Modular and Certified Semantic Labeling and Unlabeling*

Christian Sternagel and René Thiemann

Institute of Computer Science, University of Innsbruck, Austria
{christian.sternagel|rene.thiemann}@uibk.ac.at

Abstract

Semantic labeling is a powerful transformation technique to prove termination of term rewrite systems. The dual technique is unlabeling. For unlabeling it is essential to drop the so called decreasing rules which sometimes have to be added when applying semantic labeling. We indicate two problems concerning unlabeling and present our solutions.

The first problem is that currently unlabeling cannot be applied as a modular step, since the decreasing rules are determined by a semantic labeling step which may have taken place much earlier. To this end, we give an implicit definition of decreasing rules that does not depend on any knowledge about preceding labelings.

The second problem is that unlabeling is in general unsound. To solve this issue, we introduce the notion of extended termination problems. Moreover, we show how existing termination techniques can be lifted to operate on extended termination problems.

All our proofs have been formalized in Isabelle/HOL as part of the IsaFoR/CeTA project.

1998 ACM Subject Classification F.4.2

Keywords and phrases semantic labeling, certification, term rewriting, unlabeling

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

Category Regular Research Paper

1 Introduction

In recent years, termination provers for term rewrite systems (TRSs) became more and more powerful. Nowadays, we do no longer have to prove termination by embedding all rules of a TRS into a single reduction order. Instead, most provers construct multi-step proofs by combining different termination techniques[1] resulting in tree-like termination proofs. As a result, termination provers became more complex and thus, more error-prone. It is regularly demonstrated that we cannot blindly trust termination provers. Every now and then, some prover delivers a faulty proof. Most of the time, this is only detected if there is another prover giving a contradictory answer. Furthermore, it just is too much work to check a generated proof by hand. (Besides, checking by hand is not very reliable.)

To solve this issue, recent interest is in the automatic certification of termination proofs [3, 4, 18]. To this end, we formalized many termination techniques in our Isabelle/HOL [15] library IsaFoR [18] (in the remainder we just write *Isabelle*, instead of *Isabelle/HOL*). Using IsaFoR, we obtain CeTA, an automatic certifier for termination proofs.

* This research is supported by FWF (Austrian Science Fund) project P22767-N13.

[1] Several termination techniques are based upon reduction orders, but there are also techniques which do not generate orders. Hence, the multi-step proofs are not just a lexicographic combination of orders.

In this paper, we present our formalization of semantic labeling and unlabeling [19], two important termination techniques. Semantic labeling introduces differently labeled variants of the same symbol, allowing a distinction in orders, etc. Semantic labeling typically produces large TRSs. Hence, unlabeling is important to keep the number of symbols and rules small.

▶ **Example 1.1.** Consider the small TRS `Secret_05/teparla3` from the termination problem database (TPDB) which only consists of two rules, has two different symbols, and two variables. We just describe the structure of the proof that has been generated by the termination prover AProVE [10] in 21 seconds during the 2008 termination competition.[2]

After applying the dependency pair transformation [1] and some standard techniques, a termination problem containing three rules and three different symbols is obtained. Then, semantic labeling is applied. The result after simplification, is a system with five rules and seven different symbols. Unlabeling yields a problem with three rules and three symbols. Another labeling produces a new termination problem with 12 rules. This is finally proven to be terminating using a matrix interpretation [6] of dimension two.

Note that without the unlabeling step, the second labeling would have returned a system with 5025 rules instead of 12—for this huge termination problem no suitable matrix interpretation of dimension two is detected.

Whereas the previous example shows that unlabeling is essential to keep systems small, we also found examples where unlabeling was the key to get a successful termination proof at all, cf. Example 4.2 for details.

Unfortunately, unlabeling is not sound in general. In order to allow nested labeling and unlabeling and turn unlabeling into a sound and modular technique (not relying on context information), we have designed a new framework. All existing termination techniques are easily integrated in this framework. In fact, CeTA uses the new framework for certification.

Note that all the proofs that are presented (or omitted) in the following, have been formalized as part of IsaFoR. Hence, we merely give sketches of our "real" proofs. Our goal is to show the general proof outlines and help to understand the full proofs. The 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 Section 2, we recapitulate some required notions of term rewriting as well as the basic definitions of semantic labeling. Afterwards, in Section 3, we give some challenges for modular labeling and unlabeling. Then, in Section 4, we extend the previous results to the dependency pair framework. We discuss challenges for the certification in Section 5. Our experiments are presented in Section 6 before we conclude in Section 7.

2 Preliminaries

2.1 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 $f(\vec{t_n})$. For brevity we write $\vec{t_n}$ to denote a sequence of n elements $t_1, \ldots, t_n$ and $(h(\vec{t_n}))$

[2] See `http://termcomp.uibk.ac.at/termcomp/competition/resultDetail.seam?resultId=35708`

(note the additional pair of parentheses) for $(h(t_1), \ldots, h(t_n))$, i.e., mapping a function h over the elements of a sequence $\vec{t_n}$. A *context* C is a term containing exactly one hole ($\Box$). Replacing $\Box$ in a context C by a term t is denoted by $C[t]$. A *(rewrite) rule* is a pair of terms $\ell \to r$ and a TRS $\mathcal{R}$ is a set of such rules. The *rewrite relation (induced by $\mathcal{R}$)* $\to_{\mathcal{R}}$ is the closure under substitutions and contexts of $\mathcal{R}$, i.e., $s \to_{\mathcal{R}} t$ iff there is a context C, a rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$.

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^+ (S^*) for the (reflexive and) transitive closure of S.

▶ **Definition 2.1** (Termination Technique). A *termination technique* is a mapping TT from TRSs to TRSs. It is *sound* if termination of $TT(\mathcal{R})$ implies termination of $\mathcal{R}$.

Using sound termination techniques one tries to modify a given TRS $\mathcal{R}$ until the empty TRS is reached. If this succeeds, one obtains a proof tree showing termination of $\mathcal{R}$.

2.2 Semantic Labeling

An algebra $\mathcal{A}$ over $\mathcal{F}$ is a pair $(A, \{f_{\mathcal{A}}\}_{f \in \mathcal{F}})$ consisting of a non-empty carrier A and an interpretation function $f_{\mathcal{A}} \colon A^n \to A$ for every n-ary function symbol $f \in \mathcal{F}$. Given an assignment $\alpha \colon \mathcal{V} \to A$, we write $[\alpha]_{\mathcal{A}}(t)$ for the interpretation of the term t. An algebra $\mathcal{A}$ is a model of a rewrite system $\mathcal{R}$, if $[\alpha]_{\mathcal{A}}(\ell) = [\alpha]_{\mathcal{A}}(r)$ for all rules $\ell \to r \in \mathcal{R}$ and all assignments α. If the carrier A is equipped with a well-founded order $>_A$ such that $[\alpha]_{\mathcal{A}}(\ell) \geqslant_A [\alpha]_{\mathcal{A}}(r)$ for all $\ell \to r \in \mathcal{R}$ and all assignments α, then $\mathcal{A}$ is a *quasi-model* of $\mathcal{R}$.

For each function symbol f there also is a corresponding non-empty set L_f of labels for f and a labeling function $\ell_f \colon A^n \to L_f$. The labeled signature $\mathcal{F}_{\mathsf{lab}}$ consists of n-ary function symbols f_a for every n-ary function symbol $f \in \mathcal{F}$ and label $a \in L_f$. The labeling function ℓ_f determines the label of the root symbol f of a term $f(\vec{t_n})$ based on the values of the arguments $\vec{t_n}$. For every assignment $\alpha \colon \mathcal{V} \to A$ the mapping $\mathsf{lab}_\alpha \colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}_{\mathsf{lab}}, \mathcal{V})$ is inductively defined by

$$\mathsf{lab}_\alpha(t) = \begin{cases} f_{\ell_f([\alpha]_{\mathcal{A}}(\vec{t_n}))}(\mathsf{lab}_\alpha(\vec{t_n})) & \text{if } t = f(\vec{t_n}), \\ t & \text{otherwise.} \end{cases}$$

The labeled TRS $\mathsf{lab}(\mathcal{R})$ over the signature $\mathcal{F}_{\mathsf{lab}}$ consists of the rules $\mathsf{lab}_\alpha(\ell) \to \mathsf{lab}_\alpha(r)$ for all $\ell \to r \in \mathcal{R}$ and $\alpha \colon \mathcal{V} \to A$.

For quasi-models, every set of labels L_f needs to be equipped with a well-founded order $>_{L_f}$, giving rise to the set $\mathcal{D}\mathrm{ec}$ of *decreasing rules*:

$$\mathcal{D}\mathrm{ec} = \{f_a(\vec{x_n}) \to f_b(\vec{x_n}) \mid a, b \in L_f, a >_{L_f} b,\ n\text{-ary } f \in \mathcal{F}\}$$

Furthermore, every interpretation function $f_{\mathcal{A}}$ and every labeling function ℓ_f has to be weakly monotone, i.e., if $a \geqslant_A a'$ then $f_{\mathcal{A}}(a_1, \ldots, a, \ldots, a_n) \geqslant_A f_{\mathcal{A}}(a_1, \ldots, a', \ldots, a_n)$ and $\ell_f(a_1, \ldots, a, \ldots, a_n) \geqslant_{L_f} \ell_f(a_1, \ldots, a', \ldots, a_n)$.

Unlabeling a symbol is defined via the following function, removing one layer of labels. Then, the function is extended homomorphically to terms, rules, and TRSs.

$$\mathsf{unlab}(f) = \begin{cases} g & \text{if } f = g_a, \\ f & \text{if } f \text{ is not labeled.} \end{cases}$$

In [19], Zantema showed that labeled TRSs can simulate their unlabeled counterparts (corresponding to **1** and **2** in the following lemma; **3** and **4** are obvious).

