On Proving Operational Termination Incrementally 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 one of the most important properties of TRSs. Termination guarantees that 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 [2], [3], [4], [5] 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! BASIC-NAT{
  [Zero NzNat < Nat]
  op 0 ; > Zero
  op s_ : Nat -> NzNat
}
```

The module BASIC-NAT represents natural numbers. Sorts are declared in the square brackets ([ ]). BASIC-NAT has the sorts Zero, NzNat and Nat. A partial order on sorts is declared with <. The partial order ≤ is the reflexive and transitive closure of <. In BASIC-NAT, Zero < Nat and NzNat < Nat are declared, and the partial order is ≤ = { (Zero,Zero), (Zero,NzNat), (NzNat,NzNat), (NzNat, Nat), (Nat,Nat) }. Sorts stand for sets in their models. Zero, NzNat and Nat stand for the singleton set {0}, the set of the positive numbers and the set of the natural numbers respectively. A partial order stands for an inclusion relation on the sets. In BASIC-NAT, two operation symbols are declared. The operation symbol 0 is a constant symbol of the sort Zero. The operation symbol s is a unary operation symbol which takes an element of Nat and returns an element of NzNat. 0 stands for zero, and s stands for the successor function on natural numbers. A term is a tree whose nodes are operation symbols and leaves are constants or variables. The terms 0, s 0, s s 0, ... stand for the natural numbers 0, 1, 2, ..., n, respectively.

The following is another CafeOBJ module:

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

The module NAT+ represents the addition on natural numbers. A module can import other modules. pr(BASIC-NAT) means that the module NAT+ imports the module BASIC-NAT. In the importing module, the elements in the imported modules can be used, that is, Zero, NzNat, Nat. 0, s can be used in NAT+. The binary infix operation symbol + stands for the addition on natural numbers. Variables are declared with vars. M and N are variables of Nat. Equations are declared with eq. The first equation means that 0 + t = 0 for any term t of Nat. The second means s t + t' = s(t + t') for any terms t, t' of Nat.

CafeOBJ supports a rewrite engine for equational reasoning. The following is the result of applying the CafeOBJ reduction command red to the term s s 0 + s s 0 in the module NAT+:

```
NAT+> red s s 0 + s s 0 .
-- reduce in NAT+ : (s s 0 + s s 0):Nat
(s s s s s 0):NzNat
```

The above result is a proof of the equation 2 + 3 = 5. The rewrite engine is implemented based on the theory of term rewriting systems (TRS). In the TRS, 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.

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. This article is an extended version of [1].

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).

(Advance online publication: 21 May 2013)
vars M N : Nat
eq 0 * N = 0 .

In NAT*, the term \( s * s * s \) is reduced into \( s * s \) as follows: 
\[
\begin{align*}
& s * s * s \rightarrow (s * s) * s \rightarrow (s * s) * s + s \rightarrow (0 * s) * s + s \rightarrow 0 * (s + s) + s \\
& \rightarrow s + s * s \rightarrow s (s + s) \rightarrow s * s .
\end{align*}
\]
Note that the equations in both NAT* and NAT+ are used.

Termination is one of the most important properties of TRSSs, 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 TRSSs \( R_0 \) and \( R_1 \), the combination \( R_0 \cup R_1 \) 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 \( R_0 = \{ f(ax, bx) \rightarrow f(xx, xx) \} \) and \( R_1 = \{ g(x, y) \rightarrow x, g(x, y) \rightarrow y \} \) 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: 
\[
\begin{align*}
& f(ax, bx) \rightarrow f(xx, xx) \rightarrow f(g(x, x), g(x, x)) \rightarrow f(g(x, y), g(ax, bx)) \rightarrow \ldots
\end{align*}
\]
even if each \( R_i \) 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 [6], [7]. Recently, an incremental approach to termination proofs has been proposed [8]. Roughly speaking, in the incremental approach, termination of a module with module imports is shown by (1) assuming (a subclass of) termination of the imported modules, and (2) proving some well-founded properties of the importing module. The method proposed in [8] seems to be suitable for proving termination of OBJ specifications, however, practical OBJ specifications often include conditional equations which are not treated in [8], where a conditional equation is an equation with a guard condition. In this study, we extend the incremental termination method in order to cover conditional equations.

II. PRELIMINARIES

In this section, we introduce the notion of term rewriting systems [6].

