

Verifying the Correctness of Tupling Transformations based on Conditional Rewriting*

Yuki Chiba¹

1 School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
chiba@jaist.ac.jp

Abstract

Chiba et al. (2010) proposed a framework of program transformation by templates based on term rewriting. Their framework can deal with tupling, which improves efficiency of programs. Outputs of their framework, however, may not always be more efficient than inputs. In this paper, we propose a technique to show the correctness of tupling based on conditional term rewriting. We give an extended equational logic in order to add conditional rules.

1998 ACM Subject Classification I.2.2 Automatic Programming

Keywords and phrases Tupling, The Correctness Of Transformations, Conditional Term Rewriting

1 Introduction

Tupling [1] is a well-known technique to improve efficiency of programs by eliminating redundant recursive calls. Chiba et al. proposed a framework of program transformation by templates based on term rewriting for tupling [3, 4]. RAPT¹ is an implementation of their framework. RAPT transforms an input term rewriting system (TRS, for short) according to an input template together with verifying its correctness. Several automated theorem proving techniques are assembled for that verification.

The following TRS formalizes a usual definition of Fibonacci numbers ².

$$\mathcal{R}_{\text{fib}} \left\{ \begin{array}{ll} \text{fib}(0) & \rightarrow 0, \\ \text{fib}(\text{s}(0)) & \rightarrow \text{s}(0), \\ \text{fib}(\text{s}(\text{s}(x))) & \rightarrow +(\text{fib}(\text{s}(x)), \text{fib}(x)), \\ & +(\text{s}(x), y) \rightarrow \text{s}(+(x, y)), \\ & +(0, x) \rightarrow x, \\ & \pi_1(\langle x, y \rangle) \rightarrow x, \\ & \pi_2(\langle x, y \rangle) \rightarrow y \end{array} \right\}$$

$\mathcal{R}_{\text{fib}}$ can be translated to the following TRS $\mathcal{R}'_{\text{fib}}$ by RAPT according to the template 6 in <http://www.jaist.ac.jp/~chiba/RAPT/>.

$$\mathcal{R}'_{\text{fib}} \left\{ \begin{array}{ll} \text{fib}(0) & \rightarrow 0, \\ \text{fib}(\text{s}(0)) & \rightarrow \text{s}(0), \\ \text{fib}(\text{s}(\text{s}(x))) & \rightarrow \pi_1(\text{step}(\text{fibpair}(x))), \\ \text{fibpair}(0) & \rightarrow \langle \text{s}(0), 0 \rangle, \\ \text{fibpair}(\text{s}(x)) & \rightarrow \text{step}(\text{fibpair}(x)), \\ \text{step}(x) & \rightarrow \langle +(\pi_1(x), \pi_2(x)), \pi_1(x) \rangle, \\ +(0, x) & \rightarrow x, \\ \pi_1(\langle x, y \rangle) & \rightarrow x, \\ +(s(x), y) & \rightarrow \text{s}(+(x, y)), \\ \pi_2(\langle x, y \rangle) & \rightarrow y \end{array} \right\}$$

* This work was partially supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B) 23700034.

¹ <http://www.jaist.ac.jp/~chiba/RAPT/>

² Projection rules, π_1 and π_2 , are needed to show the correctness of the transformation from $\mathcal{R}_{\text{fib}}$ to $\mathcal{R}'_{\text{fib}}$.



© Yuki Chiba;
licensed under Creative Commons License CC-BY

1st International Workshop on Rewriting Techniques for Program Transformations and Evaluation (WPTE'14).
Editors: Manfred Schmidt-Schauß, Masahiko Sakai, David Sabel, and Yuki Chiba; pp. 65–68

$\mathcal{R}'_{\text{fib}}$ is not always be more efficient than $\mathcal{R}_{\text{fib}}$, because the function `step` may copy a recursive call of `fibpair(n)` when the outermost strategy is applied.

Dershowitz et al. [5] proposed a different formalization of tupled Fibonacci by using a conditional term rewriting system (CTRS, for short):

$$\mathcal{R}''_{\text{fib}} \left\{ \begin{array}{lll} \text{fib}(x) & \rightarrow z & \Leftarrow \text{fibpair}(x) \rightarrow \langle y, z \rangle \\ \text{fibpair}(0) & \rightarrow \langle s(0), 0 \rangle & \Leftarrow \\ \text{fibpair}(s(x)) & \rightarrow \langle y + z, y \rangle & \Leftarrow \text{fibpair}(x) \rightarrow \langle y, z \rangle \end{array} \right\}$$

$\mathcal{R}''_{\text{fib}}$ uses conditions for *variable binding*, which introduces extra variables, like **where** clauses in Haskell and **let** expressions in SML. Since innermost redexes are preferentially reduced by variable binding, one can solve the problem such that $\mathcal{R}'_{\text{fib}}$ may not improve the efficiency.

