Incremental Proofs of Operational Termination with Modular Conditional Dependency Pairs

Masaki Nakamura, Kazuhiro Ogata and Kokichi Futatsugi

Abstract—OBJ algebraic specification languages support semi-automated verification of algebraic specifications based on equational reasoning by term rewriting systems (TRS). Termination guarantees any execution of the specification terminates in finite times. Another important feature of OBJ languages is a module system with module imports to describe large and complex specifications in a modular way. In this study, we focus on a way to prove termination of OBJ specifications incrementally, based on the notion of modular conditional dependency pairs (MCDP).

Index Terms—Term rewriting, Operational termination, Conditional Dependency Pairs, Algebraic specification, OBJ languages.

I. INTRODUCTION

OBJ languages [1], [2], [3], [4] are algebraic specification languages which support several useful advanced features, e.g., module system, typing system with ordered sorts, mix-fix syntax, for describing specifications, and a powerful interactive theorem proving system based on term rewriting systems (TRSs). The following is an example of CafeOBJ specifications, which is one of the active OBJ languages:

mod! NAT+
{ [Nat]
  op 0 : -> Nat
  op s_ : Nat -> Nat
  op _+ : Nat Nat -> Nat
  vars M N : Nat
  eq 0 + N = N .
  eq s M + N = s(M + N) .
}

mod! NAT*
{ pr(NAT+)
  op _* : Nat Nat -> Nat
  vars M N : Nat
  eq 0 * N = 0 .
  eq s M * N = (M * N) + N .
}

where the specification consists of two modules: NAT+ and NAT*. The first module NAT+ represents natural numbers with the addition, where the sort Nat stands for the set of natural numbers, the operation symbol (constant) 0 is zero and s is the successor function. Thus, the terms 0, s 0, s s 0, . . . , s n 0, . . . stand for 0, 1, 2, . . ., n, . . . respectively. The binary infix operation symbol + stands for the addition on natural numbers, and defined by the two equations. The other module NAT* represents the multiplication. Note that NAT+ imports NAT* by the import declaration pr (NAT+). In the importing module, the elements of the imported module can be used. The sort Nat and the operation symbols 0, s, + are used for declaring the multiplication symbol * and two equations for it in NAT*.

The theory of term rewriting systems (TRS) gives us an efficient automated equational reasoning. In theTRS, equations in the modules are regarded as left-to-right rewrite rules, and a given term is reduced by applying those rules until it cannot. For example, s s 0 + s 0 is reduced into s s s 0 as follows: s s 0 + s 0 → s (s 0 + s 0) → s s (0 + s 0) → s s s 0, where the underlined subterms are matched with some left-hand sides of the rewrite rules and replaced with the instances of the corresponding right-hand sides. The first two rewrite steps come from the second equation and the last rewrite step is obtained by the first equation in NAT+. Moreover, s s 0 * s 0 is reduced into s s 0 as follows: s s 0 * s 0 → (s 0 * s 0) + s 0 → ((0 * s 0) + s 0) + s 0 → (0 + s 0) + s 0 → s 0 + s 0 → s (0 + s 0) → s s 0. Note that the equations declared not only in NAT* but in NAT+ are used.

Termination is one of the most important properties of TRSs, which guarantees that execution of the specification must terminate in finite times. Note that in NAT+ terms are reduced by applying the equations in both NAT+ and NAT*. To prove termination of NAT+, we need to consider all equations in both NAT+ and NAT*. In general, termination is not a modular property, that is, for two terminating TRSs R0 and R1, the combination R0 ∪ R1 is not always terminating even if they have no shared operation symbols. It can be seen in the following famous Toyama’s counter-example: for R0 = \{ f(a, b, x) → f(x, x, x) \} and R1 = \{ g(x, y) → x, g(x, y) → y \} where f is a ternary operation symbol, g is a binary operation symbol, a and b are constant symbols, and x and y are variables, then we have the following infinite rewrite sequence: f(a, b, g(a, b)) → R0 f(g(a, b), g(a, b), g(a, b)) → R1 f(a, g(a, b), g(a, b)) → R1, f(a, b, g(a, b)) → · · · even if each Rk is terminating (i = 0, 1).

Modularity of termination has been studied and several kinds of conditions in which termination can be modular have been proposed [5], [6]. Recently, an incremental approach to termination proofs has been proposed [7]. Roughly speaking, in the incremental approach, (1) assume (a kind of) termination of the imported module, and (2) prove some well-founded properties of the importing module, then termination of the whole specification is proved. The method proposed in

Manuscript received December 4th, 2012; revised December 21th, 2012. This work was supported in part by 23220002 Grant-in-Aid for Scientific Research (S) 23220002 from Japan Society for the Promotion of Science (JSPS) and Grant-in-Aid for Young Scientists (B) 22700027 from Ministry of Education, Culture, Sports, Science and Technology (MEXT) Japan.
Masaki Nakamura is with the Department of Information Systems Engineering, Toyama Prefectural University, Japan.
Kazuhiro Ogata and Kokichi Futatsugi are with the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST).
[7] seems to be suitable for proving termination of OBJ spec-
ifications, however, practical OBJ specifications often include
equational conditions which are not treated in [7], where a
conditional equation is an equation with a guard condition.
In this study, we extend the incremental termination method
proposed in [7] in order to cover conditional equations.

