ to a confluent system as well as the application of the algorithm of [Bellegarde and Lescanne, 1990] for extending $\mathcal{S}$ such that $\mathcal{S}$ cooperates with $\mathcal{T}$ could sometimes be a helpful improvement of our algorithm[10]. Note that after completing $\mathcal{T}$ one has to check its variable-preservation and left-linearity once again[11].

Example 3.6 ([Bachmair, 1987]) *Let $\mathcal{R}$ be the TRS specified by*

$$\begin{array}{lrcl} 1: & f(h(x)) & \to & f(i(x)) \\ 2: & g(i(x)) & \to & g(h(x)) \\ 3: & h(a) & \to & b \\ 4: & i(a) & \to & b \end{array}$$

Algorithm 3.1 generates $\mathcal{T} = \{g(h(x)) \to q(x)\}$ and $\mathcal{S} = \{1,3,4\} \cup \{g(i(x)) \to q(x)\}$. $\mathcal{S}$ does not cooperate with $\mathcal{T}$ since the non-cooperative critical pair $\langle q(a), g(b)\rangle$ exists by overlapping $g(h(x))$ and $h(a)$. However, by adding the rule $q(a) \to g(b)$ to $\mathcal{S}$, the required properties are satisfied (e.g. the RPOS *based on the total precedence $a \succ g \succ q \succ h \succ i \succ b$ can be used as the basic ordering).* ◇

3.6 Using Semantics of Operators

The algorithm can sometimes be further improved by integrating semantics (for example, in the form of lemmata of the inductive theory) of the used operators. We made the experience that the use of the semantics of the operators belonging to $\mathcal{T}$ facilitates the generation of an appropriate TO.

Example 3.7 *Consider the following (unusual) specification $\mathcal{R}$ of the factorial function ([Kamin and Lévy, 1980]):*

$$\begin{array}{lrcl} 1: & p(s(0)) & \to & 0 \\ 2: & p(s(s(x))) & \to & s(p(s(x))) \\ 3: & fac(s(x)) & \to & fac(p(s(x))) \star s(x) \end{array}$$

[10] Obviously, it is possible to complete $\mathcal{T}$ (for confluence) as well as $\mathcal{S}$ (for cooperation) whenever extending these sets by a new rule (instead of completing them only just in step 4).

[11] since it is possible that the initial system satisfies the properties and the corresponding canonical one does not. For example, the TRS $\mathcal{R} = \{x.e \to f(x,x,x), g((x.y).z) \to f(x,y,z)\}$ is variable-preserving and left-linear. The canonical version $\mathcal{R}' = \mathcal{R} \cup \{g(f(x.y, x.y, x.y)) \to f(x,y,e), g(f(x,x,x).y) \to f(x,e,y), g(f(f(x,x,x), f(x,x,x), f(x,x,x))) \to f(x,e,e)\}$ of $\mathcal{R}$ is *not* left-linear.

Due to algorithm 3.1, we have to choose $\mathcal{T}$ as $p(s(x)) \to h_1(x)$ or $fac(p(s(x))) \to h_1(x)$ or $fac(p(s(x))) \star s(y) \to h_1(x,y)$. However, all these possibilities lead to non-cooperation of $\mathcal{S}$ and $\mathcal{T}$. Note that p stands for predecessor whereas s is the abbreviation for successor. Thus, $p(s(x)) = x$ holds using the semantics of p and s. If $\mathcal{T}$ is represented by the rule $p(s(x)) \to x$ and $\mathcal{S}$ to be $fac(s(x)) \to fac(x) \star s(x)$ both systems have the demanded properties. ◇

Example 3.8 *Consider the TRS $\mathcal{R}$ for computing the addition of two natural numbers:*

$$\begin{array}{lrcl} 1: & 0+y & \to & y \\ 2: & s(x)+0 & \to & s(x) \\ 3: & s(x)+s(y) & \to & s(s(x)+(y+0)) \end{array}$$

For generating $\mathcal{T}$ we exploit the fact that $+$ is commutative and thus take the commutative version of rule 1 as $\mathcal{T}$:

$$\mathcal{T} = \{x+0 \to x\} \quad \text{and} \quad \mathcal{S} = \{1\} \cup \{s(x)+s(y) \to s(s(x)+y)\}$$

All required conditions are fulfilled. ◇

In general, the incorporation of the semantics of operators leads to final versions of $\mathcal{S}$ and $\mathcal{T}$ which are structurally easier than without semantics.

3.7 A Well-Known Example

We give an annotation to an often referenced example of Dershowitz.

Example 3.9 ([Dershowitz, 1987])

$$f(f(x)) \to f(g(f(x)))$$

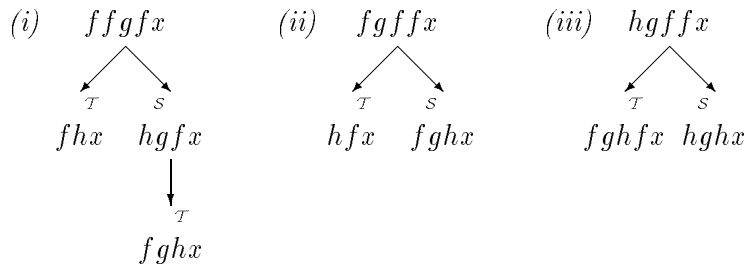
For simplicity, we use string notation, i.e. $fffx \to fgfx$. The presented technique is not able to automatically prove the termination of this rule because cooperation of $\mathcal{S}$ with $\mathcal{T}$ cannot be guaranteed. Consider algorithm 3.1 together with the completion component for $\mathcal{T}$ and $\mathcal{S}$:

$$\mathcal{T} = \{fgfx \to hx\} \qquad \mathcal{S} = \{fffx \to hx\}$$

In order to complete $\mathcal{T}$, we have to add either $hgfx \to fghx$ or $fghx \to hgfx$. We take the former rule:

$$\mathcal{T} = \{fgfx \to hx \ , \ hgfx \to fghx\}$$

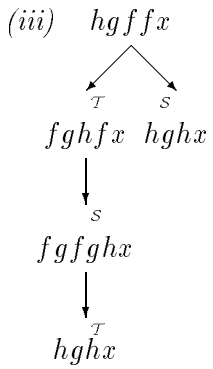
is confluent. Now, there exist three overlappings between left-hand sides of rules in $\mathcal{S}$ and in $\mathcal{T}$:



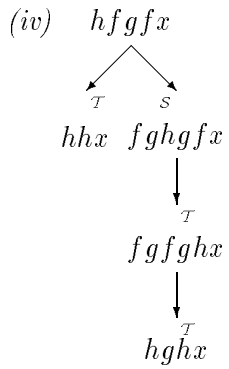
The orientation of (i) such that $fhx \rightarrow_{\mathcal{S}} fghx$ is similar to the problem of proving the termination of $ffx \rightarrow fgfx$. The second critical pair can be oriented from left to right

$$\mathcal{S} = \{ffx \rightarrow hx \ , \ hfx \rightarrow fghx\}$$

which implies the reduction as well as the cooperation of (iii):



However, another critical pair exists by overlapping the left-hand side of the new rule $hfx \rightarrow fghx$ of $\mathcal{S}$ with the left-hand side of the rule $fgfx \rightarrow hx$ of $\mathcal{T}$:



The orientation of this critical pair ($hhx \rightarrow_{\mathcal{S}} hghx$) is identical to that of $ffx \rightarrow fgfx$ if $\mathbf{f} \sim \mathbf{h}$. *Thus, instead of generating $\mathcal{T} = \{fgfx \rightarrow hx\}$ we produce $\mathcal{T} = \{fgfx \rightarrow fx\}$ (since h is identical to f) and start algorithm 3.1 once again. It provides the following transformation ordering:*

$$\mathcal{T} = \{fgfx \rightarrow fx\} \qquad \mathcal{S} = \{ffx \rightarrow fx\} \qquad \diamond$$

3.8 Using a Lemma of Bachmair and Dershowitz

In [Bellegarde and Lescanne, 1990], it is mentioned that the use of the following lemma, contained in [Bachmair and Dershowitz, 1986], helps to strengthen transformation orderings.

Lemma 3.1 ([Bachmair and Dershowitz, 1986]) *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two TRS. Then the combined TRS $\mathcal{R}_1 \cup \mathcal{R}_2$ is terminating iff both $\mathcal{R}_1/\mathcal{R}_2$ and $\mathcal{R}_2$ are terminating.* $\diamond$

Given two TRS $\mathcal{R}_1$ and $\mathcal{R}_2$, $\mathcal{R}_1/\mathcal{R}_2$ is called '$\mathcal{R}_1$ *modulo* $\mathcal{R}_2$' and stands for the relation $\mathcal{R}_2^*.\mathcal{R}_1.\mathcal{R}_2^*$. Note that $\mathcal{R}_1/\mathcal{R}_2$ and $\mathcal{R}_1/\mathcal{R}_2^*$ are identical.

The lemma listed above can be used in connection with transformation orderings as follows: Suppose the termination of a TRS $\mathcal{R}$ has to be proved. We split $\mathcal{R}$ into two completely disjoint parts $\mathcal{R}_1$ and $\mathcal{R}_2$ such that

- the termination proof of $\mathcal{R}_2$ can easily be performed (e.g. $\mathcal{R}_2 = \{l \to r \in \mathcal{R} \mid l \succ_{\mathsf{RPOS}} r\}$ w.r.t. a given RPOS),
- $\mathcal{R}_1$ represents the remaining part, i.e. $\mathcal{R}_1 = \mathcal{R} \setminus \mathcal{R}_2$,
- there exists a TO based on the TRS $\mathcal{T}$ and $\mathcal{S}$ proving the termination of $\mathcal{R}_1$ and
- $\mathcal{R}_2 \subseteq (\mathcal{T} \cup \mathcal{T}^{-1})^*$.

These four conditions ensure the termination of $\mathcal{R}_1/\mathcal{R}_2$ (with a TO) and $\mathcal{R}_2$ (with a 'classical' termination proof method). Then, $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ terminates since lemma 3.1 holds. For more details on the *theoretical* background of this procedure, see [Bachmair and Dershowitz, 1986] and [Bellegarde and Lescanne, 1990].

From a practical point of view, the most important benefit of such a split of $\mathcal{R}$ (into $\mathcal{R}_1$ and $\mathcal{R}_2$) is that the term orderings for proving the termination of $\mathcal{R}_1$ and that of $\mathcal{R}_2$ can be totally different ones (they can even overlap one to the other). Now, we show that this property is of significance in practice by considering three examples[12]. In general, we propose the following strategy for generating appropriate TRS $\mathcal{S}$ and $\mathcal{T}$:

- Let $\mathcal{T}$ be $\mathcal{R}_2^{-1}$, possibly enriched by the semantics of all or some operators occurring on the left-hand side of the rules in $\mathcal{T}$.
- If necessary, try to expand $\mathcal{T}$ such that $l\downarrow_{\mathcal{T}} = r\downarrow_{\mathcal{T}}$ for all rules $l \to r$ of $\mathcal{R}_1$ holds. This implies $\mathcal{S}$ to be empty and the cooperation to be obviously satisfied. The termination of $\mathcal{R}_1$ has to be ensured with the ordering proving the termination of $\mathcal{T}$.

Example 3.10 (Binary Trees) *Let* $\mathcal{R}$ *be*

$$
\begin{array}{lrcl}
1: & f(nil) & \to & nil \\
2: & f(nil.x) & \to & nil.f(x) \\
3: & f((x.y).z) & \to & f(x.(y.z)) \\
4: & g(nil) & \to & nil \\
5: & g(x.nil) & \to & g(x).nil \\
6: & g(x.(y.z)) & \to & g((x.y).z)
\end{array}
$$

Suppose $\mathcal{R}_1 = \{4,5,6\}$ *and*
$\mathcal{R}_2 = \{1,2,3\}$.
Furthermore, let $\mathcal{T} = \{1,4,6\} \cup \{f(x.(y.z)) \to f((x.y).z)\,,\ nil.x \to x\,,\ x.nil \to x\}$ *and*
$\mathcal{S} = \emptyset$.

