Article ID: 1002－1175(2004)04-0501-11

Operational Semantics Modelling Semantic Action Refinement for Processes with Interrupt

YUAN Hong WU Jin-Zhao

(*Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu* 610040, *China*)

(Received 21 July 2003; Revised 12 April 2004)

Abstract An operational semantics of LOTOS including an interrupt and a refinement operator is given as SOS rules such that it corresponds to the denotational semantics, i.e. the transition system derived from the denotational semantics is bisimilar to the transition system of the operational semantics. The idea behind the operational semantics is that the refined action is renamed with a fresh name and all involved choices are triggered.

Key words action refinement, denotational semantics, operational semantics, transition system

CLC TP301.2

1 Introduction

Process algebras[1] are a well-established paradigm to specify concurrent systems. In the last decade process algebras with an action refinement operator to support the top-down design of systems and to specify the implementation mechanism of a procedure call have been studied intensively, see Ref. [2] for an overview. The theory of action refinement is well established for denotational true concurrency semantics of process algebras, i.e. refinement operators are employed in event structures, petri nets and in other models of concurrency[3－5]. The philosophy behind these refinements is that an action a, refined by a process B, is interpreted as the procedure call described by B whenever a is executed.

When presenting an operational semantics of process algebras with a refinement operator in terms of transition systems the action refinement operator is often handled on the process term level by syntactic substitution[6]. It is well known that this type of operational semantics is incompatible with denotational semantics[7].

This paper aims to reconcile the expected behavior (denotational semantics) and the implementation (operational semantics). We present an operational semantics of action refinement which corresponds to the denotational one. We do so by adding transition rules that simulate the execution of a refined action within the refinement construction. For example, the process $(a.B)[a\rightarrow a_1.a_2]$ evolves into $(a.B)[a\rightarrow a_2]$ by performing a_1 provided a does not occur in B. If a occurs in B as in $a.a$ and we proceed in the same way we loose the information to which term the a's occurring in B has to be refined. i.e. $(a.a)[a\rightarrow a_1.a_2]$ may not evolve to $(a.a)[a\rightarrow a_2]$ by performing a_1. To circumvent this problem, we rename the executed action a by a fresh action name a' and extend the refinement construct by an additional refinement, i.e. $(a.a)[a\rightarrow a_1.a_2]$ evolves into $(a'.a)[a\rightarrow a_1.a_2],[a'\rightarrow a_2]$ by performing a_1, where $a'\neq a$. This is still not sufficient, since we have to trigger the choice when an action is renamed in the choice operator, i.e. $(b+(a.B))[a\rightarrow a_1.a_2]$ evolves into $(a'.B)[(a\rightarrow a_1.a_2),(a'\rightarrow a_2)]$ by performing a_1. Otherwise $(b+(a.B))[a\rightarrow a_1.a_2]$ would

be able to perform b after a_1. This is counterintuitive, and therefore the choice has to be triggered. We model the triggering of the choices and the renaming of the action by extending the transition relation using additional transition labels. In these transitions the triggering and the renaming are modelled simultaneously. The advantage of this approach is that the syntax of the language does not have to be extended to define the operational semantics. Furthermore, an interrupt operator can be adequately modelled in our setting.

We illustrate our ideas for the process algebra LOTOS[8] which we extend by a refinement expression. Furthermore, we introduce a refinement operator for extended bundle event structures[9] to present a denotational semantics. We justify our operational semantics by showing that the transition system of the operational semantics is bisimilar to the transition system derived from the denotational semantics.

2 The process algebra

Let $\surd$ and τ be two different elements, which indicate the *terminating* and the *internal action*. Furthermore, let *Obs* be a set such that $\surd$, $\tau \notin Obs$ and $|Obs| > |\mathbb{N}|$. $\mathbb{N}$ denotes the set of natural numbers and $|\mathbb{N}|$ the cardinal number of $\mathbb{N}$. We call *Obs* the set of *observable actions*. The set of all actions *Act* is defined by $Act = \{\surd, \tau\} \cup Obs$. Assume a fixed countable set of process variables *Var* which is disjoint from *Act*.

2.1 Syntax

The expressions Expr of Ref-LOTOS, ranged over by B, are defined by the following BNF-grammar.

$$B ::= 0 \mid 1 \mid a.B \mid \tau.B \mid B;\ B \mid B + B \mid B[\!> B \mid$$
$$B \parallel_A B \mid B[(a \to B_a)^{a \in A}] \mid B \setminus A \mid x,$$

where $x \in Var$, $a \in Obs$ and $A \subseteq Obs$ with $|A| \leqslant |\mathbb{N}|$. A process is a pair $\langle decl, B \rangle$ consisting of a declaration $decl: Var \to \text{Expr}$ and an expression $B \in \text{Expr}$. Let PA denote the set of all processes.

We sometimes omit 1 in the expression, i.e. we write a instead of $a.1$. The intuitive meaning of the refinement expression $B[(a \to B_a)^{a \in A}]$ is that it behaves like process B except that every execution of action a in A is substituted by the behavior of B_a. The intuitive meaning of the other operators is the standard one[8].

For simplicity we do not consider the relabelling operator in the definition of Ref-LOTOS. It is no problem to extend the theory by a relabelling operator where only relabelling functions with a countable domain are allowed.

2.2 Denotational semantics

We take a special class of bundle event structures[9] with some modifications from Ref. [11] as a model for Ref-LOTOS. We write $\mathcal{P}(M)$ to denote the power set of M and $M_1 \rightharpoonup M_2$ to denote the set of all partial functions from M_1 to M_2. Furthermore, the domain $dom: (M_1 \rightharpoonup M_2) \to \mathcal{P}(M)$ is defined by $dom(f) = \{m \in M_1 \mid f(m) \text{ is defined}\}$.

Definition 1 Let E be a countable set. A finite, monotone approximation of E is a sequence $(E_i)_{i \in \mathbb{N}}$ such that $\bigcup_{i \in \mathbb{N}} E_i = E \wedge \forall k: E_k \subseteq E_{k+1} \wedge |E_k| < \infty$.