▶ **Lemma 2.2.** *Let $\mathcal{R}$ be a TRS,$\mathcal{A}$ an algebra, and α an arbitrary assignment.*

1. *If $\mathcal{A}$ is a model of $\mathcal{R}$ then $t \to_{\mathcal{R}} u$ implies $\mathsf{lab}_\alpha(t) \to_{\mathsf{lab}(\mathcal{R})} \mathsf{lab}_\alpha(u)$.*
2. *If $\mathcal{A}$ is a quasi-model of $\mathcal{R}$ then $t \to_{\mathcal{R}} u$ implies $\mathsf{lab}_\alpha(t) \to^{+}_{\mathsf{lab}(\mathcal{R})\cup\mathcal{D}\mathrm{ec}} \mathsf{lab}_\alpha(u)$.*
3. *$t \to_{\mathsf{lab}(\mathcal{R})} u$ implies $\mathsf{unlab}(t) \to_{\mathcal{R}} \mathsf{unlab}(u)$.*
4. *$t \to_{\mathcal{D}\mathrm{ec}} u$ implies $\mathsf{unlab}(t) = \mathsf{unlab}(u)$.*

From Lemma 2.2 we obtain that $\mathcal{R}$ is terminating if and only if $\mathsf{lab}(\mathcal{R})$ ($\cup\,\mathcal{D}\mathrm{ec}$) is terminating when $\mathcal{A}$ is a (quasi-)model of $\mathcal{R}$. Completeness is achieved by unlabeling all terms in a possible infinite rewrite sequence of the labeled TRS. Soundness is proved by transforming a presupposed infinite rewrite sequence in $\mathcal{R}$ into an infinite rewrite sequence in $\mathsf{lab}(\mathcal{R})$ ($\cup\,\mathcal{D}\mathrm{ec}$). This is done by applying the labeling function $\mathsf{lab}_\alpha(\cdot)$ (for an arbitrary assignment α) to all terms in the infinite rewrite sequence of $\mathcal{R}$. Hence, semantic labeling is sound and complete for termination (using models and quasi-models, respectively).

3 Modular Semantic Labeling and Unlabeling

One problem with semantic labeling is that the labeled system is usually large. Hence, termination provers such as AProVE [10], Jambox [5], Torpa [20], and TPA [13] perform labeling, then try to simplify the resulting TRS by sound termination techniques, and afterwards *unlabel* the TRS again, to continue on a small system. This poses two challenges:

1. If labeling was performed using a quasi-model, then the decreasing rules are added. However, unlabeling a decreasing rule $f_a(\vec{x}_n) \to f_b(\vec{x}_n)$ leads to the nonterminating rule $f(\vec{x}_n) \to f(\vec{x}_n)$. Hence, one has to remove the decreasing rules before unlabeling.
2. Between labeling and unlabeling, arbitrary (sound) termination techniques may be applied. However, for unlabeling we want to remove the decreasing rules that are determined by the corresponding labeling step. Hence, unlabeling is not a modular technique that only takes a TRS as input. Instead, it relies on context information, namely the decreasing rules that have been used in the corresponding labeling step (which may occur several steps upwards in the termination proof).

Solving the first challenge is technically easy: just remove the decreasing rules before unlabeling. The only question is, whether it is always sound to remove the decreasing rules.

To handle the second challenge, we propose an implicit definition of decreasing rules.

▶ **Definition 3.1** (Decreasing rules of a TRS). We define the *decreasing rules of a TRS $\mathcal{R}$* as $\mathcal{D}(\mathcal{R}) = \{\ell \to r \in \mathcal{R} \mid \mathsf{unlab}(\ell) = \mathsf{unlab}(r) \wedge \ell \neq r\}$. We further define the *unlabeled version of a TRS* as $\mathcal{U}(\mathcal{R}) = \mathsf{unlab}(\mathcal{R} \setminus \mathcal{D}(\mathcal{R}))$.

The condition $\ell \neq r$ ensures that a labeled variant of an original rule is never decreasing. For example, if $f(\vec{x}_n) \to f(\vec{x}_n)$ is a rule (and hence the original TRS is not terminating), then each labeled variant has the form $f_a(\vec{x}_n) \to f_a(\vec{x}_n)$ for some $a \in L_f$. If we would consider such a rule as decreasing, we could transform a nonterminating TRS into a terminating one, using labeling and unlabeling.

▶ **Lemma 3.2.** *Let L_f and $>_{L_f}$ be given for each symbol f to determine $\mathcal{D}\mathrm{ec}$. Then $\mathcal{D}(\mathcal{D}\mathrm{ec}) = \mathcal{D}\mathrm{ec}$, $\mathcal{D}(\mathsf{lab}(\mathcal{R})) = \varnothing$, $\mathcal{D}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}) = \mathcal{D}\mathrm{ec}$, and $\mathcal{U}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}) = \mathcal{R}$.*

Now it is easy to define a modular version of unlabeling which does not require external knowledge about what the decreasing rules are.

▶ **Definition 3.3** (Unlabeling as modular termination technique)**.** The *unlabeling termination technique* replaces a TRS $\mathcal{R}$ by $\mathcal{U}(\mathcal{R})$.

Hence, we solved the second challenge and made unlabeling into an independent technique which does not need any knowledge on the previous application of semantic labeling that introduced the decreasing rules. Thus, termination proofs can now use the following structure where no global information has to be passed around:

1. Switch from $\mathcal{R}$ to $\mathsf{lab}(\mathcal{R})$ $(\cup \mathcal{D}\text{ec})$.
2. Modify $\mathsf{lab}(\mathcal{R})$ $(\cup \mathcal{D}\text{ec})$ by sound termination techniques resulting in $\mathcal{R}'$.
3. Unlabel $\mathcal{R}'$ resulting in $\mathcal{U}(\mathcal{R}')$.

Although this approach is used in termination provers, it is unsound in general as not every sound termination technique may be used between labeling and unlabeling. This is illustrated by the following example.

▶ **Example 3.4.** We start with the nonterminating TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b}), \mathsf{b} \to \mathsf{a}\}$. Then, we apply semantic labeling using the algebra $\mathcal{A}$ with $A = \{0, 1\}$, interpretations $\mathsf{f}_{\mathcal{A}}(x) = 0$, $\mathsf{a}_{\mathcal{A}} = 0$, $\mathsf{b}_{\mathcal{A}} = 1$, $L_{\mathsf{f}} = A$, $\ell_{\mathsf{f}}(x) = x$, and the standard order on the naturals. Note that $\mathcal{A}$ is a quasi-model of $\mathcal{R}$. The resulting labeled TRS is $\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\text{ec} = \{\mathsf{f}_0(\mathsf{a}) \to \mathsf{f}_1(\mathsf{b}), \mathsf{b} \to \mathsf{a}, \mathsf{f}_1(x) \to \mathsf{f}_0(x)\}$. It is sound to replace $\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\text{ec}$ by the (nonterminating) TRS $\mathcal{R}' = \{\mathsf{f}_1(x) \to \mathsf{f}_0(x), \mathsf{f}_0(x) \to \mathsf{f}_1(x)\}$. However, unlabeling $\mathcal{R}'$ yields $\mathcal{U}(\mathcal{R}') = \varnothing$ as both rules in $\mathcal{R}'$ are decreasing according to Definition 3.1. Hence, some of the performed deductions were not sound. Since semantic labeling and the switch from $\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\text{ec}$ to $\mathcal{R}'$ are sound, we obtain that unlabeling via $\mathcal{U}$ is unsound.

The problematic step when unlabeling, i.e., when switching from $\mathcal{R}$ to $\mathcal{U}(\mathcal{R}) = \mathsf{unlab}(\mathcal{R} \setminus \mathcal{D}(\mathcal{R}))$, is the removal of the decreasing rules. If the decreasing rules are the only source of nontermination, then this removal is unsound. However, the decreasing rules $\mathcal{D}\text{ec}$ that are obtained from semantic labeling are always terminating. Thus, after labeling we have to prove termination of the labeled system including the decreasing rules, but we may assume that the decreasing rules are terminating. If we know that the decreasing rules are terminating, then unlabeling by $\mathcal{U}$ is sound. We obtain the following structure of termination proofs:

1. Initially we have to prove $\mathsf{SN}(\mathcal{R})$.
2. After labeling, we have to prove $\mathsf{SN}(\mathcal{D}(\mathcal{R}')) \Longrightarrow \mathsf{SN}(\mathcal{R}')$ for $\mathcal{R}' = \mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\text{ec}$.
3. Then, we modify $\mathcal{R}'$ to $\mathcal{R}''$ with $\mathsf{SN}(\mathcal{D}(\mathcal{R}'')) \Longrightarrow \mathsf{SN}(\mathcal{R}'')$ implies $\mathsf{SN}(\mathcal{D}(\mathcal{R}')) \Longrightarrow \mathsf{SN}(\mathcal{R}')$.
4. Finally, we unlabel $\mathcal{R}''$ resulting in $\mathcal{U}(\mathcal{R}'')$ and have to prove $\mathsf{SN}(\mathcal{U}(\mathcal{R}''))$.

This approach works fine for termination proofs where semantic labeling is not nested. However, we are aware of termination proofs where labeling is applied in a nested way.

▶ **Example 3.5.** Consider the TRS `Gebhardt_06/16` from the TPDB. During the 2008 termination competition, Jambox proved termination of this TRS, applying the following steps: labeling - labeling - labeling - polynomial order - unlabeling - four applications of polynomial orders - unlabeling - unlabeling.[3]

To support this kind of proof we define the following variant of strong normalization.

▶ **Definition 3.6.** An *extended termination problem* is a pair $(\mathcal{R}, n)$ consisting of a TRS $\mathcal{R}$ and a number $n \in \mathbb{N}$. An extended problem $(\mathcal{R}, n)$ is *strongly normalizing* $(\mathsf{SN}(\mathcal{R}, n))$ iff

$$(\forall m < n.\, \mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))) \quad \Longrightarrow \quad \mathsf{SN}(\mathcal{R}).$$

[3] See `http://termcomp.uibk.ac.at/termcomp/competition/resultDetail.seam?resultId=27220`

An *extended termination technique* is a mapping xTT from extended termination problems to extended termination problems. It is *sound* iff $\mathsf{SN}(xTT(\mathcal{R},n))$ implies $\mathsf{SN}(\mathcal{R},n)$.

The number n in an extended termination problem $(\mathcal{R},n)$ describes how often we can assume that the decreasing rules are terminating, and hence, it tells us how often we can delete the decreasing rules during unlabeling. The following lemma provides the link between both variants of strong normalization.

