Parametrised dependency pairs for a general form of termination in annotated rewriting

Fer-Jan de Vries
Computer Science Department, University of Leicester

Abstract. We define a general form of termination, α-termination, for term rewrite systems with an annotation \( \alpha \). Standard termination and top-termination are particular instances. We present a variation on the dependency pair criterion of Arts and Giesl and prove that an \( \alpha \)-annotated term rewrite system is \( \alpha \)-terminating if and only there are no chains of \( \alpha \)-dependency pairs. The \( \alpha \)-dependency graphs can be analysed in very much the same way as standard dependency graphs, thanks to Hirokawa and Middeldorp’s Subterm Criterion as we demonstrate on some examples. The \( \alpha \)-dependency pair criterion generalises both the dependency pair criterion for termination and the strong dependency pair criterion for strong convergence.

1 Introduction

In this paper we will present a parametrised dependency pair method for automatically proving \( \alpha \)-termination, a parametrised form of termination generalising both standard termination and top-termination\(^1\).

The standard dependency pair method \([1, 2]\) of Giesl and Arts for proving termination of term rewrite systems consists of two components. The first component transforms a term rewrite system \( \mathcal{R} \) by extending it with fresh function symbols (for each defined function symbol \( f \) a new symbol \( f^\# \) is added) and a number of new rewrite rules (of the form \( t^\# \rightarrow s^\# \) where \( t^\# \) and \( s^\# \) are terms containing only one new symbol at the root of the term) called dependency pairs. A beautiful theorem Arts and Giesl then says that a finite TRS \( \mathcal{R} \) is terminating if and only if there are no infinite \( \mathcal{R} \)-chains of dependency pairs of \( \mathcal{R} \).

The second component then tries to decide whether there are no infinite \( \mathcal{R} \)-chains of dependency pairs of \( \mathcal{R} \) by analysing the dependency graph \( \text{DP}(\mathcal{R}) \). This analysis can be performed with help of automated tools. The recent rapid development of such tools (among others: CiME \([4]\), TTT \([10]\) and AProVE \([8]\) and their underlying ideas in eg. \([7, 18, 12]\)) makes the dependency pair technique for proving termination increasingly effective.

In this paper we will generalise this method to \( \alpha \)-termination. We will define \( \alpha \)-dependency pairs for TRSs enriched with an annotation \( \alpha \). From these we construct \( \alpha \)-dependency graphs which can be analysed with virtually the same toolkit as has been developed for the standard dependency graphs. The main theorem of this paper then states that the \( \alpha \)-dependency graph \( \alpha \text{DP}(\mathcal{R}) \) of an annotated finite TRS \( (\mathcal{R}, \alpha) \) has no \( \alpha \)-chains if and only if \( (\mathcal{R}, \alpha) \) is \( \alpha \)-terminating. In case \( \alpha \) is

\(^1\) A TRS is top-terminating if and only if there are no reduction sequences of length \( \omega \) with infinitely many rewrites at the root \([6]\).
the empty annotation this comes down to the theorem of Arts and Giesl. And in case \( \alpha \) is the maximal annotation this comes down to the strong dependency pair method for strong convergence as stated in [9] or in equivalent but different terms, a \( \mu \)-dependency pair method for top-termination.

In term rewriting the idea of using an annotation of argument positions of function symbols to add some form of control to a rewriting strategy is well-known. Think of the use of eager annotation in lazy languages like Haskell and Clean, or conversely the use of lazy annotation in the eager languages like Lisp or Scheme. Termination properties of such annotated strategies have been studied by Lucas [17] in the setting of context sensitive rewriting systems. However in the context of lazy data structures termination is no longer the most natural property to study. For example, consider the Fibonacci TRS:

\[
\begin{align*}
0 + m & \rightarrow m \\
\mathit{s}(n) + m & \rightarrow \mathit{s}(n + m) \\
\mathit{fib}(n, m) & \rightarrow n : \mathit{fib}(m, n + m)
\end{align*}
\]

This TRS is not terminating. Yet it has termination-like properties that can be analysed with the \( \alpha \)-dependency pair method. In section 6 we will show that the TRS is not only \( \mu \)-terminating (that is top-terminating [6]) for the maximal annotation \( \mu \) but also that it has the more precise and stronger property \( \alpha' \)-terminating where the annotation \( \alpha' \) only annotates the second argument of the list constructor. As the TRS is left-linear, \( \alpha' \)-termination implies that there exist normalising mixed eager/lazy strategies in which arguments are evaluated eagerly unless they occur inside the second argument of a list constructor.

It follows from this example that a possible application of this work may be in strictness analysis.

2 Rewriting with annotation

In this section we will define rewriting with a simple form of annotation using some basic notation and terminology based on [3, 20]. A (finite) term rewriting system \( \mathcal{R} = (\mathcal{T}, R) \) consists of a set of terms \( \mathcal{T} \) constructed from a finite signature \( \mathcal{F} \) and a disjoint set of variables \( V \) together with a finite set \( R \) of rules between terms. A substitution is a function from a finite set of variables to \( \mathcal{T} \). We write \( t\sigma \) for the result of applying \( \sigma \) to \( t \).

**Definition 1.** Let \( (\mathcal{F}, \mathit{ar}) \) be a finite set of function symbols together with an arity function \( \mathit{ar} : \mathcal{F} \rightarrow \mathbb{N} \). An annotation for a signature \( (\mathcal{F}, \mathit{ar}) \) is a function \( \alpha : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N}) \) such that \( 1 \leq i \leq \mathit{ar}(f) \) for all \( i \in \alpha(f) \) and \( f \in \mathcal{F} \). The positions \( i \in \alpha(f) \) are called annotated. An annotated TRS\(^2\) \( (\mathcal{R}, \alpha) \) is a TRS \( \mathcal{R} \) together with an annotation \( \alpha \).