[12] see appendix A for more examples

Then, the following properties are satisfied:

- $\mathcal{R}_2$ *is included in the* RPOS *based on* $f \succ .$ *and* $\tau(.) = left$.
- $\mathcal{R}_2 = \{1,2,3\} \subseteq (\mathcal{T} \cup \mathcal{T}^{-1})^*$ *since rule 1 is contained in* $\mathcal{T}$, *rule 2 can be simulated by* $nil.x \to x$ *and rule 3 is contained in* $\mathcal{T}^{-1}$.
- *The TRS* $\mathcal{T}$ *and* $\mathcal{S}$ *represent a correct* TO *since* $\mathcal{T}$ *is confluent (all critical pairs are joinable),* $\mathcal{S}$ *cooperates with* $\mathcal{T}$ *(*$\mathcal{S}$ *is empty) and* $\mathcal{S} \cup \mathcal{T}$ *is terminating (w.r.t. the* RPOS *based on* $g \succ .$ *and* $\tau(.) = right$*).*
- *The TRS* $\mathcal{R}_1$ *is included in the* TO *specified by* $\mathcal{S}$, $\mathcal{T}$ *and the* RPOS *given above:*

  1) $g(nil)\downarrow_{\mathcal{T}} = nil = nil\downarrow_{\mathcal{T}} \;\wedge\; g(nil) \succ_{\mathsf{RPOS}} nil$

  2) $g(x.nil)\downarrow_{\mathcal{T}} = g(x) = g(x).nil\downarrow_{\mathcal{T}} \;\wedge\; g(x.nil) \succ_{\mathsf{RPOS}} g(x).nil$

  3) $g(x.(y.z))\downarrow_{\mathcal{T}} = g((x.y).z) = g((x.y).z)\downarrow_{\mathcal{T}} \;\wedge\; g(x.(y.z)) \succ_{\mathsf{RPOS}} g((x.y).z)$ ◇

Example 3.11 *Let* $\mathcal{R}$ *be the TRS consisting of the rules*

$$\begin{array}{lrcl} 1: & f(0) & \to & s(0) \\ 2: & f(s(x)) & \to & s(s(g(x))) \\ 3: & g(0) & \to & 0 \\ 4: & g(s(x)) & \to & f(x) \end{array}$$

An RPOS *with the quasi-precedence* $g \sim f \succ s$ *allows to prove the termination of* $\mathcal{R}$. *Note that* f *specifies the successor function and* g *the identity on natural numbers. However, the corresponding semantics* $f(x) \to s(x)$, $g(x) \to x$ *do not help orienting* $\mathcal{R}$ *with the induced* TO *since* f *and* g *are mutually recursive. Nevertheless, consider the following* TO.

Let $\mathcal{R}_1 = \{1,2,3\}$,
$\mathcal{R}_2 = \{4\}$,
$\mathcal{T} = \{f(x) \to g(s(x))\}$ *and*
$\mathcal{S} = \{g(s(0)) \to s(0)\ ,\ g(s(s(x))) \to s(s(g(x)))\ ,\ g(0) \to 0\}$.
Then,

- $\mathcal{R}_2$ *is included in the* RPOS *based on* $g \succ f$.
- $\mathcal{R}_2 \subseteq (\mathcal{T} \cup \mathcal{T}^{-1})^*$ *since* $\mathcal{R}_2 = \mathcal{T}^{-1}$.
- *The* TO *specified by* $\mathcal{S}$, $\mathcal{T}$ *and the basic ordering* RPOS *with* $f \succ g \succ s$ *proves the termination of* $\mathcal{R}_1$ *since*

  1) $f(0)\downarrow_{\mathcal{T}} = g(s(0)) \to_{\mathcal{S}} s(0) = s(0)\downarrow_{\mathcal{T}}$

  2) $f(s(x))\downarrow_{\mathcal{T}} = g(s(s(x))) \to_{\mathcal{S}} s(s(g(x))) = s(s(g(x)))\downarrow_{\mathcal{T}}$

  3) $g(0)\downarrow_{\mathcal{T}} = g(0) \to_{\mathcal{S}} 0 = 0\downarrow_{\mathcal{T}}$ ◇

Example 3.12 (Walther) *Let* $\mathcal{R}$ *be*[13]

$$\left.\begin{array}{rrcl} 1: & f(0) & \to & s(0) \\ 2: & f(s(0)) & \to & s(0) \end{array}\right\} \mathcal{R}_2$$

$$\left.\begin{array}{rrcl} 3: & f(s(s(x))) & \to & f(f(s(x))) \end{array}\right\} \mathcal{R}_1$$

and

$$\mathcal{T} = \left\{\begin{array}{rcl} f(x) & \to & x \\ s(s(x)) & \to & s(x) \end{array}\right.$$

$$\mathcal{S} = \emptyset$$

Note that f *specifies the constant function* $s(0)$*. Thus,* $\mathcal{T}$ *models* f*. With the help of the* TO *induced by* $\mathcal{S}$*,* $\mathcal{T}$ *and the basic ordering* RPOS *with* $s \succ f$*,* $\mathcal{R}_1$ *can be proved to be terminating. The remaining properties are also satisfied.* ◇

Currently, we are implementing our algorithm ([Nessel, 1992]). A thorough investigation of it (i.e. the application to examples often occurring in practice) will be enclosed in order to get information for improving the algorithm.

4 Simulation of Some Well-Known String Orderings

In the previous section, we have developed a technique for automatically generating a transformation ordering for a given *TRS*. This section deals with a similar problem: We are generating transformation orderings for simulating *given orderings*. This problem is more interesting from a theoretical than from a practical point of view. In this article, we examine some well-known orderings on strings.

A *Semi-Thue system* over a set of *strings* Σ^* is a finite set of rules, each of the form $l \to r$, where l and r are *words*[14] in Σ^*. Σ^* is the *monoid* freely generated by a finite *alphabet* Σ under the operation '.' of concatenation, i.e. the set of all finite strings over Σ. The empty string ϵ is the identity in the monoid.

The transformation ordering TO on strings is defined as the TO on terms (see definition 2.2) with the exception that $\mathcal{S}$ and $\mathcal{T}$ are Semi-Thue systems and $\succ_{\mathsf{T}}$ is replaced by $\xrightarrow{+}_{\mathcal{T}}$. The replacement of the basic ordering $\succ_{\mathsf{T}}$ by the transformation system $\mathcal{T}$ implies the induced TO to become weaker (since $\xrightarrow{+}_{\mathcal{T}} \subseteq \succ_{\mathsf{T}}$ must hold). However, in this section we are interested in simulating orderings with terminating rule systems, only and *without* other orderings (which would not be the case if the TO were based on $\succ_{\mathsf{T}}$). Thus, in this section the *original* transformation ordering[15] TO of [Bellegarde and Lescanne, 1987] will be used which is defined as $u \succ_{\mathsf{TO}} v$ iff $u\downarrow_{\mathcal{T}} = v\downarrow_{\mathcal{T}} \wedge u \xrightarrow{+}_{\mathcal{T}} v$ or $u\downarrow_{\mathcal{T}} \neq v\downarrow_{\mathcal{T}} \wedge u\downarrow_{\mathcal{T}} \to_{\mathcal{S}} \cdot \xrightarrow{*}_{\mathcal{S}\cup\mathcal{T}} v\downarrow_{\mathcal{T}}$ (simultaneously satisfying the properties required in definition 2.2). Obviously, the test for the cooperation of $\mathcal{S}$ and $\mathcal{T}$, stated in theorem 2.2, can *always* be applied since each Semi-Thue system is left-linear and variable-preserving[16].

[13] see also example 20 of appendix A

[14] Words are also called strings.

[15] The TO based on $\succ_{\mathsf{T}}$ is defined at the end of [Bellegarde and Lescanne, 1987].

[16] Note that the subset of terms containing unary operators and variables, only (i.e. monadic terms without constants) can unequivocally be transformed into strings and vice versa. For example, fgf corresponds to $f(g(f(x)))$.

In the remaining part, we use a, b, c, d, a_i, b_i in order to denote letters of the alphabet Σ while u, v, x, y, x_i represent words over Σ^*. Furthermore, $\succ$ is a partial ordering whereas $\succsim$ is a quasi-ordering[17] on Σ. The equivalence relation $a.u \approx b.v$ on strings $a.u, b.v$ is defined as $a \sim b$ ($a = b$ if no quasi-ordering on Σ is given) and $u \approx v$.

Definition 4.1 (Division Ordering, [Higman, 1952], [Martin, 1989]) *A* division ordering[18] *on Σ^* is an ordering $\succ_{\mathsf{O}}$ satisfying*

- $u \succ_{\mathsf{O}} v \quad \hookrightarrow \quad xuy \succ_{\mathsf{O}} xvy$
- $u \succ_{\mathsf{O}} \epsilon$ ◇

It was shown by Higman that any division ordering is well-founded.

Definition 4.2 (Length Function) *The* length $|u|$ *of a word u represents the number of letters occurring in u. It is defined as*

- $|\epsilon| = 0$
- $|a.u| = |u| + 1$ ◇

Definition 4.3 (Lexicographical Extension) *Let $\succsim$ be any quasi-ordering on Σ. The relation $\succ^{lex}$ on words is defined as*

$$a.u \succ^{lex} b.v \iff a \succ b \quad or \quad a \sim b \wedge u \succ^{lex} v$$ ◇

In the following five subsections, we recapitulate the formal definitions of well-known division orderings including the subword ordering, the extended subword ordering, the length ordering, the length-lexicographical ordering and two versions of the Knuth-Bendix ordering. For each ordering, $\mathcal{S}$ and $\mathcal{T}$ are given such that the induced transformation ordering TO simulates the corresponding division ordering. The termination of $\mathcal{S} \cup \mathcal{T}$ is satisfied due to the construction of $\mathcal{S}$ and $\mathcal{T}$ (see the corresponding proofs contained in appendix B).

4.1 Subword Ordering

Definition 4.4 (Subword Ordering) *Let Σ be any alphabet. Then,*

$$a_1 \ldots a_m \succ_{\mathsf{SO}} b_1 \ldots b_n \iff a_1 \ldots a_m \neq b_1 \ldots b_n \wedge \exists 1 \leq i_1 < i_2 < \cdots < i_n \leq m : \forall j \in [1, n] : a_{i_j} = b_j$$ ◇

Lemma 4.1 (Simulation of the SO) *Let Σ be any alphabet,*

$$\mathcal{T} = \emptyset \quad and \quad \mathcal{S} = \{a \to \epsilon \mid a \in \Sigma\}.$$

Then, $u \succ_{\mathsf{SO}} v \iff u \succ_{\mathsf{TO}} v$. ◇

For example, $abbacd \succ_{\mathsf{SO}} bc$ and $abbacd \succ_{\mathsf{TO}} bc$ if $\mathcal{T} = \emptyset$ and $\mathcal{S} = \{a \to \epsilon, b \to \epsilon, c \to \epsilon, d \to \epsilon\}$ since $abbacd\downarrow_{\mathcal{T}} = abbacd \xrightarrow{+}_{\mathcal{S}} bc = bc\downarrow_{\mathcal{T}}$.

[17] $\succ$ and $\succsim$ are *precedences* on Σ.

