Submitted to the International Conference on Rewriting Techniques and Applications
http://rewriting.loria.fr/rta/

CERTIFIED SUBTERM CRITERION AND CERTIFIED USABLE RULES

CHRISTIAN STERNAGEL [1] AND RENÉ THIEMANN [1]

[1] Institute of Computer Science, University of Innsbruck, Austria

E-mail address: christian.sternagel@uibk.ac.at
E-mail address: rene.thiemann@uibk.ac.at
URL: http://cl-informatik.uibk.ac.at/software/ceta

Abstract. In this paper we present our formalization of two important termination techniques for term rewrite systems: the subterm criterion and the reduction pair processor in combination with usable rules. For both techniques we developed executable check functions using the theorem prover Isabelle/HOL. These functions are able to certify the correct application of the formalized techniques in a given termination proof. As there are several variants of usable rules, we designed our check function in such a way that it accepts all known variants, even those which are not explicitly spelled out in previous papers.

We integrated our formalization in the publicly available IsaFoR-library. This led to a significant increase in the power of CeTA, a certified termination proof checker that is extracted from IsaFoR.

1. Introduction

Termination provers for term rewrite systems (TRSs) became more and more powerful in the last years. One reason is that a proof of termination no longer is just some reduction order which contains the rewrite relation of the TRS. Currently, most provers construct a proof in the dependency pair framework (DP framework). This allows to combine basic termination techniques in a flexible way. Hence, a termination proof is a tree where at each node a specific technique is applied. So instead of just stating the precedence of some lexicographic path order or giving some polynomial interpretation, current termination provers return proof trees consisting of many different techniques and reaching sizes of several megabytes. Thus, it would be too much work to check by hand whether these trees really form a valid proof. (Additionally, checking by hand does not provide a very high degree of confidence.)

It is regularly demonstrated that we cannot blindly trust in the output of termination provers. Every now and then, some termination prover delivers a faulty proof. Most often, this is only detected if there is another prover giving a contradicting answer on the same problem. To solve this problem, three systems have been developed over the last few years: CiME/Coccinelle [4, 5], Rainbow/CoLoR [3], and CeTA/IsaFoR [23]. These systems either certify or reject a given termination proof. Here, Coccinelle and CoLoR

This research is supported by FWF (Austrian Science Fund) project P18763.

are libraries on rewriting for Coq (http://coq.inria.fr) and IsaFoR is our library on rewriting for Isabelle [21]. (Throughout this paper we just write Isabelle whenever we refer to Isabelle/HOL.)

All of these certifiers can automatically certify termination proofs that are performed within the DP framework. In this framework one tries to simplify so called DP problems $(\mathcal{P}, \mathcal{R})$ by processors until all pairs in $\mathcal{P}$ are removed.

The reduction pair processor [12, 14] is the major technique to remove pairs. Consequently, it has been formalized in all three libraries. One of the conditions of the processor demands that all rules in $\mathcal{R}$ must be weakly decreasing. If this and all other conditions are satisfied then one can remove all strictly decreasing pairs. In this paper, we present the details about the formalization of two important extensions of the reduction pair processor.

The first extension is the subterm criterion [15]. By restricting the used "reduction pair" to the subterm relation in combination with simple projections, it is possible to ignore the $\mathcal{R}$-component of a DP problem. Note that the subterm criterion has independently (and only recently) been formalized for the Coccinelle-library [4]. Here, we present the first Isabelle formalization of this important technique.

The other extension is the integration of usable rules [9, 10, 12, 14, 24]. With this extension not all rules in $\mathcal{R}$ have to be weakly decreasing but only the usable rules which are most often a strict subset of $\mathcal{R}$. However, there are several definitions of usable rules where the most powerful ones ([10] and [12]) are incomparable.

$$\text{all rules} \supseteq \text{usable rules ([24])} \supseteq \text{usable rules ([9, 14])} \begin{array}{l} \not\supseteq \text{ usable rules ([10])} \\ \not\subseteq \text{ usable rules ([12])} \end{array}$$

Although it was often stated that a combination of the definitions of usable rules of [10] and [12] would be possible there never was a refereed paper which showed such a proof. (However, there have been unpublished soundness proofs of such a combined definition.) In this paper we not only present such a combined definition and the first corresponding *formalized* soundness proof, but we also simplified and extended the existing proofs. For example, we never construct filtered terms although we consider usable rules w.r.t. some argument filter. (An independent formalization of usable rules is present in Coccinelle. However, this formalization is unpublished and it only uses the variant of [14]: it does not feature the improvements from [10] and [12].) With these two extensions of the reduction pair processor we could increase the number of TRSs (from the Termination Problem Database) where a proof can be certified by our certifier CeTA by over 50%.

Note that all the proofs that are presented (or omitted) in the following, have been formalized in our Isabelle library IsaFoR. Hence, in the paper we merely give sketches of our "real" proofs. Our goal is to show the general proof outlines and help to understand the full proofs. Our library IsaFoR with all formalized proofs, the executable certifier CeTA, and all details about our experiments are available at CeTA's website:

http://cl-informatik.uibk.ac.at/software/ceta

The paper is structured as follows: In Sec. 2, we recapitulate the required notions and notations of term rewriting and the DP framework. In Sec. 3, we describe our formalization of the subterm criterion. The reduction pair processor with usable rules and its formalization is presented in Sec. 4. Then, in Sec. 5, we shortly describe how CeTA is obtained from IsaFoR and give a summary about our experiments. We finally conclude in Sec. 6.

2. Preliminaries

Term Rewriting. We assume familiarity with term rewriting [2]. Still, we recall the most important notions that are used later on. A *(first-order) term* t over a set of *variables* $\mathcal{V}$ and a set of *function symbols* $\mathcal{F}$ is either a variable $x \in \mathcal{V}$ or an n-ary function symbol $f \in \mathcal{F}$ applied to n argument terms $f(\vec{t_n})$. A *context* C is a term containing exactly one occurrence of the special constant $\Box$ (that is assumed to be distinct from symbols in $\mathcal{F}$). Replacing $\Box$ in a context C by a term t is denoted by $C[t]$. A term t is a *(proper) subterm* of a term s—written $(s \rhd t)$ $s \trianglerighteq t$—whenever there exists a context C $(\neq \Box)$, such that $s = C[t]$. We write $s \approx t$ iff s and t are unifiable. An *argument filter* π is a mapping from symbols to integers or lists of integers. It induces a mapping from terms to terms where $\pi(x) = x$, $\pi(f(\vec{t_n})) = \pi(t_i)$ if $\pi(f) = i$, and $\pi(f(\vec{t_n})) = f(\pi(t_{i_1}), \ldots, \pi(t_{i_k}))$ if $\pi(f) = [i_1, \ldots, i_k]$. Argument filters are also used to indicate which positions in a term are regarded. Then π maps symbols to sets of positions. It will be clear from the context which kind of argument filters is used.

A *rewrite rule* is a pair of terms $\ell \to r$ and a TRS $\mathcal{R}$ is a set of rewrite rules. The *rewrite relation (induced by $\mathcal{R}$)* $\to_{\mathcal{R}}$ is the closure under substitutions and under contexts of $\mathcal{R}$, i.e., $s \to_{\mathcal{R}} t$ iff there is a context C, a rewrite rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$. Reductions at the root are denoted by $\to_{\mathcal{R},\epsilon}$.

We say that an element t is *terminating / strongly normalizing (w.r.t. some binary relation S)*, and write $\mathsf{SN}_S(t)$, if it cannot start an infinite sequence

$$t = t_1 \; S \; t_2 \; S \; t_3 \; S \cdots .$$

The whole relation is terminating, written $\mathsf{SN}(S)$, if all elements are terminating w.r.t. it. For a TRS $\mathcal{R}$ and a term t, we write $\mathsf{SN}(\mathcal{R})$ and $\mathsf{SN}_{\mathcal{R}}(t)$ instead of $\mathsf{SN}(\to_{\mathcal{R}})$ and $\mathsf{SN}_{\to_{\mathcal{R}}}(t)$. We write S^+ for the transitive closure of S, and S^* is the reflexive transitive closure.

Lemma 2.1 (Properties of Subterms)**.**

(a) *stability:* $s \rhd t \Longrightarrow s\sigma \rhd t\sigma$
(b) *subterms preserve termination:* $\mathsf{SN}_{\mathcal{R}}(s) \wedge s \rhd t \Longrightarrow \mathsf{SN}_{\mathcal{R}}(t)$. ∎

Let $\to_{\mathsf{SN}(\mathcal{R})}$ denote the restriction of $\to_{\mathcal{R}}$ to terminating terms, i.e., $\{(s,t) \mid s \to_{\mathcal{R}} t \wedge \mathsf{SN}_{\mathcal{R}}(s)\}$. Let $\overset{\rhd}{\to}_{\mathsf{SN}(\mathcal{R})}$ denote the same relation extended by the restriction of $\rhd$ to terminating terms, i.e., $\to_{\mathsf{SN}(\mathcal{R})} \cup \{(s,t) \mid s \rhd t \wedge \mathsf{SN}_{\mathcal{R}}(s)\}$.