Definition 2 (Closed bundle event structure) Suppose, $*$, $*_1$ and $*_2$ are pairwise different. Assume a fixed countable set of events $\mathcal{U}_E$ such that $* \in \mathcal{U}_E$, $* \notin \mathcal{U}_E$ and

$$\forall e, e' \in \mathcal{U}_E: (e, e'), (*_1, e), (*_2, e), (*, e), (e, *) \in \mathcal{U}_E \quad .$$

A closed bundle event structure $(c.b.e.s.) \in = (E, \rightsquigarrow, \mapsto, l)$ is an element of $\mathcal{P}(\mathcal{U}_E) \times \mathcal{P}(\mathcal{U}_E \times \mathcal{U}_E) \times \mathcal{P}(\mathcal{P}(\mathcal{U}_E) \times \mathcal{U}_E) \times (\mathcal{U}_E \rightharpoonup Act)$ such that

$$- \rightsquigarrow \subseteq (E \times E) \setminus \{(e, e) \mid e \in E\},$$

$- \mapsto \subseteq \mathcal{P}(E) \times E$,

$- dom(l) = E$,

$- \forall X \subseteq E, e \in E: X \mapsto e \Rightarrow (\forall e', e'' \in X: e' \neq e'' \Rightarrow e' \rightsquigarrow e'')$,

$-X \mapsto e$ whenever $X \subseteq E$ and there is a finite, monotone approximation $(E_n)_{n \in \mathbb{N}}$ of E such that $\forall n: \exists X': X' \mapsto e \wedge X' \cap E_n = X \cap E_n$.

Let **CEBES** denote the set of all closed bundle event structures.

We call E the set of events, $\rightsquigarrow$ the (irreflexive) asymmetric conflict relation, $\mapsto$ the bundle relation and l the action-labelling function of ε. Hereafter, we consider ε to be $(E, \rightsquigarrow, \mapsto l)$, ε_i to be $(E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ and when necessary ε to be $(E_\varepsilon, \rightsquigarrow_\varepsilon, \mapsto_\varepsilon, l_\varepsilon)$.

Definition 3 (Restriction of a c.b.e.s) Suppose $\varepsilon \in$ **CEBES** and $E' \subseteq E$. Then the restriction of ε to E' ($\varepsilon \restriction E'$) is the c.b.e.s. $(E', \rightsquigarrow \cap (E' \times E'), \mapsto', l' \restriction E)$, where $\mapsto' = \{(X \cap E', e) \mid e \in E' \wedge X \mapsto e\}$.

Definition 4 (Order on CEBES) Let $\varepsilon_i \in$ **CEBES**. Then $\varepsilon_1 \trianglelefteq \varepsilon_2$ *if and only if* $E_1 \subseteq E_2$ *and* $\varepsilon_1 = \varepsilon_2 \restriction E_1$.

Remark 5 $\langle$**CEBES**, $\trianglelefteq\rangle$ is a pointed complete partial order (pointed cpo)[10] as we proved in Ref. [11].

Definition 6 The set of initial events of the *c.b.e.s.* ε is defined by $\text{init}(\varepsilon) = \{e \in E \mid \neg(\exists X: X \mapsto e\}$. The set of successful termination events of the *c.b.e.s.* ε is defined by $\text{exit}(\varepsilon) = \{e \in E \mid l(e) = \surd\}$.

As **CEBES** is closed with respect to a slight modification of the standard operators of Ref. [9], we make use of them. For the refinement operator we take an adapted version of Ref. [12, 13].

Definition 7 (**Operators on** ε) Let $A \subseteq Obs$ with $|A| \leqslant |\mathbb{N}|$. Then define $\hat{.}: (Obs \cup \{\tau\}) \times$ **CEBES** $\to$ **CEBES** with $a.\hat{\varepsilon} = (\{\circ\} \cup (\{*_1\} \times E), \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$\tilde{\rightsquigarrow} = \{((*_1, e), (*_1, e')) \mid (e, e') \in \rightsquigarrow\},$$

$$\tilde{\mapsto} = \{(\{*_1\} \times X, (*_1, e)) \mid (X, e) \in \mapsto\} \cup \{(\{\circ\}, (*_1, e)) \mid e \in \text{init}(\varepsilon)\},$$

$$l(\tilde{e}) = \begin{cases} l(e) & \text{if } \tilde{e} = (*_1, e), \\ a & \text{if } \tilde{e} = \circ. \end{cases}$$

$+\hat{}$: **CEBES** $\times$ **CEBES** $\to$ **CEBES** with $\varepsilon_1 + \hat{} \varepsilon_2 = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$\tilde{E} = (\{*_1\} \times E_1) \cup (\{*_2\} \times E_2),$$

$$\tilde{\rightsquigarrow} = \{((*_i, e), (*_j, e')) \mid i \neq j \vee (i = j \wedge e \rightsquigarrow_i e'\},$$

$$\tilde{\mapsto} = \{(\{*_i\} \times X, (*_i, e)) \mid X \mapsto_i e\},$$

$$l((*_i, e)) = l_i(e).$$

$;\hat{}$: **CEBES** $\times$ **CEBES** $\to$ **CEBES** with $\varepsilon_1 ;\hat{} \varepsilon_2 = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$\tilde{E} = (\{*_1\} \times E_1) \cup (\{*_2\} \times E_2),$$

$$\tilde{\rightsquigarrow} = \{((*_i, e), (*_i, e')) \mid e \rightsquigarrow_i e' \vee (i = 1 \wedge e \neq e' \wedge e, e' \in \text{exit}(\varepsilon_i))\},$$

$$\tilde{\mapsto} = \{(\{*_i\} \times X, (*_i, e)) \mid X \mapsto_i e\} \cup \{(\{*_1\} \times \text{exit}(\varepsilon_1), (*_2, e)) \mid e \in \text{init}(\varepsilon_2)\},$$

$$l((*_i, e)) = \begin{cases} l_1(e) & \text{if } i = 1 \wedge e \notin \text{exit}(\varepsilon_1), \\ \tau & \text{if } i = 1 \wedge e \in \text{exit}(\varepsilon_1), \\ l_2(e) & \text{if } i = 2. \end{cases}$$