We will use the annotation \( \alpha \) to define \( \alpha \)-termination. This generalises both termination (there are no terms with infinite reductions) and top-termination [6] in our terminology a context sensitive rewrite system of Lucas is an annotated TRS together with the additional restriction that rewriting is not allowed in subterms at annotated argument positions.
(there are no terms with an infinite reduction of which infinitely many steps are reductions at the top (root)). The key idea, a notion of depth parametrised by $\alpha$, is borrowed from lambda calculus [14,15], where it was found that there are different meaningful ways of measuring the depth of subtrees in a tree. This can be best explained with pictures. As the pictures will demonstrate there are two special annotations which deserve their own notation, the empty annotation annotating nothing and the full annotation annotating everything:

**Definition 2.** Let the empty annotation $\epsilon$ be defined by $\epsilon(f) = \emptyset$ for all $f \in F$. And let the maximal annotation $\mu$ be defined by $\mu(f) = \{i \mid 1 \leq i \leq ar(f)\}$ for all $f \in F$.

If we have one binary function symbol $f$, then there are four possible annotations corresponding with the four different ways of drawing the underlying tree of the term $f(x,y)$ when annotated links are drawn horizontally and the others vertically.

\[ \begin{array}{cccc}
  x & \rightarrow & f & \rightarrow & y \\
  & \downarrow & & \downarrow & \\
  & x & & y & \\
  & & \rightarrow & & \rightarrow \\
  & & f & & f \\
  & & \rightarrow & & \rightarrow \\
  & & x & & y \\
\end{array} \]

\[ \begin{align*}
\epsilon(f) &= \emptyset \\
\alpha(f) &= \{1\} \\
\alpha(f) &= \{2\} \\
\mu(f) &= \{1,2\}
\end{align*} \]

Next we will define a notion of depth parametrised by the annotation $\alpha$. This concept is taken from lambda calculus [14].

**Definition 3.** The $\alpha$-depth of a subterm $s$ in a term $t$ in the annotated TRS $(\mathcal{R}, \alpha)$ is the number of argument positions annotated by $\alpha$ on the path from the root of $t$ to the root of $s$.

For example: if the binary function $f$ is annotated with $\alpha(f) = \{1\}$, then the picture of the tree of the term $f(f(y,z),x)$ is

\[ \begin{array}{cccc}
  f & \rightarrow & x \\
  & \downarrow & \\
  f & \rightarrow & z \\
  & \downarrow & \\
  y & & \rightarrow & \\
\end{array} \]

and we will say that the $\alpha$-depth of $x, y$ respectively $z$ in the term $f(f(y,z),x)$ is 0, 1 and 2. Given this $\alpha$ the prefix of $f(f(y,z),x)$ at $\alpha$-depth 0 is the context $f([\ ],x)$.

Note that the empty annotation $\epsilon$ puts all nodes of the tree representation of a term at the same $\epsilon$-depth 0 whereas the maximal annotation $\mu$ gives the usual tree representation of terms.

In this paper we make a distinction between “good” and “bad” reductions. Good are $\alpha$-terminating reductions that we will define in a moment. Bad will be all others. The underlying intuition is that when performing a good reduction we can be printing already during the reduction an increasingly better approximation of the final term, similar like printing the digits of the decimal expansion of $\sqrt{2}$ while computing it. This is possible if we know that there can be only a finite amount of activity at each layer of nodes at $\alpha$-depth. Hence we don’t mind infinite reductions as long as they continue to reveal more and more of the final term.
Definition 4. A $\alpha$-terminating reduction is a (possible infinite) reduction in which at most finitely many reductions take place at any $\alpha$-depth. A TRS $\mathcal{R}$ annotated by $\alpha$ is $\alpha$-terminating if all its reductions are $\alpha$-terminating.

This notion is based on an idea in our paper on infinite lambda calculus [14] where the notion of strongly convergent reduction of infinite term rewriting [13] was refined using depth as a parameter. Instantiated with the empty annotation Definition 4 redefines finite reduction. Instantiated with the maximal annotation one gets the definition of top-termination [6]. Note that although the definition of $\alpha$-termination is inspired by concepts from infinitary rewriting, we apply it in this paper only to reductions of length $\omega$ starting from finite terms in a finite TRS. While the limits of such $\alpha$-terminating reductions may not be finite, at any finite stage their terms are.

The next useful lemma generalises the fact that top-termination is a generally weaker property than termination:

Lemma 1. Let $\alpha$ and $\beta$ be two annotations for a TRS $\mathcal{R}$. If $\beta \supseteq \alpha$ (in the sense that $i \in \alpha(f) \Rightarrow i \in \beta(f)$ for all $0 \leq i \leq \text{arity}(f)$ for all $f \in \mathcal{F}$) then any $\alpha$-terminating reduction is $\beta$-terminating. $\square$

Let us look at a concrete example: the Fibonacci TRS from the introduction. Consider the annotation $\alpha'$ defined by $\alpha'(:) = \{2\}$ and $\alpha'(+) = \emptyset$. The following reductions are $\alpha'$-terminating:

$$
\text{fib}(1,1) \rightarrow 1 : \text{fib}(1,(1+1)) \rightarrow 1 : 1 : \text{fib}(1+1,1+(1+1)) \rightarrow \ldots
$$

$$
\text{fib}(1,1) \rightarrow 1 : \text{fib}(1,2) \rightarrow 1 : 1 : \text{fib}(2,3) \rightarrow 1 : 1 : 2 : \text{fib}(3,5) \rightarrow \ldots
$$

The former reduction is lazy in the sense that we don’t evaluate in the second argument of the list constructor. The latter more optimal reduction is interesting: it is not entirely lazy, as we do reduce certain terms in the second argument of the list operators. Note that for any annotation $\alpha$ such that $\alpha \supseteq \alpha'$ (for instance $\mu \supseteq \alpha'$ for the maximal annotation $\mu$) the weaker property of $\alpha$-termination holds for these reductions.

