

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSIDAD POLITÉCNICA DE VALENCIA

P.O. Box: 22012

E-46021 Valencia (SPAIN)



Informe Técnico / Technical Report

Ref. No.: DSIC-II/08/08

Pages:22

Title: Dependency Pairs for the Termination of Narrowing

Author(s): María Alpuente and Santiago Escobar and José Iborra

Date: July 2008

Keywords: Narrowing, Rewriting, Termination, Dependency Pairs

Vº Bº

Leader of research Group

Author(s)

Termination of Narrowing using Dependency Pairs^{*}

María Alpuente, Santiago Escobar, and José Iborra

Technical University of Valencia (UPV), Spain. {alpuente,sescobar,jiborra}@dsic.upv.es

Abstract. In this work, we extend the dependency pair approach for automated proofs of termination in order to automate the termination proofs of narrowing. Narrowing is a generalization of term rewriting that allows free variables in terms (as in logic programming) and replaces pattern matching by syntactic unification. Narrowing has a number of important applications including execution of functional–logic programming languages, verification of cryptographic protocols, and equational unification. Termination of narrowing is, therefore, of great interest to these applications. Our extension of the dependency pair approach generalizes the standard notion of dependency pairs by taking specifically into account the dependencies between the left-hand side of a rewrite rule and its own argument subterms. We demonstrate that the new *narrowing dependency pairs* exactly capture the narrowing termination behavior and provide an effective termination criterion which we prove to be sound and complete. Finally, we discuss how the problem of analyzing narrowing chains can be recast as a standard analysis problem for traditional (rewriting) chains, so that the proposed technique can be effectively mechanized by reusing the standard DP infrastructure.

1 Introduction

In recent years, the dependency pair (DP) method for automating the termination proofs of term rewriting has achieved tremendous success, as witnessed by the large number of publications and tools since its introduction in [5] and subsequent reformulation in [13] (see [15, 17] for extensive references thereof). In [24], the notions of dependency pairs and dependency graphs, which were originally developed for term rewriting, were adapted to the logic programming domain, leading to automated termination analyses that are directly applicable to any definite logic program.

Narrowing [12] is a generalization of term rewriting that allows free variables in terms (as in logic programming) and replaces pattern matching by syntactic unification so that it subsumes both rewriting and SLD-resolution [16]. Narrowing has many important applications including execution of functional–logic programming languages [7, 16, 22], verification of security policies [20] and cryptographic protocols [10], equational unification [20, 19], symbolic reachability [23], and model checking [11], among others. Termination of narrowing itself is, therefore, of great interest to these applications.

Narrowing termination is a much more difficult property to achieve than termination of term rewriting or termination of pure logic programs due to the high degree of nondeterminism caused by the interaction of rule selection, redex selection, and unification. In recent works [2, 3], we identified some non-trivial classes of TRSs where narrowing terminates. The results in [3] generalize previously known criteria for narrowing termination, which were essentially restricted to either confluent term rewriting systems (TRSs) [19] or to left-flat TRSs (i.e., each argument of the left-hand side of a rewrite rule is either a variable or a ground term) that are compatible with a termination ordering [6], among other applicability conditions. Roughly speaking, we proved in [3] that confluence is a superfluous requirement for the termination of narrowing. We also weakened the left-flatness condition required in [6] to the requirement that every non-ground, strict subterm of the left-hand side (lhs) of every rewrite rule must be a *rigid normal form*, i.e., unnarrowable. Finally, in [2] we proved modular

^{*} This work has been partially supported by the EU (FEDER) and Spanish MEC project TIN2007-68093-C02-02, Integrated Action Hispano-Alemana HA2006-0007, and UPV-VIDI grant 3249 PAID0607.

```

filter(pkt(src, dst, established))    → accept
filter(pkt(eth0, dst, new))          → accept
filter(pkt(194.179.1.x:port, dst, new)) → filter(pkt(secure, dst, new))
filter(pkt(158.42.x.y:port, dst, new)) → filter(pkt(secure, dst, new))
filter(pkt(secure, dst:80, new))      → accept
filter(pkt(secure, dst:other, new))   → drop
filter(pkt(ppp0, dst, new))           → drop
filter(pkt(123.123.1.1:port, dst, new)) → accept
pkt(10.1.1.1:port, ppp0, s)           → pkt(123.23.1.1:port, ppp0, s)
pkt(10.1.1.2:port, ppp0, s)           → pkt(123.23.1.1:port, ppp0, s)
pkt(src, 123.123.1.1:port, new) → natroute(pkt(src, 10.1.1.1:port, established),
                                           pkt(src, 10.1.1.2:port, established))

natroute(a, b)                       → a
natroute(a, b)                       → b

```

Fig. 1. The *FullPolicy* TRS

termination of a restriction of narrowing defined in [19] in several hierarchical combinations of TRSs, which provides new algorithmic criteria to prove termination of narrowing.

Two different adaptations of the DP technique for narrowing have been proposed recently. In [25, 26], the original dependency pair technique of [5] was adapted to the termination of narrowing, whereas [27] adapts the logic programming dependency pair approach of [30] instead, to prove narrowing termination w.r.t. a given set of queries that are supplemented with *call modes*. Unfortunately, these two methods apply only to two particular classes of TRSs: right-linear TRSs (i.e., no repeated variables occur in the right-hand sides of the rules) or constructor systems (the arguments of the lhs's of the rules are constructor –i.e., data– terms). These two classes are over restrictive for many practical uses of narrowing, such as the applications mentioned above. In this work, we are interested in developing automatable methods for proving termination of narrowing in TRSs that resist all previous techniques.

Example 1. Consider our running example, which is the non-right-linear, non-constructor-based, non-confluent TRS, adapted from [29], that is shown in Figure¹ 1. This TRSs models a security (filtering) plus routing policy that allows packets coming from external networks to be analyzed. We do not describe the intended meaning of each symbol since it is not relevant for this work, but note the kind of expressivity that is assumed in the domain of rule-based policy specification, that does not fit in the right-linear restriction or the constructor discipline. Narrowing is terminating for this TRS, but it cannot be proved by using any of the existing methods [2, 3, 25, 26, 27]. In this paper, we provide techniques that allow us to prove it automatically.

The main contributions of this paper are as follows:

- We present a new method for proving the termination of narrowing that is based on a suitable extension of the DP technique to the case of narrowing that is applicable to any class of TRSs. Our method generalizes the standard notion of dependency pairs to narrowing by taking the dependencies between the lhs of a rewrite rule and its own argument subterms specifically into account.
- We demonstrate that the new *narrowing dependency pairs* exactly capture the narrowing termination behavior. We provide a termination criterion based on *narrowing chains* which we demonstrate to be sound and complete.
- This allows us to develop a technique that is more general in all cases and, for general calls (i.e., without considering call modes) strictly subsumes the DP methods for proving termination of narrowing of [25, 26, 27], as well as all previous (decidable) narrowing termination criteria [2, 3, 6, 19].

¹ In this paper, variables are written in italic font and function symbols are in typewriter font.

- We have implemented a tool for proving the termination of narrowing automatically that is based on our technique, and we made it publicly available.

Plan of the paper

After recalling some preliminaries in Section 2, in Section 3 we discuss the problem of *echoing*, which we identify as being ultimately responsible for the non-termination of narrowing. In Section 4, we develop the notion of narrowing dependency pairs and provide a sound and complete criterion for the termination of narrowing that is based on analyzing narrowing chains. In Section 5, we discuss the effective automation of our method, which mainly consists of two steps: DP’s extraction and argument filtering transformation. Section 6 concludes.

2 Preliminaries

In this section, we briefly recall the essential notions and terminology of term rewriting [9]. For missing notions and definitions on equations, orderings and rewriting, we also refer to [28, 31].

\mathcal{V} denotes a countably infinite set of variables, and Σ denotes a set of function symbols, or signature, each of which has a fixed associated arity. Terms are viewed as labelled trees in the usual way, where $\mathcal{T}(\Sigma, \mathcal{V})$ and $\mathcal{T}(\Sigma)$ denote the non-ground term algebra and the ground algebra built on $\Sigma \cup \mathcal{V}$ and Σ , respectively. Positions are defined as sequences of natural numbers used to address subterms of a term, with ϵ as the root (or top) position (i.e., the empty sequence). Concatenation of positions p and q is denoted by $p.q$, and $p < q$ is the usual prefix ordering. The root symbol of a term is denoted by $\text{root}(t)$. Given $S \subseteq \Sigma \cup \mathcal{V}$, $\text{Pos}_S(t)$ denotes the set of positions of a term t that are rooted by function symbols or variables in S . $\text{Pos}_{\{f\}}(t)$ with $f \in \Sigma \cup \mathcal{V}$ will be simply denoted by $\text{Pos}_f(t)$, and $\text{Pos}_{\Sigma \cup \mathcal{V}}(t)$ will be simply denoted by $\text{Pos}(t)$. $t|_p$ is the subterm at the position p of t . $t[s]_p$ is the term t with the subterm at the position p replaced with term s . By $\text{Var}(s)$, we denote the set of variables occurring in the syntactic object s . By \bar{x} , we denote a tuple of pairwise distinct variables. A *fresh* variable is a variable that appears nowhere else. A *linear* term is one where every variable occurs only once.