[18] also called simplification ordering (see section 1)

4.2 Extended Subword Ordering

Definition 4.5 (Extended Subword Ordering, [Eschenbach, 1986]) *Let $\succsim$ be a quasi-ordering on an alphabet Σ. Then,*

$$\begin{aligned} & a_1 \ldots a_m \succ_{\mathsf{ESO}} b_1 \ldots b_n \\ \iff \quad & a_1 \ldots a_m \not\approx b_1 \ldots b_n \wedge \exists 1 \leq i_1 < i_2 < \cdots < i_n \leq m \colon \forall j \in [1,n] \colon a_{i_j} \succsim b_j \end{aligned} \qquad \diamond$$

Lemma 4.2 (Simulation of the ESO) *Let $\Sigma = \{a_i \mid i \in I\}$ be any alphabet and $\succsim$ be any quasi-precedence. Furthermore, let $\Sigma' = \{A_i \mid i \in I\}$ such that $\Sigma \cap \Sigma' = \emptyset$ and*

$$\begin{aligned} \mathcal{T} &= \{a_i \to A_k \mid k = min\{j \mid a_i \sim a_j\}\} \quad \textit{and} \\ \mathcal{S} &= \{A_m \to A_n \mid a_i \succ a_j, m = min\{k \mid a_k \sim a_i\}, n = min\{k \mid a_k \sim a_j\}\} \\ &\quad \cup \{A_i \to \epsilon \mid i \in I\}. \end{aligned}$$

Then, $u \succ_{\mathsf{ESO}} v \iff u \succ_{\mathsf{TO}} v$. $\diamond$

For example, let $\Sigma = \{a, b, c\}$ such that $a \sim b \succ c$. Then, $aaa \succ_{\mathsf{ESO}} bc$ and $aaa \succ_{\mathsf{TO}} bc$ if $\mathcal{T} = \{a \to A, b \to A, c \to C\}$ and $\mathcal{S} = \{A \to C, A \to \epsilon, C \to \epsilon\}$ since $aaa\downarrow_{\mathcal{T}} = AAA \stackrel{+}{\to}_{\mathcal{S}} AC = bc\downarrow_{\mathcal{T}}$. There will exist a simpler simulation of the ESO if the precedence is only a partial (not a quasi-) ordering: $\mathcal{T} = \emptyset$ and $\mathcal{S} = \{a \to b \mid a \succ b\} \cup \{a \to \epsilon \mid a \in \Sigma\}$.

4.3 Length Ordering

Definition 4.6 (Length Ordering) *Let Σ be any alphabet. Then,*

$$u \succ_{\mathsf{LEN}} v \iff |u| > |v| \qquad \diamond$$

Lemma 4.3 (Simulation of the LEN) *Let Σ be any alphabet and A a letter such that $A \notin \Sigma$. Furthermore, let*

$$\begin{aligned} \mathcal{T} &= \{a \to A \mid a \in \Sigma\} \quad \textit{and} \\ \mathcal{S} &= \{A \to \epsilon\}. \end{aligned}$$

Then, $u \succ_{\mathsf{LEN}} v \iff u \succ_{\mathsf{TO}} v$. $\diamond$

For example, let $\Sigma = \{a, b, c\}$. $abb \succ_{\mathsf{LEN}} ca$ and $abb \succ_{\mathsf{TO}} ca$ if $\mathcal{T} = \{a \to A, b \to A, c \to A\}$ and $\mathcal{S} = \{A \to \epsilon\}$ because $abb\downarrow_{\mathcal{T}} = AAA \to_{\mathcal{S}} AA = ca\downarrow_{\mathcal{T}}$.

4.4 Length-Lexicographical Ordering

Definition 4.7 (Length-Lexicographical Ordering) *Let $\succsim$ be any quasi-ordering on an alphabet Σ. Then,*

$$\begin{aligned} u \succ_{\mathsf{LLEX}} v \iff & \blacktriangleright \; u \succ_{\mathsf{LEN}} v \qquad \textit{or} \\ & \blacktriangleright \; |u| = |v| \wedge u \stackrel{lex}{\succ} v \end{aligned} \qquad \diamond$$

Lemma 4.4 (Simulation of the LLEX) *Let Σ be any alphabet, A and B are letters such that $A, B \notin \Sigma$ and $\succ$ is a precedence. Furthermore, let*

$$\begin{array}{rcll} \mathcal{T} & = & \{a \to bA \mid a \succ b\} & \cup \\ & & \{Aa \to bA \mid a,b \in \Sigma\} & \cup \\ & & \{A \to \epsilon\} & \cup \\ & & \{a \to B \mid a \in \Sigma\} & \textit{and} \\ \mathcal{S} & = & \{B \to \epsilon\}. & \end{array}$$

Then, $u \succ_{\mathsf{LLEX}} v \iff u \succ_{\mathsf{TO}} v$. ◇

Example 4.1 *Let* $\Sigma = \{a,b,c\}$ *and* $a \succ b, a \succ c$. $aabc \succ_{\mathsf{LLEX}} abab$ *and* $aabc \succ_{\mathsf{TO}} abab$ *if* $\mathcal{T} = \{a \to bA, a \to cA, Aa \to aA, Aa \to bA, Aa \to cA, Ab \to aA, Ab \to bA, Ab \to cA, Ac \to aA, Ac \to bA, Ac \to cA, A \to \epsilon, a \to B, b \to B, c \to B\}$ *and* $\mathcal{S} = \{B \to \epsilon\}$ *because* $aabc\downarrow_{\mathcal{T}} = B^4 = abab\downarrow_{\mathcal{T}}$ *and* $a\underline{a}bc \to_{\mathcal{T}} ab\underline{Abc} \to_{\mathcal{T}} aba\underline{Ac} \to_{\mathcal{T}} abab\underline{A} \to_{\mathcal{T}} abab$. ◇

4.5 Kunth-Bendix Orderings on Strings

Definition 4.8 (Weight Function, [Knuth and Bendix, 1970]) *Let* $\succ$ *be a precedence over* Σ.

- *The weight function* $\varphi_\Sigma : \Sigma \mapsto \mathbb{N}$ *assigns positive natural numbers to each letter in* Σ *with the possible exception*

$$\varphi_\Sigma(a) = 0 \qquad \textit{for only one } a \textit{ such that there is no } b \textit{ with } b \succ a.$$

- *This weight function on letters can be extended to words (denoted by* φ*) in such a way that*

$$\varphi(a.u) = \varphi_\Sigma(a) + \varphi(u).$$ ◇

Definition 4.9 (Simple Knuth-Bendix Ordering on Strings, [Knuth and Bendix, 1970]) *Let* φ_Σ *be a weight function on an alphabet* Σ. *Then,*

$$u \succ_{\mathsf{SKBO}} v \iff \varphi(u) > \varphi(v)$$ ◇

Lemma 4.5 (Simulation of the SKBO) *Let* Σ *be any alphabet,* A *a letter such that* $A \notin \Sigma$ *and* φ_Σ *be a weight function. Furthermore, let*

$$\begin{array}{rcl} \mathcal{T} & = & \{a \to A^{\varphi_\Sigma(a)} \mid a \in \Sigma\} \quad \textit{and} \\ \mathcal{S} & = & \{A \to \epsilon\}. \end{array}$$

Then, $u \succ_{\mathsf{SKBO}} v \iff u \succ_{\mathsf{TO}} v$. ◇

For example, let $\Sigma = \{a,b,c\}$ such that $\varphi_\Sigma(a) = 1, \varphi_\Sigma(b) = 0$ and $\varphi_\Sigma(c) = 2$. Then, $aaa \succ_{\mathsf{SKBO}} bbcbb$ and $aaa \succ_{\mathsf{TO}} bbcbb$ if $\mathcal{T} = \{a \to A, b \to \epsilon, c \to AA\}$ and $\mathcal{S} = \{A \to \epsilon\}$ since $aaa\downarrow_{\mathcal{T}} = AAA \to_{\mathcal{S}} AA = bbcbb\downarrow_{\mathcal{T}}$. The following Knuth-Bendix ordering on strings represents the direct transformation of the Knuth-Bendix ordering on terms given in [Knuth and Bendix, 1970].

Definition 4.10 (Knuth-Bendix Ordering on Strings, [Knuth and Bendix, 1970]) *Let* φ_Σ *be a weight function and* $\succ$ *be a precedence on an alphabet* Σ. *Then,*

$$\begin{array}{rcl} a.u \succ_{\mathsf{KBO}} b.v & \iff & \blacktriangleright \ \varphi(a.u) > \varphi(b.v) \qquad \textit{or} \\ & & \blacktriangleright \ \varphi(a.u) = \varphi(b.v) \wedge a \succ b \qquad \textit{or} \\ & & \blacktriangleright \ \varphi(a.u) = \varphi(b.v) \wedge a = b \wedge u \succ_{\mathsf{KBO}} v \end{array}$$ ◇

Our simulation of the KBO (see lemma 4.7) is mainly influenced by the following lemma.

Lemma 4.6 *Let $\succ$ be a partial precedence and φ_Σ be a weight function on an alphabet Σ. Then,*

$$a_1 \ldots a_m \succ^{lex} b_1 \ldots b_n \iff a_1^{\varphi'(a_1)} \ldots a_m^{\varphi'(a_m)} \succ^{lex} b_1^{\varphi'(b_1)} \ldots b_n^{\varphi'(b_n)}$$

where $\varphi'(a) = \begin{cases} \varphi(a) & \text{if } \varphi(a) > 0 \\ 1 & \text{otherwise} \end{cases}$ ◇

Lemma 4.7 (Simulation of the KBO Based on Positive Weights) *Let $\Sigma = \{a_1, \ldots, a_n\}$ be any alphabet, $\Sigma' = \{A_1, \ldots, A_n\}$ with $\Sigma \cap \Sigma' = \emptyset$ and $\$ \notin \Sigma \cup \Sigma'$. Furthermore, let φ_Σ be a weight function such that $\varphi_\Sigma(a_i) > 0$ and $\succ$ be a partial precedence. Let*

$$\begin{array}{lcll} \mathcal{T} & = & \{a_i \to A_i^{\varphi_\Sigma(a_i)} \mid i = 1, \ldots, n\} & \text{and} \\ \mathcal{S} & = & \{A_i A_j \to A_k \$ \mid i, j, k = 1, \ldots, n\} & \cup \\ & & \{\$ A_i \to A_j \$ \mid i, j = 1, \ldots, n\} & \cup \\ & & \{\$ \to \epsilon\} & \cup \\ & & \{A_i \to \epsilon \mid i = 1, \ldots, n\} & \cup \\ & & \{A_i \to A_j \$ \mid a_i \succ a_j\}. & \end{array}$$

Then, $u \succ_{\mathsf{KBO}} v \iff u \succ_{\mathsf{TO}} v$. ◇

The main idea of the simulation of the KBO *based on positive weights* is the following one:

- If $\varphi(s) > \varphi(t)$, then $s\downarrow_{\mathcal{T}} \neq t\downarrow_{\mathcal{T}}$ and $s\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} t\downarrow_{\mathcal{T}}$ by using the rules $A_i \to_{\mathcal{S}} \epsilon$ (analogous to lemma 4.5).

- Let $\varphi(s) = \varphi(t)$. Then, $s \succ^{lex} t$[19]. The normal forms of s and t w.r.t. $\mathcal{T}$ have the same length and the letters occurring in $s\downarrow_{\mathcal{T}}$ and $t\downarrow_{\mathcal{T}}$ are the 'expansions' of the letters occurring in s and t w.r.t. the weight function (i.e. $a_i \to_{\mathcal{T}} A_i^{\varphi_\Sigma(a_i)}$). Now, we have to prove $s\downarrow_{\mathcal{T}} \succ^{lex} t\downarrow_{\mathcal{T}}$ (by using lemma 4.6). This will be done in the following way. Let $s\downarrow_{\mathcal{T}} = w.A.s'$ and $t\downarrow_{\mathcal{T}} = w.B.t'$ with $w \in \Sigma^*$ and $A, B \in \Sigma$. Then, $A.s' \xrightarrow{+}_{\mathcal{S}} B.t'$ by applying $A \to B\$$ (a has to be greater than b since $s \succ^{lex} t$) and then rules of the form $\$ A_i \to A_j \$$ for changing s' to t'.

