Generalized and Formalized Uncurrying*

Christian Sternagel and René Thiemann

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

Abstract Uncurrying is a termination technique for applicative term rewrite systems. During our formalization of uncurrying in the theorem prover Isabelle, we detected a gap in the original pen-and-paper proof which cannot directly be filled without further preconditions. Our final formalization does not demand additional preconditions, and generalizes the existing techniques since it allows to uncurry non-applicative term rewrite systems. Furthermore, we provide new results on uncurrying for relative termination and for dependency pairs.

Keywords: uncurrying, termination, formalization, interactive theorem proving, dependency pairs, term rewriting

1 Introduction

In recent years, the way to prove termination of term rewrite systems (TRSs) has changed. Current termination tools no longer search for a single reduction order containing the rewrite relation. Instead, they combine various termination techniques in a modular way, resulting in large and tree-like termination proofs, where at each node a specific technique is applied.

On the one hand, this combination makes termination tools more powerful. On the other hand, it makes them more complex and error-prone. It is regularly demonstrated that we cannot blindly trust the output of termination provers. Every now and then, some prover delivers a faulty proof. Often, this is only detected if there is another prover giving a contradictory answer for the same input, as a manual inspection of proofs is infeasible due to their size.

The problem is solved by combining two systems. For a given TRS, we first use a termination tool to automatically detect a termination proof (which may contain errors). Then, we use a highly trusted certifier which checks whether the detected proof is indeed correct. In total, the combination yields a powerful and trustable workflow to prove termination.

To obtain a highly trustable certifier, a common approach is to first formalize the desired termination techniques once and for all (thereby ensuring their soundness) and then, for a given proof, check that the used techniques are applied correctly [2,3,14]. We formalized the dependency pair framework (DP framework) [5] and many termination techniques in our Isabelle/HOL [11] library IsaFoR [14]

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

(in the remainder we just write *Isabelle*, instead of *Isabelle/HOL*). From IsaFoR, we code-extract CeTA, an automatic certifier for termination proofs.

In this paper, we present one of the latest additions to IsaFoR: the formalization of uncurrying, as described in [8]. However, we did not only formalize uncurrying, but also generalized it. Furthermore, we found a gap in one of the original proofs, which we could fortunately close.

Note that all the proofs that are presented (or omitted) in the following, have been formalized as part of IsaFoR. Hence, in this 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. 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 Sect. 2, we shortly recapitulate some required notions of term rewriting. Afterwards, in Sect. 3, we describe applicative rewriting, give an overview of approaches using *uncurrying* for proving termination, and present our generalization of uncurrying for TRSs. Then, in Sect. 4, we show how to lift uncurrying to the DP framework. We present heuristics and our experiments in Sect. 5, before we conclude in Sect. 6.

2 Preliminaries

We assume familiarity with term rewriting [1]. 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 a function symbol $f \in \mathcal{F}$ applied to argument terms $f(t_1, \ldots, t_n)$ where the *arity* of f is $\mathsf{ar}(f) = n$. 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 rewrite rules. The set of *defined symbols* (of $\mathcal{R}$) is $\mathcal{D}_\mathcal{R} = \{f \mid f(\ldots) \to r \in \mathcal{R}\}$. 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 rewrite rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$. A term t is *root-stable w.r.t.* $\mathcal{R}$ iff there is no derivation $t \to^*_\mathcal{R} \ell\sigma$ for some $\ell \to r \in \mathcal{R}$ and substitution σ.

We say that a term t is *terminating w.r.t.* $\mathcal{R}$ ($\mathsf{SN}_\mathcal{R}(t)$) if it cannot start an infinite derivation $t = t_1 \to_\mathcal{R} t_2 \to_\mathcal{R} t_3 \to_\mathcal{R} \cdots$. A TRS is terminating ($\mathsf{SN}(\mathcal{R})$) iff all terms are terminating w.r.t. $\mathcal{R}$. A TRS $\mathcal{R}$ is *terminating relative to* a TRS $\mathcal{S}$ iff there is no infinite $\mathcal{R} \cup \mathcal{S}$-derivation with infinitely many $\mathcal{R}$-steps.

3 Applicative Rewriting and Uncurrying

An applicative term rewrite system (ATRS) is a TRS over an *applicative signature* $\mathcal{F} = \{\circ\} \cup \mathcal{C}$, where $\circ$ is a unique binary symbol (the application symbol) and all symbols in $\mathcal{C}$ are constants. ATRSs can be used to encode many higher-order functions without explicit abstraction as first-order TRSs. In the remainder we

use $\circ$ as an infix-symbol which associates to the left ($s \circ t \circ u = (s \circ t) \circ u$). In examples we omit $\circ$ whenever this increases readability.

Example 1. The following ATRS $\mathcal{R}$ (a variant of [8, Example 7], replacing addition by subtraction) contains the map function (which applies a function to all arguments of a list) and subtraction on Peano numbers in applicative form.

$$
\begin{array}{lll}
1: & \mathsf{sub}\ \mathsf{0} \to \mathsf{K}\ \mathsf{0} & 5: \quad \mathsf{K}\ x\ y \to x \\
2: & \mathsf{sub}\ x\ \mathsf{0} \to x & 6: \quad \mathsf{map}\ z\ \mathsf{nil} \to \mathsf{nil} \\
3: & \mathsf{sub}\ x\ x \to \mathsf{0} & 7: \quad \mathsf{map}\ z\ (\mathsf{cons}\ x\ xs) \to \mathsf{cons}\ (z\ x)\ (\mathsf{map}\ z\ xs) \\
4: & \mathsf{sub}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) \to \mathsf{sub}\ x\ y &
\end{array}
$$

Proving termination of ATRSs is challenging without dedicated termination techniques (e.g., for reduction orders, we cannot interpret $\mathsf{sub} \circ x \circ y$ as x, since sub is a constant and not binary).

Until now, there have at least been three approaches to tackle this problem. All of them try to uncurry a TRS such that, for example, Rule 4 from above becomes $\mathsf{sub}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{sub}(x, y)$.

To distinguish the three approaches, we need the following definitions:

Definition 2. *A term t is* head variable free *iff t does not contain a subterm of the form $x \circ s$ where x is a variable. The* applicative arity *of a constant f in an ATRS $\mathcal{R}$ ($\mathsf{aa}_{\mathcal{R}}(f)$) is the maximal number n, such that $f \circ t_1 \circ \cdots \circ t_n$ occurs as a subterm in $\mathcal{R}$.* Uncurrying *an application $f \circ t_1 \circ \cdots \circ t_n$ with $\mathsf{aa}_{\mathcal{R}}(f) = n$ yields the term $f(t_1, \ldots, t_n)$. A term t is* proper w.r.t. $\mathsf{aa}_{\mathcal{R}}$ *iff t is a variable or $t = f \circ t_1 \circ \cdots \circ t_n$ where $\mathsf{aa}_{\mathcal{R}}(f) = n$ and each t_i is proper.*

The oldest of the three approaches is from [9]. It requires that all terms in a TRS are proper w.r.t. $\mathsf{aa}_{\mathcal{R}}$, and shows that then termination of $\mathcal{R}$ is equivalent to termination of the TRS obtained by uncurrying all terms of $\mathcal{R}$. Since proper terms do not contain any partial applications, the application symbol is completely eliminated by uncurrying. However, requiring proper terms is rather restrictive: Essentially, it is demanded that the TRS under consideration is a standard first-order TRS which is just written in applicative form. For example, the approach is not applicable to Example 1, since there is a head variable in the right-hand side of Rule 7 ($z \circ x$) and sub as well as K are applied to a single argument in Rule 1, even though $\mathsf{aa}_{\mathcal{R}}(\mathsf{sub}) = 2$ and $\mathsf{aa}_{\mathcal{R}}(\mathsf{K}) = 2$.

The second approach was given in [6,13]. Here, the same preconditions as in [9] apply, but the results are extended to innermost rewriting and to the DP framework. The latter has the advantage, that only the current subproblem has to satisfy the preconditions. For example, when treating the dependency pair

$$\mathsf{map} \circ z \circ^{\sharp} (\mathsf{cons} \circ x \circ xs) \to \mathsf{map} \circ z \circ^{\sharp} xs \tag{1}$$

for the recursive call of map, we can perform uncurrying (since there are no usable rules and (1) satisfies the preconditions). Moreover, in [13] uncurrying is combined with the reduction pair processor to further relax the preconditions.