A *substitution* σ is a mapping from the set of variables \mathcal{V} into the set of terms $\mathcal{T}(\Sigma, \mathcal{V})$, with a finite domain $D(\sigma)$ and image $I(\sigma)$. A substitution is represented as $\{x_1/t_1, \dots, x_n/t_n\}$ for variables x_1, \dots, x_n and terms t_1, \dots, t_n . The application of substitution θ to term t is denoted by $t\theta$, using postfix notation. Composition of substitutions is denoted by juxtaposition, i.e., the substitution $\sigma\theta$ denotes $(\theta \circ \sigma)$. We write $\theta|_{\text{Var}(s)}$ to denote the restriction of the substitution θ to the set of variables in s ; by abuse of notation, we often simply write $\theta|_s$. Given a term t , $\theta = \nu[t]$ iff $\theta|_{\text{Var}(t)} = \nu|_{\text{Var}(t)}$, that is, $\forall x \in \text{Var}(t)$, $x\theta = x\nu$. A substitution θ is more general than σ , denoted by $\theta \leq \sigma$, if there is a substitution γ such that $\theta\gamma = \sigma$. A *unifier* of terms s and t is a substitution ϑ such that $s\vartheta = t\vartheta$. The *most general unifier* of terms s and t , denoted by $\text{mgu}(s, t)$, is a unifier θ such that for any other unifier θ' , $\theta \leq \theta'$.

A *term rewriting system* (TRS) \mathcal{R} is a pair (Σ, R) , where R is a finite set of rewrite rules of the form $l \rightarrow r$ such that $l, r \in \mathcal{T}(\Sigma, \mathcal{V})$, $l \notin \mathcal{V}$, and $\text{Var}(r) \subseteq \text{Var}(l)$. For TRS \mathcal{R} , $l \rightarrow r \ll \mathcal{R}$ denotes that $l \rightarrow r$ is a new variant of a rule in \mathcal{R} such that $l \rightarrow r$ contains only *fresh* variables, i.e., contains no variable previously met during any computation (standardized apart). We will often write just \mathcal{R} or (Σ, R) instead of $\mathcal{R} = (\Sigma, R)$. A TRS \mathcal{R} is called *left-linear* (respectively *right-linear*) if, for every $l \rightarrow r \in \mathcal{R}$, l (respectively r) is a linear term. Given a TRS $\mathcal{R} = (\Sigma, R)$, the signature Σ is often partitioned into two disjoint sets $\Sigma = \mathcal{C} \uplus \mathcal{D}$, where $\mathcal{D} = \{f \mid f(t_1, \dots, t_n) \rightarrow r \in R\}$ and $\mathcal{C} = \Sigma \setminus \mathcal{D}$. Symbols in \mathcal{C} are called *constructors*, and symbols in \mathcal{D} are called *defined functions*. The elements of $\mathcal{T}(\mathcal{C}, \mathcal{V})$ are called *constructor terms*. We let $\text{Def}(\mathcal{R})$ denote the set of defined symbols in \mathcal{R} . A constructor system is a TRS whose lhs’s are terms of the form $f(d_1, \dots, d_k)$ where $f \in \mathcal{D}$ and d_1, \dots, d_k are constructor terms.

A rewrite step is the application of a rewrite rule to an expression. A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ *rewrites* to a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, denoted by $s \xrightarrow{p}_{\mathcal{R}} t$, if there exist $p \in \text{Pos}_{\Sigma}(s)$, $l \rightarrow r \in \mathcal{R}$, and substitution

σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. When no confusion can arise, we omit the subscript in $\rightarrow_{\mathcal{R}}$. We also omit the reduced position p when it is not relevant. A term s is a *normal form* w.r.t. the relation $\rightarrow_{\mathcal{R}}$ (or simply a normal form), if there is no term t such that $s \rightarrow_{\mathcal{R}} t$. A term is a *reducible expression* or *redex* if it is an instance of the left hand side of a rule in \mathcal{R} . A term s is a *head normal form* if there are no terms t, t' s.t. $s \rightarrow_{\mathcal{R}}^* t' \xrightarrow{\epsilon}_{\mathcal{R}} t$. A term t is said *terminating* w.r.t. R if there is no infinite reduction sequence $t \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \dots$. A TRS \mathcal{R} is *(\rightarrow)-terminating* (also called strongly normalizing or noetherian) if every term is terminating w.r.t. R . A TRS \mathcal{R} is *confluent* if, whenever $t \rightarrow_{\mathcal{R}}^* s_1$ and $t \rightarrow_{\mathcal{R}}^* s_2$, there exists a term w s.t. $s_1 \rightarrow_{\mathcal{R}}^* w$ and $s_2 \rightarrow_{\mathcal{R}}^* w$.

Narrowing is a generalization of term rewriting that allows free variables in terms –as in logic programming– and replaces pattern matching with syntactic unification in order to (non-deterministically) reduce these terms. A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ narrows to a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, denoted by $s \xrightarrow{p}_{\theta, \mathcal{R}} t$, if there exist $p \in \text{Pos}_{\Sigma}(s)$, $l \rightarrow r \ll \mathcal{R}$, and substitution θ such that $\theta = \text{mgu}(s|_p, l)$ and $t = (s[r]_p)\theta$. We use $\xrightarrow{\epsilon}_{\mathcal{R}}$ (resp. $\xrightarrow{\epsilon}_{\theta, \mathcal{R}}$) to denote steps in which the selected redex (resp. *narrex*, i.e. narrowable expression) occurs below the root.

3 The *echoing* problem

The dependency pair technique [5] is one of the most powerful methods for automated termination analyses. The technique focuses on the dependency relations between defined function symbols, paying particular attention to strongly connected components within a graph of functional dependencies, in order to produce automated termination proofs. The dependency graph is typically extracted by considering the dependencies between the lhs's of a rewrite rule and all proper subterms of the rhs of the rule. An adaptation of the DP method to narrowing is given in [26] that requires the TRS to have the so-called *Top Reduced Almost Terminating* (TRAT) property, which can be briefly described as follows. Given a property P on terms, a term t is said to be a *minimal P term* if t satisfies P but none of the proper subterms of t does. Given a TRS \mathcal{R} and a binary relation \Rightarrow (being $\rightarrow_{\mathcal{R}}$ or $\sim_{\mathcal{R}}$), an infinite derivation $t \Rightarrow t_1 \Rightarrow t_2 \dots$ is called *almost terminating* if t is a minimal non-terminating term w.r.t. \Rightarrow . An almost terminating derivation $t \Rightarrow t_1 \Rightarrow t_2 \dots$ is called *top reduced* if it contains a derivation step at the root position. We say that \Rightarrow has the *top reduced almost terminating* property (TRAT) [26] if, for every non-terminating term t , there exists a top reduced almost-terminating sequence stemming from one subterm of t .

Let us briefly recall the notion of *context*. A context is a term with several occurrences of a fresh symbol \square . If $C[\]$ contains k occurrences of symbol \square at positions p_1, \dots, p_k , we write $C[t_1, \dots, t_k]$ to denote the term $(C[t_1]_{p_1}) \dots [t_k]_{p_k}$.

In [26] it is proved that every monotone relation has the TRAT property. Then, since the rewriting relation is monotone (i.e., $t \rightarrow_{\mathcal{R}} s$ implies $C[t] \rightarrow_{\mathcal{R}} C[s]$), then it has the TRAT property for every TRS \mathcal{R} (cf. [17, Lemma 1]). In term rewriting this ensures that, in every almost terminating, infinite term rewriting derivation, a rewriting step is given at the root.

Unfortunately, the narrowing relation is not monotone: $t \sim_{\sigma, \mathcal{R}} s$ does not mean that $C[t] \sim_{\sigma, \mathcal{R}} C[s]$, but that $C[t] \sim_{\sigma, \mathcal{R}} (C\sigma)[s]$ instead. For this reason narrowing does not generally satisfy TRAT, as shown in the following example.

