Dependency Pairs for Rewriting with Non-Free Constructors

Stephan Falke and Deepak Kapur

Department of Computer Science
University of New Mexico
Albuquerque, NM 87131, USA
{spf|kapur}@cs.unm.edu

Abstract. A method based on dependency pairs for showing termination of functional programs on data structures generated by constructors with relations is proposed. A functional program is specified as an equational rewrite system, where the rewrite system specifies the program and the equations express the relations on the constructors that generate the data structures. Unlike previous approaches, relations on constructors can be collapsing, including idempotency and identity relations. Relations among constructors may be partitioned into two parts: (i) equations that cannot be oriented into terminating rewrite rules, and (ii) equations that can be oriented as terminating rewrite rules, in which case an equivalent convergent system for them is generated. The dependency pair method is extended to normalized rewriting, where constructor-terms in the redex are normalized first. The method has been applied to several examples, including the Calculus of Communicating Systems and the Propositional Sequent Calculus. Various refinements, such as dependency graphs, narrowing, etc., which increase the power of the dependency pair method, are presented for normalized rewriting.

1 Introduction

Algorithms in a functional programming style can be specified elegantly within the framework of term rewrite systems. This is the approach taken by ELAN [14], Maude [3], and theorem provers such as RRL [13], where a function definition is given as a terminating rewrite system on data structures generated using constructors. We will follow that approach in this paper as well, i.e., we assume that a functional program is represented in the form of a term rewrite system. While automated termination methods (a collection of recent papers on termination is [7]) work well for establishing termination of rewrite systems defined on data structures generated using free constructors (such as natural numbers, lists, trees, etc.), they do not extend well to cases where the constructors of the data structures are related. For example, the data structure of finite sets has set union "$\cup$" as a constructor which is not only associative (A) and commutative (C), but also idempotent ($x \cup x \approx x$) and has the other constructor, the empty set "$\emptyset$", as an identity ($x \cup \emptyset \approx x$). Methods for showing termination of AC-rewrite systems

based on recursive path orderings and dependency pairs have been developed [12, 19, 15, 17]. In [6], the dependency pair method was generalized to equational rewriting with the restriction that equations need to be non-collapsing and have identical unique variables (i.e., each variable occurs exactly once on each side). So termination of rewrite systems defined on a data structure such as finite sets cannot be established using any of these previous approaches.

In this paper, we extend the dependency pair method in order to establish termination of equational rewrite systems in which equations may be collapsing. However, if collapsing relations are included in the equational system, then equational rewriting does not necessarily terminate in cases where this would intuitively be expected. The key idea to handle this problem is to partition the equational system relating the constructors of the data structures into two parts: (i) equations which can be oriented and completed in a convergent subsystem, and (ii) the remaining equations. Rewriting in an equational rewrite system is then done using *normalized rewriting*[1] à la Marché [16]. Before rewriting a term, the constructor-terms in the redex are first *normalized*, and rewriting is thus performed on normalized terms. This approach towards equational rewriting has the major advantage that algorithms can be specified elegantly since, in the specification, the constructor-terms can be assumed to be in normalized form.

Example 1. As an example, we consider the data structure of integers that are built using the constructors $\mathsf{0}$, s (successor), and p (predecessor). We have the relations $\mathcal{E} = \{\mathsf{p}(\mathsf{s}(u)) \approx \mathsf{s}(\mathsf{p}(u)), \mathsf{p}(\mathsf{s}(u)) \approx u\}$ between these constructors. Defining a simple predicate pos, which checks whether an integer is strictly positive, is highly nontrivial with ordinary equational rewriting. Using the approach of normalized rewriting, we can split $\mathcal{E}$ into $\mathcal{E}_1 = \{\mathsf{p}(\mathsf{s}(u)) \approx \mathsf{s}(\mathsf{p}(u))\}$ and $\mathcal{E}_2 = \{\mathsf{p}(\mathsf{s}(u)) \approx u\}$, where $\mathcal{E}_2$ can be oriented into $\mathcal{S} = \{\mathsf{p}(\mathsf{s}(u)) \to u\}$, which is convergent modulo $\mathcal{E}_1$. Using normalized rewriting, it is now straightforward to define pos by the rewrite rules $\mathcal{R} = \{\mathsf{pos}(\mathsf{0}) \to \mathsf{false}, \mathsf{pos}(\mathsf{s}(x)) \to \mathsf{true}, \mathsf{pos}(\mathsf{p}(x)) \to \mathsf{false}\}$. The predicate pos indeed correctly determines whether its argument is strictly positive for constructor ground terms since normalizing produces a term of the form $\mathsf{0}$, $\mathsf{s}^i(\mathsf{0})$, or $\mathsf{p}^i(\mathsf{0})$ for some $i > 0$. In contrast, evaluation with ordinary equational rewriting using $\mathcal{R}$ and $\mathcal{E}$ does not yield the desired result since, for example, $\mathsf{pos}(\mathsf{s}(\mathsf{p}(\mathsf{p}(\mathsf{0})))))$ can be rewritten to true, although $\mathsf{s}(\mathsf{p}(\mathsf{p}(\mathsf{0})))$ represents the negative integer -1. ◊

The results in this paper rely on the property that no equations involving defined symbols (i.e., outermost symbols of left sides of rules in the rewrite system) are allowed. Firstly, this allows us to permit collapsing equations as well, which are not permitted in [6]. Note that orientable equations need not be treated as rewrite rules. Instead, our method provides a uniform framework for termination analysis in both cases. Secondly, even though we allow collapsing equations, the proposed approach is conceptually simpler than the one of [6] since we do not need to consider instantiations of rewrite rules. These instantiations

[1] Strictly speaking this should be called *normalized equational rewriting*. We are following Marché's convention of calling it *normalized rewriting*.

are needed for correctness of the method in [6] and can cause problems since they may generate a huge number of rules. Thirdly, using normalized rewriting also enables us to consider equations that do not have identical unique variables, which is another severe restriction of the method in [6]. We allow equations that do not have identical unique variables as long as they can be oriented into rewrite rules. All of these features significantly increase the scope of applicability of the dependency pair approach. The proposed method for showing termination of normalized rewriting has not been implemented yet, but we believe that it can be easily incorporated into a termination tool implementing the dependency pair framework such as AProVE [8].

The paper is organized as follows. In Section 2, we review equational rewriting and, in particular, the distinction between rewriting modulo $\mathcal{E}$ and $\mathcal{E}$-extended rewriting. Normalized rewriting is then discussed in Section 3. It is argued that algorithms can be specified elegantly and in a natural way using $\mathcal{S}$-normalized $\mathcal{E}$-extended rewriting, in which constructor-terms in the redex are first normalized using $\mathcal{S}$ modulo $\mathcal{E}$, where $\mathcal{S}$ is a convergent system capturing (some of the) relations on constructors. In Section 4, the dependency pair method is extended to normalized rewriting. It is shown that if there are no infinite chains of dependency pairs, then $\mathcal{S}$-normalized $\mathcal{E}$-extended rewriting is terminating. A first method for automatically showing that there are no such chains is presented. It uses so-called *reduction pairs*, which are widely used in the dependency pair approach. Reduction pairs have the advantage that they do not need to be monotonic. In Section 5, we extend the recently formulated dependency pair framework to the context of normalized rewriting. This allows flexibility in establishing the termination of complex rewrite systems including the sequent calculus, CCS, etc. In Section 6, dependency pair (DP) processors are discussed. A DP processor transforms a DP problem into a finite set of simpler DP problems in such a way that termination of the simpler DP problems implies termination of the original DP problem. The DP processors presented make use of dependency graphs, reduction pairs, removal

The method has been used on interesting and nontrivial examples, many of which cannot be handled otherwise. Detailed discussion of these examples is contained in Appendix A.

2 Equational Rewriting

We assume familiarity with the concepts of term rewriting [2] and fix some notation in the following. For a finite signature $\mathcal{F}$ and an infinite set $\mathcal{V}$ of variables the set of *terms* over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. We often write s^* to denote a tuple of terms $s_1,\ldots,s_n$ for some $n \geq 0$. The set of function symbols occurring in the term t is denoted by $\mathcal{F}(t)$. Similarly, $\mathcal{V}(t)$ denotes the variables occurring in t. This naturally extends to sets of terms, pairs of terms, and sets of pairs of terms. The outermost function symbol of a term t is denoted by $\mathrm{root}(t)$.

An *equational system* (ES) is a finite set $\mathcal{E} = \{u_1 \approx v_1,\ldots,u_m \approx v_m\}$ of equations, and a *term rewrite system* (TRS) is a finite set of oriented equations

(called *rules*) $\mathcal{R} = \{l_1 \to r_1, \ldots, l_n \to r_n\}$, where $l_i \notin \mathcal{V}$ and $\mathcal{V}(r_i) \subseteq \mathcal{V}(l_i)$ for all $1 \leq i \leq n$. The *defined symbols* of a TRS $\mathcal{R}$ are the symbols occurring as root(l) for some rule $l \to r$ in $\mathcal{R}$. The set of defined symbols of $\mathcal{R}$ is denoted by $\mathcal{D}(\mathcal{R})$. The remaining symbols of $\mathcal{F}(\mathcal{R})$ are *constructors*.

For an ES $\mathcal{E}$ (resp. TRS $\mathcal{R}$), we write $s \to_{\mathcal{E}} t$ (resp. $s \to_{\mathcal{R}} t$) iff there exist an equation $l \approx r$ in $\mathcal{E}$ (resp. rule $l \to r$ in $\mathcal{R}$), a substitution σ, and a position p in s such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. The symmetric closure of $\to_{\mathcal{E}}$ is denoted by $\vdash\!\dashv_{\mathcal{E}}$, and the reflexive-transitive closure of $\vdash\!\dashv_{\mathcal{E}}$ is denoted by $\sim_{\mathcal{E}}$.

Definition 2 (Rewriting Modulo $\mathcal{E}$). *Let $\mathcal{R}$ be a TRS and let $\mathcal{E}$ be an ES. The term s* rewrites modulo *$\mathcal{E}$ to the term t, written $s \to_{\mathcal{R}/\mathcal{E}} t$, iff $s' \to_{\mathcal{R}} t'$ for some terms $s' \sim_{\mathcal{E}} s$ and $t' \sim_{\mathcal{E}} t$.*

Thus, in order to determine whether a term s is reducible w.r.t. $\to_{\mathcal{R}/\mathcal{E}}$, a term that is equivalent to s up to $\sim_{\mathcal{E}}$ and reducible by $\to_{\mathcal{R}}$ has to be found. If the $\mathcal{E}$-equivalence classes are impractically large or even infinite this is not feasible. To avoid this problem, virtually all implementations (e.g., ELAN [14] and Maude [3]) use $\mathcal{E}$-extended rewriting, which builds the equivalence up to $\sim_{\mathcal{E}}$ into the matching process.

Definition 3 ($\mathcal{E}$-Extended Rewriting). *Let $\mathcal{R}$ be a TRS and let $\mathcal{E}$ be an ES. The term s* rewrites *$\mathcal{E}$-extended to the term t, written $s \to_{\mathcal{E}\setminus\mathcal{R}} t$, iff $s|_p \sim_{\mathcal{E}} l\sigma$ and $t = s[r\sigma]_p$ for some rule $l \to r$ in $\mathcal{R}$, some substitution σ, and some position p in s.*

An equation $u \approx v$ is *collapsing* iff $u \in \mathcal{V}$ or $v \in \mathcal{V}$, and an ES is said to be collapsing iff it contains a collapsing equation.

Definition 4 (Identical Unique Variables). *Let $\mathcal{E}$ be an ES. Then $\mathcal{E}$ has* identical unique variables *($\mathcal{E}$ is* i.u.v.*) iff u, v are linear and $\mathcal{V}(u) = \mathcal{V}(v)$ for all equations $u \approx v$ in $\mathcal{E}$.*

In this paper we restrict ourselves to i.u.v. ESs. Note, however, that we do allow collapsing equations, in contrast to [6]. Two important cases of i.u.v. ESs are the following, which state that a binary function symbol f is associative and commutative, possibly with a unit 0.

$$\begin{aligned} \mathsf{AC}_f &= \{f(u, f(v, w)) \approx f(f(u, v), w), f(u, v) \approx f(v, u)\} \\ \mathsf{ACU}_{f,0} &= \mathsf{AC}_f \cup \{f(u, 0) \approx u\} \end{aligned}$$

Note that equations like $f(u, u) \approx u$, $f(u, 0) \approx 0$ and $f(u, u) \approx 0$ are not allowed since they are nonlinear and/or not variable-preserving.

The reason for the restriction to i.u.v. ESs is the following lemma, which does not hold true if $\mathcal{E}$ is not i.u.v. Intuitively, it states that subterms t with root$(t) \notin \mathcal{F}(\mathcal{E})$ persist in terms that are equivalent up to $\sim_{\mathcal{E}}$.

Lemma 5. *Let $\mathcal{E}$ be an i.u.v. ES and let $C[f(s^*)] \sim_{\mathcal{E}} t$ for some context C, some term $f(s^*)$ with $f \notin \mathcal{F}(\mathcal{E})$, and some term t. Then $t = C'[f(s'^*)]$ for some context $C' \sim_{\mathcal{E}} C$ and some term $f(s'^*)$ such that $s^* \sim_{\mathcal{E}} s'^*$.*

Proof. Let $C[f(s^*)] \sim_{\mathcal{E}} t$, i.e., there exist terms $t_0, \ldots, t_n$ with $n \geq 0$ such that $C[f(s^*)] = t_0 \vdash\!\dashv_{\mathcal{E}} t_1 \vdash\!\dashv_{\mathcal{E}} \ldots \vdash\!\dashv_{\mathcal{E}} t_n = t$. The claim is proved by induction on n. If $n = 0$ then $C[f(s^*)] = t$ and the claim is obvious.

If $n > 0$, the inductive hypothesis implies $t_{n-1} = C''[f(s''^*)]$ with $C'' \sim_{\mathcal{E}} C$ and $s''^* \sim_{\mathcal{E}} s^*$. Since $t_{n-1} \vdash\!\dashv_{\mathcal{E}} t_n$, there exists an equation $u \approx v$ or $v \approx u$ in $\mathcal{E}$ such that $t_{n-1}|_p = u\sigma$ and $t_n = t_{n-1}[v\sigma]_p$ for some position p and some substitution σ. Let q be the position with $t_{n-1}|_q = f(s''^*)$, i.e., $C''|_q = \Box$. We perform a case analysis on the relationship between the positions p and q.

Case 1: $p \perp q$. Then, $t_n = t_{n-1}[v\sigma]_p = (C''[f(s''^*)])[v\sigma]_p = (C''[v\sigma]_p)[f(s''^*)]$ with $C''[v\sigma]_p \sim_{\mathcal{E}} C''[u\sigma]_p = C''$.

Case 2: $p = qq'$ *for some position* $q' \neq \varepsilon$. In this case, $t_n = t_{n-1}[v\sigma]_p = (C''[f(s''^*)])[v\sigma]_{qq'} = C''[f(s''^*)[v\sigma]_{q'}]$. Since $q' \neq \varepsilon$, we can write $q' = iq''$ for some i and some position q''. Then $s'_j = s''_j$ if $i \neq j$ and $s'_i = s''_i[v\sigma]_{q''} \sim_{\mathcal{E}} s''_i[u\sigma]_{q''} = s''$, i.e., $s'^* \sim_{\mathcal{E}} s''^*$.