Example 4.2 *Let $\Sigma = \{a, b, c\}$ such that $\varphi_\Sigma(a) = 1, \varphi_\Sigma(b) = 2, \varphi_\Sigma(c) = 3$ and $a \succ b$. Then, caca $\succ_{\mathsf{KBO}}$ cbba and caca $\succ_{\mathsf{TO}}$ cbba if*

[19] This is a slight modification of definition 4.10. However, both definitions are equivalent since we do not use quasi-precedences $\succsim$ where different syntactical but $\sim$-equivalent letters may possess different weights.

$$
\begin{array}{lllllllllllll}
\mathcal{T} & = & \{ & a & \to & A & b & \to & BB & c & \to & CCC & \} \quad and \\
\mathcal{S} & = & \{ & AB & \to & A\$ & AB & \to & B\$ & AB & \to & C\$ & \\
 & & & BA & \to & A\$ & BA & \to & B\$ & BA & \to & C\$ & \\
 & & & AC & \to & A\$ & AC & \to & B\$ & AC & \to & C\$ & \\
 & & & CA & \to & A\$ & CA & \to & B\$ & CA & \to & C\$ & \\
 & & & BC & \to & A\$ & BC & \to & B\$ & BC & \to & C\$ & \\
 & & & CB & \to & A\$ & CB & \to & B\$ & CB & \to & C\$ & \\
 & & & \$A & \to & A\$ & \$A & \to & B\$ & \$A & \to & C\$ & \\
 & & & \$B & \to & A\$ & \$B & \to & B\$ & \$B & \to & C\$ & \\
 & & & \$C & \to & A\$ & \$C & \to & B\$ & \$C & \to & C\$ & \\
 & & & \$ & \to & \epsilon & A & \to & \epsilon & B & \to & \epsilon & \\
 & & & C & \to & \epsilon & A & \to & B\$ & & & & \}
\end{array}
$$

since $caca\downarrow_{\mathcal{T}} = CCCACCCA \neq CCCBBBBA = cbba\downarrow_{\mathcal{T}}$ *and*

$$
\begin{array}{llllll}
CCC\underline{A}CCCA & \to_{\mathcal{S}} & CCCB\underline{\$C}CCA & \to_{\mathcal{S}} & CCCBB\underline{\$C}CA & \to_{\mathcal{S}} \\
CCCBBB\underline{\$C}A & \to_{\mathcal{S}} & CCCBBBB\underline{\$}A & \to_{\mathcal{S}} & CCCBBBBA. & \diamond
\end{array}
$$

References

[Avenhaus and Madlener, 1990] Jürgen Avenhaus and Klaus E. Madlener. Term rewriting and equational reasoning. In R.B. Banerji, editor, *Formal Techniques in Artificial Intelligence – A Sourcebook*, pages 1–43. Elsevier Science Publishers B.V. (North-Holland), 1990.

[Bachmair and Dershowitz, 1986] Leo Bachmair and Nachum Dershowitz. Commutation, transformation and termination. In *Proc. 8th CADE (LNCS 230)*, pages 52–60, Oxford (England), 1986.

[Bachmair, 1987] Leo Bachmair. *Proof methods for equational theories.* PhD thesis, University of Illinois, Urbana-Champaign (Illinois), 1987.

[Bellegarde and Lescanne, 1987] Françoise Bellegarde and Pierre Lescanne. Transformation orderings. In *Proc. 12th CAAP (TAPSOFT)*, pages 69–80, 1987. appeared as LNCS.

[Bellegarde and Lescanne, 1988] Françoise Bellegarde and Pierre Lescanne. Termination proofs based on transformation techniques. Technical report, Centre de Recherche en Informatique de Nancy, Nancy (France), March 1988.

[Bellegarde and Lescanne, 1990] Françoise Bellegarde and Pierre Lescanne. Termination by completion. In *Applicable Algebra in Engineering, Communication and Computing (AAECC)*, volume 1, pages 79–96, 1990.

[Boyer and Moore, 1979] Robert S. Boyer and J. Strother Moore. *A computational logic.* Academic Press, 1979.

[Dauchet, 1988] Max Dauchet. Termination of rewriting is undecidable in the one-rule case. In *Proc. 13th Symposium of MFCS*, LNCS 324, pages 262–270, Carlsbad, CSFR, 1988.

[Dershowitz and Jouannaud, 1990] Nachum Dershowitz and Jean-Pierre Jouannaud. *Rewrite systems*, chapter 6, pages 243–320. Handbook of Theoretical Computer Science. Elsevier Science Publisher B.V., 1990.

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

[Eschenbach, 1986] Carola Eschenbach. Die Verwendung von Zeichenkettenordnungen im Zusammenhang mit Semi-Thue Systemen. Bericht Nr. 122, Fachbereich Informatik, Universität Hamburg, Hamburg (Germany), Juni 1986.

[Geser, 1991] Alfons Geser. *Relative Termination.* Ulmer Informatik-Berichte, Fachbereich Informatik, Universität Ulm, Ulm (Germany), 1991.

[Gramlich, 1990] Bernhard Gramlich. Completion based inductive theorem proving – A case study in verifying sorting algorithms. Technical Report SR-90-04, University of Kaiserslautern, Kaiserslautern (Germany), 1990.

[Higman, 1952] Graham Higman. Ordering by divisibility in abstract algebras. In *Proc. London Math. Soc. 2*, pages 326–336, 1952.

[Huet and Lankford, 1978] Gérard Huet and Dallas S. Lankford. On the uniform halting problem for term rewriting systems. Rapport Laboria 283, INRIA, Rocquencourt (France), 1978.

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

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

[Kandzia, 1990] Paul-Thomas Kandzia. Implementation of transformation orderings. Project report, Fachbereich Informatik, Universität Kaiserslautern, 1990. (in German).

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

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

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

[Martin, 1989] Ursula Martin. A note on division orderings on strings. Technical Report CSD-TR-612, University of London, Royal Holloway and Bedford New Colloge, Egham (England), April 1989.

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

[Nessel, 1992] Jochen Nessel. Implementing, investigating and improving a technique for automatically generating transformation orderings. Master's thesis, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern (Germany), 1992. to appear (in German).

[Plaisted, 1986] David A. Plaisted. A simple non-termination test for the Knuth-Bendix method. In *Proc. 8th CADE (LNCS 230)*, pages 79–88, Oxford (England), 1986.

[Steinbach, 1991] Joachim Steinbach. Termination proofs of rewriting systems – Heuristics for generating polynomial orderings. SEKI-Report, University of Kaiserslautern, Kaiserslautern (Germany), 1991.

[Toyama, 1987] Yoshihito Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *IPL*, 25(3):141–143, May 1987.

[Walther, 1988] Christoph Walther. A digest of argument-bounded algorithms. Technical Report 18/88, University of Karlsruhe, Karlsruhe (Germany), 1988.

A Examples

The first part of the appendix contains examples (i.e. TRS) for which the termination cannot be proved with the help of conventional techniques (like the RPOS and the KBO). The TRS dealt with in sections 2 and 3 also appear once again. All examples are presented in a uniform way:

$\mathcal{R}$: $l_1 \to r_1$	$\mathcal{T}$: $l'_1 \to r'_1$
$l_2 \to r_2$	$l'_2 \to r'_2$
$\vdots \quad \vdots$	$\vdots \quad \vdots$
ident	$\mathcal{S}$: $l''_1 \to r''_1$
	$l''_2 \to r''_2$
	$\vdots \quad \vdots$
	Order : $\cdots$

$\mathcal{R}$ contains the rules which have to be oriented w.r.t. the transformation ordering constructed by the TRS $\mathcal{T}$ and $\mathcal{S}$. The basic ordering (Order) used in the transformation ordering is either the RPOS (including τ denoting the status function on $\mathcal{F}$) or the KBO (if a weight function φ is given). If $\mathcal{T}$ and $\mathcal{S}$ (and the ordering) are empty, we *actually do not know* how to prove the termination of $\mathcal{R}$ with transformation orderings. However, we believe that such examples are also interesting. If a transformation ordering is given, then the identifier (ident) indicates whether our algorithm described in section 3 generates the presented $\mathcal{T}$ and $\mathcal{S}$. The identifier can have five different values: (i) S denotes that algorithm 3.1 is successful without any extension. (ii) SC stands for the case that algorithm 3.1 needs an extension of $\mathcal{S}$ in step 4 such that it cooperates with $\mathcal{T}$. (iii) If the algorithm only succeeds with the help of semantics of operators then SS is assigned to the identifier. (iv) Whenever the termination of $\mathcal{R}$ can, according to subsection 3.8, be ensured by proving the termination of $\mathcal{R}_1$ and $\mathcal{R}_2$ (where $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$), the representation of the examples is of the following form: The ident is set to SD, $\mathcal{R}$ is represented by $\mathcal{R}_1$ and $\mathcal{R}_2$, and the orderings for $\mathcal{R}_1$ (a TO) and for $\mathcal{R}_2$ (a classical ordering) are given on the right-hand side of the corresponding TRS. (v) The expression F implies that algorithm 3.1 cannot generate an appropriate transformation ordering.

Example 1 ([Dershowitz, 1987])

$\mathcal{R}$: $f(a) \to f(b)$	$\mathcal{T}$: $g(a) \to q$
$g(b) \to g(a)$	$\mathcal{S}$: $f(a) \to f(b)$
S	$g(b) \to q$
	Order : $a \succ b, g \succ q$

Example 2 (Difference)

$\mathcal{R}$: $0 - y \to 0$	$\mathcal{T}$: $p(s(x)) \to x$
$x - 0 \to x$	$\mathcal{S}$: $0 - y \to 0$
$x - s(y) \to if(x > s(y), s(x - p(s(y))), 0)$	$x - 0 \to x$
$p(0) \to 0$	$x - s(y) \to if(x > s(y), s(x - y), 0)$
$p(s(x)) \to x$	$p(0) \to 0$
S	Order : $- \succ if, - \succ >, - \succ s, - \succ 0$

Example 3 (Intervals of Natural Numbers)

$\mathcal{R}$: $int(0,0) \to 0.nil$
$int(0,s(y)) \to 0.int(s(0),s(y))$
$int(s(x),0) \to nil$
$intlist(nil) \to nil$
$intlist(x.y) \to s(x).intlist(y)$
$int(s(x),s(y)) \to intlist(int(x,y))$
S

$\mathcal{T}$: $int(s(x),s(y)) \to intlist(int(x,y))$
$\mathcal{S}$: $int(0,0) \to 0.nil$
$int(0,s(y)) \to 0.intlist(int(0,y))$
$int(s(x),0) \to nil$
$intlist(nil) \to nil$
$intlist(x.y) \to s(x).intlist(y)$
Order : $int \succ intlist \succ . \succ s \succ nil$

Example 4 (Greatest Common Divisor, [Boyer and Moore, 1979])

$\mathcal{R}$: $gcd(x,0) \to x$
$gcd(0,y) \to y$
$gcd(s(x),s(y)) \to if(x<y, gcd(s(x),y-x), gcd(x-y,s(y)))$
S

$\mathcal{T}$: $gcd(s(x),y-z) \to h_1(x,y,z)$
$gcd(x-y,s(z)) \to h_2(x,y,z)$
$\mathcal{S}$: $gcd(x,0) \to x$
$gcd(0,y) \to y$
$gcd(s(x),s(y)) \to if(x<y, h_1(x,y,x), h_2(x,y,y))$
Order : $gcd \succ h_1, gcd \succ h_2, gcd \succ if, gcd \succ <$

Example 5

$\mathcal{R}$: $p(s(x)) \to x$
$p(0) \to 0$
$f(x,y,z) \to g(x \le y, x, y, z)$
$g(true,x,y,z) \to z$
$g(false,x,y,z) \to f(f(p(x),y,z), f(p(y),z,x), f(p(z),x,y))$
S

$\mathcal{T}$: $g(x \le y, z, u, v) \to h_1(x,y,z,u,v)$
$\mathcal{S}$: $p(s(x)) \to x$
$p(0) \to 0$
$f(x,y,z) \to h_1(x,y,x,y,z)$
$g(true,x,y,z) \to z$
$g(false,x,y,z) \to f(f(p(x),y,z), f(p(y),z,x), f(p(z),x,y))$
Order : $g \succ f \succ h_1, g \succ p$

Example 6 ([Bellegarde and Lescanne, 1988])

$\mathcal{R}$: $f(g(x)) \to f(h(g(x)))$
S

$\mathcal{T}$: $h(g(x)) \to q(x)$
$\mathcal{S}$: $f(g(x)) \to f(q(x))$
Order : $g \succ q$

Example 7 ([Bachmair, 1987])

$\mathcal{R}$: $g(x,y) \to h(x,y)$
$h(f(x),y) \to f(g(x,y))$
S

$\mathcal{T}$: $g(x,y) \to h(x,y)$
$\mathcal{S}$: $h(f(x),y) \to f(h(x,y))$
Order : $g \succ h \succ f$

In this example, $\mathcal{R}$ can be oriented using the KBO based on $\varphi(f) = \varphi(g) = \varphi(h) = 1$ and $g \succ h \succ f$. However, its termination cannot be proved with the help of the RPOS.