Example 2. [6] Consider the TRS consisting of the rule $\mathbf{f}(\mathbf{f}(x)) \rightarrow x$, and the non-linear term $\mathbf{c}(\mathbf{f}(x), x)$. Then there does not exist any infinite narrowing derivation for the subterms, $\mathbf{f}(x)$ and x , whereas $\mathbf{c}(\mathbf{f}(x), x)$ can be infinitely narrowed without never performing a narrowing step at the root:

$$\mathbf{c}(\underline{\mathbf{f}(x)}, x) \rightsquigarrow_{\{x/\mathbf{f}(x')\}} \mathbf{c}(x', \underline{\mathbf{f}(x')}) \rightsquigarrow_{\{x'/\mathbf{f}(x'')\}} \mathbf{c}(\underline{\mathbf{f}(x'')}, x'') \dots$$

As shown by the above example, in the presence of non linearity the non-monotony of narrowing has undesirable effects for its termination, since *narrexes* can be brought into the context by the substitution computed at the preceding narrowing step, thus causing other terms in the context to grow. This *echoing* effect is the ultimate responsible for narrowing non-termination.

It is shown in [26] that narrowing has a monotone (or monotone-like) behaviour for some particular classes of TRSs, so that it happens to have the TRAT property on them, namely for right-linear TRSs (w.r.t. linear goals) and for constructor systems. Their adaptation of the DP method to narrowing only works for these classes. Let us note that a generalization of the TRAT property for narrowing can be found in [3], where it is called SQRNC (*Quasi stable rigidly normalized condition*), though in [3] we did not make the relation with TRAT explicit. For the TRSs considered in [26], the standard DP approach of term rewriting can be directly applied to narrowing without major changes.

The inspiration for this work comes from realizing that monotonicity is not really a *necessary* condition for the termination of narrowing, provided the partially computed substitutions do not *echo*, i.e., they do not bring narrexes into the context that might either introduce a term that does not terminate or *echo* again. Let us introduce the idea by means of one example.

Example 3. Consider the the non-linear input call $c(f(z), z)$ in the non-constructor TRS

$$f(g(x)) \rightarrow x \quad (1)$$

$$g(x) \rightarrow x \quad (2)$$

The only possible derivation for this term is finite: $c(f(z), z) \rightsquigarrow_{\{z/g(x)\}, \mathcal{R}} c(x, g(x)) \rightsquigarrow_{\{x/x'\}, \mathcal{R}} c(x', x')$, whereas the TRS, together with the considered non-linear input term, do not fit in any of the characterizations given for TRAT [26, 25, 27] or any decidable criteria for the termination of narrowing [2, 3, 6, 19]. Note that the argument $g(x)$ of the lhs of the first rule is a narrex.

Let us start with some lessons learnt from the termination of rewriting that would be good to transfer to the termination of narrowing. In rewriting (and narrowing), if a TRS is not terminating then there must be a minimal non-terminating term. In rewriting such a minimal non-terminating term is rooted by a defined symbol but this is not true for narrowing. As in [17], let us denote the set of all minimal non-terminating terms w.r.t. rewriting (resp. narrowing) by \mathcal{T}^∞ (resp. $\mathcal{T}_\infty^\infty$). The following definition is crucial.

Definition 1 (Echoing terms). *Let \mathcal{R} be a TRS. We define the set of minimal echoing terms w.r.t. \mathcal{R} , denoted by \mathcal{T}^\odot , as follows: $s \in \mathcal{T}^\odot$ if, given a fresh binary symbol c and a variable $x \in \text{Var}(s)$, then $c(s, x) \in \mathcal{T}_\infty^\infty$ but $s \notin \mathcal{T}_\infty^\infty$.*

Now, we provide our key result for the termination of narrowing. We write $s \supseteq t$ to denote that t is a subterm of s , and $s \triangleright t$ if t is a proper subterm of s .

Lemma 1. *Let \mathcal{R} be a TRS. For every term $t \in \mathcal{T}_\infty^\infty$, we have that either*

1. (TOP) *there exists a rewrite rule $l \rightarrow r \in \mathcal{R}$, substitutions σ, ρ , a term t' , and a non-variable subterm u of r such that $t \rightsquigarrow_{\rho, \mathcal{R}}^{\epsilon*} t' \xrightarrow{\epsilon}_{\sigma, l \rightarrow r} r\sigma \supseteq u$ and $u \in \mathcal{T}_\infty^\infty$;*
2. (HYBRID) *there are terms t', t'', u , substitutions ρ, σ , and a variable x s.t. $t \rightsquigarrow_{\rho, \mathcal{R}}^{\epsilon*} t' \xrightarrow{p}_{\sigma, \mathcal{R}} t''$, $x \in \text{Var}(t'|_p)$, $x\sigma \supseteq u$, and $u \in \mathcal{T}_\infty^\infty$; or*
3. (ECHOING) *there are terms t', t'', u , substitutions ρ, σ , and a variable x s.t. $t \rightsquigarrow_{\rho, \mathcal{R}}^{\epsilon*} t' \xrightarrow{p}_{\sigma, \mathcal{R}} t''$, $x \in \text{Var}(t'|_p)$, $x\sigma \supseteq u$, and $t'|_p, u \in \mathcal{T}^\odot$.*

Proof. Let D be an infinite narrowing sequence stemming from t . Since all proper subterms of t are terminating for narrowing, D must contain a narrowing step at the root position or there is a narrowing step which computes a substitution that carries a narrex.

- Let us consider the case where there is a narrowing step at the top and cases 2 and 3 do not apply. We consider the first such narrowing step at the root $t \rightsquigarrow_{\rho, \mathcal{R}}^{\epsilon*} t' \xrightarrow{\epsilon}_{\sigma, l \rightarrow r} r\sigma$ where $\sigma = \text{mgu}(t', l)$. By assumption, all proper subterms of t' are terminating for narrowing and thus terms brought by the substitution σ are terminating for narrowing. As $r\sigma$ is not terminating for narrowing, it has a subterm $u = r\sigma|_p$ such that u is not terminating for narrowing and, clearly, $p \in \text{Pos}(r)$ because all terms brought by σ are terminating for narrowing. Now, by structural induction on r , we easily prove that there is a minimal such u .

– Let us now consider that there is no narrowing step at the top, i.e., case 1 does not apply. We consider two cases:

- If there is a term t' and a subterm u of t' s.t. $t \xrightarrow[\rho, \mathcal{R}]{\epsilon^*} t' \supseteq u$ and $u \in \mathcal{T}^\infty$, then u is introduced by a binding of ρ , since all proper subterms of t are terminating for narrowing.
- If there is no such u , then the infinite narrowing derivation must be produced due to a rule $l \rightarrow r$ applied infinitely many times, where l, r and all their subterms are terminating for narrowing.

This can only happen if there is a proper subterm u of l s.t. $C[u] \xrightarrow[\theta, \mathcal{R}]{*} C'[u']_{p'} \xrightarrow[\theta', l \rightarrow r]{p'} C'\theta'[r\theta']_{p'} \xrightarrow[\theta'', \mathcal{R}]{*} C''[u'']_{p''} \xrightarrow[\theta'', l \rightarrow r]{p''} C''\theta''[r\theta'']_{p''} \dots$ and $u, u', u'', \dots \in \mathcal{T}^\infty$. \square

Informally, the lemma above distinguishes three different kinds of steps that may occur in a minimal narrowing derivation. A TOP step corresponds to the well-known case of [17, Lemma 1]) or the TRAT derivations of [26] where, roughly speaking, infiniteness is essentially due to unhampered recursion. ECHOING steps are unique to narrowing; intuitively, in a pure ECHOING derivation there is a narrex that *echoes*, causing terms from the lhs of a rule to enter a hole in the context (through a shared variable that acts as the input channel), as in Example 2. Finally the HYBRID case is unique to narrowing too, but it is not really interesting by itself; it is simply required for the lemma to pinpoint all possible kinds of steps occurring in a minimal narrowing derivation that may jeopardize termination.

4 Narrowing Dependency Pairs

In this section, we develop the notion of narrowing dependency pairs, and provide a sound and complete criterion for the termination of narrowing that is based on analyzing narrowing chains.

Let us call *root-defined* terms to those terms whose root symbol is a defined function. The intuitive idea behind our method is as follows. In order to construct the set of dependency pairs, we not only relate the lhs of each rule with the root-defined subterms occurring in the corresponding rhs, as in standard rewriting DP's, but also with its *own* root-defined subterms. The resulting set of dependency pairs faithfully captures the behaviour of infinite narrowing derivations which incrementally compute an infinite substitution, or put differently, where the substitution computed by narrowing contains an infinite term.

Suppose we split the substitution σ computed by a narrowing step $t \xrightarrow[l \rightarrow r, \sigma]{} s$ into two pieces, $\sigma \equiv \sigma_l \uplus \sigma_r$. The σ_l part of the substitution has the familiar effect of propagating narrexes from the left hand side to the right hand side of the rule. On the other hand, the σ_r part is responsible for the echoing of *narrexes* to the context that can fire a new narrowing step. These narrexes come from the subterms of the left hand side of the rule, as in Example 2 above, or from the term being narrowed itself, e.g. when $c(z, h(g(x), z))$ is narrowed to $c(g(x), 0)$ by using the rule $h(y, y) \rightarrow 0$ and most general unifier $\{z \mapsto g(x), y \mapsto g(x)\}$.