Lemma 2.2 (Termination Properties)**.** *Let S be some binary relation, let $\mathcal{R}$ be a TRS.*

(a) $\mathsf{SN}(S) \Longleftrightarrow \mathsf{SN}(S^+)$,
(b) $\mathsf{SN}(\overset{\rhd}{\to}_{\mathsf{SN}(\mathcal{R})})$,
(c) $\mathsf{SN}_S(s) \wedge (s,t) \in S \Longrightarrow \mathsf{SN}_S(t)$. ∎

Dependency Pair Framework. The DP framework [12] is a way to modularize termination proofs. Therefore, we switch from TRSs to so called DP problems, consisting of two TRSs. The *initial DP problem* for a TRS $\mathcal{R}$ is $(\mathsf{DP}(\mathcal{R}), \mathcal{R})$ where $\mathsf{DP}(\mathcal{R})$ are the *dependency pairs* of $\mathcal{R}$. A $(\mathcal{P}, \mathcal{R})$-*chain* is a possibly infinite derivation of the following form:

$$s_1\sigma_1 \to_{\mathcal{P}} t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2 \to_{\mathcal{P}} t_2\sigma_2 \to^*_{\mathcal{R}} s_3\sigma_3 \to_{\mathcal{P}} \cdots \qquad (\star)$$

where $s_i \to t_i \in \mathcal{P}$ for all $i > 0$ (this implies that $\mathcal{P}$-steps only occur at the root). If additionally every $t_i\sigma_i$ is terminating w.r.t. $\mathcal{R}$, then the chain is *minimal*. A DP problem

$(\mathcal{P},\mathcal{R})$ is called *finite* [12], if there is no minimal $(\mathcal{P},\mathcal{R})$-chain. Proving finiteness of a DP problem is done by simplifying $(\mathcal{P},\mathcal{R})$ by so called *processors* recursively, until the $\mathcal{P}$-components of all remaining DP problems are empty and therefore trivially finite. For this to be correct, the applied processors need to be *sound*. A processor *Proc* is sound whenever for all DP problems $(\mathcal{P},\mathcal{R})$ we have that finiteness of $(\mathcal{P}',\mathcal{R}')$ for all $(\mathcal{P}',\mathcal{R}') \in \mathit{Proc}(\mathcal{P},\mathcal{R})$ implies finiteness of $(\mathcal{P},\mathcal{R})$. The termination techniques that will be introduced in the following sections are all such sound processors.[1]

Example 2.3. In the following TRS $\mathcal{R}$ the term $\mathsf{set}(xs)$ evaluates to the list $[x \in xs \mid 0 < x]$ where duplicates are removed:

$$x < \mathsf{0} \to \bot, \tag{2.1}$$
$$\mathsf{0} < \mathsf{s}(y) \to \top, \tag{2.2}$$
$$\mathsf{s}(x) < \mathsf{s}(y) \to x < y, \tag{2.3}$$
$$\mathsf{set}(\mathsf{nil}) \to \mathsf{nil},$$
$$\mathsf{set}(x : z) \to \mathsf{if2}(\mathsf{0} < x, x, z),$$
$$\mathsf{del}(x, \mathsf{nil}) \to \mathsf{nil}, \tag{2.4}$$
$$\mathsf{del}(x, y : z) \to \mathsf{if}(x < y, y < x, x, y, z), \tag{2.5}$$
$$\mathsf{if}(\bot, \bot, x, y, z) \to \mathsf{del}(x, z), \tag{2.6}$$
$$\mathsf{if}(\top, b, x, y, z) \to y : \mathsf{del}(x, z), \tag{2.7}$$
$$\mathsf{if}(b, \top, x, y, z) \to y : \mathsf{del}(x, z), \tag{2.8}$$
$$\mathsf{if2}(\top, x, z) \to x : \mathsf{set}(\mathsf{del}(x, z)),$$
$$\mathsf{if2}(\bot, x, z) \to \mathsf{set}(z).$$

After computing the initial DP problem $(\mathsf{DP}(\mathcal{R}),\mathcal{R})$ we can split it into the three problems $(\{(2.9)\},\mathcal{R})$, $(\{(2.13)\text{–}(2.16)\},\mathcal{R})$, and $(\{(2.10)\text{–}(2.12)\},\mathcal{R})$. (This is done by applying the dependency graph processor [1, 10, 12, 14], a well-known technique to perform separate termination proofs for each recursive function.)

$$\mathsf{s}(x) <^\sharp \mathsf{s}(y) \to x <^\sharp y, \tag{2.9}$$
$$\mathsf{set}^\sharp(x : z) \to \mathsf{if2}^\sharp(\mathsf{0} < x, x, z), \tag{2.10}$$
$$\mathsf{if2}^\sharp(\top, x, z) \to \mathsf{set}^\sharp(\mathsf{del}(x, z)), \tag{2.11}$$
$$\mathsf{if2}^\sharp(\bot, x, z) \to \mathsf{set}^\sharp(z), \tag{2.12}$$
$$\mathsf{del}^\sharp(x, y : z) \to \mathsf{if}^\sharp(x < y, y < x, x, y, z), \tag{2.13}$$
$$\mathsf{if}^\sharp(\bot, \bot, x, y, z) \to \mathsf{del}^\sharp(x, z), \tag{2.14}$$
$$\mathsf{if}^\sharp(\top, b, x, y, z) \to \mathsf{del}^\sharp(x, z), \tag{2.15}$$
$$\mathsf{if}^\sharp(b, \top, x, y, z) \to \mathsf{del}^\sharp(x, z). \tag{2.16}$$

3. The Subterm Criterion

The subterm criterion [15] is a termination technique that can be employed as a processor of the DP framework. It may be seen as a variant of the reduction pair processor with an attached argument filtering [1]. The used orders ($\rhd$ and $\trianglerighteq$) allow to ignore the $\mathcal{R}$ component of a DP problem $(\mathcal{P},\mathcal{R})$. And the argument filtering is restricted to be a so called *simple projection*. A simple projection π maps a term to one of its arguments, i.e., $\pi(f(\vec{t}_n)) = t_i$ for some $0 < i \leqslant n$. For convenience we use R_π to denote the 'composition' of the binary relation on terms R and π, i.e., $(s,t) \in R_\pi$ iff $(\pi(s),\pi(t)) \in R$.

Theorem 3.1. *Finiteness of* $(\mathcal{P} \setminus \rhd_\pi,\mathcal{R})$ *implies finiteness of* $(\mathcal{P},\mathcal{R})$*, provided:*

[1] To be more precise, in IsaFoR it is shown that all these processors are chain identifying (`chain_identifying_proc`) which is a slightly stronger requirement than soundness [22, Chapter 7]. The reason is that chain identifying processors can easily be combined with semantic labeling [25]. However, we omit the details here and just refer to theory `DpFramework` for the interested reader.

(a) *all rules of* $\mathcal{P}$ *are oriented by* $\trianglerighteq_\pi$ *(i.e.,* $\mathcal{P} \subseteq \trianglerighteq_\pi$*)*
(b) *all lhss and rhss of* $\mathcal{P}$ *are non-variable and non-constant terms where the roots of rhss are not defined in* $\mathcal{R}$ *(i.e.,* $s = f(\vec{s}_n)$ *with* $n > 0$ *and* $t = g(\vec{t}_m)$ *with* $m > 0$ *and* $g \notin \mathcal{D}_\mathcal{R}$ *for all* $s \to t \in \mathcal{P}$*)*

Example 3.2. The DP problem $(\{(2.9)\}, \mathcal{R})$ from Ex. 2.3 can be solved using the simple projection $\pi(<^\sharp) = 1$, since $\pi(\mathsf{s}(x) <^\sharp \mathsf{s}(y)) = \mathsf{s}(x) \rhd x = \pi(x <^\sharp y)$. Taking $\pi(\mathsf{del}^\sharp) = 2$ and $\pi(\mathsf{if}^\sharp) = 5$ we can remove Pair (2.13) from $(\{(2.13)\text{–}(2.16)\}, \mathcal{R})$. The result $(\{(2.14)\text{–}(2.16)\}, \mathcal{R})$ is then solved by the dependency graph processor. Removing a pair from $(\{(2.10)\text{–}(2.12)\}, \mathcal{R})$ is impossible as there is no π such that Pair (2.11) is oriented. Note that $<^\sharp, \mathsf{del}^\sharp, \mathsf{if}^\sharp, \ldots \notin \mathcal{D}_\mathcal{R}$ whereas $<, \mathsf{del}, \mathsf{if}, \cdots \in \mathcal{D}_\mathcal{R}$.

Before we can prove Theorem 3.1, we need several lemmas. First, we prove that termination of some element w.r.t. some binary relation S is equivalent to termination of the same element w.r.t. S^+. Note that this is a more general result than Lem. 2.2(a) and thus allows termination analysis of a single term, no matter if the whole TRS is terminating.

Lemma 3.3. $\mathsf{SN}_S(t) \iff \mathsf{SN}_{S^+}(t)$.