The third approach is given in [8]. Here, the preconditions for uncurrying have been reduced drastically as only the left-hand sides of the TRS $\mathcal{R}$ must be head variable free. In return, we have to η-saturate $\mathcal{R}$ and add uncurrying rules. Moreover, for each constant f with $\mathsf{aa}_{\mathcal{R}}(f) = n$ we obtain n new function symbols $f_1, \ldots, f_n$ of arities $1, 2, \ldots, n$ which handle partial applications.

Example 3. When η-saturating the TRS $\mathcal{R}$ of Example 1, we have to add the rule $\mathsf{sub} \circ \mathsf{0} \circ y \to \mathsf{K} \circ \mathsf{0} \circ y$. The uncurried TRS consists of the following rules:

$$
\begin{array}{rlrl}
8: & \mathsf{sub}_1(\mathsf{0}) \to \mathsf{K}_1(\mathsf{0}) & 12: & \mathsf{sub}_2(\mathsf{s}_1(x), \mathsf{s}_1(y)) \to \mathsf{sub}_2(x, y) \\
9: & \mathsf{sub}_2(\mathsf{0}, y) \to \mathsf{K}_2(\mathsf{0}, y) & 13: & \mathsf{K}_2(x, y) \to x \\
10: & \mathsf{sub}_2(x, \mathsf{0}) \to x & 14: & \mathsf{map}_2(z, \mathsf{nil}) \to \mathsf{nil} \\
11: & \mathsf{sub}_2(x, x) \to \mathsf{0} & 15: & \mathsf{map}_2(z, \mathsf{cons}_2(x, xs)) \to \mathsf{cons}_2(z \circ x, \mathsf{map}_2(z, xs))
\end{array}
$$

Moreover, we have to add the following uncurrying rules:

$$
\begin{array}{rlrl}
16: & \mathsf{s} \circ x \to \mathsf{s}_1(x) & 21: & \mathsf{cons} \circ x \to \mathsf{cons}_1(x) \\
17: & \mathsf{K} \circ x \to \mathsf{K}_1(x) & 22: & \mathsf{cons}_1(x) \circ y \to \mathsf{cons}_2(x, y) \\
18: & \mathsf{K}_1(x) \circ y \to \mathsf{K}_2(x, y) & 23: & \mathsf{map} \circ x \to \mathsf{map}_1(x) \\
19: & \mathsf{sub} \circ x \to \mathsf{sub}_1(x) & 24: & \mathsf{map}_1(x) \circ y \to \mathsf{map}_2(x, y) \\
20: & \mathsf{sub}_1(x) \circ y \to \mathsf{sub}_2(x, y) &&
\end{array}
$$

Also [8] gives an extension to the DP framework.

To summarize, the traditional technique of uncurrying of [9] is completely subsumed by [6,13], but [6,13] and [8] are incomparable. The advantage of [6,13] is that the generated TRSs and DP problems are smaller, and that uncurrying is also available in a combination with the reduction pair processor, whereas [8] supports head variables (see [13, Chap. 6] for a more detailed comparison).

Since [8] is used in more termination tools (it is used in at least Jambox [4] and TTT2 [10] whereas we only know of AProVE [7] that implements all uncurrying techniques of [6,13]), we incorporated the techniques of [8] in our certifier CeTA.

During our formalization we have

- detected a gap in a proof of [8] which could not directly be closed without adding further preconditions to one of the main results,
- generalized the technique of uncurrying which now entails the result of [8] even without adding any additional precondition, and
- generalized the technique of *freezing* from [8].

The structure in [8] is as follows. First, uncurrying is developed for TRSs over applicative signatures $\{\circ\} \cup \mathcal{C}$. Then, it is extended to DP problems, introducing a second application symbol $\circ^\sharp$ that may only occur at root-positions of $\mathcal{P}$ and is not uncurried at all. Finally, *freezing* is applied to uncurry applications of $\circ^\sharp$.

Following this structure, we first fully formalized uncurrying on TRSs. However, in the extension to DP problems there is a missing step which is illustrated in more detail in Example 14 on page 9. The main problem is that signature restrictions on DP problems are in general unsound.

To fill the gap, one option is to use the results of [12] about signature restrictions, which can however only be applied if $\mathcal{R}$ is left-linear. This clearly weakens the applicability of uncurrying, e.g., Example 1 is not left-linear.

Alternatively, one can try to perform uncurrying without restricting to applicative signatures. This is what we did. All uncurrying techniques that we formalized work on terms and TRSs over arbitrary signatures.

The major complication is the increase of complexity in the cases that have to be considered. For example, using an applicative signature, we can assume that every term is of the form $x \circ t_1 \circ \cdots \circ t_n$ or $f \circ t_1 \circ \cdots \circ t_n$ where $n \in \mathbb{N}$, x is a variable, and f is a constant. Generalizing this to arbitrary signatures we have to consider the two cases $x \circ t_1 \circ \cdots \circ t_n$ and $f(s_1, \ldots, s_m) \circ t_1 \circ \cdots \circ t_n$ instead, where f is an m-ary symbol. Hence, when considering a possible rewrite step, we also have to consider the new case that the step is performed in some s_i.

We not only generalized uncurrying to work for arbitrary signatures and relative rewriting, but also to a free choice of the applicative arity $\mathsf{aa}(f)$. This is in contrast to [8], where the applicative arity is fixed by Definition 2. We will elaborate on this difference after presenting our main theorem.

Definition 4 (Symbol maps and applicative arity). *Let $\mathcal{F}$ be a signature. A* symbol map *is a mapping $\pi : \mathcal{F} \to [\mathcal{F}]$ from symbols to non-empty lists of symbols. It is* injective *if for all f and g, $\pi(f)$ contains no duplicates, $\pi(f)$ does not contain $\circ$, and whenever $f \neq g$ then $\pi(f)$ and $\pi(g)$ do not share symbols. If $\pi(f) = [f_0, \ldots, f_n]$, then the* applicative arity of f w.r.t. π *is $\mathsf{aa}_\pi(f) = n$. The* applicative arity of a term *is defined by $\mathsf{aa}_\pi(t) = \mathsf{aa}_\pi(f) \dot{-} n$, where $x \dot{-} y = \max(x - y, 0)$, for $t = f(s_1, \ldots, s_m) \circ t_1 \circ \cdots \circ t_n$ and is undefined otherwise.*

Intuitively, if $\pi(f) = [f_0, \ldots, f_n]$ then every application of f on $i \leqslant n$ arguments $t_1, \ldots, t_i$ will be fully uncurried to $f_i(t_1, \ldots, t_i)$. If more than n arguments are applied, then we obtain $f_n(t_1, \ldots, t_n) \circ t_{n+1} \circ \ldots \circ t_i$. A symbol map containing an entry for f, uniquely determines the applicative arity n as well as the names of the (partial) applications $f_0, \ldots, f_n$ of f.

In the following we assume a fixed symbol map π and just write $\mathsf{aa}(f)$ and $\mathsf{aa}(t)$ instead of $\mathsf{aa}_\pi(f)$ and $\mathsf{aa}_\pi(t)$, respectively. Additionally, we assume that $\pi(f) = [f_0, \ldots, f_{\mathsf{aa}(f)}]$ where in examples we write f instead of f_0. Now we can define the uncurrying TRS w.r.t. π.

Definition 5. *The* uncurrying TRS $\mathcal{U}$ *contains the rule*

$$f_k(x_1, \ldots, x_m, y_1, \ldots, y_k) \circ y_{k+1} \to f_{k+1}(x_1, \ldots, x_m, y_1, \ldots, y_{k+1})$$

for every $f \in \mathcal{F}$ with $\mathsf{ar}(f) = m$ and $\mathsf{aa}(f) = n$, and every $k < n$. The variables $x_1, \ldots, x_m, y_1, \ldots, y_{k+1}$ are pairwise distinct.

In [8], terms are uncurried by computing the unique normal form w.r.t. $\mathcal{U}$. For our formalization we instead used the upcoming uncurrying function for the following two reasons: First, we do not have any results about confluence of TRSs. Hence, to even define *the* normal form w.r.t. $\mathcal{U}$ would require to formalize several additional lemmas which show that every term has exactly one normal form. This would be quite some effort which we prefer to avoid. The second reason is efficiency. When certifying the application of uncurrying in large termination

proofs, we have to compute the uncurried version of a term. It is just more efficient to use a function which performs uncurrying directly, than to compute a normal form w.r.t. a TRS where possible redexes have to be searched, etc.