Although the narrexes coming from proper subterms of the narrex selected at the preceding step might cause non-termination, standard (rewriting) termination analyses already cope with them. However, narrexes coming from proper subterms of the lhs of the rules are specific to narrowing, and thus we focus on them in our notion of narrowing dependency pairs.

Notation Let $(\mathcal{D} \uplus \mathcal{C}, \mathcal{R})$ be a TRS defined over a signature \mathcal{F} . Let $\mathcal{F}^\# = \{f^\# \mid f \in \mathcal{D}\}$, where $f^\#$ is a fresh symbol (called *tuple symbol*) with the same arity as f . If $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is of the form $f(s_1, \dots, s_n)$ with f a defined symbol, then $t^\#$ denotes the term $f^\#(s_1, \dots, s_n)$. If Σ is a signature, $\Sigma^\#$ denotes the extension of Σ with the appropriate tuple symbols.

The following definition extends the traditional DPs with a novel kind of dependency pairs, which we call *left-dependency pairs*.

Definition 2 (Narrowing Dependency Pair). *Given a TRS \mathcal{R} , we have two types of narrowing dependency pairs:*

(1) $\text{filter}^\#(\text{pckt}(194.179.1.x:p, \text{dst}, \text{new}))$	\rightarrow	$\text{filter}^\#(\text{pckt}(\text{secure}, \text{dst}, \text{new}))$
(2) $\text{filter}^\#(\text{pckt}(194.179.1.x:p, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(\text{secure}, \text{dst}, \text{new})$
(3) $\text{filter}^\#(\text{pckt}(158.42.x.y:p, \text{dst}, \text{new}))$	\rightarrow	$\text{filter}^\#(\text{pckt}(\text{secure}, \text{dst}, \text{new}))$
(4) $\text{filter}^\#(\text{pckt}(158.42.x.y:p, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(\text{secure}, \text{dst}, \text{new})$
(5) $\text{pckt}^\#(10.1.1.1:p, \text{ppp0}, s)$	\rightarrow	$\text{pckt}^\#(123.23.1.1:p, \text{ppp0}, s)$
(6) $\text{pckt}^\#(10.1.1.2:p, \text{ppp0}, s)$	\rightarrow	$\text{pckt}^\#(123.23.1.1:p, \text{ppp0}, s)$
(7) $\text{filter}^\#(\text{pckt}(123.123.1.1:p, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(123.123.1.1:p, \text{dst}, \text{new})$
(8) $\text{pckt}^\#(\text{src}, 123.123.1.1:p, \text{new})$	\rightarrow	$\text{pckt}^\#(\text{src}, 10.1.1.1:p, \text{established})$
(9) $\text{pckt}^\#(\text{src}, 123.123.1.1:p, \text{new})$	\rightarrow	$\text{pckt}^\#(\text{src}, 10.1.1.2:p, \text{established})$
(10) $\text{filter}^\#(\text{pckt}(\text{src}, \text{dst}, \text{established}))$	\rightarrow	$\text{pckt}^\#(\text{src}, \text{dst}, \text{established})$
(11) $\text{filter}^\#(\text{pckt}(\text{eth0}, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(\text{eth0}, \text{dst}, \text{new})$
(12) $\text{filter}^\#(\text{pckt}(194.179.1.x:p, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(194.179.1.x:p, \text{dst}, \text{new})$
(13) $\text{filter}^\#(\text{pckt}(158.42.x.y:p, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(158.42.x.y:p, \text{dst}, \text{new})$
(14) $\text{filter}^\#(\text{pckt}(\text{secure}, \text{dst}:80, \text{new}))$	\rightarrow	$\text{pckt}^\#(\text{secure}, \text{dst}:80, \text{new})$
(15) $\text{filter}^\#(\text{pckt}(\text{secure}, \text{dst}:\text{other}, \text{new}))$	\rightarrow	$\text{pckt}^\#(\text{secure}, \text{dst}:\text{other}, \text{new})$
(16) $\text{filter}^\#(\text{pckt}(\text{ppp0}, \text{dst}, \text{new}))$	\rightarrow	$\text{pckt}^\#(\text{ppp0}, \text{dst}, \text{new})$
(17) $\text{pckt}^\#(\text{src}, 123.123.1.1;p, \text{new})$	\rightarrow	$\text{natroute}^\#(\text{pckt}(\text{src}, 10.1.1.1:p, \text{established}),$ $\text{pckt}(\text{src}, 10.1.1.2:p, \text{established}))$

Fig. 2. Dependency pairs of *FullPolicy*

- a lr-dependency pair (or standard² DP) of \mathcal{R} is a pair $l^\# \rightarrow t^\#$ where $l \rightarrow r \in \mathcal{R}$, $r \sqsupseteq t$, and $\text{root}(t) \in \mathcal{D}$.
- a ll-dependency pair (ll-DP) of \mathcal{R} is a pair $l^\# \rightarrow u^\#$ where $l \rightarrow r \in \mathcal{R}$, $l \triangleright u$, and $\text{root}(u) \in \mathcal{D}$.

The set of all (narrowing) dependency pairs of \mathcal{R} is denoted by $\text{NDP}_{\mathcal{R}}$.

Example 4. The TRS $\mathbf{f}(\mathbf{f}(x)) \rightarrow x$ of Example 2 has no lr-dependency pairs and the single ll-dependency pair $\mathbf{f}^\#(\mathbf{f}(x)) \rightarrow \mathbf{f}^\#(x)$.

Example 5. For the TRS of Example 1 we obtain the narrowing dependency pairs shown in Figure 2.

Recall that our purpose is to prove that there are no infinite narrowing derivations. Since dependency pairs model all function calls in \mathcal{R} , this is equivalent to proving that there are no infinite sequences or *chains* of narrowing dependency pairs.

For narrowing we consider suitable the following definition of chain. As in [15, 26], we assume that different occurrences of dependency pairs are variable disjoint. In the following, \mathcal{P} is usually a set of dependency pairs.

Definition 3 (Narrowing Chain). Let \mathcal{P}, \mathcal{R} be two TRS's. A sequence of narrowing dependency pairs $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \dots, s_n \rightarrow t_n \in \mathcal{P}$ is called a $(\mathcal{P}, \mathcal{R})$ -narrowing chain if there exist terms u_1, u_2, \dots, u_n and substitutions $\sigma_1, \rho_1, \sigma_2, \rho_2, \dots, \sigma_n, \rho_n$ s.t. $u_1 \xrightarrow{\epsilon}_{\sigma_1, s_1 \rightarrow t_1} t_1 \sigma_1 \xrightarrow{\epsilon^*}_{\rho_1, \mathcal{R}} u_2 \xrightarrow{\epsilon}_{\sigma_2, s_2 \rightarrow t_2} t_2 \sigma_2 \xrightarrow{\epsilon^*}_{\rho_2, \mathcal{R}} u_3 \cdots u_n \xrightarrow{\epsilon}_{\sigma_n, s_n \rightarrow t_n} t_n \sigma_n$.

We often omit the $(\mathcal{P}, \mathcal{R})$ prefix when referring to narrowing chains when it is clear from the context. The following result establishes the soundness of analyzing narrowing chains.

Lemma 2. Let \mathcal{R} be a TRS. For every $(\text{NDP}_{\mathcal{R}}, \mathcal{R})$ -narrowing chain $u_1^\# \xrightarrow{\epsilon}_{\sigma_1, \text{NDP}_{\mathcal{R}}} t_1^\# \sigma_1 \xrightarrow{\epsilon^*}_{\rho_1, \mathcal{R}} u_2^\# \xrightarrow{\epsilon}_{\sigma_2, \text{NDP}_{\mathcal{R}}} t_2^\# \sigma_2 \xrightarrow{\epsilon^*}_{\rho_2, \mathcal{R}} u_3^\# \cdots u_n^\# \xrightarrow{\epsilon}_{\sigma_n, \text{NDP}_{\mathcal{R}}} t_n^\# \sigma_n$, there are contexts $C_1[\], \dots, C_{n+1}[\]$, positions p_1, \dots, p_{n+1} , and substitutions τ_1, \dots, τ_n s.t. $C_1[u_1]_{p_1} \xrightarrow{p_1}_{\tau_1, \mathcal{R}} C_2[t_1 \sigma_1]_{p_2} \xrightarrow{p_2^*}_{\rho_1, \mathcal{R}} C_2 \rho_1 [u_2]_{p_2} \xrightarrow{p_2}_{\tau_2, \mathcal{R}} C_3[t_2 \sigma_2]_{p_3} \xrightarrow{p_3^*}_{\rho_2, \mathcal{R}} C_3 \rho_2 [u_3]_{p_3} \cdots C_n \rho_{n-1} [u_n]_{p_n} \xrightarrow{p_n}_{\tau_n, \mathcal{R}} C_{n+1}[t_n \sigma_n]_{p_{n+1}}$.

