Uncurrying for Termination[⋆]

Nao Hirokawa and Aart Middeldorp

Institute of Computer Science
University of Innsbruck
6020 Innsbruck, Austria
{nao.hirokawa,aart.middeldorp}@uibk.ac.at

Abstract. In this note we present a transformation from untyped applicative term rewrite systems to functional term rewrite systems that preserves termination. Our transformation can handle head variables in right-hand sides of rewrite rules.

1 Introduction

We are concerned with proving termination of applicative rewrite systems. The signature of an applicative term rewrite system (ATRS) consists of variables, constants, and a single binary function symbol called application and denoted by the infix and left-associative symbol $\circ$. In examples we often use juxtaposition instead of $\circ$. Every ordinary first-order rewrite system (TRS) can be transformed into an applicative rewrite system by currying.

Proving termination of ATRSs is challenging because simplification orders are not effective as $\circ$ is the only function symbol of non-zero arity. Moreover, the dependency pair method is of little help as $\circ$ is the only defined non-constant symbol.

Definition 1. *Let $\mathcal{F}$ be a (functional) signature. The currying system $\mathcal{C}(\mathcal{F})$ consists of the rewrite rules $f_{i+1}(x_1, \ldots, x_i, y) \rightarrow f_i(x_1, \ldots, x_i) \circ y$ for every n-ary function symbol $f \in \mathcal{F}$ and every $0 \leqslant i < n$. Here $f_n = f$ and, for every $0 \leqslant i < n$, f_i is a fresh function symbol of arity i.*

The currying system $\mathcal{C}(\mathcal{F})$ is confluent and terminating. Hence every term t has a unique normal form $t{\downarrow}_{\mathcal{C}(\mathcal{F})}$. For instance, $\mathsf{f}(\mathsf{a}, \mathsf{b})$ is transformed into $\mathsf{f}\ \mathsf{a}\ \mathsf{b}$.

Definition 2. *Let $\mathcal{R}$ be a TRS over the signature $\mathcal{F}$. The* curried *system $\mathcal{R}{\downarrow}_{\mathcal{C}(\mathcal{F})}$ is the ATRS consisting of the rules $l{\downarrow}_{\mathcal{C}(\mathcal{F})} \rightarrow r{\downarrow}_{\mathcal{C}(\mathcal{F})}$ for every $l \rightarrow r \in \mathcal{R}$. The signature of $\mathcal{R}{\downarrow}_{\mathcal{C}(\mathcal{F})}$ contains the application symbol $\circ$ and a constant f_0 for every function symbol $f \in \mathcal{F}$.*

In the following we write $\mathcal{R}{\downarrow}_{\mathcal{C}}$ for $\mathcal{R}{\downarrow}_{\mathcal{C}(\mathcal{F})}$ whenever $\mathcal{F}$ can be inferred from the context or is irrelevant. Moreover, we write f for f_0.

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

Example 3. The TRS $\mathcal{R}$

$$\begin{aligned} \mathsf{0} + y &\to y & \mathsf{0} \times y &\to \mathsf{0} \\ \mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{s}(x) \times y &\to (x \times y) + y \end{aligned}$$

is transformed into the ATRS $\mathcal{R}{\downarrow_{\mathcal{C}}}$

$$\begin{aligned} + \ \mathsf{0} \ y &\to y & \times \ \mathsf{0} \ y &\to \mathsf{0} \\ + \ (\mathsf{s} \ x) \ y &\to \mathsf{s} \ (+ \ x \ y) & \times \ (\mathsf{s} \ x) \ y &\to + \ (\times \ x \ y) \ y \end{aligned}$$

Every rewrite sequence in $\mathcal{R}$ can be transformed into a sequence in $\mathcal{R}{\downarrow_{\mathcal{C}}}$, but the reverse does not hold. For instance, with respect to the above example, the rewrite step $+ \ (\mathsf{s} \ (\times \ \mathsf{0})) \ \mathsf{0} \to \mathsf{s} \ (+ \ (\times \ \mathsf{0}) \ \mathsf{0})$ in $\mathcal{R}{\downarrow_{\mathcal{C}}}$ does not correspond to a rewrite step in $\mathcal{R}$. Nevertheless, termination of $\mathcal{R}$ implies termination of $\mathcal{R}{\downarrow_{\mathcal{C}}}$, a result due to Kennaway *et al.* [6].