A. Signature and terms

A signature is a set of operation symbols. An operation symbol \( f \) has its arity \( ar(f) \in \mathcal{N} \), where \( \mathcal{N} \) denotes the set of all natural numbers. We write the set of operation symbols like \( \Sigma = \{ f_1/n_1, f_2/n_2, \ldots \} \) where the operation symbol \( f_i \) has the arity \( ar(f_i) = n_i \). We may 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: 
\( \Sigma \subseteq T \) 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 \( e \) instead of \( c() \) if \( c/0 \in \Sigma \), 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, \( r, s, t, u \) 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 \( \omega = \omega_1* \ldots * \omega_n \) of positive integers represents a position of a term. \( O(\omega) \subseteq \mathcal{N}_+^* \) is defined as the smallest set satisfying the following: 
\( \varepsilon \in O(\omega) \) ( \( \varepsilon \) 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 \( t \) is defined as \( x = x, f(\cdot, x) = f, \) and \( (f(t_1, \ldots, t_n))_p = (t_i)_p \). We call \( t_p \) the root symbol of \( t \). The term \( t_p \) is called a subterm of \( t \) at \( p \), denoted by \( t \not\geq t_p \) if \( p \in O(t) \), and called a strict subterm, denoted by \( t \not\geq t_p \) if \( p \not\in \emptyset \), \( i.e, \ t \not\geq t_p \) and \( t \not\geq t_p \). A context \( C \) is a term with the special symbol \( \square \) which is not included in considered \( \Sigma \) and \( X \) 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 \( Var(t) \).

Example 2.1: Let \( \Sigma_+ = \{ +/2, s/1, 0/0 \} \) and \( \Sigma_* = \{ */2 \} \). Examples of terms are \( 0, (s(0)), +(s(0)), s(y) \), and so on. \( s(0) \) is a strict subterm of \( +(s(0)) \). Let \( t = +(s(x), y) \) \( \theta = +(s(0), s(0)) \) when \( \theta(x) = s(0) \) and \( \theta(y) = 0 \). \( C[t] = s(s(x), +(s(x), y)) \) when \( C = *(s(x), \square) \), and \( Var(t) = \{ x, y \} \).

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 \( \text{and, or, not, etc.} \) on \( \text{Bool} \). A CafeOBJ conditional equation has the form of \( \text{eq} l = r \text{ if } c \) where \( c \) is a term of \( \text{Bool} \), and means that the body equation \( l \not\rightarrow \) holds whenever \( c \) holds. In the reduction, the instance \( \theta \) is replaced with \( r \theta \) when \( c \theta \) is reduced into \( \text{true} \). Note that \( Var(r) \cup Var(c) \subseteq Var(l) \). Unconditional equations \( eq l = r \) can be considered as \( l \not\rightarrow \) \( \text{true} \). A \( (\Sigma, X) \)-conditional rewrite rule is a triple \( (l, r, c) \), denoted by \( l \not\rightarrow c \), such that \( l, r, c \in T(\Sigma, X) \) and \( Var(r) \cup Var(c) \subseteq Var(l) \). A conditional term rewriting system (CTRS) is a pair of a signature \( \Sigma \) and a set of \( (\Sigma, X) \)-conditional rewrite rules \( \Gamma^c \). We write \( l \not\rightarrow r \) instead of \( l \not\rightarrow c \) when \( c = \text{true} \). 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 \( (\Sigma, R_0) \) for \( \text{BOOL} \) is defined as \( \Sigma_0 = \{ \text{true}/0, \text{false}/0, \text{not}/1, \ldots \} \) and \( R_0 = \{ \text{true}/0 \rightarrow \text{false}, \text{not}/\text{false} \rightarrow \text{true}, \ldots \} \), and hereafter we assume that every CTRS \( (\Sigma, R) \) implicitly includes \( (\Sigma_0, R_0) \), that is, \( \Sigma_0 \subseteq \Sigma \) and \( R_0 \subseteq R \). We may omit \( \Sigma \) and write \( R \) as a CTRS instead of \( (\Sigma, R) \). A rewrite relation \( \not\rightarrow R \) is defined as follows: \( l \not\rightarrow R c \) if there exists \( l \not\rightarrow r \overset{c}{\rightarrow} c \in R \) and a substitution \( \theta \) such that \( s = C[\theta] \).
t = C[θ], and cθ = true. (2) s →_{i+1} t if there exists l → r ⇐ c ∈ R and a substitution θ such that s = C[θ], t = C[θ], and cθ → t true, where →_t stands for the reflexive and transitive closure of a binary relation →, (3) →_R = ∪_{e ∈ N} →_{e}.

