Title	The equivalence of the reductions with the E-strategy with and without marks
Author(s)	Nagaya, Takashi; Matsumoto, Michihiro; Ogata, Kazuhiro; Futatsugi, Kokichi
Citation	Research report (School of Information Science, Japan Advanced Institute of Science and Technology), IS-RR-97-0034F: 1-9
Issue Date	1997-08-13
Type	Technical Report
Text version	publisher
URL	http://hdl.handle.net/10119/8376
Rights	
Description	リサーチレポート（北陸先端科学技術大学院大学情報科学研究科）

The equivalence of the reductions with the E-strategy with and without marks

Takashi Nagaya †, Michihiro Matsumoto †‡
Kazuhiro Ogata †, and Kokichi Futatsugi †

IS-RR-97-0034F

†Graduate School of Information Science,
Japan Advanced Institute of Science and Technology

‡Reseach Center of PFU Limited.

ISSN 0918-7553

August 13, 1997

Abstract

The E-strategy is the evaluation strategy which is adopted by OBJ2, OBJ3, and CafeOBJ. OBJ2, OBJ3, and CafeOBJ also adopt the reduction with the E-strategy with marks. In this paper, we show the result of the reduction with marks coincides with the result of the reduction without marks if the E-strategies satisfy the conditions that the E-strategy list do not include 0, or the last element of the E-strategy list is 0. In the most of the practical usage, the above conditions are satisfied.

1 Introduction

The E-strategy [2, 3, 5] is the evaluation strategy which is adopted by OBJ2, OBJ3, and CafeOBJ, in the following referred to as the OBJs. The OBJs are algebraic specification languages which have an underlying formal semantics that is based on equational logic, and an operational semantics that is based on rewrite rules. The equations of the specification may be used to rewrite rules from left hands to right hands. Therefore, static properties of the specification are verified by the rewriting rule engine which is controlled by the E-strategies.

To verify the practical problem, the rewriting rule engine must have a good termination behavior and be implemented efficiently. The outermost strategy (lazy evaluation) often has a good termination behavior, but is implemented inefficiently. On the other hand, the innermost strategy(eager evaluation) can be implemented efficiently, but has a bad termination behavior. The E-strategy can specify the argument which is evaluated lazily. Therefore, the E-strategy can have a good termination behavior, and by eliminating rewriting of outermost redexes, the reduction become efficient.

Consequently, we think the E-strategy is suitable to control the rewriting rule engines which verify the practical problem.

The OBJs adopt the reduction with E-strategy with marks which is more efficient than the reduction without marks.

But, until now, the behavior of the reduction with marks have not been clear.

Therefore, in this paper, we show the result of the reduction with marks coincides with the result of the reduction without marks if the E-strategies satisfy the conditions that the E-strategy list do not include 0, or the last element of the E-strategy list is 0. In the most of the practical usage, the above conditions are satisfied.

We give terminology and notation in section 2, the explanation of the E-strategy in section 3, and the proof of the equivalence of the reductions with the E-strategy with and without marks in section 4. We end with a discussion of related work and conclusion.

2 Terminology and Notation

We assume that the reader is familiar with the basic concepts of rewriting. We introduce the notations used later and refer to [1, 4].

Let $\mathcal{V}$ be a set of variables and let $\mathcal{F}$ be a set of function symbols where $\mathcal{F} \cap \mathcal{V} = \phi$. Each function symbol equips with an arity (a natural number), i.e. the number of arguments it is supported to have. Let $\mathcal{T}(\mathcal{F}, \mathcal{V})$ be the set of terms which are constructed by $\mathcal{F}$ and $\mathcal{V}$. $\mathcal{T}(\mathcal{F}, \mathcal{V})$ may be abbreviated to $\mathcal{T}$. $V(t)$ stands for the set of all variables appearing in t. If $t \in \mathcal{V}$, $top(t) = t$ and if $t = f(t_1, \ldots, t_n)$, $top(t) = f$. If $t = f(t_1, \ldots, t_n)$, then $subterm(t, i) = t_i$ and $replace(t, i, s)$ denotes the term that is obtained from t by replacing t_i with s.