We will now define bad terms as terms with bad reductions:

Definition 5. An $\alpha$-bad term is a term which has an infinite reduction in which infinitely many reduction steps take place at $\alpha$-depth 0.

The following lemma explains why this was a good definition:

Lemma 2. Let $\mathcal{R}$ be a TRS. Then $\mathcal{R}$ is not $\alpha$-terminating if and only if it contains an $\alpha$-bad term.

Proof. From right to left is trivial. The argument in the other direction follows a pattern common in infinite rewriting: suppose that $\mathcal{R}$ is not $\alpha$-terminating, that is, suppose $\mathcal{R}$ contains a term $t_0$ with a reduction $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots$ in which infinitely many reductions take place at $\alpha$-depth $k > 0$. Without loss of generality we may assume that $k$ is the minimal $\alpha$-depth at which infinitely many reductions take place. This implies that there are at most finite many reduction steps at $\alpha$-depth less than $k$. Unlike [13] we don’t consider transfinite reductions.
k. Hence for some $n$ large enough we have that all reduction steps from $t_n$ onwards take place at $\alpha$-depth at least $k$. Therefore at least one of the maximal subterms at $\alpha$-depth $k$ must have an infinite reduction in which infinitely many reductions take place at $\alpha$-depth 0.

3 Dependency pairs for termination

In this section we recall the first component of the dependency pair method of Arts and Giesl [1, 2] to prove termination of a finite TRS $R$ as presented in [7, 12] except for slightly more general notions of $\Delta$-chain and $\Delta$-dependency graph. The intuitive idea behind this method is to compare left-hand sides of rules in a TRS $R$ with those subterms of the right-hand sides that may possibly start a new reduction. The resulting dependency pairs somehow capture the essence of the termination problem of $R$ because there is an infinite chain of them if and only if $R$ is not terminating.

The defined symbols of $R = (F, R)$ are the root symbols of the left-hand sides of the rules in $R$. Let $F^\sharp = \{ f^\sharp \mid f \text{ is a defined symbol of } R \}$, where $f^\sharp$ is a fresh symbol with same arity as $f$. If $t \in T(F, V)$ is of the form $f(t_1, \ldots, t_n)$ with $f$ a defined symbol, then $t^\sharp$ will denote $f^\sharp(t_1, \ldots, t_n)$. If $T \subset T(F, V)$ consists of terms with a defined root symbol, then $T^\sharp$ denotes the set $\{ t^\sharp \mid t \in T \}$.

**Definition 6.** [2, 5, 12] If $l \rightarrow r$ is a rewrite rule of $R$ and $r \equiv C[t]$ such that $C[\ ]$ is a proper (ie. non-empty) context, $t$ has a defined root symbol and $t$ is not a proper subterm of $l$, then the rewrite rule $l^\sharp \rightarrow t^\sharp$ is called a (standard) dependency pair of a TRS $R$. We denote the set of all dependency pairs of $R$ by $\text{DP}(R)$.

We will now define chains and dependency graphs as usual but in a slightly more wider setting than just for sets of standard dependency pairs.

**Definition 7.** An infinite reduction $t_1^\sharp \rightarrow R t_2^\sharp \rightarrow_\Delta t_3^\sharp \rightarrow R t_4^\sharp \rightarrow_\Delta \ldots$ in $R \cup \Delta$ such that all $t_i$ in this sequence have a defined symbol at the root is called an $\Delta$-chain for $R$. In case $\Delta$ is the (finite) set of standard dependency pairs of $R$ we simply call it a chain for $R$.

The termination problem of $R$ can now be transformed into an equivalent statement on chains for $R$:

**Theorem 1.** [1, 2, 12] A finite TRS $R$ is terminating if and only if there are no infinite chains for $R$. □

Theorem 1 will be an instance of a similar, more general theorem that we will prove in the next section on $\alpha$-termination.

4 $\alpha$-Dependency pairs for $\alpha$-termination

In this section we consider a finite TRS $R$ with annotation $\alpha$. We generalise the results for standard dependency pairs from the previous section to $\alpha$-dependency pairs. The intuition behind the definition of $\alpha$-dependency pair that we give below is that an infinite chain of $\alpha$-dependency pairs is the witness of an $\alpha$-bad term.
Suppose we have the following instance $l\sigma \rightarrow r\sigma$ of a rewrite rule of $\mathcal{R}$. There are only two possibilities for $r\sigma$ to contain a defined symbol $f$ at $\alpha$-depth 0: either $f$ occurs already in $r$ or $r$ contains a variable $x$ at $\alpha$-depth 0 and $f$ occurs in a subterm of $r\sigma$ originating from a term substituted for $x$ in $l$. The first possibility will give rise to a standard dependency pair (but note that we will not obtain all of them in this way!) and the second possibility will lead to a new type of dependency pair: a parametrised dependency pair for which we first have to define $\alpha$-decreasing rules:

**Definition 8.** A rule $l \rightarrow r$ is $\alpha$-decreasing in $x$, if $x$ occurs at $\alpha$-depth 0 in $r$ and does not occur at $\alpha$-depth 0 in $l$.

If $l \rightarrow r$ is a rewrite rule of $\mathcal{R}$ and $t$ is a subterm of $r$ with a defined root symbol such that $t$ is not a proper subterm of $l$, then the rewrite rule $l^2 \rightarrow t^2$ is called a (standard) dependency pair of a TRS $\mathcal{R}$.