II. PRELIMINARIES

In this section, we introduce the notion of term rewriting
systems [5].

A. Signature and terms

A signature is a set of operation symbols. An operation
symbol has its arity \( n \in \mathcal{N} \), where \( \mathcal{N} \) denotes the set of all
natural numbers. We write the set of operation symbols like
\( \Sigma = \{ f_0/n_0, f_1/n_1, \ldots \} \) where the operation symbol \( f_i \) has
the arity \( n_i \). We omit the arities and write \( f \) instead of
\( f/n \) if no confusion arises. For a signature \( \Sigma \) and a countable
set \( X \) of variables, the set \( T(\Sigma, X) \) (abbr. \( T \)) of \( (\Sigma, X) \)-
terms is the smallest set satisfying the following: \( X \subseteq T \) and
\( f(t_1, \ldots, t_n) \in T \) if \( f/n \in \Sigma \) and \( t_i \in T \) (\( i = 1, 2, \ldots, n \)). We write \( c \) instead of \( c() \) with \( c/0 \in \Sigma \), and call it a constant
(for both symbol and term). Throughout this article, we may
use \( x, y, z \) as variables, \( f, g, h \) as operation symbols, \( a, b, c \)
as constant symbols, \( l, r, s, t, u \) as terms, without notice.

A substitution \( \theta \) is a map from \( X \) to \( T \). We write the term
obtained by replacing all variables \( x \) in a term \( t \) with the terms \( \theta(x) \) as \( t\theta \). A sequence \( w \in \mathcal{N}_+ \) of positive integers
represents a position of a term. \( O(t) \subseteq \mathcal{N}_+ \) is defined as the
smallest set satisfying the following: \( e \in O(t) \) (\( e \) : the empty
sequence) and \( i.p \in O(f(t_1, \ldots, t_n)) \) if \( 1 \leq i \leq n \) and
\( p \in O(t_i) \). An operation symbol at the position \( p \in O(t) \) in \( t \)
is defined as \( x_e = x, f(\ldots x_e \ldots) = f, \) and \( f(t_1, \ldots, t_n).p = (t_i)p \). We call \( t_e \) the root symbol of \( t \). A context \( C \) is a
term with the special symbol \( \square \) which is not included in
considered \( \Sigma \) and \( X \) and occurs only once in \( C \). The term
obtained by replacing \( \square \) with a term \( t \) is written as \( C[t] \). The
set of all variables in \( t \) is written as \( \text{Var}(t) \).

Example 2.1: Let \( \Sigma_e = \{ +/2, s/1, 0/0 \} \) and \( \Sigma_s = \{ \ast/2 \} \). Examples of terms are \( 0, (s(0)), +(x, 0), \)
\( +((s(0)), 0) \), etc. Let \( t = \{ +(x, s(y)) \} \). \( t \theta = \)
\( +((s(0)), 0) \) when \( \theta(x) = s(0) \) and \( \theta(y) = 0 \). \( C[l] = \)
\( s(s(x), +x(s(y))) \) when \( C = s(s(x), \square) \), and \( \text{Var}(t) = \)
\( \{ x, y \} \).

B. Conditional rewrite rules and rewrite relation

We formalize conditional term rewriting systems (CTRS)
corresponding to CafeOBJ. In CafeOBJ, each module implicitly
imports a built-in Boolean module \( \text{BOOL} \). \( \text{BOOL} \) has the
sort \( \text{Bool} \) and the constants \( \text{true} \) and \( \text{false} \), and usual
logical operations and, or, not, etc., on \( \text{Bool} \). A CafeOBJ
conditional equation has the form of \( \text{ceq}(l \iff r) \), where \( l \)
is a term of \( \text{Bool} \), and means that the body equation \( l = r \)
holds whenever \( c \) holds. In the reduction, the instance \( \theta(l) \) is
replaced with \( \theta(l) \) when \( \theta(l) \) is reduced into \( \text{true} \). Note that
\( \text{Var}(r) \cup \text{Var}(c) \subseteq \text{Var}(l) \). Unconditional equations \( eq(l \iff r \iff c) \) can be considered as \( l \iff r \iff c \). \( \text{A } (\Sigma, X)\)-conditional rewrite rule is a triple \((l, r, c) \),
denoted by \( l \iff r \iff c \), such that \( l, r, c \in T(\Sigma, X) \) and \( \text{Var}(r) \cup \text{Var}(c) \subseteq \text{Var}(l) \). A
conditional term rewriting system (CTRS) is a pair of a
signature \( \Sigma \) and a set of \((\Sigma, X)\)-conditional rewrite rules.