Definition 6. *The uncurrying function* $\llcorner\cdot\lrcorner$ *on terms is defined as*

- $\llcorner x \circ t_1 \circ \cdots \circ t_n \lrcorner = x \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner$
- $\llcorner f(s_1, \ldots, s_m) \circ t_1 \circ \cdots \circ t_n \lrcorner = f_k(\llcorner s_1 \lrcorner, \ldots, \llcorner s_m \lrcorner, \llcorner t_1 \lrcorner, \ldots, \llcorner t_k \lrcorner) \circ \llcorner t_{k+1} \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner$ *where* $k = \min(n, \mathsf{aa}(f))$

It is homomorphically extended to operate on rules, TRSs, and substitutions.

We establish the link between $\mathcal{U}$ and $\llcorner\cdot\lrcorner$ in the following lemma which generalizes the corresponding results in [8].

Lemma 7.

- *if* $k < \mathsf{aa}(f)$ *and* $\mathsf{ar}(f) = m$ *then* $f_k(s_1, \ldots, s_{m+k}) \circ t \to_{\mathcal{U}} f_{k+1}(s_1, \ldots, s_{m+k}, t)$
- *if* $k + n \leqslant \mathsf{aa}(f)$ *and* $\mathsf{ar}(f) = m$ *then* $f_k(s_1, \ldots, s_{m+k}) \circ t_1 \circ \ldots \circ t_n \to^*_{\mathcal{U}} f_{k+n}(s_1, \ldots, s_{m+k}, t_1, \ldots, t_n)$
- $\llcorner s \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \to^*_{\mathcal{U}} \llcorner s \circ t_1 \circ \cdots \circ t_n \lrcorner$
- *if* $\mathsf{aa}(s) = 0$ *or* $\mathsf{aa}(s)$ *is undefined then* $\llcorner s \circ t_1 \circ \cdots \circ t_n \lrcorner = \llcorner s \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner$
- $\llcorner s \lrcorner \cdot \llcorner \sigma \lrcorner \to^*_{\mathcal{U}} \llcorner s \cdot \sigma \lrcorner$
- *if* t *is head variable free then* $\llcorner s \cdot \sigma \lrcorner = \llcorner s \lrcorner \cdot \llcorner \sigma \lrcorner$

The last two results show how uncurrying can be exchanged with applying substitutions. As we will often need the equality $\llcorner \ell \cdot \sigma \lrcorner = \llcorner \ell \lrcorner \cdot \llcorner \sigma \lrcorner$ for left-hand sides ℓ, it is naturally to demand that left-hand sides are head variable free.

Definition 8. *A TRS is* left head variable free *if all left-hand sides are head variable free.*

Termination of $\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}$ does not suffice to conclude termination of $\mathcal{R}$, cf. [8, Example 13]. The reason is that first we have to η-saturate $\mathcal{R}$.

Definition 9. *A TRS* $\mathcal{R}$ *is* η-closed *iff for every rule* $\ell \to r$ *with* $\mathsf{aa}(\ell) > 0$ *there is a rule* $\ell \circ x \to r \circ x \in \mathcal{R}$ *where* x *is fresh w.r.t.* $\ell \to r$*. The* η-saturation $\mathcal{R}_\eta$ *of* $\mathcal{R}$ *is obtained by adding new rules* $\ell \circ x \to r \circ x$ *until the result is* η*-closed.*

The upcoming theorem is the key to use uncurrying for termination proofs. It allows to simulate one $\mathcal{R}$-step by many steps in the uncurried system.

Theorem 10. *Let* $\mathcal{R}$ *be* η*-closed and left head variable free. Let there be no left-hand side of* $\mathcal{R}$ *which is a variable. If* $s \to_{\mathcal{R}} t$ *then* $\llcorner s \lrcorner \to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} \llcorner t \lrcorner$*.*

Proof. Let $s = C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma] = t$ where $\ell \to r \in \mathcal{R}$. We show $\llcorner s \lrcorner \to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} \llcorner t \lrcorner$ by induction on the size of C.

– If $C = f(s_1, \ldots, D, \ldots, s_m) \circ t_1 \circ \cdots \circ t_n$ for some $f \neq \circ$ then by the induction hypothesis we know that $\llcorner D[\ell\sigma] \lrcorner \to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} \llcorner D[r\sigma] \lrcorner$. Moreover,

$$\begin{aligned}
\llcorner s \lrcorner &= \llcorner f(s_1, \ldots, D[\ell\sigma], \ldots, s_m) \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= f_k(\llcorner s_1 \lrcorner, \ldots, \llcorner D[\ell\sigma] \lrcorner, \ldots, \llcorner s_m \lrcorner, \llcorner t_1 \lrcorner, \ldots, \llcorner t_k \lrcorner) \circ \llcorner t_{k+1} \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&\to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} f_k(\ldots, \llcorner D[r\sigma] \lrcorner, \ldots, \llcorner s_m \lrcorner, \llcorner t_1 \lrcorner, \ldots, \llcorner t_k \lrcorner) \circ \llcorner t_{k+1} \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&= \llcorner f(s_1, \ldots, D[r\sigma], \ldots, s_m) \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= \llcorner t \lrcorner
\end{aligned}$$

where $k = \min(n, \mathsf{aa}(f))$.

– If $C = t_0 \circ D \circ t_1 \circ \cdots \circ t_n$ then by the induction hypothesis we know that $\llcorner D[\ell\sigma] \lrcorner \to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} \llcorner D[r\sigma] \lrcorner$. If $\mathsf{aa}(t_0) = 0$ or $\mathsf{aa}(t_0)$ is undefined then

$$\begin{aligned}
\llcorner s \lrcorner &= \llcorner t_0 \circ D[\ell\sigma] \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= \llcorner t_0 \lrcorner \circ \llcorner D[\ell\sigma] \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&\to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} \llcorner t_0 \lrcorner \circ \llcorner D[r\sigma] \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&= \llcorner t_0 \circ D[r\sigma] \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= \llcorner t \lrcorner
\end{aligned}$$

Otherwise, $\mathsf{aa}(t_0) > 0$ and hence, $t_0 = f(s_1, \ldots, s_m) \circ s_{m+1} \circ \cdots \circ s_{m+k}$ where $k < \mathsf{aa}(f)$. It follows that

$$\begin{aligned}
\llcorner s \lrcorner &= \llcorner f(s_1, \ldots, s_m) \circ s_{m+1} \circ \cdots \circ s_{m+k} \circ D[\ell\sigma] \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= f_{k+1+n'}(\ldots, \llcorner s_{m+k} \lrcorner, \llcorner D[\ell\sigma] \lrcorner, \llcorner t_1 \lrcorner, \ldots, \llcorner t_{n'} \lrcorner) \circ \llcorner t_{n'+1} \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&\to^+_{\llcorner \mathcal{R} \lrcorner \cup \mathcal{U}} f_{k+1+n'}(\ldots, \llcorner s_{m+k} \lrcorner, \llcorner D[r\sigma] \lrcorner, \llcorner t_1 \lrcorner, \ldots, \llcorner t_{n'} \lrcorner) \circ \llcorner t_{n'+1} \lrcorner \circ \cdots \\
&= \llcorner f(s_1, \ldots, s_m) \circ s_{m+1} \circ \cdots \circ s_{m+k} \circ D[r\sigma] \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= \llcorner t \lrcorner
\end{aligned}$$

where $n' = \min(\mathsf{aa}(f) - k - 1, n)$.

– If $C = \Box \circ t_1 \circ \cdots \circ t_n$ and $n = 0 \vee \mathsf{aa}(\ell) = 0$ then

$$\begin{aligned}
\llcorner s \lrcorner &= \llcorner \ell \cdot \sigma \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= \llcorner \ell \cdot \sigma \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&= \llcorner \ell \lrcorner \cdot \llcorner \sigma \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&\to_{\llcorner \mathcal{R} \lrcorner} \llcorner r \lrcorner \cdot \llcorner \sigma \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&\to^*_{\mathcal{U}} \llcorner r \cdot \sigma \lrcorner \circ \llcorner t_1 \lrcorner \circ \cdots \circ \llcorner t_n \lrcorner \\
&\to^*_{\mathcal{U}} \llcorner r \cdot \sigma \circ t_1 \circ \cdots \circ t_n \lrcorner \\
&= \llcorner t \lrcorner
\end{aligned}$$

since ℓ is head variable free and if $n \neq 0$ then $\mathsf{aa}(\ell\sigma) = \mathsf{aa}(\ell) = 0$.