▶ **Lemma 3.7.** *1.* $\mathsf{SN}(\mathcal{R})$ *iff* $\mathsf{SN}(\mathcal{R},0)$.
2. *If* $\mathsf{SN}(\mathcal{R})$ *then* $\mathsf{SN}(\mathcal{R},n)$.

▶ **Lemma 3.8** (Extended Unlabeling). *Extended unlabeling is sound where*

$$\mathcal{U}(\mathcal{R},n) = \begin{cases} (\mathcal{U}(\mathcal{R}), n-1) & \text{if } n > 0,\\ (\mathsf{unlab}(\mathcal{R}), 0) & \text{otherwise.}\end{cases}$$

Proof. We only consider the interesting case where $n > 0$. So, we have to show $\mathsf{SN}(\mathcal{R},n)$ under the first assumption $\mathsf{SN}(\mathcal{U}(\mathcal{R}),n-1)$. To prove $\mathsf{SN}(\mathcal{R},n)$, we have to prove $\mathsf{SN}(\mathcal{R})$ under the second assumption $\forall m < n.\,\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))$. Since $n > 0$ we can choose $m = 0$ and obtain $\mathsf{SN}(\mathcal{D}(\mathcal{R}))$.

To show $\mathsf{SN}(\mathcal{R})$ we assume that there is an infinite $\to_{\mathcal{R}}$-derivation $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$ and obtain a contradiction. The infinite derivation is also an infinite $\to_{\mathcal{R}\setminus\mathcal{D}(\mathcal{R})} \cup \to_{\mathcal{D}(\mathcal{R})}$-derivation. Since $\mathcal{D}(\mathcal{R})$ is terminating, we know that there are infinitely many i with $t_i \to_{\mathcal{R}\setminus\mathcal{D}(\mathcal{R})} t_{i+1}$. Hence $\mathsf{unlab}(t_i) \to_{\mathcal{U}(\mathcal{R})} \mathsf{unlab}(t_{i+1})$ for all these i as $\mathcal{U}(\mathcal{R}) = \mathsf{unlab}(\mathcal{R}\setminus\mathcal{D}(\mathcal{R}))$. Moreover, for all i where $t_i \to_{\mathcal{D}(\mathcal{R})} t_{i+1}$, we know that $\mathsf{unlab}(t_i) \to_{\mathsf{unlab}(\mathcal{D}(\mathcal{R}))} \mathsf{unlab}(t_{i+1})$ and hence, $\mathsf{unlab}(t_i) = \mathsf{unlab}(t_{i+1})$ since every rule in $\mathsf{unlab}(\mathcal{D}(\mathcal{R}))$ has the same left- and right-hand side. Thus, we have constructed an infinite derivation for $\mathcal{U}(\mathcal{R})$ proving that $\mathsf{SN}(\mathcal{U}(\mathcal{R}))$ does not hold. Together with the assumption $\mathsf{SN}(\mathcal{U}(\mathcal{R}),n-1)$, we obtain that $\forall m < n-1.\,\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{U}(\mathcal{R}))))$ does not hold (by Definition 3.6). Hence, there is some $m < n-1$ such that $\mathsf{SN}(\mathcal{D}(\mathcal{U}^{m+1}(\mathcal{R})))$ does not hold. Now, using the second assumption and $m+1 < n$ we obtain $\mathsf{SN}(\mathcal{D}(\mathcal{U}^{m+1}(\mathcal{R})))$, providing the required contradiction. ◀

▶ **Lemma 3.9** (Extended Semantic Labeling). *Semantic labeling is sound as extended termination technique: Whenever we can switch from $\mathcal{R}$ to $\mathsf{lab}(\mathcal{R})$ $(\cup\,\mathcal{D}\mathrm{ec})$ via semantic labeling, then it is sound to switch from $(\mathcal{R},n)$ to $(\mathsf{lab}(\mathcal{R})\ (\cup\,\mathcal{D}\mathrm{ec}), n+1)$.*

Proof. Note that models are just a special case of quasi-models as already observed in [19]. Hence, we only consider quasi-models in the proof. So, assuming $\mathsf{SN}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}, n+1)$ we have to prove $\mathsf{SN}(\mathcal{R},n)$. To show the latter, we may assume $\forall m < n.\,\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))$ and have to prove $\mathsf{SN}(\mathcal{R})$. We do so by assuming that there is an infinite $\mathcal{R}$-derivation $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$ and deriving a contradiction. As we have a quasi-model we know that $\mathsf{lab}_\alpha(t_1) \to^+_{\mathsf{lab}(\mathcal{R})\cup\mathcal{D}\mathrm{ec}} \mathsf{lab}_\alpha(t_2) \to^+_{\mathsf{lab}(\mathcal{R})\cup\mathcal{D}\mathrm{ec}} \cdots$ is an infinite $\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}$-derivation, showing that $\mathsf{SN}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec})$ does not hold. By the conditions of semantic labeling, we further know $\mathsf{SN}(\mathcal{D}\mathrm{ec})$. Using $\mathsf{SN}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}, n+1)$ we conclude that $\forall m < n+1.\,\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec})))$ does not hold. Hence there is some $m < n+1$ such that $\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec})))$ does not hold. If $m = 0$ then by Lemma 3.2 we know that $\mathcal{D}(\mathcal{U}^m(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec})) = \mathcal{D}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}) = \mathcal{D}\mathrm{ec}$, and thus $\mathsf{SN}(\mathcal{D}\mathrm{ec})$ does not hold, a contradiction. Otherwise, $m = m' + 1$ for some m' where $m' < n$. Together with $\forall m < n.\,\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))$, we obtain $\mathsf{SN}(\mathcal{D}(\mathcal{U}^{m'}(\mathcal{R})))$. On the other hand, we know that $\mathsf{SN}(\mathcal{D}(\mathcal{U}^{m'+1}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec})))$ does not hold. This again leads to a contradiction since $\mathcal{D}(\mathcal{U}^{m'+1}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec})) = \mathcal{D}(\mathcal{U}^{m'}(\mathcal{U}(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}))) = \mathcal{D}(\mathcal{U}^{m'}(\mathcal{R}))$ by Lemma 3.2. ◀

The previous two lemmas show that labeling and unlabeling can be performed as independent techniques on extended termination problems.

The question remains how to integrate other existing termination techniques, i.e., which techniques may be applied between labeling and unlabeling. Here, we consider two variants.

▶ **Definition 3.10** (Lift). Let TT be some termination technique. Then $\mathsf{lift}(TT)$ and $\mathsf{lift}_0(TT)$ are extended termination techniques where $\mathsf{lift}(TT)(\mathcal{R}, n) = (TT(\mathcal{R}), n)$ and $\mathsf{lift}_0(TT)(\mathcal{R}, n) = (TT(\mathcal{R}), 0)$.

In principle $\mathsf{lift}(TT)$ is preferable, since it does not change n, allowing to remove the decreasing rules when unlabeling (which is not possible using $\mathsf{lift}_0(TT)$). However, in general the fact that TT is sound does not imply that $\mathsf{lift}(TT)$ is sound. This can easily be seen by reusing Example 3.4 where the extended termination problem $(\mathcal{R}, 0)$ is transformed to $(\mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}, 1)$ by semantic labeling, then to $(\mathcal{R}', 1)$ using $\mathsf{lift}(TT)$ for the unnamed sound termination technique TT in Example 3.4, and then to $(\varnothing, 0)$ by unlabeling. Since this establishes a complete termination proof for the nonterminating TRS $\mathcal{R}$, and since labeling and unlabeling are sound, we know that $\mathsf{lift}(TT)$ is unsound.

Since we cannot always use $\mathsf{lift}(TT)$, we give three different approaches to use termination techniques as extended termination techniques (in order of preference):

1. Identify a (hopefully large) class of termination techniques TT for which soundness of TT implies soundness of $\mathsf{lift}(TT)$.
2. Perform a direct proof that $\mathsf{lift}(TT)$ is sound as extended termination technique.
3. Use $\mathsf{lift}_0(TT)$ for any sound termination technique TT.

We first prove soundness of approach **3**.

▶ **Lemma 3.11.** *If TT is sound then $\mathsf{lift}_0(TT)$ is sound.*

Proof. We have to prove that $\mathsf{SN}(TT(\mathcal{R}), 0)$ implies $\mathsf{SN}(\mathcal{R}, n)$. So, assume $\mathsf{SN}(TT(\mathcal{R}), 0)$. Hence, $\mathsf{SN}(TT(\mathcal{R}))$ using Lemma 3.7(**1**). As TT is sound, we conclude $\mathsf{SN}(\mathcal{R})$ and this implies $\mathsf{SN}(\mathcal{R}, n)$ by Lemma 3.7(**2**). ◀

We start to prove soundness of $\mathsf{lift}(TT)$ for some sound termination technique TT in order to detect where the problem is. To prove soundness, we have to show that $\mathsf{SN}(TT(\mathcal{R}), n)$ implies $\mathsf{SN}(\mathcal{R}, n)$. Thus, assume $\mathsf{SN}(TT(\mathcal{R}), n)$. To prove $\mathsf{SN}(\mathcal{R}, n)$ we may assume that $\forall m < n. \mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))$ and have to prove $\mathsf{SN}(\mathcal{R})$. Since TT is sound, it suffices to prove $\mathsf{SN}(TT(\mathcal{R}))$. To this end, it suffices to show $\forall m < n. \mathsf{SN}(\mathcal{D}(\mathcal{U}^m(TT(\mathcal{R}))))$ by using $\mathsf{SN}(TT(\mathcal{R}), n)$. Hence, the only missing step is to conclude

$$\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R}))) \quad \Longrightarrow \quad \mathsf{SN}(\mathcal{D}(\mathcal{U}^m(TT(\mathcal{R})))). \tag{$\star$}$$

▶ **Lemma 3.12.** *If TT is sound and if ($\star$) is satisfied for all m, then $\mathsf{lift}(TT)$ is sound.*

A sufficient condition to ensure ($\star$) is to demand that $TT(\mathcal{R}) \subseteq \mathcal{R}$ as unlab, $\mathcal{D}$, and $\mathcal{U}$ are monotone w.r.t. set inclusion. Hence, all techniques that remove rules like rule removal via reduction pairs, or (RFC) matchbounds [7, 14] can safely be used between labeling and unlabeling. However, this excludes techniques like the flat context closure which is required for root-labeling.