$\hat{\rhd}$: **CEBES**$\times$ **CEBES** $\rightarrow$ **CEBES** with $\varepsilon_1 \hat{\rhd} \varepsilon_2 = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$\tilde{E} = (\{*_1\} \times E_1) \cup (\{*_2\} \times E_2),$$

$$\tilde{\rightsquigarrow} = \{((*_i, e), (*_i, e')) \mid e \rightsquigarrow_i e'\} \cup (\{*_1\} \times E_1) \times (\{*_2\} \times \text{init}(\varepsilon_2)) \cup (\{*_2\} \times E_2) \times (\{*_1\} \times \text{exit}(\varepsilon_1)),$$

$$\tilde{\mapsto} = \{(\{*_i\} \times X, (*_i, e)) \mid X \mapsto_i e\},$$

$$l((*_i, e)) = l_i(e).$$

$\hat{\|}_A$: **CEBES**$\times$ **CEBES** $\rightarrow$ **CEBES** with $\varepsilon_1 \hat{\|}_A \varepsilon_2 = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$\tilde{E} = (E_1^f \times \{*\}) \cup (\{*\} \times E_2^f) \cup E^s,$$

$$E_i^f = \{e \in E_i \mid l_i(e) \notin A \cup \{\surd\}\},$$

$$E^s = \{(e_1, e_2) \in E_1 \times E_2 \mid l_1(e_1) = l_2(e_2) \in A \cup \{\surd\}\},$$

$$\tilde{\rightsquigarrow} = \{((e_1, e_2), (e'_1, e'_2)) \mid e_1 \rightsquigarrow_1 e'_1 \vee e_2 \rightsquigarrow_2 e'_2 \vee (e_1 = e'_1 \neq * \wedge e_2 \neq e'_2) \vee (e_2 = e'_2 \neq * \wedge e_1 \neq e'_1)\},$$

$$\tilde{\mapsto} = \{(X, (e_1, e_2)) \mid \exists X_1 \subseteq E_1 : X_1 \mapsto_1 e_1 \wedge X = \{(e, e') \in \tilde{E} \mid e \in X_1\}\} \cup \{(X, (e_1, e_2)) \mid \exists X_2 \subseteq E_2 : X_2 \mapsto_2 e_2 \wedge X = \{(e, e') \in \tilde{E} \mid e \in X_2\}\},$$

$$l((e_1, e_2)) = \begin{cases} l_1(e_1) & \text{if } e_2 = *, \\ l_2(e_2) & \text{otherwise.} \end{cases}$$

$\widehat{Ref}_A$: **CEBES**$\times (A \rightarrow$ **CEBES**$) \rightarrow$ **CEBES** with $\widehat{Ref}_A(\varepsilon, \theta) = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$\tilde{E} = \{(e, e') \mid e \in E \wedge l(e) \in A \wedge e' \in E_{\theta(l(e))}\} \cup \{(e, e) \in E \times E \mid l(e) \notin A\},$$

$$\tilde{\rightsquigarrow} = \{((e_1, e'_1), (e_2, e'_2)) \mid e_1 \rightsquigarrow_2 \vee (e_1 = e_2 \wedge l(e_1) \in A \wedge e'_1 \neq e'_2 \wedge (e'_1 \rightsquigarrow_{\theta(l(e_1))} e'_2 \vee e'_2 \in \text{exit}(\theta(l(e_1)))))\},$$

$$\tilde{\mapsto} = \{(\{e\} \times X', (e, e')) \mid l(e) \in A \wedge X' \mapsto_{\theta(l(e))} e'\} \cup \{(X, (e, e')) \mid \exists X : X \mapsto e \wedge (l(e) \in A \Rightarrow e' \in \text{init}(\theta(l(e)))) \wedge \tilde{X} = \{(\tilde{e}, \tilde{e}') \in \tilde{E} \mid \tilde{e} \in X \wedge (l(\tilde{e}) \in A \Rightarrow \tilde{e}' \in \text{exit}(\theta(l(\tilde{e}))))\}\},$$

$$l((e, e')) = \begin{cases} l(e) & \text{if } l(e) \notin A, \\ l_{\theta(l(e))}(e') & \text{if } l(e) \in A \wedge l_{\theta(l(e))}(e') \neq \surd, \\ \tau & \text{if } l(e) \in A \wedge l_{\theta(l(e))}(e') = \surd. \end{cases}$$

$\hat{\setminus}A$: **CEBES** $\rightarrow$ **CEBES** with $\varepsilon_1 \hat{\setminus} A = (\tilde{E}, \tilde{\rightsquigarrow}, \tilde{\mapsto}, l)$ where

$$l(\tilde{e}) = \begin{cases} l(e) & \text{if } l(e) \notin A, \\ \tau & \text{if } l(e) \in A. \end{cases}$$

Lemma 8 All operators of Definition 7 are continuous with respect to $\trianglelefteq$.