Proof. The direction from right to left is trivial. For the other direction assume that t is not terminating w.r.t. S^+. Hence $t = t_1\ S^+\ t_2\ S^+\ t_3\ S^+ \cdots$. Let S' denote the restriction of R to terminating terms, i.e., $S' = \{(s,t) \mid s\ S\ t \wedge \mathsf{SN}_S(s)\}$. By definition we have $\mathsf{SN}(S')$ and with Lem. 2.2(a) also $\mathsf{SN}(S'^+)$. Using $\mathsf{SN}_S(t)$ and Lem. 2.2(c) together with the infinite sequence from above, we get $\mathsf{SN}_S(t_i)$ for all $i > 0$, and further $t_1\ S'^+\ t_2\ S'^+\ t_3\ S'^+ \cdots$. This contradicts $\mathsf{SN}(S'^+)$. ∎

Next consider a general result on infinite sequences conducted in the union of two binary relations N and S where often N is a non-strict relation and S a strongly normalizing relation. Intuitively it states the following: Assume that there is an infinite sequence of steps, where each step is an N-step or an S-step. Further assume that whenever there is an N-step directly followed by an S-step, those two steps can be turned into a single S-step. Additionally, there is no infinite S-sequence starting at the same point as the sequence we are reasoning about. Then, from some point in our sequence on, there are no more S-steps, i.e., it ends in N-steps. This is a versatile fact that is used at several places inside IsaFoR.

Lemma 3.4. *Let* N *and* S *be two binary relations over some carrier and* $\vec{q}$ *an infinite sequence of carrier elements. If*

(a) $(q_i, q_{i+1}) \in N \cup S$ *for all* $i > 0$,
(b) $N \circ S \subseteq S$, *and*
(c) $\mathsf{SN}_S(q_1)$,

then there is some j *such that for all* $i \geq j$ *we have* $(q_i, q_{i+1}) \in N \setminus S$.

Proof. For the sake of a contradiction assume that the lemma does not hold. Then, together with (a), we obtain $\forall i > 0.\, \exists j \geq i.\, (q_j, q_{j+1}) \in S$. Using the *Axiom of Choice* we get hold of a choice function f such that

$$\forall i > 0.\, f(i) \geq i \wedge (q_{f(i)}, q_{f(i)+1}) \in S, \qquad (\dagger)$$

i.e., $f(i)$ produces some index of an S-step after position i in $\vec{q}$. Using f we define a new sequence $[\cdot]$ of indices inductively

$$[i] = \begin{cases} i & \text{if } i = 1, \\ f([i-1]) + 1 & \text{otherwise.} \end{cases}$$

With (†) we have $f(i) \geq i$ and $(q_{f(i)}, q_{f(i)+1}) \in S$ for all $i > 0$. Since $f(i) \geq i$ there is an $N \cup S$ sequence from every q_i to the corresponding $q_{f(i)}$. Thus we obtain $(q_i, q_{f(i)+1}) \in S^+$ for all $i > 0$ using (b). This immediately implies $(q_{[i]}, q_{[i+1]}) \in S^+$ for all $i > 0$ and thereby $\neg\mathsf{SN}_{S^+}(q_{[1]})$ which is equivalent to $\neg\mathsf{SN}_S(q_{[1]})$ by Lem. 3.3. But $q_{[1]} = q_1$ and thus $\neg\mathsf{SN}_S(q_1)$. Together with (c), this provides the desired contradiction. ∎

Lemma 3.5. $\mathsf{SN}_{\mathcal{R}}(t) \Longrightarrow \mathsf{SN}_{(\rhd \cup \to_{\mathcal{R}})}(t)$.

Proof. Assume that t is not terminating w.r.t. $(\rhd \cup \to_{\mathcal{R}})$. Hence, we obtain the infinite sequence $t = t_1\ (\rhd \cup \to_{\mathcal{R}})\ t_2\ (\rhd \cup \to_{\mathcal{R}})\ t_3\ (\rhd \cup \to_{\mathcal{R}})\ \cdots$. From the assumption we have $\mathsf{SN}_{\mathcal{R}}(t_1)$ and by Lem. 2.1 and Lem. 2.2(c) we obtain $\mathsf{SN}_{\mathcal{R}}(t_i)$ for all $i > 0$. Thus, $t_i \xrightarrow{\rhd}_{\mathsf{SN}(\mathcal{R})} t_{i+1}$ for all i and since $\mathsf{SN}(\xrightarrow{\rhd}_{\mathsf{SN}(\mathcal{R})})$ by Lem. 2.2(b) we arrive at a contradiction. ∎

Proof of Theorem 3.1. In order to show that finiteness of $(\mathcal{P} \setminus \rhd_\pi, \mathcal{R})$ implies finiteness of $(\mathcal{P}, \mathcal{R})$ we prove its contraposition. Hence, we may assume (in addition to the premises of Theorem 3.1) that there is a minimal infinite $(\mathcal{P}, \mathcal{R})$-chain and have to transform it into a minimal infinite $(\mathcal{P} \setminus \rhd_\pi, \mathcal{R})$-chain. Thus we may assume that for all $i > 0$:

(a) $s_i \to t_i \in \mathcal{P}$,
(b) $t_i\sigma_i \to_{\mathcal{R}}^* s_{i+1}\sigma_{i+1}$, and
(c) $\mathsf{SN}_{\mathcal{R}}(t_i\sigma_i)$.

We start by a case distinction on $\exists j > 0.\, \forall i \geq j.\, (s_i, t_i) \in (\mathcal{P} \setminus \rhd_\pi)$. If there is such a j, we can combine this with (b) and obtain the desired minimal $(\mathcal{P} \setminus \rhd_\pi, \mathcal{R})$-chain by shifting the original chain j positions to the left. Hence, consider the second case and assume $\forall i > 0.\, \exists j \geq i.\, (s_j, t_j) \notin (\mathcal{P} \setminus \rhd_\pi)$. With (a) and the preconditions of the subterm criterion processor this results in

$$\forall i > 0.\, \exists j \geq i.\, \pi(s_j)\sigma_j \rhd \pi(t_j)\sigma_j. \tag{3.1}$$

From this point on, the proof mainly runs by instantiating the relations N and S of Lem. 3.4 appropriately and showing the assumptions Lem. 3.4(a)–Lem. 3.4(c) in turn. For N we use the reflexive and transitive closure of the rewrite relation, i.e., $\to_{\mathcal{R}}^*$. For S we use $(\rhd \cup \to_{\mathcal{R}})^+$. Finally, we use the infinite sequence q defined by $q_i = \pi(s_{i+1})\sigma_{i+1}$ (the index shift is needed to establish termination of q_1 later on). From (a) and Thm. 3.1(a), together with Lem. 2.1(a) we get

$$\pi(s_i)\sigma_i \trianglerighteq \pi(t_i)\sigma_i. \tag{3.2}$$

Furthermore, we obtain

$$\pi(t_i)\sigma_i \to_{\mathcal{R}}^* \pi(s_{i+1})\sigma_{i+1}, \tag{3.3}$$

since the roots of t_i and s_{i+1} are guaranteed to be non-constant symbols and the root of t_i is not a defined symbol by Thm. 3.1(b). In combination we get $\pi(s_i)\sigma_i \trianglerighteq \circ \to_{\mathcal{R}}^* \pi(s_{i+1})\sigma_{i+1}$ and in turn $(q_i, q_{i+1}) \in N \cup S$, thereby discharging assumption Lem. 3.4(a). For our specific relations assumption Lem. 3.4(b) trivially holds. This leaves us with showing termination of q_1 with respect to the relation $(\rhd \cup \to_{\mathcal{R}})^+$. From the minimality of the initial chain (c) we know $\mathsf{SN}_{\mathcal{R}}(t_1\sigma_1)$ and by Lem. 2.1 we get $\mathsf{SN}_{\mathcal{R}}(\pi(s_2)\sigma_2)$ and thus $\mathsf{SN}_{\mathcal{R}}(q_1)$. By Lemmas 3.5 and 3.3 we then achieve $\mathsf{SN}_{(\rhd \cup \to_{\mathcal{R}})^+}(q_1)$. At this point (by Lem. 3.4) we get grip of some $j > 0$ such that

$$\forall i \geq j.\, (q_i, q_{i+1}) \in N \setminus S. \tag{3.4}$$

Now we proof $\forall i \geq j.\, \pi(s_{i+1})\sigma_{i+1} = \pi(t_{i+1})\sigma_{i+1}$ as follows. Assume $i \geq j$ and $\pi(s_{i+1})\sigma_{i+1} \neq \pi(t_{i+1})\sigma_{i+1}$. Then with (3.2) we get $\pi(s_{i+1})\sigma_{i+1} \rhd \pi(t_{i+1})\sigma_{i+1}$. By (3.3), this results in

$\pi(s_{i+1})\sigma_{i+1} \rhd \circ \to_{\mathcal{R}}^* \pi(s_{i+2})\sigma_{i+2}$ and consequently in $(q_i, q_{i+1}) \in S$ (contradicting 3.4). Thus $\forall i \geq j.\, q_i = \pi(t_{i+1})\sigma_{i+1}$. However, this contradicts (3.1). ∎

4. Usable Rules

One important technique to prove termination within the DP framework is the reduction pair processor. A *reduction pair* $(\succ, \succsim)$ consists of a well-founded and stable relation $\succ$ in combination with a monotone and stable relation $\succsim$. Further, $\succsim$ has to be compatible with $\succ$, i.e., $\succsim \circ \succ \subseteq \succ$. Note that it is not required that $\succ$ and $\succsim$ are partial orders [23]. Examples for reduction pairs are polynomial orders [15, 19, 20], matrix orders [7, 17], and the lexicographic path order (LPO) [16]. (There are several other classes of reduction pairs. We listed those which have been formalized in IsaFoR.)

