Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I,
IMECS 2013, March 13 - 15, 2013, Hong Kong

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 is 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 [1], [2], [3], [4] are algebraic specification languages which support several useful advanced features, e.g. module system, typing system with ordered sorts, mixfix 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`, ..., $\mathtt{s}^n$ `0`, ... stand for $0, 1, 2, \ldots, n, \ldots$ 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 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. For example, `s s 0 + s 0` is reduced into `s s s 0` as follows: $\underline{\texttt{s s 0 + s 0}} \to$ `s (`$\underline{\texttt{s 0 + s 0}}$`)` $\to$ `s s (`$\underline{\texttt{0 + s 0}}$`)` $\to$ `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: $\underline{\texttt{s s 0 * s 0}} \to$ `(`$\underline{\texttt{s 0 * s 0}}$`) + s 0` $\to$ `((`$\underline{\texttt{0 * s 0}}$`) + s 0) + s 0` $\to$ `(`$\underline{\texttt{0 + s 0}}$`) + s 0` $\to$ $\underline{\texttt{s 0 + s 0}} \to$ `s (`$\underline{\texttt{0 + s 0}}$`)` $\to$ `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 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 $R_0 = \{f(a, b, x) \to f(x, x, x)\}$ and $R_1 = \{g(x, y) \to x, g(x, y) \to y\}$ 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: $f(a, b, g(a, b)) \to_{R_0} f(g(a, b), g(a, b), g(a, b)) \to_{R_1} f(a, g(a, b), g(a, b)) \to_{R_1} f(a, b, g(a, b)) \to \cdots$ 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 [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 specifications, however, practical OBJ specifications often include conditional equations 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 may omit the arities and write f instead of f/n if no confusion arises. For a signature Σ and a countable set X of variables, the set $T(\Sigma, X)$ (abbr. T) of (Σ, 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 θ 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: $\varepsilon \in O(t)$ (ε : 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 $x_\varepsilon = x$, $f(\cdots)_\varepsilon = f$, and $(f(t_1, \ldots, t_n))_{i.p} = (t_i)_p$. We call t_ε the root symbol of t. A context C is a term with the special symbol $\Box$ which is not included in considered Σ and X and occurs only once in C. The term obtained by replacing $\Box$ 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, $s(0)$, $+(x, s(0))$, $*(+(x, s(0)), s(y))$, etc. Let $t = +(x, s(y))$. $t\theta = +(s(0), s(0))$ when $\theta(x) = s(0)$ and $\theta(y) = 0$, $C[t] = *(s(x), +(x, s(y)))$ when $C = *(s(x), \Box)$, 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 `BOOL`. `BOOL` has the sort `Bool` and the constants `true` and `false`, and usual logical operations `and`, `or`, `not`, etc, on `Bool`. A CafeOBJ conditional equation has the form of `ceq` $l = r$ `if` c, where c is a term of `Bool`, and means that the body equation $l = r$ holds whenever c holds. In the reduction, the instance $l\theta$ is replaced with $r\theta$ when $c\theta$ is reduced into `true`. Note that $Var(r) \cup Var(c) \subseteq Var(l)$. Unconditional equations `eq` $l = r$ can be considered as $l \to r \Leftarrow \texttt{true}$. A (Σ, X)-conditional rewrite rule is a triple (l, r, c), denoted by $l \to r \Leftarrow c$, 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 Σ and a set of (Σ, X)-conditional rewrite rules [1]. We write $l \to r$ instead of $l \to r \Leftarrow c$ when $c = \texttt{true}$. We may call just a conditional rewrite rule, a rewrite rule or a rule, instead of a (Σ, X)-conditional rewrite rule if no confusion arises. A CTRS (Σ_b, R_b) for `BOOL` is defined as $\Sigma_b = \{\texttt{true}/0, \texttt{false}/0, \texttt{not}/1, \ldots\}$ and $R_b = \{\texttt{not}(\texttt{true}) \to \texttt{false}, \texttt{not}(\texttt{false}) \to \texttt{true}, \ldots\}$, and hereafter we assume that every CTRS (Σ, R) implicitly includes (Σ_b, R_b), that is, $\Sigma_b \subseteq \Sigma$ and $R_b \subseteq R$. We may omit Σ and write R as a CTRS instead of (Σ, R). A rewrite relation $\to_R$ is defined as follows: (1) $s \to_0 t$ if there exists $l \to r \Leftarrow c \in R$ and a substitution θ such that $s = C[l\theta]$, $t = C[r\theta]$, and $c\theta = \texttt{true}$. (2) $s \to_{i+1} t$ if there exists $l \to r \Leftarrow c \in R$ and a substitution θ such that $s = C[l\theta]$, $t = C[r\theta]$, and $c\theta \to_i^* \texttt{true}$, where $\to^*$ stands for the reflexible and transitive closure of a binary relation $\to$. (3) $\to_R = \cup_{i \in \mathcal{N}} \to_i$.

