Verifying the Correctness of Tupling Transformations based on Conditional Rewriting*

Yuki Chiba

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

Digital Object Identifier 10.4230/OASIcs.WPTE.2014.<first-page-number>

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]. RAPT 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.

\[
\mathcal{R}_{fib} = \begin{cases} 
    \text{fib}(0) & \rightarrow 0, \\
    \text{fib}(s(0)) & \rightarrow s(0), \\
    \text{fib}(s(s(x))) & \rightarrow +(\text{fib}(s(x)), \text{fib}(x)), \\
    +\langle x, y \rangle & \rightarrow \pi_1(x), \\
    \pi_2((x, y)) & \rightarrow y
\end{cases}
\]

\[
\mathcal{R}_{fib}' = \begin{cases} 
    \text{fib}(0) & \rightarrow 0, \\
    \text{fib}(s(0)) & \rightarrow s(0), \\
    \text{fib}(s(s(x))) & \rightarrow +(\text{fib}(s(x)), \text{fib}(x)), \\
    +\langle x, y \rangle & \rightarrow \pi_1(x), \\
    \pi_2((x, y)) & \rightarrow y
\end{cases}
\]

\[
\mathcal{R}_{fib}' \rightarrow \mathcal{R}_{fib} \text{ by RAPT according to the template 6 in http://www.jaist.ac.jp/~chiba/RAPT/}
\]

* 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, \(\pi_1\) and \(\pi_2\), are needed to show the correctness of the transformation from \(\mathcal{R}_{fib}\) to \(\mathcal{R}_{fib}'\).
Verifying the Correctness of Tupling Transformations based on Conditional Rewriting

\( \mathcal{R}_\text{fib}' \) is not always be more efficient than \( \mathcal{R}_\text{fib} \), because the function \texttt{step} may copy a recursive call of \texttt{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):

\[
\mathcal{R}_\text{fib}'' = \begin{cases}
\texttt{fib}(x) \rightarrow z & \iff \texttt{fibpair}(x) \rightarrow (y, z) \\
\texttt{fibpair}(0) \rightarrow \langle s(0), 0 \rangle & \iff \texttt{fibpair}(x) \rightarrow (y, z) \\
\texttt{fibpair}(s(x)) \rightarrow (y + z, y) & \iff \texttt{fibpair}(x) \rightarrow (y, z)
\end{cases}
\]

\( \mathcal{R}_\text{fib}'' \) uses conditions for \textit{variable binding}, which introduces extra variables, like \texttt{where} clauses in Haskell and \texttt{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: \textit{Introduction}, which introduces a new function as macro defined by existing functions, \textit{Addition}, which adds a pair of terms that are equivalent on current systems as a rewriting rule, and \textit{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 \textit{condition} is a collection of pairs of terms. We denote a condition as \( s_1 \rightarrow t_1 \land \cdots \land s_n \rightarrow t_n \). We treat that \( \land \) is associative and commutative. A \textit{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 \textit{conditional term rewriting system} (CTRS, for short) is a set of conditional rewriting rules. A \textit{reduction relation} by a CTRS \( \mathcal{R} \) (denoted as \( \rightarrow_{\mathcal{R}} \)) is defined as usual. A CTRS \( \mathcal{R} \) is \textit{confluent} (denoted as \( \text{CR}(\mathcal{R}) \)) if \( \leftarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}} \).

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 \( \mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow s_1 \land \cdots \land t_n \rightarrow s_n \). \( \mathcal{R} \) is confluent and for any substitution \( \sigma \), \( t_1 \sigma \rightarrow_{\mathcal{R}} s_1 \sigma, \ldots, t_n \sigma \rightarrow_{\mathcal{R}} s_n \sigma \) imply \( s \sigma \rightarrow_{\mathcal{R}\cup\mathcal{E}} t \sigma \).

\text{(Proof)}

Induction on the derivation of \( \mathcal{R}, \mathcal{E} \vdash s \approx t \Leftarrow t_1 \rightarrow s_1 \land \cdots \land t_n \rightarrow s_n \).

## 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 \textit{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 \( t \in T(\mathcal{G}) \), \( s \rightarrow_{\mathcal{R}} t \) if and only if \( s \rightarrow_{\mathcal{R}'} t \).