Case 3: $q = pp'$ *for some position* p' *(possibly* $p' = \varepsilon$*).* Since $f \notin \mathcal{F}(\mathcal{E})$, we can write $p' = p'_1 p'_2$ such that $u|_{p'_1}$ is a variable x and $x\sigma|_{p'_2} = f(s''^*)$. Since the equation $u \approx v$ (or $v \approx u$) is i.u.v., there exists an unique position p''_1 in v such that $v|_{p''_1} = x$. This implies $v\sigma|_{p''_1 p'_2} = x\sigma|_{p'_2} = f(s''^*)$. Define the substitution σ' by $y\sigma' = y\sigma$ for $y \neq x$ and $x\sigma' = x\sigma[\Box]_{p'_2}$. Let $C' = (t_{n-1}[v\sigma]_p)[\Box]_{pp''_1 p'_2} = t_{n-1}[v\sigma[\Box]_{p''_1 p'_2}]_p = t_{n-1}[v\sigma']_p \sim_{\mathcal{E}} t_{n-1}[u\sigma']_p = C''$. Thus, $t_n = t_{n-1}[v\sigma]_p = C'[f(s''^*)]$ and the claim follows. □

3 $\mathcal{S}$-Normalized Rewriting

In this paper we are concerned with proving termination of rewriting with a TRS $\mathcal{R}$ and an i.u.v. ES $\mathcal{E}$, where $\mathcal{F}(\mathcal{E})$ does not contain any defined symbols from $\mathcal{R}$. Thus, $\mathcal{E}$ is an ES of constructors which specifies some properties of data structures that the functions defined by $\mathcal{R}$ operate on. The first problem that is encountered is that $\mathcal{E}$-extended rewriting does not terminate in many cases where $\mathcal{E}$ is an i.u.v. ES of constructors that is collapsing.

Example 6. Let $\mathcal{E} = \mathsf{ACU}_{+,0}$ and $\mathcal{R} = \{(x + y) \cdot z \to x \cdot z + y \cdot z\}$. Then $\to_{\mathcal{E}\backslash\mathcal{R}}$ is not terminating since

$$\begin{aligned} \mathsf{0} \cdot z \sim_{\mathcal{E}} (\mathsf{0} + \mathsf{0}) \cdot z \quad &\to_{\mathcal{R}} \mathsf{0} \cdot z + \mathsf{0} \cdot z \\ \sim_{\mathcal{E}} (\mathsf{0} + \mathsf{0}) \cdot z + \mathsf{0} \cdot z &\to_{\mathcal{R}} \ldots \end{aligned}$$

is an infinite $\to_{\mathcal{E}\backslash\mathcal{R}}$ reduction. ◊

To overcome problems like this, the notion of *normalized rewriting* was introduced by Marché in [16]. In the following we use a slight variation of this notion. The idea is to split an ES $\mathcal{E}$, which does not necessarily need to be i.u.v., into ESs $\mathcal{E}_1$ and $\mathcal{E}_2$ such that $\mathcal{E}_1$ is i.u.v. and $\mathcal{E}_2$ contains the remaining equations.

Then, $\mathcal{E}_2$ is completed[2] into a TRS $\mathcal{S}$ that is convergent modulo $\mathcal{E}_1$. Here, the TRS $\mathcal{S}$ is convergent modulo $\mathcal{E}_1$ iff $\to_{\mathcal{S}/\mathcal{E}_1}$ is terminating and confluent modulo $\mathcal{E}_1$, i.e., whenever $t \to^*_{\mathcal{S}/\mathcal{E}_1} t_1$ and $t \to^*_{\mathcal{S}/\mathcal{E}_1} t_2$ for some terms t, t_1, t_2, then there exist terms s_1, s_2 with $s_1 \sim_{\mathcal{E}_1} s_2$ such that $t_1 \to^*_{\mathcal{S}/\mathcal{E}_1} s_1$ and $t_2 \to^*_{\mathcal{S}/\mathcal{E}_1} s_2$ (thus, $\leftarrow^*_{\mathcal{S}/\mathcal{E}_1} \circ \to^*_{\mathcal{S}/\mathcal{E}_1} \subseteq \to^*_{\mathcal{S}/\mathcal{E}_1} \circ \sim_{\mathcal{E}_1} \circ \leftarrow^*_{\mathcal{S}/\mathcal{E}_1}$)[3]. Note that $t \sim_{\mathcal{E}_1} t'$ implies $t{\downarrow_{\mathcal{S}/\mathcal{E}_1}} \sim_{\mathcal{E}_1} t'{\downarrow_{\mathcal{S}/\mathcal{E}_1}}$ if $\mathcal{S}$ is convergent modulo $\mathcal{E}_1$, where $t{\downarrow_{\mathcal{S}/\mathcal{E}_1}}$ denotes the normal form of t w.r.t. $\to_{\mathcal{S}/\mathcal{E}_1}$. This is also written $t \to^!_{\mathcal{S}/\mathcal{E}_1} t{\downarrow_{\mathcal{S}/\mathcal{E}_1}}$.

In the following table we list how some commonly occurring, not necessarily i.u.v., ESs $\mathcal{E}$ can be split into an i.u.v. ES $\mathcal{E}_1$ and a TRS $\mathcal{S}$ that is convergent modulo $\mathcal{E}_1$.

$\mathcal{E}$	$\mathcal{E}_1$	$\mathcal{S}$
AC_f	AC_f	$\emptyset$
$\mathsf{ACU}_{f,0}$	AC_f	$\{f(u,0) \to u\}$
$\mathsf{ACI}_f = \mathsf{AC}_f \cup \{f(u,u) \approx u\}$	AC_f	$\{f(u,u) \to u\}$
$\mathsf{ACUI}_{f,0} = \mathsf{ACI}_f \cup \mathsf{ACU}_{f,0}$	AC_f	$\{f(u,0) \to u, f(u,u) \to u\}$
$\mathsf{AC0}_{f,0} = \mathsf{AC}_f \cup \{f(u,0) \approx 0\}$	AC_f	$\{f(u,0) \to 0\}$
$\mathsf{ACN}_{f,0} = \mathsf{AC}_f \cup \{f(u,u) \approx 0\}$	AC_f	$\{f(u,u) \to 0\}$

As mentioned in Section 2, rewriting with $\to_{\mathcal{E}\setminus\mathcal{R}}$ tends to be infeasible and rewriting with $\to_{\mathcal{R}/\mathcal{E}}$ should be used instead. Now $\to_{\mathcal{R}/\mathcal{E}}$ is clearly contained in $\to_{\mathcal{E}\setminus\mathcal{R}}$, but the converse is not true in general.[4] For a certain class of TRSs, however, $\to_{\mathcal{E}\setminus\mathcal{R}}$ and $\to_{\mathcal{R}/\mathcal{E}}$ are essentially the same.

Definition 7 (Complete TRSs). *Let $\mathcal{R}$ be a TRS and let $\mathcal{E}$ be an i.u.v. ES. Then $\mathcal{R}$* is complete modulo $\mathcal{E}$ *iff $\to_{\mathcal{R}/\mathcal{E}} \subseteq \to_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}}$, i.e., whenever $s \to_{\mathcal{R}/\mathcal{E}} t$, then there exists a $t' \sim_{\mathcal{E}} t$ such that $s \to_{\mathcal{E}\setminus\mathcal{R}} t'$.*

In the following we assume that $\mathcal{S}$ is complete modulo $\mathcal{E}$. For $\mathcal{S}$ to satisfy Definition 7, an extension using $Ext_{\mathcal{E}}(\mathcal{S})$ might be needed, see [18, 6]. In case $\mathcal{E} = \bigcup_{f \in \mathcal{G}} \mathsf{AC}_f$ for some set $\mathcal{G}$ of binary functions the extension can be achieved by adding rules $f(l,z) \to f(r,z)$ for all rules $l \to r \in \mathcal{S}$ with $\mathrm{root}(l) = f \in \mathcal{G}$, where z is a fresh variable. If the rule $l \to r$ AC-matches the extended rule $f(l,z) \to f(r,z)$, then the extended rule does not need to be added, see [4, Lemma 6.3]. For example, the extension of $f(u,u) \to u$ for an AC-symbol f is $f(f(u,u),v) \to f(u,v)$ for a fresh variable v. Similarly, the extension of $f(u,0) \to u$ is $f(f(u,0),v) \to f(u,v)$ for a fresh variable v, but this extension does not need to be added since the rule $f(u,0) \to u$ AC-matches it.

Complete TRSs enjoy the following important property.

Lemma 8. *Let $\mathcal{R}$ be a TRS an let $\mathcal{E}$ be an i.u.v. ES such that $\mathcal{R}$ is complete modulo $\mathcal{E}$. If $s \to_{\mathcal{E}\setminus\mathcal{R}} t$ and $s' \sim_{\mathcal{E}} s$, then there exists a $t' \sim_{\mathcal{E}} t$ such that $s' \to_{\mathcal{E}\setminus\mathcal{R}} t'$.*

[2] In general, this requires $\mathcal{E}_1$-unification.

[3] Here, $\circ$ denotes composition of relations, i.e., $t \bowtie_1 \circ \bowtie_2 q$ iff $t \bowtie_1 s \bowtie_2 q$ for some s.

[4] Consider $\mathcal{E} = \{\mathsf{f(a)} \approx \mathsf{f(b)}\}$ and $\mathcal{R} = \{\mathsf{a} \to \mathsf{c}\}$. Then $\mathsf{f(b)} \to_{\mathcal{R}/\mathcal{E}} \mathsf{f(c)}$, but $\mathsf{f(b)}$ is not reducible by $\to_{\mathcal{E}\setminus\mathcal{R}}$.

Proof. Let $s \to_{\mathcal{E}\setminus\mathcal{R}} t$. Then $s \to_{\mathcal{R}/\mathcal{E}} t$ as well, and since $s' \sim_{\mathcal{E}} s$ we get $s' \to_{\mathcal{R}/\mathcal{E}} t$. Since $\mathcal{R}$ is complete modulo $\mathcal{E}$ we obtain $s' \to_{\mathcal{E}\setminus\mathcal{R}} t'$ for some $t' \sim_{\mathcal{E}} t$. □

Since we are only interested in rewriting with non-free constructors, neither $\mathcal{S}$ nor $\mathcal{E}$ contains any defined symbols from $\mathcal{R}$. We thus have the following case.

Definition 9 (Equational Systems). *An* equational system $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ *consists of two TRSs $\mathcal{R}$ and $\mathcal{S}$ and an i.u.v. ES $\mathcal{E}$ such that $\mathcal{S}$ is complete and convergent modulo $\mathcal{E}$ and $\mathcal{F}(\mathcal{E}) \cap \mathcal{D}(\mathcal{R}) = \mathcal{F}(\mathcal{S}) \cap \mathcal{D}(\mathcal{R}) = \emptyset$.*

Now *$\mathcal{S}$-normalized $\mathcal{E}$-extended rewriting* is done with an equational system $(\mathcal{R}, \mathcal{S}, \mathcal{E})$, and intuitively the arguments to a defined function f need to be normalized with $\to_{\mathcal{S}/\mathcal{E}}$ before an f-rule from $\mathcal{R}$ may be applied.

Definition 10 ($\mathcal{S}$-Normalized $\mathcal{E}$-Extended Rewriting). *Let $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ be an equational system. The term t* rewrites *$\mathcal{S}$-normalized $\mathcal{E}$-extended to the term q, written $t \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} q$, iff $t|_p\downarrow_{\mathcal{E}\setminus\mathcal{S}} \sim_{\mathcal{E}} l\sigma$ and $q = t[r\sigma]_p$ for some rule $l \to r$ in $\mathcal{R}$, some position p with $\mathrm{root}(t|_p) = \mathrm{root}(l)$ in t, and some substitution σ.*

Our notion of $\mathcal{S}$-normalized rewriting differs from [16] in that we only normalize the redex w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$ before the rule from $\mathcal{R}$ is applied, while in [16] the whole term needs to be normalized w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$.

Example 11. Continuing Example 6, we can split $\mathcal{E}$ into $\mathcal{E}_1 = \mathsf{AC}_+$ and $\mathcal{E}_2 = \{u + \mathsf{0} \approx u\}$. Then $\mathcal{E}_2$ can be completed into the TRS $\mathcal{S} = \{u + \mathsf{0} \to u\}$, which is convergent modulo $\mathcal{E}_1$ and does not need to be extended. Thus, $(\mathcal{R}, \mathcal{S}, \mathcal{E}_1)$ is an equational system and the infinite reduction from Example 6 is not possible anymore if $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1\setminus\mathcal{R}}$ is used since $\mathsf{0} \cdot z$ is in normal form w.r.t. $\to_{\mathcal{E}_1\setminus\mathcal{S}}$ and no rule of $\mathcal{R}$ applies. Also, the infinite reduction starting with $(\mathsf{0} + \mathsf{0}) \cdot z$ is not possible anymore since $(\mathsf{0} + \mathsf{0}) \cdot z$ would need to be normalized w.r.t. $\to_{\mathcal{E}_1\setminus\mathcal{S}}$ first, which again gives $\mathsf{0} \cdot z$. ◊

Apart from resulting in a terminating rewrite process in cases where $\to_{\mathcal{E}\setminus\mathcal{R}}$ is not terminating, $\mathcal{S}$-normalized rewriting also has the advantage of giving rise to "natural" function definitions since we can assume that the arguments to a function are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$ before the function is evaluated. This would not be true if $\to_{\mathcal{E}\setminus\mathcal{R}\cup\mathcal{S}}$ is used instead, as already shown in Example 1.

Example 12. Let us consider sets that are built using the empty set $\emptyset$, singleton sets $\langle\cdot\rangle$, and set union $\cup$. Then $\mathcal{E} = \mathsf{ACUI}_{\cup,\emptyset}$ specifies the expected properties of sets. This can be split into $\mathcal{E}_1 = \mathsf{AC}_\cup$ and $\mathcal{S} = \{u \cup \emptyset \to u, u \cup u \to u, (u \cup u) \cup v \to u \cup v\}$, where $\mathcal{S}$ is convergent modulo $\mathcal{E}_1$ and the third rule is the extension of the second rule. Now we consider sets of natural numbers, and we want to sum up the elements of a set. Using $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1\setminus\mathcal{R}}$, this can easily be done by letting $\mathcal{R} = \{\mathsf{sum}(\emptyset) \to \mathsf{0}, \mathsf{sum}(\langle m\rangle) \to m, \mathsf{sum}(x \cup y) \to \mathsf{sum}(x) + \mathsf{sum}(y), x + \mathsf{0} \to x, x + \mathsf{s}(y) \to \mathsf{s}(x + y)\}$. Notice that $\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}}$ does not compute the sum of the

elements in a set since multiple occurrences of the same member can be summed up more than once. For example,

$$\begin{aligned}\mathsf{sum}(\langle\mathsf{s}(0)\rangle \cup \langle\mathsf{s}(0)\rangle) &\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \mathsf{sum}(\langle 0\rangle) + \mathsf{sum}(\langle\mathsf{s}(0)\rangle)\\ &\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \mathsf{s}(0) + \mathsf{sum}(\langle\mathsf{s}(0)\rangle)\\ &\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \mathsf{s}(0) + \mathsf{s}(0)\\ &\to^*_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \mathsf{s}(\mathsf{s}(0))\end{aligned}$$

which is not what we intuitively expect. In contrast, evaluation with $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1\setminus\mathcal{R}}$ produces the correct result since the argument to sum does not contain duplicate members after normalization with $\to_{\mathcal{E}_1\setminus\mathcal{S}}$. ◊

Even more severely, $\to_{\mathcal{E}\setminus\mathcal{R}\cup\mathcal{S}}$ might not terminate while $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ does terminate.

Example 13. We again consider integers with $\mathcal{E}_1$ and $\mathcal{S}$ as in Example 1. Now we define a function for determining whether an integer is non-negative by $\mathcal{R} = \{\mathsf{nonneg}(0) \to \mathsf{true}, \mathsf{nonneg}(\mathsf{s}(x)) \to \mathsf{nonneg}(\mathsf{p}(\mathsf{s}(x)), \mathsf{nonneg}(\mathsf{p}(x)) \to \mathsf{false}\}$. Then $\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}}$ does not terminate since

$$\begin{aligned}\mathsf{nonneg}(\mathsf{s}(\mathsf{p}(0))) &\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \mathsf{nonneg}(\mathsf{p}(\mathsf{s}(\mathsf{p}(0))))\\ &\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \mathsf{nonneg}(\mathsf{p}(\mathsf{s}(\mathsf{p}(\mathsf{p}(0)))))\\ &\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}} \cdots\end{aligned}$$

is an infinite $\to_{\mathcal{E}_1\setminus\mathcal{R}\cup\mathcal{S}}$-reduction. In contrast, $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1\setminus\mathcal{R}}$ is terminating since $\mathsf{p}(\mathsf{s}(x))$ in the recursive call of nonneg is normalized to x before the nonneg-rule can be applied again. ◊

We next show two important properties of normalized rewriting with equational systems. Firstly, $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ does not distinguish between terms that are equivalent up to $\sim_{\mathcal{E}}$, in the following sense.

