Automated Incremental Termination Proofs for Hierarchically Defined Term Rewriting Systems

Xavier Urbain

Laboratoire de Recherche en Informatique (LRI)
CNRS UMR 8623
Bât. 490, Université Paris-Sud, Centre d'Orsay
91405 Orsay Cedex, France
E-mail: urbain@lri.fr

Abstract. We propose the notion of *rewriting modules* in order to provide a structural and hierarchical approach of TRS. We define then *relative dependency pairs* built upon these modules which allow us to perform termination proofs incrementally. Important results can be expressed in that new framework (regarding $\mathcal{C}_\mathcal{E}$-termination for instance), and with help of π *extendable orderings*, we give effective new incremental methods for proving termination particularly suited for automation.

1 Introduction

Rewriting is used for specification, programming or in automated proofs. Yet, if structuring is a paradigm of 'clean' programming, a TRS is still considered in practice as a single set of rules and termination proofs are run on the whole system without taking any benefit of its possible modular structure.

Programs are (should be) developed in an incremental way: one defines some basic types or functions and then builds other functions which use the basic ones and so on. In other words, some functions are somehow created 'upon' previously defined ones. A hierarchical structure over functions naturally emerges from that incremental procedure. Since termination is a difficult issue—especially when it comes to automation of proofs—it would be useful to perform automated termination proofs incrementally, that is to use normalization information about 'basic' rules to show that adding 'new' ones will preserve termination.

Unfortunately, the termination property does not behave as well as we could expect when dealing with unions of TRS. As shown by Toyama [16] if two TRS are strongly normalizing, their union does not necessarily terminate, even if the two systems do not share any symbol. The significant work of Gramlich showed that projections were to blame and gave sufficient conditions for ensuring termination of unions of TRS possibly sharing constructors [8]. But with regard

to hierarchical unions, that is unions of TRS sharing function symbols, fewer results are known. Dershowitz proposed [6] conditions over a certain kind of hierarchical unions—so called constructor based unions—but those conditions are too restrictive in practice and not very suited for automation.

Until 1997, the usual way of automatically proving termination of a TRS was to show that the relation described by that system was included in a reduction ordering, that is showing that each rule was decreasing w.r.t. the strict part of the ordering. Since that ordering was well-founded, termination of the system followed. However, constraints hence induced required search for orderings that computers had difficulties to perform (if any suitable reduction ordering ever existed) and proofs where more likely run 'by hand' with *ad hoc* techniques.

In 1997 Arts & Giesl introduced the notion of *dependency pairs* [1,3],weakening constraints, and thus replacing the previously needed reduction ordering by a *weak* reduction ordering, more suited to automated search. Moreover, dependency pairs allow to prove termination of non-simply terminating TRS, that is of systems whose rules may be non-structural recursive like for instance $f(x,\dots) \to f(g(x),\dots)$ where g is not a constructor. The dependency pair approach consists of a structural analysis of rules, requiring for a suitable ordering that rules belong to its weak part and that only the so-called dependency pairs strictly decrease. A consistent approach in respect of our aim of modularity would then be to refine that structural analysis of rules depending on where the relevant rules lie within the hierarchy.

Thus, in order to give a general framework bringing modular structure of TRS to the fore, and so as to provide automated methods to prove termination incrementally, we will firstly recall some generalities, then in Sec. 3 we shall introduce *rewrite modules*, express unions of TRS in terms of modules and explain how to show out TRS hierarchical structure. Sec. 4 will deal with general termination difficulties in unions and will provide orderings with good properties with respect to projection (the so-called π *extendable orderings*). The framework of modules settled, *relative dependency pairs* will be defined in Sec. 5 and that powerful tool will lead us to Sec. 6 and thus to new criteria for modular termination (Thm 1 and Thm 2). Those results provide an incremental proof method illustrated with the complete example of Sec. 7. We shall then discuss in Sec. 8 how this work compares to others, especially to results of Arts & Giesl [2] and Dershowitz [6]. We will eventually conclude and give ideas on how we intend to apply and extend our framework.

2 Preliminaries

We recall usual notions about rewriting [7], and give our notations.

A *signature* $\mathcal{F}$ is a finite set of *symbols* with arities. Let X be a countable set of *variables*; $T(\mathcal{F}, X)$ denotes the set of finite *terms* on $\mathcal{F}$ and X. Terms can be seen as trees: root position is then denoted by Λ, symbol at root position in a term t by $\Lambda(t)$, $t|_p$ denotes the subterm of t at position p. A *substitution* is a mapping σ from variables to terms s.t. $\{x \in X \mid \sigma(x) \neq x\}$ is finite. We use postfix notation for substitution applications. A substitution can easily be extended to endomorphisms of $T(\mathcal{F}, X)$: $f(t_1, \dots, t_n)\sigma = f(t_1\sigma, \dots, t_n\sigma)$. $t\sigma$ is then called an *instance* of t.

A *rewrite relation* is a binary relation $\rightarrow$ on terms which is monotonic and closed under substitution; $\rightarrow^\star$ will denote its reflexive-transitive closure. A *term rewriting system* (TRS for short) over a signature[1] $\mathcal{F}$ and a set of variables X is a set $R(\mathcal{F})$ of *rewrite rules* $l \rightarrow r$. A TRS R defines a rewrite relation $\rightarrow_R$ the following way: $s \rightarrow_R t$ if there is a position p s.t. $s|p = l\sigma$ and $t = s[r\sigma]_p$ for a rule $l \rightarrow r \in R$ and a substitution σ. We then say that $s|_p$ is a *redex* and that s *reduces to* t *at position* p *with* $l \rightarrow r$ or, if the rule of R is not relevant, *with* R, respectively denoted $s \xrightarrow[l \rightarrow r]{p} t$ and $s \xrightarrow[R]{p} t$.