Example 2.2: Consider the following CafeOBJ module:

mod! EVEN{
  pr (Nat+);
  op even : Nat -> Bool;
  var N : Nat;
  eq even(0) = true .
  eq even(s N) = true if odd(N) .
  eq odd(s N) = true if even(N) .
}

The corresponding CTRS R_e is defined as follows:

R_e = \{ \begin{align*}
  & \text{even}(0) \to \text{true} \\
  & \text{even}(s(x)) \to \text{true} \iff \text{odd}(x) \\
  & \text{odd}(x) \to \text{true}
\end{align*} \}

We have even(s(0)) →_{e} true since even(0) →_{0} true, odd(s(0)) →_{e} true, and even(s(s(0))) →_{e} true.

III. OPERATIONAL TERMINATION AND CONDITIONAL DEPENDENCY PAIRS

A CTRS R is terminating if there is no infinite rewrite sequence l_0 →_R l_1 →_R l_2 →_R l_3 \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 \to b \}. By removing the condition we have R = \{ a \to b \} and R is trivially terminating and thus R is also terminating since in general \to_R is a subset of \to_{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 \to 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 [9], which is defined by infinite well-formed trees in a logical inference system 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 method to prove termination of (unconditional) TRS [10], 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 operation symbol at the root position of the left-hand side of some rewrite rule is called a defined symbol, that is, D_R = \{ f ∈ Σ | f(...) → r ⇐ c ∈ R \}. The marked symbol f is defined as f# and the set of marked symbols of Σ is written as Σ#. 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 (l#, 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:

\[\text{CDP}(R) = \{ (l#, u#) \iff c \mid l \to r ⇐ c ∈ R, r = C[u], u ∈ D_R \} \cup \{ (l#, u#) \iff c \mid l \to r ⇐ c ∈ R, c = C[u], u ∈ D_R \} \]

We may write (s, t) instead of (s(0), t(0)) since (s, t) ∈ CDP(R).

Definition 3.2: Let R be a CTRS. A (possibly infinite) sequence ((l_i, u_i) | \text{even}(e) \iff c_i (i = 0, 1, 2, \ldots) of pairs of CDP(M) is called a dependency chain of R if there exist \text{even}(e) \iff c_i (i = 0, 1, 2, \ldots) such that (1) c_0 \iff \text{true}, and (2) u_i# = l_{i+1} ⇐ \text{true} for each i ∈ 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 [10] since any right-hand side does not have defined symbols, it has conditional dependency pair 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)) \cdots 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 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_e of Nat does not have infinite dependency pairs, and R_e ∪ R_+ is operationally terminating.

B. Proof of Theorem 3.3

To prove Theorem 3.3, the notion of context-sensitive rewriting (CSR) [11] and the transformation from CTRS to CSR [12] are useful. We introduce the notations and definitions of them. Let R be a (unconditional) TRS. A map μ from Σ to P(Σ) is called a replacement map if μ(f) ⊆ \{ 1, 2, \ldots, ar(f) \} for each f ∈ Σ. The set O_μ(t) of replacement positions of t is defined as ε ∈ O_μ(t) and i.p ∈ O_μ(f(t_1, \ldots, t_n)) if i ∈ μ(f) and p ∈ O_μ(t_i). A context-sensitive rewrite relation of μ, denoted by →_μ, is defined as follows: s →_μ t if and only if there exists l → r and θ such that s = C[θ], t = C[θ], C_p = ε, and p ∈ O_μ(θ).

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

Assume variables in pairs of CDPs are marked such that they are distinct from each other.
Definition 3.5: [12] 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 $\alpha : l \rightarrow r \leftarrow c$. Let $\Sigma_U = \{ U_{\alpha} : l \rightarrow r \leftarrow c \in R \}$

$$ U_{c\alpha}(R) = \{ l \rightarrow U_{c\alpha}(c, x_1, \ldots, x_n) \mid l \rightarrow U_{\alpha}(c, x_1, \ldots, x_n) \rightarrow r \leftarrow c \in R \} $$

$$ \mu_U(f) = \begin{cases} \{0\} & \text{if } f \in \Sigma_U \\ \{1, \ldots, \text{var}(f)\} & \text{if } f \notin \Sigma_U \end{cases} $$

where $\text{var}(l) = \{x_1, \ldots, x_n\}$.

Hereafter, we may write $s \rightarrow^* U$ instead of $s \rightarrow^{\mu_U} t$.