Substitution σ is a map from $\mathcal{V}$ to $\mathcal{T}$, extended to a map from $\mathcal{T}$ to $\mathcal{T}$ in such a way that $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$, for each f (of arity n) $\in \mathcal{F}$ and for all terms $t_i \in \mathcal{T}$. We also write $t\sigma$ instead of $\sigma(t)$.

A context is a term of $\mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$ containing one occurrence of a special symbol $\Box$, denoting an empty place and is denoted by $C[\ ]$. $C[t]$ denotes the term that is obtained by replacing $\Box$ with t. If there is a context $C[\ ]$ which satisfies $t = C[s]$, s is called a subterm of t.

A rewrite rule is a pair (l, r) of terms in $\mathcal{T}$ which satisfies $l \notin \mathcal{V}$ and $V(r) \subseteq V(l)$. It will be written as $l \to r$. A term rewriting system (TRS for short) is a pair $(\mathcal{F}, \mathcal{R})$ of a set $\mathcal{F}$ of function symbols and a set $\mathcal{R}$ of rewrite rules. A TRS may be denoted by $\mathcal{R}$, ignoring $\mathcal{F}$.

We define the reduction relation $\to_{\mathcal{R}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ using the set of rewrite rules of TRS $(\mathcal{F}, \mathcal{R})$, as follows. If there are $l \to r \in \mathcal{R}$, σ, and $C[\ ]$ which satisfy $t = C[l\sigma]$ and $s = C[r\sigma]$, we decide $t \to_{\mathcal{R}} s$. At this time, $l\sigma$ is called a redex and $r\sigma$ is called the contractum of $l\sigma$. We generally omit $\mathcal{R}$ from $\to_{\mathcal{R}}$ when it is clear from context. The transitive reflexive closure of $\to_{\mathcal{R}}$ is written as $\xrightarrow{*}_{\mathcal{R}}$. If $t \xrightarrow{*}_{\mathcal{R}} s$, s is a reduct of t.

We say that t is a normal form if there is no $s \in \mathcal{T}$ such that $t \to_{\mathcal{R}} s$. Further, $t \in \mathcal{T}$ has a normal form if $t \xrightarrow{*}_{\mathcal{R}} s$ for some normal form $s \in \mathcal{T}$ and s is called a normal form of t.

3 The E-strategy

The E-strategy [2, 3, 5] is an evaluation strategy of a function symbol, which specify the evaluation order of arguments. Evaluation order is specified by the list of integers. This list is called the E-strategy list. Each positive integer in the E-strategy list represents an argument. 1 represents the first argument, 2 represents the second argument, and so on. 0 represents the whole term.

If we omit a given argument number from the E-strategy list, this argument get lazy evaluation, i.e. this argument is not evaluated unless some rewrite exposes it from underneath the given operation. For example, the conditional is declared as follows.

$$op\ if\_then\_else\_fi : Bool\ Int\ Int\ -> \ Int\ \{strat : (1\ 0)\}$$

First, the first argument (conditional part) is evaluated. Then the whole argument is evaluated. Therefore, the second and third arguments (then and else parts) are not evaluated, if they are not arguments after the whole argument is evaluated.

In the rest of this paper, we assume that each function symbol has own E-strategy list and each variable has the empty E-strategy list.

The reduction with the E-strategy without marks is presented by *eval* as follows.

Definition 1 *Let eval* $: \mathcal{T} \to \mathcal{T}$ *be a function, such that*

$$\begin{aligned}
eval(t) &= reduce(t, topElist(t)) \\
reduce(t, nil) &= t \\
reduce(t, cons(0, l)) &= \begin{cases} eval(contract(t)) & if\ t\ is\ a\ redex \\ reduce(t, l) & otherwise \end{cases} \\
reduce(t, cons(n+1, l)) &= reduce(replace(t, n+1, s), l) \qquad (n \geq 0) \\
&\quad where\ \ s = eval(subterm(t, n+1))
\end{aligned}$$

$topElist(t)$ denotes the E-strategy list of $top(t)$ and $contract(t)$ denotes the function which returns the contractum of t if t is a redex.