Example 2.2: Consider the following CafeOBJ module:

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

The corresponding CTRS R_e is defined as follows:

$$R_e = \left\{ \begin{array}{rcll} even(0) & \to & \texttt{true} & \\ even(s(x)) & \to & \texttt{true} & \Leftarrow odd(x) \\ odd(s(x)) & \to & \texttt{true} & \Leftarrow even(x) \end{array} \right.$$

We have $even(s(s(0))) \to_R \texttt{true}$ since $even(0) \to_0 \texttt{true}$, $odd(s(0)) \to_1 \texttt{true}$, and $even(s(s(0))) \to_2 \texttt{true}$.

III. Operational termination and conditional dependency pairs

A CTRS R is terminating if there is no infinite rewrite sequence $t_0 \to_R t_1 \to_R t_2 \to_R \cdots$. Although termination is one of the most important properties of CTRS, it does not directly correspond to termination of computation of terms. Consider $R = \{a \to b \Leftarrow a\}$. 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 $a \to b \Leftarrow a$ to the target term a, the condition a should be checked whether it can be reduced into `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 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,

[1]Our definition of CTRSs is categorized into 1-CTRS [5] by regarding $l \to r \Leftarrow c$ as $l \to r \Leftarrow c = \texttt{true}$.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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 [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 operation symbol at the root position of the left-hand side of some rewrite rule is called a defined symbol, that is, $D_R = \{f \in \Sigma \mid f(\ldots) \to r \Leftarrow c \in R\}$. The marked symbol of f is defined as $f^{\#}$ and the set of marked symbols of Σ 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 $(l^{\#}, u^{\#})$ 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:

$$CDP(R) = \left\{ (l^{\#}, u^{\#}) \Leftarrow c \;\middle|\; \begin{array}{l} l \to r \Leftarrow c \in R, \\ r = C[u], u_\varepsilon \in D_R \end{array} \right\} \bigcup \left\{ (l^{\#}, u^{\#}) \Leftarrow \mathtt{true} \;\middle|\; \begin{array}{l} l \to r \Leftarrow c \in R, \\ c = C[u], u_\varepsilon \in D_R \end{array} \right\}$$

We may write (s, t) instead of $(s, t) \Leftarrow \mathtt{true}$.

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

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 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_+ of `NAT` does not have infinite dependency chains, and $R_e \cup R_+$ is operationally terminating.

[2]Assume variables in pairs of CDPs are marked such that they are distinct from each other.

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 μ from Σ to $\mathcal{P}(\mathcal{N})$ 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 μ, denoted by $\to_\mu$, is defined as follows: $s \to_\mu t$ if and only if there exists $l \to r \in R$ and θ such that $s = C[l\theta]$, $t = C[r\theta]$, $C_p = \Box$, and $p \in O_\mu(t)$.