The basic version of the reduction pair processor [12, 14] requires that all rules of $\mathcal{R}$ are weakly decreasing w.r.t. $\succsim$ (then $\to_{\mathcal{R}} \subseteq \succsim$) and all pairs of $\mathcal{P}$ are weakly or strictly decreasing. From ($\star$) on page 3 it is easy to see that this implies that every reduction in a $(\mathcal{P}, \mathcal{R})$-chain corresponds to a weak or strict decrease. Thus, the strictly decreasing pairs cannot occur infinitely often and can be removed from $\mathcal{P}$. This technique is already present in IsaFoR and its formalization is described in [23].

Theorem 4.1. *Finiteness of* $(\mathcal{P} \setminus \succ, \mathcal{R})$ *implies finiteness of* $(\mathcal{P}, \mathcal{R})$*, provided:*

(a) $(\succ, \succsim)$ *is a reduction pair,*
(b) $\mathcal{P} \subseteq \succ \cup \succsim$,
(c) $\mathcal{R} \subseteq \succsim$.

Starting with [24], there have been several papers [10, 12, 14] on how to improve the last requirement. Therefore, $\mathcal{R}$ in (c) is replaced by the *usable rules*.

The main idea of the usable rules is easy to explain: since in chains rewriting is only performed with instances of rhss of $\mathcal{P}$, it should suffice to rewrite with rules of defined symbols that occur in rhss of $\mathcal{P}$. If these usable rules introduce new defined symbols then the rules defining them also have to be considered as usable. Hence, in the remaining DP problem $(\{(2.10)\text{–}(2.12)\}, \mathcal{R})$ of Ex. 2.3 only rules (2.1)–(2.3) and (2.4)–(2.8) are usable. This idea is formally expressed in the following definition.

Definition 4.2 (Usable Rules)**.** The function $\mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(t)$ defines whether a term t is closed under usable rules $\mathcal{U}$ w.r.t. some TRS $\mathcal{R}$.

$$\mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(x) = \mathsf{true},$$
$$\mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(f(\vec{t_n})) = \bigwedge_{1 \leqslant i \leqslant n} \mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(t_i) \wedge \bigwedge_{\ell \to r \in \mathcal{R}} (\mathsf{root}(\ell) = f \Longrightarrow \ell \to r \in \mathcal{U}).$$

A TRS $\mathcal{Q}$ is closed under usable rules whenever all rhss are closed under usable rules, i.e.,

$$\mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(\mathcal{Q}) = \bigwedge_{\ell \to r \in \mathcal{Q}} \mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(r).$$

A DP problem $(\mathcal{P}, \mathcal{R})$ is closed under usable rules iff $\mathcal{P}$ and $\mathcal{U}$ are closed under usable rules.

$$\mathsf{urClosed}_{\mathcal{U}}(\mathcal{P}, \mathcal{R}) = \mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(\mathcal{P}) \wedge \mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(\mathcal{U}).$$

Finally, the usable rules of DP problem $(\mathcal{P}, \mathcal{R})$ are the least set $\mathcal{U}$ satisfying $\mathsf{urClosed}_{\mathcal{U}}(\mathcal{P}, \mathcal{R})$.

Note that there are several other equivalent definitions of usable rules, e.g., one can find definitions of usable rules via so called usable symbols (i.e., root symbols of lhss of usable rules). However, there are also two improvements that yield smaller sets.[2]

The first improvement is to take the reduction pair into account. If certain positions of terms are disregarded then their usable rules do not have to be considered. For example, usually due to the rhs $\mathsf{if2}^\sharp(\mathsf{0} < x, x, z)$ of DP (2.10) all $<$-rules are usable. However, if we would use a reduction pair based on polynomial orders, where $\mathcal{P}ol(\mathsf{if2}^\sharp(b, x, z)) = z$ then the call to $<$ is ignored by the order. This can be exploited by excluding the $<$-rules from the set of usable rules. This improvement is called *usable rules w.r.t. an argument filter* [12]. Here, argument filters are used to describe which positions in a term are relevant: an argument filter π with $\pi(f) = \{i_1, \ldots, i_k\}$ indicates that in a term $f(\vec{t}_n)$ only arguments $t_{i_1}, \ldots, t_{i_k}$ are regarded. This is formalized by the notion of π-compatibility.

Definition 4.3 (π-Compatibility)**.** A relation $\succsim$ is *π-compatible* iff for all n-ary symbols f, all i with $1 \leqslant i \leqslant n$, all $t_1, \ldots, t_n$, and all s and s':

$$i \notin \pi(f) \Longrightarrow f(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_n) \succsim f(t_1, \ldots, t_{i-1}, s', t_{i+1}, \ldots, t_n).$$

To formally define the usable rules w.r.t. an argument filter, a minor modification of Def. 4.2 suffices. Just

$$\text{replace} \quad \bigwedge_{1 \leqslant i \leqslant n} \mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(t_i) \quad \text{by} \quad \bigwedge_{i \in \pi(f)} \mathsf{urClosed}_{\mathcal{U},\mathcal{R}}(t_i).$$

The second improvement to reduce the set of usable rules is performed by taking the structure of terms into account. Observe that in the rhs $\mathsf{if2}^\sharp(\mathsf{0} < x, x, z)$ of DP (2.10), the first argument of $<$ is $\mathsf{0}$. Hence, only the rules (2.1) and (2.2) are possibly applicable, but not the remaining Rule (2.3). This is not captured by Def. 4.2. As there, all f-rules have to be usable whenever the symbol f occurs. On the contrary, in [10], an improved version of usable rules is described which can figure out that Rule (2.3) is not applicable. To this end, the condition $\mathsf{root}(\ell) = f$ in Def. 4.2 is replaced by a condition based on unification. Demanding $\ell \approx f(\vec{t}_n)$ would be unsound. First $f(\vec{t}_n)$ has to be preprocessed by the function tcap of [10]. This function keeps only those parts of the input term which cannot be reduced, even if the term is instantiated. All other parts are replaced by fresh variables.

Definition 4.4. Let $\mathcal{R}$ be a TRS.[3]

$$\mathsf{tcap}(t) = \begin{cases} f(\mathsf{tcap}(\vec{t}_n)) & \text{if } t = f(\vec{t}_n) \text{ and } \ell \not\approx f(\mathsf{tcap}(\vec{t}_n)) \text{ for all lhss } \ell \text{ of } \mathcal{R}, \\ \text{a fresh variable} & \text{otherwise.} \end{cases}$$

Here, $\mathsf{tcap}(\vec{t}_n)$ is the list of terms where tcap is applied to all arguments of $\vec{t}_n$.

[2]There also is another extension of usable rules, the *generalized usable rules*. It allows a variant of the reduction pair processor where reduction pairs with non-monotone $\succsim$ may be used, cf. [8, Thm. 10]. However, that reduction pair processor is incomparable to both Thm. 4.1 and the upcoming Thm. 4.6. It may be interesting to formalize the soundness of that processor, too, but that would be a different proof.

[3]Note that in IsaFoR we do not use tcap, but the more efficient and equivalent version etcap which is based on ground-contexts. Moreover, unification is replaced by ground-context matching [23, Section 4.2]. But as the notions of tcap and unification are more common, in the following we stick to these two notions.

Now the second improvement can be defined formally. Again, it is a minor but crucial modification of Def. 4.2. It suffices to

$$\text{replace} \bigwedge_{\ell\to r\in\mathcal{R}} (\mathsf{root}(\ell) = f \implies \ell \to r \in \mathcal{U}) \quad \text{by} \quad \bigwedge_{\ell\to r\in\mathcal{R}} (\ell \approx f(\mathsf{tcap}(\vec{t}_n)) \implies \ell \to r \in \mathcal{U}).$$

Hence, incorporating both improvements results in the following definition which now contains a π in the superscript to distinguish it from Def. 4.2.

Definition 4.5 (Improved Closure Under Usable Rules)**.**

$$\begin{aligned}
\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(x) &= \mathsf{true},\\
\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(f(\vec{t}_n)) &= \bigwedge_{i\in\pi(f)} \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(t_i) \wedge \bigwedge_{\ell\to r\in\mathcal{R}} (\ell \approx f(\mathsf{tcap}(\vec{t}_n)) \Longrightarrow \ell \to r \in \mathcal{U}),\\
\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{Q}) &= \bigwedge_{\ell\to r\in\mathcal{Q}} \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(r),\\
\mathsf{urClosed}^{\pi}_{\mathcal{U}}(\mathcal{P},\mathcal{R}) &= \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{P}) \wedge \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{U}).
\end{aligned}$$

Note that we do not define *the* improved usable rules of a DP problem $(\mathcal{P},\mathcal{R})$ (as we would, by demanding that $\mathcal{U}$ is the least set satisfying $\mathsf{urClosed}^{\pi}_{\mathcal{U}}(\mathcal{P},\mathcal{R})$). Hence, every set $\mathcal{U}$ that satisfies the closure properties can be used later on. It is easy to see, that the usable rules w.r.t. Def. 4.2 satisfy the closure properties as well as a version of usable rules which only incorporates one of the two improvements. Thus, by this definition we gain the advantage that we can handle several variants of usable rules.