1 We may call just a conditional rewrite rule, a rewrite
rule or a rule, instead of a \((\Sigma, X)\)-conditional rewrite rule
if no confusion arises. A CTRS \((\Sigma, R_b)\) for BOOL is
defined as \( \Sigma_b = \{ \text{true}/0, \text{false}/0, \text{not}/1\ldots \} \) and
\( R_b = \{ \text{true}/0 \rightarrow \text{false}/0, \text{not}/\ldots \rightarrow \text{true} \ldots \} \),
and hereafter we assume that every CTRS \((\Sigma, R)\) implicitly
includes \((\Sigma_b, R_b)\), that is, \( \Sigma_b \subseteq \Sigma \) and \( R_b \subseteq R \). We may
omit \( \Sigma \) and write \( R \) as a CTRS instead of \((\Sigma, R)\). A rewrite
relation \( \rightarrow_R \) is defined as follows: \((1) \) \( s \rightarrow t \) if there exists
\( l \rightarrow r \iff c \in R \) and a substitution \( \theta \) such that \( s = C[l\theta], \)
\( t = C[r\theta], \) and \( \theta(c) \) is \( \text{true} \). \((2) \) \( s \not\rightarrow t+1 \) if there exists
\( l \rightarrow r \iff c \in R \) and a substitution \( \theta \) such that \( s = C[l\theta], \)
\( t = C[r\theta], \) and \( \theta(c) \not\rightarrow \iff \text{true} \), where \( \not\rightarrow \) stands for
the reflexive and transitive closure of a binary relation \( \rightarrow \).

Example 2.2: Consider the following CafeOBJ module:
\[
\begin{align*}
\text{mod! EVEN};
\text{pr (Nat+)}; \\
\text{op even : Nat} \rightarrow \text{Bool}; \\
\text{var N : Nat}; \\
\text{eq even(0) = true}; \\
\text{ceq even(s N) = true if odd(N)}; \\
\text{ceq odd(s N) = true if even(N)}.
\end{align*}
\]
\(C_t\) is defined as follows:
\(R_b = \{ \text{even(0)} \rightarrow \text{true}, \text{even(s N)} \rightarrow \text{true if odd(N)}, \text{odd(s N)} \rightarrow \text{true if even(N)} \} \).

We have \(\text{even}(s(0)) \rightarrow_R \text{true since } \text{even}(0) \rightarrow \text{true, odd}(s(0)) \rightarrow_1 \text{true, and } \text{even}(s(0)) \rightarrow_2 \text{true.}\)

III. OPERATIONAL TERMINATION AND CONDITIONAL
DEPENDENCY PAIRS

A CTRS \( R \) is terminating if there is no infinite rewrite sequence
\( l_0 \rightarrow_R l_1 \rightarrow_R l_2 \rightarrow_R \cdots \). Although termination is
one of the most important properties of CTRS, it does not
directly correspond to termination of computation of terms.
Consider \( R = \{ a \rightarrow b \iff a \} \). By removing the condition we have
\( R' = \{ a \rightarrow b \} \) and \( R' \) is trivially terminating and
thus \( R \) is also terminating since in general \( \rightarrow_R \) is a subset of
\( \rightarrow_{R'} \) when \( R' \) is a (unconditional) TRS obtained from
\( R \) by removing all condition parts. However, computation of
reducing \( a \) does not terminate since when try to apply
\( a \rightarrow b \iff a \) to the target term \( a \), the condition \( a \) should
be checked whether it can be reduced into \( \text{true} \), and the
procedure fails into an infinite calls of the condition. To
capture the above non-terminating behaviors, the notion of
operational termination has been proposed [8], which is
defined by infinite well-formed trees in a logical inference sys-
tem of conditional rewrite relation instead of infinite rewrite
sequences. By the notion of operational termination, we can
guarantee the absence of both infinite rewrite sequences and
infinite condition calls. From the space limitation, we omit
the precise definition of operational termination. Instead,
we introduce an equivalent proposition on context-sensitive rewriting later in this section.

A. Conditional dependency pairs

The notion of dependency pairs is one of the most powerful methods to prove termination of (unconditional) TRS [9], where essential pairs of terms are extracted from rewrite rules and chains of the pairs are analyzed for proving termination. We redefine the notion of dependency pairs for CTRS. An operator symbol at the root position of the left-hand side of some rewrite rule is called a defined symbol, that is, \( D_R = \{ f \in \Sigma \mid f(\ldots) \rightarrow r \iff c \in R \} \). The marked symbol of \( f \) is defined as \( f^\# \) and the set of marked symbols of \( \Sigma \) is written as \( \Sigma^\# \). The marked term \( t^\# \) of a non-variable term \( t = f(t_1, \ldots, t_n) \) is the term obtained by renaming only the root symbol, defined as \( t^\# = f(t_1^\#, \ldots, t_n^\#) \). An ordinary dependency pair is a pair \((t^\#, u^\#)\) of the marked left-hand side and a marked subterm in the right-hand side whose root symbol is defined. For operational termination, we need to consider the condition part. Thus, besides the right-hand-side, a marked subterm \( u^\# \) of the condition \( c \) should be considered.