**Definition 9.** An $\alpha$-dependency pair of an annotated TRS $(\mathcal{R}, \alpha)$ is

- either a rewrite rule of the form $l^2 \rightarrow t^2$ where $l \rightarrow r \equiv C[t]$ is a rewrite rule of $\mathcal{R}$ such that $C[\ ]$ is a context with the hole at $\alpha$-depth 0, $t$ has a defined root symbol and $l$ is not a proper subterm of $t$ at depth 0;
- or it is a parametrised rewrite rule of the form $l^2[x := C(f(x_1, \ldots, x_n))] \rightarrow f^2(x_1, \ldots, x_n)$ where $l \rightarrow r$ is a rewrite rule in $\mathcal{R}$ that is $\alpha$-decreasing in $x$, $C[\ ]$ is a parameter ranging over contexts with hole at $\alpha$-depth 0, $f$ is a defined symbol and all the $x_i$ are chosen fresh.

We denote the set of $\alpha$-dependency pairs of a TRS $\mathcal{R}$ by $\alpha\text{DP}(\mathcal{R})$. We obtain concrete rewrite rules from this set when we instantiate the parametrised rules with contexts of $\mathcal{R}$. We will denote the resulting rewrite relation by $\rightarrow_{\alpha\text{DP}(\mathcal{R})}$.

An infinite reduction $t_1^2 \rightarrow_{\mathcal{R}} t_2^2 \rightarrow_{\alpha\text{DP}(\mathcal{R})} t_3^2 \rightarrow_{\mathcal{R}} t_4^2 \rightarrow_{\alpha\text{DP}(\mathcal{R})} \ldots$ in $\mathcal{R} \cup \alpha\text{DP}(\mathcal{R})$ such that all $t_i$ in this sequence have a defined symbol at the root is called an $\alpha$-chain for $\mathcal{R}$.

We illustrate the definition with a simple example. Consider the TRS with the annotation $\alpha$ such that $\alpha(f) = \{1\}$ and $\alpha(g) = \emptyset$:

\[
\begin{align*}
f(x) & \rightarrow x \\
a & \rightarrow f(g(a))
\end{align*}
\]

First we inspect the standard dependency pairs: there are only two standard dependency pairs $A \rightarrow F(g(a))$ and $A \rightarrow A$ and one can make exactly one infinite chain $A \rightarrow A \rightarrow A \rightarrow \ldots$ with them. This chain corresponds to the infinite-$\alpha$-terminating reduction $\alpha \rightarrow f(g(a)) \rightarrow f(g(f(g(a)))) \rightarrow \ldots$. This shows that standard dependency pairs can not be used to characterise $\alpha$-terminating reductions.

Applying the new definition of $\alpha$-dependency pair we note that $A \rightarrow A$ is no longer an $\alpha$-dependency pair and that the first rule is $\alpha$-decreasing. There are three $\alpha$-dependency pairs:

\[
A \rightarrow F(g(a)), \ F(C[f(x)]) \rightarrow F(x), \ F(C[a]) \rightarrow A
\]

of which the last two are examples of a parametrised $\alpha$-dependency pairs. From these we can form an infinite chain of $\alpha$-dependency pairs: $A \rightarrow F(g(a)) \rightarrow A \rightarrow \ldots$
First we state and prove a generalised version of Theorem 1 to the current setting of α-dependency pairs.

**Theorem 2.** An annotated finite TRS $(R, \alpha)$ is not α-terminating if and only if it has an infinite chain of α-dependency pairs.

**Proof.** Suppose that $(R, \alpha)$ has a term $s_0$ with a non-α-terminating reduction. Then by Lemma 2 we see that $s_0$ is an α-bad term. As $s_0$ is finite, also it contains a minimal subterm $t_0$ at α-depth 0 in $s_0$ that is α-bad. That is, $t_0$ has an infinite reduction containing infinitely many reductions at α-depth 0.

Because there is a first reduction at α-depth 0 in this infinite reduction from $t_0$ we can find a rewrite rule $l \rightarrow r$, a substitution $\sigma$ and a non-variable subterm $s_1$ of $t_0$ such that $t_0$ reduces in finitely many reduction steps to $l\sigma$, all of which steps occur at α-depth greater than 0 and $r\sigma$ is α-bad. As $r\sigma$ is finite it contains a minimal subterm $t_1$ at α-depth 0 such that $t_1$ has an infinite reduction with infinitely many reductions at α-depth 0. Note that the root of $t_1$ must be a defined symbol.

Now there are two possibilities. (1) Either $t_1$ overlaps with $r$. Then we have constructed the α-dependency pair $l^2 \rightarrow t_1^2$, as the root of $t_1$ has α-depth 0 in $r$ and by minimality $t_1$ can not be a proper subterm of $l$ at α-depth 0. (2) Or $t_1$ does not overlap with $r$. Then (again because the root of $t_1$ has α-depth 0 in $r$) the term $t_1$ occurs at α-depth 0 in a term $C[t_1]$ substituted for some variable $x$ in $l$ such that $x$ occurs at α-depth 0 in $r$. Now by minimality of $t_0$ in $s_0$ the term $t_1$ can not occur at α-depth 0 in $l\sigma$. Hence $C[t_1]$ can not so either, implying that $l$ has an occurrence of the variable $x$ at non-zero α-depth. Therefore the rule $l \rightarrow r$ is α-decreasing, which means that we have constructed the parametrised α-dependency pair $l^0[x := C[t_1]] \rightarrow t_1^0$.

The proof of the other direction of the theorem is simple: given an infinite chain, if we remove all the labels $\dagger$ we obtain a non-α-terminating reduction in $R$.  

**4.1 Strong dependency pairs for strong convergence**

If Definition 9 we take for $\alpha$ just the empty annotation $\alpha$, then we note that there are no ε-decreasing rules, because all variables in the left-hand sides of a rule occur at ε-depth 0. Hence $\epsilon$-dependency pairs are exactly the standard dependency pairs of Definition 6 and Theorem 2 reduce for $\alpha = \epsilon$ exactly to Theorem 1.