Example 3.6: $R_c$ in Example 2.2 is unravelled as follows:

$$ U_{c\alpha}(R_c) = \begin{cases} \text{even}(0) \rightarrow \text{true} & \mu_{U_{c\alpha}}(\text{even}(0), 0) \\ \text{even}(x(s(0))) \rightarrow \text{true} & \mu_{U_{c\alpha}}(\text{even}(s(s(0))), s(0)) \\ \text{odd}(x(s(0))) \rightarrow \text{true} & \mu_{U_{c\alpha}}(\text{odd}(s(s(0)), s(0)) \\ \text{true}(x(s(0))) \rightarrow \text{true} & \mu_{U_{c\alpha}}(\text{true}(s(s(0))) \\ \text{true}(s(0)) \rightarrow \text{true} & \mu_{U_{c\alpha}}(\text{true}(0)) \end{cases} $$

A term $\text{even}(s(s(0)))$ is reduced into $\text{true}$ as follows:

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

IV. Incremental Proofs of Operational Termination

The notion of the hierarchical extension has been defined for TRSs in [8]. We give a straightforward extension to CTRSs as follows:

Definition 4.1: A pair $[\Sigma_{\neq} : R_{\neq}]$ is called a module extending a CTRS $(\Sigma_0, R_0)$, denoted by $(\Sigma_0, R_0) \leftarrow [\Sigma_{\neq} : R_{\neq}]$.

Proposition 3.7: [12] Let $R$ be a CTRS. (1) If $s \rightarrow R t$, then $s \rightarrow^{\mu_R} t$, and (2) $R$ is operationally terminating if and only if $U_{c\alpha}(R)$ 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 $(l_0, u_0) \rightarrow (l_1, u_1)$ exists. From Definition 3.1 and 3.5, for each $(l_i, u_i)$, there exist rewrite rules $l_i \rightarrow U_{c\alpha}(c_i, x_1, \ldots, x_n)$ and $U_{c\alpha}(true, x_1, \ldots, x_n) \rightarrow C_i(u_i)$ in $U_{c\alpha}(R_c)$. From Definition 3.2 (1), $c_i \theta_j \rightarrow^{*\mu} \text{true}$ for some $\theta_j$, and thus $c_i \rightarrow^{*\mu} \text{true}$ as in Definition 3.7 (1). Thus, we have $l_i \rightarrow^{*\mu} U_{c\alpha}(c_i, \theta_j, \vec{x}) \rightarrow \text{true}(x(s(0))) \rightarrow U_{c\alpha}(\text{true}(s(s(0)))) \rightarrow C_i(u_i) \theta_j \rightarrow C_i(u_i)$ and $u_i \rightarrow^{*\mu} C_i(u_i)$. From Definition 3.2 (2), $u_i \rightarrow^{*\mu} l_{i+1} \rightarrow^{*\mu} l_{i+2} \rightarrow^{*\mu} \cdots$, and thus $u_i \rightarrow^{*\mu} \text{true}$. Note that $s \rightarrow^{*\mu} t$ implies $s \rightarrow R t$ since $R$ does not have any marked symbols. Then, we have an infinite rewrite sequence $l_0 \rightarrow^{*\mu} U_{c\alpha}(c_0, \theta_0, \vec{x}) \rightarrow \text{true}(x(s(0))) \rightarrow U_{c\alpha}(\text{true}(s(s(0)))) \rightarrow C_0(u_0) \theta_0 \rightarrow C_0(u_0)$.

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 $(x^+((s(m), n)), s^+((m, n)))$ belongs to $CDP(R_{\neq} \cup R_{\neq})$ since $+ \in D_{R_{\neq} \cup R_{\neq}}$, it is not considered for $M = [\Sigma_{\neq} : R_{\neq}]$ since $+ \notin D_{R_{\neq}}$.

Definition 4.3: Let $M = [\Sigma : R]$ be a module. The set $MC_{\neq}(R)$ of all conditional dependency pairs of module

(Archive online publication: 21 May 2013)
$M$ (abbr. MCDP) is defined as follows:

$$MCDP(R) = \left\{ (l^#, u^#) \iff c \mid l \rightarrow r \iff c \in R, \right.$$

$$\left. r = C[u], u \notin D_R \right\} \cup$$

$$\left\{ (l^#, u^#) \iff c \mid l \rightarrow r \iff c \in R, \right.$$

$$\left. c = C[u], u \notin D_R \right\}$$

The dependency chain of MCDP is defined as a sequence of MCDPs which are connected by the rewrite relation of some arbitrary CTRS $S$, which may be $R_0 \cup R_1$, for example.