First, $eval(t)$ calls *reduce* with t and the E-strategy list of $top(t)$. After that, *reduce* evaluates arguments of $top(t)$ along this list. Especially, when the element of this list is 0, if t is a redex, t is reduced and the contractum of t is evaluated by *eval*. Consequently, if the

E-strategy of the function symbol of arity n is $(1\ 2\ \cdots\ n\ 0)$, the E-strategy coincides with the leftmost innermost strategy. This means the E-strategy can simulate the innermost strategy without any cost.

E-strategy supports lazy evaluation. Therefore, termination behavior of the E-strategy can be better than it of the leftmost innermost strategy.

Example 1 *Let*

$$\mathcal{R} = \begin{cases} inf(x) \rightarrow cons(x, inf(s(x))) \\ nth(0, cons(x, y)) \rightarrow x \\ nth(s(x), cons(y, z)) \rightarrow nth(x, z). \end{cases}$$

$nth(s(0), inf(0))$ has a normal form, but has an infinite reduction sequence of the leftmost innermost strategy.

$$\begin{aligned} nth(s(0), \underline{inf(0)}) &\rightarrow nth(s(0), cons(0, \underline{inf(s(0))})) \\ &\rightarrow nth(s(0), cons(0, cons(s(0), \underline{inf(s(s(0)))}))) \\ &\rightarrow \cdots \end{aligned}$$

If the E-strategy list of cons is (0) and the E-strategy lists of other functional symbols (of arity n) are $(1\ 2\ \cdots\ n\ 0)$, the reduction sequence of these E-strategies is finite.

$$\begin{aligned} nth(s(0), \underline{inf(0)}) &\rightarrow \underline{nth(s(0), cons(0, inf(s(0))))} \\ &\rightarrow nth(0, \underline{inf(s(0))}) \\ &\rightarrow \underline{nth(0, cons(s(0), inf(s(s(0)))))} \\ &\rightarrow s(0) \end{aligned}$$

4 The equivalence of the reductions with the E-strategy with and without marks

The OBJs adopt the reduction with the E-strategy with marks. Because, it is more efficient than the reduction without marks.

Example 2 *Let*

$$\mathcal{R} = \begin{cases} f(x) \rightarrow h(x) \\ a \rightarrow b \end{cases}$$

Let the E-strategy lists of f, g, and h be (1 0). Then, the reduction sequence of f(g(a)) is as follows.

$$\begin{aligned} f(g(a)) &\rightarrow f(g(b)) \\ &\rightarrow h(g(b)) \end{aligned}$$

In the reduction without marks, g(b) of h(g(b)) is evaluated. But, in the reduction with marks, g(b) of h(g(b)) is not evaluated, because, g(b) has evaluated at f(g(b)). Most of the cases, the subterm which have moved from left hand to right hand of a rewrite rule is not needed to evaluate again. For these cases, the reduction with marks is more efficient than the reduction without marks.

Let $\mathcal{F}^*$ be a set of marked function symbols, i.e. $\mathcal{F}^* = \{ f^* \mid f \in \mathcal{F} \}$, such that f and f^* have the same arity, the same E-strategy list, and corresponding rewrite rules, and $\mathcal{F}^* \cap \mathcal{F} = \phi$. Let $\mathcal{T}^* = \mathcal{T}(\mathcal{F} \cup \mathcal{F}^*, \mathcal{V})$ be a set of marked terms. Let $mark : \mathcal{T}^* \to \mathcal{T}^*$ be a function which marks $top(t)$, i.e. $mark(f(t_1, \ldots, t_n)) = f^*(t_1, \ldots, t_n)$. Let $erase : \mathcal{T}^* \to \mathcal{T}^*$ be a function which eliminates all marks, i.e. $erase(f^*(t_1, \ldots, t_n)) = erase(f(t_1, \ldots, t_n)) = f(erase(t_1), \ldots, erase(t_n))$. Let $contract' : \mathcal{T}^* \to \mathcal{T}^*$ be a marked version of *contract* which preserves marks of substituted subterms.