Lemma 14. *Let $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ be an equational system. If $s \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} t$ and $s' \sim_{\mathcal{E}} s$, then $s' \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} t'$ for some $t' \sim_{\mathcal{E}} t$.*

Proof. Let $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ be an equational system and assume $s \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} t$. Thus, $s = C[f(u^*)]$ for some context C with $f(u^*)\!\downarrow_{\mathcal{E}\setminus\mathcal{S}} \sim_{\mathcal{E}} l\sigma$ for some rule $l \to r$ in $\mathcal{R}$ with $\mathrm{root}(l) = f$, and $t = C[r\sigma]$. Since $f \notin \mathcal{F}(\mathcal{E})$ and $s \sim_{\mathcal{E}} s'$, Lemma 5 implies $s' = C'[f(u'^*)]$ for some context C' with $C' \sim_{\mathcal{E}} C$ and $u^* \sim_{\mathcal{E}} u'^*$. Since $\to_{\mathcal{E}\setminus\mathcal{S}}$ is convergent modulo $\mathcal{E}$ we get $f(u'^*)\!\downarrow_{\mathcal{E}\setminus\mathcal{S}} \sim_{\mathcal{E}} f(u^*)\!\downarrow_{\mathcal{E}\setminus\mathcal{S}}$. Thus, $s' = C'[f(u'^*)] \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} C'[r\sigma] \sim_{\mathcal{E}} C[r\sigma] = t$. □

As a consequence we obtain that terms that are equivalent up to $\sim_{\mathcal{E}}$ have the same behavior w.r.t. starting infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reductions.

Corollary 15. *Let $(\mathcal{R},\mathcal{S},\mathcal{E})$ be an equational system and let $s \sim_{\mathcal{E}} s'$. Then s starts an infinite $\overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}}$-reduction iff s' starts an infinite $\overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}}$-reduction.*

Proof. Assume s starts an infinite $\overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}}$-reduction

$$s \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} s_1 \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} s_2 \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} s_3 \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} \dots$$

Using Lemma 14 we get

$$s' \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} s'_1 \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} s'_2 \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} s'_3 \overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}} \dots$$

where $s_i \sim_{\mathcal{E}} s'_i$, i.e., s' starts an infinite $\overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}}$-reduction as well. The inverse direction is shown the same way. □

4 Dependency Pairs

In this section we present a termination criterion for normalized rewriting with equational systems that is based on *dependency pairs*. As usual in any approach based on dependency pairs (see, e.g., [1, 6]), we extend $\mathcal{F}$ by a fresh *tuple symbol* $f^\sharp$ for each defined symbol $f \in \mathcal{D}(\mathcal{R})$, where $f^\sharp$ has the same arity as f. For any term $t = f(t^*)$, we denote the term $f^\sharp(t^*)$ by $t^\sharp$. The notion of a *dependency pair* is the standard one from [1]. Note that we do not need to add instantiations of rules, which is needed in [6]. This is due to our restriction to equations between constructors only.

Definition 16 (Dependency Pairs). *The set of* dependency pairs *for a TRS $\mathcal{R}$ is $\mathsf{DP}(\mathcal{R}) = \{l^\sharp \to t^\sharp \mid l \to r \in \mathcal{R}, t \text{ is a subterm of } r \text{ with } \mathrm{root}(t) \in \mathcal{D}(\mathcal{R})\}$.*

In order to verify termination we rely on the notion of *chains*. Intuitively, a dependency pair corresponds to a recursive call, and a chain represents a possible sequence of calls in a reduction. In the following we always assume that different (dependency) pairs are variable disjoint, and we consider substitutions whose domain may be infinite.

Definition 17 ($(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E})$-Chains). *Let $\mathcal{P}$ be a set of pairs and let $(\mathcal{R},\mathcal{S},\mathcal{E})$ be an equational system. A (possibly infinite) sequence of pairs $s_1 \to t_1, s_2 \to t_2, \dots$ from $\mathcal{P}$ is a* $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E})$-chain *iff there exists a substitution σ such that $t_i\sigma \overset{\mathcal{S}}{\to}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all i and $s_1\sigma$ is in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$.*

Example 18. We consider the following equational system $(\mathcal{R},\mathcal{S},\mathcal{E})$:

$$\begin{array}{lrlr}
\mathcal{R}: & \mathsf{p}(0) \to 0 & \mathcal{S}: & u + 0 \to u \\
& \mathsf{p}(\mathsf{s}(x)) \to x & & u + \mathsf{s}(v) \to \mathsf{s}(u + v) \\
& x - 0 \to x & & (u + \mathsf{s}(v)) + w \to \mathsf{s}(u + v) + w \\
& x - \mathsf{s}(y) \to \mathsf{p}(x - y) & \mathcal{E}: & u + (v + w) \approx (u + v) + w \\
& & & u + v \approx v + u
\end{array}$$

Here, $\mathcal{S}$ and $\mathcal{E}$ were obtained from the ES $\{u+\mathsf{0} \approx u, u+\mathsf{s}(v) \approx \mathsf{s}(u+v)\} \cup \mathsf{AC}_+$, which specifies properties of the natural numbers in Peano representation. The third rule in $\mathcal{S}$ is the extension of the second rule. Then $\mathsf{DP}(\mathcal{R})$ contains the dependency pairs

$$x -^\sharp \mathsf{s}(y) \to x -^\sharp y,$$
$$x -^\sharp \mathsf{s}(y) \to \mathsf{p}^\sharp(x - y).$$

Using the first dependency pair twice, we can construct the chain

$$x_1 -^\sharp \mathsf{s}(y_1) \to x_1 -^\sharp y_1,\ x_2 -^\sharp \mathsf{s}(y_2) \to x_2 -^\sharp y_2$$

by considering the substitution σ with $x_1\sigma = \mathsf{0}, x_2\sigma = \mathsf{0}, y_1\sigma = \mathsf{s}(\mathsf{0}), y_2\sigma = \mathsf{0}$. For this substitution, the instantiated right side of the first pair is $\mathsf{0} -^\sharp \mathsf{s}(\mathsf{0})$, the same as the instantiated left side of the second pair. Furthermore, both instantiated left sides, $\mathsf{0} -^\sharp \mathsf{s}(\mathsf{s}(\mathsf{0}))$ and $\mathsf{0} -^\sharp \mathsf{s}(\mathsf{0})$, are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$. ◊

In the definition of a $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E})$-chain we only require $s_1\sigma$ to be in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$, while we do not require this for $s_i\sigma$ for $i > 1$ since this is already true by the other requirements on a $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E})$-chain.

Lemma 19. *Let $s \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} t$. Then t is in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$.*

Proof. Let $s \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} t$. Thus, $s \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} t' \sim_{\mathcal{E}} t$ for some t', where t' is in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$. Assume that t is not in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$. Then, $t \to_{\mathcal{E}\setminus\mathcal{S}} w$ for some term w. By Lemma 8 we get $t' \to_{\mathcal{E}\setminus\mathcal{S}} w'$ for some $w' \sim_{\mathcal{E}} w$, i.e., t' is not in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$, either. □

Using chains, we obtain the following termination criterion, which is the key result of the dependency pair approach.

Theorem 20. *Let $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ be an equational system. If there are no infinite $(\mathsf{DP}(\mathcal{R}), \mathcal{R}, \mathcal{S}, \mathcal{E})$-chains, then $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ is terminating.*

Proof. Assume there exists a term t which starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction. By a minimality argument, t contains a subterm $f_1(u_1^*)$ which starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction, but none of the terms in u_1^* starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction.

Consider an infinite reduction starting with $f_1(u_1^*)$. First, the arguments u_1^* are reduced in zero or more steps to terms v_1^*, and then a rewrite rule is applied to $f_1(v_1^*)$ at the root, i.e., there exists a rule $l_1 \to r_1$ in $\mathcal{R}$ and a substitution σ_1 such that $f_1(v_1^*) \to^!_{\mathcal{E}\setminus\mathcal{S}} f_1(\overline{v}_1^*) \sim_{\mathcal{E}} l_1\sigma_1$, and hence the reduction yields $r_1\sigma_1$. Now the infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction continues with $r_1\sigma_1$, i.e., the term $r_1\sigma_1$ starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction, too. So up to now, the reduction of $f_1(u_1^*)$ has the following form:

$$f_1(u_1^*) \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} f_1(v_1^*) \to^!_{\mathcal{E}\setminus\mathcal{S}} f_1(\overline{v}_1^*) \sim_{\mathcal{E}} l_1\sigma_1 \to_{\mathcal{R}} r_1\sigma_1$$

By the definition of $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ we obtain $l_1 = f_1(w_1^*)$ and $\overline{v}_1^* \sim_{\mathcal{E}} w_1^*\sigma_1$. Using Corollary 15, this means that the terms $w_1^*\sigma_1$ do not start infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reductions since the terms $\overline{v}_1^*$ do not.

Hence, for all variables x occurring in $f_1(w_1^*)$, the terms $x\sigma_1$ do not start infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reductions. Thus, since $r_1\sigma_1$ starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction, there exists a subterm $f_2(u_2^*)$ in r_1 such that $f_2(u_2^*)\sigma_1$ starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction, whereas the terms in $u_2^*\sigma_1$ do not start infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reductions.

The first dependency pair in the infinite $(\mathsf{DP}(\mathcal{R}), \mathcal{R}, \mathcal{S}, \mathcal{E})$-chain that we are going to construct is $f_1^\sharp(w_1^*) \to f_2^\sharp(u_2^*)$, obtained from the rewrite rule $l_1 \to r_1$. The other dependency pairs of the infinite $(\mathsf{DP}(\mathcal{R}), \mathcal{R}, \mathcal{S}, \mathcal{E})$-chain are determined in the same way: let $f_{i-1}^\sharp(w_{i-1}^*) \to f_i^\sharp(u_i^*)$ be a dependency pair such that $f_i^\sharp(u_i^*)\sigma_{i-1}$ starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction and the terms $u_i^*\sigma_{i-1}$ do not. Again, in zero or more steps $f_i(u_i^*)\sigma_{i-1}$ reduces to $f_i(v_i^*)$ to which a rewrite rule $f_i(w_i^*) \to r_i$ is applied and $r_i\sigma_i$ starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction for a substitutions σ_i with $\overline{v}_i^* \sim_{\mathcal{E}} w_i^*\sigma_i$. As above, r_i contains a subterm $f_{i+1}(u_{i+1}^*)$ such that $f_{i+1}(u_{i+1}^*)\sigma_i$ starts an infinite $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$-reduction, whereas the terms $u_{i+1}^*\sigma_i$ do not. This produces the i^{th} dependency pair $f_i^\sharp(w_i^*) \to f_{i+1}^\sharp(u_{i+1}^*)$. In this way, we obtain the infinite sequence

$$f_1^\sharp(w_1^*) \to f_2^\sharp(u_2^*), f_2^\sharp(w_2^*) \to f_3^\sharp(u_3^*), f_3^\sharp(w_3^*) \to f_4^\sharp(u_4^*), \ldots$$

It remains to be shown that this sequence is indeed a $(\mathsf{DP}(\mathcal{R}), \mathcal{R}, \mathcal{S}, \mathcal{E})$-chain.

For this, note that $f_i^\sharp(u_i^*)\sigma_{i-1} \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} f_i^\sharp(v_i^*) \to^!_{\mathcal{E}\setminus\mathcal{S}} f_i^\sharp(\overline{v}_i^*) \sim_{\mathcal{E}} f_i^\sharp(w_i^*)\sigma_i$ for all $i > 1$. Since we assume that the variables of different occurrences of dependency pairs are disjoint, we obtain one substitution $\sigma = \sigma_1 \cup \sigma_2 \cup \sigma_3 \cup \ldots$ such that $f_i^\sharp(u_i^*)\sigma \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} f_i^\sharp(w_i^*)\sigma$ for all $i > 1$. □

This theorem gives rise to a first termination criterion, which relies on *reduction pairs*.

Definition 21 (Reduction Pairs). *Let $\gtrsim$ be reflexive, transitive, monotonic, and stable*[5]*. Let $\succ$ be well-founded and stable. Then $(\gtrsim, \succ)$ is a* reduction pair *iff $\succ$ is compatible with $\gtrsim$, i.e., iff $\gtrsim \circ \succ \subseteq \succ$ and $\succ \circ \gtrsim \subseteq \succ$. We denote the equivalence part $\gtrsim \cap \gtrsim^{-1}$ by $\sim$.*

Note that $\succ$ does not need to be monotonic in a reduction pair. This is the main advantage of the dependency pair approach which enables proving termination of many rewrite systems where simplification orders fail. In order to generate reduction pairs automatically, classical (monotonic) simplification orders are often used. To benefit from the possibility that $\succ$ does not need to be monotonic, *argument filterings* (which allow the deletion of certain function

[5] A relation $\bowtie$ on terms is monotonic iff $s \bowtie t$ implies $C[s] \bowtie C[t]$ for all contexts C. It is stable iff $s \bowtie t$ implies $s\sigma \bowtie t\sigma$ for all substitutions σ.

symbols and arguments) are commonly used in combination with monotonic orders (see [1]).

In the following, let $\mathcal{P}_{\bowtie} = \{(s,t) \in \mathcal{P} \mid s \bowtie t\}$ for any set $\mathcal{P}$ of pairs of terms and any relation $\bowtie$. Thus, for example, $\mathcal{R}_{\gtrsim} = \mathcal{R}$ means that $l \gtrsim r$ for all $l \to r \in \mathcal{R}$.

Theorem 22. *Let $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ be an equational system. Then $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ is terminating if there exists a reduction pair $(\gtrsim, \succ)$ such that*

- $\mathsf{DP}(\mathcal{R})_{\succ} = \mathsf{DP}(\mathcal{R})$,
- $\mathcal{R}_{\gtrsim} = \mathcal{R}$,
- $\mathcal{S}_{\gtrsim} = \mathcal{S}$, *and*
- $\mathcal{E}_{\sim} = \mathcal{E}$.

Proof. Assume there exists an infinite $(\mathsf{DP}(\mathcal{R}), \mathcal{R}, \mathcal{S}, \mathcal{E})$-chain $s_1 \to t_1, s_2 \to t_2, \ldots$. Thus, there exists a substitution σ such that $t_i\sigma \xrightarrow{\mathcal{S}}{}^{*}_{\mathcal{E}\setminus\mathcal{R}} \circ \to^{!}_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all i. Hence, there are terms t_i' and t_i'' such that $t_i\sigma \xrightarrow{\mathcal{S}}{}^{*}_{\mathcal{E}\setminus\mathcal{R}} t_i' \to^{!}_{\mathcal{E}\setminus\mathcal{S}} t_i'' \sim_{\mathcal{E}} s_{i+1}\sigma$. We show that $t_i\sigma \gtrsim s_{i+1}\sigma$.

First, note that $w \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} w''$ implies $w \gtrsim w''$. If $w \xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}} w''$, then $w \to^{*}_{\mathcal{E}\setminus\mathcal{S}} w' \to_{\mathcal{E}\setminus\mathcal{R}} w''$ for some term w'. Since $\mathcal{S} \subseteq \gtrsim$ and $\mathcal{E} \subseteq \sim$, we get $w \gtrsim w'$ because $\gtrsim$ is transitive. Similarly, $w' \gtrsim w''$ since $\mathcal{R} \subseteq \gtrsim$ as well. Thus, $w \gtrsim w''$.