The instantiation of Definition 9 with the maximal annotation $\mu$ is also of separate interest because the only context with a hole at $\mu$-depth 0 is the empty context. In [9] we introduced a method based on strong dependency pairs for proving strong convergence of TRSs. Strong convergence was coined in [13]. It is a property for TRSs that is equivalent to top-termination. We will now give the instantiation of Definition 9 with $\alpha = \mu$ for two reasons. One is that in case of $\mu$ the context parameter disappears entirely from the parametrised dependency pairs and Dershowitz’ refinement [5, 12] does not apply here. Secondly to put the record straight and to amend the short abstract [9] where the strong dependency pairs related to collapse rules were omitted.

When $\alpha = \mu$ we have that Definition 9 simplifies to:
Definition 10. A strong dependency pair or \(\mu\)-dependency pair of a TRS \(R\) is either a rewrite rule of the form \(l \rightarrow_r r\) where \(l \rightarrow_r r\) is not a collapse rule or it is a rewrite rule of the form \(l^*[x := f(x_1, \ldots, x_n)] \rightarrow f^*(x_1, \ldots, x_n)\) where \(l \rightarrow x\) is a collapse rule and \(f\) is a defined symbol and all the \(x_i\) are chosen fresh. We denote the set of strong dependency pairs of \(R\) by \(SDP(R)\).

And because top-termination is equivalent to strong convergence we obtain from Theorem 2:

**Corollary 1 ([9])**. A finite TRS \(R\) is strongly converging if and only if it has no infinite chains of strong dependency pairs.

Collapse rules occur quite naturally. Examples are rules like \(x + 0 \rightarrow x\) or rules that deconstruct a term like the rules (4,5) of the extended Fibonacci TRS in section 6.2.

5 Testing dependency graphs on cycles

Analysing whether or not there infinite chains of \(\alpha\)-dependency pairs can now be done with the exactly the same tools as developed for the analysis this problem for standard dependency pairs of \(R\).

We will now give a definition of dependency graph for \(\alpha\)-dependency pairs. Note that the set \(\alpha\)DP\((R)\) of \(\alpha\)-dependency pairs of a TRS \(R\) is finite as it contains the parametrised dependency pairs and not the great many instantiations thereof. By this convention all instances of one parametrised rule can be represented by one node in the \(\alpha\)-dependency graph of \(R\).

Let \(\Delta\) be some finite set of pairs \(s^* \rightarrow t^\star\) such that \(s, t \in T(F, V)\) have a defined root symbol.

**Definition 11.** The dependency graph \(DG(\Delta)\) is a graph with \(\Delta\) as set of nodes. There is an arrow from \(s \rightarrow t\) to \(u \rightarrow v\) if and only if there are substitutions \(\sigma\) and \(\tau\) such that \(t\sigma \rightarrow \rightarrow_{R} u\tau\). We use the notation \(\alpha\)DG\((R)\) in case \(\Delta\) is the set \(\alpha\)DP\((R)\) of \(\alpha\)-dependency pairs.

We first restate the subterm criterion of Hirokawa and Middeldorp [12] in the general setting of \(\alpha\)-dependency graphs. We refer to [12] and [11] for unexplained definitions, notations. The proofs go through.

**Definition 12.** Let \(C \subseteq \alpha\)DP\((R)\) be such that every dependency pair symbol in \(C\) has positive arity. A simple projection for \(C\) is a map \(\pi\) assigning to every \(n\)-ary dependency pair symbol \(f^\star\) in \(C\) an argument position \(i \in \{1, \ldots, n\}\). The map that assigns to every term \(f^\star(t_1, \ldots, t_n)\) with \(f^\star\) in \(f^\star\) in \(C\) its argument at position \(\pi(f^\star)\) we also denote by \(\pi\).

**Theorem 3 (Subterm Criterion [12]).** Let \(\alpha\)DP\((R)\) be the finite set of \(\alpha\)-dependency pairs for a TRS \(R\). There are no infinite \(\alpha\)-chains for \(R\) if and only if for every cycle \(C\) in \(\alpha\)DG\((R)\) there exist a simple projection \(\pi\) for \(C\) such that \(\pi(C) \subseteq \leq\) and \(\pi(C) \cap \nleq \neq \emptyset\).

**Theorem 4 ([11]).** Let \(\alpha\)DP\((R)\) be the finite set of \(\alpha\)-dependency pairs for a TRS \(R\). There are no infinite \(\alpha\)-chains for \(R\) if and only if for every maximal cycle (or strongly connected component) \(S\) in \(\alpha\)DG\((R)\) there exist an argument filtering \(\pi\) and a reduction pair \((\nleq, >)\) such that \(\pi(R) \subseteq \geq\) and \(\pi(S) \subseteq >\).
6 Three examples

In this section we present three examples of rewrite systems that are not terminating but are $\alpha$-terminating for suitable chosen annotations $\alpha$.

6.1 The simple Fibonacci TRS

Our first example is the simple Fibonacci TRS $F_1$:

\[
\begin{align*}
0 + m & \rightarrow m & (1) \\
\text{s(n)} + m & \rightarrow \text{s(n + m)} & (2) \\
fib(n, m) & \rightarrow n : fib(m, n + m) & (3)
\end{align*}
\]

The TRS $F_1$ is clearly not terminating; the term $\text{fib}(1, 1)$ computes the infinite list of Fibonacci numbers:

$\text{fib}(1, 1) \rightarrow \text{fib}(1, 2) \rightarrow \text{fib}(2, 3) \rightarrow \ldots$

where the numbers used are abbreviations for numbers in successor notation. The function symbols $+$ and $\text{fib}$ are the two defined symbols of the Fibonacci TRS.

Lemma 3. The simple Fibonacci TRS is top-terminating.

Proof. The proof consist of a simple analysis of the graph of the $\mu$-dependency pairs of $F_1$ followed by an application of the subterm criterion.