Definition 2 *Given a term $t \in \mathcal{T}^*$, if $erase(t)$ is a redex, $contract(erase(t))$ is the contractum by rewriting rule $l \to r \in \mathcal{R}$, and $\sigma : \mathcal{T}^* \to \mathcal{T}^*$ is the substitution which satisfies $t = l'\sigma$ and $erase(l') = l$, then*

$$contract'(t) = r\sigma.$$

From the definition, given an arbitrary term $t \in \mathcal{T}^*$, such that $erase(t)$ is a redex, then

$$erase(contract'(t)) = contract(erase(t)).$$

Example 3 *Let $\mathcal{R} = \{\ f(a, x) \to g(x, b)\ \}$*

$$contract'(f(a^*, b^*)) = g(b^*, b).$$

Next, we define the marked version of *eval*.

Definition 3 *Let $eval' : \mathcal{T}^* \to \mathcal{T}^*$ be a function, such that*

$$eval'(t) = \begin{cases} t & \text{if } top(t) \in \mathcal{V} \cup \mathcal{F}^* \\ reduce'(t, topElist(t)) & \text{otherwise} \end{cases}$$

$$reduce'(t, nil) = mark(t)$$

$$reduce'(t, cons(0, l)) = \begin{cases} eval'(contract'(t)) \\ \quad \text{if } erase(t) \text{ is a redex} \\ reduce'(t, l) \\ \quad \text{otherwise} \end{cases}$$

$$reduce'(t, cons(n+1, l)) = reduce'(replace(t, n+1, s), l) \qquad (n \geq 0)$$
$$\text{where} \quad s = eval'(subterm(t, n+1))$$

First, *eval'* analyzes whether $top(t)$ is marked or not. If $top(t)$ is marked, $eval'(t)$ returns t. Otherwise, *eval'* evaluates t along the E-strategy list and finally, marks down $top(t)$.

However, there is a case that the evaluation result of non marked term by *eval* differs from it by *eval'*.

Example 4 *Let*

$$\mathcal{R} = \begin{cases} f(x) \to x & (1) \\ g(b) \to c & (2) \\ a \to b & (3) \end{cases}$$

Let the E-strategy lists of f, g, and a be (1 0), (0 1), and (0) respectively. Let $t = f(g(a))$, then $eval(t) = c$, but $eval'(t) = g^*(b^*)$.

In $eval(t)$, first, the first argument $g(a)$ is rewritten to $g(b)$. Then $f(g(b))$ is rewritten to $g(b)$ by (1). Finally, $g(b)$ is rewritten to c. Therefore, $eval(t) = c$.

But, in $eval'(t)$, after $f(g^*(b^*))$ is rewritten to $g^*(b^*)$, $g^*(b^*)$ is not rewritten to c because $top(g^*(b^*)) \in \mathcal{F}^*$. Therefore, $eval'(t) = g^*(b^*)$.

The problem of this case is that the result of the one step rewriting of the whole term t by eval (which we call t') differ from the result of the one step rewriting of the whole term t' by *eval* (In above example, $t = g(b)$).

Consequently, we can predict that if the E-strategy lists of all function symbols is restricted to the conditions as follows, this problem is avoided.

Condition 1 The E-strategy list do not include 0. or

Condition 2 The last element of the E-strategy list is 0.

In the rest of this paper, we assume that the E-strategy lists of all function symbols satisfy **Condition 1** or **Condition 2**,
and show $eval(t) = erase(eval'(t))$ for an arbitrary term $t \in \mathcal{T}$.

To simplify the proof, we define eval2 and eval2'.

