Vademecum of Divergent Term Rewriting Systems*

Miki HERMANN
Centre de Recherche en Informatique de Nancy
CNRS and INRIA-Lorraine
Campus Scientifique, BP 239,
54506 Vandœuvre-lès-Nancy, France

e-mail: hermann@loria.crin.fr

Abstract

This paper presents two structural patterns to detect divergence of the completion procedure, followed by a detailed overview of different examples of divergent rewrite systems. Further it introduces five different empirical methods to avoid divergence, applicable during a session with a rewrite rule laboratory.

1 Introduction

Applying the Knuth–Bendix procedure [36] to complete a given term rewriting system may result in producing the canonical term rewriting system or failure, if it halts, or in a diverging process trying to generate an infinite set of rewrite rules. The first result concerning structural properties of an infinite set of rules generated by the Knuth–Bendix completion procedure is due to Huet [26]. Further attempts to study the reasons of Knuth–Bendix procedure divergence through investigation of the structural properties of the generated rules were initiated in [22], and more developed in [20]. The notion of *crossed term rewriting systems* as a sufficient criterion for proving divergence of the Knuth–Bendix procedure emerged from these investigations. Recently, a complementary research focused on the finite description of infinite sets of rules generated by diverging Knuth–Bendix procedure, developing the notion of *meta–rules*, is due to Kirchner [35].

This paper has a twofold purpose. First, it serves as a comprehensive collection of examples for divergent rewrite systems. Second, it presents a possible instruction kit for dealing with divergent rewrite systems. The proposed remedies are strictly empirical and can be divided mainly into five categories:

1. changes within an ordering class to obtain a new particular ordering of the same type (e.g. in the class of recursive path orderings change the operator precedence), to order an equation in the opposite direction,
2. choosing another ordering class (e.g. instead of recursive path ordering choose recursive decomposition ordering, polynomial ordering or transformation ordering) in order to get the right direction of rewrite rules,
3. separating the rule that causes divergence into a sequence of less complex equations, if the constructor(s) in the equational theory are provided,
4. dividing the rule that causes divergence into two rules,
5. enriching the system by new rewrite rules to avoid divergence originated from underspecification.

*Research report CRIN 88-R-022. Appeared in "*Avancées en Programation*" – *Journées AFCET-GROPLAN*, Nice (France), BIGRE, volume 70, January 1990.

These actions are to be taken into account during sessions with a rewrite rule laboratory such as REVE [39], when a decision must be met quickly in order to obtain a canonical (i.e. confluent, terminating, and interreduced) rewrite system. Of course, these suggestions need not to produce the desired effect, mostly for the reason that the changed rewrite system does not correspond with the user's ideas. In this case the user is proposed to upgrade to formal methods developed by Kirchner [35].

2 Completion of term rewriting systems

For completeness, the notation used in the term rewriting system theory is introduced in the first part of this section. The second part contains the basic definitions concerning term rewriting systems in general with references to related work, and introduces a review of notions from background papers which are frequently referred to in this paper. The last part contains a definition of a completion procedure in a general framework.

2.1 Basic notation and definitions

We adopt the notation of [12].

Let $\mathcal{F}$ be a finite or enumerable set of *function symbols* graded by arity (signature). $\mathcal{F}_0$ denotes the constants. Let $\mathcal{X}$ be an enumerable set of *variables* such that $\mathcal{F} \cap \mathcal{X} = \emptyset$. Denote by $\mathcal{T}(\mathcal{F}, \mathcal{X})$ the set of all *terms* (free algebra) over variables $\mathcal{X}$ and symbols $\mathcal{F}$. $\mathcal{V}ar(t)$ denotes the set of all variables in the term t. Denote by $\mathcal{G}(\mathcal{F})$ the set of all *ground terms* with function symbols $\mathcal{F}$.

Let N^* be the set of strings of natural numbers with a special symbol $\Lambda \in N^*$ for the empty string and a concatenation operation on N^*. Using the elements of N^* as labels, the terms can be viewed as labeled trees. A term t is a partial function $N^* \to \mathcal{F} \cup \mathcal{X}$ such that its domain $\mathcal{P}os(t)$ satisfies the following properties:

1. if $t \in \mathcal{F}_0 \cup \mathcal{X}$ then $\mathcal{P}os(t) = \{\Lambda\}$,
2. if $t = f(t_1, \ldots, t_n)$ then $\mathcal{P}os(t) = \{\Lambda\} \cup \{i.a \mid i = 1, \ldots, n \text{ and } a \in \mathcal{P}os(t_i)\}$

$\mathcal{P}os(t)$ is the set of *positions* of the term t. The subset of non-variable positions of t is denoted by $\mathcal{FP}os(t)$.

A *subterm* of t at a position $a \in \mathcal{P}os(t)$ is denoted by $t|_a$. If $t = f(t_1, \ldots, t_n)$ then $t|_\Lambda = t$ and $t|_{ia} = t_i|_a$ for all $i = 1, \ldots, n$. Denote by $s[t]_a$ a new term obtained from the term s after replacing its subterm $s|_a$ by t. For properties of replacement see the article [47].

A *substitution* is a function $\sigma\colon \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $x\sigma = x$ holds for all but a finite number of variables. Denote a substitution σ by $[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ when the terms t_i are substituted for the variables x_i and $x_i \neq t_i$, for $i = 1, \ldots, n$. An empty substitution is denoted by $[]$. Substitutions have a homomorphic extension on the terms. Denote by $\mathcal{D}om(\sigma)$, $\mathcal{R}an(\sigma)$, and $\mathcal{V}ar(\sigma)$ the domain, range, and all variables of a substitution σ, respectively. A substitution σ, such that $x\sigma \in \mathcal{X}$ and $x\sigma = y\sigma$ implies $x = y$ for all $x, y \in \mathcal{D}om(\sigma)$, is a *variable renaming*. Substitutions need not be idempotent in our approach.

Two terms s and t are *unifiable* if, and only if, there is an idempotent substitution σ such that $s\sigma = t\sigma$. The substitution σ is called a *unifier*. The substitution σ is called the *most general unifier* (up to variable renaming) for s and t if for all unifiers φ of s and t there exists a substitution ψ, such that $\varphi = \sigma\psi$. The substitution σ on term t is a *substitution in own variables* of t if it does not introduce new variables, i.e. $\mathcal{V}ar(t\sigma) \subseteq \mathcal{V}ar(t)$, and does not contain a variable renaming. This notion can be enlarged to a set of substitutions.

An *equation* is a pair of terms $e = (s, t)$. For convenience, the equations are written as $s \simeq t$ with undistinguished left and right hand side. A *rewrite rule* is an ordered pair of terms $r = (s, t)$ such that $\mathcal{V}ar(t) \subseteq \mathcal{V}ar(s)$. The rules are written $s \to t$. A *term rewriting system* (or *rewrite system* for short) is a finite set of rules $R = \{s \to t \mid s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})\}$.

A *rewriting relation* $\longrightarrow_R$ (or $\longrightarrow$ when R is obvious) is the smallest relation containing R, closed under substitution and replacement. The relation $\overset{*}{\longrightarrow}$ denotes the reflexive and transitive closure of $\longrightarrow$, the relation $\longleftarrow$ denotes the relation symmetric to $\longrightarrow$, the equivalence relation $\overset{*}{\longleftrightarrow}$ denotes the reflexive, symmetric, and transitive closure of $\longrightarrow$. An ordering $\succ$ is *compatible* with R if $\overset{*}{\longrightarrow}_R \subseteq \succ$.

A term t is *reducible* by the rule $l \to r$ if there exists an position $a \in \mathcal{P}os(t)$ and a substitution σ such that $t|_a = l\sigma$. A term is *R-reducible* if it is reducible by some rule from R. On the contrary, a term is *R-irreducible* if it is not reducible by any rule from R. A term t is a *R-normal form* of a term s if $s \overset{*}{\longrightarrow} t$ and t is R-irreducible. A rewrite system R is *interreduced* if for all rules $l \to r \in R$ the left-hand side l is $R - \{l \to r\}$-irreducible and the right-hand side r is R-irreducible.

2.2 Term rewriting systems

The following definitions review basic notions from term rewriting systems theory used in this paper. For details and related theorems see the references.