– If $C = \Box \circ t_1 \circ \cdots \circ t_n$ with $n > 0$ and $\mathsf{aa}(\ell) > 0$ then $\ell' \to r' = \ell \circ x \to r \circ x \in \mathcal{R}$ since $\mathcal{R}$ is η-closed. Moreover, by changing σ to $\delta = \sigma \uplus \{x/t_1\}$ we achieve $s = \ell \cdot \sigma \circ t_1 \circ \cdots \circ t_n = \ell'\delta \circ t_2 \circ \cdots \circ t_n = D[\ell'\delta]$ and $r = r \cdot \sigma \circ t_1 \circ \cdots \circ t_n = r'\delta \circ t_2 \circ \cdots \circ t_n = D[r'\delta]$ for the context $D = \Box \circ t_2 \circ \cdots t_n$ which is strictly smaller than C. Hence, the result follows by the induction hypothesis.

– If $C = \Box \circ t_1 \circ \cdots \circ t_n$ with $n > 0$ and $\mathsf{aa}(\ell)$ is undefined then $\ell = x \circ \ell_1 \circ \cdots \circ \ell_k$ with $k \geq 0$. But if $k > 0$ then ℓ is not head variable free and if $k = 0$ then $\mathcal{R}$ contains a variable as left-hand side. In both cases we get a contradiction to the preconditions in the theorem. □

Note that the condition that the left-hand sides of $\mathcal{R}$ are not variables is new in comparison to [8]. Nevertheless, in the following corollary we can drop this condition, since otherwise $\llcorner\mathcal{R}\lrcorner$ is not terminating anyway.

Corollary 11. *If $\mathcal{R}_\eta$ is left head variable free then termination of $\llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U}$ implies termination of $\mathcal{R}$.*

When using uncurrying for relative termination of $\mathcal{R}/\mathcal{S}$, it turns out that the new condition on the left-hand sides can only be ignored for $\mathcal{R}$ – since otherwise relative termination of $\llcorner\mathcal{R}\lrcorner/\llcorner\mathcal{S}\lrcorner \cup \mathcal{U}$ does not hold – but not for $\mathcal{S}$.

Corollary 12. *If $\mathcal{R}_\eta \cup \mathcal{S}_\eta$ is left head variable free and the left-hand sides of $\mathcal{S}$ are not variables, then relative termination of $\llcorner\mathcal{R}_\eta\lrcorner/\llcorner\mathcal{S}_\eta\lrcorner \cup \mathcal{U}$ implies relative termination of $\mathcal{R}/\mathcal{S}$.*

Example 13. Let $\mathcal{R} = \{\mathsf{f} \circ \mathsf{f} \circ x \to \mathsf{f} \circ x\}$ and $\mathcal{S} = \{x \to \mathsf{f} \circ x\}$. Then $\mathcal{R}/\mathcal{S}$ is not relative terminating since $\mathsf{f} \circ \mathsf{f} \circ x \to_\mathcal{R} \mathsf{f} \circ x \to_\mathcal{S} \mathsf{f} \circ \mathsf{f} \circ x \to_\mathcal{R} \ldots$ is an infinite $\mathcal{R} \cup \mathcal{S}$-derivation with infinitely many $\mathcal{R}$-steps.

For $\pi(\mathsf{f}) = [\mathsf{f}, \mathsf{f}_1, \mathsf{f}_2]$ we have $\mathcal{R}_\eta = \mathcal{R}$, $\mathcal{S}_\eta = \mathcal{S}$, $\llcorner\mathcal{R}_\eta\lrcorner = \{\mathsf{f}_2(\mathsf{f}, x) \to \mathsf{f}_1(x)\}$, $\llcorner\mathcal{S}_\eta\lrcorner = \{x \to \mathsf{f}_1(x)\}$, and $\mathcal{U} = \{\mathsf{f} \circ x \to \mathsf{f}_1(x), \mathsf{f}_1(x) \circ y \to \mathsf{f}_2(x, y)\}$. It is easy to see that $\llcorner\mathcal{R}_\eta\lrcorner/\llcorner\mathcal{S}_\eta\lrcorner \cup \mathcal{U}$ is relative terminating by counting the number of f-symbols. Since both $\mathcal{R}_\eta$ and $\mathcal{S}_\eta$ are head variable free, we have shown that Corollary 12 does not hold if one would drop the new variable condition on $\mathcal{S}$.

As already mentioned, Corollary 11 generalizes the similar result of [8, Theorem 16] in two ways: first, there is no restriction to applicative signatures, and second, one can freely choose the applicative arities. Since in principle the choice of π does not matter (uncurrying preserves termination for every choice of π), we can only heuristically determine whether the additional freedom increases termination proving power and therefore refer to our experiments in Sect. 5.

4 Uncurrying in the Dependency Pair Framework

The DP framework [5] facilitates modular termination proofs. Instead of single TRSs, we consider *DP problems* $(\mathcal{P}, \mathcal{R})$, consisting of two TRSs $\mathcal{P}$ and $\mathcal{R}$ where elements of $\mathcal{P}$ are often called *pairs* to distinguish them from the *rules* of $\mathcal{R}$. The *initial DP problem* for a TRS $\mathcal{R}$ is $(\mathsf{DP}(\mathcal{R}), \mathcal{R})$, where $\mathsf{DP}(\mathcal{R}) = \{f^\sharp(s_1, \ldots, s_n) \to g^\sharp(t_1, \ldots, t_m) \mid f(s_1, \ldots, s_n) \to C[g(t_1, \ldots, t_m)] \in \mathcal{R}, g \in \mathcal{D}_\mathcal{R}\}$ is the set of *dependency pairs* of $\mathcal{R}$, incorporating a fresh *tuple symbol* $f^\sharp$ for each defined symbol f. The initial DP problem is also a *standard* DP problem, i.e., root symbols of pairs do not occur elsewhere in $\mathcal{P}$ or $\mathcal{R}$.[1] A $(\mathcal{P}, \mathcal{R})$-*chain*

[1] Several termination provers only work on standard DP problems.

is a possibly infinite derivation of the 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 $i > 0$. 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* [5], 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 with empty $\mathcal{P}$-component (such DP problems are trivially finite). To conclude finiteness of the initial DP problem, 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 $\mathit{Proc}(\mathcal{P}, \mathcal{R})$ implies finiteness of $(\mathcal{P}, \mathcal{R})$.

In the following we explain how uncurrying is used as processor in the DP framework. In Sect. 3 it was already mentioned that in [8] the notion of applicative TRS was lifted to applicative DP problem by allowing a new binary application symbol $\circ^\sharp$ (where we sometimes just write $^\sharp$ in examples). The symbol $\circ^\sharp$ naturally occurs as tuple symbol of $\circ$.

To prove soundness of the uncurrying processor, in [8] it is assumed that there is a minimal $(\mathcal{P}, \mathcal{R})$-chain $s_1\sigma_1 \to_{\mathcal{P}} t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2 \to_{\mathcal{P}} \cdots$, which is converted into a minimal $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$-chain by reusing the results for TRSs. However, there is a gap in this reasoning. Right in the beginning it is silently assumed that all terms $s_i\sigma_i$ and $t_i\sigma_i$ have tuple symbols as roots and that their arguments are applicative terms, i.e., terms over an applicative signature $\{\circ\} \cup \mathcal{C}$. Without this assumption it is not possible to apply the results of uncurrying for TRSs, since those are only available for applicative terms in [8].

The following variant of [12, Example 14] shows that in general restricting substitutions in chains to an applicative signature $\{\circ\} \cup \mathcal{C}$ is unsound.