² Modern formulations exclude pairs $l^\# \rightarrow u^\#$ when $l \triangleright u$. This refinement could be applied to lr-DPs in our definition, but the pair would not be actually discarded, since it is also computed as a ll-DP.

Proof. By induction on the number n of $\text{NDP}_{\mathcal{R}}$ steps in the chain. The case $n = 0$ is immediate. Let us consider the suffix of the chain that first narrows using the rules from \mathcal{R} and then a dependency pair from $\text{NDP}_{\mathcal{R}}$, i.e. $t_{n-1}^{\#} \sigma_{n-1} \xrightarrow{\rho_n, \mathcal{R}}^{\epsilon^*} u_n^{\#} \xrightarrow{\sigma_n, s_n^{\#} \rightarrow t_n^{\#}}^{\epsilon} t_n^{\#} \sigma_n$. By induction hypothesis, we assume there are contexts $C_1[\], \dots, C_n[\]$, positions p_1, \dots, p_n , and substitutions $\tau_1, \dots, \tau_{n-1}$ s.t. $C_1[u_1]_{p_1} \xrightarrow{\tau_1, \mathcal{R}}^{p_1^1} C_2[t_1 \sigma_1]_{p_2} \xrightarrow{\rho_1, \mathcal{R}}^{p_2^2} C_2 \rho_1[u_2]_{p_2} \xrightarrow{\tau_2, \mathcal{R}}^{p_2^2} C_3[t_2 \sigma_2]_{p_3} \xrightarrow{\rho_2, \mathcal{R}}^{p_3^3} C_3 \rho_2[u_3]_{p_3} \dots C_{n-1} \rho_{n-2}[u_{n-1}]_{p_{n-1}} \xrightarrow{\tau_{n-1}, \mathcal{R}}^{p_{n-1}^{n-1}} C_n[t_{n-1} \sigma_{n-1}]_{p_n}$. Now, let us consider the two possibilities for $s_n \rightarrow t_n \in \text{NDP}_{\mathcal{R}}$.

- If $s_n^{\#} \rightarrow t_n^{\#}$ is a vanilla dependency pair, then there is a rule $s_n \rightarrow r$ and a position q s.t. $r|_q = t_n$. Therefore, $t_{n-1}^{\#} \sigma_{n-1} \xrightarrow{\rho_n, \mathcal{R}}^{\epsilon^*} u_n^{\#} \xrightarrow{\sigma_n, s_n^{\#} \rightarrow t_n^{\#}}^{\epsilon} t_n^{\#} \sigma_n$ implies $C_n[t_{n-1} \sigma_{n-1}]_{p_n} \xrightarrow{\rho_n, \mathcal{R}}^{p_n^*} C_n \rho_n[u_n]_{p_n} \xrightarrow{\sigma_n, s_n \rightarrow r}^{p_n} C_n \rho_n \sigma_n[r \sigma_n]_{p_n}$, where there exists a context C_{n+1} s.t. $C_n \rho_n \sigma_n[r \sigma_n]_{p_n} = C_{n+1}[t_n \sigma_n]_q$ and $p_n \leq q$.
- If $s_n^{\#} \rightarrow t_n^{\#}$ is a ll-dependency pair, then there is a rule $s_n \rightarrow r$ and a position q s.t. $s_n|_q = t_n$. Therefore, $t_{n-1}^{\#} \sigma_{n-1} \xrightarrow{\rho_n, \mathcal{R}}^{\epsilon^*} u_n^{\#} \xrightarrow{\sigma_n, s_n^{\#} \rightarrow t_n^{\#}}^{\epsilon} t_n^{\#} \sigma_n$ implies there is a substitution τ_n and a variable $x \in \text{Var}((\)u_n) \cap \text{Var}((\)C_n \rho_n)$ s.t. $\tau_n(x) = t_n$ and $\tau_n(y) = \sigma_n(y)$ for any other variable y , and $C_n[t_{n-1} \sigma_{n-1}]_{p_n} \xrightarrow{\rho_n, \mathcal{R}}^{p_n^*} C_n \rho_n[u_n]_{p_n} \xrightarrow{\tau_n, s_n \rightarrow r}^{p_n} C_n \rho_n \tau_n[r \tau_n]_{p_n}$, where there exists a context C_{n+1} and a position $q' \in \text{Pos}_x(u_n)$ s.t. $C_n \rho_n \tau_n[r \tau_n]_{p_n} = C_{n+1}[t_n \sigma_n]_{q'}$. \square

Now we are able to show that, whenever there are no infinite narrowing chains, narrowing does terminate.

Theorem 1 (Termination Criterion). *A TRS \mathcal{R} is terminating for narrowing if and only if no infinite $(\text{NDP}_{\mathcal{R}}, \mathcal{R})$ -narrowing chain exists.*

Proof. The *if* case is straightforward from Lemma 1 and the *only if* case is straightforward from Lemma 2. \square

Example 6. Consider the ll-DP $d \equiv \mathbf{f}^{\#}(f(x)) \rightarrow \mathbf{f}^{\#}(x)$ of Example 4. There is a narrowing chain $\mathbf{f}^{\#}(x) \xrightarrow{\{x \mapsto \mathbf{f}(x')\}, d} \mathbf{f}^{\#}(x') \xrightarrow{\{x' \mapsto \mathbf{f}(x'')\}, d} \mathbf{f}^{\#}(x'') \dots$.

5 Automating the method

In order to automate the task of proving the absence of narrowing chains, it would be very convenient to reformulate the problem using only rewriting chains, as it is done e.g. in [26, 27, 30], since this allows us to reuse existing tools and techniques of the rewriting DP literature. We develop our method inspired by [25] but we provide proofs of all results without requiring TRAT, which is the main novel contribution of this section. Let us recall the notion of argument filtering.

Definition 4 (Argument Filtering). [5] *An argument filtering (AF) for a signature \mathcal{F} is a mapping π that assigns to every n -ary function symbol $f \in \mathcal{F}$ an argument position $i \in \{1, \dots, n\}$, or a (possibly empty) list $[i_1, \dots, i_m]$ of argument positions with $1 \leq i_1 < \dots < i_m \leq n$. The signature \mathcal{F}_{π} consists of all function symbols \mathbf{f} s.t. $\pi(\mathbf{f})$ is some list $[i_1, \dots, i_m]$, where in \mathcal{F}_{π} the arity of \mathbf{f} is m . Every AF π induces a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to $\mathcal{T}(\mathcal{F}_{\pi}, \mathcal{V})$:*

$$\pi(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \pi(t_i) & \text{if } t = \mathbf{f}(t_1, \dots, t_n) \text{ and } \pi(\mathbf{f}) = i \\ \mathbf{f}(\pi(t_{i_1}), \dots, \pi(t_{i_m})) & \text{if } t = \mathbf{f}(t_1, \dots, t_n) \text{ and } \pi(\mathbf{f}) = [i_1, \dots, i_m] \end{cases}$$

We extend π to a TRS \mathcal{R} as follows.

$$\pi(\mathcal{R}) = \{\pi(l) \rightarrow \pi(r) \mid l \rightarrow r \in \mathcal{R} \text{ and } \pi(l) \neq \pi(r)\}$$

For any argument filtering π and ordering $>$, we define $s \geq_{\pi} t \iff \pi(s) > \pi(t) \text{ or } \pi(s) \equiv \pi(t)$.

We also define the filtering of a position p w.r.t. a term t as follows. Given a n -ary symbol $f \in \mathcal{F}$ and $i \in \{1, \dots, n\}$, $\pi(i, f) = j$ if $\pi(f) = [i_1, \dots, i_j, \dots, i_k]$, $i_j = i$. Given a term t and a position $p \in \text{Pos}(t)$, the filtering of p w.r.t. t is defined as follows:

$$\pi(p, t) = \begin{cases} \epsilon & \text{if } p = \epsilon \\ \pi(q, t) & \text{if } p = q.i, i \in \mathbb{N}, \pi(\text{root}(t|_q)) = i \\ \pi(q, t).\pi(i, \text{root}(t|_q)) & \text{if } p = q.i, i \in \mathbb{N}, \pi(\text{root}(t|_q)) = [i_1, \dots, i, \dots, i_k] \end{cases}$$

Example 7. Consider the TRS of Example 1 and the argument filtering $\pi(\text{pkt}) = \{1, 3\}$ and $\pi(f) = \{1, \dots, \text{ar}(f)\}$ for any other $f \in \mathcal{F}$. Let us consider the term $t = \text{filter}(\text{pkt}(\text{secure}, \text{dst}, \text{new}))$, its filtered version is $\pi(t) = \text{filter}(\text{pkt}(\text{secure}, \text{new}))$ and the filtering of position 1.3 is $\pi(1.3, \pi(t)) = 1.2$ where $\pi(1.2, \pi(t))$ is undefined.