Theorem 4. *A TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}{\downarrow_{\mathcal{C}}}$ is terminating.* □

2 Uncurrying

As an immediate consequence we get the following transformation method for proving termination of ATRSs.

Corollary 5. *An ATRS $\mathcal{R}$ is terminating if and only if there exists a terminating TRS $\mathcal{S}$ such that $\mathcal{S}{\downarrow_{\mathcal{C}}} = \mathcal{R}$ (modulo renaming).* □

In [5] this method is called *transformation $\mathcal{A}$*. As can be seen from the following example, the method does not handle partially applied terms and, more seriously, head variables. Hence the method is of limited applicability as it cannot cope with the higher-order aspects modeled by ATRSs.

Example 6. Consider the ATRS $\mathcal{R}$ (from [2])

$$\begin{aligned} 1\colon& & \mathsf{id} \ x &\to x & 4\colon& & \mathsf{map} \ f \ \mathsf{nil} &\to \mathsf{nil} \\ 2\colon& & \mathsf{add} \ \mathsf{0} &\to \mathsf{id} & 5\colon& & \mathsf{map} \ f \ (: \ x \ y) &\to \ : \ (f \ x) \ (\mathsf{map} \ f \ y) \\ 3\colon& & \mathsf{add} \ (\mathsf{s} \ x) \ y &\to \mathsf{s} \ (\mathsf{add} \ x \ y) \end{aligned}$$

Rules 1 and 4 are readily translated into functional form:

$$1\colon \quad \mathsf{id}_1(x) \to x \qquad 4\colon \mathsf{map}_2(f, \mathsf{nil}) \to \mathsf{nil}$$

However, we cannot find functional forms for rules 2 and 3 because the 'arity' of add is 1 in rule 2 and 2 in rule 3. Because of the presence of the head variable f in the subterm $f \ x$, there is no functional term t such that $t{\downarrow_{\mathcal{C}}} = \ : \ (f \ x) \ (\mathsf{map} \ f \ y)$. Hence rule 5 cannot be transformed.

For *simply typed* rewrite systems Aoto and Yamada [1, 2] introduced transformation techniques that use type information. Head variables are removed by instantiating them with 'template' terms of the appropriate type. In [3] they adapted the dependency pair method to deal with simply typed ATRSs.

In this note we present a different solution for the untyped case. At the end of this section we relate our approach to the one of [1, 2].

Definition 7. *Let $\mathcal{F}$ be an applicative signature. The uncurrying system $\mathcal{U}(\mathcal{F})$ consists of the rewrite rules $f_i(x_1, \ldots, x_i) \circ y \to f_{i+1}(x_1, \ldots, x_i, y)$ for every constant $f \in \mathcal{F}$ and every $i \geqslant 0$. Here $f_0 = f$ and, for every $i > 0$, f_i is a fresh function symbol of arity i.*

The uncurrying system $\mathcal{U}(\mathcal{F})$ is confluent and terminating. Hence every term t has a unique normal form $t{\downarrow_{\mathcal{U}(\mathcal{F})}}$.

Definition 8. *Let $\mathcal{R}$ be an ATRS over the signature $\mathcal{F}$. The* uncurried *system $\mathcal{R}{\downarrow_{\mathcal{U}(\mathcal{F})}}$ is the TRS consisting of the rules $l{\downarrow_{\mathcal{U}(\mathcal{F})}} \to r{\downarrow_{\mathcal{U}(\mathcal{F})}}$ for every $l \to r \in \mathcal{R}$.*

In the following we write $\mathcal{R}{\downarrow_\mathcal{U}}$ for $\mathcal{R}{\downarrow_{\mathcal{U}(\mathcal{F})}}$ whenever $\mathcal{F}$ can be inferred from the context or is irrelevant.