Example 8

$\mathcal{R}$: $f(x,x) \to f(a,b)$ $b \to c$	$\mathcal{T}$: $b \to c$ $f(a,c) \to q$
S	$\mathcal{S}$: $f(x,x) \to q$
	Order : $f \succ q, b \succ c$

Example 9

$\mathcal{R}$: $f(a,b) \to f(a,c)$ $f(c,d) \to f(b,d)$	$\mathcal{T}$: $f(b,d) \to q$
S	$\mathcal{S}$: $f(a,b) \to f(a,c)$ $f(c,d) \to q$
	Order : $f \succ q, b \succ c$

Example 10 (Factorial Function)

$\mathcal{R}$: $p(s(x)) \to x$ $f(0) \to s(0)$ $f(s(x)) \to s(x) \star f(p(s(x)))$	$\mathcal{T}$: $p(s(x)) \to x$
S	$\mathcal{S}$: $f(0) \to s(0)$ $f(s(x)) \to s(x) \star f(x)$
	Order : $f \succ \star, f \succ s$

Example 11

$\mathcal{R}$: $f(g(x,y),y) \to f(h(g(x,y)),a)$	$\mathcal{T}$: $h(g(x,y)) \to q(x,y)$
S	$\mathcal{S}$: $f(g(x,y),y) \to f(q(x,y),y)$
	Order : $g \succ q$

Example 12 ([Middeldorp, 1988])

$\mathcal{R}$: $h(x,x,y) \to g(x)$ $g(a) \to h(a,b,a)$ $i(x) \to f(x,x)$ $f(x,y) \to x$	$\mathcal{T}$: $h(a,b,a) \to q$
S	$\mathcal{S}$: $h(x,x,y) \to g(x)$ $g(a) \to q$ $i(x) \to f(x,x)$ $f(x,y) \to x$
	Order : $h \succ g \succ q, i \succ f$

Note that a polynomial ordering can orient $\mathcal{R}$ (see [Steinbach, 1991]).

Example 13 ([Bachmair, 1987])

$\mathcal{R}$: $f(h(x)) \to f(i(x))$ $g(i(x)) \to g(h(x))$ $h(a) \to b$ $i(a) \to b$	$\mathcal{T}$: $g(h(x)) \to q(x)$
SC	$\mathcal{S}$: $f(h(x)) \to f(i(x))$ $h(a) \to b$ $i(a) \to b$ $g(i(x)) \to q(x)$ $q(a) \to g(b)$
	Order : $a \succ g \succ q \succ h \succ i \succ b$

Example 14 (Cartesian Category Combinators, [Bellegarde and Lescanne, 1988])

$\mathcal{R}$: $(x \circ y) \circ z \to x \circ (y \circ z)$ $f(x,y) \circ z \to f(x \circ z, y \circ z)$ $x \circ a \to x$ $a \circ x \to x$ $F \circ f(x,y) \to x$ $S \circ f(x,y) \to y$ $b \circ f(b \circ f(x,y),z) \to b \circ f(x, b \circ f(y,z))$	$\mathcal{T}$: $b \circ f(x,y) \to h(x,y)$
SC	$\mathcal{S}$: $(x \circ y) \circ z \to x \circ (y \circ z)$ $f(x,y) \circ z \to f(x \circ z, y \circ z)$ $x \circ a \to x$ $a \circ x \to x$ $F \circ f(x,y) \to x$ $S \circ f(x,y) \to y$ $h(h(x,y),z) \to h(x,h(y,z))$ $h(x,y) \circ z \to b \circ (f(x,y) \circ z)$
	Order : $\circ \succ f, \circ \succ h, \tau(\circ) = \tau(h) = \text{left}$

Example 15 (Factorial Function, [Kamin and Lévy, 1980])

$\mathcal{R}$: $p(s(s(x))) \to s(p(s(x)))$ $p(s(0)) \to 0$ $fac(s(x)) \to fac(p(s(x))) \star s(x)$	$\mathcal{T}$: $p(s(x)) \to x$ $\mathcal{S}$: $fac(s(x)) \to fac(x) \star s(x)$ Order : $fac \succ \star$
SS	

Example 16 (Addition)

$\mathcal{R}$: $0 + y \to y$ $s(x) + 0 \to s(x)$ $s(x) + s(y) \to s(s(x) + (y + 0))$	$\mathcal{T}$: $x + 0 \to x$ $\mathcal{S}$: $0 + y \to y$ $s(x) + s(y) \to s(s(x) + y)$ Order : $+ \succ s$
SS	

Example 17 (Associativity and Endomorphism, [Bellegarde and Lescanne, 1988])

$\mathcal{R}_2$: $(x + y) + z \to x + (y + z)$ $f(x) + f(y) \to f(x + y)$ $\mathcal{R}_1$: $f(x) + (f(y) + z) \to f(x + y) + z$	Order : $+ \succ f, \tau(+) = left$ $\mathcal{T}$: $x + (y + z) \to (x + y) + z$ $f(x + y) \to f(x) + f(y)$ Order : $f \succ +, \tau(+) = right$
SD	

$\mathcal{R}$: $(x + y) + z \to x + (y + z)$ $f(x) + f(y) \to f(x + y)$ $f(x) + (f(y) + z) \to f(x + y) + z$	$\mathcal{T}$: $f(x) + y \to f(x + y)$ $x + f(y) \to f(x + y)$ $(x + y) + z \to x + (y + z)$ $\mathcal{S}$: $f(f(x)) \to f(x)$ Order : $+ \succ f, \tau(+) = \text{left}$
F	

Example 18 (Binary Trees)

$\mathcal{R}_2$: $f(nil) \to nil$ $f(nil.x) \to nil.f(x)$ $f((x.y).z) \to f(x.(y.z))$ $\mathcal{R}_1$: $g(nil) \to nil$ $g(x.nil) \to g(x).nil$ $g(x.(y.z)) \to g((x.y).z)$	Order : $f \succ ., \tau(.) = left$ $\mathcal{T}$: $f(nil) \to nil$ $g(nil) \to nil$ $g(x.(y.z)) \to g((x.y).z)$ $f(x.(y.z)) \to f((x.y).z)$ $nil.x \to x$ $x.nil \to x$ Order : $g \succ ., \tau(.) = right$
SD	

Example 19

$\mathcal{R}_2$: $g(s(x)) \to f(x)$ $\mathcal{R}_1$: $f(0) \to s(0)$ $f(s(x)) \to s(s(g(x)))$ $g(0) \to 0$	Order : $g \succ f$ $\mathcal{T}$: $f(x) \to g(s(x))$ $\mathcal{S}$: $g(s(0)) \to s(0)$ $g(s(s(x))) \to s(s(g(x)))$ $g(0) \to 0$ Order : $f \succ g \succ s$
SD	

Example 20 (Walther)

$\mathcal{R}_2$: $f(0) \to s(0)$ $f(s(0)) \to s(0)$
$\mathcal{R}_1$: $f(s(s(x))) \to f(f(s(x)))$
SD

Order : $f \succ s$
$\mathcal{T}$: $f(x) \to x$ $s(s(x)) \to s(x)$
Order : $s \succ f$

$\mathcal{R}$: $f(0) \to s(0)$ $f(s(0)) \to s(0)$ $f(s(s(x))) \to f(f(s(x)))$
F

$\mathcal{T}$: $f(s(x)) \to s(x)$
$\mathcal{S}$: $f(0) \to s(0)$ $s(s(x)) \to s(x)$
Order : $f \succ s$

Example 21 ([Dershowitz, 1987])

$\mathcal{R}$: $f(f(x)) \to f(g(f(x)))$
F

$\mathcal{T}$: $f(g(f(x))) \to f(x)$
$\mathcal{S}$: $f(f(x)) \to f(x)$
Order : empty $\succ$

See also subsection 3.7.

Example 22 (Binary Numbers, Geser (see [Bellegarde and Lescanne, 1990]))

$\mathcal{R}$: $div2(\emptyset) \to \emptyset$ $div2(s(\emptyset)) \to \emptyset$ $div2(s(s(x))) \to s(div2(x))$ $lastbit(\emptyset) \to 0$ $lastbit(s(\emptyset)) \to 1$ $lastbit(s(s(x))) \to lastbit(x)$ $conv(\emptyset) \to \varepsilon \ \& \ 0$ $conv(s(x)) \to conv(div2(s(x))) \ \& \ lastbit(s(x))$
F

$\mathcal{T}$: $conv(\emptyset) \to \varepsilon \ \& \ 0$ $conv(div2(x)) \to p(x)$
$\mathcal{S}$: $div2(\emptyset) \to \emptyset$ $div2(s(\emptyset)) \to \emptyset$ $div2(s(s(x))) \to s(div2(x))$ $lastbit(\emptyset) \to 0$ $lastbit(s(\emptyset)) \to 1$ $lastbit(s(s(x))) \to lastbit(x)$ $conv(s(x)) \to p(s(x)) \ \& \ lastbit(s(x))$ $p(\emptyset) \to \varepsilon \ \& \ 0$ $p(s(\emptyset)) \to \varepsilon \ \& \ 0$ $p(s(s(x))) \to conv(s(div2(x)))$
Order : $s \succ conv \succ p \succ \& \succ \varepsilon \succ div2 \succ lastbit \succ \emptyset \succ 1 \succ 0$

Example 23 ([Plaisted, 1986])

$\mathcal{R}$: $f(c) \to g(h(c))$ $h(g(x)) \to g(h(f(x)))$ $k(x, h(x), c) \to h(x)$ $k(f(x), y, x) \to f(x)$
F

$\mathcal{T}$: $g(h(c)) \to d$ $g(h(f(x))) \to q(x)$ $h(g(x)) \to q(x)$ $g(h(d)) \to q(d)$ $q(h(f(x))) \to h(q(x))$ $q(h(c)) \to d$
$\mathcal{S}$: $k(d, x, c) \to d$ $k(c, x, c) \to c$ $k(x, h(x), c) \to h(x)$ $k(f(x), y, x) \to f(x)$ $f(c) \to d$ $k(g(c), q(d), c) \to q(d)$
Order : $g \succ q \succ h \succ f \succ d$

Example 24 (Reverse)

$$\begin{array}{l} \mathcal{R}\colon\quad rev(nil) \to nil \\ \quad rev(x \circ y) \to rev_1(x,y) \circ rev_2(x,y) \\ \quad rev_1(x,nil) \to x \\ rev_1(x, y \circ z) \to rev_1(y,z) \\ \quad rev_2(x,nil) \to nil \\ rev_2(x, y \circ z) \to rev(x \circ rev(rev_2(y,z))) \end{array}$$

F

Example 25 (Purging, [Walther, 1988])

$$\begin{array}{l} \mathcal{R}\colon\ purge(nil) \to nil \\ \quad purge(x.l) \to x.purge(remove(x,l)) \\ remove(x,nil) \to nil \\ remove(x,y.l) \to if(x \equiv y, remove(x,l), y.remove(x,l)) \end{array}$$

F

Example 26

$$\begin{array}{l} \mathcal{R}\colon\ f(a,h(y)) \to h(h(y)) \\ \quad f(h(x),a) \to f(x,h(a)) \\ f(h(x),h(y)) \to g(h(y),x,h(y)) \\ g(h(x),y,z) \to f(y,g(x,y,z)) \\ \quad g(a,y,z) \to z \end{array}$$

F

Example 27 ([Toyama, 1987])

$$\mathcal{R}\colon\ f(0,1,x) \to f(x,x,x)$$

F

Example 28

$$\begin{array}{l} \mathcal{R}\colon\ x/x \to 1 \\ \quad x/1 \to x \\ i(x/y) \to y/x \\ (x/y)/z \to x/(z/i(y)) \end{array}$$

F

The KBO (based on $\varphi(i) = 0$, $i \succ /$ and $\tau(/) = left$) is able to prove the termination of $\mathcal{R}$ whereas the RPOS is not.