Definition 5. Given a TRS \mathcal{R} and an AF π , we say that π is a sound AF for \mathcal{R} iff $\pi(\mathcal{R})$ is a TRS, i.e., the rhs's of the rules do not contain extra variables not appearing in the corresponding lhs.

Our main result in this section is Theorem 2 below that relates infinite narrowing $(\mathcal{P}, \mathcal{R})$ -chains to infinite rewriting $(\pi(\mathcal{P}), \pi(\mathcal{R}))$ -chains. In order to prove this result, we first need two auxiliary lemmata. The first one establishes a correspondence between rewriting derivations in \mathcal{R} and derivations in the filtered TRS $\pi(\mathcal{R})$.

Lemma 3. Given a TRS \mathcal{R} , a sound AF π , and terms s and t , $s \rightarrow_{\mathcal{R}}^* t$ implies $\pi(s) \rightarrow_{\pi(\mathcal{R})}^* \pi(t)$. Moreover, the derivation in $\pi(\mathcal{R})$ uses the same rules in the same order at the corresponding filtered positions (whenever the filtered position exists).

The second lemma extends the correspondence established in Lemma 3 to narrowing, which can be done only when the original filtered term is ground. The key point is that the correspondence holds regardless of the substitution computed by narrowing. It is in fact a (one-way) lifting lemma from narrowing derivations in \mathcal{R} to rewriting sequences in $\pi(\mathcal{R})$.

Lemma 4. Given a TRS \mathcal{R} and a sound AF π , let s and t be terms s.t. $\pi(s)$ is ground. Then $s \rightsquigarrow_{\sigma, \mathcal{R}}^* t$ implies $\pi(s) \rightarrow_{\pi(\mathcal{R})}^* \pi(t)$. Moreover, the derivation in $\pi(\mathcal{R})$ use the same rules in the same order at the corresponding filtered positions (whenever the filtered position exists).

Proof. By the soundness of narrowing proved in Hullot's Lifting Lemma [19], $s \rightsquigarrow_{\sigma, \mathcal{R}}^* t$ implies that $\sigma s \rightarrow_{\mathcal{R}}^* \sigma t$, and from Lemma 3 we deduce that $\pi(\sigma s) \rightarrow_{\pi(\mathcal{R})}^* \pi(\sigma t)$. And finally, since $\pi(s)$ is ground, $\pi(t)$ must also be ground as π is sound and thus $\pi(\mathcal{R})$ is a TRS, hence $\pi(\sigma s) \rightarrow_{\pi(\mathcal{R})}^* \pi(\sigma t) \equiv \pi(s) \rightarrow_{\pi(\mathcal{R})}^* \pi(t)$. \square

Let us recall here the standard definition of chain for rewriting.

Definition 6 (Chain). [5, 15] Let \mathcal{P}, \mathcal{R} be two TRS's. A (possibly infinite) sequence of pairs $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \dots$ from \mathcal{P} is a $(\mathcal{P}, \mathcal{R})$ -chain if there exists a substitution σ with $t_i \sigma \rightarrow s_{i+1}$ for all i .

The following result allows us to prove the absence of narrowing chains by analyzing standard rewriting chains. This is very useful because it means that we can reuse all the DP infrastructure available for rewriting.

Theorem 2. Let \mathcal{R} be a TRS over a signature \mathcal{F} , \mathcal{P} be a TRS over a signature $\mathcal{F}^\#$, and π a sound AF over $\mathcal{F}^\#$ s.t. $\pi(t)$ is ground for at least one pair $s \rightarrow t \in \mathcal{P}$ in every $(\mathcal{P}, \mathcal{R})$ -narrowing chain. If there exists no infinite $(\pi(\mathcal{P}), \pi(\mathcal{R}))$ -chain, then there exists no infinite $(\mathcal{P}, \mathcal{R})$ -narrowing chain.

Proof. By contradiction. Suppose there is such an infinite chain, and therefore an infinite narrowing derivation exists of the form

$$t_1 \xrightarrow{\varepsilon}_{\sigma_1, \mathcal{P}} u_1 \rightsquigarrow_{\sigma_2, \mathcal{R}}^* t_2 \xrightarrow{\varepsilon}_{\sigma_2, \mathcal{P}} u_2 \rightsquigarrow_{\sigma_3, \mathcal{R}}^* \dots$$

As some $\pi(u_i)$ is ground by assumption, by Lemma 4 there is a rewriting derivation in $\pi(\mathcal{R})$ of the form

$$\pi(u_i) \rightarrow_{\pi(\mathcal{R})}^* \pi(t_1) \xrightarrow{\varepsilon}_{\pi(\mathcal{P})} \pi(u_{i+1}) \xrightarrow{\varepsilon}_{\pi(\mathcal{P})} \dots$$

which by assumption must be finite. Note that every step given with a rule from \mathcal{P} is given in the filtered derivation too, as those steps are given at the root position. By Lemma 4, finiteness of the filtered derivation implies that the narrowing derivation cannot have infinite \mathcal{P} steps, and we reach a contradiction which proves the theorem. \square

The following straightforward consequence of Theorems 1 and 2 characterizes narrowing termination as a rewriting problem.

Corollary 1. *Let \mathcal{R} be a TRS over a signature \mathcal{F} , and π a sound AF over $\mathcal{F}^\#$ s.t. $\pi(t)$ is ground for at least one pair $s \rightarrow t \in NDP_{\mathcal{R}}$ in every $(NDP_{\mathcal{R}}, \mathcal{R})$ -narrowing chain. If there exists no infinite $(\pi(NDP_{\mathcal{R}}), \pi(\mathcal{R}))$ -chain, then narrowing terminates in \mathcal{R} .*

5.1 Extending the DP framework to narrowing

By means of Theorem 2, it is possible now to recast the problem of narrowing termination in the DP framework of [13]. In this framework a DP problem is a tuple $(\mathcal{P}, \mathcal{R})$ of two TRSs, \mathcal{R} and \mathcal{P} , where initially $\mathcal{P} = NDP_{\mathcal{R}}$. If there is no associated infinite narrowing chain, we say that the problem is *finite*. Termination methods are then formulated as *DP processors* that take a DP problem and return a new set of DP problems. A DP processor is *sound* if the input problem is finite whenever all the output problems are. We speak of *narrowing DP problems* to distinguish them from the standard ones.

In the usual style [27, 30], we show here how to adapt a few of the most important DP processors, and then give one that transforms a narrowing DP problem into a rewriting one, which allows us to use any of the existing DP processors for termination of rewriting.

The following definition adapts the standard notion of dependency graph to our setting by simply considering narrowing dependency pairs instead of vanilla DPs.

Definition 7 (Dependency Graph). *Given a (narrowing) DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ its (resp. narrowing) dependency graph is the directed graph where the nodes are the elements of \mathcal{P} , and there is an edge from $s \rightarrow t \in \mathcal{P}$ to $u \rightarrow v \in \mathcal{P}$ if $s \rightarrow t, u \rightarrow v$ is a (resp. narrowing) chain from \mathcal{P} .*

The theorem below establishes that the narrowing dependency graph of a narrowing DP problem is equal to the dependency graph of the rewriting DP problem defined by the same TRS and DP set.

Theorem 3. *Given a narrowing DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$, its narrowing dependency graph is the same as the dependency graph of the rewriting DP problem defined by $\langle \mathcal{P}, \mathcal{R} \rangle$.*

Proof. Let NG be the narrowing dependency graph for $\langle \mathcal{P}, \mathcal{R} \rangle$, and G be the dependency graph for the rewriting DP problem. Both graphs contain the elements of \mathcal{P} as nodes.

(\Rightarrow) We show that every edge in NG is also in G. From the definition of narrowing chain, there is an edge from $l^\# \rightarrow r^\#$ to $s^\# \rightarrow t^\#$ if there exist terms u_1, u_2 and substitutions σ_1, σ_2 s.t. $u_1 \xrightarrow{\varepsilon}_{\sigma_1, l^\# \rightarrow r^\#} r^\# \sigma_1 \xrightarrow{\geq \varepsilon^*}_{\rho_1, \mathcal{R}} u_2 \xrightarrow{\varepsilon}_{\sigma_2, s^\# \rightarrow t^\#} t^\# \sigma_2$.

The definition of rewriting chain requires that there exists a substitution σ s.t. $r^\# \sigma \rightarrow_{\mathcal{R}}^* s^\# \sigma$. By Hullot's lifting lemma such a σ exists and is equal to or less general than $\sigma_1 \rho_1 \sigma_2$.