We restrict ourselves to the study of *finitely branching* TRS, i.e. TRS s.t. the set of rules that can be applied to a term is always finite. A term is *strongly normalizable* (SN) if it cannot reduce indefinitely. A rewrite relation is *strongly normalizing* or *terminates* if any term is SN. Termination is usually proven with the help of *reduction orderings* [5] or quasi-orderings with dependency pairs. We briefly recall what we need. A *term ordering*, also known as *ordering pair* [10] is a pair $(\succeq, \succ)$ of relations over $T(\mathcal{F}, X)$ such that: $\succeq$ is a quasi-ordering, i.e. reflexive and transitive; $\succ \subseteq \succeq - \preceq$ is a strict ordering, i.e. irreflexive and transitive and s.t. $\succ\subseteq\succeq\succ\subseteq\succ$, $\succ \cdot \succeq \subseteq \succ$ and $\succeq \cdot \succ \subseteq \succ$. A term ordering is said to be *well-founded* if there is no infinite strictly decreasing sequence $t_1 \succ t_2 \succ \cdots$; *stable* if both $\succ$ and $\succeq$ are stable under substitutions; *weakly* (resp. *strictly*) *monotonic* if for all terms t_1 and t_2, for all $f \in \mathcal{F}$, if $t_1 \succeq$ (resp. $\succ$) t_2 then $f(\dots, t_1, \dots) \succeq$ (resp. $\succ$) $f(\dots, t_2, \dots)$; A term ordering $(\succeq, \succ)$ is called a *weak (resp. strict) reduction ordering* if it is well-founded, stable and weakly (resp. strictly) monotonic.

3 Rewrite Modules

We define *hierarchical extensions* of TRS and introduce the notion of *rewrite modules*, thus providing a rather structural view of complex TRS that leads to an incremental approach.

[1] We shall omit the signature if there is no ambiguity.

3.1 Extensions & Modules

From an operational point of view, a module consists of new symbols together with rules that define them.

Definition 1. *A* module extending *a system $R_1(\mathcal{F}_1)$ is a couple $[\mathcal{F}_2 \mid R_2]$ s.t.:*

1. *$\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$;*
2. *R_2 is a TRS over $\mathcal{F}_1 \cup \mathcal{F}_2$;*
3. *For each $l \to r \in R_2$, $\Lambda(l) \in \mathcal{F}_2$.*

System $R_1 \cup R_2$ over $\mathcal{F}_1 \cup \mathcal{F}_2$ is then a hierarchical extension of system $R_1(\mathcal{F}_1)$ by module $[\mathcal{F}_2 \mid R_2]$*; we denote $R_1(\mathcal{F}_1) \leftarrow [\mathcal{F}_2 \mid R_2]$ that extension.*

The usual notions of unions can be expressed in terms of modules in a straightforward way. We will say that $[\mathcal{F}_1 \mid R_1]$ extends *$[\mathcal{F}_0 \mid R_0]$* regardless of *$[\mathcal{F}_2 \mid R_2]$ if $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$, $[\mathcal{F}_0 \mid R_0] \leftarrow [\mathcal{F}_1 \mid R_1]$ and $[\mathcal{F}_0 \mid R_0] \leftarrow [\mathcal{F}_2 \mid R_2]$; such extension is indeed a* union of composable TRS *[9, 12, 15]. We will talk about the* disjoint union *$R_1 \cup R_2$ if $[\mathcal{F}_1 \mid R_1]$ and $[\mathcal{F}_2 \mid R_2]$ extend $[\emptyset \mid \emptyset]$. The union will be* constructor sharing *if $[\mathcal{F}_1 \mid R_1]$ and $[\mathcal{F}_2 \mid R_2]$ extend $[\mathcal{F}_0 \mid \emptyset]$.*

A property P is said modular *for a specific kind of union if R_1 and R_2 having property P implies that the relevant union of R_1 and R_2 verifies P.*

3.2 Modular splitting up of TRS

One may in practice provide a TRS as a hierarchy of modules and use it as it is. Every system $R(\mathcal{F})$ can anyway be automatically split up as an extension by *minimal* modules: the ones that cannot be seen as extensions of non-empty systems (see Example 1 below). This is done in two steps:

1. Build a graph $\mathcal{G}$ over symbols of $\mathcal{F}$ s.t. there is an arc from x to y if and only if there is a rule $l \to r \in R$ such that $x = \Lambda(l)$ and y occurs in l or in r.
2. Pack together symbols occuring in strongly connected parts of $\mathcal{G}$, that is symbols f and g such that $f \underset{\mathcal{G}}{\to}^\star g$ and $g \underset{\mathcal{G}}{\to}^\star f$.

Now, let us consider 'packs' of $\mathcal{G}$ as signatures of our modules, we just have to add rules the left-hand side root symbols of which are in those and we are done. Note that there is no cycle in the resulting hierarchy since 'mutually recursively' defined symbols were packed together.

Remark 1. For the sake of a better readability we may afterward gather constructors symbols that can be reached from the same packs.

Example 1. Let us consider addition and multiplication *à la* Peàno.

$$\mathcal{F} = \{s, 0, +, \times\}$$
$$R : \begin{cases} \underbrace{x + 0 \to x,}_{} & \underbrace{x \times 0 \to 0,}_{} \\ \underbrace{x + s(y) \to s(x + y),}_{R_+} & \underbrace{x \times s(y) \to x + (x \times y).}_{R_\times} \end{cases}$$

A natural (for programmers) and modular way of seeing it is to consider addition built upon constructors, and multiplication upon addition and constructors. Describing this TRS in module framework consists in introducing $+$ with addition rules and *then* $\times$ with multiplication rules. The extension scheme is then:

$$[\{s, 0\} \mid \emptyset] \leftarrow [\{+\} \mid R_+] \leftarrow [\{\times\} \mid R_\times]$$

Note that those modules are minimal (in respect of Remark 1).

Further note that every system $R(\mathcal{F})$ can be seen that way as an extension of the empty system over *constructors* of $\mathcal{F}$ by the module consisting of *defined symbols*[2] (as signature part) together with *all rules* of R (as rules part).

4 Termination and unions of TRS

Unfortunately, termination is not a modular property of TRS, even for disjoint unions, as shown by Toyama's famous counter-example [16].