Example 9. The ATRS $\mathcal{R}$ of Example 6 is transformed into $\mathcal{R}{\downarrow_\mathcal{U}}$:

1:	$\mathsf{id}_1(x) \to x$	4:	$\mathsf{map}_2(f, \mathsf{nil}) \to \mathsf{nil}$
2:	$\mathsf{add}_1(\mathsf{0}) \to \mathsf{id}$	5:	$\mathsf{map}_2(f, {:}_2(x, y)) \to {:}_2(f \circ x, \mathsf{map}_2(f, y))$
3:	$\mathsf{add}_2(\mathsf{s}_1(x), y) \to \mathsf{s}_1(\mathsf{add}_2(x, y))$		

The TRS $\mathcal{R}{\downarrow_\mathcal{U}}$ is an obvious candidate for $\mathcal{S}$ in Corollary 5. However, as can be seen from the following example, the rules of $\mathcal{R}{\downarrow_\mathcal{U}}$ are not enough to simulate an arbitrary rewrite sequence in $\mathcal{R}$.

Example 10. The non-terminating ATRS $\mathcal{R} = \{\mathsf{id}\ x \to x, \mathsf{f}\ x \to \mathsf{id}\ \mathsf{f}\ x\}$ is transformed into the terminating TRS $\mathcal{R}{\downarrow_\mathcal{U}} = \{\mathsf{id}_1(x) \to x, \mathsf{f}_1(x) \to \mathsf{id}_2(\mathsf{f}, x)\}$.

In the above example we need rules that connect id_2 and id_1 as well as f_1 and f. The natural idea is now to add sufficiently many rules of $\mathcal{F}{\downarrow_\mathcal{U}}$.

Definition 11. *Let $\mathcal{R}$ be an ATRS over a signature $\mathcal{F}$. The* applicative arity *$\mathrm{aa}(f)$ of a constant $f \in \mathcal{F}$ is defined as the maximum n such that $f \circ t_1 \circ \cdots \circ t_n$ is a subterm in the left- or right-hand side of a rule in $\mathcal{R}$. This notion is extended to terms as follows:*

$$\mathrm{aa}(t) = \begin{cases} \mathrm{aa}(f) & \text{if } t \text{ is a constant } f \\ \mathrm{aa}(t_1) - 1 & \text{if } t = t_1 \circ t_2 \end{cases}$$

Note that $\mathrm{aa}(t)$ is undefined if the head symbol of t is a variable. We define $\mathcal{U}(\mathcal{R}, \mathcal{F})$ as the union of $\mathcal{R}{\downarrow_\mathcal{U}}$ and the subset of $\mathcal{F}{\downarrow_\mathcal{U}}$ consisting of all rules $f_i(x_1, \ldots, x_i) \circ y \to f_{i+1}(x_1, \ldots, x_i, y)$ with $0 \leqslant i < \mathrm{aa}(f)$. We say that $\mathcal{R}$ is left head variable free *if $\mathrm{aa}(t)$ is defined for every subterm t of a left-hand side of a rule in $\mathcal{R}$.*

In the following we write $\mathcal{U}(\mathcal{R})$ for $\mathcal{U}(\mathcal{R}, \mathcal{F})$ whenever $\mathcal{F}$ can be inferred from the context or is irrelevant.

Example 12. Consider the ATRS $\mathcal{R}$ in Example 10. We have $\mathrm{aa}(\mathsf{id}) = 2$ and $\mathrm{aa}(\mathsf{f}) = 1$. The TRS $\mathcal{U}(\mathcal{R})$ consists of the following rules

$$\begin{array}{rcr} \mathsf{id}_1(x) \to x & \mathsf{id} \circ x \to \mathsf{id}_1(x) & \mathsf{f} \circ x \to \mathsf{f}_1(x) \\ \mathsf{f}_1(x) \to \mathsf{id}_2(\mathsf{f}, x) & \mathsf{id}_1(x) \circ y \to \mathsf{id}_2(x, y) & \end{array}$$

and is easily shown to be terminating.