**Definition 3.1:** Let \( R \) be a CTRS. The set \( CDP(R) \) of all conditional dependency pair (CDP) of \( R \) is defined as follows:

\[
CDP(R) = \left\{ (l^\#, u^\#) \iff c \mid l \rightarrow r \iff c \in R, \quad r = C[u], u_c \in D_R \right\} \cup \left\{ (l^\#, u^\#) \iff true \mid l \rightarrow r \iff c \in R, \quad c = C[u], u_c \in D_R \right\}
\]

We may write \((s, t)\) instead of \((s, t) \iff true\).

**Definition 3.2:** Let \( R \) be a CTRS. A (possibly infinite) sequence \((l_i^\#, u_i^\#) \iff c_i (i = 0, 1, 2, \ldots)\) of pairs of \( CDP(M) \) is called a dependency chain of \( R \) if there exist \( \theta_i (i = 0, 1, 2, \ldots)\) such that (1) \( c_i \theta_i \rightarrow_R^* true \), and (2) \( u_i^\# \theta_i \rightarrow_R l_i^\# \theta_{i+1}\) for each \( i \in \mathbb{N} \).

The following sufficient condition of operational termination holds.

**Theorem 3.3:** A CTRS \( R \) is operationally terminating if and only if there exists no infinite chain of \( R \).

**Example 3.4:** While \( R_e \) does not have any ordinary dependency pair [9] since any right-hand side does not have defined symbols, it has conditional dependency pairs since defined symbols occur in the conditions. We have \( CDP(R_e) = \{(even^\#(s(x)), odd^\#(x)), (odd^\#(s(x)), even^\#(x))\}\). There is no infinite chain of \( R_e \) since any chain should be in the form of \((even^\#(s_0), odd^\#(t_0))\) \((odd^\#(s_1), even^\#(t_1))\) \((even^\#(s_2), odd^\#(t_2))\) \ldots and the argument of each CDP should decrease \((s_i > t_i)\) because of \((s(x) > x)\) and the connected terms are equivalent \((t_i = s_{i+1})\) in the meaning of the model of \( \text{NAT}^+ \). The strict order > on natural number is well-founded, i.e., there is no decreasing sequence \( n_0 > n_1 > n_2 > \cdots \). Similarly, we can see that \( R_p \) does not have infinite dependency chains, and \( R_e \cup R_p \) is operationally terminating.

B. Proof of Theorem 3.3

To prove Theorem 3.3, the notion of context-sensitive rewriting (CSR) [10] and the transformation from CTRS to CSR [11] are useful. We introduce the notations and definitions of them. Let \( R \) be a (unconditional) TRS. A map \( \mu \) from \( \Sigma \) to \( P(\Sigma) \) is called a replacement map if \( \mu(f) \subseteq \{1, 2, \ldots, ar(f)\} \) for each \( f \in \Sigma \). The set \( O_\mu(t) \) of replacement positions of \( t \) is defined as \( \varepsilon \in O_\mu(t) \) and \( i_p \in O_\mu(f(t_1, \ldots, t_n)) \) if \( i \in \mu(f) \) and \( p \in O_\mu(t_i) \). A context-sensitive rewrite relation of \( \mu \), denoted by \( \rightarrow_{\mu} \), is defined as follows: \( s \rightarrow_{\mu} t \) if and only if there exists \( l \rightarrow r \in R \) and \( \theta \) such that \( s = C[\theta], t = C[\theta], G_p = \varepsilon, \) and \( p \in O_\mu(t_i) \).

The following unraveling technique with CSR can simulate computation of CTRS completely.

**Definition 3.5:** [11] Let \((\Sigma, R)\) be a CTRS. The (unconditional) TRS \((\Sigma \cup \Sigma_U, U_c(R))\) and the replacement map \( \mu_U \) are defined as follows: Let each conditional rule in \( R \) be labeled like \( \alpha : l \rightarrow r \iff c \).\n
\[
\begin{align*}
\Sigma_U & = \{ U_\alpha \mid \alpha : l \rightarrow r \iff c \in R \} \\
U_c(R) & = \{ l \rightarrow U_\alpha(c, x_1, \ldots, x_n) \mid \alpha : l \rightarrow r \iff c \in R \} \\
\mu_U(f) & = \{ \{1, \ldots, ar(f)\} \iff f \in \Sigma \} \\
\{1\} & \iff f \in \Sigma_U
\end{align*}
\]

where \( Var(l) = \{x_1, \ldots, x_n\} \).

Hereafter, we may write \( s \rightarrow_{U} t \) instead of \( s \rightarrow_{\mu_U} t \).