Having defined all necessary notions, we are ready to present the improved reduction pair processor with usable rules where the second part also incorporates [9, Theorem 28] which allows to delete rules by a syntactic criterion.

Theorem 4.6 (Reduction Pair Processors with Usable Rules)**.** *Let* c *be some binary symbol, let* $\mathcal{C}_\varepsilon = \{\mathsf{c}(x,y) \to x, \mathsf{c}(x,y) \to y\}$,[4] *and let* $\mathcal{U}$ *be some TRS (called the usable rules). For every signature* $\mathcal{F}$ *and TRS* $\mathcal{R}$*, let* $\mathcal{R}_{\neg\mathcal{F}} = \{\ell \to r \in \mathcal{R} \mid \ell \notin \mathcal{T}(\mathcal{F},\mathcal{V})\}$*. Finiteness of* $(\mathcal{P} \setminus \succ, \mathcal{R})$ *implies finiteness of* $(\mathcal{P},\mathcal{R})$*, provided:*

- (a) $(\succ, \succsim)$ *is a reduction pair,*
- (b) $\mathcal{P} \subseteq \succ \cup \succsim$*,*
- (c) $\mathcal{U} \cup \mathcal{C}_\varepsilon \subseteq \succsim$*,*
- (d) $\succsim$ *is* π*-compatible,*
- (e) $\mathsf{urClosed}^{\pi}_{\mathcal{U}}(\mathcal{P},\mathcal{R})$*.*

If additionally $\mathcal{C}_\varepsilon \subseteq \succ$*,* $\succ$ *is monotone,* $\mathcal{U} \subseteq \mathcal{R}$*, and* $\mathcal{F}$ *is the set of all symbols occurring in rhss of* $\mathcal{P}\cup\mathcal{U}$*, then* $\mathcal{R}$ *can be replaced by* $\mathcal{U}$ *without any strictly decreasing rules and one can remove all rules which contain symbols of* $\mathcal{F}$ *in their left-hand side. Formally, finiteness of* $(\mathcal{P}_{\neg\mathcal{F}} \setminus \succ, \mathcal{U}_{\neg\mathcal{F}} \setminus \succ)$ *implies finiteness of* $(\mathcal{P},\mathcal{R})$*.*

[4] The real definition of $\mathcal{C}_\varepsilon$ in IsaFoR is slightly different due to technical reasons. Since there is no restriction on the type of variables, there might be only one variable. To this end x and y are replaced by all possible terms. And as the type of function symbols is also unrestricted there might be no fresh symbol c. However, in IsaFoR the signature is implicit where every arity is allowed. For example, the term $\mathsf{c}(\mathsf{c},\mathsf{c})$ contains one symbol $(\mathsf{c},1)$ and two symbols $(\mathsf{c},0)$ where $(\mathsf{c},1) \neq (\mathsf{c},0)$. In this way, we can always get a fresh symbol (c,n) where n is larger than all arities that occur in $\mathcal{R}$ and we use a constant $(\mathsf{d},0)$ to obtain this high arity. Hence, we define $\mathcal{C}_\varepsilon = \bigcup_{s,t}\{\mathsf{c}(s,t,\mathsf{d},\dots,\mathsf{d}) \to s, \mathsf{c}(s,t,\mathsf{d},\dots,\mathsf{d}) \to t\}$.

Regarding requirement (c), observe that most reduction pairs that are currently used in automated termination tools do satisfy $\mathcal{C}_\varepsilon \subseteq \succsim$. Therefore, requirement (c) of Thm. 4.1 can usually be replaced by $\mathcal{U} \subseteq \succsim$. Requirement (d) is easy to satisfy by choosing an appropriate argument filter π which depends on the reduction pair. And if $\mathcal{U}$ is chosen as the usable rules w.r.t. any known definition of usable rules, then condition (e) is satisfied. Thus, for most reduction pairs one has only replaced requirement $\mathcal{R} \subseteq \succsim$ of Thm. 4.1 by the weaker condition $\mathcal{U} \subseteq \succsim$ in Thm. 4.6.

Example 4.7. To solve the remaining DP problem $(\{(2.10)\text{–}(2.12)\}, \mathcal{R})$ of Ex. 3.2 we use a polynomial order [19] where $\mathcal{P}ol(\mathsf{set}^\sharp(x)) = \mathcal{P}ol(\mathsf{del}(y,x)) = x$, $\mathcal{P}ol(x\!:\!y) = \mathcal{P}ol(\mathsf{if}(\ldots,y)) = y{+}1$, $\mathcal{P}ol(\mathsf{if2}^\sharp(x,y,z)) = x{+}z$, $\mathcal{P}ol(x < y) = \mathcal{P}ol(\bot) = \mathcal{P}ol(\top) = 0$, and $\mathcal{P}ol(\mathsf{c}(x,y)) = x{+}y$. A corresponding compatible argument filter π is defined by $\pi(\mathsf{set}^\sharp) = \{1\}$, $\pi(\mathsf{del}) = \pi(:) = \{2\}$, $\pi(\mathsf{if}) = \{5\}$, $\pi(\mathsf{if2}^\sharp) = \{1,3\}$, $\pi(<) = \pi(\bot) = \pi(\top) = \emptyset$, and $\pi(\mathsf{c}) = \{1,2\}$. For this argument filter, the minimal set of usable rules is $\mathcal{U} = \{(2.1), (2.2), (2.4)\text{–}(2.8)\}$. Note that it would also be accepted if, e.g., Rule (2.3) would be added.

Then all conditions of Thm. 4.6 are satisfied and one can remove Pair (2.10) as it is strictly decreasing. The remaining DP problem $(\{(2.11),(2.12)\}, \mathcal{R})$ is then easily solved by another application of the reduction pair processor where one chooses $\mathcal{P}ol(\mathsf{set}^\sharp(x)) = 0$, $\mathcal{P}ol(\mathsf{if2}^\sharp(x,y,z)) = 1$, $\mathcal{P}ol(\mathsf{c}(x,y)) = x + y$, and where $\mathcal{U} = \emptyset$.

In theory `UsableRules` we have proven Thm. 4.6. Although a similar proof has already been performed by the authors of [10] and [12]—on paper, not formalized—this proof has never been published in some reviewed article, it is only available in [22]. Moreover, our proof is not just a formalization of the proof in [22], but there are some essential differences which are pointed out in the following.

The standard proof of Thm. 4.6 is by transforming a minimal chain $t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2 \to_{\mathcal{P}} t_2\sigma_2 \ldots$ into a chain over *filtered* terms $\pi(t_1)\delta_1 \to^*_{\pi(\mathcal{U})\cup\mathcal{C}_\varepsilon} \pi(s_2)\delta_2 \to_{\pi(\mathcal{P})} \pi(t_2)\delta_2 \ldots$ and then uses the preconditions of the theorem to show that certain pairs of $\pi(\mathcal{P})$ (and therefore also pairs of $\mathcal{P}$) cannot occur infinitely often. Here, one uses a transformation $\mathcal{I}_\pi$ which transforms σ_i into δ_i. For the second part of the theorem where $\succ$ is monotone, one requires another transformation $\mathcal{I}$ which does not apply any argument filter. Hence, there are two transformations $\mathcal{I}$ and $\mathcal{I}_\pi$ where for both transformations similar results are shown.

In our formalization we were able to simplify the proofs considerably by not constructing filtered terms. Moreover, we use the same transformation $\mathcal{I}$ for both parts of the theorem.

A problem in the standard proof of the reduction pair processor with usable rules is the implicit assumption that the TRS $\mathcal{R}$ meets the variable condition, i.e., $\mathcal{V}(\ell) \supseteq \mathcal{V}(r)$ and $\ell \notin \mathcal{V}$ for all rules $\ell \to r \in \mathcal{R}$. Although in practice this condition is nearly always satisfied, we have to deal with this assumption, where there are three alternatives. First, one can check that the TRS $\mathcal{R}$ meets the variable condition whenever the theorem is applied on some concrete DP problem $(\mathcal{P}, \mathcal{R})$. This would clearly increase the runtime for certifying a given proof. Second, one can define finiteness of DP problems or soundness of processors in a way that it incorporates the variable condition. However, this will make the development of other processors more complicated which do not care about the variable condition. And third, one can try to prove Thm. 4.6 without assuming the variable condition. This is the alternative we have finally formalized and which does not appear in the literature so far.

For the upcoming formal definition of $\mathcal{I}$ we essentially use a combination of the definitions of [9] and [10] where $x \# xs$ denotes the Isabelle list with head x and tail xs.