Definition 9 Let Φ_ρ: Expr$\times (Var \rightarrow$ **CEBES**$) \rightarrow$ **CEBES** be defined as follows

$$\Phi_\rho(\mathbf{1}) = (\emptyset, \emptyset, \emptyset, \emptyset), \qquad \Phi_\rho(\mathbf{0}) = (\{\circ\}, \emptyset, \emptyset, \{(\circ, \surd)\}),$$

$$\Phi_\rho(a.B) = a\ \hat{.}\ \Phi_\rho(B), \qquad \Phi_\rho(\tau.B) = \tau\ \hat{.}\ \Phi_\rho(B),$$

$$\Phi_\rho(B_1 + B_2) = \Phi_\rho(B_1) \hat{+} \Phi_\rho(B_2), \qquad \Phi_\rho(B_1; B_2) = \Phi_\rho(B_1)\ \hat{;}\ \Phi_\rho(B_2),$$

$$\Phi_\rho(B_1 [\!> B_2) = \Phi_\rho(B_1) [\!>\hat{} \Phi_\rho(B_2), \qquad \Phi_\rho(B_1 \|_A B_2) = \Phi_\rho(B_1)\ \hat{\|}_A \Phi_\rho(B_2),$$

$$\Phi_\rho(B \setminus A) = \Phi_\rho(B)\ \hat{\setminus} A, \qquad \Phi_\rho(x) = \rho(x),$$

$$\Phi_\rho(B[(a \to B_a)^{a\in A}]) = \widehat{Ref}_A(\Phi_\rho(B), (a \to \Phi_\rho(B))^{a\in A}).$$

Remark 10 From Lemma 8 it can be easily verified that $\Phi_\rho(B)$ is continuous for every $B \in$ Expr.

Assume $decl$: $Var \to$Expr. Then define $\mathscr{F}_{decl}$: ($Var \to$**CEBES**)$\to$($Var \to$**CEBES**) with $\mathscr{F}_{decl}(\rho)_{(x)} = \Phi_\rho(decl(x))$. From Remark 10 it follows that $\mathscr{F}_{decl}$ is continuous. Therefore, from the cpo theory[1] we get Ψ: ($Var \to$ Expr)$\to$($Var \to$**CEBES**) with $\Psi(decl) = \mathrm{fix}(\mathscr{F}_{decl}) = \bigsqcup_n \mathscr{F}^n_{decl}(\perp)$ is well defined.

Definition 11 (Denotational Semantics) Define Φ: **PA**$\to$**CEBES** by $\Phi(\langle decl, B\rangle) = \Phi(B)_{\Psi(decl)}$.

2.3 Operational semantics

The operational semantics is given by a transition System.

Definition 12 (Transition system) A transition system is a quadruple $(S, I, \to, \bar{s})$ with

— S, a non-empty set of states,

— L, a set of labels,

— $\to \subseteq S \times L \times S$, a transition relation,

— $\bar{s} \in S$, the initial state.

We will write $p \xrightarrow{\gamma} q$ rather than $(p, \gamma, q) \in \to$. The class of all transition systems is denoted by **TS**.

Before we introduce the transition rules for Ref-LOTOS, we have to determine all actions which occur in a given expression.

Definition 13 Let $\mathscr{P}_{count}(Obs) = \{A \subseteq Obs : |A| \leqslant |\mathrm{N}|\}$. The function $\mathscr{L}$: Expr$\to\mathscr{P}_{count}(Obs)$ is defined as follows.

$$\mathscr{L}(0) = \mathscr{L}(1) = \mathscr{L}(x) = \emptyset, \qquad \mathscr{L}(a.B) = \{a\} \cup \mathscr{L}(B),$$

$$\mathscr{L}(\tau.B) = \mathscr{L}(B), \qquad \mathscr{L}(B_1 \|_A B_2) = \mathscr{L}(B_1) \cup \mathscr{L}(B_2) \cup A,$$

$$\mathscr{L}(B \setminus A) = \mathscr{L}(B) \cup A, \qquad \mathscr{L}(B[(a \to B_a)^{a\in A}]) = \mathscr{L}(B) \cup A \cup \bigcup_{a\in A} \mathscr{L}(B_a),$$

$$\mathscr{L}(B_1 + B_2) = \mathscr{L}(B_1; B_2) = \mathscr{L}(B_1 [\!> B_2) \mathscr{L}(B_1) \cup \mathscr{L}(B_2).$$

Then define $\mathscr{L}$: **PA**$\to\mathscr{P}_{count}(Obs)$ by $\mathscr{L}(\langle \boldsymbol{decl}, \boldsymbol{B}\rangle) = \mathscr{L}(B) \cup \bigcup_{x\in Var} \mathscr{L}(decl(x))$.

The transition rules of $\to \subseteq$Expr$\times$($Act \cup (Obs \times Obs)$)$\times$Expr are presented in Tab. 1.

Here $\xrightarrow{(a,b)}$ means that one executable action a is renamed by b and all choices which would be triggered by the execution of this a are taken. An interpretation is that action a has been started and that it will finish by performing action b.

In rule A_2 the action a is relabelled by b. Rule C makes no difference between labels from Act and ($Obs \times Obs$), hence, in both cases the choice is triggered.

The interrupt operator $[\!>$ also interrupts the process if a (a, b) transition takes place, since the starting of an action triggers the interruption. The parallel operator works asynchronously exactly for those actions which are not in the synchronization set A united with the termination action. For synchronization purposes, (a, b) is regarded to be in A if and only if $a \in A$. The rule for synchronization is divided into P_2 and P_3, since in the case where (a, b)

is performed we have to take care that further derivations will synchronize on b.

Tab. 1 Transition rules

let γ be an element of $Act \cup (Obs \times Obs)$

$$T: \frac{}{1 \xrightarrow{\surd} 0} \qquad A_1: \frac{}{a.B \xrightarrow{a} B} \qquad A_2: \frac{a, b \in Obs}{a.B \xrightarrow{(a,b)} b.B} \qquad C: \frac{B_1 \xrightarrow{\gamma} B_2}{B_1 + B_2 \xrightarrow{\gamma} B'} \; B_2 + B_1 \xrightarrow{\gamma} B'$$

$$S_1: \frac{B_1 \xrightarrow{\gamma} B'_1 \quad \gamma \neq \surd}{B_1; B_2 \xrightarrow{\gamma} B'_1; B_2} \qquad S_2: \frac{B_1 \xrightarrow{\surd} B'_1}{B_1; B_2 \xrightarrow{\tau} B_2}$$