▶ **Definition 3.13** (Root-Labeling). Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$. Let $\mathcal{A}_\mathcal{F}$ be an algebra with carrier $\mathcal{F}$. Moreover, for every n-ary $f \in \mathcal{F}$, we fix the interpretation function $f_{\mathcal{A}_\mathcal{F}}(\vec{x}_n) = f$, the set of labels $L_f = \mathcal{F}^n$, and the labeling function $\ell_f(\vec{x}_n) = (\vec{x}_n)$.

Note that root-labeling is just a specific instantiation of general semantic labeling with models. Hence, it is sound whenever $\mathcal{A}_\mathcal{F}$ is a model of $\mathcal{R}$. However, in general $\mathcal{A}_\mathcal{F}$ does not constitute a model of $\mathcal{R}$. Hence, a transformation technique was introduced that modifies $\mathcal{R}$ in a way that $\mathcal{A}_\mathcal{F}$ always is a model of the result: the *closure under flat contexts.*

▶ **Definition 3.14** (Flat Context Closure)**.** For an n-ary symbol f, the *flat context for the i-th argument* is $\mathcal{FC}^i(f) = f(x_1, \dots, x_{i-1}, \Box, x_{i+1}, \dots, x_n)$, where all the x_j are fresh variables. The set of *flat contexts over* $\mathcal{F}$ is defined by $\mathcal{FC}(\mathcal{F}) = \{\mathcal{FC}^i(f) \mid n\text{-ary } f \in \mathcal{F},\ 1 \leqslant i \leqslant n\}$. The *closure under flat contexts of a TRS $\mathcal{R}$ w.r.t. the signature $\mathcal{F}$* is given by

$$\mathcal{FC}_\mathcal{F}(\mathcal{R}) = \{C[\ell] \to C[r] \mid C \in \mathcal{FC}(\mathcal{F}), \ell \to r \in \mathcal{R}_\mathsf{a}\} \cup (\mathcal{R} \setminus \mathcal{R}_\mathsf{a})$$

where $\mathcal{R}_\mathsf{a}$ denotes those rules of $\mathcal{R}$, for which the root of the left-hand side and the root of the right-hand side differ.

Since Jambox applies root-labeling recursively (the labeling in Example 3.5 is root-labeling), we definitely would like to aim at a larger class of termination techniques than those which satisfy $TT(\mathcal{R}) \subseteq \mathcal{R}$. A natural extension would be to use the weaker condition $\to_{TT(\mathcal{R})} \subseteq \to_\mathcal{R}$. Then, also root-labeling together with the closure under flat contexts would be supported. Unfortunately, $\to_{TT(\mathcal{R})} \subseteq \to_\mathcal{R}$ does not imply $\to_{\mathcal{D}(\mathcal{U}^m(TT(\mathcal{R})))} \subseteq \to_{\mathcal{D}(\mathcal{U}^m(\mathcal{R}))}$ and thus, does not imply ($\star$). Moreover, in the following example we show that even if TT is sound and $\to_{TT(\mathcal{R})} \subseteq \to_\mathcal{R}$ then soundness of $\mathsf{lift}(TT)$ cannot be guaranteed.

▶ **Example 3.15.** Consider the TRS $\mathcal{R} = \{\mathsf{f}_1(x) \to \mathsf{f}_0(\mathsf{a}), \mathsf{f}_0(x) \to \mathsf{f}_1(x)\}$. Let TT be the termination technique that replaces $\mathcal{R}$ by $\mathcal{R}' = \{\mathsf{f}_1(\mathsf{a}) \to \mathsf{f}_0(\mathsf{a}), \mathsf{f}_0(x) \to \mathsf{f}_1(x)\}$. Then, TT is sound as $\mathcal{R}'$ is not terminating. Moreover, $\to_{\mathcal{R}'} \subseteq \to_\mathcal{R}$. Nevertheless, $\mathsf{lift}(TT)$ is unsound, since it would replace $(\mathcal{R}, 1)$ by $(\mathcal{R}', 1)$. That this replacement is unsound can be seen as follows: $\mathsf{SN}(\mathcal{R}, 1)$ does not hold since $\mathcal{R}$ is not terminating but the decreasing rules of $\mathcal{R}$ (i.e., $\mathcal{D}(\mathcal{R}) = \{\mathsf{f}_0(x) \to \mathsf{f}_1(x)\}$) are terminating. However, $\mathsf{SN}(\mathcal{R}', 1)$ is satisfied as $\mathcal{D}(\mathcal{R}') = \mathcal{R}'$ and hence termination of $\mathcal{D}(\mathcal{R}')$ implies termination of $\mathcal{R}'$.

We have seen that requiring $TT(\mathcal{R}) \subseteq \mathcal{R}$ is too restrictive to allow root-labeling. But only requiring $\to_{TT(\mathcal{R})} \subseteq \to_\mathcal{R}$ is unsound. However, there is another condition which is weaker than set inclusion, implies soundness, *and* allows the application of flat context closures.

▶ **Definition 3.16.** The *context subset relation* $\subseteq_c$ is defined as

$$\mathcal{R} \subseteq_c \mathcal{S} \text{ iff } \forall \ell \to r \in \mathcal{R}.\, \exists C, \ell' \to r' \in \mathcal{S}.\, \ell = C[\ell'] \wedge r = C[r'].$$

▶ **Lemma 3.17.** *1.* *$\mathcal{R} \subseteq \mathcal{S}$ implies $\mathcal{R} \subseteq_c \mathcal{S}$*
2. *$\mathcal{R} \subseteq_c \mathcal{S}$ implies $\to_\mathcal{R} \subseteq \to_\mathcal{S}$*
3. *$\mathcal{R} \subseteq_c \mathcal{S}$ implies $\mathcal{D}(\mathcal{R}) \subseteq_c \mathcal{D}(\mathcal{S})$ and $\mathcal{U}(\mathcal{R}) \subseteq_c \mathcal{U}(\mathcal{S})$*
4. *If TT is sound and $\forall \mathcal{R}.\, TT(\mathcal{R}) \subseteq_c \mathcal{R}$ then $\mathsf{lift}(TT)$ is sound*

Proof. **1.** To show $\mathcal{R} \subseteq_c \mathcal{S}$, let $\ell \to r \in \mathcal{R}$. Using $\mathcal{R} \subseteq \mathcal{S}$ we know that $\ell \to r \in \mathcal{S}$. Hence, $\exists C, \ell' \to r' \in \mathcal{S}.\, \ell = C[\ell'] \wedge r = C[r']$ by choosing $C = \Box$ and $\ell' \to r' = \ell \to r$.
2. Assume $t = D[\ell\sigma] \to_\mathcal{R} D[r\sigma] = s$ using some rule $\ell \to r \in \mathcal{R}$. As $\mathcal{R} \subseteq_c \mathcal{S}$, we obtain C and $\ell' \to r' \in \mathcal{S}$ such that $\ell = C[\ell']$ and $r = C[r']$. Hence, $t = D[\ell\sigma] = D[C[\ell']\sigma] = D[C\sigma[\ell'\sigma]] \to_\mathcal{S} D[C\sigma[r'\sigma]] = D[C[r']\sigma] = D[r\sigma] = s$.
3. We first show $\mathcal{D}(\mathcal{R}) \subseteq_c \mathcal{D}(\mathcal{S})$. So, let $\ell \to r \in \mathcal{D}(\mathcal{R})$. Hence, $\ell \to r \in \mathcal{R}$, $\mathsf{unlab}(\ell) = \mathsf{unlab}(r)$ and $\ell \neq r$. Using $\mathcal{R} \subseteq_c \mathcal{S}$ we obtain C and $\ell' \to r' \in \mathcal{S}$ such that $\ell = C[\ell']$ and $r = C[r']$. Thus, $\mathsf{unlab}(C)[\mathsf{unlab}(\ell')] = \mathsf{unlab}(C[\ell']) = \mathsf{unlab}(\ell) = \mathsf{unlab}(r) = \mathsf{unlab}(C[r']) =$

$\mathsf{unlab}(C)[\mathsf{unlab}(r')]$ shows that $\mathsf{unlab}(\ell') = \mathsf{unlab}(r')$. Similarly, $C[\ell'] = \ell \neq r = C[r']$ implies $\ell' \neq r'$. So, $\ell' \to r' \in \mathcal{D}(\mathcal{S})$ and thus, $\exists C, \ell' \to r' \in \mathcal{D}(\mathcal{S}).\, \ell = C[\ell'] \wedge r = C[r']$.
Now let us show $\mathcal{U}(\mathcal{R}) = \mathsf{unlab}(\mathcal{R} \setminus \mathcal{D}(\mathcal{R})) \subseteq_c \mathsf{unlab}(\mathcal{S} \setminus \mathcal{D}(\mathcal{S})) = \mathcal{U}(\mathcal{S})$. This property is the crucial part, since potentially we remove less rules from $\mathcal{R}$ than from $\mathcal{S}$. Assume $\mathsf{unlab}(\ell) \to \mathsf{unlab}(r) \in \mathcal{U}(\mathcal{R})$, i.e., $\ell \to r \in \mathcal{R}$ and $\mathsf{unlab}(\ell) \neq \mathsf{unlab}(r) \vee \ell = r$. As $\mathcal{R} \subseteq_c \mathcal{S}$ we obtain C and $\ell' \to r' \in \mathcal{S}$ such that $\ell = C[\ell']$ and $r = C[r']$. Hence, $\mathsf{unlab}(\ell) = \mathsf{unlab}(C[\ell']) = \mathsf{unlab}(C)[\mathsf{unlab}(\ell')]$ and $\mathsf{unlab}(r) = \mathsf{unlab}(C[r']) = \mathsf{unlab}(C)[\mathsf{unlab}(r')]$. Thus, we can simplify $\mathsf{unlab}(\ell) \neq \mathsf{unlab}(r) \vee \ell = r$ to $\mathsf{unlab}(C)[\mathsf{unlab}(\ell')] \neq \mathsf{unlab}(C)[\mathsf{unlab}(r')] \vee C[\ell'] = C[r']$ and further to $\mathsf{unlab}(\ell') \neq \mathsf{unlab}(r') \vee \ell' = r'$. Using $\ell' \to r' \in \mathcal{S}$ this shows that $\ell' \to r' \in \mathcal{S} \setminus \mathcal{D}(\mathcal{S})$ and thus, $\mathsf{unlab}(\ell') \to \mathsf{unlab}(r') \in \mathcal{U}(\mathcal{S})$. By choosing the context $\mathsf{unlab}(C)$ and the rule $\mathsf{unlab}(\ell') \to \mathsf{unlab}(r')$ we have finally shown that $\exists C, \ell' \to r' \in \mathcal{U}(\mathcal{S}).\, \mathsf{unlab}(\ell) = C[\ell'] \wedge \mathsf{unlab}(r) = C[r']$.