In this paper, we propose a technique to show the correctness of transformations from unconditional TRSs to CTRSs containing variable binding. We extend *equivalent transformations* of TRSs [8, 3, 4] to show the equivalence of TRSs and CTRSs. The framework of equivalent transformations consists of three transformation rules: *Introduction*, which introduces a new function as macro defined by existing functions, *Addition*, which adds a pair of terms that are equivalent on current systems as a rewriting rule, and *Elimination*, which eliminates redundant rules. We propose an extended equational logic (EEL, for short) in order to add conditional rules by Addition rule.

2 Extended Equational Logic

A *condition* is a collection of pairs of terms. We denote a condition as $s_1 \rightarrow t_1 \wedge \dots \wedge s_n \rightarrow t_n$. We treat that $\wedge$ is associative and commutative. A *conditional rewriting rule* is a triple $\langle l, r, C \rangle$ of terms l and r and a condition C . We usually denote a conditional rewriting rule $\langle l, r, C \rangle$ as $l \rightarrow r \Leftarrow C$. A *conditional term rewriting system* (CTRS, for short) is a set of conditional rewriting rules. A *reduction relation by a CTRS* $\mathcal{R}$ (denoted as $\rightarrow_{\mathcal{R}}$) is defined as usual. A CTRS $\mathcal{R}$ is *confluent* (denoted as $\text{CR}(\mathcal{R})$) if $\xleftarrow{*}_{\mathcal{R}} \cdot \xrightarrow{*}_{\mathcal{R}} \subseteq \xrightarrow{*}_{\mathcal{R}} \cdot \xleftarrow{*}_{\mathcal{R}}$.

Let s and t be terms, C a condition, $\mathcal{R}$ a CTRS and $\mathcal{E}$ a set of equations. We write $\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow C$ if there exists a derivation by inference rules in Table 1.

► **Theorem 1** (Soundness of EEL). *Suppose $\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow s_1 \wedge \dots \wedge t_n \rightarrow s_n$. $\mathcal{R}$ is confluent and for any substitution σ , $t_1\sigma \xrightarrow{*}_{\mathcal{R}} s_1\sigma$, $\dots$, $t_n\sigma \xrightarrow{*}_{\mathcal{R}} s_n\sigma$ imply $s\sigma \xrightarrow{*}_{\mathcal{R} \cup \mathcal{E}} t\sigma$*

(Proof)

Induction on the derivation of $\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow s_1 \wedge \dots \wedge t_n \rightarrow s_n$. □

3 Equivalent Transformation for Conditional Rewriting

To show the correctness of transformations, we define the notion of the equivalence of rewriting systems.

► **Definition 2.** Two CTRSs $\mathcal{R}$ and $\mathcal{R}'$ are said to be *equivalent* for a set $\mathcal{G}$ of function symbols (notation, $\mathcal{R} \simeq_{\mathcal{G}} \mathcal{R}'$) if for any ground term $s \in T(\mathcal{G})$ and ground constructor term $t \in T(\mathcal{C})$, $s \xrightarrow{*}_{\mathcal{R}} t$ iff $s \xrightarrow{*}_{\mathcal{R}'} t$ holds.

We now give an extension of equivalent transformation. Let $\mathcal{R}_0$ be a left-linear constructor system over $\mathcal{F}_0$ and $\mathcal{E}$ a set of equations over $\mathcal{F}_0$. An *equivalent transformation sequence under $\mathcal{E}$* is a sequence $\mathcal{R}_0, \dots, \mathcal{R}_n$ of CTRSs (over $\mathcal{F}_0, \dots, \mathcal{F}_n$, respectively) such that $\mathcal{R}_{k+1}$ is obtained from $\mathcal{R}_k$ by applying one of the following inference rules:

■ **Table 1** Inference rules of extended equational logic (EEL)

$$\begin{array}{c}
\frac{}{\overline{\mathcal{R}, \mathcal{E} \vdash s \approx s}} \quad \frac{}{\overline{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t \rightarrow s}} \\
\frac{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow C}{\mathcal{R}, \mathcal{E} \vdash t \approx s \Leftarrow C} \quad \frac{\mathcal{R}, \mathcal{E} \vdash s \approx u \Leftarrow C_1 \quad \mathcal{R}, \mathcal{E} \vdash u \approx t \Leftarrow C_2}{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow C_1 \wedge C_2} \\
\\
\frac{}{\overline{\mathcal{R}, \mathcal{E} \vdash l\sigma \approx r\sigma}} \text{ if } l \sim r \in \mathcal{E} \\
\\
\frac{\mathcal{R} \vdash s_1\sigma \approx t_1\sigma \quad \cdots \quad \mathcal{R} \vdash s_n\sigma \approx t_n\sigma}{\mathcal{R}, \mathcal{E} \vdash l\sigma \approx r\sigma} \text{ if } l \rightarrow r \Leftarrow \overline{t_i \rightarrow s_i} \in \mathcal{R} \\
\\
\frac{\mathcal{R}, \mathcal{E} \vdash s_1 \approx t_1 \quad \cdots \quad \mathcal{R}, \mathcal{E} \vdash s_n \approx t_n}{\mathcal{R}, \mathcal{E} \vdash f(s_1, \dots, s_n) \approx f(t_1, \dots, t_n)} \\
\\
\frac{}{\overline{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow x_1 \wedge \cdots \wedge t_n \rightarrow x_n}} \text{ if } \begin{array}{l} s = t\{x_i \mapsto t_i \mid 1 \leq i \leq n\} \\ \text{and } \forall i. x_i \notin \text{Var}(s) \end{array} \\
\\
\frac{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow s_1 \wedge C}{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t'_1 \rightarrow s_1 \wedge C} \text{ if } t_1 \xrightarrow{*} t'_1 \\
\\
\frac{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow s_1 \wedge \cdots \wedge t_n \rightarrow s_n \wedge C}{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow c(t_1, \dots, t_n) \rightarrow c(s_1, \dots, s_n) \wedge C} \text{ if } c \in \mathcal{C} \\
\\
\frac{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow C}{\mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t' \rightarrow s' \wedge C} \text{ if } s' \in \text{T}(\mathcal{C}, \mathcal{V})
\end{array}$$

(I) *Introduction*

$$\mathcal{R}_{k+1} = \mathcal{R}_k \cup \{f(x_1, \dots, x_n) \rightarrow r\}$$

provided that $f \notin \mathcal{F}_k$, and $r \in \text{T}(\mathcal{F}_k, \{x_1, \dots, x_n\})$, where $x_1, \dots, x_n$ are mutually distinct variables. We put $\mathcal{F}_{k+1} = \mathcal{F}_k \cup \{f\}$.

(A) *Addition*

$$\mathcal{R}_{k+1} = \mathcal{R}_k \cup \{l \rightarrow r \Leftarrow C\}$$

provided $\mathcal{R}_k, \mathcal{E} \vdash l \approx r \Leftarrow C$.

(E) *Elimination*

$$\mathcal{R}_{k+1} = \mathcal{R}_k \setminus \{l \rightarrow r\}$$

If this is the case, we write $\mathcal{R}_k \Rightarrow \mathcal{R}_{k+1}$. (In the Addition and Elimination rules, $\mathcal{F}_{k+1}$ can be any set of function symbols such that $\mathcal{F}_{k+1} \subseteq \mathcal{F}_k$ provided that $\mathcal{R}_{k+1}$ is a TRS over $\mathcal{F}_{k+1}$.) The reflexive transitive closure of $\Rightarrow$ is denoted by $\xRightarrow{*}$. We indicate the rule of $\Rightarrow$ by $\xRightarrow{I}$, $\xRightarrow{A}$, or $\xRightarrow{E}$. Finally, we say there exists an *equivalent transformation* from $\mathcal{R}$ to $\mathcal{R}'$ under $\mathcal{E}$ if there exists an equivalent transformation sequence $\mathcal{R} \xRightarrow{I}^* \cdot \xRightarrow{A}^* \cdot \xRightarrow{E}^* \mathcal{R}'$ under $\mathcal{E}$.