$$I_1: \frac{B_1 \xrightarrow{\gamma} B'_1 \quad \gamma \neq \surd}{B_1 \rhd B_2 \xrightarrow{\gamma} B'_1 \rhd B_2} \qquad I_2: \frac{B_1 \xrightarrow{\surd} B'_1}{B_1 \rhd B_2 \xrightarrow{\surd} B'_1} \qquad I_3: \frac{B_2 \xrightarrow{\gamma} B'_2}{B_1 \rhd B_2 \xrightarrow{\gamma} B'_2}$$

$$P_1: \frac{B_1 \xrightarrow{\gamma} B'_1 \quad \gamma \notin \{\surd\} \cup A \cup (A \times Obs)}{B_1 \|_A B_2 \xrightarrow{\gamma} B'_1 \|_A B_2} \; B_2 \|_A B_1 \xrightarrow{\gamma} B_2 \|_A B'_1$$

$$P_2: \frac{B_1 \xrightarrow{\gamma} B'_1 \quad B_2 \xrightarrow{\gamma} B'_2 \quad \gamma \notin \{\surd\} \cup A}{B_1 \|_A B_2 \xrightarrow{\gamma} B'_1 \|_A B'_2} \qquad P_3: \frac{B_1 \xrightarrow{(a,b)} B'_1 \quad B_2 \xrightarrow{(a,b)} B'_2 \quad a \in A}{B_1 \|_A B_2 \xrightarrow{(a,b)} B'_1 \|_{A \cup \{b\}} B'_2}$$

$$Ref_1: \frac{B \xrightarrow{\gamma} B' \quad \gamma \notin A \cup (A \times Obs)}{B[(a \to B_a)^{a \in A}] \xrightarrow{\gamma} B'[(a \to B_a)^{a \in A}]}$$

$$Ref_2: \frac{B \xrightarrow{(\hat{a},b)} B' \quad \hat{a} \in A \quad b \notin A \cup \mathscr{A}(\langle decl, B \rangle) \quad B_{\hat{a}} \xrightarrow{\gamma} B'' \quad \gamma \neq \surd}{B[(a \to B_a)^{a \in A}] \xrightarrow{\gamma} B'[(a \to B_a)^{a \in A}, (b \to B'')]}$$

$$Ref_3: \frac{B \xrightarrow{\hat{a}} B' \quad \hat{a} \in A \quad B_{\hat{a}} \xrightarrow{\surd} B''}{B[(a \to B_a)^{a \in A}] \xrightarrow{\tau} B'[(a \to B_a)^{a \in A}]}$$

$$Res_1: \frac{B \xrightarrow{\gamma} B' \quad \gamma \notin \{\surd\} \cup (A \times Obs)}{B \setminus A \xrightarrow{\gamma} B' \setminus A} \qquad Res_2: \frac{B \xrightarrow{a} B' \quad a \in A}{B \setminus A \xrightarrow{\tau} B' \setminus A}$$

$$Rec: \frac{decl(x) \xrightarrow{\gamma} B'}{x \xrightarrow{\gamma} B'}$$

Rule Ref_1 considers the case when an action is performed which is not refined. In this case only the term which is refined is modified. Rule Ref_2 considers the case when an action $\hat{a}$ is performed which is refined by a non-terminating process. Here $\hat{a}$ is renamed by a fresh action name, which is ensured by $b \notin A \cup \mathscr{A}(\langle decl, B \rangle)$. Such a b always exists, since $|Obs| > |\mathbb{N}|$. Furthermore, we keep all present refinements and add a new refinement for the performed action $\hat{a}$, which is now labelled with b. The case when the refined process terminates is considered in rule Ref_3. In this case $\hat{a}$ terminates. Therefore, it has to be removed, which is done by taking the transition labelled with $\hat{a}$. The refinement remains unaffected.

The remaining rules are the standard ones.

By performing (a, b) rules P_1, P_3, Ref_1, Res_1 sometimes derive processes with undesired behavior. For example, $a.1 \|_{\{b\}} 1 \xrightarrow{(a,b)} b.1 \|_{\{b\}} 1$, which deadlocks. The undesired behavior can only appear if action b occurs in the considered process.

Such undesired behavior does not cause any problems, since we are only interested in transitions labelled with elements from Act. The only situation where we need (a, b) transitions is in rule Ref_2, but there we take care that b is fresh.

Example 14 We just illustrate the transition rules by presenting a derivation path of $(a.\mathbf{1} \parallel_a (a.\mathbf{1}+b.0))[a\to c.1][c\to d.1]$ in Fig. 1.

$$(a.1 \parallel_{\{a\}} (a.\mathbf{1}+b.0))[a\to c.1][c\to d.1]$$
$$\downarrow d$$
$$(c.\mathbf{1} \parallel_{\{a,c\}} c.\mathbf{1})[a\to c.1, c\to c'.1][c\to d.\mathbf{1}, c'\to 1]$$
$$\downarrow \tau$$
$$(c''.\mathbf{1} \parallel_{\{a,c,c''\}} c''.\mathbf{1})[a\to c.1, c\to c'.1, c''\to 1][c\to d.\mathbf{1}, c'\to 1]$$
$$\downarrow \tau$$
$$(\mathbf{1} \parallel_{\{a,c,c''\}} \mathbf{1})[a\to c.1, c\to c'.1, c''\to 1][c\to d.\mathbf{1}, c'\to 1]$$
$$\downarrow \surd$$
$$(\mathbf{0} \parallel_{\{a,c,c''\}} \mathbf{0})[a\to c.1, c\to c'.1, c''\to 1][c\to d.\mathbf{1}, c'\to 1]$$

Fig. 1　Example of derivation

In the first step action a is renamed with c. This seems strange, since c will be refined to d. But it does not cause a problem, because the operational semantics refines c to c' before it gets under the influence of $[c\to d.\mathbf{1}]$. The triggering of the choice by the renaming transition is also illustrated by the first derivation.