Example 14. Consider the applicative and standard DP problem $(\mathcal{P}, \mathcal{R})$ where $\mathcal{P} = \{\mathsf{g}^\sharp\ (\mathsf{f}\ x\ y\ z\ z\ u\ v) \to \mathsf{g}^\sharp\ (\mathsf{f}\ x\ y\ x\ y\ x\ (\mathsf{h}\ y\ u))\}$ and $\mathcal{R}$ contains the rules:

$$\begin{array}{cc}
\mathsf{a} \to \mathsf{b} & \mathsf{f}\ \mathsf{a}\ x_2\ x_3\ x_4\ x_5 \to \mathsf{f}\ \mathsf{a}\ x_2\ x_3\ x_4\ x_5 \\
\mathsf{a} \to \mathsf{c} & \mathsf{f}\ x_1\ \mathsf{a}\ x_3\ x_4\ x_5 \to \mathsf{f}\ x_1\ \mathsf{a}\ x_3\ x_4\ x_5 \\
\mathsf{h}\ x\ x \to \mathsf{h}\ x\ x & \mathsf{f}\ (y\ z)\ x_2\ x_3\ x_4\ x_5 \to \mathsf{f}\ (y\ z)\ x_2\ x_3\ x_4\ x_5 \\
 & \mathsf{f}\ x_1\ (y\ z)\ x_3\ x_4\ x_5 \to \mathsf{f}\ x_1\ (y\ z)\ x_3\ x_4\ x_5
\end{array}$$

There is a minimal $(\mathcal{P}, \mathcal{R})$-chain taking $s_i = \mathsf{g}^\sharp\ (\mathsf{f}\ x\ y\ z\ z\ u\ v)$, $t_i = \mathsf{g}^\sharp\ (\mathsf{f}\ x\ y\ x\ y\ x\ (\mathsf{h}\ y\ u))$, and $\sigma_i = \{x/\mathsf{k}(\mathsf{a}), y/\mathsf{k}(\mathsf{b}), z/\mathsf{k}(\mathsf{b}), u/\mathsf{k}(\mathsf{c}), v/\mathsf{h}\ (\mathsf{k}(\mathsf{b}))\ (\mathsf{k}(\mathsf{c}))\}$ where k is a unary symbol. However, there is no minimal $(\mathcal{P}, \mathcal{R})$-chain using substitutions over the signature $\{\circ\} \cup \mathcal{C}$, regardless of the choice of constants $\mathcal{C}$.

Since our generalizations in Sect. 3 do not have any restrictions on the signature, we were able to correct the corresponding proofs in [8] such that the major theorems are still valid.[2] It follows the generalization of [8, Theorem 33].

[2] After the authors of [8] where informed of the gap, they independently developed an alternative fix, which is part of an extended, but not yet published version of [8].

Theorem 15. *The uncurrying processor $\mathcal{U}_1'$ is sound where* $\mathcal{U}_1'(\mathcal{P},\mathcal{R}) =$

$$\begin{cases} (\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U}) & \text{if } \mathcal{P}\cup\mathcal{R}_\eta \text{ is left head variable free and } \pi \text{ is injective,} \\ (\mathcal{P},\mathcal{R}) & \text{otherwise.} \end{cases}$$

Proof. The proof mainly uses the results from the previous section. We assume an infinite minimal $(\mathcal{P},\mathcal{R})$-chain $s_1\sigma_1 \to_\mathcal{P} t_1\sigma_1 \to_\mathcal{R}^* s_2\sigma_2 \to_\mathcal{P} t_2\sigma_2 \to_\mathcal{R}^* \cdots$ and construct an infinite minimal $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$-chain as follows.

We achieve $\llcorner s_i\sigma_i\lrcorner = \llcorner s_i\lrcorner \cdot \llcorner\sigma_i\lrcorner$ and $\llcorner t_i\lrcorner \cdot \llcorner\sigma_i\lrcorner \to_\mathcal{U}^* \llcorner t_i\sigma_i\lrcorner$ since $\mathcal{P}$ is left head variable free. Moreover, using $t_i\sigma_i \to_\mathcal{R}^* s_{i+1}\sigma_{i+1}$ and Theorem 10 we conclude $\llcorner t_i\sigma_i\lrcorner \to_{\llcorner\mathcal{R}_\eta\lrcorner\cup\mathcal{U}}^* \llcorner s_{i+1}\sigma_{i+1}\lrcorner$. Here, the condition that $\mathcal{R}_\eta$ must not contain variables as left-hand sides is ensured by the minimality of the chain: if $x \to r \in \mathcal{R}_\eta$ then $\mathsf{SN}_\mathcal{R}(t_i\sigma)$ does not hold. Hence, we constructed a $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$-chain as

$$\llcorner s_i\sigma_i\lrcorner = \llcorner s_i\lrcorner \cdot \llcorner\sigma_i\lrcorner \to_{\llcorner\mathcal{P}\lrcorner} \llcorner t_i\lrcorner \cdot \llcorner\sigma_i\lrcorner \to_\mathcal{U}^* \llcorner t_i\sigma_i\lrcorner \to_{\llcorner\mathcal{R}_\eta\lrcorner\cup\mathcal{U}}^* \llcorner s_{i+1}\sigma_{i+1}\lrcorner$$

for all i. To ensure that the chain is minimal it is demanded that π is injective. Otherwise, two different symbols can be mapped to the same new symbol which clearly can introduce nontermination. The structure of the proof that minimality is preserved is similar to the one in [8] and we just refer to IsaFoR for details.

The uncurrying processor of Theorem 15 generalizes [8, Theorem 33] in three ways: the signature does not have to be applicative, we can freely choose the applicative arity via π, and we can freely choose the application symbol. The last generalization lets Theorem 15 almost subsume the technique of freezing [8, Corollary 40] which is used to uncurry $\circ^\sharp$.

Definition 16 (Freezing [8]). *A* simple freeze $\maltese$ *is a subset of* $\mathcal{F}$.[3] Freezing *is applied on non-variable terms as follows*

$$\maltese(f(t_1,\ldots,t_n)) = \begin{cases} f(t_1,\ldots,t_n) & \text{if } n = 0 \text{ or } f \notin \maltese \\ f^g(s_1,\ldots,s_m,t_2,\ldots,t_m) & \text{if } t_1 = g(s_1,\ldots,s_m) \text{ and } f \in \maltese \end{cases}$$

where each f^g is a new symbol. It is homomorphically extended to rules and TRSs. The freezing DP processor is defined as $\maltese(\mathcal{P},\mathcal{R}) =$

$$\begin{cases} (\maltese(\mathcal{P}),\mathcal{R}) & \text{if } (\mathcal{P},\mathcal{R}) \text{ is a standard DP problem where for all } s \to f(t_1,\ldots,t_n) \\ & \in \mathcal{P} \text{ with } f \in \maltese, \text{ both } t_1 \notin \mathcal{V} \text{ and all instances of } t_1 \text{ are root-stable,} \\ (\mathcal{P},\mathcal{R}) & \text{otherwise.} \end{cases}$$

In [8, Theorem 39], it is shown that freezing is sound.

Example 17. In the following we use numbers to refer to rules from previous examples. We consider the DP problem $(\mathcal{P},\mathcal{R})$ where $\mathcal{P} = \{\mathsf{sub}\ (\mathsf{s}\ x)^\sharp(\mathsf{s}\ y) \to \mathsf{sub}\ x^\sharp\ y\}$ and $\mathcal{R} = \{2\text{–}4\}$. Uncurrying $\circ$ with $\pi(\mathsf{s}) = [\mathsf{s},\mathsf{s}_1]$, $\pi(\mathsf{sub}) = [\mathsf{sub},\mathsf{sub}_1,\mathsf{sub}_2]$,

[3] In [8] one can also specify an argument position for each symbol. This can be simulated by permuting the arguments accordingly.

$\pi(0) = [0]$, and $\pi(\sharp) = [\sharp]$ yields the DP problem $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$ where $\llcorner\mathcal{P}\lrcorner = \{\mathsf{sub}_1(\mathsf{s}_1(x))\ ^\sharp\ \mathsf{s}_1(y) \to \mathsf{sub}_1(x)\ ^\sharp\ y\}$ and $\llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U}$ consists of $\{10\text{–}12, 16, 19, 20\}$.

Afterwards we uncurry the resulting DP problem using $\circ^\sharp$ as application symbol and π where $\pi(\mathsf{sub}_1) = [\mathsf{sub}_1, -^\sharp]$ and $\pi(f) = [f]$ for all other symbols. We obtain $(\mathcal{P}', \mathcal{R}')$ where $\mathcal{P}' = \{\mathsf{s}_1(x)\ -^\sharp\ \mathsf{s}_1(x)) \to x\ -^\sharp\ y\}$ and $\mathcal{R}' = \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U} \cup \{\mathsf{sub}_1(x)\ ^\sharp\ y \to\ -^\sharp(x, y)\}$. Note that freezing returns nearly the same DP problem. The only difference is that uncurrying produces the additional rule $\mathsf{sub}_1(x)\ ^\sharp\ y \to x\ -^\sharp\ y$ which we do not obtain via freezing. However, since this rule is not usable it also does not harm that much.