Example 2 (Toyama). These two TRS (over disjoint signatures) are terminating:

$$R_1 : \{ f(0, 1, x) \to f(x, x, x) \qquad R_2 : \begin{cases} g(x, y) \to x \\ g(x, y) \to y. \end{cases}$$

Their union nevertheless allows infinite reductions, for instance: $f(g(0,1), g(0,1), g(0,1)) \underset{R_2}{\longrightarrow} f(0, g(0,1), g(0,1)) \underset{R_2}{\longrightarrow} f(0, 1, g(0,1))$ $\underset{R_1}{\longrightarrow} f(g(0,1), g(0,1), g(0,1)) \underset{R_2}{\longrightarrow} \ldots$

One may observe that R_2 is not confluent, but similar counter-examples exist for confluent TRS [16]. However, problems come indeed with R_2 and its 'projective' behaviour as studied by Gramlich [8]. Propositions to contain projections lead to the definition of *$\mathcal{C}_\mathcal{E}$-termination*[3] [8, 15].

Definition 2. *Let G be a new (regarding all involved signatures) symbol; we denote π the projective TRS $\{G(x, y) \to x, G(x, y) \to y\}$.*

A system R is said $\mathcal{C}_\mathcal{E}$-terminating *if $R \cup \pi$ is terminating.*

[2] Constructors and defined symbols are here Arts & Giesl's notions.

[3] We use here the terminology introduced by Olhebusch [14].

That notion gives precious results. In particular: $\mathcal{C}_\mathcal{E}$-termination is a modular property for disjoint union of TRS [8, 15], as well as for union of composable TRS as shown by Kurihara & Ohushi [9].

Note that before the introduction of dependency pairs termination proofs were usually performed using *simplification orderings* (reduction ordering with subterm property), thus restricting automated proofs to *simplifying* TRS.

Definition 3. *A TRS R over $\mathcal{F}$ is said* simply terminating *(or* simplifying*) if $R \cup \{f(x_1, \ldots, x_i, \ldots, x_n) \to x_i | 1 \leq i \leq n$ and $f \in \mathcal{F}\}$ is terminating.*

Proposition 1 (Gramlich [8]). *A simplifying TRS $\mathcal{C}_\mathcal{E}$-terminates.*

Thanks to Proposition 1: lying between termination and *simple termination*, $\mathcal{C}_\mathcal{E}$-termination is far from being too restrictive a property in practice. The example of Sec. 7 and, for instance, all examples by Arts & Giesl [1–3] (even those that do not simply terminate) $\mathcal{C}_\mathcal{E}$-terminate. Actually, all known automated techniques prove indeed $\mathcal{C}_\mathcal{E}$-termination.

In order to specify a class of orderings making the most of $\mathcal{C}_\mathcal{E}$-termination, we define π extendable orderings.

Definition 4. *A term ordering $(\succeq, \succ)$ on $T(\mathcal{F}, X)$ is said to be* π extendable *if there is a reduction ordering $(\succeq', \succ')$ over $T(\mathcal{F} \cup \{G\}, X)$ s.t.: 1) The restriction of $(\succeq', \succ')$ on $T(\mathcal{F}, X)$ is exactly $(\succeq, \succ)$ and 2) $G(s,t) \succeq' s$ and $G(s,t) \succeq' t$ for all s and t in $T(\mathcal{F} \cup \{G\}, X)$.*

A π extendable ordering is a strong π extendable *ordering if both $(\succeq, \succ)$ and a suitable $(\succeq', \succ')$ are strictly monotonic. Otherwise it is* weak π extendable*.*

Consequently, if $(\succeq, \succ)$ is a strictly monotonic π extendable ordering and if $l \succ r$ for all rules $l \to r$ of R, then R $\mathcal{C}_\mathcal{E}$-terminates.

5 Relative Dependency Pairs

Now that we settled our modular approach, we would like to use it for proving termination. Most methods dealing with union of TRS use a notion of *rank*, that is rely on how signatures are interlaced within a term [8]. We define here *relative dependency pairs*, dependency pairs of modules with two aims in mind: Firstly to 'capture' signature swaps by selecting relevant subterms, secondly to 'forget' other defined symbols since the whole termination proof is supposed to be performed incrementally.

Definition 5. *Let $R_1(\mathcal{F}_1)$ and $[\mathcal{F}_2 \mid R_2]$ be such that $R_1(\mathcal{F}_1) \leftarrow [\mathcal{F}_2 \mid R_2]$.*

For each rule $l \to r \in R_2$, a pair $\langle l, r' \rangle$ where r' is a subterm of r such that $\Lambda(r') \in \mathcal{F}_2$ is called a dependency pair *of module $[\mathcal{F}_2 \mid R_2]$.*

The set of dependency pairs of all rules of a module M is denoted $DP(M)$.

Example 3 (Example 1 revisited). Dependency pairs of $[\mathcal{F}_\times \mid R_\times]$ consist in:

$$\mathrm{DP}([\mathcal{F}_\times \mid R_\times]) : \left\{ \langle x \times s(y), x \times y \rangle \right\}$$

Notice that $\langle x \times s(y), (x \times y) + x \rangle$ is *not* a dependency pair, unlike in Arts & Giesl non-modular approach[4].

Remark 2. For a TRS seen as an extension of constructors by rules and defined symbols, relative dependency pairs and Arts & Giesl ones coincide.

Definition 6. *A* dependency chain *of a module* $[\mathcal{F} \mid R]$ over R' *(with* $R \subseteq R'$*) is a sequence of pairs* $\ldots \langle s_j, t_j \rangle \ldots$ *of* $DP([\mathcal{F} \mid R])$ *together with a substitution* σ *such that for any two successive pairs* $\langle s_i, t_i \rangle$, $\langle s_{i+1}, t_{i+1} \rangle$*:*

$$t_i\sigma \xrightarrow[R']{\neq \Lambda}{}^{\star} s_{i+1}\sigma.$$

We shall consider *minimal* dependency chains, in the sense that each proper subterm of any left-hand side σ-instantiated is strongly normalizable by R'.