Definition 15 (Operational Semantics)　The operational semantics $\mathscr{O}$: **PA**→**TS** is given by $\mathscr{O}(\langle decl, B\rangle) = (\mathrm{Expr}, Act, \to_o, B)$ where $\to_o = \to_{decl} \cap (\mathrm{Expr} \times Act \times \mathrm{Expr})$.

We still have to argue that the presented operational semantics is reasonable. The fact that the transitions labelled with elements of $Obs \times Obs$ always generate an infinite transition system is problematic. More precisely, if process B can perform $a \in Obs$ then it can perform (a, b) for every $b \in Obs$, which yields an uncountable infinitely branching transition system. Moreover, it is also possible to derive undesirable infinitely long derivations, for example $a_1.\mathbf{1} \xrightarrow{(a_1, a_2)} \cdots \xrightarrow{(a_i, a_i+1)} \cdots$. Nevertheless, this causes no problem since only the transitions labelled by elements of Act are used to define the operational semantics.

But we still have an infinitely branching transition system by rule Ref_2, for example $a[a\to a_1.a_2] \xrightarrow{a_1} b[a\to a_1, a_2, b\to a_2]$ for all $b \in Obs \setminus \{a\}$. There is also no problem, since all the expressions are α-equivalent, i.e. they can be translated into each other by action renaming of the bounded actions. In other words, in rule Ref_2 it is not important which $b \notin A \cup \mathscr{A}(\langle decl, B\rangle)$ is chosen, since all choices yield α-equivalent expressions.

It is also possible to apply similar techniques of Ref. [16] to obtain a unique action renaming. This will make α-conversion obsolete.

3　Connection between the semantics

In this section we argue that the operational semantics is a meaningful semantics with respect to the denotational semantics. For this reason, we associate a transition system with a *c.b.e.s.* Then we show that the transition system derived from the denotational semantics is bisimilar to the operational semantics.

3.1　Transition systems from CEBES

Definition 16 (Remainder of *c.b.e.s.*)　Let $\varepsilon \in$ **CEBES** and $e \in \mathrm{init}(\varepsilon)$. Then the remainder $\varepsilon_{[e]}$ of ε is given by $(E', \rightsquigarrow', \mapsto', l')$ where

$$E' = \{e' \in E \mid e' \neq e \wedge \neg(e' \rightsquigarrow e)\},$$
$$\rightsquigarrow' = \rightsquigarrow \cap (E' \times E'),$$
$$\mapsto' = \{(X \cap E', e') \mid e' \in E' \wedge X \mapsto e' \wedge e \notin X\},$$
$$l' = l \upharpoonright E'.$$

Definition 17　The transition relation $\to_\varepsilon \subseteq$ **CEBES** × *Act* × **CEBES** is defined by $\to_\varepsilon = \{(\varepsilon, l(e), \mathrm{E}_{[e]}) \mid$

$\varepsilon \in \mathbf{ECEBES} \wedge e \in init(\varepsilon)\}$.

3.2 Bisimilarity

Definition 18 (Bisimulation) Suppose $T_i = (S_i, L, \rightarrow_i, \bar{s}_i)$ are two labelled transition systems. Then they are bisimilar, denoted $T_1 \sim T_2$, if there is bisimulation, i.e. a relation $\mathscr{R} \subseteq S_1 \times S_2$ with $(\bar{s}_1, \bar{s}_2) \in \mathscr{R}$ and for which for all $(p_1, p_2) \in \mathscr{R}$ we have for all $i, j \in \{1, 2\}$ with $i \neq j$:

if $p_i \xrightarrow{a}_i q_i$ then there is $q_j \in S_j$ such that $(q_1, q_2) \in \mathscr{R}$ and $p_j \xrightarrow{a}_j q_j$.

Theorem 19 Suppose $\langle decl, B\rangle \in \mathbf{PA}$. Then $\mathscr{O}(\langle decl, B\rangle)$ and $(\mathbf{CEBES}, Act, \rightarrow_\varepsilon, \Phi(\langle decl, B\rangle))$ are bisimilar.

Proof (**Sketch**) An event based transition relation is defined, especially to handle unguarded recursion. Then we show that its corresponding transition system is bisimilar to $\mathscr{O}(\langle decl, B\rangle)$ and in addition bisimilar to $(\mathbf{CEBES}, Act, \rightarrow_\varepsilon, \Phi(\langle decl, B\rangle))$. Hence, the statement follows by the transitivity of the bisimilarity.

4 Modified operational semantics

The presented operational semantics produces unnecessary long terms, since we have to copy each refinement even though it is only used once. Consider for example the process $(a.B')[a \rightarrow a_1.a_2 \cdots a_n.\mathbf{1}]$. Then we have $(a.B')[a \rightarrow a_1.a_2.\cdots a_n.\mathbf{1}] \xrightarrow{a_1} \cdots \xrightarrow{a_i} (a^{(i)}.B')[(a \rightarrow a_1.a_2.\cdots a_n.\mathbf{1}), (a^{(1)} \rightarrow a_2.\cdots a_n.\mathbf{1}), \cdots, (a^{(i)} \rightarrow a_{i+1}.\cdots a_n.\mathbf{1})]$ for $i \leqslant n$ and suitable actions $a^{(j)}$. Here, the refinement of $a^{(j)} \rightarrow a_{j+1}.\cdots a_n.\mathbf{1}$ is unnecessary information for all $j < i$, since $a^{(j)}$ dose not occur in $(a^{(i)}.B')$.

The fact described above does not only produce unnecessary long terms, it also produces infinite transition systems in cases where finite ones will be sufficient. For example, let $decl(X) = a.X$ then $X[a \rightarrow b] \left(\xrightarrow{b} \xrightarrow{\tau}\right)^i X[a \rightarrow b, a^{(1)} \rightarrow \mathbf{1}, \cdots, a^{(i)} \rightarrow \mathbf{1}]$. Therefore we get an infinite transition system. But it is sufficient that $X[a \rightarrow b]$ evolves to $X[a \rightarrow b]$ by performing $(b\tau)^i$, since $X[a \rightarrow b]$ contains all necessary information.