Moreover, uncurrying sometimes is applicable where freezing is not. If we would have started with the DP problem $(\mathcal{P}, \mathcal{R}'')$ where $\mathcal{R}'' = \{1\text{–}5\}$ then uncurrying would result in $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}''_\eta\lrcorner \cup \mathcal{U}')$ where $\llcorner\mathcal{R}''_\eta\lrcorner \cup \mathcal{U}' = \{8\text{–}13, 16\text{–}20\}$. On this DP problem freezing is not applicable (the instances of $\mathsf{sub}_1(x)$ in the right-hand side of the only pair in $\llcorner\mathcal{P}\lrcorner$ are not root-stable due to Rule 8). Nevertheless, one can uncurry $\circ^\sharp$, resulting in $(\mathcal{P}', \llcorner\mathcal{R}''_\eta\lrcorner \cup \mathcal{U}' \cup \mathcal{R}_{\mathsf{new}})$ where $\mathcal{R}_{\mathsf{new}} = \{\mathsf{sub}_1(x)\ ^\sharp\ y \to x\ -^\sharp\ y, 0\ -^\sharp\ y \to \mathsf{K}_1(0)\ ^\sharp\ y\}$. Note that the uncurrying of $\circ^\sharp$ transformed a standard DP problem into a non-standard one, as $-^\sharp$ occurs as root of a term in $\mathcal{P}'$, but also within $\mathcal{R}_{\mathsf{new}}$.

Whenever freezing with $❄ = \{\circ^\sharp\}$ is applicable, then also uncurrying of $\circ^\sharp$ is possible: the condition $t_1 \notin \mathcal{V}$ in Definition 16 implies that $\mathcal{P} \cup \mathcal{R}_\eta$ is left head variable free. The only difference is that uncurrying produces more rules than freezing, namely the uncurrying rules and the uncurried rules of those rules which have to be added for the η-saturation. However, if freezing is applicable then none of these additional rules are usable.[4] Hence, all techniques which only consider the usable rules (like the reduction pair processor) perform equally well, no matter whether one has applied freezing or uncurrying. Still, one wants to get rid of the additional rules, especially since they are also the reason why standard DP problems are transformed into non-standard ones.

In Example 17 we have seen that sometimes uncurrying of tuple symbols is applicable where freezing is not. Thus, to have the best of both techniques we devised a special uncurrying technique for tuple symbols which fully subsumes freezing without the disadvantage of $\mathcal{U}'_1$: if freezing is applicable then standard DP problems are transformed into standard DP problems by the new technique.

Before we describe the new uncurrying processor formally, we shortly list the differences to the uncurrying processor of Theorem 15:

- Since the task is to uncurry tuple symbols, we restrict the applicative arities to be at most one. Moreover, uncurrying is performed only on the top-level. Finally, the application symbol may be of arbitrary non-zero arity.
- Rules that have to be added for the η-saturation and the uncurrying rules are added as pairs (to the $\mathcal{P}$-component), and not as rules (to the $\mathcal{R}$-component).
- If freezing is applicable, we do neither add the uncurrying rules nor do we apply η-saturation.

[4] In detail: a technique that can detect that instances of a subterm of a right-hand side of $\mathcal{P}$ are root-stable can also detect that the additional rules are not usable.

Example 18. We continue with the DP problems of Example 17.

If one applies the special uncurrying processor on $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$ then one obtains $(\mathcal{P}', \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$ which is the same as $❁(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}_\eta\lrcorner \cup \mathcal{U})$ for $❁ = \{\circ^\sharp\}$.

And if one applies the special uncurrying processor on $(\llcorner\mathcal{P}\lrcorner, \llcorner\mathcal{R}''_\eta\lrcorner \cup \mathcal{U}')$ then one obtains the standard DP problem $(\mathcal{P}' \cup \mathcal{R}_{\mathsf{new}}, \llcorner\mathcal{R}''_\eta\lrcorner \cup \mathcal{U}')$.

Definition 19. *Let $\circ^\sharp$ be an n-ary application symbol where $n > 0$. Let π be an injective symbol map where $\pi(f) \in \{[f], [f, f^\sharp]\}$ for all f. The top-uncurrying function $\ulcorner\cdot\urcorner$ maps terms to terms. It is defined as $\ulcorner t \urcorner =$*

$$\begin{cases} f^\sharp(s_1, \ldots, s_m, t_2, \ldots, t_n) & \text{if } t = \circ^\sharp(f(s_1, \ldots, s_m), t_2, \ldots, t_n) \text{ and } \pi(f) = [f, f^\sharp] \\ t & \text{otherwise} \end{cases}$$

and is homomorphically extended to pairs, rules, and substitutions. The top-uncurrying rules *are defined as*

$$\mathcal{U}^t = \{\circ^\sharp(f(x_1, \ldots, x_m), y_2, \ldots, y_n) \to f^\sharp(x_1, \ldots, x_m, y_2, \ldots, y_n) \mid \pi(f) = [f, f^\sharp]\}$$

Then the top-uncurrying processor *is defined as* $\mathsf{top}(\mathcal{P}, \mathcal{R}) =$

$$\begin{cases} (\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_?, \mathcal{R}) & \text{if } \circ^\sharp \text{ is not defined w.r.t. } \mathcal{R} \text{ and for all } s \to t \in \mathcal{P}_\eta : s, t \notin \mathcal{V}, \\ & \quad s \neq \circ^\sharp(x, s_2, \ldots, s_n), \text{ and the root of } t \text{ is not defined w.r.t. } \mathcal{R} \\ (\mathcal{P}, \mathcal{R}) & \text{otherwise} \end{cases}$$

where $\mathcal{U}^t_? = \emptyset$ and $\mathcal{P}_\eta = \mathcal{P}$ if for all $s \to \circ^\sharp(t_1, \ldots, t_n) \in \mathcal{P}$ and σ the term $t_1\sigma$ is root-stable, and $\mathcal{U}^t_? = \mathcal{U}^t$ and $\mathcal{P}_\eta = \mathcal{P} \cup \{\circ^\sharp(\ell, x_2, \ldots, x_n) \to \circ^\sharp(r, x_2, \ldots, x_n) \mid \ell \to r \in \mathcal{R}, \mathrm{root}(\ell) = g, \pi(g) = [g, g^\sharp]\}$, otherwise. Here, $x_2, \ldots, x_n$ are distinct fresh variables that do not occur in $\ell \to r$.

Theorem 20. *The top-uncurrying processor* top *is sound.*

Proof. The crucial part is to prove that whenever $t = f(t_1, \ldots, t_m) \to^*_{\mathcal{R}} s$, $f \notin \mathcal{D}_{\mathcal{R}}$, t is an instance of a right-hand side of $\mathcal{P}$, and $\mathsf{SN}_{\mathcal{R}}(t)$, then $\ulcorner t \urcorner \to^*_{\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_? \cup \mathcal{R}} \ulcorner s \urcorner$ where $\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_?$-steps are root steps and all terms in this derivation are terminating w.r.t. $\mathcal{R}$.

Using this result, the main result is established as follows. Assume there is an infinite minimal $(\mathcal{P}, \mathcal{R})$-chain. Then every step $s\sigma \to_{\mathcal{P}} t\sigma \to^*_{\mathcal{R}} s'\sigma'$ in the chain is transformed as follows. Since $s \to t \in \mathcal{P}$, we conclude that $t\sigma = f(t_1\sigma, \ldots, t_m\sigma)$ where $f \notin \mathcal{D}_{\mathcal{R}}$ and $\mathsf{SN}_{\mathcal{R}}(t\sigma)$. Hence, using the crucial step we know that $\ulcorner t\sigma \urcorner \to^*_{\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_? \cup \mathcal{R}} \ulcorner s'\sigma' \urcorner$. Moreover, by case analysis on t one can show that $\ulcorner t \urcorner\sigma \to^*_{\mathcal{U}^t_?} \ulcorner t\sigma \urcorner$ via root reductions, and similarly, by additionally using the restrictions on s one derives $\ulcorner s\sigma \urcorner = \ulcorner s \urcorner\sigma$. Hence,

$$\ulcorner s\sigma \urcorner = \ulcorner s \urcorner\sigma \to_{\ulcorner\mathcal{P}\urcorner} \ulcorner t \urcorner\sigma \to^*_{\mathcal{U}^t_?} \ulcorner t\sigma \urcorner \to^*_{\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_? \cup \mathcal{R}} \ulcorner s'\sigma' \urcorner$$

where all terms in this derivation right of $\to_{\ulcorner\mathcal{P}\urcorner}$ are terminating w.r.t. $\mathcal{R}$ and where all $\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_?$-steps are root reductions. Thus, we can turn the root reductions into pairs, resulting in an infinite minimal $(\ulcorner\mathcal{P}_\eta\urcorner \cup \mathcal{U}^t_?, \mathcal{R})$-chain.