Definition 2.1 *A term rewriting system R (and also the rewriting relation $\xrightarrow{*}_R$) is*

- **terminating** *if there is no infinite rewrite sequence $t_1 \longrightarrow t_2 \longrightarrow \ldots$,*
- **confluent** *if $\xleftarrow{*}_R \cdot \xrightarrow{*}_R \subseteq \xrightarrow{*}_R \cdot \xleftarrow{*}_R$,*
- **convergent** *if it is confluent and terminating,*
- **canonical** *if it is convergent and interreduced.*

Both basic properties, confluence and termination of a rewrite system, are undecidable in general. For issues on termination and **reduction orderings** (well-founded orderings on terms closed under term replacement and substitution) see the review by Dershowitz [10]. For issues concerning confluence see the article by Huet [25].

The following notion of *critical pairs* as overlap instances of rewrite rules was introduced by Knuth and Bendix [36]. Their production and orientation into new rewrite rules is the backbone of their completion procedure.

Definition 2.2 *Let $s_1 \to t_1$ and $s_2 \to t_2$ be two rewrite rules such that $s_1|_a\sigma = s_2\sigma$ holds for a most general unifier σ, and nonvariable position $a \in \mathcal{FP}os(s_1)$. Then $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle$ is called a* **critical pair** *of terms. Denote the set of all critical pairs generated from a rewrite system R by $cp(R)$. A critical pair $\langle t, t\rangle$, for some term t, is called* **trivial**.

The following construction was originated by Lankford and Musser [38] and comes from Guttag, Kapur and Musser [18]. They are mentioned also by Dershowitz [10].

Definition 2.3 *Let R be an arbitrary set of rules. The set of* **overlap closures** *$OC(R)$ of R is inductively defined as follows:*

1. *Every rule $s \to t$ from R is an overlap closure $s \rightarrowtail t$.*
2. *Let $s_1 \rightarrowtail t_1$, $s_2 \rightarrowtail t_2$ be two overlap closures. If $t_1|_a\sigma = s_2\sigma$ holds for a most general unifier σ and position $a \in \mathcal{FP}os(t_1)$, then $s_1\sigma \rightarrowtail t_1\sigma[t_2\sigma]_a$ is an overlap closure.*
3. *Let $s_1 \rightarrowtail t_1$, $s_2 \rightarrowtail t_2$ be two overlap closures. If $t_1\sigma = s_2|_a\sigma$ holds for a most general unifier σ and position $a \in \mathcal{FP}os(s_2)$, then $s_2\sigma[s_1\sigma]_a \rightarrowtail t_2\sigma$ is an overlap closure.*

An overlap closure $s \rightarrowtail s$ with the same terms on both sides is called **reflexive**.

This notion is related to the narrowing process, as defined by Fay [15], Hullot [29], Lankford [37] and Slagle [50], and to the superposition process defined in [18]. The overlap closure provides the essence of a specific method for proving termination of term rewriting systems. A survey of related results can be found in [10].

To define better the divergence patterns let us introduce the following relation between substitutions. It was actually defined in [19].

Definition 2.4 *The substitutions φ and ψ are* **coherent** *(denote it by $\varphi - \psi$) if $\mathcal{D}om(\varphi) \cap \mathcal{V}ar(\psi) = \emptyset$ or $\mathcal{V}ar(\varphi) \cap \mathcal{D}om(\psi) = \emptyset$.*

2.3 Completion procedure

Given a finite set E of equations presenting a theory and a program for (possibly incremental) computing a reduction ordering $\succ$, a *completion procedure* deduces consequences of E in its attempt to find a convergent (confluent and terminating) rewrite system R^∞ for E. The central idea of completion is to limit attention to the *critical pair* deductions obtained from overlappings of left-hand sides of rules. These critical pair overlappings are used to generate new rules. The first completion procedure was proposed by Knuth and Bendix [36], and completely proved correct by Huet [26]. Bachmair, Dershowitz and Hsiang [5] have put completion in a more abstract framework, based on the notion of *inference rules* (see also [4] and [11]). An *inference rule* (for this purpose) is a binary relation between pairs $(E; R)$, where E is a set of equations and R is a set of rewrite rules. The generalized Knuth–Bendix completion procedure is based on the set KB of six inference rules, presented in Figure 1.

A **completion procedure** is a *control* strategy for applying inference rules of KB to given input equations and rewrite rules, using a reduction ordering compatible with these rewrite rules. The result of a (possibly infinite) completion sequence $(E_0; R_0) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots$ are the set $E^\infty = \lim_{n\to\infty} E_n$ of persisting equations and the set $R^\infty = \lim_{n\to\infty} R_n$ of persisting rules.

This inference rule based completion procedure must be *fair* (processing of each critical pair cannot be postponed infinitely many times), *correct* (whenever the procedure finishes successfully, it produces a convergent rewrite system), and *sound* (the smallest equivalence relation generated by $E \cup R$ remains the

Delete:	$(E \cup \{s \simeq s\}; R)$	$\vdash$	$(E; R)$	
Compose:	$(E; R \cup \{s \to t\})$	$\vdash$	$(E; R \cup \{s \to u\})$	if $t \longrightarrow_R u$
Simplify:	$(E \cup \{s \simeq t\}; R)$	$\vdash$	$(E \cup \{u \simeq t\}; R)$	if $s \longrightarrow_R u$
Orient:	$(E \cup \{s \simeq t\}; R)$	$\vdash$	$(E; R \cup \{s \to t\})$	if $s \succ t$
Collapse:	$(E; R \cup \{s \to t\})$	$\vdash$	$(E \cup \{u \simeq t\}; R)$	if $s \longrightarrow_R u$ by $l \to r \in R$ with $s \triangleright\!\bullet\, l$
Deduce:	$(E; R)$	$\vdash$	$(E \cup \{s \simeq t\}; R)$	if $s \simeq t \in cp(R) - E$

where $\triangleright\!\bullet$ denotes a proper encompassment ordering.

Figure 1: Inference rules of the completion procedure

same through the completion process). Formal definitions of these three concepts can be found in [11].

The inference rules KB can be applied in many different ways, but all of them fair, correct, and sound. We choose two of them. The first is a general completion procedure *complete* as it was presented e.g. by Huet [26] or in a more sophisticated way by Lescanne [41], the second is a nonreducing completion process *nr-complete*, generating all critical pair consequences without interreduction. For corresponding control strategies see [20].

Summarizing the possible performances, the completion procedure can *succeed* in generating a finite convergent/canonical rewrite system R^∞, *fail* due to an equation unorientable in a reduction ordering $\succ$, or **diverge** trying to produce an infinite rewrite system R^∞.

3 Divergent term rewriting systems

This section recalls the theoretical notions and results from the paper [20], with a collection of examples presenting the divergence patterns.

For given rewrite rules R (or equations E), the completion procedure performance depends on the applied control expression and the input ordering $\succ$. Therefore, more hierarchically ordered types of divergence can be observed.

Definition 3.1 *Let R be a term rewriting system. R is* **divergent** *in the ordering $\succ$ if complete(R) is infinite. R is* **weakly divergent** *in the ordering $\succ$ if nr-complete(R) is infinite. R is* **inherently (weakly) divergent** *if R is (weakly) divergent for all orderings including $\xrightarrow{*}_R$.*

Whether a rewrite system is inherently divergent it depends on the structure of its rules. Each divergent rewrite system is also weakly divergent. A connection between divergent and weakly divergent rewrite systems is established by the following proposition.

Fact 3.2 *Let R be a weakly divergent rewrite system and let $R' \subseteq R$ be a subsystem which is also weakly divergent. If nr-complete(R') $\subseteq$ complete(R), then R is divergent.*

In establishing conditions for detection of divergence there arises a question how large is the actual class of divergent systems described by them, and/or if it covers all possible divergent systems. Unfortunately, the *divergence problem is undecidable* in general even if it contains only monadic function symbols and constants, as it was proved in [20] by a modification of the proof method used by Narendran and Stillman [45].

From the undecidability of divergence follows that it is reasonable to search only for sufficient conditions to detect divergent rewrite systems. The rest of this section introduces as sufficiently general conditions of this kind as possible to be able to describe the largest class of divergent systems. The attention is oriented on examples actually presenting the divergence patterns.