**Definition 4.4:** Let $M = [\Sigma | R]$ be a module and $S$ be a CTRS. A (possibly infinite) sequence $(l^i_R, u^i_R) \iff c_i \ (i = 0, 1, 2, \ldots)$ of pairs of $MCDP(M)$ is called a dependency chain of $M$ over $S$ if there exist $c_i \ (i = 0, 1, 2, \ldots)$ such that $(1) \ c_i \theta_i \rightarrow_S \text{true}$, and $(2) \ u^i_R \theta_i \rightarrow_S l^{i+1}_{R_i} \theta_{i+1}$ for each $i \in N$.

The ordinary $CDP(R)$ and the dependency chain are equivalent to $MCDP([\Sigma | R])$ and the dependency chain of $[\Sigma | R]$ over $R$ respectively. Collapse-extended termination ($CE$-termination) is a subclass of termination, which plays important role in proving termination in a modular way [7]. A CTRS $R$ is $CE$-operationally terminating if $R \cup \{ \}$ 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 $G \notin S$ for any considered $S$. Our main theorem is that $CE$-termination implies $R_0$ and the absence of infinite chains of $R_1$ over $R_0 \cup R_1$ imply ($CE$-operation) termination of $R_0 \cup R_1$.

Before the main theorem, we show the following theorem between $CE$-operation termination and MCDP.

**Theorem 4.5:** A CTRS $(\Sigma, R)$ is $CE$-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 $MCDP([\Sigma | R]) = MCDP([\Sigma \cup \{G/2\}] \cup R \cup \pi)$. □

The following is a main theorem of this paper.

**Theorem 4.6:** Let $[\Sigma_0 | R_0] \iff [\Sigma_1 | R_1]$. If $(1) \ R_0$ is $CE$-operationally terminating, and $(2)$ there exists no infinite chain of $[\Sigma_1 | R_1]$ over $R_0 \cup R_1 \cup \pi$, then $R_0 \cup R_1$ is $CE$-operationally terminating.

**Proof:** (Sketch) It can be proved by the same proof strategy with the unconditional version of this theorem in the literature [8] (Theorem 1). An infinite chain of $CDP(R_0 \cup R_1 \cup \pi)$ consists of either

(a) those of $MCDP([\Sigma_0 | R_0])$,

(b) those of $MCDP([\Sigma_1 | R_1])$, or

(c) $(l^#, u^#) \iff c$ such that $l \in \Sigma_1$ and $u \in \Sigma_0$.

In the example of $M_+ \iff M_*$,

(a) $MCDP(M_+) = \{ (\#(s(m, n)), \#(m, n)) \}$

(b) $MCDP(M_+) = \{ (\#(s(m, n)), \#(m, n)) \}$

(c) $CDP(R_+ \cup R_1 \cup \pi, MCDP(R_+ \cup MCDP(R_1)) = \{ (\#(s(m, n)), \#(s(m, n)) \}$

Since $\Sigma_0 \cap \Sigma_1 = \emptyset$, it suffices to consider infinite chains constructed from (a) only or (b) only. An infinite chain of (a) does not contradict (3) since infinite chains of $CDP(R_0)$ is over $R_0$ while infinite chains of (a) is over $R_0 \cup R_1 \cup \pi$. We can make a chain of $CDP(R_0)$ from a chain of (a) with the same dependency pairs of the sequence and different substitution and connective rewrite relation according to the term interpretation from $\Sigma_0^n \cup \Sigma_1$ to $\Sigma_0^n \cup \{G/2\}$ in [8]. Then, we have the interpreted infinite chain of $CDP(R_0)$ which contradicts (1) and Theorem 3.3.

The condition “$CE$” in Theorem 4.6 (1) is essential since Toyama’s example is a counter-example such that $R_0 = \{ f(a, b, x) \rightarrow f(x, x, x) \}$ and $R_1 = \pi$. Then, $MCDP([G] \mid \pi) = \emptyset$ and no infinite chain exists, however, $R_0 \cup \pi$ is not terminating as shown in 1. INTRODUCTION.

**V. PROVING OPERATIONAL TERMINATION**

To prove the absence of infinite dependency chains of ordinary dependency pairs, the notion of weak reduction pairs is useful [13]. We show that the weak reduction pair also gives a useful sufficient condition for prove operational termination via MCDP.