4. By Lemma 3.12 we only have to prove ($\star$). Using $TT(\mathcal{R}) \subseteq_c \mathcal{R}$ and **3** one can show that $\mathcal{U}^m(TT(\mathcal{R})) \subseteq_c \mathcal{U}^m(\mathcal{R})$ by induction on m. Using **3** again, we conclude $\mathcal{D}(\mathcal{U}^m(TT(\mathcal{R}))) \subseteq_c \mathcal{D}(\mathcal{U}^m(\mathcal{R}))$ and thus, $\to_{\mathcal{D}(\mathcal{U}^m(TT(\mathcal{R})))} \subseteq \to_{\mathcal{D}(\mathcal{U}^m(\mathcal{R}))}$ by **2**. Then ($\star$) immediately follows. ◀

▶ **Corollary 3.18.** *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$. Then $\mathsf{lift}(\mathcal{FC}_\mathcal{F})$ is sound.*

Proof. It was shown in [16] that $\mathcal{FC}_\mathcal{F}$ is sound for TRSs. Furthermore, $\mathcal{FC}_\mathcal{F}(\mathcal{R}) \subseteq_c \mathcal{R}$ by definition of $\mathcal{FC}(\mathcal{F})$ and thus, by Lemma 3.17(**4**), $\mathsf{lift}(\mathcal{FC}_\mathcal{F})$ is sound, too. ◀

Note that several termination techniques TT satisfy $TT(\mathcal{R}) \subseteq_c \mathcal{R}$ and hence, can be used between labeling and unlabeling. However, there are still some techniques which do not satisfy this requirement. Examples would be string reversal and uncurrying [11].

Of course, it is possible to use $\mathsf{lift}_0(TT)$, however, for string reversal also a direct soundness proof can be performed to show that lifting string reversal is sound.

▶ **Theorem 3.19.** *Let TT be the technique of string reversal where $TT(\mathcal{R}) = \mathsf{rev}(\mathcal{R})$, if $\mathcal{R}$ is a string rewrite system, and $TT(\mathcal{R}) = \mathcal{R}$, otherwise. Then $\mathsf{lift}(TT)$ is sound.*

Proof. By Lemma 3.12 we just have to prove ($\star$), i.e., we have to show for all m that $\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))$ implies $\mathsf{SN}(\mathcal{D}(\mathcal{U}^m(\mathsf{rev}(\mathcal{R}))))$. To this end, we have proven that reversing can be commuted with both $\mathcal{D}$ and $\mathcal{U}$: $\mathsf{rev}(\mathcal{D}(\mathcal{R})) = \mathcal{D}(\mathsf{rev}(\mathcal{R}))$ and $\mathsf{rev}(\mathcal{U}(\mathcal{R})) = \mathcal{U}(\mathsf{rev}(\mathcal{R})$. Hence, $\mathsf{rev}(\mathcal{D}(\mathcal{U}^m(\mathcal{R}))) = \mathcal{D}(\mathcal{U}^m(\mathsf{rev}(\mathcal{R})))$. This completes the proof: since string reversal is complete, we know that termination of $\mathcal{D}(\mathcal{U}^m(\mathcal{R}))$ implies termination of $\mathsf{rev}(\mathcal{D}(\mathcal{U}^m(\mathcal{R})))$ and therefore, also of $\mathcal{D}(\mathcal{U}^m(\mathsf{rev}(\mathcal{R})))$. ◀

To summarize, we can now certify termination proofs where labeling and unlabeling are modular techniques (and hence, can be applied recursively), and where all supported techniques of CeTA (except uncurrying) can be used between labeling and unlabeling.

An easy alternative to our extended termination techniques would be the use of relative rewriting. The obvious idea is to add the decreasing rules as relative rules when performing semantic labeling. In this way, unlabeling would directly be modular and sound, since one can always remove relative rules where both sides of the rule are identical. This alternative is used in the independent and unpublished formalization of semantic labeling in the CoLoR library. The main problem with this alternative is that some techniques like RFC matchbounds can be used in our framework, but not in combination with relative rewriting in general (during the termination competition in 2010 a tool has been disqualified for giving a wrong answer for a relative termination problem; the reason was the use of RFC matchbounds). For a further discussion on matchbounds and relative rewriting we refer to [12].

4 Dependency Pair Framework

The DP framework [8] is a way to modularize termination proofs. Instead of TRSs one investigates 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 $\mathsf{DP}(\mathcal{R})$ denotes the *dependency pairs* of $\mathcal{R}$ [1]. A $(\mathcal{P}, \mathcal{R})$-*chain* is a possibly infinite derivation of the form:

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

where $s_i \rightarrow t_i \in \mathcal{P}$ for all $i > 0$. 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* [8], if there is no minimal infinite $(\mathcal{P}, \mathcal{R})$-chain. Proving finiteness of a DP problem is done by simplifying $(\mathcal{P}, \mathcal{R})$ using so called *processors* recursively. A processor transforms a DP problem into a new DP problem. The aim is to reach a DP problem where the $\mathcal{P}$-component is empty (such DP problems are trivially finite). To conclude finiteness of the initial DP problem, the applied processors need to be *sound*. A processor $\mathcal{P}roc$ is sound whenever for all DP problems $(\mathcal{P}, \mathcal{R})$ we have that finiteness of $\mathcal{P}roc(\mathcal{P}, \mathcal{R})$ implies finiteness of $(\mathcal{P}, \mathcal{R})$.

Semantic labeling can easily be lifted to DP problems. Soundness of the following processor is an immediate consequence of [19].

▶ **Theorem 4.1.** *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem and $\mathcal{A}$ be an algebra. If $\mathcal{A}$ is a quasi-model of $\mathcal{R}$, then it is sound to return the DP problem* $(\mathsf{lab}(\mathcal{P}), \mathsf{lab}(\mathcal{R}) \cup \mathcal{D}ec)$.

The following example shows that unlabeling is not only necessary for efficiency, but that unlabeling is required to apply other techniques.

▶ **Example 4.2.** We consider the TRS `Secret_07/4` from the TPDB.

1:	$\mathsf{g}(\mathsf{c}, \mathsf{g}(\mathsf{c}, x)) \rightarrow \mathsf{g}(\mathsf{e}, \mathsf{g}(\mathsf{d}, x))$	4:	$\mathsf{g}(x, \mathsf{g}(y, \mathsf{g}(x, y))) \rightarrow \mathsf{g}(\mathsf{a}, \mathsf{g}(x, \mathsf{g}(y, \mathsf{b})))$
2:	$\mathsf{g}(\mathsf{d}, \mathsf{g}(\mathsf{d}, x)) \rightarrow \mathsf{g}(\mathsf{c}, \mathsf{g}(\mathsf{e}, x))$	5:	$\mathsf{f}(\mathsf{g}(x, y)) \rightarrow \mathsf{g}(y, \mathsf{g}(\mathsf{f}(\mathsf{f}(x)), \mathsf{a}))$
3:	$\mathsf{g}(\mathsf{e}, \mathsf{g}(\mathsf{e}, x)) \rightarrow \mathsf{g}(\mathsf{d}, \mathsf{g}(\mathsf{c}, x))$		

In the 2008 termination competition AProVE found a termination proof of the following structure (we present a simplified version, missing some unnecessary steps that have been applied in the original proof).[4] First, the initial DP problem is transformed into $(\mathcal{P}, \{1\text{–}4\})$ where $\mathcal{P}$ consists of the pairs $\mathsf{G}(\mathsf{c}, \mathsf{g}(\mathsf{c}, x)) \rightarrow \mathsf{G}(\mathsf{e}, \mathsf{g}(\mathsf{d}, x))$, $\mathsf{G}(\mathsf{d}, \mathsf{g}(\mathsf{d}, x)) \rightarrow \mathsf{G}(\mathsf{c}, \mathsf{g}(\mathsf{e}, x))$, and $\mathsf{G}(\mathsf{e}, \mathsf{g}(\mathsf{e}, x)) \rightarrow \mathsf{G}(\mathsf{d}, \mathsf{g}(\mathsf{c}, x))$. Then, labeling and further processing yields the DP problem $(\mathcal{P}', \mathcal{R}')$ where $\mathcal{P}'$ contains the pairs

$$\mathsf{G}_{00}(\mathsf{c}, \mathsf{g}_{00}(\mathsf{c}, x)) \rightarrow \mathsf{G}_{00}(\mathsf{e}, \mathsf{g}_{00}(\mathsf{d}, x)) \qquad \mathsf{G}_{00}(\mathsf{e}, \mathsf{g}_{00}(\mathsf{e}, x)) \rightarrow \mathsf{G}_{00}(\mathsf{d}, \mathsf{g}_{00}(\mathsf{c}, x))$$
$$\mathsf{G}_{00}(\mathsf{d}, \mathsf{g}_{00}(\mathsf{d}, x)) \rightarrow \mathsf{G}_{00}(\mathsf{c}, \mathsf{g}_{00}(\mathsf{e}, x))$$

and $\mathcal{R}'$ is the following TRS.