3.1 Forward crossed systems

The following definition describes a divergence pattern for the class of rewrite systems with forward oriented critical pairs. This pattern represents a further generalization of the crossed rules notion defined in [21, 22]. It also comprises the divergence types (Λ, γ) (actually crossed rules) and $(\Lambda, \Lambda/\gamma)$ (self-crossed rule) described by Mong and Purdom [44]. For the technical description we refer to [20]. The presentation here is focused more on the examples.

Definition 3.3 *The rewrite rule $s_1 \to t_1$ and the nonreflexive overlap closure $s_2 \rightarrowtail t_2$ (with supposed disjoint variables) form a* **forward crossed rewrite**

system *if there are substitutions σ_2, φ_1, φ_2 in own variables of s_2, an idempotent substitution σ_1, and positions $a \in \mathcal{FP}os(s_1)$, $b \in \mathcal{FP}os(t_2)$ such that*

1. $s_1|_a \sigma_1 = s_2\sigma_2$
2. $t_2|_b \varphi_1 = s_2\varphi_2$
3. $\varphi_1 - (\varphi_2 \cup \sigma_2)$

The second condition plus the coherence relation $\varphi_1 - \varphi_2$ define $s_2 \rightarrowtail t_2$ to be a *forward chain* [19].

The previous definition supplies the static part of conditions to detect forward divergent systems. The dynamic part of conditions, introduced in the following definition, puts requirements on the ordering of generated critical pairs.

Definition 3.4 *The rewrite system R is* **LR-persistent** *in the ordering $\succ$ if for each nontrivial critical pair of terms $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle \in cp(R^\infty)$ follows $s_1\sigma[t_2\sigma]_a \succ t_1\sigma$.*

The prefix *LR-* indicates that the produced critical pairs are oriented in the straight direction, from *left* to *right*.

From these two notions the following proposition was proved in [20], which guarantees that the previous conditions are sufficient for weak divergence. Of course, interreduction is not admitted, but for many examples the Fact 3.2 can be applied, so that divergence of these systems is implied as well. For conditions on interreduction in divergence see [20].

Theorem 3.5 *If R contains a LR-persistent forward crossed rewrite system then R is weakly divergent.*

Let us consider now the examples that belong to the forward crossed divergence pattern. It is a collection of divergent systems observed during real specifications as well as artificial cases constructed in the effort to find more and more complex conditions that would cover even a greater class of divergent systems.

Example 3.6 It has been almost a common folklore (this example was mentioned among others by Fribourg [16] and by Göbel [17]) that if you extract the rules

$$(x' + y') + z \quad \rightarrow \quad x' + (y' + z) \qquad (1)$$
$$x + s(y) \quad \rightarrow \quad s(x + y) \qquad (2)$$

from a rewrite system specifying natural numbers with addition and you try to complete them, using recursive path ordering RPO with the precedence $+ > s$ and a left-to-right status of $+$, the result will be a divergent process. The rule (2) forms the forward chain, the positions are $a = 1$ and $b = 1$, and the substitutions are $\sigma_1 = [x' \mapsto x, y' \mapsto s(y)]$, $\sigma_2 = []$, $\varphi_1 = [y \mapsto s(y)]$, $\varphi_2 = []$. The completion procedure generates the infinite family of rules

$$s^n(x + y) + z \quad \rightarrow \quad x + (s^n(y) + z)$$

from them. This is also the reason why the proof by consistency [32] of the associativity in the rewrite system

$$x + 0 \quad \rightarrow \quad x$$
$$x + s(y) \quad \rightarrow \quad s(x + y)$$

using an unappropriate reduction ordering (RPO with the precedence $+ > s$ in this case) results in an infinite loop, as described by Dershowitz [11] and Fribourg [16].

In [21] the rewrite system

$$x + 0 \quad \rightarrow \quad x \qquad (3)$$
$$x + s(y) \quad \rightarrow \quad s(x + y) \qquad (4)$$
$$gcd(x, 0) \quad \rightarrow \quad x$$
$$gcd(0, x) \quad \rightarrow \quad x$$
$$gcd(x' + y', y') \quad \rightarrow \quad gcd(x', y') \qquad (5)$$

specifying the greatest common divisor of two natural numbers was proved as divergent. It contains a forward crossed system consisting of the rules (4) and (5). The rule (4) forms the forward chain, the positions are $a = 1$ and $b = 1$, and the substitutions are $\sigma_1 = [x' \mapsto x, y' \mapsto s(y)]$, $\sigma_2 = []$, $\varphi_1 = [y \mapsto s(y)]$, $\varphi_2 = []$, as in the previous case. The completion procedure, using RPO with the precedence $+ > s$, generates the infinite family of rules

$$gcd(s^n(x + y), s^n(y)) \quad \rightarrow \quad gcd(x, s^n(y))$$

from rules (4) and (5), plus another infinite family

$$gcd(s^n(x), s^n(0)) \quad \rightarrow \quad gcd(x, s^n(0))$$

derived from the first one by the rule (3). The second infinite family cannot be produced independently because the underlying overlap closure $x + s(0) \rightarrowtail s(x)$, constructed consecutively from the rules (4) and (3), does not form a forward chain.

To show that divergence problems are not caused only by natural number specifications, let us consider also other algebraic structures.

Example 3.7 A very simple and elegant example of a forward crossed system is *Associativity & Endomorphism*. It was considered by Bellegarde [7],

BenCherifa with Lescanne [9], and Martin [43]. The rewrite system

$$\begin{aligned}(x' + y') + z &\rightarrow x' + (y' + z)\\ f(x) + f(y) &\rightarrow f(x + y)\end{aligned}$$

under completion, using RPO with the precedence $+ > f$ and the left-to-right status of $+$, produces a forward crossed divergent system. The *Endomorphism* rule forms the forward chain, the positions are $a = 1$ and $b = 1$, and the substitutions are $\sigma_1 = [x' \mapsto f(x), y' \mapsto f(y)]$, $\sigma_2 = [\,]$, $\varphi_1 = [x \mapsto f(x), y \mapsto f(y)]$, $\varphi_2 = [\,]$. The completion procedure generates the infinite family of rules

$$f^n(x + y) + z \rightarrow f^n(x) + (f^n(y) + z)$$

from it.

Similar to the previous system is *Associativity & Distributivity* studied by Lescanne [40], and mentioned also by Martin [43] and Mong together with Purdom [44]. The rewrite system

$$\begin{aligned}(x' + y') + z' &\rightarrow x' + (y' + z')\\ (x * y) + (x * z) &\rightarrow x * (y + z)\end{aligned}$$

under completion, using RPO with the precedence $+ > *$ and a left-to-right status of $+$, produces a divergent system. It is actually a forward crossed system with the *Distributivity* rule forming the forward chain, and the substitutions $\sigma_1 = [x' \mapsto x * y, y' \mapsto x * z]$, $\sigma_2 = [\,]$, $\varphi_1 = [y \mapsto x * y, z \mapsto x * z]$, $\varphi_2 = [\,]$. As a matter of fact, it is a variation of the previous divergent system, where the operation f is interpreted as a curried multiplication.

A divergent system need not consist of only two rules, as the previous examples were. The cardinality of divergent systems is not even bound. On the other hand, one rule can be sufficient to produce a divergent system.

Example 3.8 There exists a one-rule forward crossed system

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

introduced by Ardis [1], and observed among others by Dershowitz with Marcus [13, 14] and Kirchner [35]. This only rule stands for both objects required by the Definition 3.3. The positions are $a = 1.1$ and $b = 1$, the substitutions are $\sigma_1 = [x \mapsto g(f(x))]$[1], $\sigma_2 = [\,]$, $\varphi_1 = [x \mapsto g(f(x))]$, $\varphi_2 = [\,]$. The completion procedure generates the infinite family of rules

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

from it.

This one-rule divergent system bears another phenomenon. If we want to assure termination of rewriting by the generated system, its rules cannot be ordered in the opposite direction. Therefore the divergence of this system is *inherent*.