(\Leftarrow) We show that for every edge in G is in NG too. There is an edge in G from $l^\# \rightarrow r^\#$ to $s^\# \rightarrow t^\#$ if there exists a substitution σ s.t. $\sigma r^\# \rightarrow_{\mathcal{R}}^* \sigma s^\#$. By the soundness of narrowing $r^\# \sigma \xrightarrow{\sim}_{id, \mathcal{R}}^* s^\# \sigma$. If we take $u_1 \equiv l^\# \sigma$, then the following narrowing chain exists: $u_1 \xrightarrow{\varepsilon}_{\sigma, l^\# \rightarrow r^\#} r^\# \sigma \xrightarrow{\sim}_{id, \mathcal{R}}^* s^\# \sigma \xrightarrow{\sim}_{\sigma, s^\# \rightarrow t^\#}^* t^\# \sigma$. \square

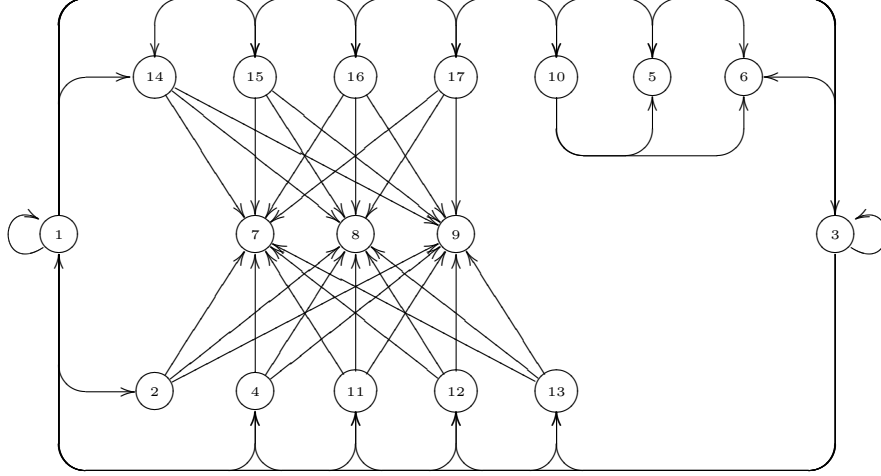


Fig. 3. Estimated dependency graph of *FullPolicy*

It is well known that computing the exact dependency graph is undecidable and thus several approximations [15, 18] are used to compute an *estimated* dependency graph which includes the exact graph. The following approximation is commonly used.

Definition 8 (Estimated Dependency Graph). [13] Let $\langle \mathcal{P}, \mathcal{R} \rangle$ be a DP problem. Let $\text{CAP}_{\mathcal{R}}(t)$ be the result of replacing³ all the proper subterms of t with a defined root symbol by a fresh variable, and $\text{REN}(t)$ the linearization of t (replacing all occurrences of a non linear variable with independent fresh variables). The nodes of the estimated dependency graph (EDG) are the pairs of \mathcal{P} and there is an edge from $s^{\#} \rightarrow t^{\#}$ to $u^{\#} \rightarrow v^{\#}$ iff $\text{REN}(\text{CAP}_{\mathcal{R}}(t))$ and u are unifiable.

Example 8. For the problem of Example 1 and the set of DPs obtained in Example 5, the EDG is shown in Figure 3.

For finite TRSs, chains show up as cycles in the dependency graph⁴. We can analyze separately every chain, that is, every cycle in the dependency graph. This is accomplished by the dependency graph processor.

Theorem 4 (Dependency Graph Processor). [13] For a DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$, Let *Proc* be the processor that returns problems $\{ \langle \mathcal{P}_1, \mathcal{R} \rangle, \dots, \langle \mathcal{P}_n, \mathcal{R} \rangle \}$, where $\mathcal{P}_1, \dots, \mathcal{P}_n$ are the sets of nodes of every cycle in the estimated dependency graph. *Proc* is sound.

Proof. Let EG be the EDG, G the exact one, and NG the narrowing one. One just needs to prove that every chain generates a cycle. For that it suffices to prove that every pair in a chain corresponds to an edge in a cycle in EG. By definition this holds for G, and NG. Hence one must show that $G \subseteq EG$. This has already been done in e.g. [13], and by Theorem 3 we have that $NG \equiv G$, which proves the claim. \square

Example 9. In the graph obtained in the EDG of Example 8, the only cycle consists of (1) and (3). Thus the dependency graph processor deletes all the other dependency pairs, and returns the problems $\{ (\{(1), (3)\}, \mathcal{R}), (\{(1)\}, \mathcal{R}), (\{(3)\}, \mathcal{R}) \}$.



³ This function was first defined for approximating loops in dependency graphs in [4], where it is called $\overset{\circ}{t}$.

⁴ The converse does not hold, not every cycle forms a chain.

The next processor we adapt is the standard reduction pair processor. The following is the standard notion of reduction pair.

Definition 9 (Reduction Pair). A reduction pair (\succeq, \succ) consists of a quasi-rewrite ordering \succeq and an ordering \succ with the following properties.

- \succ is closed under substitutions and well founded.
- $(\succeq \circ \succ) \subseteq \succ$ or $(\succ \circ \succeq) \subseteq \succ$.

For a narrowing DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$, this processor tries to find a reduction pair (\succeq, \succ) and a suitable filtering π s.t. all the filtered \mathcal{R} -rules are weakly decreasing w.r.t. \succeq , and all filtered \mathcal{P} pairs are weakly or strictly decreasing.

Notation For any TRS \mathcal{P} and relation \succ , let $\mathcal{P}_{\succ} = \{s \rightarrow t \mid s \succ t\}$

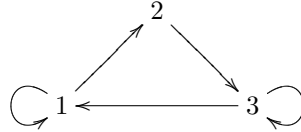
Theorem 5 (Reduction Pair processor). Let $(\mathcal{P}, \mathcal{R})$ be a narrowing DP problem s.t. \mathcal{P} is a cycle⁵, (\succeq, \succ) be a reduction pair, and π be an argument filtering s.t. $\pi(t)$ is ground for at least one pair $s \rightarrow t \in \pi(\mathcal{P})$. Then $\text{Proc}_{\pi}(\mathcal{P}, \mathcal{R})$ returns

- $\{(\mathcal{P} \setminus \mathcal{P}_{\text{succ}_{\pi}}, \mathcal{R})\}$ if $\mathcal{P}_{\succ_{\pi}} \cup \mathcal{P}_{\succeq_{\pi}} = \mathcal{P}$, $\mathcal{P}_{\succ_{\pi}}$ is not empty, and $\mathcal{R}_{\succeq_{\pi}} = \mathcal{R}$.
- $\{(\mathcal{P}, \mathcal{R})\}$ otherwise.

Proof. By [5, Theorem 11] the constraints guarantee that there is no infinite rewriting chain from \mathcal{P} in \mathcal{R} , which implies no infinite rewriting chain from $\pi(\mathcal{P})$ in $\pi(\mathcal{R})$. Now, by Theorem 2, there is no infinite narrowing chain from \mathcal{P} in \mathcal{R} . \square

Note that it is not enough to consider all the pairs in a strongly connected component (SCC) at once, as it is commonly done in rewriting, and that we consider cycles instead. The reason is that the condition of Theorem 2, groundness of one DP rhs per chain (cycle), would not be ensured when working with SCCs instead.

Example 10. Consider a TRS \mathcal{R} with the following Dependency Graph.



Our dependency graph processor decomposes this problem into three subproblems corresponding to the cycles $\{1\}, \{3\}$ and $\{1,2,3\}$. A SCC problem would consider only the last of these three. Suppose we did indeed use SCCs. The reduction pair processor defined above can synthesize a filtering π_2 s.t. the rhs of (2) is ground and an ordering s.t. (3) can be oriented strictly; upon doing so it will remove (3) of the DP problem, thus leaving only $\{1,2\}$. This eliminates two cycles at once, $\{3\}$ and $\{1,2,3\}$. But this is unsound, since we cannot eliminate the cycle in $\{3\}$ unless we find an argument filtering π_3 s.t. the rhs of (3) is ground and there is a suitable ordering.

We claim that it is straightforward to adapt most of the standard DP processors in order to deal with the grounding AF requirement, and due to lack of space we will present only one more processor, which can be used to transform a narrowing DP problem into an ordinary one. Afterwards, any existing DP processor for rewriting becomes applicable.

Theorem 6 (Argument Filtering Processor). Let $(\mathcal{P}, \mathcal{R})$ be a narrowing DP problem s.t. \mathcal{P} is a cycle, and π be an argument filtering s.t. $\pi(t)$ is ground for at least one pair $s \rightarrow t \in \pi(\mathcal{P})$. Then, $\text{Proc}_{\pi}(\mathcal{P}, \mathcal{R}) = \{(\mathcal{P}_{\pi}, \pi(\mathcal{R}))\}$, where \mathcal{P}_{π} is defined as follows.

$$\mathcal{P}_{\pi} = \{\pi(l) \rightarrow \pi(r) \mid l \rightarrow r \in \mathcal{P}, l \not\geq r\}$$

Proc_{π} is a sound narrowing DP processor.