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

Definition 3.5: [11] Let (Σ, R) be a CTRS. The (unconditional) TRS $(\Sigma \cup \Sigma_U, U_{cs}(R))$ and the replacement map μ_U are defined as follows: Let each conditional rule in R be labeled like $\alpha : l \to r \Leftarrow c$.

$$\begin{array}{rcl} \Sigma_U & = & \{U_\alpha \mid \alpha : l \to r \Leftarrow c \in R\} \\ U_{cs}(R) & = & \left\{ \begin{array}{c} l \to U_\alpha(c, x_1, \ldots, x_n) \\ U_\alpha(\mathtt{true}, x_1, \ldots, x_n) \to r \end{array} \;\middle|\; \begin{array}{l} \alpha : l \to r \\ \Leftarrow c \in R \end{array} \right\} \\ & \bigcup & \left\{ l \to r \mid l \to r \Leftarrow \mathtt{true} \in R \right\} \\ \mu_U(f) & = & \left\{ \begin{array}{ll} \{1, \ldots, ar(f)\} & \text{if } f \in \Sigma \\ \{1\} & \text{if } f \in \Sigma_U \end{array} \right. \end{array}$$

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

Hereafter, we may write $s \to_U t$ instead of $s \to_{\mu_U} t$.

Example 3.6: R_e in Example 2.2 is unravelled as follows:

$$U_{cs}(R_e) = \left\{ \begin{array}{rcl} even(0) & \to & \mathtt{true} \\ even(s(x)) & \to & U_e(odd(x), x) \\ U_e(\mathtt{true}, x) & \to & \mathtt{true} \\ odd(s(x)) & \to & U_o(even(x), x) \\ U_o(\mathtt{true}, x) & \to & \mathtt{true} \end{array} \right.$$

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

$$\begin{array}{rll} even(s(s(0))) & \to_\mu & U_e(odd(s(0), s(0)) \\ & \to_\mu & U_e(U_o(even(0), 0), s(0)) \\ & \to_\mu & U_e(U_o(\mathtt{true}, 0), s(0)) \\ & \to_\mu & U_e(\mathtt{true}, s(0)) \\ & \to_\mu & \mathtt{true} \end{array}$$

A TRS is called μ-terminating if there is no infinite rewrite sequence $t_0 \to_\mu t_1 \to_\mu t_2 \to_\mu \cdots$. Operational termination of CTRS and μ-termination of the transformed TRS have been shown to be equivalent in [11].

Proposition 3.7: [11] Let R be a CTRS. (1) If $s \to_R t$, then $s \to_U^+ t$, and (2) R is operationally terminating if and only if $U_{cs}(R)$ is μ_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 $(l_i^{\#}, u_i^{\#}) \Leftarrow c_i$ $(i \in \mathcal{N})$ exists. From Definition 3.1 and 3.5, for each $(l_i^{\#}, u_i^{\#}) \Leftarrow c_i$, there exist rewrite rules $l_i \to U_{\alpha_i}(c_i, x_1^i, \ldots, x_n^i)$ and $U_{\alpha_i}(\mathtt{true}, x_1^i, \ldots, x_n^i) \to C_i[u_i]$ in $U_{cs}(R)$. From Definition 3.2 (1), $c_i\theta_i \to_R^* \mathtt{true}$ for some θ_i, and thus $c_i\theta_i \to_U^* \mathtt{true}$ from Proposition 3.7 (1). Thus, we have

Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I,
IMECS 2013, March 13 - 15, 2013, Hong Kong