There exists also a forward crossed rewrite system consisting of a given number of rules, where the presence of all rules is necessary to maintain the divergence. As it was proved in [21], the rewrite system

$$\begin{aligned}f_{n+1}(f_{n-1}(x')) &\rightarrow x'\\ f_i(f_n(x)) &\rightarrow f_{i-1}(x) \qquad \text{for } i = 2, \ldots, n-1\\ f_1(f_n(x)) &\rightarrow f_0(f_{n-1}(x))\end{aligned}$$

ordered by RPO based on the precedence $f_i > f_j$ for $i > j$, is divergent for all n, but each proper subset of it can be completed to a finite canonical system. The first rule presents the basis of divergence and the rest forms the forward chain $f_{n-1}(f_n^{n-1}(x)) \blacktriangleright\!\!\rightarrow f_0(f_{n-1}(x))$. The substitutions are $\sigma_1 = [x' \mapsto f_n^{n-1}(x)]$, $\sigma_2 = [\,]$, $\varphi_1 = [x \mapsto f_n^{n-1}(x)]$, $\varphi_2 = [\,]$. The completion procedure generates the infinite family of rules

$$f_{n+1}(f_0^k(f_{n-1}(x))) \rightarrow f_n^{k(n-1)}(x)$$

from it, plus all the intermediate infinite families according to the rules that participate in forming the forward chain.

When someone looks properly on the previously presented examples, he or she may ask why the Definition 3.3 is so complicated. Not the definition is too complicated, but the examples were too easy. They all satisfy the simpler conditions from [22]. Let us analyze more sophisticated systems where all the conditions of Definition 3.3 must be applied and where the conditions from [22] appear not to be sufficient enough. The only drawback of these examples is that they are artificially constructed and do not reflect any reasonable algebraic structure. Nevertheless, when they are divergent, they should be covered by the definition, as it is also true in our case.

Example 3.9 A more general forward crossed rewrite system is[2]

$$\begin{aligned}d(x' \oplus h(y')) &\rightarrow y'\\ (x \otimes y) \oplus y &\rightarrow k(x \oplus k(y))\end{aligned}$$

[1] The ambiguity in variables can be resolved by doubling the rewrite rule and splitting the variables.

[2] The circled operators are to be considered only as syntactic objects, and not as real operators.

If we try to complete this system, using RPO based on the operator precedence $\otimes > \oplus$ and $\otimes > k$, we get a divergent process. The second rule forms the forward chain. The substitutions are $\sigma_1 = [x' \mapsto x \otimes h(y), y' \mapsto y]$, $\sigma_2 = [y \mapsto h(y)]$, $\varphi_1 = [x \mapsto x \otimes k(y)]$, $\varphi_2 = [y \mapsto k(y)]$. The completion procedure generates the infinite family of rules

$$d(k^n(x \oplus k^n(h(y)))) \quad \to \quad y$$

from it.

A similar case presents the forward crossed rewrite system

$$\begin{aligned} d(x' \oplus (x' \otimes y')) &\to y' \\ g(x) \oplus y &\to g(x \oplus (x \oslash y)) \end{aligned}$$

only that it is of the second type. If we try to complete it, using RPO based on the operator precedence $\oplus > \oslash$, $\oplus > g$ and a left-to-right status of $\oplus$, we get a divergent process. The second rule forms the forward chain. The substitutions are $\sigma_1 = [x' \mapsto g(x), y' \mapsto y]$, $\sigma_2 = [y \mapsto g(x) \otimes y]$, $\varphi_1 = [x \mapsto g(x)]$, $\varphi_2 = [y \mapsto g(x) \oslash y]$. The completion procedure generates the infinite family of rules

$$d(g^n(x \oplus (x \oslash (g(x) \oslash \ldots (g^n(x) \otimes y))))) \quad \to \quad y$$

from it.

Of course, not all systems are so clear to analyze. There can be systems with more than one forward chain or systems with multiple overlap positions. Let us look at such system which has been encountered as a practical case.

Example 3.10 The following rewrite system specifies the signed binary trees theory [34]. It is presented here as an extract from a (canonical) rewrite system specifying groups. It is the system

$$\begin{aligned} i(i(x)) &\to x \\ i(x * y) &\to i(y) * i(x) \\ (y * x) * i(x) &\to y \\ i(x) * (x * y) &\to y \end{aligned}$$

If we try to complete this system, using RPO based on the operator precedence $i > *$, we get a divergent process. First, the rules

$$\begin{aligned} (y * i(x)) * x &\to y \\ x * (i(x) * y) &\to y \end{aligned}$$

are produced, then the infinite iterative process begins. The clue here is that the underlying rewrite system

$$\begin{aligned} i(i(x)) &\to x \\ i(x * y) &\to i(y) * i(x) \end{aligned}$$

generates an infinite set of *independent* forward chains. Therefore an infinite set of infinite families of rules is produced. Although it is a little bit surprising, it is coherent with the theories in [20] and in [35]. The forward chain producing rewrite system is canonical, thus decidable, so that every computation modulo whichever forward chain is also decidable.

3.2 Backward crossed systems

The following definitions describe another divergence pattern for the class of rewrite systems with backward oriented critical pairs. This pattern was introduced in [22] only by an example, and was not treated formally. Its definition covers the divergence types (γ, Λ) (reverse crossed rules) and $(\Lambda/\gamma, \Lambda)$ (reverse self-crossed rule) described by Mong and Purdom [44], but it is more compact. For its technical description we refer once more to [20]. Here we focus our attention more on the examples actually presenting the divergence pattern.

Definition 3.11 *The overlap closure $s_1 \rightarrowtail t_1$ and the nonreflexive rewrite rule $s_2 \to t_2$ (with supposed disjoint variables) form a* **backward crossed rewrite system** *if t_1 is not a variable, there are substitutions σ_1, φ_1, φ_2 in own variables of s_1, an idempotent substitution σ_2, and position $b \in \mathcal{FP}os(s_1)$ such that*

1. $s_1|_b \sigma_1 = s_2\sigma_2$
2. $t_1\varphi_1 = s_1|_b \varphi_2$
3. $\varphi_1 - (\varphi_2 \cup \sigma_1)$

The second condition plus the coherence relation $\varphi_1 - \varphi_2$ define $s_1 \rightarrowtail t_1$ to be a *backward chain* [19].

The previous definition supplies the static part of conditions to detect backward divergent systems. If it was only for the static parts, a *duality principle* could be established between forward and backward crossed rewrite systems, originating from a similar one between forward and backward chains [19]. The dynamic part of conditions, introduced in the following definition, which puts requirements on the ordering of generated critical pairs, is the main difference between the notions defined in 3.3 and 3.11, making them to two completely different divergence patterns.

Definition 3.12 *The rewrite system R is* **RL-persistent** *in the ordering $\succ$ if for each nontrivial critical pair of terms $\langle s_1\sigma[t_2\sigma]_a, t_1\sigma\rangle \in cp(R^\infty)$ follows $t_1\sigma \succ s_1\sigma[t_2\sigma]_a$.*

The prefix *RL-* indicates that the produced critical pairs are oriented in the opposite direction, from *right* to *left*.

From the previously defined notions the following proposition was proved in [20], which guarantees that the previous conditions are sufficient for weak divergence. The same discussion concerning interreduction as in the case of forward crossed systems applies in this case, too.

Theorem 3.13 *If R contains a RL-persistent backward crossed rewrite system then R is weakly divergent.*

Let us consider the examples that belong to the backward crossed divergence pattern. This divergence type is not so common as the previous one, at least not so many real cases of it are known yet.

Example 3.14 An elegant example of a backward crossed system is the (decidable) theory of *bands* (idempotent semigroups)

$$\begin{array}{rcl}(x * y) * z & \to & x * (y * z)\\ x' * x' & \to & x'\end{array}$$

studied by Siekmann with Szabó [49], and in connection with divergence by Dershowitz [11] and Kirchner [35]. The *Associativity* rule produces an infinite number of independent backward chains, depending on the multiple choice of overlap positions.

As a marginal remark, it should be mentioned also that there exists no canonical unconditional rewrite system for idempotent semigroups [49].

Also in this type there exist multiple rule divergent systems as well as a one-rule divergent system. The one-rule system is an extract from a real example in this case.

Example 3.15 Extracting the rule

$$(x\backslash y)\backslash z \quad \to \quad y\backslash(i(x)\backslash z)$$

from a rewrite system for deciding groups with left division, studied by Lescanne in [42], and ordering it by RPO based on the precedence $\backslash > i$ and the left-to-right status of $\backslash$, presents a backward crossed system. Once more, an infinite number of independent backward chains is produced from the starting rule.