Definition 4.8. Let $\mathcal{R}$ and $\mathcal{U}$ be two TRSs, let $\mathcal{F}$ be some signature, and let c be the binary symbol of $\mathcal{C}_\varepsilon$. We define $\mathcal{I}$ as a function from terms to terms as follows:

$$\begin{aligned}
\mathsf{comb}([t]) &= t,\\
\mathsf{comb}(t \mathbin{\#} s \mathbin{\#} ts) &= \mathsf{c}(t, \mathsf{comb}(s \mathbin{\#} ts)),\\
\mathsf{rewrite}(\mathcal{R}, t) &= \{C[r\sigma]_p \mid \ell \to r \in \mathcal{R}, t = C[u], \mathsf{match}(u, \ell) = \sigma\},\\
\mathcal{I}(x) &= x,\\
\mathcal{I}(f(\vec{t_n})) &= \begin{cases} f(\vec{t_n}) & \text{if } \neg\mathsf{SN}_\mathcal{R}(f(\vec{t_n})),\\ f(\mathcal{I}(\vec{t_n})) & \text{if } \mathsf{SN}_\mathcal{R}(f(\vec{t_n})),\ f \in \mathcal{F}, \text{ and } \forall \ell \to r \in \mathcal{R} \setminus \mathcal{U}.\ell \not\approx f(\mathsf{tcap}(\vec{t_n})),\\ \mathsf{comb}(f(\mathcal{I}(\vec{t_n})) \mathbin{\#} \mathcal{I}(\mathsf{rewrite}(\mathcal{R}, f(\vec{t_n})))) & \text{otherwise.}\end{cases}
\end{aligned}$$

$\mathcal{I}$ and tcap are homeomorphically extended to operate on lists, i.e., $\mathcal{I}(\vec{t_n}) = (\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$.

The function comb just combines a non-empty list of terms into one term. It is easy to prove that one can access all terms in the list by rewriting with $\mathcal{C}_\varepsilon$: $\mathsf{comb}([\ldots, t_i, \ldots]) \to^*_{\mathcal{C}_\varepsilon} t_i$.

The function $\mathsf{rewrite}$ computes the list of one-step reducts of a given term by using a sound and complete matching algorithm match. The major difference between $\{s \mid t \to_\mathcal{R} s\}$ and $\mathsf{rewrite}(\mathcal{R}, t)$ is that the latter instantiates a rule by the matcher of the lhs and the corresponding redex (as usual), but it never instantiates variables which only occur in the rhs of the rule. For example, if $\mathcal{R} = \{\mathsf{a} \to x\}$ then $\{s \mid \mathsf{a} \to_\mathcal{R} s\}$ is the set of all terms, whereas $\mathsf{rewrite}(\mathcal{R}, \mathsf{a}) = [x]$. Hence, $\mathsf{rewrite}$ is sound ($\mathsf{rewrite}(\mathcal{R}, t) \subseteq \{s \mid t \to_\mathcal{R} s\}$) but in general not complete. However, under one condition completeness is achieved: whenever $t \to_\mathcal{R} s$ by a reduction with a rule that satisfies the variable condition, then $s \in \mathsf{rewrite}(\mathcal{R}, t)$.

The main reason for introducing $\mathsf{rewrite}$ is that without the assumption of the variable condition on $\mathcal{R}$ the set $\{s \mid t \to_\mathcal{R} s\}$ may be infinite. Then the definition of $\mathcal{I}$ as in [10]—where $\{\mathcal{I}(s) \mid f(\vec{t_n}) \to_\mathcal{R} s\}$ is used instead of $\mathcal{I}(\mathsf{rewrite}(\mathcal{R}, f(\vec{t_n})))$—does not work in combination with comb, as comb expects a list (or a *finite* set) as input. Also note that this input must be finite as one finally wants to obtain a single term containing all input terms.

The first case of $\mathcal{I}(f(\vec{t_n}))$ is mainly a technicality. Usually, $\mathcal{I}$ is only defined on terminating terms. To make $\mathcal{I}$ a total function on all terms we inserted the case distinction on $\mathsf{SN}_\mathcal{R}(f(\vec{t_n}))$. Termination of $\mathcal{I}$ is then proven using well-founded induction on $\overset{\triangleright}{\to}_{\mathsf{SN}(\mathcal{R})}$ where in this proof the soundness result for $\mathsf{rewrite}$ is crucial.

The transformation $\mathcal{I}$ is constructed in such a way that for every reduction $t \to_\mathcal{R} s$ one obtains a weak decrease, provided that the usable rules and $\mathcal{C}_\varepsilon$ are weakly decreasing. Therefore, in the definition of $\mathcal{I}$ there are essentially two cases for a term $f(\vec{t_n})$. If only usable rules can be used to reduce $f(\vec{t_n})$ at the root position, then $\mathcal{I}(f(\vec{t_n}))$ is obtained by applying $\mathcal{I}$ on the arguments, resulting in $f(\mathcal{I}(\vec{t_n}))$. The corresponding reduction will then result in a weak decrease as one can also perform the reduction with the usable rules on the transformed term. The condition that only usable rules are applicable is ensured by demanding that no lhs of a non-usable rule in $\mathcal{R} \setminus \mathcal{U}$ can be unified with $f(\mathsf{tcap}(\vec{t_n}))$.

Otherwise, all rules may have been used to rewrite $f(\vec{t_n})$. Then, in addition to $f(\mathcal{I}(\vec{t_n}))$ we have to store all one-step reducts of t. This is done by encoding them in a single term using comb. Now every possible reduct can be accessed using $\mathcal{C}_\varepsilon$. And since $\mathcal{C}_\varepsilon$ is weakly decreasing we obtain a weak decrease. This is proven formally in the upcoming lemma.

Lemma 4.9 (Properties of $\mathcal{I}$)**.** *Let $(\succsim, \succ)$ be a reduction pair, let $\succsim$ be π-compatible, let $\mathcal{U} \cup \mathcal{C}_\varepsilon \subseteq \succsim$, let $\mathsf{urClosed}^\pi_{\mathcal{U},\mathcal{R}}(\mathcal{U})$, and let the rhss of $\mathcal{U}$ be terms within $\mathcal{T}(\mathcal{F},\mathcal{V})$. Let $\mathsf{SN}_\mathcal{R}(t)$, $\mathsf{SN}_\mathcal{R}(t\sigma)$, and $\mathsf{SN}_\mathcal{R}(f(\vec{t}_n))$.*

(i) *If $\mathsf{urClosed}^\pi_{\mathcal{U},\mathcal{R}}(t)$ and $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ then $t\mathcal{I}(\sigma) \succsim^* \mathcal{I}(t\sigma)$.*
(ii) *$\mathcal{I}(t\sigma) \to^*_{\mathcal{C}_\varepsilon} t\mathcal{I}(\sigma)$. And if $t \notin \mathcal{T}(\mathcal{F},\mathcal{V})$ then $\mathcal{I}(t\sigma) \to^+_{\mathcal{C}_\varepsilon} t\mathcal{I}(\sigma)$.*
(iii) *If $f(\vec{t}_n) \to_{\mathcal{R},\epsilon} s$ using a rule $\ell \to r$ and $\mathcal{I}(f(\vec{t}_n)) = f(\mathcal{I}(\vec{t}_n))$ then $\ell \to r \in \mathcal{U}$, $\mathcal{I}(f(\vec{t}_n)) \succsim^* \circ \to_\mathcal{U} \circ \succsim^* \mathcal{I}(s)$. And $\mathcal{I}(f(\vec{t}_n)) \to^+_{\mathcal{C}_\varepsilon} \circ \to_\mathcal{U} \circ \succsim^* \mathcal{I}(s)$ if $\ell \notin \mathcal{T}(\mathcal{F},\mathcal{V})$.*
(iv) *If $f(\vec{t}_n) \to_\mathcal{R} s$ and $\mathcal{I}(f(\vec{t}_n)) = \mathsf{comb}(\dots)$ then $\mathcal{I}(f(\vec{t}_n)) \to^+_{\mathcal{C}_\varepsilon} \mathcal{I}(s)$.*
(v) *If $t \to_\mathcal{R} s$ then $\mathcal{I}(t) \succsim^* \mathcal{I}(s)$. Moreover, $t \to_{\mathcal{U}_{\neg\mathcal{F}}} s$ or $\mathcal{I}(t) \to^+_{\mathcal{C}_\varepsilon} \circ \succsim^* \mathcal{I}(s)$.*
(vi) *If $t \to^*_\mathcal{R} s$ then $\mathcal{I}(t) \succsim^* \mathcal{I}(s)$. Moreover, $\mathcal{I}(t) \succsim^* \circ \to_{\mathcal{C}_\varepsilon} \circ \succsim^* \mathcal{I}(s)$ or $t \to^*_{\mathcal{U}_{\neg\mathcal{F}}} s$.*

Here, $\mathcal{I}$ is homeomorphically extended to substitutions, i.e., $\mathcal{I}(\sigma)(x) = \mathcal{I}(\sigma(x))$.

In the following proof sketch of Lem. 4.9, all essential points are included, especially those where we deviate from the standard proofs.

Proof. The proof of (vi) is a straight-forward induction on the reduction length using (v).