$$\mathsf{g}_{00}(\mathsf{c}, \mathsf{g}_{00}(\mathsf{c}, x)) \rightarrow \mathsf{g}_{00}(\mathsf{e}, \mathsf{g}_{00}(\mathsf{d}, x)) \qquad \mathsf{g}_{00}(\mathsf{c}, \mathsf{g}_{01}(\mathsf{c}, x)) \rightarrow \mathsf{g}_{00}(\mathsf{e}, \mathsf{g}_{01}(\mathsf{d}, x))$$
$$\mathsf{g}_{00}(\mathsf{d}, \mathsf{g}_{00}(\mathsf{d}, x)) \rightarrow \mathsf{g}_{00}(\mathsf{c}, \mathsf{g}_{00}(\mathsf{e}, x)) \qquad \mathsf{g}_{00}(\mathsf{d}, \mathsf{g}_{01}(\mathsf{d}, x)) \rightarrow \mathsf{g}_{00}(\mathsf{c}, \mathsf{g}_{01}(\mathsf{e}, x))$$
$$\mathsf{g}_{00}(\mathsf{e}, \mathsf{g}_{00}(\mathsf{e}, x)) \rightarrow \mathsf{g}_{00}(\mathsf{d}, \mathsf{g}_{00}(\mathsf{c}, x)) \qquad \mathsf{g}_{00}(\mathsf{e}, \mathsf{g}_{01}(\mathsf{e}, x)) \rightarrow \mathsf{g}_{00}(\mathsf{d}, \mathsf{g}_{01}(\mathsf{c}, x))$$

[4] See `http://termcomp.uibk.ac.at/termcomp/competition/resultDetail.seam?resultId=35909`

Hence, all labeled versions of Rule 4 have been deleted, and unlabeling yields the DP problem $(\mathcal{P}, \{1\text{–}3\})$. This DP problem is applicative. Hence, we may apply the $\mathcal{A}$-transformation [9] to obtain the DP problem having the pairs

$$\mathsf{C}(\mathsf{c}(x)) \to \mathsf{E}(\mathsf{d}(x)) \qquad \mathsf{D}(\mathsf{d}(x)) \to \mathsf{C}(\mathsf{e}(x)) \qquad \mathsf{E}(\mathsf{e}(x)) \to \mathsf{D}(\mathsf{c}(x))$$

and the rules

$$\mathsf{c}(\mathsf{c}(x)) \to \mathsf{e}(\mathsf{d}(x)) \qquad \mathsf{d}(\mathsf{d}(x)) \to \mathsf{c}(\mathsf{e}(x)) \qquad \mathsf{e}(\mathsf{e}(x)) \to \mathsf{d}(\mathsf{c}(x))$$

This DP problem is solved using standard techniques. Note that for the $\mathcal{A}$-transformation it was essential that unlabeling was performed, as the DP problem $(\mathcal{P}', \mathcal{R}')$ is not applicative.

Unfortunately, unlabeling as processor is in general unsound. In contrast to unlabeling on TRSs, here a problem already arises when using the model-version of semantic labeling without decreasing rules. The main reason is that unlabeling might introduce nontermination. Hence, minimality of an unlabeled infinite chain cannot be guaranteed.[5]

▶ **Example 4.3.** Consider the DP problem $(\mathcal{P}, \varnothing)$ where $\mathcal{P} = \{\mathsf{F}(x) \to \mathsf{F}(\mathsf{g}(\mathsf{a}))\}$. This DP problem is obviously not finite. Applying semantic labeling is trivially possible since there are no rules which have to satisfy the (quasi-)model condition. We choose $A = \{0, 1\}$, and for each f we define $f_{\mathcal{A}}(\dots) = 0$ and $\ell_f(\vec{x}_n) = (\vec{x}_n)$. We obtain the labeled pairs $\mathsf{lab}(\mathcal{P}) = \{\mathsf{F}_0(x) \to \mathsf{F}_0(\mathsf{g}_0(\mathsf{a})), \mathsf{F}_1(x) \to \mathsf{F}_0(\mathsf{g}_0(\mathsf{a}))\}$ and by Theorem 4.1 we know that the DP problem $(\mathsf{lab}(\mathcal{P}), \varnothing)$ is again not finite. We can further modify the DP problem by replacing it with $(\mathsf{lab}(\mathcal{P}), \mathcal{R})$ where $\mathcal{R} = \{\mathsf{g}_1(x) \to \mathsf{g}_1(x)\}$. Note that this modification is sound since $(\mathsf{lab}(\mathcal{P}), \mathcal{R})$ still allows a minimal infinite chain and is therefore not finite.

However, applying unlabeling we obtain the DP problem $(\mathcal{P}, \mathsf{unlab}(\mathcal{R}))$ which is finite as now the right-hand side $\mathsf{F}(\mathsf{g}(\mathsf{a}))$ of the only pair in $\mathcal{P}$ is not terminating w.r.t. $\mathcal{U}(\mathcal{R}) = \{\mathsf{g}(x) \to \mathsf{g}(x)\}$. Hence, unlabeling is unsound in general. The main problem is again that the notion of soundness is too weak. It allows the application of processors between labeling and unlabeling which may replace $(\mathsf{lab}(\mathcal{P}), \varnothing)$ by $(\mathsf{lab}(\mathcal{P}), \mathcal{R})$.

To solve this problem, we again add a counter n which tells us how often we may unlabel.

▶ **Definition 4.4.** An *extended DP problem* is a triple $(\mathcal{P}, \mathcal{R}, n)$ where $(\mathcal{P}, \mathcal{R})$ is a DP problem and $n \in \mathbb{N}$. An extended DP problem $(\mathcal{P}, \mathcal{R}, n)$ is *finite* iff there is no infinite chain

$$s_1\sigma_1 \to_{\mathcal{P}} t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2 \to_{\mathcal{P}} t_2\sigma_2 \to^*_{\mathcal{R}} s_3\sigma_3 \to_{\mathcal{P}} t_3\sigma_3 \to^*_{\mathcal{R}} \cdots$$

such that for all i: $\forall m \leqslant n.\, \mathsf{SN}_{\mathcal{U}^m(\mathcal{R})}(\mathsf{unlab}^m(t_i\sigma_i))$.

Hence, the only difference between finiteness of DP problems and extended DP problems is the minimality condition ($\mathsf{SN}_{\mathcal{R}}(t_i\sigma_i)$ versus $\forall m \leqslant n.\, \mathsf{SN}_{\mathcal{U}^m(\mathcal{R})}(\mathsf{unlab}^m(t_i\sigma_i))$). We therefore obtain a similar lemma to Lemma 3.7, but now for DP problems.

▶ **Lemma 4.5.** ***1.*** *$(\mathcal{P}, \mathcal{R})$ is finite iff $(\mathcal{P}, \mathcal{R}, 0)$ is finite.*
2. *If $(\mathcal{P}, \mathcal{R})$ is finite then $(\mathcal{P}, \mathcal{R}, n)$ is finite.*

As for termination techniques we can lift every processor to an extended processor.

[5] There is no problem in the formalization of semantic labeling in CoLoR at this point, as it does not feature *minimal* chains.

▶ **Definition 4.6** (Lift). Let $\mathcal{P}\mathrm{roc}$ be a processor with $\mathcal{P}\mathrm{roc}(\mathcal{P},\mathcal{R}) = (\mathcal{P}',\mathcal{R}')$. Then $\mathsf{lift}(\mathcal{P}\mathrm{roc})$ and $\mathsf{lift}_0(\mathcal{P}\mathrm{roc})$ are extended processors where $\mathsf{lift}(\mathcal{P}\mathrm{roc})(\mathcal{P},\mathcal{R},n) = (\mathcal{P}',\mathcal{R}',n)$ and $\mathsf{lift}_0(\mathcal{P}\mathrm{roc})(\mathcal{P},\mathcal{R},n) = (\mathcal{P}',\mathcal{R}',0)$.

We obtain similar results for lift_0 as for termination techniques: whenever $\mathcal{P}\mathrm{roc}$ is sound then $\mathsf{lift}_0(\mathcal{P}\mathrm{roc})$ is sound. However, additionally demanding that $\mathcal{R}' \subseteq_c \mathcal{R}$ or even $\mathcal{P}' \subseteq \mathcal{P} \wedge \mathcal{R}' = \mathcal{R}$ where $\mathcal{P}\mathrm{roc}(\mathcal{P},\mathcal{R}) = (\mathcal{P}',\mathcal{R}')$ does not suffice to ensure soundness of $\mathsf{lift}(\mathcal{P}\mathrm{roc})$. This is demonstrated in the upcoming example.

▶ **Example 4.7.** Let $\mathcal{P} = \{\mathsf{F}_0(x) \to \mathsf{F}_0(\mathsf{b})\}$, $\mathcal{P}' = \{\mathsf{F}_0(x) \to \mathsf{F}_0(\mathsf{g}_0(\mathsf{b}))\}$, and $\mathcal{R} = \{\mathsf{g}_1(x) \to \mathsf{g}_0(\mathsf{h}_1(x))\}$. Then $(\mathcal{P},\mathcal{R},1)$ is not finite as obviously there is an infinite $(\mathcal{P},\mathcal{R})$-chain where all terms in the chain are $\mathsf{F}_0(\mathsf{b})$ and moreover, $\mathsf{F}_0(\mathsf{b})$ is terminating w.r.t. $\mathcal{R}$ and $\mathsf{unlab}(\mathsf{F}_0(\mathsf{b})) = \mathsf{F}(\mathsf{b})$ is terminating w.r.t. $\mathcal{U}(\mathcal{R}) = \{\mathsf{g}(x) \to \mathsf{g}(\mathsf{h}(x))\}$. Hence, also $(\mathcal{P} \cup \mathcal{P}',\mathcal{R},1)$ is not finite by constructing the same chain.

Note that the processor $\mathcal{P}\mathrm{roc}$ which replaces $(\mathcal{P}\cup\mathcal{P}',\mathcal{R})$ by $(\mathcal{P}',\mathcal{R})$ is sound, since $(\mathcal{P}',\mathcal{R})$ is not finite: again, there is an infinite $(\mathcal{P}',\mathcal{R})$-chain, and every chain is also minimal since $\mathcal{R}$ is terminating. However, $\mathsf{lift}(\mathcal{P}\mathrm{roc})$ is unsound as $(\mathcal{P}',\mathcal{R},1)$ is finite: otherwise, there would be an infinite chain where $\mathsf{F}_0(\mathsf{g}_0(\mathsf{b}))$ is terminating w.r.t. $\mathcal{R}$ and $\mathsf{unlab}(\mathsf{F}_0(\mathsf{g}_0(\mathsf{b}))) = \mathsf{F}(\mathsf{g}(\mathsf{b}))$ is terminating w.r.t. $\mathcal{U}(\mathcal{R})$. But it is easy to see that $\mathsf{F}(\mathsf{g}(\mathsf{b}))$ is not terminating w.r.t. $\mathcal{U}(\mathcal{R})$.