This implies $t_i\sigma \gtrsim t_i'$. Also, $t_i' \to^{!}_{\mathcal{E}\setminus\mathcal{S}} t_i''$ implies $t_i' \gtrsim t_i''$ using the same argument. Finally, $t_i'' \sim s_{i+1}\sigma$ since $t_i'' \sim_{\mathcal{E}} s_{i+1}\sigma$ and $\mathcal{E} \subseteq \sim$.

In total, then, $t_i\sigma \gtrsim s_{i+1}\sigma$ for all i. Since $\mathsf{DP}(\mathcal{R}) \subseteq \succ$ we obtain $s_i\sigma \succ t_i\sigma$ for all i. Hence, the infinite chain gives rise to

$$s_1\sigma \succ t_1\sigma \gtrsim s_2\sigma \succ t_2\sigma \gtrsim \ldots$$

Using the compatibility of $\succ$ with $\gtrsim$, this contradicts the well-foundedness of $\succ$. Thus, there are no infinite chains and $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ is terminating by Theorem 20. □

Example 23. We now apply Theorem 22 in order to show that $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ is terminating, where $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ is the equational system from Example 18. Thus, we need to find a reduction pair $(\gtrsim, \succ)$ such that

$$\begin{array}{rcl} x -^{\sharp} \mathsf{s}(y) \succ x -^{\sharp} y & \qquad & u + \mathsf{0} \gtrsim u \\ x -^{\sharp} \mathsf{s}(y) \succ \mathsf{p}^{\sharp}(x - y) & & u + \mathsf{s}(v) \gtrsim \mathsf{s}(u + v) \\ x - \mathsf{0} \gtrsim x & & (u + \mathsf{s}(v)) + w \gtrsim \mathsf{s}(u + v) + w \\ x - \mathsf{s}(y) \gtrsim \mathsf{p}(x - y) & & u + (v + w) \sim (u + v) + w \\ \mathsf{p}(\mathsf{0}) \gtrsim \mathsf{0} & & u + v \sim v + u \\ \mathsf{p}(\mathsf{s}(x)) \gtrsim x & & \end{array}$$

Using the reduction pair based on the polynomial order induced by $\mathcal{P}ol(\mathsf{0}) = 0, \mathcal{P}ol(\mathsf{s}(x)) = x + 1, \mathcal{P}ol(x + y) = x + y, \mathcal{P}ol(\mathsf{p}(x)) = x, \mathcal{P}ol(\mathsf{p}^{\sharp}(x)) = x, \mathcal{P}ol(x - y) = x + y$, and $\mathcal{P}ol(x -^{\sharp} y) = x + y$, these constraints are satisfied. ◊

5 Dependency Pair Framework

Theorem 20 provides a first method for proving termination, but this method is inflexible. For regular rewriting, a huge number of techniques has been developed atop the basic dependency pair approach (see, e.g., [9, 11, 10]). In order to show soundness of these techniques independently, and in order to be able to freely combine them in a flexible manner in implementations like AProVE [8], the notions of DP problems and DP processors were introduced in the context of regular rewriting in [9], giving rise to the DP framework. In [20] the DP framework was extended to equational rewriting under the restrictions of [6]. Here, we extend these notions to normalized rewriting.

Definition 24 (DP Problems). *A* DP problem *is a tuple* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ *where* $\mathcal{P}$ *is a set of pairs such that either*

1. $\gamma = \mathbf{n}$ *and* $(\mathcal{R}, \mathcal{S}, \mathcal{E})$ *is an equational system, or*
2. $\gamma = \mathbf{e}$*,* $\mathcal{S} = \emptyset$*,* $\mathcal{E}$ *is an i.u.v. ES that is not collapsing, and* $\mathcal{R}$ *is a TRS that is complete modulo* $\mathcal{E}$*.*

While DP problems with $\gamma = \mathbf{n}$ are used with $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$, DP problems with $\gamma = \mathbf{e}$ are used with $\rightarrow_{\mathcal{E}\setminus\mathcal{R}}$. In order to have a uniform notation we introduce the following definition.

Definition 25 ($\rightarrow_{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)}$). *Let* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ *be a DP problem. The relation* $\rightarrow_{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)}$ *is defined to be* $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ *if* $\gamma = \mathbf{n}$ *and* $\rightarrow_{\mathcal{E}\setminus\mathcal{R}}$ *if* $\gamma = \mathbf{e}$*.*

Since we deal with two different kinds of DP problems we accordingly have to modify the definition of a chain.

Definition 26 ($(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain). *Let* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ *be a DP problem. A sequence of pairs* $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \ldots$ *from* $\mathcal{P}$ *is a* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain *iff either*

1. $\gamma = \mathbf{n}$ *and* $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \ldots$ *is a* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E})$*-chain, or*
2. $\gamma = \mathbf{e}$ *and there exists a substitution* σ *such that* $t_i\sigma \rightarrow^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ *for all* i*.*

DP problems are now classified by whether they allow infinite chains.

Definition 27 (Finite DP Problems). *A DP problem* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ *is* finite *iff there do not exist infinite* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$*-chains. Otherwise, the DP problem is* infinite.[6]

According to Theorem 20 we are interested in showing that the DP problem $(\mathsf{DP}(\mathcal{R}), \mathcal{R}, \mathcal{S}, \mathcal{E}, \mathbf{n})$ is finite for an equational system $(\mathcal{R}, \mathcal{S}, \mathcal{E})$. In order to show the finiteness of a DP problem, it is transformed into a set of DP problems whose finiteness has to be shown instead. This transformation is done by *DP processors*.

[6] Note that this definition of (in)finite DP problems is simpler than the one used in [9]. This simpler notion is sufficient for our purposes.

Definition 28 (DP Processors). *A* DP processor *is a function Proc which takes a DP problem as input and returns a set of DP problems as output. Proc is* sound *iff for all DP problems $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ the finiteness of all DP problems in $Proc(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ implies the finiteness of $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$.*

Note that $Proc(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)\}$ is possible. This can be interpreted as a failure of *Proc* on its input and indicates that a different DP processor should be applied. The following is immediate from Definition 27, Definition 28, and Theorem 20.

Corollary 29. *Let $(\mathcal{R},\mathcal{S},\mathcal{E})$ be an equational system. We construct a tree whose nodes are labelled with DP problems or "*yes*" and whose root is labelled with $(\mathsf{DP}(\mathcal{R}),\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n})$. For every inner node labelled with the DP problem d, there is a sound DP processor Proc satisfying one of the following conditions:*

- *$Proc(d) = \emptyset$ and the node has just one child, labelled with "*yes*"*
- *$Proc(d) \neq \emptyset$ and the children of the node are labelled with the DP problems in $Proc(d)$.*

If all leaves of the tree are labelled with "yes*", then $\xrightarrow{\mathcal{S}}_{\mathcal{E}\setminus\mathcal{R}}$ is terminating.*

6 DP Processors

In this section we introduce a variety of DP processors and prove their soundness. Most DP processors are inspired by similar DP processors in the context of regular rewriting (see [9, 10]).

6.1 A Trivial DP Processor

The first DP processor determines the finiteness of a trivial DP problem that does not contain any pairs.[7]

Lemma 30 (Trivial DP Processor). *Let Proc be a DP processor such that $Proc(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \emptyset$ if $\mathcal{P} = \emptyset$ and $Proc(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)\}$ if $\mathcal{P} \neq \emptyset$. Then Proc is sound.*

Proof. If $\mathcal{P} = \emptyset$, then $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ is clearly finite. In the other case soundness is obvious as well. □

[7] This DP processor is actually not needed since the DP processor from Theorem 37 in Section 6.3 serves the same purpose.

6.2 A DP Processor for Switching to Equational Rewriting

It is immediate from the definitions that $\overset{\mathcal{S}}{\to}_{\mathcal{E}\setminus\mathcal{R}}$ is contained in $\to_{\mathcal{E}\setminus\mathcal{R}\cup\mathcal{S}}$. The following DP processor thus transforms the DP problem $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n})$ into the DP problem $(\mathcal{P},\mathcal{R}\cup\mathcal{S},\emptyset,\mathcal{E},\mathbf{e})$. It is of particular interest if there are sound DP processors that are only applicable (or more powerful) if $\mathcal{S}=\emptyset$. An example of such a DP processor is given in Section 6.6. The disadvantage of using this DP processor is that the information that certain terms are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$ is lost, which might be useful information for a DP processor. For example, the DP processor in Section 6.3 makes use of this.

Theorem 31 (DP Processor for Switching to Equational Rewriting). *Let Proc be a DP processor such that* $Proc(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n}) = \{(\mathcal{P},\mathcal{R}\cup\mathcal{S},\emptyset,\mathcal{E},\mathbf{e})\}$ *if* $\mathcal{E}$ *is not collapsing and* $Proc(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)\}$ *in all other cases. Then Proc is sound.*

Proof. Let $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n})$ be a DP problem where $\mathcal{E}$ is not collapsing and assume there exists an infinite $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n})$-chain $s_1 \to t_1, s_2 \to t_2 \ldots$. Thus, there exists a substitution σ such that $t_i\sigma \overset{\mathcal{S}}{\to}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all $i \geq 1$. Hence $t_i\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}\cup\mathcal{S}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ as well and $s_1 \to t_1, s_2 \to t_2, \ldots$ is also an infinite $(\mathcal{P},\mathcal{R}\cup\mathcal{S},\emptyset,\mathcal{E},\mathbf{e})$-chain. It still remains to be shown that $(\mathcal{P},\mathcal{R}\cup\mathcal{S},\emptyset,\mathcal{E},\mathbf{e})$ is a DP problem, i.e., that $\mathcal{R}\cup\mathcal{S}$ is complete modulo $\mathcal{E}$. First note that $\mathcal{S}$ is complete modulo $\mathcal{E}$ by assumption. $\mathcal{R}$ is complete modulo $\mathcal{E}$ by [6, Definition 8 and Lemma 10] since $\mathcal{F}(\mathcal{E}) \cap \mathcal{D}(\mathcal{R}) = \emptyset$. Thus, $\mathcal{R}\cup\mathcal{S}$ is complete modulo $\mathcal{E}$.

In all other cases soundness is obvious. □

Since the rule from $\mathcal{S}$ are put into $\mathcal{R}$, function symbols that used to be constructors might effectively become defined symbols. We hence modify the definition of defined symbols.

Definition 32 (Defined Symbols). *Let* $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ *be a DP problem. The* defined symbols *of* $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ *are defined by*

$$\mathcal{D}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \begin{cases} \mathcal{D}(\mathcal{R}) & \text{if } \gamma = \mathbf{n} \\ \mathcal{D}(\mathcal{R},\mathcal{E}) & \text{if } \gamma = \mathbf{e} \end{cases}$$

where $\mathcal{D}(\mathcal{R},\mathcal{E})$ *is the smallest set such that* $\mathcal{D}(\mathcal{R},\mathcal{E}) = \{\mathrm{root}(l) \mid l \to r \in \mathcal{R}\} \cup \{\mathrm{root}(v) \mid u \approx v \in \mathcal{E} \text{ or } v \approx u \in \mathcal{E}, \mathrm{root}(u) \in \mathcal{D}(\mathcal{R},\mathcal{E})\}$.

Here, $\mathcal{D}(\mathcal{R},\mathcal{E})$ was already used in [6, Definition 12] and requires that $\mathcal{E}$ is not collapsing.

6.3 A DP Processor Based on Dependency Graphs

The DP processor introduced in this section decomposes a DP problem into several independent DP problems by determining which pairs of $\mathcal{P}$ may follow each other in a $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$-chain. The processor relies on the *dependency graph*, which is also used in regular rewriting (see [1]).

Definition 33 (Dependency Graphs). *Let $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ be a DP problem. The nodes of the the $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-*dependency graph $\mathsf{DG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ *are the pairs in $\mathcal{P}$ and there is an arc from $s_1 \to t_1$ to $s_2 \to t_2$ iff $s_1 \to t_1, s_2 \to t_2$ is a $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain.*

A set $\mathcal{P}' \subseteq \mathcal{P}$ of pairs is a *cycle* iff for all pairs $s_1 \to t_1$ and $s_2 \to t_2$ in $\mathcal{P}'$ there exists a path from $s_1 \to t_1$ to $s_2 \to t_2$ that only traverses pairs from $\mathcal{P}'$. A cycle is a *strongly connected component* (SCC) if it is not a proper subset of any other cycle.[8] Now, every infinite $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain corresponds to a cycle in $\mathsf{DG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$, and it is thus sufficient to prove the absence of infinite chains for all SCCs.

In general $\mathsf{DG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ cannot be computed exactly since it is undecidable whether two pairs form a chain. Thus, an estimation has to be used instead. The idea of the estimation is that subterms of t_1 with a defined root symbol are abstracted by a fresh variable. Then, it is checked whether this term and s_2 are $\mathcal{E} \cup \mathcal{S}$-unifiable.

Definition 34 (Estimated Dependency Graphs). *Let $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ be a DP problem. The* estimated $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-dependency graph $\mathsf{EDG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ *has the pairs in $\mathcal{P}$ as nodes and there is an arc from $s_1 \to t_1$ to $s_2 \to t_2$ iff $\mathrm{CAP}(t_1)$ and s_2 are $\mathcal{E} \cup \mathcal{S}$-unifiable with an unifier μ such that $s_1\mu$ and $s_2\mu$ are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$.*

Here, CAP *is defined as*

$$\mathrm{CAP}(x) = y \quad \textit{for variables } x$$

$$\mathrm{CAP}(f(t_1, \ldots, t_n)) = \begin{cases} y & \textit{if } f \in \mathcal{D}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma) \\ f(\mathrm{CAP}(t_1), \ldots, \mathrm{CAP}(t_n)) & \textit{if } f \notin \mathcal{D}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma) \end{cases}$$

where y is the next variable in an infinite list $y_1, y_2, \ldots$ of fresh variables.

Next, we show that the estimated dependency graph is indeed an overapproximation of the dependency graph, i.e., every arc in $\mathsf{DG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ is also present in $\mathsf{EDG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$.

Lemma 35. *Let $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ be a DP problem. Then $\mathsf{EDG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ is an overapproximation of $\mathsf{DG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$, i.e., if there is an arc from $s_1 \to t_1$ to $s_2 \to t_2$ in $\mathsf{DG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$, then there is an arc from $s_1 \to t_1$ to $s_2 \to t_2$ in $\mathsf{EDG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$.*

Proof. Let $s_1 \to t_1, s_2 \to t_2$ be a $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain. We need to show that $\mathrm{CAP}(t_1)$ and s_2 are $\mathcal{E} \cup \mathcal{S}$-unifiable with an unifier μ such that $s_1\mu$ and $s_2\mu$ are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$.

Since $s_1 \to t_1, s_2 \to t_2$ is a $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain, there exists a substitution σ such that $t_1\sigma \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} u \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} s_2\sigma$ if $\gamma = \mathbf{n}$ or $t_1\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} u \sim_{\mathcal{E}} s_2\sigma$ if

[8] Note that the notions of cycle and SCC are different from the ones used in graph theory. We follow the notions used in the dependency pair literature.

$\gamma = \mathbf{e}$. Note that $s_1\sigma$ and $s_2\sigma$ are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$ by Definition 17 and Lemma 19 if $\gamma = \mathbf{n}$. For $\gamma = \mathbf{e}$ we have $\mathcal{S} = \emptyset$ and $s_1\sigma$ and $s_2\sigma$ are clearly in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$ as well.

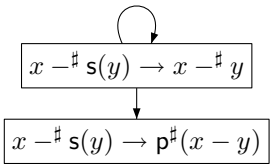
We first prove the following property by induction on t_1.

> If $t_1\sigma \to^*_{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)} u$ for some term u, then there exists a substitution τ whose domain contains only variables that are introduced in the construction of $\text{CAP}(t_1)$ such that $\text{CAP}(t_1)\sigma\tau = u$.