We prove (v), by induction over t. First note that t is not a variable. Otherwise, there would be some $x \to r \in \mathcal{R}$, contradicting $\mathsf{SN}_\mathcal{R}(t)$. Hence the base case is trivial. In the step-case, we make a case distinction on how $t = f(\vec{t}_n)$ is transformed. The case $\mathcal{I}(t) = \mathsf{comb}(\dots)$ is solved by (iv). Otherwise, $\mathcal{I}(t) = f(\mathcal{I}(\vec{t}_n))$. For a root reduction we use (iii). Otherwise, $s = f(t_1,\dots,s_i,\dots,t_n)$ and $t_i \to_\mathcal{R} s_i$. Applying the induction hypothesis is easy, but some additional effort is required to prove $\mathcal{I}(s) = f(\mathcal{I}(t_1),\dots,\mathcal{I}(s_i),\dots,\mathcal{I}(t_n))$.

Proving (iv), essentially requires completeness of $\mathsf{rewrite}$. First we prove that if $f(\vec{t}_n) \to_\mathcal{R} s$ then the employed rule $\ell \to r$ must satisfy $\mathcal{V}(\ell) \supseteq \mathcal{V}(r)$, as otherwise $\mathsf{SN}_\mathcal{R}(f(\vec{t}_n))$ would not hold. Under this condition, completeness of $\mathsf{rewrite}$ states that $s \in \mathsf{rewrite}(\mathcal{R}, f(\vec{t}_n))$. The remaining proof of (iv) can be done by simple inductions using the definitions of comb and $\mathcal{C}_\varepsilon$.

To prove (iii), we first show that $\mathcal{I}(f(\vec{t}_n)) = f(\mathcal{I}(\vec{t}_n))$ ensures that the employed rule $\ell \to r$ is usable. The reason is that $f(\vec{t}_n) = \ell\sigma$ implies $f(\mathsf{tcap}(\vec{t}_n)) \approx \ell$ and hence, $\ell \to r \notin \mathcal{R} \setminus \mathcal{U}$ by the definition of $\mathcal{I}$. By the requirement on the rhss of $\mathcal{U}$ we know that $r \in \mathcal{T}(\mathcal{F},\mathcal{V})$. Hence, we can build the following steps using (ii) and (i) in combination with $\mathsf{urClosed}^\pi_{\mathcal{U},\mathcal{R}}(\mathcal{U})$: $\mathcal{I}(f(\vec{t}_n)) = \mathcal{I}(\ell\sigma) \succsim^* \ell\mathcal{I}(\sigma) \to_\mathcal{U} r\mathcal{I}(\sigma) \succsim^* \mathcal{I}(r\sigma) = \mathcal{I}(s)$. And if $\ell \notin \mathcal{T}(\mathcal{F},\mathcal{V})$ then we additionally get $\mathcal{I}(\ell\sigma) \to^+_{\mathcal{C}_\varepsilon} \ell\mathcal{I}(\sigma)$ by (ii).

Proving (ii), is a straight-forward induction on t.

And finally, for (i), we also use induction on t. In the step-case we first prove that $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ in combination with $\mathsf{urClosed}^\pi_{\mathcal{U},\mathcal{R}}(f(\vec{t}_n))$ implies $f(\mathcal{I}(\vec{t}_n\sigma)) = \mathcal{I}(f(\vec{t}_n\sigma))$. Then, for all argument positions $i \in \pi(f)$, we apply the induction hypothesis to obtain $t_i\mathcal{I}(\sigma) \succsim^* \mathcal{I}(t_i\sigma)$. Then, by monotonicity of $\succsim$, we obtain $f(\dots,t_i\mathcal{I}(\sigma),\dots) \succsim^* f(\dots,\mathcal{I}(t_i\sigma),\dots)$. For all other positions, π-compatibility of $\succsim$ provides the same inequality. ∎

With the help of Lem. 4.9 it is now possible to prove the main result of this section.

Proof of Thm. 4.6. Assume that there is a minimal infinite chain where $s_i\sigma_i \to_\mathcal{P} t_i\sigma_i \to^*_\mathcal{R} s_{i+1}\sigma_{i+1}$ and $\mathsf{SN}_\mathcal{R}(t_i\sigma_i)$ for all i. Let $\mathcal{F}$ be the set of all symbols that occur in rhss of $\mathcal{P} \cup \mathcal{U}$. Then by the conditions of the theorem and by using Lem. 4.9 (i), (vi), and (ii), for all i we conclude

$$s_i\mathcal{I}(\sigma_i) \to_\mathcal{P} t_i\mathcal{I}(\sigma_i) \succsim^* \mathcal{I}(t_i\sigma_i) \succsim^* \mathcal{I}(s_{i+1}\sigma_{i+1}) \succsim^* s_{i+1}\mathcal{I}(\sigma_{i+1}). \qquad (\ddagger)$$

By using $\mathcal{P} \subseteq \succ \cup \succsim$ we obtain a strict or weak decrease between every two terms $s_i\mathcal{I}(\sigma_i)$ and $s_{i+1}\mathcal{I}(\sigma_{i+1})$. Thus, by Lem. 3.4, the strictly decreasing pairs can only occur finitely often. This shows that there must be some n such that for all i, $s_{i+n} \to t_{i+n} \in \mathcal{P} \setminus \succ$. Hence, there is an infinite minimal $(\mathcal{P} \setminus \succ, \mathcal{R})$-chain.

If additionally, $\mathcal{C}_\varepsilon \subseteq \succ$ and $\succ$ is monotone, we first prove that there is some n with $t_{n+i}\sigma_{n+i} \to^*_{\mathcal{U}_{\neg\mathcal{F}}} s_{n+i+1}\sigma_{n+i+1}$ and $s_{n+i} \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ for all i. If this would not be the case, then infinitely often $t_i\sigma \to^*_{\mathcal{U}_{\neg\mathcal{F}}} s_{i+1}\sigma_{i+1}$ does not hold or infinitely often $s_i \notin \mathcal{T}(\mathcal{F}, \mathcal{V})$. Hence, by Lem. 4.9(vi), for infinitely many i, $\mathcal{I}(t_i\sigma_i) \succsim^* \circ \to_{\mathcal{C}_\varepsilon} \circ \succsim^* \mathcal{I}(s_{i+1}\sigma_{i+1})$ or by Lem. 4.9(ii), for infinitely many i, $\mathcal{I}(s_i\sigma_i) \to^+_{\mathcal{C}_\varepsilon} s_i\mathcal{I}(\sigma_i)$. As $\succ$ contains $\mathcal{C}_\varepsilon$ and is monotone, we also have $\to_{\mathcal{C}_\varepsilon} \subseteq \succ$, and hence in both cases we obtain infinitely many strict decreases. Using the same reasoning as for (‡), we have infinitely many i with $s_i\mathcal{I}(\sigma_i)$ $(\succ \cup \succsim) \circ \succsim^*$ $\circ \succ \circ \succsim^*$ $s_{i+1}\mathcal{I}(\sigma_{i+1})$ and for the remaining is we can use the previous results showing $s_i\mathcal{I}(\sigma)$ $(\succ \cup \succsim) \circ \succsim^*$ $s_{i+1}\mathcal{I}(\sigma_{i+1})$. Then, using Lem. 3.4 where $N = (\succ \cup \succsim)^*$ and $S = N \circ \succ \circ N$, yields a contradiction.

Hence, for all i we obtain $s_{n+i}\sigma_{n+i} \to_{\mathcal{P}_{\neg\mathcal{F}}} t_{n+1}\sigma_{n+i} \to^*_{\mathcal{U}_{\neg\mathcal{F}}} s_{n+i+1}\sigma_{n+i+1}$. Since $\mathcal{P} \cup \mathcal{U} \subseteq \succ \cup \succsim$ and since both $\succ$ and $\succsim$ are monotone and stable, we conclude (again by Lem. 3.4) that from some point onwards only rules from $(\mathcal{P}_{\neg\mathcal{F}} \cup \mathcal{U}_{\neg\mathcal{F}}) \setminus \succ$ are used. Hence, we have constructed a $(\mathcal{P}_{\neg\mathcal{F}} \setminus \succ, \mathcal{U}_{\neg\mathcal{F}} \setminus \succ)$-chain which is also minimal since $\mathsf{SN}_{\mathcal{R}}(t_i\sigma_i)$ implies $\mathsf{SN}_{\mathcal{U}_{\neg\mathcal{F}} \setminus \succ}(t_i\sigma_i)$ as $\mathcal{U}_{\neg\mathcal{F}} \subseteq \mathcal{R}$ by the requirement $\mathcal{U} \subseteq \mathcal{R}$. ∎

To obtain an executable function which checks for correct applications of Thm. 4.6 it is only demanded that the input and output DP problem, the reduction pair (without the details of the fresh symbol c), and the usable rules are given. The corresponding interpretation / precedence / ... for c is then added automatically. Moreover, the argument filter is constructed from the reduction pair. For example, for polynomial interpretations one always defines π in a way that $i \in \pi(f)$ iff x_i occurs within $\mathcal{P}ol(f(\vec{x}_n))$. In this way, $\succsim$ is always π-compatible and $\mathcal{C}_\varepsilon \subseteq \succsim$. Hence, for the automation of Thm. 4.6 where $\succ$ is not monotone, one only has to check $\mathcal{P} \subseteq \succ \cup \succsim$, $\mathcal{U} \subseteq \succsim$, and $\mathsf{urClosed}^\pi_{\mathcal{U}}(\mathcal{P}, \mathcal{R})$ since the remaining requirements are satisfied by construction.