$l_i\theta_i \to_U U_{\alpha_i}(c_i\theta_i, \vec{x}) \to_U^* U_{\alpha_i}(\mathtt{true}, \vec{x}) \to_U C_i[u_i]\theta_i = C_i\theta_i[u_i\theta_i]$. From Definition 3.2 (2), $u_i^{\#}\theta_i \to_R^* l_{i+1}^{\#}\theta_{i+1}$ and thus $u_i\theta_i \to_U^* l_{i+1}\theta_{i+1}$. Note that $s^{\#} \to_R t^{\#}$ implies $s \to_R t$ since R does not have any marked symbols $f^{\#}$. Then, we have an infinite rewrite sequence $l_0\theta_0 \to_U^+ C_0\theta_0[u_0\theta_0] \to_U^* C_0\theta_0[l_1\theta_1] \to_U^+ C_0\theta_0[C_1\theta_1[u_1\theta_1]] \to_U^* \cdots \to_U^+ C_0\theta_0[C_1\theta_1[\cdots C_n\theta_n[u_n\theta_n]]] \to_U \cdots$. Since $l_0\theta_0$ 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. $U_{cs}(R)$ is not μ_U-terminating from Proposition 3.7 (2). We take a minimal infinite rewrite sequence of μ_U-termination: $t_0^0 \to_U t_1^0 \to_U t_2^0 \to_U \ldots$ where $t_0^0 \in T(\Sigma, X)$, and there is no infinite rewrite sequence $t_0' \to_U t_1' \to_U \cdots$ beginning with a strict subterm t_0' of t_0^0. From the minimal assumption and the fact that $t_0^0 \in T(\Sigma, X)$, there exists t_i^0 such that $t_i^0 = l_0\theta_0$ and $t_{i+1}^0 = U_{\alpha_0}(c_0\theta_0, \vec{v})$ for some $l_0 \to r_0 \Leftarrow c_0 \in R$, where $v_k = \theta_0(x_k)$. Note that $\theta_0(z) \in T(\Sigma, X)$ for each $z \in \vec{x}$ since $t_0^0 \in T(\Sigma, X)$. We consider the case that $c_0\theta_0$ is not μ_U-terminating. Then, we take a minimal infinite rewrite sequence $c_0\theta_0 \trianglerighteq u_0\theta_0 = t_0^1 \to_U t_1^1 \to_U \cdots$. From the minimality, $root(u_0) \in D_R$. Then, we take $(l_0^{\#}, u_0{}^{\#}) \Leftarrow \mathtt{true}$ as the first CDP of the chain. Next, we consider the other case that $c_0\theta_0$ is μ_U-terminating. Since $\mu_U(U_{\alpha_0}) = \{1\}$ and t_{i+1}^0 is not μ-terminating, $c_0\theta_0 \to_U^* \mathtt{true}$ holds, and $t_{i+1}^0 = U_{\alpha_0}(c_0\theta_0, \vec{v}) \to_U^* U_{\alpha_0}(\mathtt{true}, \vec{v}) \to_U r_0\theta_0$ where $r_0\theta_0$ is not μ-terminating. Then, we take a minimal infinite rewrite sequence $r_0\theta_0 \trianglerighteq u_0\theta_0 = t_0^1 \to_U t_1^1 \to_U \cdots$. From the minimality, $root(u_0) \in D_R$. Then, we take $(l_0^{\#}, u_0{}^{\#}) \Leftarrow c_0$ as the first CDP of the chain. From Proposition 3.7 (1), $c_0\theta_0 \to_R^* \mathtt{true}$ holds. Now, for both cases, there exists t_i^1 such that $t_i^1 = l_1\theta_1$ and $t_{i+1}^1 = U_{\alpha_1}(c_1\theta_1, \vec{v})$ for some $l_1 \to r_1 \Leftarrow c_1 \in R$, and similarly we can take the next CDP $(l_1^{\#}, u_1{}^{\#}) \Leftarrow c_1$ with θ_1. Since $u_0\theta_0 \to_U^* t_i^1 = l_1\theta_1$, we have $u_0\theta_0 \to_R^* t_i^1 = l_1\theta_1$ from Proposition 3.7(1). As a similar way, we can make an infinite chain $(l_k^{\#}, u_k{}^{\#}) \Leftarrow c_k$ with θ_k $(k \in \mathcal{N})$. $\square$

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 $[\Sigma_1 \mid R_1]$ is called a module extending a CTRS (Σ_0, R_0), denoted by $(\Sigma_0, R_0) \leftarrow [\Sigma_1 \mid R_1]$, if (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 (Σ_0, R_0) with module $[\Sigma_1 \mid R_1]$.