As the above example shows, we do not yet have a sound transformation. The ATRS $\mathcal{R}$ admits the cycle $\mathsf{f}\ x \to \mathsf{id}\ \mathsf{f}\ x \to \mathsf{f}\ x$. In $\mathcal{U}(\mathcal{R})$ we have $\mathsf{f}_1(x) \to \mathsf{id}_2(\mathsf{f}, x)$ but the term $\mathsf{id}_2(\mathsf{f}, x)$ does not rewrite to $\mathsf{f}_1(x)$. It would if the rule $\mathsf{id}\ x\ y \to x\ y$ would be present in $\mathcal{R}$. This inspires the following definition.

Definition 13. *Let $\mathcal{R}$ be a left head variable free ATRS. The* η-saturated *ATRS $\mathcal{R}_\eta$ is the smallest extension of $\mathcal{R}$ such that $l \circ x \to r \circ x \in \mathcal{R}_\eta$ whenever $l \to r \in \mathcal{R}_\eta$ and $\mathrm{aa}(l) > 0$.*

Note that $\to_{\mathcal{R}} = \to_{\mathcal{R}_\eta}$, so the rules added during η-saturation do not affect the termination behaviour of $\mathcal{R}$. We can now state our main result.

Theorem 14. *A left head variable free ATRS $\mathcal{R}$ is terminating if $\mathcal{U}(\mathcal{R}_\eta)$ is terminating.* □

We skip the proof due to lack of space. Instead we revisit Example 6.

Example 15. Consider again the ATRS $\mathcal{R}$ of Example 6. Proving termination of the transformed TRS $\mathcal{U}(\mathcal{R}_\eta)$

$$\begin{array}{rcr} \mathsf{id}_1(x) \to x & : \circ x \to :_1(x) & \mathsf{id} \circ x \to \mathsf{id}_1(x) \\ \mathsf{add}_1(0) \to \mathsf{id} & :_1(x) \circ y \to :_2(x, y) & \mathsf{add} \circ x \to \mathsf{add}_1(x) \\ \mathsf{add}_2(0, y) \to \mathsf{id}_1(y) & & \mathsf{add}_1(x) \circ y \to \mathsf{add}_2(x, y) \\ \mathsf{add}_2(\mathsf{s}_1(x), y) \to \mathsf{s}_1(\mathsf{add}_2(x, y)) & & \mathsf{s} \circ x \to \mathsf{s}_1(x) \\ \mathsf{map}_2(f, \mathsf{nil}) \to \mathsf{nil} & & \mathsf{map} \circ x \to \mathsf{map}_1(x) \\ \mathsf{map}_2(f, :_2(x, y)) \to :_2(f \circ x, \mathsf{map}_2(f, y)) & & \mathsf{map}_1(x) \circ y \to \mathsf{map}_2(x, y) \end{array}$$

is a piece of cake with the dependency pair method (recursive SCC algorithm with three applications of the subterm criterion).

The next example shows that left-head variable freeness condition cannot be weakened to the well-definedness of $\mathrm{aa}(l)$ for every left-hand side l.

Example 16. Consider the non-terminating ATRS $\mathcal{R}$

$$\mathsf{f}\ (x\ \mathsf{a}) \to \mathsf{f}\ (\mathsf{g}\ \mathsf{b}) \qquad\qquad \mathsf{g}\ \mathsf{b} \to \mathsf{h}\ \mathsf{a}$$

The transformed TRS $\mathcal{U}(\mathcal{R}_\eta)$

$$\begin{array}{cc} \mathsf{f}_1(x \circ \mathsf{a}) \to \mathsf{f}_1(\mathsf{g}_1(\mathsf{b})) & \mathsf{f} \circ x \to \mathsf{f}_1(x) \\ \mathsf{g}_1(\mathsf{b}) \to \mathsf{h}_1(\mathsf{a}) & \mathsf{g} \circ x \to \mathsf{g}_1(x) \end{array}$$

is terminating because its rules are oriented from left to right by the lexicographic path order with precedence $\circ \succ \mathsf{g}_1 \succ \mathsf{f}_1 \succ \mathsf{h}_1 \succ \mathsf{a} \succ \mathsf{b}$. Note that $\mathrm{aa}(\mathsf{f}\ (x\ \mathsf{a})) = 0$.