For the other case, where also rules of $\mathcal{R}$ are deleted, it is additionally checked that $\succ$ is monotone and that $\mathcal{U} \subseteq \mathcal{R}$. To achieve the former for polynomials, it is ensured that the coefficients of all variables are larger than zero and that all remaining parts of the polynomial are non-negative. For path orders in combination with argument filters, it is ensured that no argument is dropped, i.e., the argument filter may only permute and duplicate arguments.

5. Experiments

In the end, what we want to have is a workflow which automatically certifies or rejects a given termination proof in CPF-format (a common format for termination proofs that is supported by all certifiers).[5] To this end, we have to parse the proof, detect which processors have been applied on which DP problems, and ensure that the preconditions of every processor are met. We achieved this goal by writing a CPF parser and for each processor an executable function which checks the preconditions. If a processor application cannot be certified, the function rejects, providing an informative error message. As the parser and

[5] http://cl-informatik.uibk.ac.at/software/cpf/

the check-functions are written in Isabelle, we just invoke Isabelle's code-generator [13] to obtain the executable program CeTA from IsaFoR.

This is in contrast to the other two certifiers CiME/Coccinelle and Rainbow/CoLoR.[6] Both of them provide a parser (as part of CiME and Rainbow) that takes a termination proof and produces a Coq-script as output. The resulting script refers to facts proven in Coccinelle and CoLoR, respectively, which can then be checked by Coq.

For more details on this difference and the architecture of the overall proof-checking function in CeTA we refer to [23].

To measure the impact of our results we used 5 strategies for the two termination tools AProVE [11] and T$_{\text{T}}$T$_2$ [18].

- In the ***basic*** strategy the termination tools only use the dependency graph processor and the reduction pair processor without usable rules. (These are the only techniques that have been described in [23].)
- The ***sc*** strategy is an extension of ***basic*** by the subterm criterion processor.
- Similarly, ***ur*** is like ***basic*** except that usable rules may be used.
- The fourth strategy, ***sc+ur***, is a combination of the previous three.
- Finally, ***full***, is a strategy where all CeTA-certifiable processors may be used. We refer to http://cl-informatik.uibk.ac.at/software/ceta/versions.php for the details where all techniques are listed. (Our experiments have been performed using CeTA version 1.10.)

Note that for ***basic***, ***sc***, ***ur***, and ***sc+ur***, we only take linear polynomial interpretations as reduction pairs, whereas in ***full*** also other reduction pairs are used.

For each of the 2132 standard TRSs in the Termination Problem Database (version 7.0.2)[7] and for each strategy, we first ran both termination tools for at most one minute and then tried to certify all successful proofs with CeTA. The experiments were performed on a machine with 8 Dual Core AMD Opteron 885 processors and 64 GB RAM running Linux. An overview of the results is given in the following table where the column labels A, C, and T, refer to AProVE, CeTA, and T$_{\text{T}}$T$_2$, respectively.

	basic		***sc***		***ur***		***sc+ur***		***full***	
	A	C	A	C	A	C	A	C	A	C
YES	453	453	566	566	681	681	684	684	1242	1242
avg. time		0.063		0.051		0.064		0.061		0.051
	T	C	T	C	T	C	T	C	T	C
YES	439	439	553	553	663	663	669	669	1223	1223
avg. time		0.059		0.048		0.065		0.062		0.074

The table rows show successful termination proofs / certificates for termination proofs (YES), and the average time for certifying (in seconds). All details on the experiments are available on CeTA's website.

When comparing ***basic*** with ***sc+ur*** one can observe that adding the subterm criterion and usable rules helps to increase the number of certified termination proofs by over 50% for both termination tools. Moreover, checking the additional application conditions of the new techniques—where $\mathsf{urClosed}^{\pi}_{\mathcal{U}}(\mathcal{P},\mathcal{R})$ is the most expensive one—does not have any measurable impact on the certification time. That checking AProVE's proofs is slightly

[6] Note that for a restricted set of techniques, CoLoR also features code-generation.

[7] available at http://termcomp.uibk.ac.at/

faster is explained by the fact that TTT2 always produces proofs with polynomial orders over the rationals, even if all coefficients are naturals. And thus, for TTT2's proofs, CeTA always has to perform computations over the rationals.

Our results also helped to win the annual termination competition for certified termination of TRSs in 2009.[8] First several termination tools were run to generate proofs on a random subset of 365 TRSs from the TPDB. For this, the tools where usually configured for a specific certifier in mind by restricting the set of termination techniques correspondingly. Then, all certifiers were run on all proofs. The following table summarizes the results, where the bold entries correspond to those proofs which were constructed specifically for that certifier.

tool	AProVE	TTT2	AProVE	CiME	AProVE	total
intended certifier	CeTA		Coccinelle		CoLoR	
proofs	259	264	165	56	220	964
CeTA	**259**	**264**	94	50	107	774
Coccinelle	12	2	**104**	**55**	30	203
CoLoR	67	53	92	9	**178**	399

We observe that many proofs generated for CeTA cannot be handled by the other certifiers—only between 1 % and 26 % of these proofs have been certified—where one major reason is that the other certifiers do not incorporate usable rules.

Looking at the other direction we see, that even if the termination tool produced proofs for another certifier, CeTA (version 1.09) achieved between 60 % and 90 % of the score of the intended certifier.

In total, only 190 proofs have been rejected by CeTAin the competition. Of these proofs, 65 are supported in the meantime (the competition version did not feature monotone matrix interpretations [7], which are supported by version 1.10), 117 contain unsupported techniques (non-linear polynomial orders and RPO [6]), 6 are obviously buggy (e.g., the subterm criterion is applied with a projection that maps a binary symbol to its third argument), and 2 are faulty (some LPO was wrongly applied and some argument filter delivers an unsolvable constraint). (At least for one of this proofs we know that this was due to an output bug of the proof producing tool.)

6. Conclusion

We have presented the first formalization of two important termination techniques within the theorem prover Isabelle/HOL: the subterm criterion and the reduction pair processor with usable rules, where we combined the improvements of [10] and [12]. The integration of these techniques into our termination proof certifier CeTA allowed to certify significantly more termination proofs.

However, there are several termination techniques that have not been certified by now. To change this, in the future we aim at certifying several techniques for innermost termination like narrowing, rewriting, and instantiation [1, 12], or estimations of innermost dependency graphs [1, 10, 14].

[8] http://termcomp.uibk.ac.at/termcomp/competition/certificationResults.seam?cat=10235&comp=101722

References

[1] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[3] F. Blanqui, W. Delobel, S. Coupet-Grimal, S. Hinderer, and A. Koprowski. CoLoR, a Coq library on rewriting and termination. In *Proc. WST'06*, pages 69–73, 2006.

[4] É. Contejean, P. Courtieu, J. Forest, A. Paskevich, O. Pons, and X. Urbain. A3PAT, an approach for certified automated termination proofs. In *Proc. PEPM'10*. To appear.

[5] É. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In *Proc. FroCoS'07*, LNAI 4720, pages 148–162, 2007.

[6] N. Dershowitz. Termination of rewriting. *J. Symb. Comp.*, 3:69–116, 1987.

[7] J. Endrullis, J. Waldmann, and H. Zantema. Matrix Interpretations for Proving Termination of Term Rewriting. *Journal of Automated Reasoning*, 40(2-3):195–220, 2008.

[8] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. Maximal termination. In *Proc. RTA'08*, LNCS 5117, pages 110–125, 2008.

[9] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. LPAR'04*, LNAI 3452, pages 301–331, 2005.

[10] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. FroCoS'05*, LNAI 3717, pages 216–231, 2005.

[11] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the DP framework. In *Proc. IJCAR'06*, LNAI 4130, pages 281–286, 2006.

[12] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[13] F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In *Proc. FLOPS'10*. To appear.

[14] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1-2):172–199, 2005.

[15] N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.

[16] S. Kamin and J. J. Lévy. Two generalizations of the recursive path ordering. Unpublished Manuscript, University of Illinois, IL, USA, 1980.

[17] A. Koprowski and J. Waldmann. Arctic termination ...below zero. In *Proc. RTA'08*, LNCS 5117, pages 202–216, 2008.

[18] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In *Proc. RTA'09*, volume 5595 of *LNCS*, pages 295–304, 2009.

[19] D. Lankford. On proving term rewriting systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, Ruston, LA, USA, 1979.

[20] S. Lucas. Polynomials over the reals in proofs of termination: From theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547–586, 2005.

[21] T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. LNCS 2283. Springer, 2002.

[22] R. Thiemann. *The DP Framework for Proving Termination of Term Rewriting*. PhD thesis, RWTH Aachen University, 2007. Available as Technical Report AIB-2007-17, http://aib.informatik.rwth-aachen.de/2007/2007-17.pdf.

[23] R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Proc. TPHOLs'09*, LNCS 5674, pages 452–468, 2009.

[24] X. Urbain. Modular & incremental automated termination proofs. *Journal of Automated Reasoning*, 32(4):315–355, 2004.

[25] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.