**Example 3.6:** \( R_e \) in Example 2.2 is unravelled as follows:

\[
\begin{align*}
U_{\alpha}(R_e) & = \{ even(0) \rightarrow true, even(s(x)) \rightarrow U_e(odd(x), x), odd(s(x)) \rightarrow U_e(even(x), x), U_e(true, x) \rightarrow true \} \\
U_{\alpha}(true, x) & \rightarrow true
\end{align*}
\]

A term \( even(s(0)) \) is reduced into \( true \) as follows:

\[
\begin{align*}
even(s(0)) & \rightarrow_{\mu} U_e(odd(s(0)), s(0)) \\
& \rightarrow_{\mu} U_e(U_e(odd(s(0)), s(0)) \\
& \rightarrow_{\mu} U_e(U_e(true, 0), s(0)) \\
& \rightarrow_{\mu} U_e(true, s(0)) \\
& \rightarrow_{\mu} true
\end{align*}
\]

A TRS is called \( \mu \)-terminating if there is no infinite rewrite sequence \( l_0 \rightarrow_{U}^* t_1 \rightarrow_{U}^* t_2 \rightarrow_{U}^* \cdots \). Operational termination of CTRS and \( \mu \)-termination of the transformed TRS have been shown to be equivalent in [11].

**Proposition 3.7:** [11] Let \( R \) be a CTRS. (1) If \( s \rightarrow_R t \), then \( s \rightarrow_{U}^* t \). (2) \( R \) is operationally terminating if and only if \( U_c(R) \) is \( \mu_U \)-terminating on \( T(\Sigma, X) \).

Now we give a proof of Theorem 3.3.

**Theorem 3.3:** A CTRS \( R \) is operationally terminating if and only if there exists no infinite chain of \( R \).

**Proof:** (only if part): We will prove by contraposition. Assume an infinite chain \((l_i^\#, u_i^\#) \iff c_i (i \in \mathbb{N})\) exists. From Definition 3.1 and 3.5, for each \((l_i^\#, u_i^\#) \iff c_i \), there exist rewrite rules \( l_i \rightarrow U_c(\alpha_i, x_1, \ldots, x_n) \) and \( U_c(\alpha_i, x_1, \ldots, x_n) \rightarrow C[u_i] \) in \( U_c(R) \). From Definition 3.2 (1), \( c_i \theta_i \rightarrow_R^* true \) for some \( \theta_i \) and thus \( c_i \theta_i \rightarrow_U^* true \) from Proposition 3.7 (1). Thus, we have

\[\text{true} \]
\[ l_{\theta_1} \rightarrow U_{\alpha_1}(c_1, \theta_1, \bar{x}) \rightarrow l_{\theta_1} U_{\alpha_1}(\text{true}, \bar{x}) \rightarrow U_{\alpha_1} C_1[u_\theta_1] = C_1[\theta_1, u_\theta_1], \]

From Definition 3.2 (2), \( u_\theta \rightarrow R_{\theta} l_{\theta + 1} l_{\theta + 1 + 1} \) and thus \( u_{\theta} \rightarrow_R l_{\theta} l_{\theta + 1 + 1}. \) Note that \( s^# \rightarrow_R t^# \) implies \( s \rightarrow_R t \) since \( R \) does not have any marked symbols \( f^#. \) Then, we have an infinite rewrite sequence \( l_{\theta_0} \rightarrow C_0 \theta_0[u_\theta_0] \rightarrow C_0 \theta_0[l_{\theta_1}] \rightarrow \cdots C_0 \theta_0[l_{\theta_1}] \rightarrow \cdots \rightarrow C_0 \theta_0[\{C_1 \theta_1[l_{\theta_1}], \cdots, C_0 \theta_0[u_\theta_0]\}] \rightarrow U_{\alpha_1}. \) Since \( l_{\theta_0} \) belongs to the original \( T(\Sigma, X), R \) is not operationally terminating from Proposition 3.7 (2).

**(if part):** Assume \( R \) is not operationally terminating. \( U_{\alpha_1}(R) \) is not \( \mu_U \)-terminating from Proposition 3.7 (2). We take a minimal infinite rewrite sequence \( \mu_U \)-termination: \( l_{\theta_0} \rightarrow U_{\alpha_0}(R) \rightarrow \cdots \rightarrow R \rightarrow \cdots \rightarrow X \rightarrow \cdots \) where \( \mu_U \) is not infinite rewrite sequence \( l_{\theta_0} \rightarrow U_{\alpha_0}(R) \rightarrow \cdots \rightarrow R \rightarrow \cdots \). From the minimality, \( \alpha_0 \subseteq U_{\alpha_1}(R) \). Then, we take \( l_{\theta_0} \rightarrow U_{\alpha_1}(R) \) as the first CDP of the chain. From Proposition 3.7 (1), \( l_{\theta_0} \rightarrow U_{\alpha_1}(R) \) is \( \mu_U \)-true holds. Note that \( \theta_0 \) is a module. The set \( \alpha_0 \subseteq U_{\alpha_1}(R) \).