A set $\mathcal{F}$ of function symbols is divided to two sets $\mathcal{F}_D$ and $\mathcal{F}_C$ of *defined symbols* and *constructor symbols*, respectively. A CTRS $\mathcal{R}$ is *sufficiently complete with respect to a set $\mathcal{G}$ of function symbols* (denoted as $\text{SC}(\mathcal{R}, \mathcal{G})$) if for any ground term $s \in \text{T}(\mathcal{G})$, there exist a ground constructor term $t \in \text{T}(\mathcal{F}_C)$ such that $s \xrightarrow{*}_{\mathcal{R}} t$. A set $\mathcal{E}$ of equations is *inductive*

consequences of a CTRS $\mathcal{R}$ with respect to a set $\mathcal{G}$ of function symbols if for any $l \approx r \in \mathcal{E}$ and a substitution σ on $\mathcal{G}$, $l\sigma \xrightarrow{*}_{\mathcal{R}} r\sigma$.

► **Theorem 3** (Show the equivalence by equivalent transformations). *Let $\mathcal{G}$ and $\mathcal{G}'$ be sets of function symbols such that $\mathcal{F}_C \subseteq \mathcal{G}, \mathcal{G}' \subseteq \mathcal{F}$. Let $\mathcal{R}$ be a left-linear constructor system over $\mathcal{G}$, $\mathcal{E}$ a set of equations over $\mathcal{G}$, and $\mathcal{R}'$ a CTRS over $\mathcal{G}'$. Suppose that $\mathcal{R}, \mathcal{G} \vdash_{ind} \mathcal{E}$ and there exists an equivalent transformation from $\mathcal{R}$ to $\mathcal{R}'$ under $\mathcal{E}$. Then, $CR(\mathcal{R}) \wedge SC(\mathcal{R}, \mathcal{G}) \wedge SC(\mathcal{R}', \mathcal{G}') \text{ imply } \mathcal{R} \simeq_{\mathcal{G} \cap \mathcal{G}'} \mathcal{R}'$.*

(Proof)

Similar to the proof of Theorem 1 in [3]. □

► **Example 4.** There exists an equivalent transformation from $\mathcal{R}_{fib}$ to $\mathcal{R}_{fib}''$.

4 Conclusion

We propose a technique to show the equivalence of CTRSs in order to verify the correctness of tupling transformations. Our framework extends equivalent transformations in [3].

For automated verification of the equivalence of CTRSs based on theorem 3, sufficient completeness of CTRSs should be verified automatically. There are several techniques for checking sufficient completeness of CTRSs [2, 6]. We try to show sufficient completeness of $\mathcal{R}_{fib}''$ by the Maude Sufficient Completeness Checker³ (Maude-SCC, for short). It, however, is failed to show it. One can manually show sufficient completeness of $\mathcal{R}_{fib}''$ by induction with existential quantifier like $\forall n. \exists n', m'. fibpair(n) \xrightarrow{*}_{\mathcal{R}_{fib}''} \langle n', m' \rangle$.

In our framework, input systems for transformations should be unconditional TRSs. We need to extend the commutativity of confluence [7] in order to deal with CTRSs as inputs of transformations.

References

- 1 R. S. Bird. Using circular programs to eliminate multiple traversals of data. *Acta Informatica*, 21:239–250, 1984.
- 2 A. Bouhoula and F. Jacquemard. Sufficient completeness verification for conditional and constrained term rewriting systems. *Journal of Applied Logic*, 10(1):127–143, 2012.
- 3 Y. Chiba, T. Aoto, and Y. Toyama. Program transformation by templates based on term rewriting. In *Proc. of PPDP 2005*, pages 59–69. ACM Press, 2005.
- 4 Y. Chiba, T. Aoto, and Y. Toyama. Program transformation templates for tupling based on term rewriting. *IEICE Transactions*, 93-D(5):963–973, 2010.
- 5 N. Dershowitz, M. Okada, and G. Sivakumar. Confluence of conditional rewrite systems. In *Proceedings of the 1st International Workshop on Conditional Term Rewriting System*, volume 308 of *LNCS*, pages 31–44. Springer-Verlag, 1988.
- 6 J. Hendrix, M. Clavel, and J. Meseguer. A sufficient completeness reasoning tool for partial specifications. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications*, volume 3467 of *LNCS*, pages 165–174. Springer-Verlag, 2005.
- 7 Y. Toyama. Commutativity of term rewriting systems. In *The Second France-Japan Artificial Intelligence and Computer Science Symposium*, 1987.
- 8 Y. Toyama. How to prove equivalence of term rewriting systems without induction. *Theoretical Computer Science*, 90:369–390, 1991.

³ <http://maude.cs.uiuc.edu/tools/scc/>