The rule (1) is the only collapse rule and there are two $\mu$-dependency pairs:

\[
\begin{align*}
0 +^2 (n + m) & \rightarrow n +^2 m & (4) \\
0 +^2 \text{fib}(n, m) & \rightarrow \text{fib}(n, m) & (5)
\end{align*}
\]

The $\mu$-dependency graph has one cycle $C = \{4\}$ involving only the dependency pair symbol $+^2$. Taking the simple projection $\pi(+^2) = 2$ and applying it to $C$ we obtain

\[
n + m \rightarrow m
\]

which is strictly decreasing with respect to the subterm relation. Using the subterm criterion of Theorem 3 it follows directly that there are no infinite chains of $\mu$-dependency pairs. Hence the TRS $F_1$ is strongly converging by Corollary 1.

Lemma 4. Let the annotation $\alpha'$ be defined by $\alpha'(s) = \{2\}$ and $\alpha'(+) = \emptyset$. The simple Fibonacci TRS $F_1$ is $\alpha'$-terminating.

Proof. By a similar analysis of the graph of $\alpha'$-dependency pairs. There are no $\alpha'$-decreasing rules, and there is only one $\alpha'$-dependency pair:

\[
s(n) +^2 m \rightarrow n +^2 m (7)
\]

As in the previous lemma, the dependency graph of $\alpha'$-dependency pairs of $F_1$ has only one cycle $C = \{7\}$ involving the binary dependency pair symbol $+^2$. If we now take the simple projection $\pi(+^2) = 1$ and apply it to $C$ we obtain

\[
s(n) \rightarrow n
\]

which is strictly decreasing with respect to the subterm relation. Using the subterm criterion of Theorem 2 it follows that there are no infinite $\alpha'$-chains. Hence the TRS $F_1$ is $\alpha'$-terminating by Theorem 2.
6.2 The extended Fibonacci TRS

The Fibonacci TRS $\mathcal{R}_1$ extended with a mechanism to extract the $n$-th term of a list.

\begin{align*}
0 + m &\rightarrow m \quad (1) \\
(s(n) + m) &\rightarrow s(n + m) \quad (2) \\
\text{fib}(n, m) &\rightarrow n : \text{fib}(m, n + m) \quad (3) \\
\text{nth}(0, x : xs) &\rightarrow x \quad (4) \\
\text{nth}(s(n), x : xs) &\rightarrow \text{nth}(n, xs) \quad (5)
\end{align*}

As it extends $\mathcal{F}_1$ the extended Fibonacci TRS $\mathcal{F}_2$ is also not terminating. We have now three defined symbols $+,$ $\text{fib}$ and $\text{nth}.$

**Lemma 5.** The extended Fibonacci TRS $\mathcal{F}_2$ is $\mu$-terminating.

**Proof.** By an analysis of its $\mu$-dependency pairs. There are two collapse rules (1) and (4), and there are the following $\mu$-dependency pairs:

\begin{align*}
\text{nth}^\sharp(s(n), x : xs) &\rightarrow \text{nth}^\sharp(n, xs) \quad (6) \\
0 +\sharp (n + m) &\rightarrow n +\sharp m \quad (7) \\
0 +\sharp \text{fib}(n, m) &\rightarrow \text{fib}^\sharp(n, m) \quad (8) \\
0 +\sharp \text{nth}(n, x) &\rightarrow \text{nth}^\sharp(n, x) \quad (9) \\
\text{nth}^\sharp(s(n), x : (n + m)) &\rightarrow n +\sharp m \quad (10) \\
\text{nth}^\sharp(s(n), x : \text{fib}(n, m)) &\rightarrow \text{fib}^\sharp(n, m) \quad (11) \\
\text{nth}^\sharp(s(n), x : \text{nth}(n, x)) &\rightarrow \text{nth}^\sharp(n, x) \quad (12)
\end{align*}

There is one cycle $C = \{7\}$ and one SCC $S = \{6, 12\}. If we now take the simple projection $\pi(\text{nth}^\sharp) = \pi(+\sharp) = 2$ and apply it to rules 6, 7 and 12 we obtain respectively

\begin{align*}
x : xs &\rightarrow xs \quad (13) \\
n + m &\rightarrow m \quad (14) \\
x : \text{nth}(n, x) &\rightarrow x \quad (15)
\end{align*}

which are all strictly decreasing with respect to the subterm relation. By repeated use of the subterm criterion of Theorem 3 it follows that there are no infinite $\mu$-chains. Hence $\mathcal{F}_2$ is $\mu$-terminating by Theorem 2. \hfill \Box

**Lemma 6.** The extended Fibonacci TRS $\mathcal{F}_2$ is $\alpha'$-terminating for the annotation $\alpha'$ of Lemma 4.

**Proof.** By analysis of the graph of $\alpha'$-dependency pairs of $\mathcal{F}_2.$ There is one $\alpha'$-decreasing rule (5) and there are the following $\alpha'$-dependency pairs:

\begin{align*}
s(n) +\sharp m &\rightarrow n +\sharp m \quad (6) \\
\text{nth}^\sharp(s(n), x : xs) &\rightarrow \text{nth}^\sharp(n, xs) \quad (7) \\
\text{nth}^\sharp(s(n), x : C[n + m]) &\rightarrow n +\sharp m \quad (8) \\
\text{nth}^\sharp(s(n), x : C[\text{fib}(n, m)]) &\rightarrow \text{fib}^\sharp(n, m) \quad (9) \\
\text{nth}^\sharp(s(n), x : C[\text{nth}(n, x)]) &\rightarrow \text{nth}^\sharp(n, x) \quad (10)
\end{align*}
The α'-dependency graph contains one cycle $C = \{6\}$ and one SCC $S = \{7, 10\}$. If we now take the simple projection $\pi(\mathcal{+}^\sharp) = \pi(nth^\sharp) = 1$ and apply it to rules 6, 7 and 10 we obtain in each case the same rule:

$$s(n) \to n \quad (11)$$

which is strictly decreasing with respect to the subterm relation. By repeated use of the subterm criterion of Theorem 3 it follows that there are no infinite α'-chains. Hence $\mathcal{F}_2$ is α'-terminating by Theorem 2.