Since requiring just $\mathcal{R}' \subseteq_c \mathcal{R}$ (or even $\mathcal{P}' \subseteq \mathcal{P} \wedge \mathcal{R}' = \mathcal{R}$) does not suffice to ensure soundness of $\mathsf{lift}(\mathcal{P}\mathrm{roc})$ we demand a slightly stronger property than soundness.

▶ **Definition 4.8.** A processor $\mathcal{P}\mathrm{roc}$ is *chain-identifying* iff whenever $\mathcal{P}\mathrm{roc}(\mathcal{P},\mathcal{R}) = (\mathcal{P}',\mathcal{R}')$ and there is some minimal infinite $(\mathcal{P},\mathcal{R})$-chain

$$s_1\sigma_1 \to_{\mathcal{P}} t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2 \to_{\mathcal{P}} t_2\sigma_2 \to^*_{\mathcal{R}} s_3\sigma_3 \to_{\mathcal{P}} t_3\sigma_3 \to^*_{\mathcal{R}} \cdots$$

then $\mathcal{R}' \subseteq_c \mathcal{R}$ and there is some k such that

$$s_k\sigma_k \to_{\mathcal{P}'} t_k\sigma_k \to^*_{\mathcal{R}'} s_{k+1}\sigma_{k+1} \to_{\mathcal{P}'} t_{k+1}\sigma_{k+1} \to^*_{\mathcal{R}'} s_{k+2}\sigma_{k+2} \to_{\mathcal{P}'} t_{k+2}\sigma_{k+2} \to^*_{\mathcal{R}'} \cdots$$

is an infinite $(\mathcal{P}',\mathcal{R}')$-chain.

Chain-identifying processors ensure that every minimal infinite chain of $(\mathcal{P},\mathcal{R})$ has an infinite tail where $\mathcal{R}^*$-steps can be replaced by $\mathcal{R}'^*$-steps and all pairs are from $\mathcal{P}'$. Note that every chain-identifying processor is sound. Moreover, several processors are indeed chain-identifying. Some examples are the reduction pair processor, the dependency graph processor, and all standard processors which just remove pairs and rules. The following lemma shows that chain-identifying processors can be used as extended processors via lift.

▶ **Lemma 4.9.** *1. If* $\mathcal{P}\mathrm{roc}$ *is sound, then* $\mathsf{lift}_0(\mathcal{P}\mathrm{roc})$ *is sound.*
2. If $\mathcal{P}\mathrm{roc}$ *is chain-identifying then* $\mathsf{lift}(\mathcal{P}\mathrm{roc})$ *is sound.*

Proof. Let $\mathcal{P}$, $\mathcal{R}$, $\mathcal{P}'$, and $\mathcal{R}'$ be given such that $\mathcal{P}\mathrm{roc}(\mathcal{P},\mathcal{R}) = (\mathcal{P}',\mathcal{R}')$.

1. We assume that $(\mathcal{P}',\mathcal{R}',0)$ is finite and have to show that $(\mathcal{P},\mathcal{R},n)$ is finite. By Lemma 4.5(**1**) and the assumption we know that $(\mathcal{P}',\mathcal{R}')$ is finite. Thus, also $(\mathcal{P},\mathcal{R})$ is finite using the soundness of $\mathcal{P}\mathrm{roc}$. By Lemma 4.5(**2**) we conclude finiteness of $(\mathcal{P},\mathcal{R},n)$.
2. Here, we may assume that $(\mathcal{P}',\mathcal{R}',n)$ is finite and have to show that $(\mathcal{P},\mathcal{R},n)$ is finite. We show finiteness of $(\mathcal{P},\mathcal{R},n)$ via contraposition. So, assume $(\mathcal{P},\mathcal{R},n)$ is not finite. This shows that there is an infinite $(\mathcal{P},\mathcal{R})$-chain

$$s_1\sigma_1 \to_{\mathcal{P}} t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2 \to_{\mathcal{P}} t_2\sigma_2 \to^*_{\mathcal{R}} s_3\sigma_3 \to_{\mathcal{P}} t_3\sigma_3 \to^*_{\mathcal{R}} \cdots$$

such that for all i we have $\forall m \leqslant n. \mathsf{SN}_{\mathcal{U}^m(\mathcal{R})}(\mathsf{unlab}^m(t_i\sigma_i))$. By choosing $m = 0$ we also have $\mathsf{SN}_{\mathcal{R}}(t_i\sigma_i)$ for all i. Hence, the chain is also a minimal infinite $(\mathcal{P}, \mathcal{R})$-chain. Since $\mathcal{P}\mathrm{roc}$ is chain-identifying we know that $\mathcal{R}' \subseteq_c \mathcal{R}$ and there is some k such that

$$s_k\sigma_k \to_{\mathcal{P}'} t_k\sigma_k \to^*_{\mathcal{R}'} s_{k+1}\sigma_{k+1} \to_{\mathcal{P}'} t_{k+1}\sigma_{k+1} \to^*_{\mathcal{R}'} s_{k+2}\sigma_{k+2} \to_{\mathcal{P}'} t_{k+2}\sigma_{k+2} \to^*_{\mathcal{R}'} \cdots$$

is an infinite $(\mathcal{P}', \mathcal{R}')$-chain. We continue to prove that for every i and every $m \leqslant n$ we have $\mathsf{SN}_{\mathcal{U}^m(\mathcal{R}')}(\mathsf{unlab}^m(t_i\sigma_i))$. This leads to the desired contradiction, since then we have shown that $(\mathcal{P}', \mathcal{R}', n)$ is not finite.
To prove $\mathsf{SN}_{\mathcal{U}^m(\mathcal{R}')}(\mathsf{unlab}^m(t_i\sigma_i))$ we first use minimality of the $(\mathcal{P}, \mathcal{R})$-chain to conclude $\mathsf{SN}_{\mathcal{U}^m(\mathcal{R})}(\mathsf{unlab}^m(t_i\sigma_i))$. Then the result immediately follows since the rewrite relation of $\mathcal{U}^m(\mathcal{R}')$ is a subset of the rewrite relation of $\mathcal{U}^m(\mathcal{R})$ by Lemma 3.17, **2** and **3**. ◀

Using these results allowed us to develop the first certified proof of the TRS in Example 4.2. We only had to change the given proof such that uncurrying [11] is used instead of the $\mathcal{A}$-transformation, since we have only formalized the former technique. The detailed proof is provided in the IsaFoR/CeTA repository.[6]

However, unlike for TRSs, root-labeling is not directly supported as root-labeling on DP problems [16, 17] is not a chain-identifying processor. Here again, root-labeling itself is not the problem, but making sure that the fixed algebra is a model of $\mathcal{R}$, which is again done by closing under flat contexts. In the DP framework we need the auxiliary function $\mathsf{block}_\triangle$, given by the equations $\mathsf{block}_\triangle(f(\vec{t_n})) = f(\triangle(\vec{t_n}))$ and $\mathsf{block}_\triangle(x) = x$.

▶ **Definition 4.10** (Flat Context Closure)**.** Let $(\mathcal{P}, \mathcal{R})$ be a DP problem such that $\mathcal{R}$ is left-linear and $\mathcal{F}$ is a superset of the signature of $\mathcal{R}$ combined with the non-root symbols of $\mathcal{P}$. Furthermore, let $\triangle$ be a function symbol not in $\mathcal{F}$. Then the closure under flat contexts of $(\mathcal{P}, \mathcal{R})$ is given by $\mathcal{FC}_{\mathcal{F}}(\mathcal{P}, \mathcal{R}) = (\mathsf{block}_\triangle(\mathcal{P}), \mathcal{FC}_{\{\triangle\}\cup\mathcal{F}}(\mathcal{R}))$.

As the pairs of a DP problem are modified, we do not get soundness of $\mathsf{lift}(\mathcal{FC}_{\mathcal{F}})$ via Lemma 4.9. Nevertheless, by using the definition of finiteness of extended DP problems and providing a manual proof one can show that $\mathsf{lift}(\mathcal{FC}_{\mathcal{F}})$ is indeed sound.

5 Problems in Certification

We present three problems that arose when trying to certify proofs with semantic labeling.

The first problem for the certifier is that internally it only works on extended termination/DP problems, whereas in the provided proofs just TRSs and DP problems are given without the additional numbers. However, this problem is fixed by computing the number during certification. This is easy and seems to be a safe solution: the format for termination proofs remains unchanged, and so far no termination proof was refused with the reason that the internal computation of the number was wrong.

The second and third problem are concerned with how semantic labeling is applied, since usually variations of Lemma 3.9 and Theorem 4.1 are used in termination provers.

The second problem occurs for TRSs as well as DP problems. The theory about semantic labeling demands that $\mathcal{D}\mathrm{ec}$ is added to the new TRS when using quasi-models. However, termination provers typically reduce the set of rules and "optimize" semantic labeling by only adding rules $\mathcal{D}\mathrm{ec}'$ such that $\to_{\mathcal{D}\mathrm{ec}} \subseteq \to^+_{\mathcal{D}\mathrm{ec}'}$.

[6] `http://cl2-informatik.uibk.ac.at/rewriting/mercurial.cgi/IsaFoR/raw-file/v1.16/examples/secret_07_trs_4_top.proof.xml`

For example, if $L_f = \{0,1,2\}$ and the order is the standard order on the naturals, then $\mathcal{D}\text{ec} = \{f_2(x) \to f_1(x), f_1(x) \to f_0(x), f_2(x) \to f_0(x)\}$. However, the last rule is often omitted since it can be simulated by the previous two rules. To certify these termination proofs, we fist need to show that we may safely replace $\mathcal{D}\text{ec}$ by any $\mathcal{D}\text{ec}'$ where $\to_{\mathcal{D}\text{ec}} \subseteq \to^{+}_{\mathcal{D}\text{ec}'}$. Moreover, we have to provide a certified algorithm which for a given TRS $\mathcal{D}\text{ec}'$ and a given order can ensure that the condition $\to_{\mathcal{D}\text{ec}} \subseteq \to^{+}_{\mathcal{D}\text{ec}'}$ is satisfied. Furthermore, the algorithm should accept *all* $\mathcal{D}\text{ec}'$ where the condition is satisfied.