Example 29 (Quick-Sort, [Gramlich, 1990])

$$\begin{array}{l} \mathcal{R}\colon\quad qsort(nil) \to nil \\ \quad qsort(x.y) \to qsort(lowers(x,y)) \circ (x.qsort(greaters(x,y))) \\ lowers(x,nil) \to nil \\ lowers(x,y.z) \to if(y \leq x, y.lowers(x,z), lowers(x,z)) \\ greaters(x,nil) \to nil \\ greaters(x,y.z) \to if(y \leq x, greaters(x,z), y.greaters(x,z)) \end{array}$$

F

Example 30 (Minimum-Sort, [Gramlich, 1990])

$\mathcal{R}$: $minsort(nil) \to nil$
$minsort(x.y) \to min(x,y).minsort(del(min(x,y),x.y))$
$min(x,nil) \to x$
$min(x,y.z) \to if(x \leq y, min(x,z), min(y,z))$
$del(x,nil) \to nil$
$del(x,y.z) \to if(x = y, z, y.del(x,z))$

F

Example 31 (Bubble-Sort, [Gramlich, 1990])

$\mathcal{R}$: $bsort(nil) \to nil$
$bsort(x.y) \to last(bubble(x.y)).bsort(butlast(bubble(x.y)))$
$bubble(nil) \to nil$
$bubble(x.nil) \to x.nil$
$bubble(x.(y.z)) \to if(x \leq y, y.bubble(x.z), x.bubble(y.z))$
$last(nil) \to 0$
$last(x.nil) \to x$
$last(x.(y.z)) \to last(y.z)$
$butlast(nil) \to nil$
$butlast(x.nil) \to nil$
$butlast(x.(y.z)) \to x.butlast(y.z)$

F

B Proofs

This appendix consists of the proofs of the theorems and the lemmata contained in the sections 3 and 4. We use the following abbreviations: $\leadsto$ for therefore and $\because$ for because.

Theorem B.1 *Algorithm 3.1 always terminates. If it does not fail, $\mathcal{R}$ is proved to be terminating with the help of the* TO *defined by $\mathcal{S}$ and $\mathcal{T}$.*
Proof: *The termination of the algorithm is obviously satisfied since $\mathcal{R}$ is finite. The correctness is guaranteed because the required properties of the induced* TO *are checked at the end of the algorithm (in step 4).* $\diamond$

Lemma B.1 (Simulation of the SO) *Let Σ be any alphabet,*

$$\begin{array}{lcll} \mathcal{T} & = & \emptyset & \text{and} \\ \mathcal{S} & = & \{a \to \epsilon \mid a \in \Sigma\}. & \end{array}$$

Then, $u \succ_{\mathsf{SO}} v \iff u \succ_{\mathsf{TO}} v$.
Proof: *It is obvious that $\mathcal{T}$ is confluent, $\mathcal{S} \cup \mathcal{T}$ is well-founded and $\mathcal{S}$ cooperates with $\mathcal{T}$.*

'$\Rightarrow$' *Let $a_1 \ldots a_m \succ_{\mathsf{SO}} b_1 \ldots b_n$.*
$\leadsto$ $a_1 \ldots a_m \neq b_1 \ldots b_n \wedge (\exists 1 \leq i_1 < i_2 < \cdots < i_n \leq m)(\forall j \in [1,n])\, a_{i_j} = b_j$
$\because$ *definition of the* SO
$\leadsto$ $a_1 \ldots a_m = u_1 b_1 u_2 b_2 \ldots u_n b_n u_{n+1}$ *such that* $u_i \in \Sigma^* \wedge u_k \neq \epsilon$, *for one k at least*
$\leadsto$ $u_1 b_1 u_2 b_2 \ldots u_n b_n u_{n+1}\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} b_1 \ldots b_n\downarrow_{\mathcal{T}}$
$\because$ *definition of $\mathcal{S}$ (and since $\mathcal{T} = \emptyset$)*
$\leadsto$ $a_1 \ldots a_m \succ_{\mathsf{TO}} b_1 \ldots b_n$
$\because$ *definition of the* TO *(since $a_1 \ldots a_m\downarrow_{\mathcal{T}} \neq b_1 \ldots b_n\downarrow_{\mathcal{T}}$)*

'$\Leftarrow$' *Let $a_1 \ldots a_m \succ_{\mathsf{TO}} b_1 \ldots b_n$.*
$\leadsto$ • $a_1 \ldots a_m\downarrow_{\mathcal{T}} \neq b_1 \ldots b_n\downarrow_{\mathcal{T}}$
$\because$ $\mathcal{T} = \emptyset$
• $a_1 \ldots a_m\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} b_1 \ldots b_n\downarrow_{\mathcal{T}}$
$\because$ *definition of the* TO *(since $\mathcal{T} = \emptyset$)*
$\leadsto$ $a_1 \ldots a_m = u_1 b_1 u_2 b_2 \ldots u_n b_n u_{n+1}$ *such that* $u_i \in \Sigma^* \wedge u_k \neq \epsilon$, *for one k at least*
$\leadsto$ $a_1 \ldots a_m \succ_{\mathsf{SO}} b_1 \ldots b_n$
$\because$ *definition of the* SO $\diamond$

Lemma B.2 (Simulation of the ESO) *Let $\Sigma = \{a_i \mid i \in I\}$ be any alphabet and $\gtrsim$ be any quasi-precedence. Furthermore, let $\Sigma' = \{A_i \mid i \in I\}$ such that $\Sigma \cap \Sigma' = \emptyset$ and*

$$\begin{array}{lcl} \mathcal{T} & = & \{a_i \to A_k \mid k = min\{j \mid a_i \sim a_j\}\} \quad \text{and} \\ \mathcal{S} & = & \{A_m \to A_n \mid a_i > a_j, m = min\{k \mid a_k \sim a_i\}, n = min\{k \mid a_k \sim a_j\}\} \\ & & \cup \{A_i \to \epsilon \mid i \in I\}. \end{array}$$

Then, $u \succ_{\mathsf{ESO}} v \iff u \succ_{\mathsf{TO}} v$.
Proof: *Note that*

- *$\mathcal{T}$ is confluent because there exist no critical pairs.*
- *$\mathcal{S} \cup \mathcal{T}$ is well-founded: In order to prove the termination we choose the* RPOS *based on the precedence $(\forall i \in I)\, a_i \succ A_i \wedge (\forall i,j \in I)\, a_i \succ a_j \vee (a_i \sim a_j \wedge i > j) \hookrightarrow A_i \succ A_j$.*

- $\mathcal{S}$ *cooperates with* $\mathcal{T}$ *since there are no overlappings between* $\mathcal{S}$ *and* $\mathcal{T}$.

'$\Rightarrow$' *Let* $b_1 \ldots b_m \succ_{\mathsf{ESO}} c_1 \ldots c_n$.

$\rightsquigarrow$ $b_1 \ldots b_m \not\approx c_1 \ldots c_n \wedge (\exists 1 \leq i_1 < i_2 < \cdots < i_n \leq m)(\forall j \in [1,n])\, b_{i_j} \succsim c_j$

$\because$ *definition of the* ESO

$\rightsquigarrow$ • $b_1 \ldots b_m\downarrow_{\mathcal{T}} \neq c_1 \ldots c_n\downarrow_{\mathcal{T}}$

$\because$ *definition of* $\approx$ *and* $\mathcal{T}$

• $b_1 \ldots b_m = u_1 d_1 u_2 d_2 \ldots u_n d_n u_{n+1}$ *such that* $u_i \in \Sigma^* \wedge (\forall i \in [1,n])\, d_i \succsim c_i$

$\because$ *definition of the* ESO

$\rightsquigarrow$ $u_1 d_1 u_2 d_2 \ldots u_n d_n u_{n+1}\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} c_1 c_2 \ldots c_n\downarrow_{\mathcal{T}}$

$\because$ $d_i\downarrow_{\mathcal{T}} = c_i\downarrow_{\mathcal{T}}$ *if* $c_i \sim d_i$ *and by definition of* $\mathcal{S}$

$\rightsquigarrow$ $b_1 \ldots b_m \succ_{\mathsf{TO}} c_1 \ldots c_n$

$\because$ *definition of the* TO

'$\Leftarrow$' *Let* $b_1 \ldots b_m \succ_{\mathsf{TO}} c_1 \ldots c_n$.

$\rightsquigarrow$ $b_1 \ldots b_m\downarrow_{\mathcal{T}} \neq c_1 \ldots c_n\downarrow_{\mathcal{T}}$:

Assume that $b_1 \ldots b_m\downarrow_{\mathcal{T}} = c_1 \ldots c_n\downarrow_{\mathcal{T}}$.

$\rightsquigarrow$ $b_1 \ldots b_m \xrightarrow{+}_{\mathcal{T}} c_1 \ldots c_n$

$\because$ $b_1 \ldots b_m \succ_{\mathsf{TO}} c_1 \ldots c_n$ *and by definition of the* TO

$\rightsquigarrow$ *contradiction*

$\because$ *each right-hand side of* $\mathcal{T}$ *contains only letters from* Σ'

$\rightsquigarrow$ $b_1 \ldots b_m\downarrow_{\mathcal{T}} \neq c_1 \ldots c_n\downarrow_{\mathcal{T}}$

$\rightsquigarrow$ $b_1 \ldots b_m\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}} c_1 \ldots c_n\downarrow_{\mathcal{T}}$

$\because$ *definition of the* TO

Let $b_1 \ldots b_m\downarrow_{\mathcal{T}} = \overline{B}_1 \ldots \overline{B}_m$ *and* $c_1 \ldots c_n\downarrow_{\mathcal{T}} = \overline{C}_1 \ldots \overline{C}_n$ *where* $b_i\downarrow_{\mathcal{T}} = \overline{B}_i$ *and* $c_i\downarrow_{\mathcal{T}} = \overline{C}_i$.