If $\text{root}(t_1) \in \mathcal{D}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ or $t_1 \in \mathcal{V}$, then $\text{CAP}(t_1)$ is a fresh variable y. Letting $\tau = \{y \mapsto u\}$ establishes the claim since $\text{CAP}(t_1)\sigma\tau = y\sigma\tau = y\tau = u$ because y is a fresh variable. Otherwise, $t_1 = c(t'_1,\ldots,t'_n)$ with $c \notin \mathcal{D}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ and $t'_i\sigma \to^*_{(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)} u'_i$, where $u = c(u'_1,\ldots u'_n)$. By the inductive hypothesis there exist substitutions τ_i such that $\text{CAP}(t'_i)\sigma\tau_i = u'_i$. Since the variables newly introduced in $\text{CAP}(t'_i)$ are disjoint from the variables newly introduced in $\text{CAP}(t'_j)$ for all $i \neq j$, we can let $\tau = \tau_i \cup \ldots \cup \tau_n$. Then $\text{CAP}(t_1)\sigma\tau = c(\text{CAP}(t'_1)\sigma\tau,\ldots,\text{CAP}(t'_n)\sigma\tau) = c(u'_1,\ldots u'_n) = u$.

Since $u \to^!_{\mathcal{E}\setminus\mathcal{S}} \sim_{\mathcal{E}} s_2\sigma$ if $\gamma = \mathbf{n}$ or $u \sim_{\mathcal{E}} s_2\sigma$ if $\gamma = \mathbf{e}$ we get $u \sim_{\mathcal{E}\cup\mathcal{S}} s_2\sigma$. Thus, $\text{CAP}(t_1)\sigma\tau \sim_{\mathcal{E}\cup\mathcal{S}} s_2\sigma = s_2\sigma\tau$ since the variables of $s_2\sigma$ do not occur in the domain of τ. Hence, $\text{CAP}(t_1)$ and s_2 are $\mathcal{E}\cup\mathcal{S}$-unifiable with an unifier $\mu = \sigma\tau$ and it remains to be shown that $s_1\mu$ and $s_2\mu$ are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$. But $s_1\mu = s_1\sigma\tau = s_1\sigma$ and $s_2\mu = s_2\sigma\tau = s_2\sigma$ since the variables of $s_1\sigma$ and $s_2\sigma$ do not occur in the domain of τ. Thus, $s_1\mu$ and $s_2\mu$ are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$ since $s_1\sigma$ and $s_2\sigma$ are in normal form w.r.t. $\to_{\mathcal{E}\setminus\mathcal{S}}$. □

Example 36. With $\mathcal{P} = \{x -^\sharp \mathsf{s}(y) \to x -^\sharp y, x -^\sharp \mathsf{s}(y) \to \mathsf{p}^\sharp(x - y)\}$ and $\mathcal{R},\mathcal{S}$ and $\mathcal{E}$ as in Example 23 we obtain the following $\mathsf{EDG}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n})$.



◊

In this example, $\mathsf{EDG}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ and $\mathsf{DG}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ coincide, but in general $\mathsf{EDG}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ is a strict supergraph of $\mathsf{DG}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$.

Theorem 37 (DP Processor Based on Dependency Graphs). *Let Proc be a DP processor with* $\mathit{Proc}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \{(\mathcal{P}_1,\mathcal{R},\mathcal{S},\mathcal{E},\gamma),\ldots,(\mathcal{P}_n,\mathcal{R},\mathcal{S},\mathcal{E},\gamma)\}$, *where* $\mathcal{P}_1,\ldots,\mathcal{P}_n$ *are the SCCs of* $(\mathsf{E})\mathsf{DG}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$.[9] *Then Proc is sound.*

Proof. After a finite number of pairs in the beginning, any infinite $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$-chain only contains pairs from some SCC. Hence, every infinite $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$-chain gives rise to an infinite $(\mathcal{P}_i,\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$-chain for some $1 \leq i \leq n$. □

Example 38. Continuing Example 36 we have $\mathit{Proc}(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n}) = \{(\{x -^\sharp \mathsf{s}(y) \to x -^\sharp y\},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{n})\}$. ◊

[9] Note, in particular, that $\mathit{Proc}(\emptyset,\mathcal{R},\mathcal{S},\mathcal{E},\gamma) = \emptyset$.

6.4 A DP Processor Based on Reduction Pairs

The DP processor presented in this section is closely related to the first termination criterion given in Theorem 22. It now, however, operates on DP problems $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$, and we do not require all pairs in $\mathcal{P}$ to be strictly decreasing.

Theorem 39 (DP Processor Based on Reduction Pairs). *Let $(\gtrsim, \succ)$ be a reduction pair. Let Proc be a DP processor such that $Proc(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ returns*

- $\{(\mathcal{P} - \mathcal{P}_\succ, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)\}$*, if*
  - $\mathcal{P}_\succ \cup \mathcal{P}_\gtrsim = \mathcal{P}$*,*
  - $\mathcal{R}_\gtrsim = \mathcal{R}$*,*
  - $\mathcal{S}_\gtrsim = \mathcal{S}$*, and*
  - $\mathcal{E}_\sim = \mathcal{E}$*.*
- $\{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)\}$*, otherwise.*

Then Proc is sound.

Proof. The proof for the first case is similar to the proof of Theorem 22. Since $\mathcal{P}$ is finite, any infinite $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain has to traverse at least one pair from $\mathcal{P}$ infinitely often. These pairs cannot be in $\mathcal{P}_\succ$ since this would contradict the well-foundedness of $\succ$.

In the other case soundness is obvious. □

Example 40. We consider the DP problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \mathbf{n})$ with $\mathcal{P} = \{x -^\sharp \mathsf{s}(y) \to x -^\sharp y\}$ from Example 38. Using the reduction pair based on the polynomial order induced by $\mathcal{P}ol(\mathsf{0}) = 0, \mathcal{P}ol(\mathsf{s}(x)) = x + 1, \mathcal{P}ol(x + y) = x + y, \mathcal{P}ol(\mathsf{p}(x)) = x, \mathcal{P}ol(x - y) = x + y$, and $\mathcal{P}ol(x -^\sharp y) = x + y$ the constraints for the first case of Theorem 39 are satisfied and the (only) pair $x -^\sharp \mathsf{s}(y) \to x -^\sharp y$ is strictly decreasing. It can thus be removed and we obtain the trivial DP problem $(\emptyset, \mathcal{R}, \mathcal{S}, \mathcal{E}, \mathbf{n})$. ◊

6.5 A DP Processor Based on Removal of Rules

In this section we present a DP processor for the modular removal of rules. For this, a DP problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ may be processed with a monotonic reduction pair $(\gtrsim, \succ)$. Then, rules $l \to r \in \mathcal{R}$ satisfying $l \succ r$ may be removed. For regular rewriting a corresponding DP processor was introduced in [21].

Theorem 41 (DP Processor Based on Removal of Rules). *Let $(\gtrsim, \succ)$ be a reduction pair where $\succ$ is monotonic. Let Proc be a DP processor such that $Proc(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$ returns*

- $\{(\mathcal{P} - \mathcal{P}_\succ, Ext_\mathcal{E}(\mathcal{R} - \mathcal{R}_\succ), \mathcal{S}, \mathcal{E}, \gamma)\}$[10]*, if*
  - $\mathcal{P}_\succ \cup \mathcal{P}_\gtrsim = \mathcal{P}$*,*
  - $\mathcal{R}_\succ \cup \mathcal{R}_\gtrsim = \mathcal{R}$*,*
  - $\mathcal{S}_\gtrsim = \mathcal{S}$*, and*

[10] The extension of $\mathcal{R} - \mathcal{R}_\succ$ is of course only needed if $\gamma = \mathbf{e}$.

  - $\mathcal{E}_\sim = \mathcal{E}$.
- $\{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)\}$, *otherwise.*

Then Proc is sound.

Proof. For the first case, let $s_1 \to t_1, s_2 \to t_2, \ldots$ be an infinite $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \gamma)$-chain. Thus, there exists a substitution σ such that either $t_i\sigma \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all i and $\gamma = \mathbf{n}$, or $\gamma = \mathbf{e}$ and $t_i\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all i. Assume for a contradiction that rules from $\mathcal{R}_\succ$ are applied for infinitely many i. Then, $t_i\sigma \succ s_{i+1}\sigma$ for infinitely many i since $\succ$ is monotonic, which contradicts the well-foundedness of $\succ$.

Thus, there exists some $n \geq 1$ such that either $t_i\sigma \xrightarrow{\mathcal{S}}{}^*_{\mathcal{E}\setminus\mathcal{R}-\mathcal{R}_\succ} \circ \to^!_{\mathcal{E}\setminus\mathcal{S}} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all $i \geq n$ and $\gamma = \mathbf{n}$, or $\gamma = \mathbf{e}$ and $t_i\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}-\mathcal{R}_\succ} \circ \sim_{\mathcal{E}} s_{i+1}\sigma$ for all $i \geq n$. Thus, $s_n \to t_n, s_{n+1} \to t_{n+1}, \ldots$ is an infinite $(\mathcal{P}, \mathcal{R} - \mathcal{R}_\succ, \mathcal{S}, \mathcal{E}, \gamma)$-chain. As in the proof of Theorem 39, pairs from $\mathcal{P}_\succ$ can only be traversed finitely often and there thus exists an infinite $(\mathcal{P} - \mathcal{P}_\succ, \mathcal{R} - \mathcal{R}_\succ, \mathcal{S}, \mathcal{E}, \gamma)$-chain.

In the other case soundness is obvious. □

Removing rules has several advantages. Firstly, it might be possible to remove "problematic" rules which prevent finding a reduction pair which yields a strict decrease in at least one pair of $\mathcal{P}$. Secondly, it might happen that $\mathcal{P}$ contains no cycle anymore after some rules are removed from $\mathcal{R}$ since some defined symbols might become constructors.

Example 42. We take the equational system from Example 18, but replace the second "−"-rule by

$$x - \mathsf{s}(y) \to \mathsf{p}(x - \mathsf{p}(\mathsf{s}(y)))$$

After computing the estimated dependency graph, we then obtain the DP problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \mathbf{n})$ with $\mathcal{P} = \{x -^\sharp \mathsf{s}(y) \to x -^\sharp \mathsf{p}(\mathsf{s}(y))\}$. In order to apply the DP processor from Theorem 39 we need to find a reduction pair $(\gtrsim, \succ)$ such that $x -^\sharp \mathsf{s}(y) \succ x -^\sharp \mathsf{p}(\mathsf{s}(y))$ and $\mathsf{p}(\mathsf{s}(x)) \gtrsim x$. It can be shown that there does not exist a reduction pair based on a simplification order with an argument filtering that satisfies these constraints, i.e., an automated proof will most likely fail.

Instead, we may apply the DP processor from Theorem 41 with the monotonic polynomial order based on $\mathcal{P}ol(\mathsf{0}) = 0, \mathcal{P}ol(\mathsf{s}(x)) = x + 1, \mathcal{P}ol(\mathsf{p}(x)) = x, \mathcal{P}ol(x + y) = \mathcal{P}ol(x - y) = \mathcal{P}ol(x -^\sharp y) = x + y$. Then all of $\mathcal{P}, \mathcal{R}$ and $\mathcal{S}$ are at least weakly decreasing, and the rule $\mathsf{p}(\mathsf{s}(x)) \to x$ is strictly decreasing and can thus be removed. Next, we can apply the DP processor from Theorem 39 with the polynomial order based on $\mathcal{P}ol(\mathsf{0}) = \mathcal{P}ol(\mathsf{p}(x)) = 0, \mathcal{P}ol(\mathsf{s}(x)) = x + 1, \mathcal{P}ol(x + y) = \mathcal{P}ol(x - y) = \mathcal{P}ol(x -^\sharp y) = x + y$. Then, the pair in $\mathcal{P}$ is strictly decreasing and all rules in $\mathcal{R}$ and $\mathcal{S}$ are at least weakly decreasing, i.e., we obtain the trivial DP problem $(\emptyset, \mathcal{R}, \mathcal{S}, \mathcal{E}, \mathbf{n})$. ◊

As mentioned in [21] and shown in the example, the DP processor from Theorem 41 can be automated efficiently by using monotonic polynomial orders induced by linear polynomials.

6.6 A DP Processor Based on Narrowing

In the context of regular rewriting it is often necessary to apply transformations to the pairs in a cycle in order to obtain a successful termination proof (see [1, 10]). In this section we introduce one such transformation within our framework. The transformation, however, can only be applied to DP problems with $f = \mathbf{e}$.

If it can be shown that for each chain containing a pair $s \to t$, the reduction from the instantiation of t to the instantiation of the left side of the next pair in the chain requires at least one $\to_{\mathcal{E}\setminus\mathcal{R}}$-step, then we can perform all possible $\to_{\mathcal{E}\setminus\mathcal{R}}$-reductions in order to obtain new pairs that replace the pair $s \to t$. Since we also need to determine the instantiations of t, we use the concept of *narrowing*. For this, let $\mathcal{CU}_{\mathcal{E}}(s,t)$ denote a complete set of $\mathcal{E}$-unifiers for the terms s and t, i.e., for each $\mathcal{E}$-unifier σ of s and t there exist a substitution $\mu \in \mathcal{CU}_{\mathcal{E}}(s,t)$ and some substitution ρ such that $\sigma =_{\mathcal{E}} \mu\rho$.

Definition 43 (Narrowing). *Let $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{e})$ be a DP problem. The term t* narrows *to the term t' with the substitution μ, written $t \rightsquigarrow^{\mu}_{\mathcal{E}\setminus\mathcal{R}} t'$, iff there exists a non-variable position p in t such that $\mu \in \mathcal{CU}_{\mathcal{E}}(t|_p, l)$[11] for some rule $l \to r$ in $\mathcal{R}$ and $t' = t\mu[r\mu]_p$.*

Now, given the DP problem $(\mathcal{P},\mathcal{R},\mathcal{S},\mathcal{E},\mathbf{e})$, we may replace some pair $s \to t \in \mathcal{P}$ by all of its narrowings if t satisfies certain conditions. Narrowing of dependency pairs has also been considered in [1, 4].

Theorem 44 (DP Processor Based on Narrowing). *Let Proc be a DP processor such that $Proc(\mathcal{P} \cup \{s \to t\},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$ returns*

- *$(\mathcal{P} \cup \{s\mu \to t' \mid t \rightsquigarrow^{\mu}_{\mathcal{E}\setminus\mathcal{R}} t'\},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$, if*
  - *$\gamma = \mathbf{e}$,*
  - *t is not $\mathcal{E}$-unifiable with any (variable-renamed) left side of a pair in $\mathcal{P} \cup \{s \to t\}$, and*
  - *t is linear.*
- *$(\mathcal{P} \cup \{s \to t\},\mathcal{R},\mathcal{S},\mathcal{E},\gamma)$, otherwise.*

Then Proc is sound.

Proof. If the second case applies then *Proc* is clearly sound. Otherwise, let $\mathcal{P}' = \mathcal{P} \cup \{s \to t\}$ and $\mathcal{P}'' = \mathcal{P} \cup \{s\mu \to t' \mid t \rightsquigarrow^{\mu}_{\mathcal{E}\setminus\mathcal{R}} t'\}$. We show that for every $(\mathcal{P}',\mathcal{R},\emptyset,\mathcal{E},\mathbf{e})$-chain $\ldots, v_1 \to w_1, s \to t, v_2 \to w_2, \ldots$ there exists a narrowing t' of t with the substitution μ such that $\ldots, v_1 \to w_1, s\mu \to t', v_2 \to w_2, \ldots$ is a $(\mathcal{P}'',\mathcal{R},\emptyset,\mathcal{E},\mathbf{e})$-chain (here, $s \to t$ may also be the first element in the chain, i.e., $v_1 \to w_1$ may be missing). Then, all occurrences of $s \to t$ in a chain may be replaced by pairs from $\{s\mu \to t' \mid t \rightsquigarrow^{\mu}_{\mathcal{E}\setminus\mathcal{R}} t'\}$ and every infinite $(\mathcal{P}',\mathcal{R},\emptyset,\mathcal{E},\mathbf{e})$-chain results in an infinite $(\mathcal{P}'',\mathcal{R},\emptyset,\mathcal{E},\mathbf{e})$-chain.