To prove the crucial part we perform induction on the number of steps where the base case – no reductions – is trivial. Otherwise, $t = f(t_1, \dots, t_m) \to_{\mathcal{R}}^* u \to_{\mathcal{R}} s$. Using $\mathsf{SN}_{\mathcal{R}}(t)$ we also know $\mathsf{SN}_{\mathcal{R}}(u)$ and since $f \notin \mathcal{D}_{\mathcal{R}}$ we know that $u = f(u_1, \dots, u_m)$ and $t_i \to_{\mathcal{R}}^* u_i$ for all $1 \leqslant i \leqslant m$. Moreover, $s = f(s_1, \dots, s_m)$ and s is obtained from u by a reduction $u_i \to_{\mathcal{R}} s_i$ for some $1 \leqslant i \leqslant m$. Hence, we may apply the induction hypothesis and conclude $\ulcorner t \urcorner \to_{\ulcorner \mathcal{P}_\eta \urcorner \cup \mathcal{U}_?^t \cup \mathcal{R}}^* \ulcorner u \urcorner$.

It remains to simulate the reduction $u \to_{\mathcal{R}} s$. The simulation is easy if $f \neq \circ^\sharp$, since then $\ulcorner u \urcorner = u \to_{\mathcal{R}} s = \ulcorner s \urcorner$. Otherwise, $f = \circ^\sharp$ and $m = n$. We again first consider the easy case where $u_i \to_{\mathcal{R}} s_i$ for some $i > 1$. Then an easy case analysis on u_1 yields $\ulcorner u \urcorner \to_{\mathcal{R}} \ulcorner s \urcorner$ since u and s are uncurried in the same way (since $u_1 = s_1$). Otherwise, $u = \circ^\sharp(u_1, \dots, u_m)$, $s = \circ^\sharp(s_1, u_2, \dots, u_m)$ and $u_1 \to_{\mathcal{R}} s_1$. If $u_1 \to_{\mathcal{R}} s_1$ is a reduction below the root then both s and t are uncurried in the same way and again $\ulcorner u \urcorner \to_{\mathcal{R}} \ulcorner s \urcorner$ is easily established. If however $u_1 = \ell\sigma \to r\sigma = s_1$ for some rule $\ell \to r \in \mathcal{R}$ then we know that u_1 is not root-stable and hence also t_1 is not root-stable. As $t = \circ^\sharp(t_1, \dots, t_n)$ is an instance of a right-hand side of $\mathcal{P}$ we further know that there is a pair $s' \to \circ^\sharp(t'_1, \dots, t'_n) \in \mathcal{P}$ where $t_1 = t'_1\sigma$. Since $t'_1\sigma$ is not root-stable $\mathcal{U}_?^t = \mathcal{U}^t$ and $\mathcal{P}_\eta \supseteq \{\circ^\sharp(\ell, x_2, \dots, x_n) \to \circ^\sharp(r, x_2, \dots, x_n) \mid \ell \to r \in \mathcal{R}, \mathrm{root}(\ell) = g, \pi(g) = [g, g^\sharp]\}$. Let $\ell = g(\ell_1, \dots, \ell_k)$. If $\pi(g) = [g]$ then

$$\begin{aligned}
\ulcorner u \urcorner &= \ulcorner \circ^\sharp(g(\ell_1, \dots, \ell_k)\sigma, u_2, \dots, u_n) \urcorner \\
&= \circ^\sharp(g(\ell_1, \dots, \ell_k)\sigma, u_2, \dots, u_n) \\
&\to_{\mathcal{R}} \circ^\sharp(r\sigma, u_2, \dots, u_n) \\
&\to_{\mathcal{U}_?^t}^* \ulcorner \circ^\sharp(r\sigma, u_2, \dots, u_n) \urcorner \\
&= \ulcorner s \urcorner.
\end{aligned}$$

And otherwise, $\pi(g) = [g, g^\sharp]$. Hence, $\circ^\sharp(\ell, x_2, \dots, x_n) \to \circ^\sharp(r, x_2, \dots, x_n) \in \mathcal{P}_\eta$. We define $\delta = \sigma \uplus \{x_2/u_2, \dots, x_n/u_n\}$ and achieve

$$\begin{aligned}
\ulcorner u \urcorner &= \ulcorner \circ^\sharp(g(\ell_1, \dots, \ell_k)\sigma, u_2, \dots, u_n) \urcorner \\
&= g^\sharp(\ell_1\sigma, \dots, \ell_k\sigma, u_2, \dots, u_n) \\
&= g^\sharp(\ell_1, \dots, \ell_k, x_2, \dots, x_n)\delta \\
&= \ulcorner \circ^\sharp(g(\ell_1, \dots, \ell_k), x_2, \dots, x_n) \urcorner \delta \\
&= \ulcorner \circ^\sharp(\ell, x_2, \dots, x_n) \urcorner \delta \\
&\to_{\ulcorner \mathcal{P}_\eta \urcorner} \ulcorner \circ^\sharp(r, x_2, \dots, x_n) \urcorner \delta \\
&\to_{\mathcal{U}_?^t}^* \ulcorner \circ^\sharp(r, x_2, \dots, x_n)\delta \urcorner \\
&= \ulcorner \circ^\sharp(r\sigma, u_2, \dots, u_n) \urcorner \\
&= \ulcorner s \urcorner.
\end{aligned}$$

Using that π is injective one can also show that termination of all terms in the derivation is guaranteed where we refer to our library IsaFoR for details.

Note that the top-uncurrying processor fully subsumes freezing since the step from $(\mathcal{P}, \mathcal{R})$ to $(❄(\mathcal{P}), \mathcal{R})$ using $❄ = \{f_1, \dots, f_n\}$ can be simulated by n

applications of top where in each iteration one chooses f_i as application symbol and defines $\pi(g) = [g, f_i^g]$ for all $g \neq f_i$. The following example shows that top is also useful where freezing is not applicable.

Example 21. Consider the TRS $\mathcal{R}$ where $x \div y$ computes $\lceil \frac{x}{2y} \rceil$.

$$\begin{aligned}
\mathsf{s}(x) - \mathsf{s}(y) &\to x - y & \qquad \mathsf{double}(x) &\to x + x \\
0 - y &\to 0 & \mathsf{double}(0) &\to 0 \\
x - 0 &\to x & \mathsf{double}(\mathsf{s}(x)) &\to \mathsf{s}(\mathsf{s}(\mathsf{double}(x))) \\
0 + y &\to y & 0 \div \mathsf{s}(y) &\to 0 \\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{s}(x) \div \mathsf{s}(y) &\to \mathsf{s}((\mathsf{s}(x) - \mathsf{double}(\mathsf{s}(y))) \div \mathsf{s}(y))
\end{aligned}$$

Proving termination is hard for current termination provers. Let us consider the interesting DP problem $(\mathcal{P}, \mathcal{R})$ where $\mathcal{P} = \{\mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (\mathsf{s}(x) - \mathsf{double}(\mathsf{s}(y))) \div^\sharp \mathsf{s}(y)\}$. The problem is that one cannot use standard reduction pairs with argument filters since one has to keep the first argument of $-$, and then the filtered term of $\mathsf{s}(x)$ is embedded in the filtered term of $\mathsf{s}(x) - \mathsf{double}(\mathsf{s}(y))$. Consequently, powerful termination provers such as AProVE and TTT2 fail on this TRS.