$\rightsquigarrow$ $\overline{B}_1 \ldots \overline{B}_m \xrightarrow{+}_{\mathcal{S}} \overline{C}_1 \ldots \overline{C}_n$

$\because$ *each right-hand side of* $\mathcal{S}$ *contains only letters from* Σ'

$\rightsquigarrow$ $\overline{B}_1 \ldots \overline{B}_m = u_1 \overline{D}_1 u_2 \overline{D}_2 \ldots u_n \overline{D}_n u_{n+1}$ *such that* $\overline{D}_i \succ \overline{C}_i \vee \overline{D}_i = \overline{C}_i$

$\because$ *definition of* $\mathcal{S}$

$\rightsquigarrow$ $b_1 \ldots b_m \succ_{\mathsf{ESO}} c_1 \ldots c_n$

$\because$ $\overline{D}_i \succ \overline{C}_i \hookrightarrow d_i \succ c_i$ *and* $\overline{D}_i = \overline{C}_i \hookrightarrow d_i \sim c_i$ $\diamond$

Lemma B.3 (Simulation of the LEN**)** *Let* Σ *be any alphabet and* A *a letter such that* $A \notin \Sigma$. *Furthermore, let*

$$\begin{array}{lcl} \mathcal{T} & = & \{a \to A \mid a \in \Sigma\} \quad \textit{and} \\ \mathcal{S} & = & \{A \to \epsilon\}. \end{array}$$

Then, $u \succ_{\mathsf{LEN}} v \iff u \succ_{\mathsf{TO}} v$.

Proof: *First of all, according to the definition of the transformation ordering, we have to show that*

- $\mathcal{T}$ *is confluent: There are no critical pairs.*
- $\mathcal{S} \cup \mathcal{T}$ *is noetherian: Choose an* RPOS *based on* $a \succ A$, *for all* $a \in \Sigma$.
- $\mathcal{S}$ *cooperates with* $\mathcal{T}$*: There exist no critical pairs.*

'$\Rightarrow$' *Let* $u \succ_{\mathsf{LEN}} v$.

$\leadsto$ $|u| > |v|$

$\because$ *definition of* $|.|$

$\leadsto$ • $|u\downarrow_{\mathcal{T}}| = |A^{|u|}| \neq |A^{|v|}| = |v\downarrow_{\mathcal{T}}|$

$\because$ *definition of* $\mathcal{T}$

• $u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v\downarrow_{\mathcal{T}}$

$\because$ $|u| > |v|$

$\leadsto$ $u \succ_{\mathsf{TO}} v$

$\because$ *definition of the* TO

'$\Leftarrow$' *Let* $u \succ_{\mathsf{TO}} v$*:*

$\leadsto$ $u \not\xrightarrow{+}_{\mathcal{T}} v$

$\because$ *the structure of* $\mathcal{T}$ *(otherwise* v *must contain at least one* A *which is not in* Σ*)*

$\leadsto$ $u\downarrow_{\mathcal{T}} \neq v\downarrow_{\mathcal{T}} \wedge u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}} v\downarrow_{\mathcal{T}}$

$\leadsto$ $u\downarrow_{\mathcal{T}} = A^{|u|}$ *and* $v\downarrow_{\mathcal{T}} = A^{|v|}$ *with* $|u| \neq |v|$

$\because$ *definition of* $\mathcal{T}$

$\leadsto$ $u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v\downarrow_{\mathcal{T}}$

$\because$ *definition of* $\mathcal{S}$

$\leadsto$ $|u| > |v|$

$\because$ $|u| = |u\downarrow_{\mathcal{T}}|$ *and* $|v| = |v\downarrow_{\mathcal{T}}|$

$\leadsto$ $u \succ_{\mathsf{LEN}} v$

$\because$ *definition of* $|.|$ $\diamond$

Lemma B.4 (Simulation of the LLEX**)** *Let* Σ *be any alphabet,* A *and* B *are letters such that* $A, B \notin \Sigma$ *and* $\succ$ *is a precedence. Furthermore, let*

$$\begin{array}{rcll} \mathcal{T} & = & \{a \to bA \mid a \succ b\} & \cup \\ & & \{Aa \to bA \mid a, b \in \Sigma\} & \cup \\ & & \{A \to \epsilon\} & \cup \\ & & \{a \to B \mid a \in \Sigma\} & \textit{and} \\ \mathcal{S} & = & \{B \to \epsilon\}. & \end{array}$$

Then, $u \succ_{\mathsf{LLEX}} v \iff u \succ_{\mathsf{TO}} v$.

Proof: *First of all, we have to show that* $\mathcal{T}$ *is confluent,* $\mathcal{S} \cup \mathcal{T}$ *is well-founded and* $\mathcal{S}$ *cooperates with* $\mathcal{T}$.

1. $\mathcal{T}$ *is confluent: The following critical pairs are all possible divergencies.*

$$\begin{array}{lcl} bA \leftarrow a \to cA \;\; \textit{if} \;\; a \succ b, a \succ c & \hookrightarrow & bA \to b \to B \leftarrow c \leftarrow cA \\ AbA \leftarrow Aa \to cA \;\; \textit{if} \;\; a \succ b & \hookrightarrow & AbA \to bA \to b \to B \leftarrow c \leftarrow cA \\ bA \leftarrow a \to B \;\; \textit{if} \;\; a \succ b & \hookrightarrow & bA \to b \to B \\ bA \leftarrow Aa \to cA & \hookrightarrow & bA \to b \to B \leftarrow c \leftarrow cA \\ bA \leftarrow Aa \to a & \hookrightarrow & bA \to b \to B \leftarrow a \\ bA \leftarrow Aa \to AB & \hookrightarrow & bA \to b \to B \leftarrow AB \end{array}$$

The unique normal form of a word u *is* $u\downarrow_{\mathcal{T}} = B^{|u|}$*. This is valid since each letter of* Σ *can be transformed into* B *(by using the set* $\{a \to B \mid a \in \Sigma\}$*). Furthermore, each additional* A *can be removed by applying the rule* $A \to \epsilon$.

2. *$\mathcal{S} \cup \mathcal{T}$ is well-founded: In order to prove the termination of $\mathcal{S} \cup \mathcal{T}$ we use the* KBO *based on the following parameters:*

   - $(\forall a \in \Sigma)\, \varphi(a) = 1$ *and* $\varphi(A) = 0, \varphi(B) = 1$
   - *The precedence is identical to that of the* LLEX *extended by* $A \succ a \succ B$, $\forall a \in \Sigma$.

3. *$\mathcal{S}$ cooperates with $\mathcal{T}$: This assertion is true because there are no critical pairs (there is no left-hand side of $\mathcal{T}$ containing B).*

'$\Rightarrow$'

- *Let* $|u| > |v|$.
  $\rightarrowtail$ $u\downarrow_{\mathcal{T}} \neq v\downarrow_{\mathcal{T}} \wedge\ u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v\downarrow_{\mathcal{T}}$
  $\because$ $w\downarrow_{\mathcal{T}} = B^{|w|}$, $|u| > |v|$ *and* $B \rightarrow_{\mathcal{S}} \epsilon$
  $\rightarrowtail$ $u \succ_{\mathsf{TO}} v$
  $\because$ *definition of the* TO
- *Let* $|u| = |v| \wedge\ u \succ^{lex} v$.
  $\rightarrowtail$ $u\downarrow_{\mathcal{T}} = v\downarrow_{\mathcal{T}}$
  $\because$ $w\downarrow_{\mathcal{T}} = B^{|w|}$
  $\rightarrowtail$ *We have to prove that* $u \xrightarrow{+}_{\mathcal{T}} v$.
  Let $u = a^m a_1 \ldots a_n, v = a^m b_1 \ldots b_n$ *such that* $a_1 \succ b_1$ *(since* $u \succ^{lex} v$*).*
  $\rightarrowtail$ *We must show that* $a_1 \ldots a_n \xrightarrow{+}_{\mathcal{T}} b_1 \ldots b_n$ *with* $a_1 \succ b_1$.
  Since $a_1 \succ b_1$, *there exists a rule* $a_1 \rightarrow b_1 A$ *in* $\mathcal{T}$. *By applying this rule we get* $a_1 a_2 \ldots a_n \rightarrow_{\mathcal{T}} b_1 A a_2 \ldots a_n$.
  $\rightarrowtail$ *We have to show that* $A a_2 \ldots a_n \xrightarrow{*}_{\mathcal{T}} b_2 \ldots b_n$.
  This is possible by applying rules of the second class ($Aa \rightarrow bA, \forall a, b \in \Sigma$) n *times and by removing* A *using the rule* $A \rightarrow \epsilon$.

'$\Leftarrow$'

- *Let* $u\downarrow_{\mathcal{T}} \neq v\downarrow_{\mathcal{T}} \wedge\ u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}} v\downarrow_{\mathcal{T}}$
  $\rightarrowtail$ $u\downarrow_{\mathcal{T}} = B^{|u|}$, $v\downarrow_{\mathcal{T}} = B^{|v|}$ *such that* $|u| > |v|$
  $\because$ $w\downarrow_{\mathcal{T}} = B^{|w|}$ *and by definition of* $\mathcal{S}$ *and* $\mathcal{T}$
  $\rightarrowtail$ $u \succ_{\mathsf{LEN}} v$
  $\because$ *definition of* $|.|$
  $\rightarrowtail$ $u \succ_{\mathsf{LLEX}} v$
  $\because$ *definition of the* LLEX
- *Let* $u\downarrow_{\mathcal{T}} = v\downarrow_{\mathcal{T}} \wedge\ u \xrightarrow{+}_{\mathcal{T}} v$.
  $\rightarrowtail$ $|u| = |v|$
  $\because$ $|w\downarrow_{\mathcal{T}}| = |w|$
  $\rightarrowtail$ *We have to show that* $u \succ^{lex} v$ *(according to the definition of the* LLEX*).*
  Let $u = a_1 \ldots a_n$ *and* $v = b_1 \ldots b_n$ *such that* $a_i, b_i \in \Sigma$ *and* $u \xrightarrow{+}_{\mathcal{T}} v$.
  $\rightarrowtail$ *We must prove that* $\mathcal{T}$ *cannot transform* u *into a* v *such that*
  $b_1 = a_1, \ldots, b_i = a_i, b_{i+1} \succ a_{i+1}$ *or* $b_{i+1} \sharp a_{i+1}$.
  However, this is not possible since
  · *we have to apply a rule of the set* $\{a \rightarrow bA \mid a \succ b\}$, *first*
  · *we cannot shift an* A *to the left*
  $\rightarrowtail$ $u = a_1 \ldots a_n \succ^{lex} b_1 \ldots b_n = v$
  $\rightarrowtail$ $u \succ_{\mathsf{LLEX}} v$ $\diamond$

Lemma B.5 (Simulation of the SKBO) *Let Σ be any alphabet, A a letter such that $A \notin \Sigma$ and φ be a weight function. Furthermore, let*

$$\begin{array}{lcl} \mathcal{T} & = & \{a \to A^{\varphi(a)} \mid a \in \Sigma\} \quad \textit{and} \\ \mathcal{S} & = & \{A \to \epsilon\}. \end{array}$$

Then, $u \succ_{\mathsf{SKBO}} v \iff u \succ_{\mathsf{TO}} v$.