Proposition 2. *Let* R *be a TRS over* $\mathcal{F}$*, then* R *is* $\mathcal{C}_\mathcal{E}$*-strongly normalizing if and only if there is no infinite (dependency) chain of* $[\mathcal{F} \mid R]$ *over* $R \cup \pi$*.*

Proof. Straigthforward from Remark 2 and since $\mathrm{DP}(R \cup \pi) = \mathrm{DP}(R)$.

Corollary 1. *Let* $(\succeq, \succ)$ *be a (weak)* π *extendable ordering.*

If $l \succeq r$ *for all* $l \to r \in R$ *and* $s \succ t$ *for all* $\langle s, t \rangle \in DP(R)$*, then* R *is* $\mathcal{C}_\mathcal{E}$*-strongly normalizing.*

6 Incremental proofs

The modules framework is particularly efficient when it comes to modular and incremental proofs. We give our contributions for extensions of a system by, firstly one module, and secondly two disjoint modules. Thus, those apply to any hierarchical scheme. We obtain also effective reformulations of previous results.

6.1 One system and one module

The basic case, typically a hierarchical extension of a system. Relative dependency pairs give us lighter conditions over systems to ensure termination.

Theorem 1. *Let* $R_1(\mathcal{F}_1)$ *and* $[\mathcal{F}_2 \mid R_2]$ *be such that* $[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_2 \mid R_2]$*.*

1. *If* R_1 *is* $\mathcal{C}_\mathcal{E}$*-strongly normalizing and*
2. *If there is no infinite dependency chain of* $[\mathcal{F}_2 \mid R_2]$ *over* $R_1 \cup R_2$*,*

Then $R_1 \cup R_2$ *is strongly normalizing.*

[4] We do not consider here the restricted case of innermost termination [2] where *usable rules* modify the set of dependency pairs.

By contradiction: We will assume that there is an infinite dependency chain of $R_1 \cup R_2$, then we will conclude either on an infinite dependency chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2$, or on non $\mathcal{C}_\mathcal{E}$-termination of R_1 which contradicts premises.

Let us suppose $R_1 \cup R_2$ non terminating. Thus, there is an infinite dependency chain of $[\mathcal{F}_1 \cup \mathcal{F}_2 \mid R_1 \cup R_2]$ over $R_1 \cup R_2$. Among pairs of DP($[\mathcal{F}_1 \cup \mathcal{F}_2 \mid R_1 \cup R_2]$) are: Pairs of $[\mathcal{F}_1 \mid R_1]$; Pairs of $[\mathcal{F}_2 \mid R_2]$; Eventually pairs $\langle l, r' \rangle$ s.t. $l \to r \in R_2$ and r' is a subterm of r with $\Lambda(r')$ in $\mathcal{F}_1$.
On the first two cases we have knowledge. In order to get rid of the third kind, we need the following lemma.

Lemma 1. *Let $R_1(\mathcal{F}_1)$ and $[\mathcal{F}_2 \mid R_2]$ be such that $[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_2 \mid R_2]$. Then for each $\langle u_1, v_1 \rangle \in DP(R_1)$ and for each pair $\langle u_2, v_2 \rangle$ of $[\mathcal{F}_2 \mid R_2]$ there is no substitution σ such that: $v_1\sigma \xrightarrow{\neq\Lambda}{}^{\star} u_2\sigma$.*

Proof. Actually $\Lambda(u_2) \neq \Lambda(v_1)$ since $\Lambda(u_2\sigma) = \Lambda(u_2) \in \mathcal{F}_2$ and $\Lambda(v_1\sigma) = \Lambda(v_1) \in \mathcal{F}_1$.

According to Lemma 1, pairs of second and of third kinds can only follow pairs of second kind, pairs of first kind can only follow pairs of first and of third kind. Thus, we face three cases. The dependency chain either contains: Only pairs of second kind, that is pairs of $[\mathcal{F}_2 \mid R_2]$ or only pairs of first kind, that is pairs of $[\mathcal{F}_1 \mid R_1]$ or a finite number of pairs (maybe none) of second kind then *one* pair of third kind and then an infinity of pairs of first kind.

- First case: Infinite dependency chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2$ contradicts second premise of Theorem 1.
- Cases 2 & 3: In both cases we eventually get an infinite dependency chain of $[\mathcal{F}_1 \mid R_1]$ over $R_1 \cup R_2$. We are now going to show how that chain can be transformed into a chain of $[\mathcal{F}_1 \mid R_1]$ over $R_1 \cup \pi$. Doing so, we will end up with an infinite chain of $[\mathcal{F}_1 \mid R_1]$ over $R_1 \cup \pi$, that is an infinite chain of $[\mathcal{F}_1 \cup \{G\} \mid R_1 \cup \pi]$ contradicting first premise stating that R_1 is $\mathcal{C}_\mathcal{E}$-terminating.
  We actually use (here and also in proof of Theorem 2) a more general result (Lemma 2) whose proof provides a way to build such a suitable chain.

  Lemma 2. *Let S_1 and S_2 two TRS over $\mathcal{F}_1$. Let $S_3(\mathcal{F}_1 \cup \mathcal{F}_2)$ be such that:*
  - *$\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$;*
  - *For each $l \to r \in S_3$, $\Lambda(l) \in \mathcal{F}_2$.*

  Then from any infinite minimal chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup S_3$, it is possible to build an infinite chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup \pi$ with the same sequence of pairs but new instantiation and rewriting steps.

The remaining part of Theorem 1 proof consists in applying Lemma 2 with $R_1 = S_1 = S_2$ and $R_2 = S_3$. Hence for any infinite dependency chains of $[\mathcal{F}_1 \mid R_1]$ over $R_1 \cup R_2$ we can build a corresponding chain of $[\mathcal{F}_1 \mid R_1]$ over $R_1 \cup \pi$, that is a dependency chain of $R_1 \cup \pi$. Since R_1 is supposed to be $\mathcal{C}_\mathcal{E}$-terminating, we raise a contradiction. Q.e.d.