6.3 The Sieve of Eratosthenes

The Sieve of Eratosthenes is the well-known recursive method to sieve the prime numbers from the natural numbers. It is a well-used example in the literature on functional programming [19]. The Sieve can be represented by the next simple TRS $\mathcal{S}$ using the successor format of natural numbers [9] without any need for explicit arithmetic operations:

$$\text{sieve}(\star : y) \to \text{sieve}(y) \quad (1)$$

$$\text{sieve}(s(n) : y) \to s(n) : \text{sieve}(\text{mark}(y, n, n)) \quad (2)$$

$$\text{mark}(x : y, 0, s(m)) \to \star : \text{mark}(y, m, s(m)) \quad (3)$$

$$\text{mark}(x : y, s(n), s(m)) \to x : \text{mark}(y, n, s(m)) \quad (4)$$

$$\text{from}(n) \to n : \text{from}(s(n)) \quad (5)$$

As in the previous example of the Fibonacci TRS we will use digits as convenient shorthand for the corresponding numbers in successor notation.

The subTRS $\mathcal{S}_1$ given by the rules (1,2,3,4) contains the sieve component of the algorithm. It is not hard to see that $\mathcal{S}_1$ itself is terminating. Because $\mathcal{S}_1$ is orthogonal, it is confluent and therefore complete: each term has a unique normal form which will be found by any rewriting strategy. Hence the term $\text{sieve}(2 : 3 : 4 : \ldots : n)$ terminates for any number $n$. And by a well-known argument it follows that the computed sublist is the list of primes $\leq n$.

The Sieve of Eratosthenes can also be applied to the infinite list $2 : 3 : 4 : \ldots$ to enumerate the infinite list of all prime numbers. Rule (5) introduces the possibility of computing an infinite lists of integers. Now the finite term $\text{sieve}(\text{from}(s(s(0))))$ can now be evaluated with some “lazy” care to the infinite list of all primes.

Lemma 7. The Sieve of Eratosthenes TRS $\mathcal{S}$ is is µ-terminating.

Proof. The TRS $\mathcal{S}$ has only one µ-dependency pair:

$$\text{sieve}^\sharp(\star : y) \to \text{sieve}^\sharp(y) \quad (6)$$

There is one cycle $\mathcal{C} = \{6\}$ involving the unary dependency pair symbol $\text{sieve}^\sharp$. If we now take the simple projection $\pi(\text{sieve}^\sharp) = 1$ and apply it to $\mathcal{C}$ we obtain

$$\star : y \to y \quad (7)$$

which is strictly decreasing with respect to the subterm relation. Hence the TRS $\mathcal{S}$ is µ-terminating.
Lemma 8. The Sieve of Eratosthenes TRS is \(\alpha\)-terminating for the annotation \(\alpha\) that only annotates the second argument of the list constructor "::"

Proof. The TRS \(S\) has one \(\alpha\)-decreasing rule (1) and it has the following \(\alpha\)-dependency pairs:

\[
\begin{align*}
\text{sieve}^\sharp(\star : y) & \rightarrow \text{sieve}^\sharp(y) \quad (8) \\
\text{sieve}^\sharp(\star : C[\text{sieve}(x)]) & \rightarrow \text{sieve}^\sharp(x) \quad (9) \\
\text{sieve}^\sharp(\star : C[\text{mark}(x,y,z)]) & \rightarrow \text{mark}^\sharp(x,y,z) \quad (10) \\
\text{sieve}^\sharp(\star : C[\text{from}(x)]) & \rightarrow \text{from}^\sharp(x) \quad (11)
\end{align*}
\]

There is only one SCC \(S = \{8,9\}\) involving the unary dependency pair symbol \(\text{sieve}^\sharp\). If we now take the simple projection \(\pi(\text{sieve}^\sharp) = 1\) and apply it to \(S\) we obtain

\[
\begin{align*}
\star : y & \rightarrow y \quad (12) \\
\star : C[\text{sieve}(x)] & \rightarrow x \quad (13)
\end{align*}
\]

which are both strictly decreasing with respect to the subterm relation. Hence the TRS \(S\) is \(\alpha\)-terminating.

Note that the Sieve can be extended like the Fibonacci TRS with rules to extract the \(n\)-th term of a list. The extended Sieve is also strongly converging and \(\alpha\)-terminating.

Also note that thanks to Hirokawa and Middeldorp’s subterm criterion [12] we could sidestep the presence of the context parameters so smoothly in this last proof.

6.4 Real numbers

The following TRS \(R\) performs addition on positive (binary) real numbers. The intuition behind the notation is that \(x : y = 10x + y\) and \(x ; y = x + \frac{1}{10} y\). So the positive real number 110.011 will be represented by \(((1 : 1) : 0) ; (0 ; (1 ; 1))\).

\[
\begin{align*}
0 + 0 & \rightarrow 0 \quad (1) & x ; 0 & \rightarrow x \quad (9) \\
0 + 1 & \rightarrow 1 \quad (2) & (x ; y) + z & \rightarrow (x + z) ; y \quad (10) \\
1 + 0 & \rightarrow 1 \quad (3) & x + (y ; z) & \rightarrow (x + y) ; z \quad (11) \\
1 + 1 & \rightarrow 1 : 0 \quad (4) & (x ; y) ; z & \rightarrow x ; (y + z) \quad (12) \\
(x : y) + z & \rightarrow y : (x + z) \quad (5) & (x ; y) ; z & \rightarrow x ; (y + z) \quad (13) \\
(x : y) + z & \rightarrow x : (y + z) \quad (6) & x ; (y : z) & \rightarrow (x + y) ; z \quad (14) \\
x : (y : z) & \rightarrow (x + y) ; z \quad (7) & x ; (y ; z) & \rightarrow (x : y) ; z \quad (15) \\
0 : x & \rightarrow x \quad (8) & w ; ((x ; y) ; z) & \rightarrow (w + x) ; (y ; z) \quad (16)
\end{align*}
\]

The rules (1-8) originate from the Walters’ TRS for integer arithmetic [22]. The rules (9-16) extend it to positive real numbers. On finite terms the TRS is terminating. This was originally proved with help of semantic labelling [21]. The tools TTT and AProVE can do it automatically. And since it is \((\epsilon)\)-terminating it is \(\alpha\)-terminating for any annotation \(\alpha\).