Definition 4 *Let* $eval2 : \mathcal{T} \rightarrow \mathcal{T}$ *be a function, such that*

$$
\begin{aligned}
eval2(t) &= reduce2(t, topElist(t)) \\
reduce2(t, nil) &= t \\
reduce2(t, cons(0, l)) &= \begin{cases} reduce2(contract(t), topElist(contract(t))) \\ \quad \textit{if } t \textit{ is a redex} \\ reduce2(t, l) \\ \quad \textit{otherwise} \end{cases} \\
reduce2(t, cons(n+1, l)) &= reduce2(replace(t, n+1, s), l) \qquad (n \geq 0) \\
&\quad \textit{where} \;\; s = reduce2(subterm(t, n+1), topElist(subterm(t, n+1)))
\end{aligned}
$$

$eval = eval2$ is trivial.

Definition 5 *Let* $eval2' : \mathcal{T}^* \rightarrow \mathcal{T}^*$ *be a function, such that*

$$
\begin{aligned}
eval2'(t) &= reduce2'(t, topElist(t)) \\
reduce2'(t, l) &= \begin{cases} t & \textit{if } top(t) \in \mathcal{V} \cup \mathcal{F}^* \\ reduce2''(t, l) & \textit{otherwise} \end{cases} \\
reduce2''(t, nil) &= mark(t) \\
reduce2''(t, cons(0, l)) &= \begin{cases} reduce2'(contract'(t), topElist(contract'(t))) \\ \quad \textit{if } erase(t) \textit{ is a redex} \\ reduce2'(t, l) \\ \quad \textit{otherwise} \end{cases} \\
reduce2''(t, cons(n+1, l)) &= reduce2'(replace(t, n+1, s), l) \qquad (n \geq 0) \\
&\quad \textit{where} \;\; s = reduce2'(subterm(t, n+1), topElist(subterm(t, n+1)))
\end{aligned}
$$

The difference between $eval'$ and $eval2'$ is that $eval2'$ checks marks at $reduce2'$. Therefore, it is easy to prove $eval' = eval2'$.

Definition 6 *We define the well-marked term as follows.*

- $x \in \mathcal{V}$ *is a well-marked term,*

- *If* $t_1, \ldots, t_n \in \mathcal{T}^*$ *are well-marked terms, then* $f(t_1, \ldots, t_n)$ *is a well-marked term.*

- $f^*(t_1, \ldots, t_n)$ *is a well-marked term if it satisfies the following conditions.*

1. $t_1, \ldots, t_n \in \mathcal{T}^*$ *are well-marked terms,*
2. *Given an arbitrary element* i *of the E-strategy list of* f^**, if* $i \neq 0$*, then* $top(t_i) \in \mathcal{V} \cup \mathcal{F}^*$*,*
3. *If the last element of the E-strategy list of* f^* *is* 0*, then* $erase(f^*(t_1, \ldots, t_n))$ *is not a redex.*

Lemma 1 *Let* $t \in \mathcal{T}^*$ *be a well-marked term, such that* $erase(t)$ *is a redex, then*

$$contract'(t) \text{ is a well}-\text{marked term.}$$

Proof There are $l \to r \in \mathcal{R}$, σ, and l' which satisfy $t = l'\sigma$, $contract'(t) = r\sigma$, and $erase(l') = l$. Because a subterm of a well-marked term is a well-marked term, $x\sigma$ is a well-marked term for all $x \in V(l')$. Because $V(r) \subseteq V(l')$ and r do not have a marked function symbol, $contract'(t)$ is a well-marked term. □

Lemma 2 *Let* $t \in \mathcal{T}^*$ *be a well-marked term, such that* $top(t) \in \mathcal{V} \cup \mathcal{F}^*$ *and* l *be a list constructed by the elements of* $topElist(t)$*, then*

$$reduce2(erase(t), l) = erase(t).$$

Proof We show this lemma by induction over the lexicographic ordering of pairs of the size of t and the length of l. When l is the empty list, it is trivial. Therefore, we show the cases of $l = cons(i, l')$. First, we show the case of $i = 0$. Because $topElist(t)$ satisfies **Condition 2** and t is a well-marked term, $erase(t)$ is not a redex. Because the induction hypothesis is $reduce2(erase(t), l') = erase(t)$, $reduce2(erase(t), l) = reduce2(erase(t), l') = erase(t)$. Next, we show the case of $i \neq 0$. We may write $t = f^*(t_1, \ldots, t_n)$. Because t_i is a well-marked term which $top(t_i) \in \mathcal{V} \cup \mathcal{F}^*$ and $top(t) \in \mathcal{F}^*$, the induction hypotheses are $reduce2(erase(t_i), topElist(erase(t_i))) = erase(t_i)$ and $reduce2(erase(t), l') = erase(t)$. Because $erase(t) = f(erase(t_1), \ldots, erase(t_n))$, $replace(erase(t), i, erase(t_i)) = erase(t)$.Therefore,