Now we have got to prove Lemma 2. The proof is rather technical and use some interpretation of terms akin to Gramlich's [8]. We denote $T_\infty(\mathcal{F}, X)$ the set of infinite terms over signature $\mathcal{F}$ and a set X of variables.

Definition 7. *Let S be $S_1 \cup S_2 \cup S_3$. Let $>$ be an arbitrary yet total ordering over $T(\mathcal{F}_1 \cup \mathcal{F}_2, X)$.*

Interpretation $I_S(x) : T(\mathcal{F}_1 \cup \mathcal{F}_2, X) \to T_\infty(\mathcal{F}_1 \cup \{G:2\} \cup \{\bot:0\}, X)$ can be defined as:

$$
\begin{aligned}
I_S(x) &= x \text{ if } x \in X \\
I_S(f(t_1 \dots t_n)) &= \begin{cases} f(I_S(t_1) \dots I_S(t_n)) \text{ if } f \in \mathcal{F}_1, \\ \mathit{Comb}(\mathit{Red}(f(t_1 \dots t_n))) \text{ if } f \in \mathcal{F}_2. \end{cases} \\
\text{where} \quad \mathit{Red}(t) &= \{I_S(t') / \; t \xrightarrow[S_1 \cup S_2 \cup S_3]{} t'\}, \\
\mathit{Comb}(\emptyset) &= \bot, \\
\mathit{Comb}(\{a\} \cup \mathit{set}\,) &= G(a, \mathit{Comb}(\mathit{set})) \text{ where for all } x \in \mathit{set}, a < x).
\end{aligned}
$$

Comb(E) is built from an unordered set E, ordering $>$ is then needed to remove any ambiguity.

Remark 3. Please note that interpreted terms appear in a 'comb-like' shape. Each 'tooth' being itself the interpretation of a one-step-reduced term, any of these can be reached by an appropriate $\underset{\pi_2}{\longrightarrow}^\star \underset{\pi_1}{\longrightarrow}$ reduction.

We give yet another few technical lemmas showing I_S good behaviour.

Lemma 3. *For each $t \in T(\mathcal{F}_1, X)$ and each substitution σ, $I(t\sigma) = tI(\sigma)$.*

Proof. Structural induction on t.

Lemma 4. *For all $t_1, \dots, t_n$ in $T(\mathcal{F}_1 \cup \mathcal{F}_2, X)$ and for any context C over $\mathcal{F}_1$ with n holes, $I_S(C[t_1, \dots, t_n]) = C[I_S(t_1), \dots, I_S(t_n)]$.*

Proof. Structural induction on C.

Lemma 5. *For any term t strongly normalizable by $S_1 \cup S_2 \cup S_3$, $I(t)$ is finite.*

Proof. Trivial since we are interested in finitely branching TRS only.

Lemma 6. *For any s and t in $T(\mathcal{F}_1 \cup \mathcal{F}_2, X)$, and $l \to r \in S_1 \cup S_2$*

$$\text{if } s \xrightarrow[l\to r]{p} t \text{ then } I(s) \xrightarrow[S_1\cup S_2\cup\pi]{}{}^{+} I(t).$$

Moreover, if $\Lambda(s) \in \mathcal{F}_1$ then $I(s) \xrightarrow[S_1\cup S_2\cup\pi]{\neq\Lambda}{}^{+} I(t)$.

Proof. Two cases depending on symbols occurring along path from Λ to p.
• If there are only symbols of $\mathcal{F}_1$ then $s = C[s_1, \ldots, l\sigma, \ldots, s_n]$, $s|_p = l\sigma$ and C is a context with n holes over $\mathcal{F}_1$.

$$\begin{aligned}
I(s) &= I(C[s_1, \ldots, l\sigma, \ldots, s_n]) \\
&= C[I(s_1), \ldots, I(l\sigma), \ldots, I(s_n)] \text{ (Lemma 4)} \\
&= C[I(s_1), \ldots, lI(\sigma), \ldots, I(s_n)] \text{ (Lemma 3)} \\
&\xrightarrow[S\cup S_2]{p} C[I(s_1), \ldots, rI(\sigma), \ldots, I(s_n)] \text{ (hypothesis)} \\
&= C[I(s_1), \ldots, I(r\sigma), \ldots, I(s_n)] \text{ (Lemma 3)} \\
&= I(C[s_1, \ldots, r\sigma, \ldots, s_n]) = I(t).
\end{aligned}$$

• If symbols of $\mathcal{F}_2$ occur, then there is a smallest $p' < p$ (w.r.t. prefix ordering) s.t. $\Lambda(s|_{p'}) \in \mathcal{F}_2$. We may again assume w.l.o.g. that $s = C[s_1, \ldots, s', \ldots, s_n]$ where C is a context with n holes (possibly empty) over $\mathcal{F}_1$, $p = p'q$ and $s|_{p'} = s'$ with $s' = C'[l\sigma] \xrightarrow[S\cup S_2]{} C'[r\sigma] = t'$. Then

$$I(s) = I(C[s_1, \ldots, s', \ldots, s_n]) = C[I(s_1), \ldots, I(s'), \ldots, I(s_n)] \text{ (Lemma 4).}$$

From Def. 7, $I(s') = \text{Comb}(\text{Red}(s'))$. But $s'|_q = l\sigma \xrightarrow[S_1\cup S_2]{} r\sigma$, thus, by definition of Red, $r\sigma \in \text{Red}(l\sigma)$. Hence, $I(t')$ is a subterm of $I(s')$ and $I(s') \underset{\pi}{\to}^{+} I(t')$. We get

$$\begin{aligned}
C[I(s_1), \ldots, I(s'), \ldots, I(s_n)] &\underset{\pi}{\to}^{+} C[I(s_1), \ldots I(t'), \ldots, I(s_n)] \\
&= I(C[s_1, \ldots, t', \ldots, s_n]) \text{ (Lemma 4)} = I(t).
\end{aligned}$$

Lemma 7. *For all s and t in $T(\mathcal{F}_1 \cup \mathcal{F}_2, X)$, if $s \xrightarrow[S_3]{p} t$ then $I(s) \underset{\pi}{\to}^{+} I(t)$.*

Moreover, if $\Lambda(s) \in \mathcal{F}_1$ then $I(s) \xrightarrow[\pi]{\neq\Lambda}{}^{+} I(t)$.