⁵ Note that this requirement is easily fulfilled by running the dependency graph processor first.

Proof. Given the narrowing DP problem $P \equiv (\mathcal{P}, \mathcal{R})$, let $Proc_\pi(P) = P'$. If P' is finite then there are no infinite rewriting $(\mathcal{P}_\pi, \pi(\mathcal{R}))$ -chains, which means that there are no infinite $(\pi(\mathcal{P}), \pi(\mathcal{R}))$ -chains, as the set of discarded pairs $\pi(\mathcal{P}) \setminus \mathcal{P}_\pi$ cannot produce infinite rewriting chains [8]. By Theorem 2, this implies that there are no infinite $(\mathcal{P}, \mathcal{R})$ -narrowing chains. \square

Finally, we include the subterm refinement in the AF processor as it can be the case that the rhs of a DP becomes a subterm of the lhs after the filtering.

Example 11. The set of narrowing DP problems resulting of Example 9 can be solved by using the AF processor to transform them into rewriting problems.

- $(\{(1)\}, \mathcal{R})$

For this problem soundness requires that $\pi(F) = \{1, 3\}$. Using the identity for all other symbols, we get the following (rewriting) DP problem that is finite, as one can easily check with a modern termination tool implementing the DP method such as Aprove [14], or Mu-Term [1, 21]: $(\{\text{filter}^\#(\text{pckt}(194.179.1.x:p, \text{new})) \rightarrow \text{filter}^\#(\text{pckt}(\text{secure}, \text{new})), \mathcal{R})$

- $(\{(3)\}, \mathcal{R})$

In this case, we proceed in a similar way, and the same AF π allows us to transform the current subproblem into a finite (rewriting) DP problem.

- $(\{(1), (3)\}, \mathcal{R})$

Finally, by using the same AF π , we get a finite DP problem.

This finally proves that the FullPolicy TRS is terminating for narrowing.

6 Conclusion

We have introduced a new technique for termination proofs of narrowing via termination of rewriting that is based on a suitable generalization of dependency pairs. Although several refinements of the notion of dependency pairs such as [15, 17] had been proposed previously for termination analysis of TRSs, this is the first time that the notion of dependency pair has been *extended* to generally dealing with narrowing. This is possible because we first identified the problem of *echoing*, which is ultimately responsible for narrowing non-termination. Our contribution is threefold: 1) we ascertained the suitable notions that allow us to detect when the terms in a narrowing derivation actually do echo; 2) our approach leads to much weaker conditions for verifying the termination of narrowing that subsume all previously known narrowing termination criteria; 3) the resulting method can be effectively mechanized. Actually, we implemented our results in a tool that is publicly available⁶, and satisfactorily evaluated this tool on large example sets. We think that our termination results might be of great interest not only to experts of termination analysis but also to those researchers who work within the wide area of narrowing applications. Currently, we are working in developing good strategies and heuristics for computing the grounding argument filterings, as well as exploring new DP processors specific to narrowing.

Finally, there is a large number of strategies and restrictions of narrowing. Termination of narrowing implies the termination of any strategy, but obviously, strategies can terminate more often, as their main goal is usually cutting down the search space. It is not trivial at all to adapt the results presented here to narrowing strategies, but we are starting to consider the problem now too.

⁶ <http://ender4.dsic.upv.es/narradar>

Bibliography

- [1] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving termination of context-sensitive rewriting with Mu-Term. *Electr. Notes Theor. Comput. Sci.*, 188:105–115, 2007.
- [2] M. Alpuente, S. Escobar, and J. Iborra. Modular Termination of Basic Narrowing. In *Proc. 19th Int'l Conf. on Rewriting Techniques and Applications, RTA '08*, LNCS, Berlin, 2008. Springer-Verlag. To appear.
- [3] M. Alpuente, S. Escobar, and J. Iborra. Termination of Narrowing revisited. *Theoretical Computer Science*, 2008. To appear.
- [4] M. Alpuente, M. Falaschi, and G. Vidal. Compositional Analysis for Equational Horn Programs. In *4th Int'l Conf. on Algebraic and Logic Programming, ALP'94*, volume 850 of *LNCS*, pages 77–94. Springer, 1994.
- [5] T. Arts and J. Giesl. Termination of Term Rewriting using Dependency Pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.
- [6] J. Christian. Some termination criteria for narrowing and e-narrowing. In *11th Int'l Conf. on Automated Deduction CADE'92*, volume 607 of *LNCS*, pages 582–588. Springer, 1992.
- [7] N. Dershowitz. Goal Solving as Operational Semantics. In *Int'l Logic Programming Symposium, ILPS'95*, pages 3–17, Cambridge, MA, 1995. MIT Press.
- [8] N. Dershowitz. Termination dependencies. In *Proc. of the 6th Int'l Workshop on Termination*, Technical Report DSIC-II/15/03, pages 27–30, 2003.
- [9] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 244–320. Elsevier Science, 1990.
- [10] S. Escobar, C. Meadows, and J. Meseguer. A Rewriting-Based Inference System for the NRL Protocol Analyzer and its Meta-Logical Properties. *Theoretical Computer Science*, 367(1-2):162–202, 2006.
- [11] S. Escobar and J. Meseguer. Symbolic model checking of infinite-state systems using narrowing. In *18th Int'l Conference on Rewriting Techniques and Applications, RTA 2007*, volume 4533 of *Lecture Notes in Computer Science*, pages 153–168. Springer-Verlag, 2007.
- [12] M. Fay. First-Order Unification in an Equational Theory. In *4th Int'l Conference on Automated Deduction, CADE'79*, pages 161–167, 1979.
- [13] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *11th Int'l Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2005*, volume 3452 of *LNCS*, pages 301–331. Springer, 2005.
- [14] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with AProVe. In *Proc. 15th Int'l Conf. on Rewriting Techniques and Applications, RTA '04*, pages 210–220, 2004.
- [15] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *J. Autom. Reasoning*, 37(3):155–203, 2006.
- [16] M. Hanus. The Integration of Functions into Logic Programming: From Theory to Practice. *Journal of Logic Programming*, 19&20:583–628, 1994.
- [17] N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In *Proc. 15th Int'l Conf. on Rewriting Techniques and Applications, RTA '04*, volume 3091 of *LNCS*, pages 249–268. Springer, 2004.
- [18] N. Hirokawa and A. Middeldorp. Automating the Dependency Pair Method. *Inf. Comput.*, 199(1-2):172–199, 2005.
- [19] J.-M. Hullot. Canonical Forms and Unification. In *5th Int'l Conference on Automated Deduction CADE'80*, volume 87 of *LNCS*, pages 318–334, Berlin, 1980. Springer-Verlag.

- [20] C. Kirchner, H. Kirchner, and A. Santana de Oliveira. Analysis of Rewrite-Based Access Control Policies. In *Proc. 3rd Int'l Workshop on Security and Rewriting Techniques, SecreT 2008*. Elsevier ENTCS, 2008.
- [21] S. Lucas. Mu-Term: A tool for proving termination of context-sensitive rewriting. In *Proc. 15th Int'l Conf. on Rewriting Techniques and Applications, RTA'04*, LNCS, pages 200–209, 2004.
- [22] J. Meseguer. Multiparadigm logic programming. In *3rd Int'l Conference on Algebraic and Logic Programming, ALP'92*, volume 632 of *LNCS*, pages 158–200, Berlin, 1992. Springer-Verlag.
- [23] J. Meseguer and P. Thati. Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. *Higher-Order and Symbolic Computation*, 20(1-2):123–160, 2007.
- [24] M. T. Nguyen, P. Schneider-Kamp, D. de Schreye, and J. Giesl. Termination Analysis of Logic Programs based on Dependency Graphs. In *17th Int'l Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2007*, volume 4915 of *LNCS*, pages 8–22. Springer, 2008.
- [25] N. Nishida and K. Miura. Dependency graph method for proving termination of narrowing. In *8th Int'l Workshop on Termination, WST'06*, pages 12–16, 2006.
- [26] N. Nishida, M. Sakai, and T. Sakabe. Narrowing-based simulation of term rewriting systems with extra variables. *Electr. Notes Theor. Comput. Sci.*, 86(3), 2003.
- [27] N. Nishida and G. Vidal. Termination of Narrowing via Termination of Rewriting. 2008. Submitted for Publication. Preliminary version in *Proc. FLOPS 2008*, Springer LNCS 4989:113–129, 2008.
- [28] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.
- [29] A. Santana de Oliveira. *R  criture et Modularit   pour les Politiques de S  curit  *. PhD thesis, Universit   Henri Poincar  , 2008.
- [30] P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann. Automated Termination Analysis for Logic Programs by Term Rewriting. In *16th Int'l Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR'06*, volume 4407 of *LNCS*, pages 177–193. Springer, 2007.
- [31] TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, Cambridge, UK, 2003.