**Definition 5.1:** A pair $(\geq, >)$ of binary relations on terms is a weak reduction pair if it satisfies the following conditions: (1)$\geq$ is monotonic $(s \geq t \Rightarrow C[s] \geq C[t])$ and stable $(s \geq t \Rightarrow \theta \geq \theta)$ (2) $>$ stable and well-founded (no infinite sequence $t_0 > t_1 > t_2 > \ldots$), and (3) either $\geq \times \geq >$ or $> \times \geq >$.

**Definition 5.2:** Let $M = [\Sigma | R]$ be a module and $S$ be a CTRS. A weak reduction pair $(\geq, >)$ is compatible with $M$ over $S$ if (1) $\forall l \rightarrow r \iff c \in S, l \geq r$ and (2) $\forall (l^#, u^#) \iff c \in MCDP(M), l^# > u^#$.

**Theorem 5.3:** Let $M = [\Sigma | R]$ be a module and $S$ be a CTRS. If there exists a weak reduction pair $(\geq, >)$ compatible with $M$ over $S$, then there is no infinite dependency chain of $M$ over $S$.

**Proof:** Since $G$ is monotonic and stable, $s = C[\theta] \rightarrow_S C[\theta']$ implies $s \geq t$. Since $s > t$, stable, $l^# \theta > \theta' \iff$ for each $(l^#, u^#) \iff c \in MCDP(M).$ An infinite chain $(l^i_0, u^i_0) \iff c_i (i = 0, 1, \ldots)$, we have $l^{i+1}_0 \theta_0 > u^{i+1}_0 \theta_0 \geq \ldots \geq l^0_0 \theta_0 > u^0_0 \theta_0 \geq \ldots$. Now we assume $> \times \geq >$ of Definition 5.1 (3), we have $l^0_0 \theta_0 > u^0_0 \theta_0 > u^0_0 \theta_0 > \ldots$. It contradicts the well-foundedness of $>$. Similarly, for the case of $> \times \geq >$, we have $l^0_0 \theta_0 > l^0_0 \theta_0 > l^0_0 \theta_2 > \ldots$. □

There are two approaches to make a weak reduction pair: semantic methods, e.g. by weakly monotone well-founded $\Sigma$-algebra, and syntactic methods, e.g. by recursive path ordering [6], [7].

**A. Semantic methods**

A weakly monotone $\Sigma$-algebra $(A, \geq)$ consists of a set $A$ and interpretation $A : A^m \rightarrow A$ of each $f/n \in \Sigma$ with a quasi-ordering $\geq$ (a reflexive and transitive binary relation on $A$) 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 $f(\ldots, a, \ldots) \geq f(\ldots, b, \ldots)$. A weakly monotone $\Sigma$-algebra $(A, \geq)$ is well-founded if $> \geq >$ is well-founded. A map $a : X \rightarrow A$ is called an assignment, and it can be extended to a map $a : (\Sigma, X) \rightarrow A$ such that $a(f(t_0, \ldots, t_n)) = f(a(t_0), \ldots, a(t_n))$. The order $\Sigma A$ on $T$ is defined as $t \geq_A t'$ if $\forall a : X \rightarrow A.a(t) \geq a(t')$. The
stable-strict ordering \( \succ_A \) is defined as \( s \succ_A t \) if and only if \( s \theta \succeq_A \theta t \) and \( \theta t \not\succeq_A \theta s \) for each ground substitution \( \theta : X \to T(\Sigma, \emptyset) \). Then, for a weakly monotone well-founded \( \Sigma \)-algebra \((A, \succeq)\), a pair \((\succeq_A, \succ_A)\) is a weak reduction pair [7].

Example 5.4:
1) For \( R_e \), a \( \Sigma_1 \cup \Sigma_\# \cup \{G/2\}\)-algebra \((A, \succeq)\) is defined as follows: \( A = \mathcal{N}, A_0 = 0, A_1(x) = x + 1, A_\neq(x, y) = x + y, A_\#(x, y) = x, \) and \( A_{G}(x, y) = x + y \). Then, for an assignment \( a \), all \( l \to r \in R_e \) are interpreted to equations \( a(l) = a(r) \). For \( \pi = \{G(x, y) \to x, G(y, x) \to y\} \), \( a(G(x, y)) = a(x) + a(y) \geq a(x) \) and \( a(G(y, x)) \geq a(y) \). For \( (+\#(s(m), n), +\#(m, n)) \in MCDP(R_e) \), we have
\[
a(+\#(s(m), n)) = a(s(m)) = a(m) + 1 \geq a(m) = (+\#(m, n)).