There exists also a backward crossed rewrite system consisting of a given number of rules, where the presence of all of them is necessary to maintain the divergence. It is the rewrite system

$$\begin{array}{rcll}f_1(f_0(x')) & \to & x' & \\ f_n(f_{i-1}(x)) & \to & f_i(f_n(x)) & \text{for } i = 2, \ldots, n-1\\ f_n(f_{n-1}(x)) & \to & f_1(f_n(x)) & \end{array}$$

oriented by RPO based on the precedence $f_i > f_j$ for $i > j$. It is divergent for all n, but each proper subset of it can be completed to a finite canonical system. The first rule presents the basis of divergence and the rest forms the backward chain $f_n^{n-1}(f_1(x)) \rightarrowtail f_1(f_n^{n-1}(x))$. The substitutions are $\sigma_1 = [x \mapsto f_0(x)]$, $\sigma_2 = [x' \mapsto x]$, $\varphi_1 = []$, $\varphi_2 = [x \mapsto f_n^{n-1}(x)]$. The completion procedure generates the infinite family of rules

$$f_1(f_n^{k(n-1)}(f_0(x))) \quad \to \quad f_n^{k(n-1)}(x)$$

from it, plus all the intermediate infinite families according to the rules that participate in forming the backward chain.

Also in this divergence type artificially constructed systems can be presented that reflect very well the conditions required by the Definition 3.11.

Example 3.16 Consider the rewrite system

$$\begin{array}{rcl}f(x \vee g(y)) & \to & f(x) \vee y\\ (x' \wedge y') \vee y' & \to & y'\end{array}$$

If we try to complete this system, using RPO based on the operator precedence $f > \vee$, we get a divergent process. The first rule forms the backward chain. The substitutions are $\sigma_1 = [x \mapsto x \wedge g(y)]$, $\sigma_2 = [y' \mapsto g(y)]$, $\varphi_1 = [y \mapsto g(y)]$, $\varphi_2 = [x \mapsto f(x)]$. The completion procedure generates the infinite family of rules

$$f^n(x \wedge g^n(y)) \vee y \quad \to \quad f^n(g^n(y))$$

from it.

A similar case presents the backward crossed rewrite system

$$\begin{array}{rcl}(x \otimes f(y)) \oplus y & \to & (x \oplus y) \otimes y\\ (x' \oslash y') \otimes y' & \to & x'\end{array}$$

If we try to complete this system, using RPO based on the operator precedence $\oplus > \otimes$, we get a divergent process. The first rule forms the backward chain. The substitutions are $\sigma_1 = [x \mapsto x \oslash f(y)]$, $\sigma_2 = [x' \mapsto x, y' \mapsto f(y)]$, $\varphi_1 = [y \mapsto f(y)]$, $\varphi_2 = [x \mapsto x \oplus f(y)]$. The completion procedure generates the infinite family of rules

$$\begin{array}{r}((((x \oslash f^{n+1}(y)) \oplus f^n(y)) \oplus \ldots \oplus f(y)) \oplus y) \otimes y \to\\ ((x \oplus f^n(y))) \oplus \ldots \oplus f(y) \oplus y\end{array}$$

from it.

3.3 Undecidability of crossed systems

In many rewrite systems the existence of a crossed subsystem can be shown immediately. Of course, we may ask if it is decidable that $complete(R)$ contains a crossed system. Unfortunately, it is not.

Theorem 3.17 *It is undecidable in general whether the completion procedure generates a crossed system.*

Proof: The result was proved by Narendran and Stillman [45] for crossed pairs, so that it can be immediately applied to forward crossed systems. A minor modification extends it also to backward crossed ones. □

The result of Narendran and Stillman [45] has an immediate implication that all divergence conditions, which cover the case of the one-rule system $f(g(f(x))) \to g(f(x))$, are undecidable.

4 Avoiding divergence of completion

A divergent term rewriting system is an unpleasant fact one has to deal with. The intention is directed towards the goal to obtain a finite canonical rewrite system. Therefore something must be done with the original system to avoid its divergence but to maintain its semantics. Basically, there exist two approaches: a theoretical one and an empirical one.

The theoretical approach was formally defined and exploited by Kirchner [35]. It is an universal one that allows describing divergent systems by meta-rules. It is the only complete method that allows to cope directly with inherently divergent systems. On the other hand, it is not necessary to use the "heavy artillery" immediately when a divergent system is encountered, because the divergence of most rewrite systems is a result of specification errors. Also the meta-rule approach is not, to our knowledge, yet implemented in existing rewrite rule laboratories. Therefore it can be more convenient to use empirical methods which are, of course, not as sophisticated as the theoretical approach, but can be applied in the existing rewrite rule laboratories.

The following parts describe five empirical methods for avoiding divergence of rewrite systems, graded by the complexity of actions provided by the user. As a matter of fact, there exists one more method to avoid divergence, namely upgrading to equational rewriting and thus to equational completion [4, 30]. Although equational term rewriting provides a powerful generalization which eliminates many abnormal failure and divergent cases, its functionality is based on the premise of existence of a complete and finite equational unification algorithm, which rarely exists for an arbitrary set of equations E. Moreover, the equational term rewriting method bears another problem, namely the infinite complete set of unifiers, which presents another type of divergence [35]. Therefore it needs a broader space of investigation. For these reasons it is not analyzed here. On the other hand, the unfailing completion procedure, as it was presented in [6, 24], does not avoid the divergence problem. One can prove very easily that if the completion procedure with the control *complete* diverges then the unfailing completion procedure diverges as well.

4.1 Changes within ordering class

The first empirical method to attack divergence consists of changes within an ordering class to obtain a new concrete ordering. It can be a change of the underlying operator precedence and/or status in a recursive path ordering [10], in a recursive decomposition ordering (with status) [42], or in another related incremental ordering. It can also be a change of the underlying symbol precedence and/or weight in a Knuth–Bendix ordering [36, 43], or a change of the polynomial interpretation in a polynomial ordering [9].

Basically, it is always a (relatively minor) change to an underlying structure of the used ordering class. Within the incremental orderings it can be performed by backtracking. The aim of these changes is to order one or more equations in the opposite direction, so that the critical overlaps, which were the starting points of divergence, disappear. This method replaces objects satisfying either the Definition 3.3 or the Definition 3.11. The equivalence relation $\overset{*}{\longleftrightarrow}$, generated by the rewrite system, remains the same, but the normal forms obtained by the rearranged system (under the assumption that it can be completed to a finite canonical system) may not correspond with the initial intentions. This method has also its limits because there need not be enough possibilities to choose from during the completion process.

Example 4.1 One of the possibilities to resolve the divergence of the natural number rewrite system is to change the precedence to $s > +$ in RPO and thus obtain the canonical system

$$\begin{aligned} (x+y)+z &\to x+(y+z) \\ s(x+y) &\to x+s(y) \end{aligned}$$

where the *second* rule was ordered in the opposite direction. Although it is a possible solution, it does not satisfy the requirement that the successor operator s should be a constructor, if the rule $x + 0 \to x$ would be added.

Another and a better possibility consists of changing only the status of $+$ to right-to-left. In this case we obtain the canonical system

$$\begin{aligned} x + (y + z) &\to (x + y) + z \\ x + s(y) &\to s(x + y) \end{aligned}$$

where now the *first* rule was ordered in the opposite direction. The successor operator s can be then declared a constructor in the enlarged system.

The divergent system presented by *Associativity & Endomorphism* can be resolved by changing the precedence to $f > +$ and thus producing the canonical system

$$\begin{aligned} (x + y) + z &\to x + (y + z) \\ f(x + y) &\to f(x) + f(y) \end{aligned}$$

where it is also the second rule that has been ordered in the opposite direction. So far it seems to be an acceptable solution, unless you want to reduce the number of f operators in terms by the completed rewrite system.

In artificially constructed divergent systems the orientation of rules does not play a significant role, therefore this method can be fully applied in Examples 3.9 and 3.16. We get consecutively the canonical systems

$$\begin{aligned} d(x \oplus h(y)) &\to y \\ k(x \oplus k(y)) &\to (x \otimes y) \oplus y \end{aligned}$$

by changing the precedence to $k > \otimes$ and $k > \oplus$,

$$\begin{aligned} d(x \oplus (x \otimes y)) &\to y \\ g(x \oplus (x \oslash y)) &\to g(x) \oplus y \end{aligned}$$

by changing the precedence to $g > \oplus$,

$$\begin{aligned} f(x) \vee y &\to f(x \vee g(y)) \\ (x \wedge y) \vee y &\to y \end{aligned}$$

by changing the precedence to $\vee > f$, $\vee > g$, and a left-to-right status of $\vee$, and last but not least

$$\begin{aligned} (x \oplus y) \otimes y &\to (x \otimes f(y)) \oplus y \\ (x \oslash y) \otimes y &\to x \end{aligned}$$

by changing the precedence to[3] $\otimes > \oplus$, $\otimes > f$, and a left-to-right status of $\oplus$.