Suppose we add infinite binary expansions to the TRS by adding simple the rules:

\[
\begin{align*}
\text{string} & \rightarrow 0 ; \text{string} \quad (17) \\
\text{string} & \rightarrow 1 ; \text{string} \quad (18)
\end{align*}
\]

It is clear that we have lost termination. But we don’t lose \(\alpha\)-termination for the annotation \(\alpha\) that only annotates the second argument of the constructor "::".

Lemma 9. The real number TRS $\mathcal{R}$ extended with the rules (17, 18) is $\alpha'$-terminating.

Proof. First we look at the $\alpha'$-dependency graph of the non-extended TRS. As this TRS is $\alpha'$-terminating it does not contain any cycles. Adding the new rules (17, 18) extends the $\alpha'$-dependency graph but as they can only occur at the beginning of chains this does not introduce any new cycles. Hence the extended TRS is also $\alpha'$-terminating. 

All this remains true if we extend $\mathcal{R}$ with multiplication and unary minus and or change to larger bases as in [22, 21].

7 $\alpha$-Termination versus context sensitive rewriting

Consider an annotated TRS $(\mathcal{R}, \alpha)$. In the Lucas’ context-sensitive rewriting the annotation $\alpha$ is used to block rewriting in subterms at annotated argument positions, that is in our terminology of $\alpha$-depth only reduction steps at $\alpha$-depth 0 are allowed. This restriction makes it a suitable formalism for defining reduction strategies.

We observe that if $\mathcal{R}$ is $\alpha$-terminating, then context sensitive rewriting in $(\mathcal{R}, \alpha)$ is terminating. For if the CRS, $(\mathcal{R}, \alpha)$ is not terminating then there is an infinite reduction consisting of reduction steps at $\alpha$-depth 0, contradicting $\alpha$-termination of $\mathcal{R}$. The converse is not true. If context-sensitive rewriting of a term has terminated, then a reduction at greater $\alpha$-depth may still trigger further reduction steps at $\alpha$-depth 0. In general, termination of the CRS $(\mathcal{R}, \alpha)$ does not imply that the TRS $\mathcal{R}$ is $\alpha$-terminating. 

It follows that the dependency method for proving $\alpha$-termination can be used to prove termination of context sensitive rewriting systems with replacement map $\mu$. In Lucas [16] a partial converse is claimed: if a left-linear CRS $\mathcal{R}$ with the canonical replacement map $\mu_{\mathcal{R}}^{can}$ is terminating, then $\mathcal{R}$ is strongly convergent, i.e. $\mu$-terminating for the maximal annotation $\mu$. Since $\mu \supseteq \mu_{\mathcal{R}}^{can}$ it follows simply from Lemma 1 that $\mu$-termination implies $\mu_{\mathcal{R}}^{can}$ termination which in turn implies that the CRS $(\mathcal{R}, \mu_{\mathcal{R}}^{can})$ is terminating. Left-linearity is not used in this argument.

8 Conclusion

In this paper we have presented a method based on $\alpha$-dependency pairs for deciding whether or not a TRS is $\alpha$-terminating for a given annotation $\alpha$. To what extent this method can be automated we haven’t investigated yet. The examples show that this might well be possible despite the fact that the $\alpha$-dependency pairs may contain formal context parameters.

Given a TRS $\mathcal{R}$ this method can in principle be used to search for annotations for which $\mathcal{R}$ is $\alpha$-terminating. If we know that a TRS is $\mathcal{R}$ $\alpha$-terminating we can then search for normalising reduction strategies.

For instance the following strategies are normalising when executed in a fair manner:

- $\alpha$-lazy strategy: given a term $t$ search for and then contract a redex of minimal $\alpha$-depth.
hybrid \( \alpha \)-eager/lazy strategy: given a term \( t \), consider all redexes of minimal \( \alpha \)-depth and then contract an innermost redex from this set.

The intuition here is that the less an annotation \( \alpha \) annotates the more eager the hybrid \( \alpha \)-eager/lazy strategy will be. The annotation \( \alpha' \) that annotated only the second argument position of the list constructor in case of both the Fibonacci TRS and the Eratosthenes TRS is a nice example.

If the TRS is \( \alpha \)-terminating, these strategies find possible infinite normal forms. In case the \( R \) is confluent, these strategies will find the unique normal form. Whether this will result in a reduction of length at most \( \omega \) depends on whether or not the TRS is left-linear. It is not difficult to come up with examples of non-left-linear TRSs with terms that can not normalise in at most \( \omega \)-many steps, but can in a transfinite \( \alpha \)-terminating reduction. If the TRS is left-linear we can prove along the lines of [13] that any transfinite \( \alpha \)-terminating reduction can be compressed into one of length at most \( \omega \).

It will be interesting and worthwhile to see whether this method can lead to an effective tool for strictness analysis.

9 Acknowledgements

I acknowledge the help, discussions and criticisms of Thomas Arts, Jürgen Giesl, Aart Middeldorp, Vincent van Oostrom, Hitoshi Ohsaki, Paula Severi and Hans Zantema on versions this paper. I also wish to acknowledge the friendly hospitality of ETL (now AIST) where an early draft of this paper was conceived.

References