**IV. INCREMENTAL PROOFS OF OPERATIONAL TERMINATION**

**A. Hierarchical extension**

The notion of the hierarchical extension has been defined for TRSs in [7]. We give a straightforward extension to CTRSs as follows:

**Definition 4.1:** A pair \( [\Sigma_1 | R_1] \) is called a module extending a CTRS \( (\Sigma_0, R_0), \) denoted by \((\Sigma_0, R_0) \leftarrow [\Sigma_1 | R_1], \)

1. \( \Sigma_0 \cap \Sigma_1 = \emptyset, \)
2. \((\Sigma_0 \cup \Sigma_1, R_1) \) is a CTRS and
3. \( D_{R_1} = \Sigma_1. \) The CTRS \((\Sigma_0 \cup \Sigma_1, R_0 \cup R_1) \) is called a hierarchical extension of \((\Sigma_0, R_0) \) with module \( [\Sigma_1 | R_1]. \)

We denote \((\Sigma_0, R_0) \leftarrow [\Sigma_1 | R_1] \leftarrow [\Sigma_2 | R_2] \) when \([\Sigma_2 | R_2] \) extends \((\Sigma_0 \cup \Sigma_1, R_0 \cup R_1). \) A CTRS \((\Sigma_0, R_0) \) can be regarded as a module \([\Sigma_0 | R_0] \) extending the empty CTRS \((\emptyset, \emptyset)\). Hereafter we may use the module expression for an ordinary CTRSs.

**Example 4.2:** Let \( M_k = [\Sigma_k | R_k] \) and \( M_n = [\Sigma_n | R_n] \) such that \( \Sigma_k = \{0, s, \} \) declared in \( \text{NAT}_+, \) \( \Sigma_n = \{\} \) declared in \( \text{NAT}_+, R_k \) is the TRS which has two rewrite rules in \( \text{NAT}_+, \) and \( R_n \) is the TRS which has two rewrite rules in \( \text{NAT}_+. \) Then, \( M_k \leftarrow M_k \) since \( \Sigma_k \cap \Sigma_n = \emptyset \) and the root symbols of the two rewrite rules in \( R_k \) is \( * \in \Sigma_k. \) Also \( M_k \leftarrow M_k = \{\Sigma_k \mid R_k\} \) for \( \Sigma_k = \{\text{even}, \text{odd}\} \) \( R_k \) and \( R_n \) in Example 2.2.

**B. Modular Conditional Dependency Pairs**

To give a sufficient condition of operational termination in a modular way, we introduce conditional dependency pairs of a module, which is defined by ignoring the subterms \( u \) whose root symbol is not in the module in ordinary CDP defined above. While \((u^#(m, n), u^#(n, s(m, n))) \) belongs to \( \text{CDP}(R_1 \cup R_2) \) since \( e \in D_{R_1}, R_2 \), it is not considered for \( M_k = [\Sigma_k | R_k] \) since \( e \notin D_{R_k}. \)

**Definition 4.3:** Let \( M = [\Sigma | R] \) be a module. The set \( \text{MCSPD}(R) \) of all conditional dependency pair of module \( M \) (abbr. MCSPD) is defined as follows:

\[ \text{MCSPD}(R) = \{ (l^#, u^#) \mid l \rightarrow r \in R, r \in [\text{C}[a], u \in D_R \} \]

The dependency chain of MCSPD is defined as a sequence of MCSPDs which are connected by the rewrite relation of some arbitrary CTRS \( S, \) which may be \( R_1 \cup R_2, \) for example.

**Definition 4.4:** Let \( M = [\Sigma | R] \) be a module and \( S \) be a CTRS. A (possibly infinite) sequence \( \{l_i^#, u_i^#\} = c_i, (i = 0, 1, 2, \ldots) \) of pairs of \( \text{MCSPD}(M) \) is called a dependency chain of \( M \) over \( S \) if there exist \( \theta_i \) \( (i = 0, 1, 2, \ldots) \) such that \( 1 \theta_i \rightarrow \theta_{i+1} \) true and \( 2 \theta_i \rightarrow \theta_{i+1} \) true for all \( i \in N. \)

The ordinary CDP \( (R) \) and the dependency chain are equivalent to \( \text{MCSPD}([\Sigma | R]) \) and the dependency chain of \([\Sigma | R] \) over \( R \) respectively. Collapse-extended termination \( (CE\text{-termination}) \) is a subclass of termination, which plays important role in proving termination in a modular way [6]. A CTRS \( R \) is \( CE \text{-operationally} \) terminating if \( R \cup \pi \) is (operationally) terminating, where \( \pi = \{G(x, y) \rightarrow x, G(x, y) \rightarrow y\} \) with a special fresh binary operation symbol \( G/2, \) which means that \( G \notin \Sigma \) for any considered \( \Sigma. \) Our main theorem is that \( CE \text{-operational} \) termination of \( R_0 \) and the absence of infinite chains of \( R_1 \) over \( R_0 \cup R_1 \) imply \((CE\text{-operational}) \) termination of \( R_0 \cup R_1. \)

Before the main theorem, we show the following theorem between \( CE \text{-operational} \) termination and MCDP.