[3] There are also other possible changes of precedence.

Let us consider also rewrite systems ordered by other ordering classes, in particular the Knuth–Bendix ordering and the polynomial ordering.

Example 4.2 As it was pointed out in [36], the classical presentation of group theory by the rewrite system

$$\begin{aligned} e * x &\to x \\ i(x) * x &\to e \\ (x * y) * z &\to x * (y * z) \end{aligned}$$

ordered by the Knuth–Bendix ordering with the precedence $i > * > e$, the weight $w(*) = 0$, and the weight of the inverse operator being $w(i) > 0$, causes divergence under completion. As analyzed by Mong and Purdom [44], the completion procedure produces two crossed subsystems. It is the forward crossed system

$$\begin{aligned} x * i(y * x) &\to i(y) \\ (x * y) * z &\to x * (y * z) \end{aligned} \tag{6}$$

and the backward crossed system

$$\begin{aligned} i(x) * i(y) &\to i(y * x) \\ i(x * i(y)) &\to y * i(x) \end{aligned} \tag{7}$$

The rule (7) in the second system is the real culprit as indicated in [36], therefore it has to be reoriented. As a consequence, the rule (6) is not generated any more. The reorientation can be achieved by changing the weight of the operator i to $w(i) = 0$. After that the standard ten-rule canonical rewrite system for groups is generated under completion [43].

Example 4.3 The *Associativity & Endomorphism* rewrite system in Example 3.7 can be ordered in the same sense by the polynomial interpretation $[f](X) = 2X$ and $[+](X, Y) = X^2 + Y$. It still remains divergent, producing the same infinite family of rules [9]. Fortunately, this system can be proved to be terminating also using the polynomial interpretation $[f](X) = 2X$ and $[+](X, Y) = XY + X$. The nice property of this interpretation is that it makes possible to complete the rules to the canonical system

$$\begin{aligned} (x + y) + z &\to x + (y + z) \\ f(x) + f(y) &\to f(x + y) \\ f(x) + (f(y) + z) &\to f(x + y) + z \end{aligned}$$

where the number of f operators in terms can be reduced by the completed system.

The presented method to avoid divergence has its limits. Changing the precedence to $* > i$ in the Example 3.10 does not bring anything, and in both one-rule systems there exists only one choice of precedence in the recursive path ordering. Moreover, the recursive path ordering remains always too uniformly persistent, thus it orders critical pairs notoriously in the same direction. Therefore a change of its underlying precedence must make the closure chain non-operational.

4.2 Changing the ordering class

The second empirical method consist of changing completely the ordering class. It has to be applied when the previous one does not resolve the problem of divergence (ordering class is somehow monotone) or the completed canonical system does not correspond with the user's intentions. Although there exist some ordering hierarchies [48], no one ordering class is necessarily 'better' then another, because the termination of rewrite systems is undecidable in general [10, 28]. The aim of this method is to break a persistence inherent in certain orderings.

The proposed method attacks the conditions of the Definition 3.4 or of the Definition 3.12. The equivalence relation $\overset{*}{\longleftrightarrow}$ remains again the same. The user's contribution to this method is essential, even if the proposed ordering is implemented in the used rewrite rule laboratory, hence it is automatized. On the other hand, one can use also the method of Purdom [46], based on no ordering class, but on checking the set of rules for *looping*.

Example 4.4 Let us consider once more the *Associativity & Endomorphism* rewrite system. Suppose that f is a costly operator, and this rewrite system is proposed to optimize the term expressions with respect to f, so that the user can decrease the number of its uses to minimum. Therefore ordering the rule

$$f(x) + f(y) \quad \rightarrow \quad f(x + y)$$

forces to use the precedence $+ > f$ in RPO. This causes divergence of completion, as already described. Just as we look at the first generated rule

$$f(x + y) + z \quad \rightarrow \quad f(x) + (f(y) + z)$$

we can see that it does not fit our intentions with decreasing the number of f operators.

The recursive path ordering and also the recursive decomposition ordering are not suitable for ordering this rewrite system according to our conditions. As it was shown already in the Example 4.3, there exists a polynomial interpretation which allows to produce a finite canonical rewrite system by the polynomial ordering. The same finite canonical system can be obtained if using the Knuth–Bendix ordering in which we set $w(f) > 0$ [43]. There is also a possibility to use a *transformation ordering* [3, 8].

4.3 Separating closure chains

This method is applicable within the enlarged completion procedure in the sense of Huet and Hullot [27], which takes advantage from an explicit declaration of constructors. The method proposes the separation of a closure chain into a sequence of less complex equations. It can be described formally by the inference rule *Separate* (see Figure 2). As it can be seen from the inference rule *Separate*, the transformation is applicable only to closure chains which have a common root symbol in the term on both sides. The chaining process is supposed to disappear after this transformation. The equivalence relation $\overset{*}{\longleftrightarrow}$, generated by the original system, remains the same.

Example 4.5 Consider the one-rule backward crossed rewrite system

$$f(g(f(x))) \quad \rightarrow \quad f(h(x)) \tag{8}$$

ordered by RPO based on the precedence $f > h$ (or $g > h$). Trying to complete the system results in generation of the infinite family of rules

$$f(h^n(g(f(x)))) \quad \rightarrow \quad f(h^{n+1}(x))$$

Now, if we declare f to be a constructor, the rule (8) (considered as equation) is separated into a new equation, which can be ordered to the rule

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

choosing the precedence $g > h$. This new rule presents already a canonical system.

4.4 Dividing closure chains

The fourth empirical method is rooted in a process proposed as early as in the pioneering article [36], but cannot be considered as completely valid in comparison with standard completion, because the term algebra is changed (the signature is extended during the completion by a new function symbol). This method consists of dividing the underlying closure chain into two different parts, introducing a new operation symbol, and breaking this way the chaining process. It can be formally expressed by the inference rule *Divide* (see Figure 2). The method is especially devoted to inherently divergent and one-rule crossed systems.

Separate: $(E \cup \{f(s_1,\ldots,s_n) \simeq f(t_1,\ldots,t_n)\}; R) \vdash (E \cup \{s_i \simeq t_i \mid i = 1,\ldots,n\}; R)$ if f is a constructor

Divide: $(E \cup \{s \simeq t\}; R) \vdash (E \cup \{s \simeq f(x_1,\ldots,x_n),\ t \simeq f(x_1,\ldots,x_n)\}; R)$ where f is a new symbol, and $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \{x_1,\ldots,x_n\}$

Figure 2: Additional inference rules

Example 4.6 Consider once more the one-rule forward crossed system

$$f(g(f(x))) \quad \to \quad g(f(x)) \tag{9}$$

observed in Example 3.8 as inherently divergent. Let us divide the rule (9) into the (not yet reduced) rewrite system

$$\begin{aligned} f(g(f(x))) &\to h(x) \\ g(f(x)) &\to h(x) \end{aligned}$$

and enrich the precedence with $g > h$. If we complete the previous system, we get the canonical system

$$\begin{aligned} f(h(x)) &\to h(x) \\ g(f(x)) &\to h(x) \\ g(h(x)) &\to h(h(x)) \end{aligned}$$

Although it has some successful applications, this method is fragile. Applying it to the example of the idempotent semigroup or the one-rule system

$$(x\backslash y)\backslash z \quad \to \quad y\backslash(i(x)\backslash z)$$

does not bring the desired effect.

4.5 Enriching underspecified systems

The last proposed and probably the most powerful empirical method to avoid divergence consists of enriching given systems by new rewrite rules. It must be applied very carefully, otherwise the original rewrite system can be compromised and another finite canonical rewrite system is obtained, which does not correspond with the original rules. It can be accepted only as a practical method, because the underlying equational theory is changed by the added rule. This method attacks the conditions of the Fact 3.2, where the enriched system remains weakly divergent but not generally divergent any more. The new rule is added to interfere with interreduction, that causes a divergence object to disappear in some completion step.