However, one can uncurry the tuple symbol $\div^\sharp$ where $\pi(-) = [-, -^\sharp]$, $\pi(\mathsf{s}) = [\mathsf{s}, \mathsf{s}^\sharp]$, and $\pi(f) = [f]$, otherwise. Then the new DP problem $(\mathcal{P}', \mathcal{R})$ is created where $\mathcal{P}'$ consists of the following pairs

$$\begin{aligned}
(x - y) \div^\sharp z &\to -^\sharp(x, y, z) & \qquad -^\sharp(\mathsf{s}(x), \mathsf{s}(y), z) &\to -^\sharp(x, y, z) \\
\mathsf{s}(x) \div^\sharp y &\to \mathsf{s}^\sharp(x, y) & -^\sharp(0, y, z) &\to 0 \div^\sharp z \\
\mathsf{s}^\sharp(x, \mathsf{s}(y)) &\to -^\sharp(\mathsf{s}(x), \mathsf{double}(\mathsf{s}(y)), \mathsf{s}(y)) & -^\sharp(x, 0, z) &\to x \div^\sharp z
\end{aligned}$$

where the subtraction is computed via the new pairs, and not via the rules anymore. The right column consists of the uncurried and η-saturated $-$-rules, and the left column contains the two uncurrying rules followed by the uncurried pair of $\mathcal{P}$. Proving finiteness of this DP problem is possible using standard techniques: linear 0/1-polynomial interpretations and the dependency graph suffice. Therefore, termination of the whole example can be proven fully automatically by using a new version of TTT2 where top-uncurrying is integrated.

5 Heuristics and Experiments

The generalizations for uncurrying described in this paper are implemented in TTT2 [10]. To fix the symbol map we used the following three heuristics:

- π_+ corresponds to the definition of applicative arity of [8]. More formally, $\pi_+(f) = [f_0, \ldots, f_n]$ where n is maximal w.r.t. all $f(\ldots) \circ t_1 \circ \cdots \circ t_n$ occurring in $\mathcal{R}$. The advantage of π_+ is that all uncurryings are performed.
- $\pi_\pm$ is like π_+, except that the applicative arity is reduced whenever we would have to add a rule during η-saturation. Formally, $\pi_\pm(f) = [f_0, \ldots, f_n]$ where $n = \min(\mathsf{aa}_{\pi_+}(f), \min\{k \mid f(\ldots) \circ t_1 \circ \cdots \circ t_k \to r \in \mathcal{R}\})$.
- π_- is almost dual to π_+. Formally, $\pi_-(f) = [f_0, \ldots, f_n]$ where n is minimal w.r.t. all maximal subterms of the shape $f(\ldots) \circ t_1 \circ \cdots \circ t_n$ occurring in $\mathcal{R}$. The idea is to reduce the number of uncurrying rules.

We conducted two sets of experiments to evaluate our work. Note that all proofs generated during our experiments are certified by CeTA (version 1.18). Our

experiments were performed on a server with eight dual-core AMD Opteron® processors 885, running at a clock rate of 2.6 GHz and on 64 GB of main memory. The time limit for the first set of experiments was 10 s (as in [8]), whereas the time limit for the second set was 5 s (T$_{\text{T}}$T$_2$'s time limit in the termination competition).

The first set of experiments was run with a setup similar to [8]. Accordingly, as input we took the same 195 ATRSs from the termination problem database (TPDB). For proving termination, we switch from the input TRS to the initial DP problem and then repeat the following as often as possible: compute the estimated dependency graph, split it into its strongly connected components and apply the "main processor." Here, as "main processor" we incorporated the subterm criterion and matrix interpretations (of dimensions one and two). Concerning uncurrying, the following approaches were tested: no uncurrying (none), uncurry the given TRS before computing the initial DP problem (trs), apply $\mathcal{U}'_1/\mathcal{U}'_2$ as soon as all other processors fail (where $\mathcal{U}'_2$ is the composition of $\mathcal{U}'_1$ and top). The results can be found in Table 1. Since on ATRSs, our generalization of uncurrying corresponds to standard uncurrying, it is not surprising that the numbers of the first three columns coincide with those of [8] (modulo *mirroring* and a slight difference in the used strategy for trs). They are merely included to see the relative gain when using uncurrying on ATRSs.

With the second set of experiments, we tried to evaluate the total gain in certified termination proofs. Therefore, we took a restricted version of T$_{\text{T}}$T$_2$'s competition strategy that was used in the July 2010 issue of the international termination competition[5] (called *base strategy* in the following). The restriction was to use only those termination techniques that where certifiable by CeTA before our formalization of uncurrying. Then, we used this base strategy to filter the TRSs (we did ignore all SRSs) of the TPDB (version 8.0). The result were 511 TRSs for which T$_{\text{T}}$T$_2$ did neither generate a termination proof nor a non-termination proof using the base strategy. For our experiments we extended the base strategy by the generalized uncurrying techniques using different heuristics for the applicative arity. The results can be found in Table 2. It turned out, that the π_- heuristic is rather weak. Concerning $\pi_\pm$, there is at least one TRS that could not be proven using π_+, but with $\pi_\pm$. The total of 35 in the first row of Table 2 is already reached without taking $\mathcal{U}'_1$ into account. This indicates that in practice a combination of uncurrying as initial step (trs) and the processor $\mathcal{U}'_2$, gives the best results. Finally, note that in comparison to the July 2010 termination competition (where T$_{\text{T}}$T$_2$ could generate 262 certifiable proofs), the number of certifiable proofs of T$_{\text{T}}$T$_2$ is increased by over 10 % using the new techniques. In these experiments, termination has been proven for 10 *non-applicative* TRSs where our generalizations of uncurrying have been the key to success.

6 Conclusions

This paper describes the first formalization of uncurrying, an important technique to prove termination of higher-order functions which are encoded as first-

[5] http://termcomp.uibk.ac.at

Table 1. Experiments as in [8]

	direct		processor	
	none	trs	$\mathcal{U}'_1$	$\mathcal{U}'_2$
subterm criterion	41	53	41	66
matrix (dimension 1)	66	98	95	114
matrix (dimension 2)	108	137	133	138

Table 2. Newly certified proofs

	direct	processor		total
	trs	$\mathcal{U}'_1$	$\mathcal{U}'_2$	
π_+	26	16	22	35
$\pi_\pm$	28	15	17	29
π_-	24	14	14	24
total	28	16	24	36

order TRSs. The formalization revealed a gap in the original proof which is now fixed. Adding the newly developed generalization of uncurrying to our certifier CeTA, increased the number of certifiable proofs on the TPDB by 10 %.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1999)
2. Blanqui, F., Delobel, W., Coupet-Grimal, S., Hinderer, S., Koprowski, A.: CoLoR, a Coq library on rewriting and termination. In: WST'06. pp. 69–73 (2006)
3. Contejean, É., Paskevich, A., Urbain, X., Courtieu, P., Pons, O., Forest, J.: A3PAT, an approach for certified automated termination proofs. In: Gallagher, J.P., Voigtländer, J. (eds.) PEPM'10. pp. 63–72. ACM New York, NY, USA (2010)
4. Endrullis, J.: Jambox, available at http://joerg.endrullis.de
5. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. Journal of Automated Reasoning 37(3), 155–203 (2006)
6. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Gramlich, B. (ed.) FroCoS'05. LNAI, vol. 3717, pp. 216–231. Springer (2005)
7. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic termination proofs in the DP framework. In: Furbach, U., Shankar, N. (eds.) IJCAR'06. LNAI, vol. 4130, pp. 281–286. Springer (2006)
8. Hirokawa, N., Middeldorp, A., Zankl, H.: Uncurrying for termination. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR'08. LNAI, vol. 5330, pp. 667–681. Springer (2008)
9. Kennaway, R., Klop, J.W., Sleep, R., de Vries, F.J.: Comparing curried and uncurried rewriting. Journal of Symbolic Computation 21(1), 15–39 (1996)
10. Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: Treinen, R. (ed.) RTA'09. LNCS, vol. 5595, pp. 295–304. Springer (2009)
11. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
12. Sternagel, C., Thiemann, R.: Signature extensions preserve termination. In: Dawar, A., Veith, H. (eds.) CSL'10. LNCS, vol. 6247, pp. 514–528. Springer (2010)
13. Thiemann, R.: The DP Framework for Proving Termination of Term Rewriting. Ph.D. thesis, RWTH Aachen University (2007), Technical Report AIB-2007-17
14. Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs'09. LNCS, vol. 5674, pp. 452–468. Springer (2009)