The uncurrying transformation is not always useful.

Example 17. Consider the one rule TRS $\mathcal{R} = \{\mathsf{C}\ x\ y\ z\ u \to x\ z\ (x\ y\ z\ u)\}$ from [4]. The termination of $\mathcal{R}$ is proved by the lexicographic path order with empty precedence. The transformed TRS $\mathcal{U}(\mathcal{R}_\eta)$ consists of

$$\begin{array}{cc} \mathsf{C}_4(x,y,z,u) \to x \circ z \circ (x \circ y \circ z \circ u) & \\ \mathsf{C} \circ x \to \mathsf{C}_1(x) & \mathsf{C}_2(x,y) \circ z \to \mathsf{C}_3(x,y,z) \\ \mathsf{C}_1(x) \circ y \to \mathsf{C}_2(x,y) & \mathsf{C}_3(x,y,z) \circ w \to \mathsf{C}_4(x,y,z,w) \end{array}$$

None of the tools that participated in the 2005 termination competition is able to prove the termination of $\mathcal{U}(\mathcal{R}_\eta)$.

At the end of this section we briefly discuss the work of Aoto and Yamada [1, 2] for proving termination of simply typed ATRSs by transformation techniques. After performing η-saturation, they eliminate head variables by instantiating them with 'template' terms of the appropriate type. In a final step, the resulting ATRS is translated into functional form.

Example 18. Consider again the ATRS $\mathcal{R}$ of Example 6. Suppose we adopt the following type declarations: $\mathsf{0}\colon \mathsf{int}$, $\mathsf{s}\colon \mathsf{int} \to \mathsf{int}$, $\mathsf{nil}\colon \mathsf{list}$, $(:)\colon \mathsf{int} \to \mathsf{list} \to \mathsf{list}$, $\mathsf{id}\colon \mathsf{int} \to \mathsf{int}$, $\mathsf{add}\colon \mathsf{int} \to \mathsf{int} \to \mathsf{int}$, and $\mathsf{map}\colon (\mathsf{int} \to \mathsf{int}) \to \mathsf{list} \to \mathsf{list}$. The head variable f in the right-hand side $:\ (\mathsf{id}\ x)\ (\mathsf{map}\ f\ y)$ has type $\mathsf{int} \to \mathsf{int}$. There are three template terms of this type: s, id, and $\mathsf{add}\ z$. Instantiating f by these three terms in $\mathcal{R}_\eta$ produces the ATRS $\mathcal{R}'$:

$$\begin{array}{cc} \mathsf{id}\ x \to x & \mathsf{map}\ f\ \mathsf{nil} \to \mathsf{nil} \\ \mathsf{add}\ \mathsf{0} \to \mathsf{id} & \mathsf{map}\ \mathsf{s}\ (:\ x\ y) \to\ :\ (\mathsf{s}\ x)\ (\mathsf{map}\ \mathsf{s}\ y) \\ \mathsf{add}\ \mathsf{0}\ y \to \mathsf{id}\ y & \mathsf{map}\ \mathsf{id}\ (:\ x\ y) \to\ :\ (\mathsf{id}\ x)\ (\mathsf{map}\ \mathsf{id}\ y) \\ \mathsf{add}\ (\mathsf{s}\ x)\ y \to \mathsf{s}\ (\mathsf{add}\ x\ y) & \mathsf{map}\ (\mathsf{add}\ z)\ (:\ x\ y) \to\ :\ (\mathsf{add}\ z\ x)\ (\mathsf{map}\ (\mathsf{add}\ z)\ y) \end{array}$$

The TRS $\mathcal{R}'{\downarrow_\mathcal{U}}$ is terminating because its rules are oriented from left to right by the lexicographic path order. According to the main result of [2], the *simply typed* ATRS $\mathcal{R}$ is terminating, too.