We write $(\Sigma_0, R_0) \leftarrow [\Sigma_1 \mid R_1] \leftarrow [\Sigma_2 \mid R_2]$ when $[\Sigma_2 \mid R_2]$ extends $(\Sigma_0 \cup \Sigma_1, R_0 \cup R_1)$. A CTRS (Σ_0, R_0) can be regarded as a module $[\Sigma_0 \mid R_0]$ extending the empty CTRS $(\emptyset, \emptyset)$. Hereafter we may use the module expression for an ordinary CTRSs.

Example 4.2: Let $M_+ = [\Sigma_+ \mid R_+]$ and $M_* = [\Sigma_* \mid R_*]$ such that $\Sigma_+ = \{0, s, +\}$ declared in `NAT+`, $\Sigma_* = \{*\}$ declared in `NAT*`, R_+ is the TRS which has two rewrite rules in `NAT+`, and R_* is the TRS which has two rewrite rules in `NAT*`. Then, $M_+ \leftarrow M_*$ since $\Sigma_+ \cap \Sigma_* = \emptyset$ and the root symbols of the two rewrite rules in R_* is $* \in \Sigma_*$. Also $M_+ \leftarrow M_e (= [\Sigma_e \mid R_e])$ for $\Sigma_e = \{even, odd\} (= D_{R_e})$ and R_e 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 $(*^{\#}(s(m), n), +^{\#}(n, *(m, n))$ belongs to $CDP(R_+ \cup R_*)$ since $+ \in D_{R_+ \cup R_*}$, it is not considered for $M_* = [\Sigma_* \mid R_*]$ since $+ \notin D_{R_*}$.

Definition 4.3: Let $M = [\Sigma \mid R]$ be a module. The set $MCDP(R)$ of all conditional dependency pair of module M (abbr. MCDP) is defined as follows:

$$MCDP(R) = \left\{ (l^{\#}, u^{\#}) \Leftarrow c \;\middle|\; \begin{array}{l} l \to r \Leftarrow c \in R, \\ r = C[u], u_\varepsilon \in D_R \end{array} \right\} \bigcup \left\{ (l^{\#}, u^{\#}) \Leftarrow \mathtt{true} \;\middle|\; \begin{array}{l} l \to r \Leftarrow c \in R, \\ c = C[u], u_\varepsilon \in D_R \end{array} \right\}$$

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 $M = [\Sigma \mid R]$ be a module and S be a CTRS. A (possibly infinite) sequence $(l_i^{\#}, u_i^{\#}) \Leftarrow c_i$ $(i = 0, 1, 2, \ldots)$ of pairs of $MCDP(M)$ is called a dependency chain of M over S if there exist θ_i $(i = 0, 1, 2, \ldots)$ such that (1) $c_i\theta_i \to_S^* \mathtt{true}$, and (2) $u_i^{\#}\theta_i \to_S^* l_{i+1}^{\#}\theta_{i+1}$ for each $i \in \mathcal{N}$.

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

Before the main theorem, we show the following theorem between C_E-operational termination and MCDP.

Theorem 4.5: A CTRS (Σ, R) is C_E-operationally terminating if and only if there exists no infinite chain of $[\Sigma \mid R]$ over $R \cup \pi$.

Proof: Since G is fresh, R does not have G and $MCDP([\Sigma \mid R]) = MCDP([\Sigma \cup \{G/2\} \mid R \cup \pi])$. $\square$

The following is a main theorem of this paper.

Theorem 4.6: Let $[\Sigma_0 \mid R_0] \leftarrow [\Sigma_1 \mid R_1]$. If (1) R_0 is C_E-operationally terminating, and (2) there exists no infinite chain of $[\Sigma_1 \mid R_1]$ over $R_0 \cup R_1 \cup \pi$, then $R_0 \cup R_1$ is C_E-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 $CDP(R_0 \cup R_1 \cup \pi)$ consists of either

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

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

(b) those of $MCDP([\Sigma_1 \mid R_1])$, or
(c) $(l^\#, u^\#) \Leftarrow c$ such that $l_\varepsilon \in \Sigma_1$ and $u_\varepsilon \in \Sigma_0$.