If $\ldots, v_1 \to w_1, s \to t, v_2 \to w_2, \ldots$ is a $(\mathcal{P}',\mathcal{R},\emptyset,\mathcal{E},\mathbf{e})$-chain, then there exists a substitution σ such that for all pairs in the chain the instantiated right side

[11] Here, the variables of $l \to r$ have been renamed to fresh variables.

reduces by $\to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}}$ to the instantiated left side of the next pair in the chain. Let σ be such a substitution where the length of the reduction

$$t\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} v_2\sigma$$

has a minimal number of $\to_{\mathcal{E}\setminus\mathcal{R}}$ steps. Note that there is at least one $\to_{\mathcal{E}\setminus\mathcal{R}}$ step since t and v_2 are not $\mathcal{E}$-unifiable. Hence we have $t\sigma \to_{\mathcal{E}\setminus\mathcal{R}} q \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} v_2\sigma$ for some term q.

First, we assume that the reduction $t\sigma \to_{\mathcal{E}\setminus\mathcal{R}} q$ takes place "in σ". Hence, $t|_p = x$ for some position p such that $x\sigma \to_{\mathcal{E}\setminus\mathcal{R}} r$ and $q = t[r]_p$. The variable x occurs only once in t since t is linear, and therefore we have $q = t\sigma'$, where σ' is the substitution with $x\sigma' = r$ and $y\sigma' = y\sigma$ for all $y \neq x$. As all (occurrences of) pairs in the chain can be assumed to be variable disjoint, σ' behaves like σ on all pairs except $s \to t$. For this pair we have

$$w_1\sigma' = w_1\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} s\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} s\sigma'$$

and

$$t\sigma' = q \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} v_2\sigma = v_2\sigma'$$

Using Lemma 8 we obtain $w_1\sigma' \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} s\sigma'$. Hence, σ' is also a substitution where each instantiated right side reduces by $\to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}}$ to the instantiation of the next left side in the chain. But as the reduction from $t\sigma'$ to $v_2\sigma'$ has less $\to_{\mathcal{E}\setminus\mathcal{R}}$-steps than the reduction from $t\sigma$ to $v_2\sigma$, this is a contradiction to the assumption that σ yields a reduction with a minimal number of $\to_{\mathcal{E}\setminus\mathcal{R}}$-steps.

So the reduction $t\sigma \to_{\mathcal{E}\setminus\mathcal{R}} q$ cannot take place "in σ". Hence, t contains some subterm $t|_p = f(u^*)$ such that a rule $l \to r$ has been applied to $f(u^*)\sigma$. In other words, there exists a substitution ρ such that $f(u^*)\sigma \sim_{\mathcal{E}} l\rho$. Hence, the reduction has the form

$$t\sigma = t\sigma[f(u^*)\sigma]_p \sim_{\mathcal{E}} t\sigma[l\rho]_p \to_{\mathcal{R}} t\sigma[r\rho]_p = q$$

Since we can assume that the variables of $l \to r$ have been renamed to fresh ones, we can extend σ to behave like ρ on the variables of l and r (but it still remains the same on all other variables). Now, σ is an $\mathcal{E}$-unifier of $f(u^*)$ and l, and hence there exists a substitution $\mu \in \mathcal{CU}_{\mathcal{E}}(f(u^*), l)$ and some substitution τ such that $\sigma \sim_{\mathcal{E}} \mu\tau$.

Let t' be the term $t\mu[r\mu]_p$. Then $t \rightsquigarrow^{\mu}_{\mathcal{E}\setminus\mathcal{R}} t'$. As we can assume $s\mu \to t'$ to be variable disjoint from all other pairs in the chain we can extend σ to behave like τ on the variables of $s\mu$ and t'. Then we have

$$w_1\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} s\sigma \sim_{\mathcal{E}} s\mu\tau = s\mu\sigma$$

and

$$t'\sigma = t'\tau = t\mu\tau[r\mu\tau]_p \sim_{\mathcal{E}} t\sigma[r\sigma]_p = q \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} v_2\sigma$$

Hence, $w_1\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} s\mu\sigma$ and $t'\sigma \to^*_{\mathcal{E}\setminus\mathcal{R}} \circ \sim_{\mathcal{E}} v_2\sigma$, where the latter is a consequence of Lemma 8. Thus, $\ldots, v_1 \to w_1, s\mu \to t', v_2 \to w_2, \ldots$ is a $(\mathcal{P}', \mathcal{R}, \emptyset, \mathcal{E}, \mathbf{e})$-chain. □

Example 45. We again consider the DP problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mathcal{E}, \mathbf{n})$ from Example 42. By applying the DP processor from Theorem 31, we obtain the DP problem $(\mathcal{P}, \mathcal{R} \cup \mathcal{S}, \emptyset, \mathcal{E}, \mathbf{e})$. For the only pair $x -^\sharp \mathsf{s}(y) \to x -^\sharp \mathsf{p}(\mathsf{s}(y))$ the right side is linear and does not $\mathcal{E}$-unify with the (variable-renamed) left side. We can thus replace that pair by its narrowings. The only narrowing of the pair is the pair $x -^\sharp \mathsf{s}(y) \to x -^\sharp y$, resulting in a DP problem that can be handled like the one in Example 40. ◊

7 Conclusions

We have proposed normalized rewriting as an alternative to $\mathcal{E}$-extended rewriting for equational rewrite systems in which equations only relate constructors. The paper extends the dependency pair framework in order to establish termination of normalized rewriting for such equational rewrite systems. It is shown that whereas $\mathcal{E}$-extended rewriting for such systems may not terminate, normalized rewriting often does terminate. Based on our experience in specifying a number of examples on data structures generated by non-free constructors, we feel that algorithms can be specified naturally and elegantly as rewrite systems (pos is one such example). Unlike previous related work [6], the equations relating constructors may be collapsing and, in some cases, do not need to have identical unique variables. In particular, properties such as idempotency, identity, etc., of constructors on data structures are allowed.

Many functional programming languages use eager evaluation as the evaluation strategy. Then, termination of the functional program corresponds to *innermost* termination of the equational rewrite system. We believe that our method can be extended to show innermost termination, similarly to how this can be done for regular rewriting [1]. This needs to be investigated. An implementation of the proposed approach in AProVE [8] is planned.

References

1. Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.
2. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
3. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. The Maude 2.0 system. In Robert Nieuwenhuis, editor, *Proceedings of the 14th International Conference on Rewriting Techniques and Applications (RTA '03)*, volume 2706 of *Lecture Notes in Computer Science*, pages 76–87. Springer-Verlag, 2003.
4. Stephan Falke. Automated termination analysis for equational rewriting. Diplomarbeit, Department of Computer Science, Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, 2004.
5. Jürgen Giesl and Deepak Kapur. Dependency pairs for equational rewriting. Technical Report TR-CS-2000-53, Department of Computer Science, University of New Mexico, Albuquerque, NM, USA, 2000.

6. Jürgen Giesl and Deepak Kapur. Dependency pairs for equational rewriting. In Aart Middeldorp, editor, *Proceedings of the 12th International Conference on Rewriting Techniques and Applications (RTA '01)*, volume 2051 of *Lecture Notes in Computer Science*, pages 93–108. Springer-Verlag, 2001.
7. Jürgen Giesl and Deepak Kapur, editors. *Journal of Automated Reasoning*, 34(2), 34(4) & 37(3), 2005–2006. Special issues on *Techniques for Automated Termination Proofs*.
8. Jürgen Giesl, Peter Schneider-Kamp, and René Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR '06)*, volume 4130 of *Lecture Notes in Computer Science*, pages 281–286. Springer-Verlag, 2006.
9. Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In Franz Baader and Andrei Voronkov, editors, *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '04)*, volume 3452 of *Lecture Notes in Computer Science*, pages 301–331. Springer-Verlag, 2004.
10. Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.
11. Nao Hirokawa and Aart Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 2007. To appear.
12. Deepak Kapur and G. Sivakumar. Proving associative-commutative termination using RPO-compatible orderings. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics: Selected Papers*, volume 1761 of *Lecture Notes in Artificial Intelligence*, pages 40–62. Springer-Verlag, 2000.
13. Deepak Kapur and Hantao Zhang. An overview of rewrite rule laboratory (RRL). *Computers & Mathematics with Applications*, 29(2):91–114, 1995.
14. Hélène Kirchner and Pierre-Etienne Moreau. Promoting rewriting to a programming language: A compiler for nondeterministic rewrite programs in associative-commutative theories. *Journal of Functional Programming*, 11(2):207–251, 2001.
15. Keiichirou Kusakari and Yoshihito Toyama. On proving AC-termination by AC-dependency pairs. *IEICE Transactions on Information and Systems*, E84-D(5):604–612, 2001.
16. Claude Marché. Normalized rewriting: An alternative to rewriting modulo a set of equations. *Journal of Symbolic Computation*, 21(3):253–288, 1996.
17. Claude Marché and Xavier Urbain. Modular and incremental proofs of AC-termination. *Journal of Symbolic Computation*, 38(1):873–897, 2004.
18. Gerald E. Peterson and Mark E. Stickel. Complete Sets of Reductions for Some Equational Theories. *Journal of the Association for Computing Machinery*, 28(2):233–264, 1981.
19. Albert Rubio. A fully syntactic AC-RPO. *Information and Computation*, 178(2):515–533, 2002.
20. Christian Stein. Das Dependency Pair Framework zur automatischen Terminierungsanalyse von Termersetzung modulo Gleichungen. Diplomarbeit, Department of Computer Science, Rheinisch-Westfälische Technische Hochschule, Germany, 2006.

21. René Thiemann, Jürgen Giesl, and Peter Schneider-Kamp. Improved modular termination proofs using dependency pairs. In David A. Basin and Michaël Rusinowitch, editors, *Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR '04)*, volume 3097 of *Lecture Notes in Computer Science*, pages 75–90. Springer-Verlag, 2004.

A Examples

A.1 Natural numbers 1

The following example was already erroneously used in [5], but since it contains a collapsing equation their approach is not applicable.

$$\begin{aligned}\mathcal{R}: (x+y)-y &\to x\\ 0 \div \mathsf{s}(y) &\to 0\\ \mathsf{s}(x) \div \mathsf{s}(y) &\to \mathsf{s}((x-y) \div \mathsf{s}(y))\end{aligned} \qquad \begin{aligned}\mathcal{E}: \quad u+0 &\approx u\\ u+\mathsf{s}(v) &\approx \mathsf{s}(u+v)\\ u+(v+w) &\approx (u+v)+w\\ u+v &\approx v+u\end{aligned}$$

Then, $(\mathcal{R}, \emptyset, \mathcal{E})$ is an equational system and we want to show that $\xrightarrow{\emptyset}_{\mathcal{E}\setminus\mathcal{R}}$ is terminating.

There is only one dependency pair:

$$\mathsf{s}(x) \mathbin{\dot{\div}^\sharp} \mathsf{s}(y) \to (x-y) \mathbin{\dot{\div}^\sharp} \mathsf{s}(y)$$

Using Theorem 22, we obtain the constraints

$$\begin{aligned}\mathsf{s}(x) \mathbin{\dot{\div}^\sharp} \mathsf{s}(y) &\succ (x-y) \mathbin{\dot{\div}^\sharp} \mathsf{s}(y)\\ (x+y)-y &\gtrsim x\\ 0 \div \mathsf{s}(y) &\gtrsim 0\\ \mathsf{s}(x) \div \mathsf{s}(y) &\gtrsim \mathsf{s}((x-y) \div \mathsf{s}(y))\end{aligned} \qquad \begin{aligned} u+0 &\sim u\\ u+\mathsf{s}(v) &\sim \mathsf{s}(u+v)\\ u+(v+w) &\sim (u+v)+w\\ u+v &\sim v+u\end{aligned}$$

The constraints are satisfied by the polynomial order induced by

$$\begin{aligned}\mathcal{P}ol(0) &= 0\\ \mathcal{P}ol(\mathsf{s}(x)) &= x+1\\ \mathcal{P}ol(x+y) &= x+y\\ \mathcal{P}ol(x-y) &= x\\ \mathcal{P}ol(x \div y) &= x\\ \mathcal{P}ol(x \mathbin{\dot{\div}^\sharp} y) &= x\end{aligned}$$

A.2 Natural numbers 2

This example considers a distributive rule on natural numbers.

$$\mathcal{R}: (x+y)\cdot z \to x\cdot z + y\cdot z \qquad \begin{aligned}\mathcal{E}: \quad u+0 &\approx u\\ u+\mathsf{s}(v) &\approx \mathsf{s}(u+v)\\ u+(v+w) &\approx (u+v)+w\\ u+v &\approx v+u\end{aligned}$$

Then $\to_{\mathcal{E}\setminus\mathcal{R}}$ is not terminating, as already shown in Example 6. As shown in Example 11 we obtain the equational system

$$\begin{aligned}\mathcal{R}: \quad (x+y)\cdot z &\to x\cdot z + y\cdot z\\ \mathcal{E}_1: u+(v+w) &\approx (u+v)+w\\ u+v &\approx v+u\end{aligned} \qquad \begin{aligned}\mathcal{S}: \quad u+0 &\to u\\ u+\mathsf{s}(v) &\to \mathsf{s}(u+v)\\ (u+\mathsf{s}(v))+w &\to \mathsf{s}(u+v)+w\end{aligned}$$

and we want to show that $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1\backslash\mathcal{R}}$ is terminating.

We get the dependency pairs

$$\begin{aligned}(x+y) \mathbin{.^\sharp} z &\to x \mathbin{.^\sharp} z\\ (x+y) \mathbin{.^\sharp} z &\to y \mathbin{.^\sharp} z\end{aligned}$$

Using Theorem 22, we obtain the constraints

$$\begin{aligned}(x+y) \mathbin{.^\sharp} z &\succ x \mathbin{.^\sharp} z\\ (x+y) \mathbin{.^\sharp} z &\succ y \mathbin{.^\sharp} z\\ (x+y)\cdot z &\gtrsim x\cdot z + y\cdot z\end{aligned}\qquad\begin{aligned}u+\mathsf{0} &\gtrsim u\\ u+\mathsf{s}(v) &\gtrsim \mathsf{s}(u+v)\\ (u+\mathsf{s}(v))+w &\gtrsim \mathsf{s}(u+v)+w\\ u+(v+w) &\sim (u+v)+w\\ u+v &\sim v+u\end{aligned}$$

These constraints are satisfied by the AC-RPO of [19] with the precedence $\mathbin{.^\sharp} \succ_\mathcal{F} \cdot \succ_\mathcal{F} + \succ_\mathcal{F} \mathsf{s}$.

Alternatively, the polynomial order induced by

$$\begin{aligned}\mathcal{P}ol(\mathsf{0}) &= 0\\ \mathcal{P}ol(\mathsf{s}(x)) &= x\\ \mathcal{P}ol(x+y) &= x+y+1\\ \mathcal{P}ol(x \mathbin{.^\sharp} y) &= x\\ \mathcal{P}ol(x\cdot y) &= x\end{aligned}$$

can be used.