The probably best way to proceed in this method is described in following steps:

1. remove the divergence basis rule(s) from the system R, producing R',
2. complete the residual set of rules R' to a finite canonical rewrite system R_2 (Recently, there was a method for proving inductive theorems proposed in [23], which does not require the underlying system to be confluent, not even on ground terms. Therefore this step may be skipped when using the just mentioned method.),
3. prove in R_2 by consistency [27, 32], or by inductive reducibility [31, 33], with a possible involvement of Fribourg's method [16] or even the more general Bachmair's one [2], an inductive theorem, derived from the structure of the generated infinite family of rules, and add it as a rule to the existing system R_2, producing R_2',
4. add the rules removed in step 1 to R_2', forming an enriched system R_e, and try to complete it.

Sometimes even a more-cycle iteration of the previous steps may lead to a desired solution with a finite canonical system.

The theoretical justification to extend the original rewrite system by inductive theorems presents the fact that the extended system R_e is *ground consistent* with the original system R [32].

Proposition 4.7 [32] *For all ground terms $s,t \in \mathcal{G}(\mathcal{F})$ the equivalence $s \overset{*}{\longleftrightarrow}_{R_e} t$ holds if, and only if, $s \overset{*}{\longleftrightarrow}_{R} t$.*

It means that rewriting ground terms with both the basic and the extended systems yields the same result. An open question remains the fact how to infer the new rule, to be proved as an inductive theorem of the original system, with which the extension is to be done.

When we add inductive theorems to the original system, we are interested in its initial model. If we want to prove an inductive theorem in an initial model, this model must be nonempty. This requires the existence of at least one constant. If there is none so we just introduce a dummy one.

$$
\begin{aligned}
-\ @\ x &\rightarrow x \\
x\ @\ - &\rightarrow x \\
(x\ @\ y)\ @\ z &\rightarrow x\ @\ (y\ @\ z) \\
flatten(-) &\rightarrow - \\
flatten([x]) &\rightarrow flatten(x) \\
flatten([x]\ @\ y) &\rightarrow flatten(x)\ @\ flatten(y) \qquad (10) \\
flatten(flatten(x)) &\rightarrow flatten(x) \qquad (11)
\end{aligned}
$$

ordered by RPO based on the precedence $flatten > @$ and a left-to-right status of @.

Figure 3: Rewrite system with the operation *flatten*

Example 4.8 [4] Let us consider a rewrite system specifying lists, with the operators $-$ for an empty list, [_] for an one-element list, and the append operator @. Let us define a *flatten* operator that transforms structured lists to lists consisting of only simple elements. The rewrite system is presented in Figure 3. If we try to complete this system, we get a divergent process. The rules (10) and (11) form a forward crossed rewrite system with the forward chain (10).

We remove the involution (11) and the residual system is already canonical. We are able now to prove the endomorphism rule

$$flatten(x\ @\ y) \rightarrow flatten(x)\ @\ flatten(y)$$

as its inductive theorem. We add it to the system and take back also the involution rule (11), forming the extended system. This extended system is completable to the finite canonical system

$$
\begin{aligned}
-\ @\ x &\rightarrow x \\
x\ @\ - &\rightarrow x \\
(x\ @\ y)\ @\ z &\rightarrow x\ @\ (y\ @\ z) \\
flatten(-) &\rightarrow - \\
flatten([x]) &\rightarrow flatten(x) \\
flatten(x\ @\ y) &\rightarrow flatten(x)\ @\ flatten(y) \\
flatten(flatten(x)) &\rightarrow flatten(x)
\end{aligned}
$$

The rule to be removed must be considered carefully. If we remove the wrong one, it can be even generated during completion of the residual system (dragon's head).

Example 4.9 Let us consider once more the rewrite system in Example 3.10 that could not be resolved by the previous empirical methods. If we remove the antimorphism rule

$$i(x * y) \rightarrow i(y) * i(x)$$

and try to complete the residual system, then the just removed rule will be generated from the other rules again.

The right way consists of removing the rules

$$
\begin{aligned}
(y * x) * i(x) &\rightarrow y \\
i(x) * (x * y) &\rightarrow y
\end{aligned}
$$

and the residual system

$$
\begin{aligned}
i(i(x)) &\rightarrow x \\
i(x * y) &\rightarrow i(y) * i(x)
\end{aligned}
$$

is already canonical. There is no constant in the system so we introduce a dummy one denoted by e. We are able now to prove the associativity rule

$$(x * y) * z \rightarrow x * (y * z)$$

as its inductive theorem. We add it to the system and take back also the removed rules, forming the extended system. If we try to complete it, the critical pair

$$x * i(x) = i(y) * y$$

is produced, which cannot be ordered because of the disjoint variables on the left- and right-hand side of the equation. It can be regarded as a request for a new operator. Therefore we divide the critical pair to the rules

$$
\begin{aligned}
x * i(x) &\rightarrow e \\
i(y) * y &\rightarrow e
\end{aligned}
$$

using the already introduced symbol e as the neutral element and extending the precedence with $* >$

[4] This example is constructed from a divergent inductive theorem proof.

e. After this action the rules are completable to the standard ten-rule canonical rewrite system for groups.

It must be admitted that this method has its limits, too, especially in the case of one-rule divergent systems. There is nothing to be removed; if the only rule is removed then virtually nothing can be proved as an inductive theorem in an empty system. Therefore one has to proceed in another way. One of the operators has to be decided as a non-constructor and then the rules to define it completely have to be added.

Example 4.10 Consider once more the extracted rule

$$(x\backslash y)\backslash z \quad \rightarrow \quad y\backslash(i(x)\backslash z)$$

from the group specification with left division. Let us decide the inverse operator i to be a non-constructor and therefore to add the rules[5]

$$\begin{aligned} i(x\backslash y) &\rightarrow y\backslash x \\ i(i(x)) &\rightarrow x \\ i(e) &\rightarrow e \end{aligned}$$

to define it completely. To make possible the ordering of the new rules in the proposed direction, the underlying precedence must be changed to $i \approx \backslash$. The enriched system is already canonical. If we add also the rule

$$x\backslash e \quad \rightarrow \quad i(x)$$

we get the finite canonical system

$$\begin{aligned} (x\backslash y)\backslash z &\rightarrow y\backslash(i(x)\backslash z) \\ i(x\backslash y) &\rightarrow y\backslash x \\ i(i(x)) &\rightarrow x \\ i(e) &\rightarrow e \\ x\backslash e &\rightarrow i(x) \\ e\backslash x &\rightarrow x \end{aligned}$$

5 Conclusion

We described two basic divergence patterns, the *forward* and *backward crossed rewrite systems* with *LR-* or *RL-persistence* respectively, by their definition from [20]. Further we presented in detail a number of examples to make the definitions more understandable and to support them as well. Although the divergence property is undecidable in general, the described divergence patterns seem to be sufficient enough to cover a large class of divergent systems, if not all of them. In the last section we introduced five empirical methods to avoid divergence of completion, which can be applied by an user during a session with a rewrite rule laboratory.

[5] It is necessary to introduce the neutral element e as a constant.

Acknowledgements

I would like to express my warmest thanks to Hélène Kirchner, Leo Bachmair, Jieh Hsiang, and Pierre Lescanne for comments and suggestions which helped to ameliorate the early versions of this paper.

References

[1] M.A. Ardis. Data abstraction transformations. Technical report TR-925, University of Maryland, Maryland (USA), 1980.

[2] L. Bachmair. Proof by consistency in equational theories. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science (LICS'88), Edinburgh (Scotland)*, pages 228–233, July 1988.

[3] L. Bachmair and N. Dershowitz. Commutation, transformation and termination. In J.H. Siekmann, editor, *Proceedings 8th International Conference on Automated Deduction (CADE'86), Oxford (England)*, volume 230 of *Lecture Notes in Computer Science*, pages 5–20. Springer-Verlag, July 1986.

[4] L. Bachmair and N. Dershowitz. Completion for rewriting modulo a congruence. *Theoretical Computer Science*, 67(2-3):173–202, October 1989.

[5] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proceedings 1st IEEE Symposium on Logic in Computer Science (LICS'86), Cambridge, (Massachusetts, USA)*, pages 346–357, June 1986.