$$\begin{aligned} reduce2(erase(t), l) &= reduce2(replace(erase(t), i, erase(t_i)), l') \\ &= reduce2(erase(t), l') \\ &= erase(t) \end{aligned}$$

□

Let l and l' be lists. If l is a suffix of l', i.e. $\exists l''.l' = l''\hat{}l$, we write $l'/l = l''$.

Lemma 3 *Let* t *be a well-marked term and* l *be a suffix of* $topElist(t)$*, such that*

(a) *Given an arbitrary* $i \in topElist(t)/l$*, if* $i \neq 0$*, then* $top(subterm(t, i)) \in \mathcal{V} \cup \mathcal{F}^*$*,*

(b) *If the last element of* $topElist(t)/l$ *is* 0*, then* $erase(t)$ *is not a redex.*

If $reduce2'(t, l) = s$*, then*

(i) s *is a well-marked term and* $top(s) \in \mathcal{V} \cup \mathcal{F}^*$*,*

(ii) $reduce2(erase(t), l) = erase(s)$*.*

Proof We show this lemma by induction on the number that $reduce2'$ is called by $reduce2'(t, l)$. If $top(t) \in \mathcal{V} \cup \mathcal{F}^*$, (i) holds because $s = t$. And (ii) holds because $reduce2(erase(t), l) = erase(t)$ by Lemma 2.
Next, we show the cases of $top(t) \in \mathcal{F}$.

Case 1. Suppose that $l = nil$. Because $topElist(t) = topElist(t)/nil$, and (a) and (b) hold, (i) holds. Because $s = mark(t)$, $reduce2(erase(t), nil) = erase(t) = erase(s)$, i.e. (ii) holds.

Case 2. Suppose that $l = cons(0, l')$. If $erase(t)$ is a redex, $contract'(t)$ is a well-marked term by Lemma 1 and $reduce2'(t, l) = reduce2'(contract'(t), topElist(contract'(t))) = s$. Also $contract'(t)$ and $topElist(contract'(t))$ satisfy the conditions (a) and (b). By the induction hypothesis for $reduce2'(contract'(t), topElist(contract'(t))) = s$,

(1) s is a well-marked term and $top(s) \in \mathcal{V} \cup \mathcal{F}^*$,

(2) $reduce2(erase(contract'(t)), topElist(contract'(t))) = erase(s)$

By (1), (i) holds. By (2),

$$\begin{aligned} reduce2(erase(t), l) &= reduce2(contract(erase(t)), \\ &\quad\; topElist(contract(erase(t)))) \\ &= reduce2(erase(contract'(t)), topElist(contract'(t))) \\ &= erase(s) \end{aligned}$$

Therefore, (ii) holds.
If $erase(t)$ is not a redex, $reduce2'(t, l) = reduce2'(t, l') = s$. Also t and l' satisfy the condition (a) and (b). By the induction hypothesis for $reduce2'(t, l') = s$,

(3) s is a well-marked term and $top(s) \in \mathcal{V} \cup \mathcal{F}^*$,

(4) $reduce2(erase(t), l') = erase(s)$

By (3), (i) holds. By (4), $reduce2(erase(t), l) = reduce2(erase(t), l') = erase(s)$, i.e. (ii) holds.