The advantage of the simply typed approach is that the uncurrying rules are not necessary because the application symbol has been eliminated from $\mathcal{R}'{\downarrow_\mathcal{U}}$. This typically results in simpler termination proofs. It is worthwhile to investigate whether a version of head variable instantiation can be developed for the untyped case. We would like to stress that with the simply typed approach one obtains termination only for those terms which are simply typed. Our approach, when it works, provides termination for all terms, irrespective of *any* typing discipline.

3 Mirroring

In the final part of this note we describe a trivial mirroring technique for TRSs. This technique can be used to eliminate some of the head variables in an ATRS.

Definition 19. *Let t be a term. The term t^M is defined as follows: $t^M = t$ if t is a variable and $t^M = f(t_n^M, \ldots, t_1^M)$ if $t = f(t_1, \ldots, t_n)$. Moreover, if $\mathcal{R}$ is a TRS then $\mathcal{R}^M = \{l^M \to r^M \mid l \to r \in \mathcal{R}\}$.*

We obviously have $t \to_{\mathcal{R}} u$ if and only if $t^M \to_{\mathcal{R}^M} u^M$. This gives the following result.

Theorem 20. *A TRS $\mathcal{R}$ is terminating if and only if $\mathcal{R}^M$ is terminating.* □

Example 21. Consider the one-rule ATRS $\mathcal{R}$

$$x\ (\mathsf{a}\ \mathsf{a}\ \mathsf{a}) \to \mathsf{a}\ (\mathsf{a}\ \mathsf{a})\ x$$

While $\mathcal{R}$ has a head variable in its left-hand side, the mirrored version $\mathcal{R}^M$

$$\mathsf{a}\ (\mathsf{a}\ \mathsf{a})\ x \to x\ (\mathsf{a}\ \mathsf{a}\ \mathsf{a})$$

is left head variable free. The transformed TRS $\mathcal{U}((\mathcal{R}^M)_\eta)$

$$\mathsf{a}_2(\mathsf{a}_1(\mathsf{a}), x) \to x \circ \mathsf{a}_2(\mathsf{a}, \mathsf{a}) \qquad \mathsf{a} \circ x \to \mathsf{a}_1(x) \qquad \mathsf{a}_1(x) \circ y \to \mathsf{a}_2(x, y)$$

is easily proved terminating with dependency pairs and a linear polynomial interpretation.

Needless to say, mirroring may also introduce head variables in ATRSs. We anticipate that *partial* mirroring will be more effective in practice.

References

1. T. Aoto and T. Yamada. Termination of simply typed term rewriting by translation and labelling. In *Proc. 14th RTA*, volume 2706 of *LNCS*, pages 380–394, 2003.
2. T. Aoto and T. Yamada. Termination of simply-typed applicative term rewriting systems. In *Proc. 2nd HOR*, Technical Report AIB-2004-03, RWTH Aachen, pages 61–65, 2004.
3. T. Aoto and T. Yamada. Dependency pairs for simply typed term rewriting. In *Proc. 16th RTA*, volume 2706 of *LNCS*, pages 120–134, 2005.
4. N. Dershowitz. 33 Examples of termination. In *French Spring School of Theoretical Computer Science*, volume 909 of *LNCS*, pages 16–26, 1995.
5. J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. 5th FroCoS*, volume 3717 of *LNAI*, pages 216–231, 2005.
6. R. Kennaway, J.W. Klop, M.R. Sleep, and F.J. de Vries. Comparing curried and uncurried rewriting. *Journal of Symbolic Computation*, 21(1):15–39, 1996.