A.3 Natural numbers 3

Here, we consider a gcd-function on natural numbers. As in Example A.1, we do not need to split $\mathcal{E}$.

$$\mathcal{R}:\quad\begin{aligned}\mathsf{gcd}(x,\mathsf{0}) &\to x\\ \mathsf{gcd}(\mathsf{0},\mathsf{s}(y)) &\to \mathsf{s}(y)\\ \mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(x+y)) &\to \mathsf{gcd}(\mathsf{s}(x),y)\\ \mathsf{gcd}(\mathsf{s}(x+y),\mathsf{s}(y)) &\to \mathsf{gcd}(x,\mathsf{s}(y))\end{aligned}\qquad\mathcal{E}:\quad\begin{aligned}u+\mathsf{0} &\approx u\\ u+\mathsf{s}(v) &\approx \mathsf{s}(u+v)\\ u+(v+w) &\approx (u+v)+w\\ u+v &\approx v+u\end{aligned}$$

We get the dependency pairs

$$\begin{aligned}\mathsf{gcd}^\sharp(\mathsf{s}(x),\mathsf{s}(x+y)) &\to \mathsf{gcd}^\sharp(\mathsf{s}(x),y)\\ \mathsf{gcd}^\sharp(\mathsf{s}(x+y),\mathsf{s}(y)) &\to \mathsf{gcd}^\sharp(x,\mathsf{s}(y))\end{aligned}$$

Using Theorem 22, we obtain the constraints

$$\begin{aligned}\mathsf{gcd}^\sharp(\mathsf{s}(x),\mathsf{s}(x+y)) &\succ \mathsf{gcd}^\sharp(\mathsf{s}(x),y)\\ \mathsf{gcd}^\sharp(\mathsf{s}(x+y),\mathsf{s}(y)) &\succ \mathsf{gcd}^\sharp(x,\mathsf{s}(y))\\ \mathsf{gcd}(x,\mathsf{0}) &\gtrsim x\\ \mathsf{gcd}(\mathsf{0},\mathsf{s}(y)) &\gtrsim \mathsf{s}(y)\\ \mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(x+y)) &\gtrsim \mathsf{gcd}(\mathsf{s}(x),y)\\ \mathsf{gcd}(\mathsf{s}(x+y),\mathsf{s}(y)) &\gtrsim \mathsf{gcd}(x,\mathsf{s}(y))\end{aligned}\qquad\begin{aligned}u+\mathsf{0} &\sim u\\ u+\mathsf{s}(v) &\sim \mathsf{s}(u+v)\\ u+(v+w) &\sim (u+v)+w\\ u+v &\sim v+u\end{aligned}$$

The constraints are satisfied by the polynomial order induced by

$$\begin{aligned}\mathcal{P}ol(\mathsf{0}) &= 0\\ \mathcal{P}ol(\mathsf{s}(x)) &= x+1\\ \mathcal{P}ol(x+y) &= x+y\\ \mathcal{P}ol(\mathsf{gcd}(x,y)) &= x+y\\ \mathcal{P}ol(\mathsf{gcd}^\sharp(x,y)) &= x+y\end{aligned}$$

A.4 Integers

Addition on integers can be defined as follows.

$$\mathcal{R}: \begin{array}{rl} x+\mathsf{0} & \to x\\ x+\mathsf{s}(y) & \to \mathsf{s}(x+y)\\ x+\mathsf{p}(y) & \to \mathsf{p}(x+y)\end{array} \qquad \mathcal{E}: \begin{array}{l}\mathsf{p}(\mathsf{s}(u)) \approx \mathsf{s}(\mathsf{p}(u))\\ \mathsf{p}(\mathsf{s}(u)) \approx u\end{array}$$

Then $\to_{\mathcal{E}\setminus\mathcal{R}}$ is not terminating since

$$\mathsf{0}+\mathsf{0} \sim_{\mathcal{E}} \mathsf{0}+\mathsf{p}(\mathsf{s}(\mathsf{0})) \to_{\mathcal{R}} \mathsf{p}(\mathsf{0}+\mathsf{s}(\mathsf{0})) \to_{\mathcal{R}} \mathsf{p}(\mathsf{s}(\mathsf{0}+\mathsf{0})) \sim_{\mathcal{E}} \ldots$$

is an infinite reduction.

We split $\mathcal{E}$ into $\mathcal{E}_1 = \{\mathsf{p}(\mathsf{s}(u)) \approx \mathsf{s}(\mathsf{p}(u))\}$ and $\mathcal{E}_2 = \{\mathsf{p}(\mathsf{s}(u)) \approx u\}$. Then $\mathcal{E}_2$ can be completed into $\mathcal{S} = \{\mathsf{p}(\mathsf{s}(u)) \to u\}$, which is convergent modulo $\mathcal{E}_1$. Since we do not need to add extended rules to $\mathcal{S}$ (see [5, Example A.8]), we obtain the equational system

$$\mathcal{R}: \begin{array}{rl} x+\mathsf{0} & \to x\\ x+\mathsf{s}(y) & \to \mathsf{s}(x+y)\\ x+\mathsf{p}(y) & \to \mathsf{p}(x+y)\end{array} \qquad \begin{array}{ll}\mathcal{S}: & \mathsf{p}(\mathsf{s}(u)) \to u\\ \mathcal{E}_1: & \mathsf{p}(\mathsf{s}(u)) \approx \mathsf{s}(\mathsf{p}(u))\end{array}$$

and we want to show that $\overset{\mathcal{S}}{\to}_{\mathcal{E}_1\setminus\mathcal{R}}$ is terminating.

We get the dependency pairs

$$\begin{aligned}\mathsf{p}(x) +^\sharp y &\to x +^\sharp y\\ \mathsf{s}(x) +^\sharp y &\to x +^\sharp y\end{aligned}$$

Using Theorem 22, we obtain the constraints

$$\begin{array}{rl} \mathsf{p}(x) +^\sharp y & \succ x +^\sharp y\\ \mathsf{s}(x) +^\sharp y & \succ x +^\sharp y\\ \mathsf{p}(\mathsf{s}(u)) & \sim \mathsf{s}(\mathsf{p}(u))\end{array} \qquad \begin{array}{rl} x+\mathsf{0} & \gtrsim x\\ x+\mathsf{s}(y) & \gtrsim \mathsf{s}(x+y)\\ x+\mathsf{p}(y) & \gtrsim \mathsf{p}(x+y)\\ \mathsf{p}(\mathsf{s}(x)) & \gtrsim x\end{array}$$

These constraints are satisfied by an RPO with the non-strict precedence $+ \succ_{\mathcal{F}} \mathsf{s}$, $+ \succ_{\mathcal{F}} \mathsf{p}$ and $\mathsf{p} \sim_{\mathcal{F}} \mathsf{s}$.

A.5 Sets

Sets are built from the empty set $\emptyset$ and singleton sets $\langle\cdot\rangle$ using set union $\cup$. We have the following axioms:

$$\begin{aligned}\mathcal{E}\colon\ & u \cup (v \cup w) \approx (u \cup v) \cup w \\ & u \cup v \approx v \cup u \\ & u \cup \emptyset \approx u \\ & u \cup u \approx u\end{aligned}$$

We split $\mathcal{E}$ into $\mathcal{E}_1 = \{(u \cup v) \cup w \approx u \cup (v \cup w), u \cup v \approx v \cup u\}$ and (after adding an extended rule) $\mathcal{S} = \{u \cup \emptyset \to u, u \cup u \to u, (u \cup u) \cup v \to u \cup v\}$.

Consider a quicksort function that takes a set and returns a sorted list:

$$\begin{aligned}\mathcal{R}:\quad x < 0 &\to \mathsf{false} \\ 0 < \mathsf{s}(y) &\to \mathsf{true} \\ \mathsf{s}(x) < \mathsf{s}(y) &\to x < y \\ \mathsf{app}(\mathsf{nil}, y) &\to y \\ \mathsf{app}(\mathsf{cons}(m, x), y) &\to \mathsf{cons}(m, \mathsf{app}(x, y)) \\ \mathsf{low}(n, \emptyset) &\to \emptyset \\ \mathsf{low}(n, \langle m\rangle) &\to \mathsf{if}(m < n, m) \\ \mathsf{low}(n, x \cup y) &\to \mathsf{low}(n, x) \cup \mathsf{low}(n, y) \\ \mathsf{high}(n, \emptyset) &\to \emptyset \\ \mathsf{high}(n, \langle m\rangle) &\to \mathsf{if}(n < m, m) \\ \mathsf{high}(n, x \cup y) &\to \mathsf{high}(n, x) \cup \mathsf{high}(n, y) \\ \mathsf{if}(\mathsf{true}, m) &\to \langle m\rangle \\ \mathsf{if}(\mathsf{false}, m) &\to \emptyset \\ \mathsf{qsort}(\emptyset) &\to \mathsf{nil} \\ \mathsf{qsort}(\langle m\rangle) &\to \mathsf{cons}(m, \mathsf{nil}) \\ \mathsf{qsort}(\langle n\rangle \cup y) &\to \mathsf{app}(\mathsf{qsort}(\mathsf{low}(n, y)), \mathsf{cons}(n, \mathsf{qsort}(\mathsf{high}(n, y))))\end{aligned}$$

We get the dependency pairs

$$\mathsf{s}(x) <^\sharp \mathsf{s}(y) \to x <^\sharp y \tag{1}$$
$$\mathsf{app}^\sharp(\mathsf{cons}(m, x), y) \to \mathsf{app}^\sharp(x, y) \tag{2}$$
$$\mathsf{low}^\sharp(n, \langle m\rangle) \to \mathsf{if}^\sharp(m < n, m) \tag{3}$$
$$\mathsf{low}^\sharp(n, \langle m\rangle) \to m <^\sharp n \tag{4}$$
$$\mathsf{low}^\sharp(n, x \cup y) \to \mathsf{low}^\sharp(n, x) \tag{5}$$
$$\mathsf{low}^\sharp(n, x \cup y) \to \mathsf{low}^\sharp(n, y) \tag{6}$$
$$\mathsf{high}^\sharp(n, \langle m\rangle) \to \mathsf{if}^\sharp(n < m, m) \tag{7}$$
$$\mathsf{high}^\sharp(n, \langle m\rangle) \to m <^\sharp n \tag{8}$$
$$\mathsf{high}^\sharp(n, x \cup y) \to \mathsf{high}^\sharp(n, x) \tag{9}$$
$$\mathsf{high}^\sharp(n, x \cup y) \to \mathsf{high}^\sharp(n, y) \tag{10}$$

$$\mathsf{qsort}^\sharp(\langle n\rangle \cup y) \to \mathsf{app}^\sharp(\mathsf{qsort}(\mathsf{low}(n,y)), \mathsf{cons}(n, \mathsf{qsort}(\mathsf{high}(n,y)))) \quad (11)$$
$$\mathsf{qsort}^\sharp(\langle n\rangle \cup y) \to \mathsf{qsort}^\sharp(\mathsf{low}(n,y)) \quad (12)$$
$$\mathsf{qsort}^\sharp(\langle n\rangle \cup y) \to \mathsf{low}^\sharp(n,y) \quad (13)$$
$$\mathsf{qsort}^\sharp(\langle n\rangle \cup y) \to \mathsf{qsort}^\sharp(\mathsf{high}(n,y)) \quad (14)$$
$$\mathsf{qsort}^\sharp(\langle n\rangle \cup y) \to \mathsf{high}^\sharp(n,y) \quad (15)$$

The estimated dependency graph contains 5 SCCs:

(1)
(2)
(5), (6)
(9), (10)
(12), (14)

In order to satisfy the constraints of Theorem 39 for all SCCs simultaneously and such that the resulting DP problems do not contain any further cycles, we obtain the constraints

$$\begin{aligned}
\mathsf{s}(x) <^\sharp \mathsf{s}(y) &\succ x <^\sharp y\\
\mathsf{app}^\sharp(\mathsf{cons}(m,x),y) &\succ \mathsf{app}^\sharp(x,y)\\
\mathsf{low}^\sharp(n,x\cup y) &\succ \mathsf{low}^\sharp(n,x)\\
\mathsf{low}^\sharp(n,x\cup y) &\succ \mathsf{low}^\sharp(n,y)\\
\mathsf{high}^\sharp(n,x\cup y) &\succ \mathsf{high}^\sharp(n,x)\\
\mathsf{high}^\sharp(n,x\cup y) &\succ \mathsf{high}^\sharp(n,y)\\
\mathsf{qsort}^\sharp(\langle n\rangle \cup y) &\succ \mathsf{qsort}^\sharp(\mathsf{low}(n,y))\\
\mathsf{qsort}^\sharp(\langle n\rangle \cup y) &\succ \mathsf{qsort}^\sharp(\mathsf{high}(n,y))\\
x < \mathsf{0} &\gtrsim \mathsf{false}\\
\mathsf{0} < \mathsf{s}(y) &\gtrsim \mathsf{true}\\
\mathsf{s}(x) < \mathsf{s}(y) &\gtrsim x < y\\
\mathsf{app}(\mathsf{nil},y) &\gtrsim y\\
\mathsf{app}(\mathsf{cons}(m,x),y) &\gtrsim \mathsf{cons}(m,\mathsf{app}(x,y))\\
\mathsf{low}(n,\emptyset) &\gtrsim \emptyset\\
\mathsf{low}(n,\langle m\rangle) &\gtrsim \mathsf{if}(m<n,m)\\
\mathsf{low}(n,x\cup y) &\gtrsim \mathsf{low}(n,x)\cup\mathsf{low}(n,y)\\
\mathsf{high}(n,\emptyset) &\gtrsim \emptyset\\
\mathsf{high}(n,\langle m\rangle) &\gtrsim \mathsf{if}(n<m,m)\\
\mathsf{high}(n,x\cup y) &\gtrsim \mathsf{high}(n,x)\cup\mathsf{high}(n,y)\\
\mathsf{if}(\mathsf{true},m) &\gtrsim \langle m\rangle\\
\mathsf{if}(\mathsf{false},m) &\gtrsim \emptyset\\
\mathsf{qsort}(\emptyset) &\gtrsim \mathsf{nil}\\
\mathsf{qsort}(\langle m\rangle) &\gtrsim \mathsf{cons}(m,\mathsf{nil})\\
\mathsf{qsort}(\langle n\rangle \cup y) &\gtrsim \mathsf{app}(\mathsf{qsort}(\mathsf{low}(n,y)),\mathsf{cons}(n,\mathsf{qsort}(\mathsf{high}(n,y))))\\
u\cup\emptyset &\gtrsim u
\end{aligned}$$

$$\begin{aligned} u \cup u &\gtrsim u \\ (u \cup u) \cup v &\gtrsim u \cup v \\ (u \cup v) \cup w &\sim u \cup (v \cup w) \\ u \cup v &\sim v \cup u \end{aligned}$$

We apply an argument filtering that collapses $\mathsf{low}, \mathsf{low}^\sharp, \mathsf{high}$, and $\mathsf{high}^\sharp$ to their last argument and eliminates the first argument of if and cons.

The filtered constraints are

$$\begin{aligned} \mathsf{s}(x) <^\sharp \mathsf{s}(y) &\succ x <^\sharp y \\ \mathsf{app}^\sharp(\mathsf{cons}(x), y) &\succ \mathsf{app}^\sharp(x, y) \\ x \cup y &\succ x \\ x \cup y &\succ y \\ x \cup y &\succ x \\ x \cup y &\succ y \\ \mathsf{qsort}^\sharp(\langle n \rangle \cup y) &\succ \mathsf{qsort}^\sharp(y) \\ \mathsf{qsort}^\sharp(\langle n \rangle \cup y) &\succ \mathsf{qsort}^\sharp(y) \\ x < 0 &\gtrsim \mathsf{false} \\ 0 < \mathsf{s}(y) &\gtrsim \mathsf{true} \\ \mathsf{s}(x) < \mathsf{s}(y) &\gtrsim x < y \\ \mathsf{app}(\mathsf{nil}, y) &\gtrsim y \\ \mathsf{app}(\mathsf{cons}(x), y) &\gtrsim \mathsf{cons}(\mathsf{app}(x, y)) \\ \emptyset &\gtrsim \emptyset \\ \langle m \rangle &\gtrsim \mathsf{if}(m) \\ x \cup y &\gtrsim x \cup y \\ \emptyset &\gtrsim \emptyset \\ \langle m \rangle &\gtrsim \mathsf{if}(m) \\ x \cup y &\gtrsim x \cup y \\ \mathsf{if}(m) &\gtrsim \langle m \rangle \\ \mathsf{if}(m) &\gtrsim \emptyset \\ \mathsf{qsort}(\emptyset) &\gtrsim \mathsf{nil} \\ \mathsf{qsort}(\langle m \rangle) &\gtrsim \mathsf{cons}(\mathsf{nil}) \\ \mathsf{qsort}(\langle n \rangle \cup y) &\gtrsim \mathsf{app}(\mathsf{qsort}(y), \mathsf{cons}(\mathsf{qsort}(y))) \\ u \cup \emptyset &\gtrsim u \\ u \cup u &\gtrsim u \\ (u \cup u) \cup v &\gtrsim u \cup v \\ (u \cup v) \cup w &\sim u \cup (v \cup w) \\ u \cup v &\sim v \cup u \end{aligned}$$

These constraints are satisfied by an AC-RPO with the precedence $\mathsf{qsort} \succ_\mathcal{F} \mathsf{app} \succ_\mathcal{F} \mathsf{cons} \succ_\mathcal{F} \mathsf{nil}$, $< \succ_\mathcal{F} \mathsf{true} \succ_\mathcal{F} \mathsf{false}$, $\mathsf{if} \succ_\mathcal{F} \emptyset$, and $\mathsf{if} \sim_\mathcal{F} \langle\cdot\rangle$.