Proof. Similar to the proof of Lemma 6, case 2.

Proof of Lemma 2 Let $\langle u_1, v_1\rangle, \langle u_2, v_2\rangle, \ldots$ be a dependency chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup S_3$ with a substitution σ. Let σ' be the substitution such that for all x, $x\sigma' = I(x\sigma)$.

Since the considered chain is minimal, σ is strongly normalizable so, from Lemma 5, σ' is indeed a substitution to finite terms.

We show that $\langle u_1, v_1\rangle, \langle u_2, v_2\rangle, \ldots$ with σ' is a chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup \pi$. For that purpose we have to show that for each i, $v_i\sigma' \xrightarrow[S_1 \cup S_2 \cup \pi]{\neq \Lambda}{}^{\star} u_{i+1}\sigma'$.

We know that $v_i\sigma \xrightarrow[S_1 \cup S_2 \cup S_3]{\neq \Lambda}{}^{\star} u_{i+1}\sigma$. Let us consider any step $s \xrightarrow[S_1 \cup S_2 \cup S_3]{p} t$ from that reduction. Since we have $\Lambda(s) = \Lambda(t) = \Lambda(v_i) = \Lambda(u_{i+1}) \in \mathcal{F}_1$, either from Lemma 7 or Lemma 6 we get that $I(s) \xrightarrow[S_1 \cup S_2 \cup \pi]{\neq \Lambda}{}^{\star} I(t)$. Thus, putting steps together we obtain $I(v_i\sigma) \xrightarrow[S_1 \cup S_2 \cup \pi]{\neq \Lambda}{}^{\star} I(u_{i+1}\sigma)$. Since $I(v_i\sigma) = v_i\sigma'$ and $I(u_{i+1}\sigma) = u_{i+1}\sigma'$ from Lemma 3 we are done.

The following corollary enables us to compose our termination results, allowing this way incremental proofs.

Corollary 2. *Let $R_1(\mathcal{F}_1)$ and $[\mathcal{F}_2 \mid R_2]$ be such that $[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_2 \mid R_2]$.*

If R_1 is $\mathcal{C}_\mathcal{E}$-terminating and if there is no infinite dependency chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2 \cup \pi$, then $R_1 \cup R_2$ is $\mathcal{C}_\mathcal{E}$-terminating.

Proof. We apply Theorem 1 with $[\mathcal{F}_2 \cup \{G\} \mid R_2 \cup \pi]$ extending system R_1.

Those results can be turned into an effective method by Corollary 3.

Corollary 3. *Let $R_1(\mathcal{F}_1)$ and $[\mathcal{F}_2 \mid R_2]$ be such that $[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_2 \mid R_2]$.*

If R_1 is $\mathcal{C}_\mathcal{E}$-terminating and if there is a weak reduction ordering (resp. π extendable ordering) $(\succeq, \succ)$ such that: $R_1 \cup R_2 \subseteq \succeq$ and DP$([\mathcal{F}_2 \mid R_2]) \subseteq \succ$, then $R_1 \cup R_2$ is terminating (resp. $\mathcal{C}_\mathcal{E}$-terminating).

Proof. All chains of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2$ are actually finite. Since steps between pairs decrease w.r.t. $\succeq$ and DP steps decrease w.r.t. $\succ$, an infinite reduction would contradict the premise stating that $(\succeq, \succ)$ is well-founded.

6.2 One system and two modules

The other basic case, allowing us to apply our results to any hierarchical scheme.

Theorem 2. *Let $R_1(\mathcal{F}_1)$, $[\mathcal{F}_2 \mid R_2]$ and $[\mathcal{F}_3 \mid R_3]$ be such that:*
$[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_2 \mid R_2]$ and $[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_3 \mid R_3]$ with $\mathcal{F}_2 \cap \mathcal{F}_3 = \emptyset$.

1. *If $R_1 \cup R_2$ is $\mathcal{C}_\mathcal{E}$-strongly normalizing and*

2. *If there is no infinite dependency chain of* $[\mathcal{F}_3 \mid R_3]$ *over* $R_1 \cup R_3 \cup \pi$,

Then $R_1 \cup R_2 \cup R_3$ *is* $\mathcal{C}_\mathcal{E}$*-strongly normalizing.*

Remark 4. Note that R_2 **does not interfere** with the premise over $[\mathcal{F}_3 \mid R_3]$.

Further note that the $\mathcal{C}_\mathcal{E}$-premise cannot be omitted: Toyama's counter-example (see Example 2) would then be a special case of the relevant extension.

As a corollary and using Prop. 2 we obtain a previous result by Kurihara & Ohuchi [9]: $\mathcal{C}_\mathcal{E}$-termination is modular for composable unions of TRS.

Theorem 2 applies to automated proof when used as the following corollary:

Corollary 4. *Let* $R_1(\mathcal{F}_1)$, $[\mathcal{F}_2 \mid R_2]$ *and* $[\mathcal{F}_3 \mid R_3]$ *be such that:*
$[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_2 \mid R_2]$ *and* $[\mathcal{F}_1 \mid R_1] \leftarrow [\mathcal{F}_3 \mid R_3]$ *with* $\mathcal{F}_2 \cap \mathcal{F}_3 = \emptyset$.