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

(a) $MCDP(M_+) = \{(+^\#(s(m), n), +^\#(m, n))\}$
(b) $MCDP(M_*) = \{(*^\#(s(m), n), *^\#(m, n))\}$
(c) $CDP(R_+ \cup R_*) \setminus (MCDP(R_+) \cup MCDP(R_*))$
$= \{(*^\#(s(m), n), +^\#(*(m, n), n))\}$

Since $\Sigma_0 \cap \Sigma_1 = \emptyset$, it suffices to consider infinite chains constructed from (a) only or (b) only. An infinite chain of (b) contradicts the assumption (2) directly. An infinite chain of (a) does not directly contradict (1) since 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 $\Sigma_0^\# \cup \Sigma_1$ to $\Sigma_0^\# \cup \{G/2\}$ in [7]. Then, we have the interpreted infinite chain of $CDP(R_0)$ which contradicts (1) and Theorem 3.3. $\Box$

The condition "C_E" in Theorem 4.6 (1) is essential since Toyama's example is a counter-example such that $R_0 = \{f(a, b, x) \to f(x, x, x)\}$ 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 I. 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 [12]. We show that the weak reduction pair also gives a useful sufficient condition for prove operational termination via MCDP.

Definition 5.1: [12] 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 ($s \geq t \Rightarrow s\theta \geq t\theta$) (2) $>$ is stable and well-founded (no infinite sequence $t_0 > t_1 > t_2 > \cdots$), and (3) either $\geq \cdot > \subseteq >$ or $> \cdot \geq \subseteq >$.

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

Theorem 5.3: Let $M = [\Sigma \mid R]$ 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 $\geq$ is monotonic and stable, $s = C[l\theta] \to_S C[r\theta] = t$ implies $s \geq t$. Since $>$ is stable, $l^\#\theta > r^\#\theta$ for each $(l^\#, r^\#) \Leftarrow c \in MCDP(M)$. From an infinite chain $(l_i^\#, u_i^\#) \Leftarrow c_i$ $(i = 0, 1, \ldots)$, we have $l_i^\#\theta_0 > u^\#\theta_0 \geq \cdots \geq l_i^\#\theta_1 > u^\#\theta_1 \geq \cdots$. Now we assume $\geq \cdot > \subseteq >$ of Definition 5.1 (3). we have $l_i^\#\theta_0 > u^\#\theta_0 > u^\#\theta_1 > u^\#\theta_2 > \cdots$. It contradicts the well-foundedness of $>$. Similarly, for the case of $> \cdot \geq \subseteq >$, we have $l_i^\#\theta_0 > l^\#\theta_1 > l^\#\theta_2 > \cdots$. $\Box$

To make a weak reduction pair, the notion of Σ-algebra is useful [5], [6]. A weakly monotone Σ-algebra $(A, \geq)$ consists of a set A and interpretation $A_f : A^n \to A$ 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 Σ-algebra $(A, \geq)$ is well-founded if $>$ is well-founded. A map $a : X \to A$ is called an assignment, and it can be extended to a map $a : T(\Sigma, X) \to A$ such that $a(f(t_0, \ldots, t_n)) = f(a(t_0), \ldots, a(t_n))$. The order $\geq_A$ on T is defined as $t \geq_A t'$ if $\forall a : X \to A.a(t) \geq a(t')$. The stable-strict ordering $>_A$ is defined as $s >_A t$ if and only if $s\theta \geq_A t\theta$ and $t\theta \not\geq_A s\theta$ for each ground substitution $\theta : X \to T(\Sigma, \emptyset)$. Then, for a weakly monotone well-founded Σ-algebra $(A, \geq)$, a pair $(\geq_A, >_A)$ is a weak reduction pair [6].