[6] L. Bachmair, N. Dershowitz, and D.A. Plaisted. Completion without failure. In H. Aït-Kaci and M. Nivat, editors, *Proceedings of Resolution of Equations in Algebraic Structures, Lakeway, (Texas, USA); Volume 2: Rewrite Techniques*, pages 1–30. MCC Corporation & INRIA, Academic Press, 1989.

[7] F. Bellegarde. Rewriting systems on FP expressions to reduce the number of sequences yielded.

Science of Computer Programming, 6(1):11–34, January 1986.

[8] F. Bellegarde and P. Lescanne. Transformation ordering. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *Proceedings of CAAP '87, Pisa (Italy)*, volume 249 of *Lecture Notes in Computer Science*, pages 69–80. TAPSOFT '87, volume 1, Springer-Verlag, March 1987.

[9] A. BenCherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9(2):137–159, October 1987.

[10] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1 & 2):69–116, 1987. Special issue on Rewriting Techniques and Applications.

[11] N. Dershowitz. Completion and its applications. In H. Aït-Kaci and M. Nivat, editors, *Proceedings of Resolution of Equations in Algebraic Structures, Lakeway, (Texas, USA); Volume 2: Rewriting Techniques*, pages 31–86. MCC Corporation & INRIA, Academic Press, 1989.

[12] N. Dershowitz and J.-P. Jouannaud. Notations for rewriting. *Bulletin of the European Association for Theoretical Computer Science*, 43:162–172, February 1991.

[13] N. Dershowitz and L. Marcus. Existence and construction of rewrite systems. Technical Report ATR-82(8478)-3, Aerospace Corporation, El Segundo, California, 1982.

[14] N. Dershowitz, L. Marcus, and A. Tarlecki. Existence, uniqueness, and construction of rewrite systems. *SIAM Journal on Computing*, 17(4):629–639, August 1988.

[15] M. Fay. First-order unification in an equational theory. In S. Sickel, editor, *Proceedings of the 4th Workshop on Automated Deduction, Austin (Texas, USA)*, pages 161–167, February 1979.

[16] L. Fribourg. A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8(3):253–276, September 1989.

[17] R. Göbel. Ground confluence. In P. Lescanne, editor, *Proceedings 2nd Conference on Rewriting Techniques and Applications (RTA'87), Bordeaux (France)*, volume 256 of *Lecture Notes in Computer Science*, pages 156–167. Springer-Verlag, May 1987.

[18] J.V. Guttag, D. Kapur, and D.R. Musser. On proving uniform termination and restricted termination of rewrite systems. *SIAM Journal on Computing*, 12(1):189–214, February 1983.

[19] M. Hermann. Chain properties of rule closures. In B. Monien and R. Cori, editors, *Proceedings 6th Symposium on Theoretical Aspects of Computer Science (STACS'89), Paderborn (Germany)*, volume 349 of *Lecture Notes in Computer Science*, pages 339–347. Springer-Verlag, February 1989.

[20] M. Hermann. Crossed term rewriting systems. Research report 89-R-003, Centre de Recherche en Informatique de Nancy, 1989. Included in [?].

[21] M. Hermann and I. Prívara. On nontermination of Knuth-Bendix algorithm. Research report VUSEI-AR-OPS-3/85, Institute of Socio-Economic Information and Automation in Management, Bratislava, Czechoslovakia, November 1985.

[22] M. Hermann and I. Prívara. On nontermination of Knuth-Bendix algorithm. In L. Kott, editor, *Proceedings 13th ICALP Conference, Rennes (France)*, volume 226 of *Lecture Notes in Computer Science*, pages 146–156. Springer-Verlag, July 1986.

[23] D. Hofbauer and R.-D. Kutsche. Proving inductive theorems based on term rewriting systems. In J. Grabowski, P. Lescanne, and W. Wechler, editors, *Proceedings of the International Workshop on Algebraic and Logic Programming, Gaussig (Germany)*, volume 343 of *Lecture Notes in Computer Science*, pages 180–190. Springer-Verlag, November 1988.

[24] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In T. Ottmann, editor, *Proceedings of the 14th ICALP, Karlsruhe (Germany)*, volume 267 of *Lecture Notes in Computer Science*, pages 54–71. Springer-Verlag, July 1987.

[25] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27(4):797–821, 1980.

[26] G. Huet. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal of Computer and System Science*, 23(1):11–21, August 1981. Also as: Rapport 25, INRIA, 1980.

[27] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Science*, 25(2):239–266, October 1982.

[28] G. Huet and D.S. Lankford. On the uniform halting problem for term rewriting systems. Rapport de recherche 283, Institut de Recherche en Informatique et en Automatique, Le Chesnay, France, 1978.

[29] J.-M. Hullot. Canonical forms and unification. In W. Bibel and R. Kowalski, editors, *Proceedings 5th International Conference on Automated Deduction (CADE'80), Les Arcs (France)*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer-Verlag, July 1980.

[30] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, November 1986.

[31] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82:1–33, 1989.

[32] D. Kapur and D.R. Musser. Proof by consistency. *Artificial Intelligence*, 31(2):125–157, February 1987.

[33] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24(4):395–415, August 1987.

[34] C. Kirchner and H. Kirchner. Résolution d'équations dans les algèbres libres et les variétés équationelles d'algèbres. Master's thesis, Université de Nancy I, 1982.

[35] H. Kirchner. Schematization of infinite sets of rewrite rules generated by divergent completion process. *Theoretical Computer Science*, 67(2-3):303–332, 1989.

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

[37] D.S. Lankford. Canonical inference. Research report ATP-32, Department of Mathematics and Computer Science, University of Texas, Austin, Texas (USA), December 1975.

[38] D.S. Lankford and D.R. Musser. A finite termination criterion. Unpublished draft, Information Sciences Institute, University of Southern California, Marina-del-Rey, CA, 1978.

[39] P. Lescanne. Computer experiments with the REVE term rewriting system generator. In *Proceedings of the 10th ACM POPL Symposium, Austin, (Texas, USA)*, pages 99–108, January 1983.

[40] P. Lescanne. Divergence of the Knuth-Bendix completion procedure and termination orderings. *Bulletin of the European Association for Theoretical Computer Science*, 30:80–83, October 1986.

[41] P. Lescanne. Completion procedures as transition rules + control. In J. Díaz and F. Orejas, editors, *Proceedings of TAPSOFT '89, Volume 1: CAAP '89; Barcelona (Spain)*, volume 351 of *Lecture Notes in Computer Science*, pages 28–41. Springer-Verlag, March 1989.

[42] P. Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *Journal of Automated Reasoning*, 6:39–49, 1990.

[43] U. Martin. How to choose the weights in the Knuth-Bendix ordering. In P. Lescanne, editor, *Proceedings 2nd Conference on Rewriting Techniques and Applications (RTA'87), Bordeaux (France)*, volume 256 of *Lecture Notes in Computer Science*, pages 42–53. Springer-Verlag, May 1987.

[44] C.-T. Mong and P.W. Purdom. Divergence in the completion of rewriting systems. Technical report, Dept. of Comp. Science, Indiana University, 1987.

[45] P. Narendran and J. Stillman. It is undecidable whether the Knuth-Bendix completion procedure generates a crossed pair. In B. Monien and R. Cori, editors, *Proceedings 6th Symposium on Theoretical Aspects of Computer Science (STACS'89), Paderborn (Germany)*, volume 349 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, February 1989.

[46] P.W. Purdom. Detecting looping simplifications. In P. Lescanne, editor, *Proceedings 2nd Conference on Rewriting Techniques and Applications (RTA'87), Bordeaux (France)*, volume 256 of *Lecture Notes in Computer Science*, pages 54–61. Springer-Verlag, May 1987.

[47] B. K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the Association for Computing Machinery*, 20(1):160–187, January 1973.

[48] M. Rusinowitch. Path of subterms ordering and recursive decomposition ordering revisited. *Journal of Symbolic Computation*, 3(1 & 2):117–131, 1987.

[49] J. Siekmann and P. Szabó. A Nœtherian and confluent rewrite system for idempotent semigroups. *Semigroup Forum*, 25:83–110, 1982.

[50] J.R. Slagle. Automated theorem-proving for theories with simplifiers, commutativity, and associativity. *Journal of the Association for Computing Machinery*, 21:622–642, 1974.