If $R_1 \cup R_2$ *is* $\mathcal{C}_\mathcal{E}$*-terminating and if there is a* π *extendable weak reduction ordering* $(\succeq, \succ)$ *s.t.* $R_1 \cup R_3 \subseteq \succeq$ *and DP*$([\mathcal{F}_3 \mid R_3]) \subseteq \succ$, *then* $R_1 \cup R_2 \cup R_3$ *is* $\mathcal{C}_\mathcal{E}$*-terminating.*

7 A complete example

Consider system $R_\#$ over signature $\mathcal{F}_\# = \{\# : \textit{constant};\ 0, 1 : \textit{postfix unary}\}$ describing integers in binary notation: $R_\# = \{\#0 \to \#$. We want some arithmetic over them and define addition with $[\mathcal{F}_+ \mid R_+]$:

$$\mathcal{F}_+ \ \{+ : binary\},$$
$$R_+ \begin{cases} x + \# \to x, & x0 + y0 \to (x + y)0, & x0 + y1 \to (x + y)1, \\ \# + x \to x, & x1 + y0 \to (x + y)1, & x1 + y1 \to ((x + y) + \#1)0, \\ x + (y + z) \to (x + y) + z. \end{cases}$$

R_+ has critical pairs: its innermost termination does not implies termination and we cannot apply modularity criteria from Arts & Giesl [2]. We use relative dependency pairs and a polynomial interpretation.

$$\mathrm{DP}([\mathcal{F}_+ \mid R_+]) : \left\{ \begin{array}{l} \langle x0 + y0, x + y\rangle,\ \langle x0 + y1, x + y\rangle,\ \langle x1 + y0, x + y\rangle, \\ \langle x1 + y1, x + y\rangle,\ \langle x1 + y1, (x + y) + \#1)\rangle, \\ \langle x + (y + z), x + y\rangle,\ \langle x + (y + z), (x + y) + z\rangle. \end{array} \right\}$$

$[\![\#]\!] = 0,\ [\![0]\!](x) = [\![x]\!] + 1,\ [\![1]\!](x) = [\![x]\!] + 3,\ [\![+]\!](x, y) = [\![x]\!] + 2[\![y]\!] + 1.$
Since $\mathrm{DP}([\mathcal{F}_+ \mid R_+])$ strictly decreases and $R_\# \cup R_+ \cup \pi$ weakly decreases, by Cor. 3, $R_\# \cup R_+$ $\mathcal{C}_\mathcal{E}$-terminates. We might then add subtraction:

$$\mathcal{F}_- \ \{- : \textit{binary}\},$$
$$R_- \begin{cases} x - \# \to x, & x1 - y1 \to (x - y)0, & x0 - y0 \to (x - y)0, \\ \# - x \to \#, & x1 - y0 \to (x - y)1, & x0 - y1 \to ((x - y) - \#1)1. \end{cases}$$

Relative dependency pairs and polynomial interpretation again suffice to prove that $R_{\#} \cup R_{-}$ $\mathcal{C}_{\mathcal{E}}$-terminates. We use: $[\![\#]\!] = 0,\ [\![0]\!](x) = [\![x]\!] + 1,\ [\![1]\!](x) = [\![x]\!] + 1,\ [\![-]\!](x, y) = [\![x]\!]$. Dependency pairs of $[\mathcal{F}_{-} \mid R_{-}]$ strictly decrease while $R_{\#} \cup R_{-}$ weakly decreases w.r.t. to that interpretation. Applying Cor. 4 we conclude that $R_{\#} \cup R_{-} \cup R_{+}$ $\mathcal{C}_{\mathcal{E}}$-terminates.

In order to use comparison we need Booleans and provide $[\mathcal{F}_{\mathsf{Bool}} \mid R_{\mathsf{Bool}}]$.

$$\mathcal{F}_{\mathsf{Bool}}\ \{\mathit{true}, \mathit{false} : \mathit{constants};\ \neg : \mathit{unary};\ \wedge : \mathit{infix\ binary};\ \mathit{if} : \mathit{ternary}\},$$
$$R_{\mathsf{Bool}} \begin{cases} \neg(\mathit{true}) \to \mathit{false}, & \neg(\mathit{false}) \to \mathit{true}, & x \wedge \mathit{true} \to x, \\ x \wedge \mathit{false} \to \mathit{false}, & \mathit{if}(\mathit{true}, x, y) \to x, & \mathit{if}(\mathit{false}, x, y) \to y. \end{cases}$$

That system has no dependency pair, hence $\mathcal{C}_{\mathcal{E}}$-terminates.

We now define comparisons in $[\mathcal{F}_{ge} \mid R_{ge}]$ extending $R_{\#}$ and R_{Bool}.

$$\mathcal{F}_{ge}\ \{ge : binary\},$$
$$R_{ge} \begin{cases} ge(x0, y0) \to ge(x, y), & ge(\#, x1) \to \mathit{false}, & ge(x1, y0) \to ge(x, y), \\ ge(x1, y1) \to ge(x, y), & ge(x, \#) \to \mathit{true}, & ge(\#, x0) \to ge(\#, x), \\ ge(x0, y1) \to \neg ge(y, x). \end{cases}$$

Termination of $R_{\#} \cup R_{\mathsf{Bool}} \cup R_{ge}$ is proven using RPO with precedence $\{ge > \neg > (\mathit{true}, \mathit{false})\}$. RPO is a simplification ordering thus π extendable: that union $\mathcal{C}_{\mathcal{E}}$-terminates by Prop. 1. We may then apply Thm 2 to conclude that $R_{\#} \cup R_{\mathsf{Bool}} \cup R_{ge} \cup R_{+} \cup R_{-}$ $\mathcal{C}_{\mathcal{E}}$-terminates.