Example 5.4:

1) For R_+, a $\Sigma_+ \cup \Sigma_+^\# \cup \{G/2\}$-algebra $(A, \geq)$ is defined as follows: $A = \mathcal{N}$, $A_0 = 0$, $A_s(x) = x + 1$, $A_+(x, y) = x + y$, $A_{+^\#}(x, y) = x$, and $A_G(x, y) = x + y$. Then, for an assignment a, all $l \to r \in R_+$ are interpreted to equations $a(l) = a(r)$. For $\pi = \{G(x, y) \to x, G(x, y) \to y\}$, $a(G(x, y)) = a(x) + a(y) \geq a(x)$ and $a(G(x, y)) \geq a(y)$. For $(+^\#(s(m), n), +^\#(m, n)) \in MCDP(M_+)$, we have
$$\begin{aligned} a(+^\#(s(m), n)) &= a(s(m)) = a(m) + 1 \\ &> a(m) = a(+^\#(m, n))). \end{aligned}$$
Then, $(\geq_A, >_A)$ is a weak reduction pair compatible with M_+ over $R_+ \cup \pi$, and from Theorem 4.6 and 5.3, R_+ is C_E-operationally terminating. Note that for $(\emptyset, \emptyset) \leftarrow M_+$, the empty CTRS is trivially C_E-operationally terminating.
2) Next consider $M_+ \leftarrow M_*$. A $\Sigma_+ \cup \Sigma_* \cup \Sigma_*^\# \cup \{G/2\}$-algebra $(A, \geq)$ is defined as follows: the same interpretation is given for Σ_+ and $G/2$ with $A = \mathcal{N}$. $A_*(x, y) = x \times y$, and $A_{*^\#}(x, y) = x$. Then, for an assignment a, all $l \to r \in R_*$ are interpreted to equations $a(l) = a(r)$. For $(*^\#(s(m), n), *^\#(m, n)) \in MCDP(M_*)$, we have
$$\begin{aligned} a(*^\#(s(m), n)) &= a(s(m)) = a(m) + 1 \\ &> a(m) = a(*^\#(m, n)). \end{aligned}$$
Then, $(\geq_A, >_A)$ is a weak reduction pair compatible with M_* over $R_* \cup R_+ \cup \pi$, and similarlly we have that $R_+ \cup R_*$ is C_E-operationally terminating.
3) Consider $M_+ \leftarrow M_e$. A $\Sigma_+ \cup \Sigma_e \cup \Sigma_e^\# \cup \{G/2\}$-algebra $(A, \geq)$ is defined as follows: the same interpretation is given for Σ_+ and $G/2$ with $A = \mathcal{N}$. $A_{true} = 0$ and $A_{even}(x) = A_{odd}(x) = A_{even^\#}(x) = A_{odd^\#}(x) = x$. Then, $a(l) \geq a(r)$ for all $l \to r \Leftarrow c \in R_e$ and $a(l^\#) > a(u^\#)$ for all $(l^\#, u^\#) \Leftarrow c \in MCDP(M_e)$. Thus, $R_+ \cup R_e$ 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

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
mod* ACCOUNT {
  pr(INT + USER)
  *[ Sys ]*
  op  init : -> Sys
  bop balance : User Sys -> Int
  bop deposit : User Int Sys -> Sys
  bop withdraw : User Int Sys -> Sys
  vars U U' : User
  var I : Int
  var A : Sys
  eq balance(U,init) = 0 .
```