\]
Then, \((\succeq_A, \succ_A)\) is a weak reduction pair compatible with \( M_e \) over \( R_e \cup \pi \), and from Theorem 4.6 and 5.3, \( R_e \) is \( C_E \)-operationally terminating. Note that for \((\emptyset, \emptyset) \leftarrow M_e \), the empty CTRS is trivially \( C_E \)-operationally terminating.

2) Next consider \( M_e \leftarrow M_e \). A \( \Sigma_1 \subseteq \Sigma_\# \subseteq \{G/2\}\)-algebra \((A, \succeq)\) is defined as follows: the same interpretation is given for \( \Sigma_e \) and \( G/2 \) with \( A = \mathcal{N}, A_1(x, y) = x + y, \) and \( A_\#(x, y) = x \). Then, for an assignment \( a \), all \( l \to r \in R_e \) are interpreted to equations \( a(l) = a(r) \). For \((+\#(s(m), n), +\#(m, n)) \in MCDP(M_e) \), we have
\[
a(+\#(s(m), n)) = a(s(m)) = a(m) + 1 \geq a(m) = (+\#(m, n)).
\]
Then, \((\succeq_A, \succ_A)\) is a weak reduction pair compatible with \( M_e \) over \( R_e \cup R_e \cup \pi \), and similarly we have that \( R_e \cup R_e \) is \( C_E \)-operationally terminating.

3) Consider \( M_e \leftarrow M_e \). A \( \Sigma_1 \cup \Sigma_\# \cup \{G/2\}\)-algebra \((A, \succeq)\) is defined as follows: the same interpretation is given for \( \Sigma_e \) and \( G/2 \) with \( A = \mathcal{N}, A_{t_{G}} = 0 \) and \( A_{t_{\neq}}(x) = A_{t_{\#}}(x) = A_{t_{G}}(x) = A_{t_{\#}}(x) = x \). Then, \( a(l) \geq a(r) \) for all \( l \to r \in c \in R_e \) and \( a(l) > a(u) \) for all \( (l, u) \in c \in MCDP(M_e) \). Thus, \( R_e \cup R_e \) is \( C_E \)-operationally terminating.

B. Syntactic methods

The notion of recursive path ordering (RPO) is one of the most classical syntactic methods [6]. For a quasi-order \( \succeq \), we write \( \succeq \succeq_{\geq} \succeq \) and \( \succeq \succeq_{\leq} \succeq \).

Definition 5.5: [6] Let \( \Sigma \) be a signature. A status function \( \tau \) maps \( f \in \Sigma \) to either \( mul \) or \( lex_p \) where \( \sigma \) is a permutation on \( \{1, 2, \ldots, \ar(f)\} \). \( \leq \) be a well-founded precedence (quasi-order) on \( \Sigma \) such that \( \tau \) is compatible with \( \succeq \), that is, if \( f \to g \) then \( (1) \tau(f) = \tau(g) = mul \) or \( (2) \tau(f) = lex_p \) and \( \tau(g) = lex_p \). Then, the recursive path order (RPO) \( \succeq_{\text{pre}} \) on terms defined as follows:
\[
s = f(s_1, \ldots, s_m) \succeq_{\text{pre}} g(t_1, \ldots, t_n) \iff
\]
1. \( s_i \neq t_j \) iff \( s_i \succeq_{\text{pre}} t_j \), or
2. \( f \succeq g \land \exists j \in \{1, \ldots, n\}, s_{\text{pre}} >_{\text{pre}} t_j \), or
3. \( f \succeq g \land \{s_1, \ldots, s_m\} \succeq_{\text{pre}} \{\tau(f); \tau(g)\} \{t_1, \ldots, t_n\} \)

where \( \triangleright_{\text{pre}} \leq_{\text{pre}} \leq_{\text{pre}} \). \( \leq_{\text{pre}} \) is a lexicographic extension of \( \leq_{\text{pre}} \) after permutation of \( \{s_1, \ldots, s_m\} \) by \( \sigma \) and \( \{t_1, \ldots, t_n\} \) by \( \sigma' \), and \( \succeq_{\text{pre}} \) is a multisets extension of \( \succeq_{\text{pre}} \).