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 \textit{equivalent transformation sequence} under \( \mathcal{E} \) is a sequence \( \mathcal{R}_0, \ldots, \mathcal{R}_n \) of CTRSs (over \( \mathcal{F}_0, \ldots, \mathcal{F}_n \), 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)

\[
\begin{align*}
R, E \vdash s \approx s & \quad R, E \vdash s \approx t \leftarrow C \\
R, E \vdash s \approx t \leftarrow C & \quad R, E \vdash s \approx u \leftarrow C_1 \quad R, E \vdash u \approx t \leftarrow C_2 \\
R, E \vdash t \approx s \leftarrow C & \quad R, E \vdash s \approx t \leftarrow C_1 \land C_2
\end{align*}
\]

\[
R, E \vdash l \sigma \approx r \sigma \text{ if } l \sim r \in \mathcal{E}
\]

\[
R \vdash s_1 \sigma \approx t_1 \sigma \quad \ldots \quad R \vdash s_n \sigma \approx t_n \sigma \text{ if } l \rightarrow r \Rightarrow l \sim s_i \in R
\]

\[
R, E \vdash s \approx t \leftarrow t_1 \rightarrow x_1 \land \cdots \land t_n \rightarrow x_n \text{ if } s = t[x_1 \mapsto t_1 | 1 \leq i \leq n] \text{ and } \forall i, x_i \notin \text{Var}(s)
\]

\[
R, E \vdash s \approx t \leftarrow t_1 \rightarrow s_1 \land C \quad R, E \vdash t \approx t_1' \rightarrow s_1 \land C \text{ if } t_1 \sim t_1'
\]

\[
R, E \vdash s \approx t \leftarrow c(t_1, \ldots, t_n) \rightarrow c(s_1, \ldots, s_n) \land C \text{ if } c \in F
\]

\[
R, E \vdash s \approx t \leftarrow C \quad R, E \vdash t \approx s' \rightarrow s' \land C \text{ if } s' \in T(F', F')
\]

(I) Introduction

\[
R_{k+1} = R_k \cup \{ f(x_1, \ldots, x_n) \rightarrow r \}
\]

provided that \( f \notin F_k \), and \( r \in T(F_k, \{x_1, \ldots, x_n\}) \), where \( x_1, \ldots, x_n \) are mutually distinct variables. We put \( F_{k+1} = F_k \cup \{ f \} \).

(A) Addition

\[
R_{k+1} = R_k \cup \{ l \rightarrow r \leftarrow C \}
\]

provided \( R_k, E \vdash l \approx r \leftarrow C \).

(E) Elimination

\[
R_{k+1} = R_k \setminus \{ l \rightarrow r \}
\]

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

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

WPTE’14
Verifying the Correctness of Tupling Transformations based on Conditional Rewriting

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 $\sigma$ on $\mathcal{G}$, $l \sigma \leftrightarrow^R r \sigma$.

▶ Theorem 3 (Show the equivalence by equivalent transformations). Let $\mathcal{G}$ and $\mathcal{G}'$ be sets of function symbols such that $\mathcal{F}_\text{C} \subseteq \mathcal{G}, \mathcal{G}' \subseteq \mathcal{F}$. 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 $\mathcal{R}, \mathcal{G} \vdash_{\text{ind}} \mathcal{E}$ and there exists an equivalent transformation from $\mathcal{R}$ to $\mathcal{R}'$ under $\mathcal{E}$. Then, $\text{CR}(\mathcal{R}) \land \text{SC}(\mathcal{R}, \mathcal{G}) \land \text{SC}(\mathcal{R}', \mathcal{G}')$ imply $\mathcal{R} \sim_{\mathcal{G} \cap \mathcal{G}'} \mathcal{R}'$.

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

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

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 $\mathcal{R}_{\text{fib}}''$ by the Maude Sufficient Completeness Checker$^3$ (Maude-SCC, for short). It, however, is failed to show it. One can manually show sufficient completeness of $\mathcal{R}_{\text{fib}}''$ by induction with existential quantifier like $\forall n. \exists n', m'. \text{fibpair}(n) \rightarrow_{\mathcal{R}_{\text{fib}}} \langle n', m' \rangle$.

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


$^3$ http://maude.cs.uiuc.edu/tools/scc/