Case 3. Suppose that $l = cons(i, l')$ $(i \neq 0)$. Let $t_i = subterm(t, i)$ and $s' = reduce2'(t_i, topElist(t_i))$, then $reduce2'(t, l) = reduce2'(replace(t, i, s'), l') = s$. Also t_i and $topElist(t_i)$ satisfy the condition (a) and (b). By the induction hypothesis for $reduce2'(t_i, topElist(t_i)) = s'$,

(1) s' is a well-marked term and $top(s') \in \mathcal{V} \cup \mathcal{F}^*$

(2) $reduce2(erase(t_i), topElist(t_i)) = erase(s')$

By (1), $replace(t, i, s')$ and l' satisfy the condition (a) and (b). By the induction hypothesis for $reduce2'(replace(t, i, s'), l') = s$,

(3) s is a well-marked term and $top(s) \in \mathcal{V} \cup \mathcal{F}^*$,

(4) $reduce2(erase(replace(t, i, s')), l') = erase(s)$

By (3), (i) holds. By (2) and $subterm(erase(t), i) = erase(t_i)$,

$$\begin{aligned} & reduce2(subterm(erase(t), i), topElist(subterm(erase(t), i))) \\ &\quad = reduce2(erase(t_i), topElist(t_i)) \\ &\quad = erase(s') \end{aligned}$$

By this and (4),

$$\begin{aligned} reduce2(erase(t), l) &= reduce2(replace(erase(t), i, erase(s')), l') \\ &= reduce2(erase(replace(t, i, s')), l') \\ &= erase(s) \end{aligned}$$

i.e. (ii) holds. □

Lemma 4 *Let $t \in \mathcal{T}^*$ be a well-marked term, l be a suffix of $topElist(t)$. If $reduce2(erase(t), l) = s$, then there is the s' which satisfies $reduce2'(t, l) = s'$ and $erase(s') = s$.*

Proof We can prove this lemma as in Lemma 3. □

Theorem 1 *Let $t \in \mathcal{T}$. $eval2(t)$ terminates iff $eval2'(t)$ terminates. And this time, $eval2(t) = erase(eval2'(t))$.*

Proof By Lemma 3 and Lemma 4. □

Corollary 1 *Let $t \in \mathcal{T}$. $eval(t)$ terminates iff $eval'(t)$ terminates. And this time, $eval(t) = erase(eval'(t))$.*

Therefore, we can adopt the reduction with marks, if we can assure the E-strategy lists of all function symbols satisfy **Condition 1** or **Condition 2**.

5 Related Work

The idea of the well-marked term is originally from [6]. In [6], the occurrences of subterms which are known to be in strong head-normal form are marked and this marks are used for future searches of indexes.

6 Conclusion

In this paper, we showed that the result of the reduction with marks coincides with the result of the reduction without marks if the E-strategies satisfy the following conditions,

Condition 1 The E-strategy list do not include 0. or

Condition 2 The last element of the E-strategy list is 0.

In the most of the practical usage, the above conditions are satisfied.

7 Acknowledgements

We are specially grateful to Prof. Yoshihito Toyama for his valuable comments and helpful suggestions on previous versions of this paper.

Bibliography

[1] N. Dershowitz and J.-P. Jouannaud, Rewrite systems, Handbook of Theoretical Computer Science, vol.B, p243-320, North-Holland, 1990.

[2] K. Futatsugi, J.A. Goguen, J.-P. Jouannaud, and J. Meseguer, Principles of OBJ2, Proc. of 12th ACM Symposium on Principles of Programming Languages, p52-66, 1985.

[3] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud, Introducing OBJ, Technical Report SRI-CSL-93-03, SRI International, 1992.

[4] J.W. Klop, Term rewriting systems, Handbook of Logic in Computer Science, vol.2, p1-116, Oxford University Press, 1992.

[5] A. Nakagawa, R. Diaconescu, K. Futatsugi, and T. Sawada, CafeOBJ User's Manual, ftp://ftp.ipa.go.jp/pub/Cafe/manual-en.ps

[6] Y. Toyama, S. Smetsers, M.v. Eekelen, and R. Plasmeijer, The Functional Strategy and Transitive Term Rewriting Systems, Term Graph Rewriting: Theory and Practice, p61-75, John Wiley & Sons, 1993.