Proof: *It is obvious that $\mathcal{T}$ is confluent, $\mathcal{S} \cup \mathcal{T}$ is noetherian and that $\mathcal{S}$ cooperates with $\mathcal{T}$ (see the proof of lemma 4.3).*

'$\Rightarrow$' *Let $u \succ_{\mathsf{SKBO}} v$.*

$\rightsquigarrow$ $\varphi(u) > \varphi(v)$

$\because$ *definition of the* SKBO

$\rightsquigarrow$ $u|_{\mathcal{T}} = A^{\varphi(u)}, v|_{\mathcal{T}} = A^{\varphi(v)}$

$\because$ *definition of $\mathcal{T}$*

$\rightsquigarrow$ • $u|_{\mathcal{T}} \neq v|_{\mathcal{T}}$

$\because$ $\varphi(u) > \varphi(v)$

• $u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v|_{\mathcal{T}}$

$\because$ *definition of $\mathcal{S}$ and because $\varphi(u) > \varphi(v)$*

$\rightsquigarrow$ $u \succ_{\mathsf{TO}} v$

$\because$ *definition of the* TO

'$\Leftarrow$' *Let $u \succ_{\mathsf{TO}} v$.*

$\rightsquigarrow$ $u \not\xrightarrow{+}_{\mathcal{T}} v$

$\because$ *the structure of $\mathcal{T}$, otherwise v must contain at least one A which is not in Σ*

$\rightsquigarrow$ $u|_{\mathcal{T}} \neq v|_{\mathcal{T}} \wedge u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}} v|_{\mathcal{T}}$

$\rightsquigarrow$ $u|_{\mathcal{T}} = A^{\varphi(u)}$ *and* $v|_{\mathcal{T}} = A^{\varphi(v)}$ *with* $\varphi(u) \neq \varphi(v)$

$\because$ *definition of $\mathcal{T}$*

$\rightsquigarrow$ $u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v|_{\mathcal{T}}$

$\because$ *definition of $\mathcal{S}$*

$\rightsquigarrow$ $\varphi(u) > \varphi(v)$

$\because$ $\varphi(u) = |u|_{\mathcal{T}}|$ *and* $\varphi(v) = |v|_{\mathcal{T}}|$

$\rightsquigarrow$ $u \succ_{\mathsf{SKBO}} v$

$\because$ *definition of the* SKBO $\diamond$

Lemma B.6 *Let $\succ$ be a partial precedence and φ be a weight function on an alphabet Σ. Then,*

$$a_1 \ldots a_m \succ^{lex} b_1 \ldots b_n \iff a_1^{\varphi'(a_1)} \ldots a_m^{\varphi'(a_m)} \succ^{lex} b_1^{\varphi'(b_1)} \ldots b_n^{\varphi'(b_n)}$$

where $\varphi'(a) = \begin{cases} \varphi(a) & \textit{if } \varphi(a) > 0 \\ 1 & \textit{otherwise} \end{cases}$

Proof:

'$\Rightarrow$' *Let $a_1 \ldots a_m \succ^{lex} b_1 \ldots b_n$.*

$\rightsquigarrow$ $(\exists i)(\forall j < i)\, a_j = b_j \wedge a_i \succ b_i$

$\rightsquigarrow$ $a_1^{\varphi'(a_1)} \ldots a_{i-1}^{\varphi'(a_{i-1})} = b_1^{\varphi'(b_1)} \ldots b_{i-1}^{\varphi'(b_{i-1})}$

$\because \quad (\forall j < i)\, a_j^{\varphi'(a_j)} = b_j^{\varphi'(b_j)}$ *(because $\succ$ is a partial precedence) and a letter a with weight zero does not disappear*

$\leadsto\ a_1^{\varphi'(a_1)} \ldots a_{i-1}^{\varphi'(a_{i-1})} a_i^{\varphi'(a_i)} \ldots a_m^{\varphi'(a_m)} \succ^{lex} b_1^{\varphi'(b_1)} \ldots b_{i-1}^{\varphi'(b_{i-1})} b_i^{\varphi'(b_i)} \ldots b_n^{\varphi'(b_n)}$

$\because \quad a_i \succ b_i$

'$\Leftarrow$' *analogous to the $\Rightarrow$-case* $\diamond$

Lemma B.7 (Simulation of the KBO Based on Positive Weights) *Let $\Sigma = \{a_1, \ldots, a_n\}$ be any alphabet, $\Sigma' = \{A_1, \ldots, A_n\}$ with $\Sigma \cap \Sigma' = \emptyset$ and $\$ \notin \Sigma \cup \Sigma'$. Furthermore, let φ be a weight function such that $\varphi_\Sigma(a_i) > 0$ and $\succ$ be a partial precedence. Let*

$$\begin{array}{rcll} \mathcal{T} & = & \{a_i \to A_i^{\varphi_\Sigma(a_i)} \mid i = 1, \ldots, n\} & \text{and} \\ \mathcal{S} & = & \{A_i A_j \to A_k \$ \mid i, j, k = 1, \ldots, n\} & \cup \\ & & \{\$ A_i \to A_j \$ \mid i, j = 1, \ldots, n\} & \cup \\ & & \{\$ \to \epsilon\} & \cup \\ & & \{A_i \to \epsilon \mid i = 1, \ldots, n\} & \cup \\ & & \{A_i \to A_j \$ \mid a_i \succ a_j\}. & \end{array}$$

Then, $u \succ_{\mathsf{KBO}} v \iff u \succ_{\mathsf{TO}} v$.

Proof: *First of all, we have to show that $\mathcal{T}$ is confluent, $\mathcal{S} \cup \mathcal{T}$ is well-founded and $\mathcal{S}$ cooperates with $\mathcal{T}$.*

1. *$\mathcal{T}$ is confluent since there are no overlappings.*

2. *$\mathcal{S} \cup \mathcal{T}$ is well-founded: In order to prove the termination of $\mathcal{S} \cup \mathcal{T}$ we use the* KBO *based on φ' and $\succ'$:*

$$\begin{array}{l|l} (\forall i)\, \varphi'(A_i) = 2 & (\forall i)\, \$ \succ' A_i \\ (\forall i)\, \varphi'(a_i) = 2\varphi(a_i) + 1 & A_i \succ' A_j \text{ if } a_i \succ a_j \\ \varphi'(\$) = 0 & \end{array}$$

3. *$\mathcal{S}$ cooperates with $\mathcal{T}$: This is valid because there is no left-hand side of $\mathcal{T}$ containing A_i or $\$$.*

'$\Rightarrow$' *Let $u\downarrow_{\mathcal{T}} = A_1 \ldots A_m$, $v\downarrow_{\mathcal{T}} = B_1 \ldots B_k$.*

- *Let $\varphi(u) > \varphi(v)$.*
  $\leadsto\ u\downarrow_{\mathcal{T}} \neq v\downarrow_{\mathcal{T}}$
  $\because \quad m > k$ *(because $\varphi(u) > \varphi(v)$)*
  $\leadsto$ *We have to show that $u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v\downarrow_{\mathcal{T}}$:*
  $\underline{A_1 A_2} A_3 \ldots A_m \to_{\mathcal{S}} B_1 \underline{\$ A_3} \ldots A_m \to_{\mathcal{S}} B_1 B_2 \$ A_4 \ldots A_m \to_{\mathcal{S}} \cdots \to_{\mathcal{S}}$
  $B_1 B_2 \ldots B_k \underline{\$} A_{k+2} \ldots A_m \to_{\mathcal{S}} B_1 B_2 \ldots B_k A_{k+2} \ldots A_m$
  where $A_{k+2} \ldots A_m$ could be the empty word ϵ.
  If $A_{k+2} \ldots A_m \neq \epsilon$, then $B_1 B_2 \ldots B_k A_{k+2} \ldots A_m \to_{\mathcal{S}}^{+} B_1 B_2 \ldots B_k$
  by applying $A_{k+2} \to \epsilon, \ldots, A_m \to \epsilon$
- *Let $\varphi(u) = \varphi(v) \wedge u \succ^{lex} v$.*
  $\leadsto\ u\downarrow_{\mathcal{T}} \neq v\downarrow_{\mathcal{T}}$
  $\because \quad$ *definition of $\mathcal{T}$ (otherwise $u = v$)*
  $\leadsto$ *We have to show that $u\downarrow_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v\downarrow_{\mathcal{T}}$:*

$u \not\succ^{lex} v$

$\leadsto$ $u|_{\mathcal{T}} \not\succ^{lex} v|_{\mathcal{T}}$

$\because$ *lemma 4.6*

Let $u|_{\mathcal{T}} = A_1 \ldots A_m \not\succ^{lex} B_1 \ldots B_m = v|_{\mathcal{T}}$

$\leadsto$ $(\exists i)(\forall j < i)\, A_j = B_j \ \wedge\ A_i \succ B_i$

$\leadsto$ *It is to show that* $A_1 \ldots A_{i-1} A_i \ldots A_m \xrightarrow{+}_{\mathcal{S}} A_1 \ldots A_{i-1} B_i \ldots B_m$*:*

$A_1 \ldots A_{i-1} A_i \ldots A_m \rightarrow_{\mathcal{S}} A_1 \ldots A_{i-1} B_i \$ A_{i+1} \ldots A_m \rightarrow_{\mathcal{S}}$

$A_1 \ldots A_{i-1} B_i B_{i+1} \$ A_{i+2} \ldots A_m \rightarrow_{\mathcal{S}} \ldots \rightarrow_{\mathcal{S}}$

$A_1 \ldots A_{i-1} B_i B_{i+1} \ldots B_m \$ \rightarrow_{\mathcal{S}} A_1 \ldots A_{i-1} B_i \ldots B_m$

'$\Leftarrow$'

- *Let* $u|_{\mathcal{T}} = v|_{\mathcal{T}} \ \wedge\ u \xrightarrow{+}_{\mathcal{T}} v$.

  $\leadsto$ $u = v$

  $\leadsto$ $u \not\xrightarrow{+}_{\mathcal{T}} v$

  $\leadsto$ *This case is not possible.*

- *Let* $u|_{\mathcal{T}} \neq v|_{\mathcal{T}} \ \wedge\ u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}} v|_{\mathcal{T}}$.

  Let $u = a_1 \ldots a_m$, $v = b_1 \ldots b_k$.

  $\leadsto$ $u|_{\mathcal{T}} = A_1^{\varphi_\Sigma(a_1)} \ldots A_m^{\varphi_\Sigma(a_m)}$, $v|_{\mathcal{T}} = B_1^{\varphi_\Sigma(b_1)} \ldots B_k^{\varphi_\Sigma(b_k)}$

  $\leadsto$ $u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S} \cup \mathcal{T}} v|_{\mathcal{T}}$

  $\leadsto$ $u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v|_{\mathcal{T}}$

  $\because$ *definition of* $\mathcal{S}$ *and* $\mathcal{T}$

  $\leadsto$ $u|_{\mathcal{T}} \xrightarrow{+}_{\mathcal{S}} v|_{\mathcal{T}}$ *implies that one of the rules of the sets* $\{A_i A_j \rightarrow A_l \$ \mid i,j,l = 1,\ldots,n\}$, $\{A_i \rightarrow \epsilon \mid i = 1,\ldots,n\}$ *or* $\{A_i \rightarrow A_j \$ \mid a_i \succ a_j\}$ *must be applied*

  $\because$ $u|_{\mathcal{T}}$ *and* $v|_{\mathcal{T}}$ *do not contain* \$

  1. *Applying* $A_i A_j \rightarrow A_l \$$*:*

     $\leadsto$ $|u|_{\mathcal{T}}| > |v|_{\mathcal{T}}|$

     $\because$ *all other rules of* $\mathcal{S}$ *do not increase the number of letters of* $\Sigma' \setminus \{\$\}$

     $\leadsto$ $\varphi(u) > \varphi(v)$

     $\because$ $|w|_{\mathcal{T}}| = \varphi(w)$

  2. *Applying* $A_i \rightarrow \epsilon$*: Analogous to 1.*

  3. *Applying no rule of the set* $\{A_i A_j \rightarrow A_l \$ \mid i,j,l \in [1,n]\} \cup \{A_i \rightarrow \epsilon \mid i \in [1,n]\}$

     $\leadsto$ *We have to apply a rule of the set* $\{A_i \rightarrow A_j \$ \mid a_i \succ a_j\}$ *at least once. The remaining part of the proof can be performed analogously to the case '*$\Leftarrow$*:* $u|_{\mathcal{T}} = v|_{\mathcal{T}} \ \wedge\ u \xrightarrow{+}_{\mathcal{T}} v$*' of lemma 4.5.* $\diamond$