Let us provide logarithm *Log* over intergers. It is technically easier firstly to define $Log'(x) = Log(x) + 1$ with convention $Log'(0) = 0$.

$$\mathcal{F}_{Log'}\ \{Log' : \mathit{unary}\},\ R_{Log'} \begin{cases} Log'(\#) \to \#,\ Log'(x1) \to Log'(x) + \#1, \\ Log'(x0) \to \mathit{if}(ge(x, \#1), Log'(x) + \#1, \#). \end{cases}$$

We use again relative dependency pairs and a polynomial interpretation. $\mathrm{DP}([\mathcal{F}_{Log'} \mid R_{Log'}]) : \{\langle Log'(x1), Log'(x)\rangle,\ \langle Log'(x0), Log'(x)\rangle\}$. For $[\![\#]\!] = 0,\ [\![0]\!](x) = [\![x]\!] + 1,\ [\![1]\!](x) = [\![x]\!] + 1,\ [\![+]\!](x, y) = [\![x]\!] + [\![y]\!]\ [\![\mathit{false}]\!] = 0,\ [\![\mathit{true}]\!] = 0,\ [\![\neg]\!](x) = 0,\ [\![ge]\!](x) = 0,\ [\![\mathit{if}]\!](x, y, z) = [\![y]\!] + [\![z]\!],\ [\![\wedge]\!](x, y) = [\![x]\!],\ [\![Log']\!](x) = [\![x]\!]$, relative dependency pairs strictly decrease while rules of $R_{\#} \cup R_{+} \cup R_{\mathsf{Bool}} \cup R_{ge} \cup R_{Log'}$ weakly decrease. By Cor. 4 we prove $\mathcal{C}_{\mathcal{E}}$-termination of $R_{\#} \cup R_{+} \cup R_{\mathsf{Bool}} \cup R_{ge} \cup R_{Log'} \cup R_{-}$.

We may now compute logarithm:

$$\mathcal{F}_{Log}\ \{Log : \mathit{unary}\},\ R_{Log}\ \{Log(x) \to Log'(x) - \#1.$$

Since $[\mathcal{F}_{Log} \mid R_{Log}]$ has no dependency pair, applying Thm 2 we conclude on $\mathcal{C}_{\mathcal{E}}$-termination of $R_{\#} \cup R_{+} \cup R_{-} \cup R_{\mathsf{Bool}} \cup R_{ge} \cup R_{Log'} \cup R_{Log}$.

8 Related Work

Our work has to compare firstly with Dershowitz's results [6]. Since we do not restrict ourselves to constructor based systems, and use a slightly more general definition of *hierarchical extensions*, we thus obtain more general conditions. Moreover our criteria (fully syntactical and applicable to most TRS met in practice) seem more suited for automation—because they were designed for it—than the finely tuned conditions of Dershowitz. Secondly, Arts & Giesl exploit the modular structure of dependency graphs [2]. Still, their criterion puts conditions over the whole system (whatever the extension might be). That is a drawback we wanted to get rid of, because it fundamentally acts as a break upon real incremental proving. As noticed in Rem. 4, criteria based on Thm 2 do not require anything from irrelevant sets of rules. Our framework furthermore provides for the general case the results they got for the special case of *innermost* rewriting.

9 Conclusion & Future Work

We proposed the notion of rewrite modules in order to express structural and hierarchical information about TRS. Different kinds of extensions can be seen as special cases of modules hierarchy schemes. Modules are a fertile ground for automated incremental termination proofs: we introduced relative dependency pairs and criteria that allow proofs to be split up an run in an incremental fashion. We eventually obtain as a corollary a former result from Kurihara & Ohuchi [9] that is directly and easily expressed in that framework.

Relative dependency pairs approach with the help of π extendable orderings provide criteria particularly suited for automation. The resulting tests are finite and purely syntactical, thus implementable. Moreover that method weakens constraints over orderings to be found in two ways. Firstly, the (really) incremental scheme itself filters only relevant rules; secondly, the relative dependency pairs widen the class of suitable orderings over these relevant rules.

Marked pairs, dependency graphs and thus graphs refinements can be defined in the module framework; that extension is straightforward. Another extension, quite important in practice, is the application of modules to TRS with associative and commutative symbols. This could lead to relative pairs somewhat similar to AC-extended pairs [11].

Finally, modules and relative dependency pairs are now full parts of the C*i*ME2 [4] termination tool developped in our research team.

Acknowledgments The author would like to thank Claude Marché for his most valuable comments and is grateful to Bernhard Gramlich for his judicious remarks.

References

1. T. Arts. *Automatically proving termination and innermost normalisation of term rewriting systems*. PhD thesis, Universiteit Utrecht, 1997.
2. T. Arts and J. Giesl. Modularity of termination using dependency pairs. In Nipkow [13], pages 226–240.
3. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Comput. Sci.*, 236:133–178, 2000.
4. E. Contejean, C. Marché, B. Monate, and X. Urbain. Cime version 2, 2000. Prerelease available at `http://www.lri.fr/~demons/cime.html`.
5. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69–115, Feb. 1987.
6. N. Dershowitz. Hierarchical termination. In N. Dershowitz and N. Lindenstrauss, editors, *Proceedings of the Fourth International Workshop on Conditional and Typed Rewriting Systems (Jerusalem, Israel, July 1994)*, volume 968, pages 89–105, Berlin, 1995. Springer-Verlag.
7. N. Dershowitz and J.-P. Jouannaud. Notations for rewriting. *EATCS Bulletin*, 43:162–172, 1990.
8. B. Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 5:131–158, 1994.
9. M. Kurihara and A. Ohuchi. Decomposable termination of composable term rewriting systems. *IEICE*, E78–D(4):314–320, Apr. 1995.
10. K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In G. Nadathur, editor, *Principles and Practice of Declarative Programming, International Conference PPDP'99*, volume 1702 of *Lecture Notes in Computer Science*, pages 47–61, Paris, 1999. Springer-Verlag.
11. C. Marché and X. Urbain. Termination of associative-commutative rewriting by dependency pairs. In Nipkow [13], pages 241–255.
12. A. Middeldorp and Y. Toyama. Completeness of combinations of constructor systems. In *Proc. 4th Rewriting Techniques and Applications, LNCS 488*, Como, Italy, 1991.
13. T. Nipkow, editor. *9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, Tsukuba, Japan, Apr. 1998. Springer-Verlag.
14. E. Ohlebusch. On the modularity of termination of term rewriting systems. *Theoretical Comput. Sci.*, 136:333–360, 1994.
15. E. Ohlebusch. Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 20:1–41, 1995.
16. Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Inf. Process. Lett.*, 25:141–143, Apr. 1987.