**Theorem 4.5:** A CTRS \((\Sigma, R) \) is \( CE \text{-operationally} \) terminating if and only if there exists no infinite chain of \([\Sigma | R] \) over \( R \cup \pi. \)

**Proof:** Since \( G \) is fresh, \( R \) does not have \( G \) and \( \text{MCSPD}([\Sigma | R]) = \text{MCSPD}([\Sigma | G/2] \cup R \cup \pi. \)

The following is a main theorem of this paper.

**Theorem 4.6:** Let \([\Sigma_0 | R_0] \leftarrow [\Sigma_1 | R_1]. \) If \( 1 \) \( R_0 \) is \( CE \text{-operationally} \) terminating, and \( 2 \) there exists no infinite chain of \([\Sigma_2 | R_2] \) \( R_0 \cup R_1 \cup \pi, \) then \( R_0 \cup R_1 \) is \( CE \text{-operationally} \) terminating.

**Proof:** (Sketch) It can be proved by the same proof strategy with the unconditional version of this theorem in the literature [7] (Theorem 1). An infinite chain of \( \text{CDP}(R_0 \cup R_1 \cup \pi) \) consists of either

(a) those of \( \text{MCSPD}([\Sigma_0 | R_0]). \)
of each \( f/n \in \Sigma \) with a quasi-ordering \( \geq \) (a reflexive and transitive binary relation on \( \Sigma \)) such that every algebra operation is weakly monotone in all of its arguments, that is, for each \( f/n \in \Sigma \) and \( a, b \in A \) with \( a \geq b \) we have \( f(a, \ldots, a, b) \geq f(a, \ldots, b) \). A weakly monotone \( \Sigma \)-algebra \( (A, \geq) \) is well-founded if \( \geq \) is well-founded. A map \( f: X \to A \) is called an assignment, and it can be extended to a map \( a: \Sigma \to A \) such that \( a(f(t_0, \ldots, t_m)) = f(a(t_0), \ldots, a(t_n)) \). The order \( \geq \) on \( T \) is defined as \( t \geq t' \) if \( \forall a: X \to A, a(t) \geq a(t') \). The stable-strict ordering \( > \) is defined as \( s > t \) if and only if \( s \geq t \) and \( t \Sigma s \) for each ground substitution \( \theta : X \to \Sigma, \theta \). Then, for a weakly monotone well-founded \( \Sigma \)-algebra \( (A, \geq) \), a pair \((A, \geq)\) is a weak reduction pair [6].

**Example 5.4:**

1) For \( R_+ \), a \( \Sigma_+ \cup \Sigma_0 \cup \{G/2\} \)-algebra \( (A, \geq) \) is defined as follows: \( A = N \), \( A_0 = 0 \), \( A_x(x, y) = x + 1 \), \( A_x(x, y) = x + y \), \( A_x(x, y) = x \), and \( A_{G/2}(x, y) = x + y \). Then, for an assignment \( a \), all \( t \in R_+ \) are interpreted to equations \( a(l) = a(r) \). For \( \pi = \{G(x, y) \to x, G(x, y) \to y\} \), \( A(G(x, y)) = (x + y) \geq (x) \). For \( (s)^m(s(m, n)), (s)^m(m, n)) \in MCDP(M_+), \) we have

\[
(a(s(m, n))) = a(s(m)) = (a(m) + 1 > a(m) = (a(s(m, n))).
\]

Then, \((A, \geq)\) is a weak reduction pair compatible with \( M_+ \) over \( R_+ \), and from Theorem 4.6 and 5.3, \( R_+ \) is \( C_E \)-operationally terminating. Note that for \((\emptyset, \emptyset) \leftrightarrow M_-, \) the empty CTRS is trivially \( C_E \)-operationally terminating.

2) Next consider \( M_\times \leftrightarrow M_-, \) \( \Sigma \cup \Sigma_0 \cup \Sigma_0^\emptyset \cup \{G/2\} \)-algebra \( (A, \geq) \) is defined as follows: the same interpretation is given for \( \Sigma_+ \) and \( G/2 \) with \( A = N \), \( A_x(x, y) = x \times y \), and \( A_{G/2}(x, y) = x \). Then, for an assignment \( a \), all \( t \in R_- \) are interpreted to equations \( a(l) = a(r) \). For \( (s)^m(s(m, n)), (s)^m(m, n)) \in MCDP(M_-), \) we have

\[
(a(s(m, n))) = a(s(m)) = (a(m) + 1 > a(m) = (a(s(m, n))).
\]

Then, \((A, \geq)\) is a weak reduction pair compatible with \( M_- \) over \( R_- \), and similarly we have that \( R_+ \cup R_- \) is \( C_E \)-operationally terminating.

3) Consider \( M_\times \leftrightarrow M_-, \) \( \Sigma \cup \Sigma_0 \cup \Sigma_0^\emptyset \cup \{G/2\} \)-algebra \( (A, \geq) \) is defined as follows: the same interpretation is given for \( \Sigma_+ \) and \( G/2 \) with \( A = N \), \( A_{\text{even}}(x) = 0 \) and \( A_{\text{odd}}(x) = 1 \). Then, for an assignment \( a \), all \( t \in R_- \) are interpreted to equations \( a(l) \geq a(r) \). For \( (s)^m(s(m, n)), (s)^m(m, n)) \in MCDP(M_-), \) we have

\[
(a(s(m, n))) = a(s(m)) = (a(m) + 1 > a(m) = (a(s(m, n))).
\]

Then, \((A, \geq)\) is a weak reduction pair compatible with \( M_- \) over \( R_- \). Thus, \( R_+ \cup R_- \) is \( C_E \)-operationally terminating.