A.6 CCS

This examples considers a subset of Milner's Calculus of Communicating Systems (CCS). Processes are built using the terminated process $\mathsf{0}$, the prefixing operator ".", nondeterministic choice $+$, and parallel composition $|$. The complementary

action of an action a is denoted by $\overline{a}$. The silent action is denoted by τ. Properties of processes are modelled using the following ES.

$$\begin{aligned}\mathcal{E}: \qquad p + q &\approx q + p\\ p + (q + r) &\approx (p + q) + r\\ p + p &\approx p\\ p + \mathsf{0} &\approx p\\ p \mid q &\approx q \mid p\\ p \mid (q \mid r) &\approx (p \mid q) \mid r\\ p \mid \mathsf{0} &\approx p\\ \overline{\overline{a}} &\approx a\end{aligned}$$

This ES can be split into an ES $\mathcal{E}_1$ and a TRS $\mathcal{S}$ which is complete and convergent modulo $\mathcal{E}_1$ as follows.

$$\begin{aligned}\mathcal{E}_1: \qquad p + q &\approx q + p\\ p + (q + r) &\approx (p + q) + r\\ p \mid q &\approx q \mid p\\ p \mid (q \mid r) &\approx (p \mid q) \mid r\end{aligned} \qquad \begin{aligned}\mathcal{S}: \qquad p + p &\to p\\ (p + p) + q &\to p + q\\ p + \mathsf{0} &\to p\\ p \mid \mathsf{0} &\to p\\ \overline{\overline{a}} &\to a\end{aligned}$$

Here, the second rule in $\mathcal{S}$ is the extension of the first one.

Now, we can define a function eval that computes a possible trace of a process description.

$$\begin{aligned}\mathcal{R}: \qquad \mathsf{eval}(\mathsf{0}) &\to \mathsf{0}\\ \mathsf{eval}(a.p) &\to a.\mathsf{eval}(p)\\ \mathsf{eval}(p + q) &\to \mathsf{if}_+(\mathsf{eval}(p))\\ \mathsf{if}_+(a.p) &\to a.\mathsf{eval}(p)\\ \mathsf{eval}(p \mid q) &\to \mathsf{if}_|(\mathsf{eval}(p), q)\\ \mathsf{if}_|(a.p, q) &\to a.\mathsf{eval}(p \mid q)\\ \mathsf{eval}(p \mid q) &\to \mathsf{if}_\tau(\mathsf{eval}(p), \mathsf{eval}(q))\\ \mathsf{if}_\tau(a.p, \overline{a}.q) &\to \tau.\mathsf{eval}(p \mid q)\end{aligned}$$

For showing termination of $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1 \setminus \mathcal{R}}$, we obtain the following dependency pairs.

$$\begin{aligned}\mathsf{eval}^\sharp(a.p) &\to \mathsf{eval}^\sharp(p)\\ \mathsf{eval}^\sharp(p + q) &\to \mathsf{if}^\sharp_+(\mathsf{eval}(p))\\ \mathsf{eval}^\sharp(p + q) &\to \mathsf{eval}^\sharp(p)\\ \mathsf{if}^\sharp(a.p) &\to \mathsf{eval}^\sharp(p)\\ \mathsf{eval}^\sharp(p \mid q) &\to \mathsf{if}^\sharp_|(\mathsf{eval}(p), q)\\ \mathsf{eval}^\sharp(p \mid q) &\to \mathsf{eval}^\sharp(p)\\ \mathsf{if}^\sharp_|(a.p, q) &\to \mathsf{eval}^\sharp(p \mid q)\\ \mathsf{eval}^\sharp(p \mid q) &\to \mathsf{if}^\sharp_\tau(\mathsf{eval}(p), \mathsf{eval}(q))\\ \mathsf{eval}^\sharp(p \mid q) &\to \mathsf{eval}^\sharp(p)\\ \mathsf{eval}^\sharp(p \mid q) &\to \mathsf{eval}^\sharp(q))\end{aligned}$$

These dependency pairs form an SCC of the estimated dependency graph. Using the polynomial order induced by

$$\begin{aligned}
\mathcal{P}ol(\mathsf{0}) &= 1\\
\mathcal{P}ol(\tau) &= 0\\
\mathcal{P}ol(\overline{x}) &= x\\
\mathcal{P}ol(x.y) &= y+2\\
\mathcal{P}ol(x+y) &= 2xy+2x+2y+1\\
\mathcal{P}ol(x \mid y) &= x+y+1\\
\mathcal{P}ol(\mathsf{eval}(x)) &= x\\
\mathcal{P}ol(\mathsf{if}_+(x)) &= x\\
\mathcal{P}ol(\mathsf{if}_\mid(x,y)) &= x+y+1\\
\mathcal{P}ol(\mathsf{if}_\tau(x,y)) &= x+y\\
\mathcal{P}ol(\mathsf{eval}^\sharp(x)) &= x\\
\mathcal{P}ol(\mathsf{if}^\sharp_+(x)) &= x\\
\mathcal{P}ol(\mathsf{if}^\sharp_\mid(x,y)) &= x+y\\
\mathcal{P}ol(\mathsf{if}^\sharp_\tau(x,y)) &= x+y
\end{aligned}$$

the following constraints are satisfied:

$$\begin{aligned}
\mathsf{eval}^\sharp(a.p) &\succ \mathsf{eval}^\sharp(p)\\
\mathsf{eval}^\sharp(p+q) &\succ \mathsf{if}^\sharp_+(\mathsf{eval}(p))\\
\mathsf{eval}^\sharp(p+q) &\succ \mathsf{eval}^\sharp(p)\\
\mathsf{if}^\sharp(a.p) &\succ \mathsf{eval}^\sharp(p)\\
\mathsf{eval}^\sharp(p \mid q) &\succ \mathsf{if}^\sharp_\mid(\mathsf{eval}(p),q)\\
\mathsf{eval}^\sharp(p \mid q) &\succ \mathsf{eval}^\sharp(p)\\
\mathsf{if}^\sharp_\mid(a.p,q) &\succ \mathsf{eval}^\sharp(p \mid q)\\
\mathsf{eval}^\sharp(p \mid q) &\succ \mathsf{if}^\sharp_\tau(\mathsf{eval}(p),\mathsf{eval}(q))\\
\mathsf{eval}^\sharp(p \mid q) &\succ \mathsf{eval}^\sharp(p)\\
\mathsf{eval}^\sharp(p \mid q) &\succ \mathsf{eval}^\sharp(q)\\
\mathsf{eval}(\mathsf{0}) &\gtrsim \mathsf{0}\\
\mathsf{eval}(a.p) &\gtrsim a.\mathsf{eval}(p)\\
\mathsf{eval}(p+q) &\gtrsim \mathsf{if}_+(\mathsf{eval}(p))\\
\mathsf{if}_+(a.p) &\gtrsim a.\mathsf{eval}(p)\\
\mathsf{eval}(p \mid q) &\gtrsim \mathsf{if}_\mid(\mathsf{eval}(p),q)\\
\mathsf{if}_\mid(a.p,q) &\gtrsim a.\mathsf{eval}(p \mid q)\\
\mathsf{eval}(p \mid q) &\gtrsim \mathsf{if}_\tau(\mathsf{eval}(p),\mathsf{eval}(q))\\
\mathsf{if}_\tau(a.p,\overline{a}.q) &\gtrsim \tau.\mathsf{eval}(p \mid q)
\end{aligned}$$

$$
\begin{aligned}
p + p &\gtrsim p \\
(p + p) + q &\gtrsim p + q \\
p + \mathsf{0} &\gtrsim p \\
p \mid \mathsf{0} &\gtrsim p \\
\overline{\overline{a}} &\gtrsim a \\
p + q &\sim q + p \\
p + (q + r) &\sim (p + q) + r \\
p \mid q &\sim q \mid p \\
p \mid (q \mid r) &\sim (p \mid q) \mid r
\end{aligned}
$$

According to Theorem 39, all dependency pairs can be deleted and termination has been shown.

A.7 Sequent Calculus

We consider the propositional sequent calculus for formulas built from $\wedge$ and $\neg$. Sequents are built from two sets of formulas using $\Longrightarrow$. Set of formulas are built from the empty set $\emptyset$ using "," to add a formula. Similarly, sets of sequents are built using $\Box$ and $\bullet$.

From the ES

$$
\begin{aligned}
\mathcal{E}\colon \quad u, (v, w) &\approx v, (u, w) \\
u, (u, v) &\approx u, v \\
u \bullet (v \bullet w) &\approx v \bullet (u \bullet w) \\
u \bullet (u \bullet v) &\approx u \bullet v
\end{aligned}
$$

we obtain

$$
\begin{aligned}
\mathcal{E}_1\colon \quad u, (v, w) &\approx v, (u, w) \\
u \bullet (v \bullet w) &\approx v \bullet (u \bullet w)
\end{aligned}
\qquad
\begin{aligned}
\mathcal{S}\colon \quad u, (u, v) &\to u, v \\
u \bullet (u \bullet v) &\to u \bullet v
\end{aligned}
$$

Now the sequent calculus rules for $\wedge$ and $\neg$ are modelled by the function eval defined by the following TRS.

$$
\begin{aligned}
\mathcal{R}\colon \quad \mathsf{eval}(\Box) &\to \Box \\
\mathsf{eval}((x, y \implies x, z) \bullet s) &\to \mathsf{eval}(s) \\
\mathsf{eval}((\neg x, y \implies z) \bullet s) &\to \mathsf{eval}((y \implies x, z) \bullet s) \\
\mathsf{eval}((x \implies \neg y, z) \bullet s) &\to \mathsf{eval}((y, x \implies z) \bullet s) \\
\mathsf{eval}((x \wedge y, z \implies z') \bullet s) &\to \mathsf{eval}((x, (y, z) \implies z') \bullet s) \\
\mathsf{eval}((x \implies y \wedge z, z') \bullet s) &\to \mathsf{eval}((x \implies y, z') \bullet ((x \implies z, z') \bullet s))
\end{aligned}
$$

Then $\mathcal{R}$ has five dependency pairs, which form an SCC in the estimated dependency graph.

$$
\begin{aligned}
\mathsf{eval}^\sharp((x, y \implies x, z) \bullet s) &\to \mathsf{eval}^\sharp(s) \\
\mathsf{eval}^\sharp((\neg x, y \implies z) \bullet s) &\to \mathsf{eval}^\sharp((y \implies x, z) \bullet s) \\
\mathsf{eval}^\sharp((x \implies \neg y, z) \bullet s) &\to \mathsf{eval}^\sharp((y, x \implies z) \bullet s) \\
\mathsf{eval}^\sharp((x \wedge y, z \implies z') \bullet s) &\to \mathsf{eval}^\sharp((x, (y, z) \implies z') \bullet s) \\
\mathsf{eval}^\sharp((x \implies y \wedge z, z') \bullet s) &\to \mathsf{eval}^\sharp((x \implies y, z') \bullet ((x \implies z, z') \bullet s))
\end{aligned}
$$

From $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{E}_1$ we obtain the following constraints:

$$\begin{aligned}
\mathsf{eval}(\Box) &\gtrsim \Box \\
\mathsf{eval}((x, y \implies x, z) \bullet s) &\gtrsim \mathsf{eval}(s) \\
\mathsf{eval}((\neg x, y \implies z) \bullet s) &\gtrsim \mathsf{eval}((y \implies x, z) \bullet s) \\
\mathsf{eval}((x \implies \neg y, z) \bullet s) &\gtrsim \mathsf{eval}((y, x \implies z) \bullet s) \\
\mathsf{eval}((x \wedge y, z \implies z') \bullet s) &\gtrsim \mathsf{eval}((x, (y, z) \implies z') \bullet s) \\
\mathsf{eval}((x \implies y \wedge z, z') \bullet s) &\gtrsim \mathsf{eval}((x \implies y, z') \bullet ((x \implies z, z') \bullet s)) \\
u, (u, v) &\gtrsim u, v \\
u \bullet (u \bullet v) &\gtrsim u \bullet v \\
u, (v, w) &\sim v, (u, w) \\
u \bullet (v \bullet w) &\sim v \bullet (u \bullet w)
\end{aligned}$$

We first apply the polynomial order induced by

$$\begin{aligned}
\mathcal{P}ol(x \wedge y) &= xy + x + y + 1 \\
\mathcal{P}ol(\neg x) &= x \\
\mathcal{P}ol(x \implies y) &= xy + x + y \\
\mathcal{P}ol(x, y) &= xy + x + y \\
\mathcal{P}ol(\Box) &= 0 \\
\mathcal{P}ol(x \bullet y) &= x + y \\
\mathcal{P}ol(\mathsf{eval}(x)) &= 0 \\
\mathcal{P}ol(\mathsf{eval}^\sharp(x)) &= x + 1
\end{aligned}$$

Then the constraints from $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{E}_1$ are satisfied. Additionally, the fourth and fifth dependency pair are strictly decreasing, while the remaining dependency pairs are weakly decreasing. Using Theorem 39, the fourth and fifth dependency pair may thus be deleted.

Next, we apply the polynomial order induced by

$$\begin{aligned}
\mathcal{P}ol(x \wedge y) &= 0 \\
\mathcal{P}ol(\neg x) &= x + 1 \\
\mathcal{P}ol(x \implies y) &= x + y \\
\mathcal{P}ol(x, y) &= x + y \\
\mathcal{P}ol(\Box) &= 0 \\
\mathcal{P}ol(x \bullet y) &= x + y \\
\mathcal{P}ol(\mathsf{eval}(x)) &= 0 \\
\mathcal{P}ol(\mathsf{eval}^\sharp(x)) &= x
\end{aligned}$$

Then the constraints from $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{E}_1$ are again satisfied, and the second and third dependency pair are strictly decreasing, while the first dependency pair is

weakly decreasing. After application of Theorem 39, only the first dependency pair is left.

The first dependency pair is now strictly decreasing if we use the polynomial order induced by

$$\begin{aligned}
\mathcal{P}ol(x \wedge y) &= 0\\
\mathcal{P}ol(\neg x) &= 0\\
\mathcal{P}ol(x \implies y) &= 0\\
\mathcal{P}ol(x, y) &= 0\\
\mathcal{P}ol(\Box) &= 0\\
\mathcal{P}ol(x \bullet y) &= y + 1\\
\mathcal{P}ol(\mathsf{eval}(x)) &= 0\\
\mathcal{P}ol(\mathsf{eval}^\sharp(x)) &= x
\end{aligned}$$

which also satisfies the constraints from $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{E}_1$. This concludes the proof of the termination of $\xrightarrow{\mathcal{S}}_{\mathcal{E}_1 \setminus \mathcal{R}}$.