To circumvent the unnecessary copy of the refinement, we divide Obs into two parts. One part Obs_a is used for the active actions, i.e. actions which have been renamed. The other part Obs_P is used for the action in the original expression, i.e. the expression before the transition rules were applied. By this approach we know that an executable action from Obs_a only appears 'once' in the process. More precisely, suppose B only contains actions from Obs_P and $B \xrightarrow{\gamma_1} \cdots \xrightarrow{\gamma_i} B' \xrightarrow{(b, c)} B''$ and $b \in Obs_a$ and $c \notin \mathscr{A}(\langle decl, B'\rangle)$ then $B'[b \rightarrow \tilde{B}] \sim B''[c \rightarrow \tilde{B}]$. This gives us the advantage of keeping actions from Obs_a unrenamed.

Furthermore, when an action from Obs_a terminates, we remove it in the expressions from those positions where they were inserted by rules P_3 and Ref_2. This has the advantage of generating more finite state transition systems. This can also be satisfied when no α-conversion is considered, for example by choosing always the 'smallest' action from Obs_a which does not occur for action renaming.

The ideas mentioned above are formalized as follows. Let Obs_P, $Obs_a \subseteq Obs$ with $|Obs_P| = |Obs_a| = |\mathbb{N}|$ and $Obs_P \cap Obs_a = \emptyset$. Then the modified transition rules are those of Tab. 1 where A_2, P_2, Ref_2 and Ref_3 are replaced by the transition rules presented in Tab. 2.

In rule A_2^m it is only possible to start (renaming) actions from Obs_P, since actions from Obs_a are considered to be active and therefore they cannot be started again. Furthermore, the action a in rule A_2^m can only be replaced by an action from Obs_a, since a becomes active.

Rule P_2 is modified in the way that we remove action γ from the synchronization set if it is an active action.

This is sound, since γ terminates in this rule and therefore it does not appear in $B'_1 \|'_A B'_2$ when it is from Obs_a.

Tab. 2　Modified transition rules

Let γ be an element of $Act \cup (Obs \times Obs)$

$$A_2^m: \frac{a \in Obs_P \quad b \in Obs_a}{a.B \xrightarrow{(a,b)} b.B}$$

$$P_2^m: \frac{B_1 \xrightarrow{\gamma} B'_1 \quad B_2 \xrightarrow{\gamma} B'_2 \quad \gamma \notin \{\surd\} \cup A \quad A' = A \setminus (\{\gamma\} \cap Obs_a)}{B_1 \|_A B_2 \xrightarrow{\gamma} B'_1 \|_{A'} B'_2}$$

$$Ref_{2.1}^m: \frac{B \xrightarrow{(\hat{a},b)} B' \quad \hat{a} \in A \cap Obs_P \quad b \notin A \cup \mathscr{A}(\langle decl, B\rangle) \quad B_{\hat{a}} \xrightarrow{\gamma} B'' \quad \gamma \neq \surd}{B[(a \to B_a)^{a \in A}] \xrightarrow{\gamma} B'[(a \to B_a)^{a \in A}, (b \to B'')]}$$

$$Ref_{2.2}^m: \frac{B \xrightarrow{\hat{a}} B' \quad \hat{a} \in A \cap Obs_a \quad B_{\hat{a}} \xrightarrow{\gamma} B'' \quad \gamma \neq \surd}{B[(a \to B_a)^{a \in A}] \xrightarrow{\gamma} B[(a \to B_a)^{a \in A \setminus \{\hat{a}\}}, (\hat{a} \to B'')]}$$

$$Ref_3^m: \frac{B \xrightarrow{\gamma} B' \quad \hat{a} \in A \quad B_{\hat{a}} \xrightarrow{\surd} B'' \quad A' = A \setminus (\{\hat{a}\} \cap Obs_a)}{B[(a \to B_a)^{a \in A}] \xrightarrow{\tau} B'[(a \to B_a)^{a \in A'}]}$$

The rule Ref_2 is split into two rules. Rule $Ref_{2.1}^m$ considers the case when an action from Obs_P is refined. In this case the rule stays the same. And rule $Ref_{2.2}^m$ considers the case when an active action is refined. Here we do not rename the action nor do we change the expression which gets refined at all, i.e. keeping B. We only check whether $\hat{a}$ is an intial action, which is done by $B \xrightarrow{\hat{a}} B'$. This is necessary for the soundness, since we have an interrupt operator, and so not every started action which has not terminated has to be active. The change in the refinement in rule $Ref_{2.2}^m$ is directly done at $\hat{a}$, i.e. $\hat{a}$ is refined by B''.

In rule Ref_3^m we remove, similar to rule P_2^m, action γ from the expression if it is an active action.

Theorem 20　Suppose $\langle decl, B\rangle \in \mathbf{PA}$ and $\mathscr{A}(\langle decl, B\rangle) \subseteq Obs_P$. Then $\mathscr{O}^m(\langle decl, B\rangle)$ and $(\mathbf{CEBES}, Act, \to_\varepsilon, \Phi(\langle decl, B\rangle))$ are bisimilar, where $\mathscr{O}^m$ is derived as $\mathscr{O}$ except that the modified transition rules are used.

Proof　similar to the proof of Theorem 19.

5　Related work

The idea to adjust the operational semantics of action refinement such that it coincides with the denotational semantics has been studied previously for languages without interrupt by Ref. [14, 2, 15]. Operational semantics for the ST-approaches, which can be adjusted to operational semantics for actions refinement, are examined in Ref. [16, 17, 2]. These papers concentrate on the operational and axiomatic levels rather than obtaining a direct connection between the denotational and the operational semantics.

The advantages of our approach are:

—We can handle an interrupt operator. This is not possible in the approaches of Ref. [14—17, 2], which is explained in the following.