**VI. APPLICATION TO PRACTICAL OBJ SPECIFICATIONS**

The notion of observational transition system (OTS) gives a way to describe a state machine in CafeOBJ, and OTS/CafeOBJ specifications have been used for modeling and analyzing several practical systems, for example, authentication protocols, e-government systems, etc [13], [14], [15]. An OTS/CafeOBJ specification

---

**Bibliographic Information:**


**Authors:**

- [Author 1]
- [Author 2]

**ISBN:** 978-988-19251-8-3

**ISSN:** 2078-0958 (Print); 2078-0966 (Online)
Fig. 1. An OTS/CafeOBJ specification of bank account systems.

<table>
<thead>
<tr>
<th align="left">mod* ACCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left">pr (INT + USER)</td>
</tr>
<tr>
<td align="left">* [Sys] *</td>
</tr>
<tr>
<td align="left">op init : -&gt; Sys</td>
</tr>
<tr>
<td align="left">bop balance : User Sys -&gt; Int</td>
</tr>
<tr>
<td align="left">bop deposit : User Int Sys -&gt; Sys</td>
</tr>
<tr>
<td align="left">bop withdraw : User Int Sys -&gt; Sys</td>
</tr>
<tr>
<td align="left">vars U U' : User</td>
</tr>
<tr>
<td align="left">var I : Int</td>
</tr>
<tr>
<td align="left">var A : Sys</td>
</tr>
<tr>
<td align="left">eq balance(U, init) = 0 .</td>
</tr>
</tbody>
</table>

Fig. 1. An OTS/CafeOBJ specification of bank account systems.

consists of a system module and data modules. A data module is a specification of an abstract data type used in the system, for example, integers with operations, enumerated types, user-defined structures, and so on. A system module is a specification of a state machine defined via observations. We show an example of OTS/CafeOBJ specifications: a specification of a bank account system (Fig.1). The module ACCOUNT imports INT and USER as data modules, where INT is a specification of integers and their operations, and USER is a specification of a user database. balance takes a state of the bank account system and returns the balance value of each user. Users can withdraw from and deposit in their account. For example, the third conditional equation eq balance(U, withdraw(U',I,A)) = (if U = U' then balance(U,A) - I else balance(U,A) fi) if balance(U,A) >= I and I >= 0 means that the balance of the user U' after withdrawing I is the remainder of subtracting I from the balance of the current state A if the balance of the user U' of the current state A is more than or equal to I and I is not negative, and that of the other user U (≠ U') is not changed.

Let M₀ = [S₀ | R₀] be the module corresponding to ACCOUNT. Then, the MCDP(M₀) consists of four conditional dependency pairs:

```
  \begin{align*}
  \text{bal}#(u, \text{dep}(u', i, a), \text{bal}#(u, a)) & \leq (i, 0) \\
  \text{bal}#(u, \text{with}(u', i, a), \text{bal}#(u, a)) & \leq \text{and}((\geq (\text{bal}#(u', a), i), \geq (i, 0)) \\
  \text{bal}#(u, \text{with}(u', i, a), \text{bal}#(u, a)) & \leq \text{true} \\
  \text{with}(u', i, a), \text{bal}#(u, a)) & \leq \text{true}
  \end{align*}
```

Note that we do not need to consider operation symbols =, ≥, +, -, if_then_else_fi as defined symbols for MCDP(M₀) even if they may be defined in the imported modules. The last MCDP can be ignored in an infinite chain because with# does not occur in any right-hand side of MCDP. For each of the remaining three MCDPs, the second argument of bal# strictly decreases in the meaning of the number of operation symbols dep and with. Thus, there is no infinite chain of MCDP, and if the imported modules are already proved Cₖₜ-operationally terminating, then the whole CTRS is Cₖₜ-operationally terminating. Formally, it can be proved by the notion of recursive path ordering (RPO) [5], for example.

VII. CONCLUSIVE REMARKS

One of our goals is to implement a light-weight termination checker in CafeOBJ. Computing MCDPs is easy to be implemented. There are several ordering on terms for automated checking which can be transformed to a weakly reduction pair, for example, RPO. Considering a large specification with several modules. (1) We first describe basic modules, that is, without imports, and prove Cₖₜ-operational termination of them and label them so. (2) Next we describe a module which imports modules labelled by "Cₖₜ-operational termination". Then, it suffices to show the absence of infinite chains of the module for proving Cₖₜ-operational termination of the whole system. One of the future work is to improve our results for other features of OBJ languages, e.g. order-sorted specifications, the evaluation strategy, operator attributes for associative and commutative axioms.

REFERENCES