```
ceq balance(U, deposit(U',I, A)) =
 (if U = U' then I + balance(U,A) else balance(U,A) fi)
  if  I >= 0 .
ceq deposit(U',I,A) = A
  if not ( I >= 0) .
ceq 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  .
ceq withdraw(U',I,A) = A
  if  not(balance(U',A) >= I and I >= 0) .
}
```

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 `ceq 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` ($\neq$ `U'`) is not changed.

Let $M_a = [\Sigma_a \mid R_a]$ be the module corresponding to `ACCOUNT`. Then, the $MCDP(M_a)$ consists of only four conditional dependency pairs:

$$(bal^{\#}(u, dep(u', i, a)), bal^{\#}(u, a)) \Leftarrow \geq (i, 0)$$
$$(bal^{\#}(u, with(u', i, a)), bal^{\#}(u, a)) \Leftarrow and(\geq (bal(u', a), i), \geq (i, 0))$$
$$(bal^{\#}(u, with(u', i, a)), bal^{\#}(u, a)) \Leftarrow true$$
$$(with^{\#}(u', i, a), bal^{\#}(u, a)) \Leftarrow true$$

Note that we do not need to consider operation symbols $=$, $\geq$, $+$, $-$, $if\_then\_else\_fi$ as defined symbols for $MCDP(M_a)$ 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 C_E-operationally terminating, then the whole CTRS is C_E-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. Consider describing a large specification with several modules. (1) We first describe basic modules, that is, without imports, and prove C_E-operational termination of them and label them so. (2) Next we describe a module which imports modules labelled by "C_E-operational termination". Then, it suffices to show the absence of infinite chains of the module for proving 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

[1] K. Futatsugi, J. A. Goguen, J.-P. Jouannaud, and J. Meseguer, "Principles of obj2." in *Proceedings of the 12th ACM Symposium on Principles of Programming Languages, POPL*, 1985, pp. 52–66.
[2] J. A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud, *Software Engineering with OBJ: Algebraic Specification in Action*. Kluwers Academic Publishers, 2000, ch. Introducing OBJ*.
[3] *CafeOBJ*, http://www.ldl.jaist.ac.jp/cafeobj/.
[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, Eds., *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, ser. Lecture Notes in Computer Science, vol. 4350. Springer, 2007.
[5] Terese, *Term Rewriting Systems*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003, vol. 55.
[6] E. Ohlebusch, *Advanced topics in term rewriting*. Springer, 2002.
[7] X. Urbain, "Modular & incremental automated termination proofs," *J. Autom. Reasoning*, vol. 32, no. 4, pp. 315–355, 2004.
[8] S. Lucas, C. Marché, and J. Meseguer, "Operational termination of conditional term rewriting systems," *Inf. Process. Lett.*, vol. 95, no. 4, pp. 446–453, 2005.
[9] T. Arts and J. Giesl, "Termination of term rewriting using dependency pairs," *Theor. Comput. Sci.*, vol. 236, no. 1-2, pp. 133–178, 2000.
[10] S. Lucas, "Context-sensitive computations in functional and functional logic programs," *Journal of Functional and Logic Programming*, vol. 1998, no. 1, January 1998.
[11] F. Schernhammer and B. Gramlich, "Characterizing and proving operational termination of deterministic conditional term rewriting systems," *J. Log. Algebr. Program.*, vol. 79, no. 7, pp. 659–688, 2010.
[12] K. Kusakari, M. Nakamura, and Y. Toyama, "Argument filtering transformation," in *PPDP*, ser. Lecture Notes in Computer Science, G. Nadathur, Ed., vol. 1702. Springer, 1999, pp. 47–61.
[13] K. Ogata and K. Futatsugi, "Proof scores in the ots/cafeobj method," in *FMOODS*, ser. Lecture Notes in Computer Science, E. Najm, U. Nestmann, and P. Stevens, Eds., vol. 2884. Springer, 2003, pp. 170–184.
[14] W. Kong, K. Ogata, and K. Futatsugi, "Towards reliable e-government systems with the ots/cafeobj method," *IEICE Transactions*, vol. 93-D, no. 5, pp. 974–984, 2010.
[15] I. Ouranos, K. Ogata, and P. S. Stefaneas, "Formal analysis of tesla protocol in the timed ots/cafeobj method," in *ISoLA (2)*, ser. Lecture Notes in Computer Science, T. Margaria and B. Steffen, Eds., vol. 7610. Springer, 2012, pp. 126–142.