In Ref. [16, 17, 14], a refinement expression $B[a \to Q]$ is modelled by invoking Q in parallel with the remaining process whenever a is activated in B. This leads to contradictions if an interrupt operator is involved. For example, the action a can be interrupted during its execution. But by the above approach the process to which action a refined is executed, even though a is interrupted. The execution of an action in Ref. [2, 15] is only allowed if all active actions are still executable afterwards. This is not a reasonable approach if a language with an

interrupt operator is considered, since this operator can remove active actions in a feasible way.

—The idea of our approach is easy to understand and the transition rules are straightforward. We do not need equivalence notions on the syntactical level as Ref. [14], where the activating causes have to distribute over the operators.

—We do not have to restrict to guarded expressions, i. e. the consistence of the operational and the denotational semantics is also proved for unguarded expressions.

Another interesting fact is that we derive the transition system from **CEBES** in terms of $b.e.s$. This is in contrast to Ref. [2, 15], where the transition system is derived from configurations. It is not excluded that this leads to different theories when transition systems corresponding to coarsest equivalences are considered, compared with Ref. [18].

6 Conclusion

We presented an operational semantics for action refinement which corresponds to the well-established action refinement in the denotational semantics. This operational semantics does not need an extension of the syntax of the expressions and it is easy to comprehend. Furthermore, it is reasonable for process algebras with an interrupt operator.

In our theory forgetful refinements, i.e. refining an action by 0, do not cause any problems. An interpretation of such a refinement is that the environment can not provide the refined action as well as all actions that causally depend on it.

The hiding operator $\backslash A$ is redundant, since it can be modelled by refinements to 1. Furthermore, the sequential operator can be modelled by the action prefix, i.e. $B_1; B_2$ can be modelled by $(a.B_2)[a \rightarrow B_1]$ where $a \notin \mathscr{A}(\langle decl, B_2 \rangle)$.

The operators on **CEBES** presented here differ slightly from the original ones[9]: we have to define disjoint union of the set of events explicitly, since we cannot guarantee the disjointness of them as by an event-based process algebra approach. Furthermore, we extend the conflict relation in $\hat{+}$, $\hat{.}$ and in $\hat{\rhd}$ to get a closer correspondence between the operational and the denotational semantics. For this reason we also give a definition of the remainder (Definition 16) which differs from Ref. [9].

References

[1] Robin Milner. Communication and concurrency. International series in Computer Science. Prentice Hall, 1989

[2] Roberto Gorrieri, Arend Rensink. Action refinement. In: J A Bergstra, A Ponse, S A Smolka, editors. Handbook of Process Algebra. North-Holland, 2001. 1047—1147

[3] Philippe Darondeau, Pierpaolo Degano. Refinement of actions in event structures and causal trees. *Theoretical Computer Science*, 1993, 118: 21—48

[4] Rob van Glabbeek, Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 2001, 37: 221—327

[5] Walter Vogler. Failures semantics based on interval semiwords is a congruence for refinement. *Distributed Computing*, 1991, 4: 139—162

[6] L Aceto, M Hennessy. Adding action refinement to a finite process algebra. *Information and Computation*, 1994, 115: 179—247

[7] Ursula Goltz, Roberto Gorrieri, Arend Rensink. Comparing syntactic and semantic action refinement. *Information and Computation*, 1996, 125: 118—143

[8] Tommaso Bolognesi, Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*. 1987, 14: 25—49

[9] Rom Langerak. Transformations and Semantics for LOTOS: [PhD thesis]. Department of Computer Science, University of Twente. 1992

[10] Smson Abramsky, Achim Jung. Domain theory. In: Samson Abramsky, Dov M Gabbay, T S E Maibaum, editors. *Handbook of Logic in Computer Science*, Oxford: Clarendon Press, 1994, Vol. 3: 1—168

[11] Harald Fecher, Mila Majster-Cederbaum, Jinzhao Wu. Bundle event structures: a revised cpo approach. *Information Processing Letters*, 2002, 83: 7—12

[12] Mila Majster-Cederbaum, Jinzhao Wu. Action refinement for ture concurrent real-time. In: Proc. 7th IEEE int. Conf. on Engineering of Complex Computer Systems. IEEE Computer Society press, 2001. 58—68

[13] Mila Majster-Cederbaum, J Wu. Towards action refinement for ture concurrent real time. *Acta Informatica*, 2003, 39: 1—47

[14] Pierpaolo Degano, Roberto Gorrieri. A causal operational semantics of action refinement. *Information and Computation*. 1995, 122: 97—119

[15] Arend Rensink. An event-based SOS for a language with refinement. In: Structures in Concurrency Theory. Workshops in Computing, 1995. 294—309

[16] Mario Bravetti, Roberto Gorrieri. Axiomatizing ST bisimulation for a process algebra with recursion and action refinement (extended abstract). In: Electronic Notes in Theoretical Computer Science. Elsevier Science Publishers, 1999. 27

[17] Nadia Busi, Rob van Glabbeek, Roberto Gorrieri. Axiomatising ST bisimulation equivalence. In: E R Olderog, editor. Proceedings IFIP Working Conference on Programming Concepts, Methods and Calculi. Elsevier Science, 1994. 169—188

[18] Mila Majster-Cederbaum, Markus Roggenbach. Transition systems from event structures revisited. *Information Processing Letters*. 1998, 67: 119—124

具有中断算子进程的语义动作精化的操作语义

袁　红　吴尽昭

(中国科学院成都计算机应用研究所, 成都 610041)

摘　要　以SOS 规则的方式定义了含中断和精化算子的进程代数 LOTOS 的操作语义, 使得这一操作语义与指称语义相对应, 即: 由指称语义导出的传输系统和操作语义定义的传输系统双模拟. 操作语义的基本思想是: 重新命名被精化的动作并且使所有与之相关的选择都被激发.

关键词　动作精化, 指称语义, 操作语义, 传输系统