It is known that \((\succeq_{\text{pre}}, \succeq_{\text{pre}})\) is a weak reduction pair. To describe a specification whose operational termination can be proved by RPO, the notion of argument decreasing is useful [14, 15].

Definition 5.6: [14, 15] For a module \( M = [\Sigma, R] \). The quasi order \( \succeq_M \) on \( \Sigma \) is defined as the smallest reflexive and transitive relation satisfying the following condition: \( f \succeq_M g \) whenever there exists \( f(\cdots) \to r \in c \in R \) such that \( r_p = g \) or \( c_p = g \) for some \( p \). We write \( \succeq_M \succeq_M \subseteq M \).

Definition 5.7: [14, 15] Let \( \tau \) be a status function compatible with \( \succeq_M \). A rewrite rule \( f(l_1, \ldots, l_m) \to r \) is argument decreasing if for any subterm \( g(r_1, \ldots, r_n) \) of \( r \) or \( c \) such that \( f \succeq_M g \) and \( \{l_1, \ldots, l_m\} \succeq_{\text{pre}} \{r_1, \ldots, r_n\} \).

If there exists a status function \( \tau \) such that all component modules of a given specification are argument decreasing, then its operational termination can be proved by the weak reduction pair \((\succeq_{\text{pre}}, \succeq_{\text{pre}})\) with \( \tau \).

Example 5.8: Let \( \tau(f) = mul \) for all operation symbol \( f \) in \( M_1, M_2, M_3 \). Those modules are argument decreasing, and they are all operationally terminating. For example, for the rewrite rule \( + (s(m), n) \to s + (m, n) \in R_e \), we need to check \( (s(m), n) \succeq_{\text{pre}} mul, mul \{m, n\} \). Then \( \tau = \{s(m), n\} - \{m, n\} \). It holds that \( s(m) \succeq_{\text{pre}} m \).

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 a variety of practical systems, for example, authentication protocols, e-government systems, etc [16, 17, 18]. An OTS/CafeOBJ specification 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 \text{ACCOUNT} imports \text{INT} and \text{USER} as data modules, where \text{INT} is a specification of integers and their operations, and \text{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 \text{ceq balance(U, withdraw(U''), I, A)} = \text{(if U = U' then balance(U''A) \# E balance(U, A) \# I then balance(U, A) \# I else balance(U, A) \# F) 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 characterized by the number of each element x, denoted by \( M(x) \) and \( M(N) = \text{N} = M(N) = M(N) \) if \( M(x) = N(x) \), o.w. 0, \( M >_{\text{pre}} N \) if and only if \( \forall x \in M - N \exists y \in N - M \).
Let \( M_a = [\mathcal{S}_a \mid R_a] \) be the module corresponding to \( \text{ACCOUNT} \). Then, the \( \text{MCDP}(M_a) \) consists of only four conditional dependency pairs:

\[
\begin{align*}
&\text{bal}^#(u; \text{dep}(u',i,a)), \text{bal}^#(u,a) \iff (i,0) \\
&\text{bal}^#(u; \text{with}(u',i,a)), \text{bal}^#(u,a) \iff \text{and}(\text{bal}(u',a), i, > (i,0)) \\
&\text{bal}^#(u; \text{with}(u',i,a)), \text{bal}^#(u,a) \iff \text{true} \\
&\text{bal}^#(u',i,a), \text{bal}^#(u,a) \iff \text{true}
\end{align*}
\]

Note that we do not need to consider operation symbols \(_=, \geq, +, -, \text{if then else fi}\) as defined symbols for \( \text{MCDP}(M_a) \) even if they may be defined in the imported modules. The last MCDP can be ignored in an infinite chain because \( \text{with}^# \) does not occur in any right-hand side of MCDP. For each of the remaining three MCDPs, the second argument of \( \text{bal}^# \) strictly decreases in the meaning of the number of operation symbols \( \text{dep} \) and \( \text{with} \). Thus, there is no infinite chain of MCDP, and if the imported modules are already proved \( \text{C}_E \)-operationally terminating, then the whole \( \text{CTRS} \) is \( \text{C}_E \)-operationally terminating. It can be proved by RPO straightforwardly.

### 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. Consider describing a large specification with several modules. (1) We first describe basic modules, that is, without imports, and prove \( \text{C}_E \)-operational termination of them and label them so. (2) Next we describe a module which imports modules labelled by “\( \text{C}_E \)-operational termination”. Then, it suffices to show the absence of infinite chains of the module for proving \( \text{C}_E \)-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