The third problem only occurs when dealing with quasi-models in the DP framework. Note that in standard DP problems the roots of $\mathcal{P}$ are special symbols (tuple symbols) which do not occur in the remaining DP problem. However, when applying Theorem 4.1 as it is, this invariant is destroyed since the decreasing rules for tuple symbols are added as new rules. We illustrate the problem and two possible solutions in the following example.

▶ **Example 5.1.** Consider a DP problem $(\mathcal{P},\mathcal{R})$ where $\mathsf{F}(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}(x,\mathsf{b}) \in \mathcal{P}$ and b is not defined in $\mathcal{R}$. Then, the dependency graph estimation EDG [1] can detect that there is no connection from the mentioned pair to itself. However, when performing labeling with a quasi-model where $\mathsf{s}(x)$ is interpreted as $\min(x+1,2)$ and where $\ell_{\mathsf{F}}(x,y) = x$ then for the mentioned pair we get all three rules $\{6,8,10\}$ in the labeled pairs $\mathcal{P}'$ and the decreasing rules for F are $\mathcal{D}\text{ec}_{\mathsf{F}} = \{7,9,11\}$.

$$\begin{array}{llllll}
6: & \mathsf{F}_2(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_2(x,\mathsf{b}) & 8: & \mathsf{F}_2(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_1(x,\mathsf{b}) & 10: & \mathsf{F}_1(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_0(x,\mathsf{b}) \\
7: & \mathsf{F}_2(x,y) \to \mathsf{F}_1(x,y) & 9: & \mathsf{F}_2(x,y) \to \mathsf{F}_0(x,y) & 11: & \mathsf{F}_1(x,y) \to \mathsf{F}_0(x,y)
\end{array}$$

Note that when adding $\mathcal{D}\text{ec}_{\mathsf{F}}$ as new rules, then the EDG contains an edge from $\mathsf{F}_2(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_1(x,\mathsf{b})$ to all other pairs since F_1 is defined in $\mathcal{D}\text{ec}_{\mathsf{F}}$. Hence, this is not the preferred way to add decreasing rules: not even the decrease in the labels is recognized.

One solution is to add $\mathcal{D}\text{ec}_{\mathsf{F}}$ as new pairs. Then one obtains a standard DP problem and the decrease in the labels is reflected in the EDG. But there still is a path from $\mathsf{F}_2(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_2(x,\mathsf{b})$ to $\mathsf{F}_1(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_0(x,\mathsf{b})$ via the pair $\mathsf{F}_2(x,y) \to \mathsf{F}_1(x,y)$, since the information that the second argument of F_n is b is lost when passing the pair $\mathsf{F}_2(x,y) \to \mathsf{F}_1(x,y)$.

To encounter this problem, there is another solution where $\mathcal{D}\text{ec}_{\mathsf{F}}$ is not produced at all, but where the labels of all tuple-symbols in right-hand sides of $\mathcal{P}'$ are decreased. In this example, one would have to add the additional pair $\mathsf{F}_2(\mathsf{s}(x),\mathsf{a}) \to \mathsf{F}_0(x,\mathsf{b})$ to $\mathcal{P}'$.

Hence, termination proofs might have used one of the two variants instead of Theorem 4.1. Here, the first variant returns the problem $(\mathsf{lab}(\mathcal{P}) \cup \mathcal{D}\text{ec}_{\mathcal{F}^\sharp}, \mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\text{ec}_{\mathcal{F}})$ and the second variant returns $(\mathsf{lab}(\mathcal{P})^{\geq}, \mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\text{ec}_{\mathcal{F}})$ where $\mathcal{D}\text{ec}_{\mathcal{F}^\sharp}$ are the decreasing rules for all tuple symbols, $\mathcal{D}\text{ec}_{\mathcal{F}}$ are the decreasing rules for the remaining symbols, and $\mathsf{lab}(\mathcal{P})^{\geq} = \{s \to f_{\ell'}(\vec{t}) \mid s \to f_{\ell}(\vec{t}) \in \mathsf{lab}(\mathcal{P}), \ell \geq_{L_f} \ell'\}$.

To certify these termination proofs the problem was mainly in formalizing that these variants of Theorem 4.1 are indeed sound.

▶ **Theorem 5.2.** *Both variants of Theorem 4.1 are sound, provided that they are applied on DP problems $(\mathcal{P},\mathcal{R})$ where neither left- nor right-hand sides of $\mathcal{P}$ are variables and the roots of $\mathcal{P}$ are distinguished tuple symbols which do not occur in the remaining DP problem.*

We shortly describe the proof idea. The main problem is that we cannot w.l.o.g. restrict the substitutions in a chain such that they do not contain tuple symbols [17]. Thus, we may have to apply rules in $\mathcal{D}\text{ec}_{\mathcal{F}^\sharp}$ also below the root, in order to simulate a reduction $t_i\sigma_i \to^{*}_{\mathcal{R}} s_{i+1}\sigma_{i+1}$. The trick is to introduce a second set of labels and labeling functions for the tuple symbols. The new labeling functions label all tuple symbols by the same element.

Hence, no decreasing rules are required for them (w.r.t. the second set of labeling functions) and on all other symbols the labeling functions coincide.

Afterwards, we use a combined labeling of terms: The root of the term is labeled according to the original function, and below the root it is labeled w.r.t. the second labeling function. In this way no decreasing rules for the tuple symbols have to be applied below the root and moreover, on all terms in the DP problem, the original and the combined labeling produce the same result. Thus, we can transform a given $(\mathcal{P}, \mathcal{R})$-chain into a $(\mathsf{lab}(\mathcal{P}) \cup \mathcal{D}\mathrm{ec}_{\mathcal{F}^\sharp}, \mathsf{lab}(\mathcal{R}) \cup \mathcal{D}\mathrm{ec}_{\mathcal{F}})$-chain. Theorem 5.2 easily follows.

To summarize, we discussed some problems which occurred when trying to certify existing proofs which are mainly due to optimizations of the basic semantic labeling theorems. Of course, we also need to check the model condition, whether the orders are weakly monotone when using quasi-orders, etc. Whereas the general theorems about soundness of semantic labeling have been formalized for arbitrary carriers, for the certification we currently only support finite carriers. Then checking the required conditions is performed via enumerating all possible assignments.

In total, our formalization of pure semantic labeling consists of 3300 lines of Isabelle, where roughly half of it is about semantic labeling on generic algebras, and the other half contains executable functions for the certifier using algebras over finite carriers and soundness proofs for these functions. Moreover, the theory about the semantic labeling framework with extended termination techniques, extended DP problems, etc., consists of another 1000 lines.

6 Experiments

To test the impact of our formalization we ran AProVE on the TPDB (version 8.0), considering all 2795 TRSs. We used two different strategies which are similar to the strategy CERT that was used during the 2010 termination competition in the certified termination category: −SL is like CERT but with semantic labeling removed, and +SL is like CERT including all three variants of semantic labeling that are supported by AProVE (root-labeling, semantic-labeling on finite carriers with models and quasi-models).

We performed all our experiments on a machine with two 2.8 GHz Quad-Core Intel Xeon processors and 6 GB of main memory. The following results where obtained using a 60 seconds timeout.

	−SL	+SL	total
termination proofs	1137	1207	1227
nontermination proofs	225	218	227
total time (in minutes)	1186	1219	
certification time (in minutes)	1	3	

CeTA (version 1.17) certified all but two proofs. On one TRS, both −SL and +SL delivered a faulty proof, caused by a bug in the LPO output of AProVE (which will be fixed soonish).

The results show that by using semantic labeling we obtain 90 new certified termination proofs. This is an increase of nearly 8 %. Note that +SL has not solved all TRSs where −SL was successful. This is due to timing issues in the strategy.

7 Conclusion

During our formalization of semantic labeling we have detected that unlabeling is unsound when using the current semantics of termination problems. We solved the problem by extending termination problems and the DP framework such that recursive labeling and unlabeling are supported, as well as all other existing termination techniques. This framework forms the semantic basis of our certifier CeTA which now fully supports semantic labeling.

Acknowledgments We thank Christian Kuknat and Carsten Fuhs for their support in providing certifiable proofs with semantic labeling generated by AProVE.

References

1 T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theor. Comput. Sci.*, 236(1-2):133–178, 2000.

2 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge Univ. Press, 1999.

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

4 É. Contejean, A. Paskevich, X. Urbain, P. Courtieu, O. Pons, and J. Forest. A3PAT, an approach for certified automated termination proofs. In *PEPM*, pages 63–72, 2010.

5 J. Endrullis. Jambox. Available at `http://joerg.endrullis.de`.

6 J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *J. Autom. Reasoning*, 40(2-3):195–220, 2008.

7 A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Inf. Comput.*, 205(4):512–534, 2007.

8 J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *J. Autom. Reasoning*, 37(3):155–203, 2006.

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

10 J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *IJCAR*, LNAI 4130, pages 281–286, 2006.

11 N. Hirokawa, A. Middeldorp, and H. Zankl. Uncurrying for termination. In *LPAR*, LNAI 5330, pages 667–681, 2008.

12 D. Hofbauer and J. Waldmann. Match-bounds for relative termination. In *WST*, 2010.

13 A. Koprowski. TPA: Termination proved automatically. In *RTA*, LNCS 4098, 2006.

14 M. Korp and A. Middeldorp. Match-bounds revisited. *Inf. Comput.*, 207(11):1259–1283, 2009.

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

16 C. Sternagel and A. Middeldorp. Root-labeling. In *RTA*, LNCS 5117, pages 336–350, 2008.

17 C. Sternagel and R. Thiemann. Signature extensions preserve termination. In *CSL*, LNCS 6247, pages 514–528, 2010.

18 R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *TPHOLs*, LNCS 5674, pages 452–468, 2009.

19 H. Zantema. Termination of term rewriting by semantic labelling. *Fundam. Inform.*, 24(1-2):89–105, 1995.

20 H. Zantema. Termination of string rewriting proved automatically. *J. Autom. Reasoning*, 34(2):105–139, 2005.