PAPER *Special Section on Selected Papers from the 12th Workshop on Circuit and Systems in Karuizawa*

Least Fixpoint and Greatest Fixpoint in a Process Algebra with Conjunction and Disjunction

Yoshinao ISOBE[†], **Yutaka SATO**[†], *and* **Kazuhito OHMAKI**[†], *Members*

SUMMARY We have already proposed a process algebra μLOTOS as a mathematical framework to synthesize a process from a number of (incomplete) specifications, in which requirements for the process do not have to be completely determined. It is guaranteed that the synthesized process satisfies all the given specifications, if they are consistent. For example, μLOTOS is useful for incremental design. The advantage of μLOTOS is that *liveness properties* can be expressed by *least fixpoints* and *disjunctions* $\vee$. In this paper, we present $\mu\text{LOTOS}^{\text{R}}$, which is a refined μLOTOS. The improvement is that $\mu\text{LOTOS}^{\text{R}}$ has a *conjunction operator* $\wedge$. Therefore, the consistency between a number of specifications $S_1, \cdots, S_2$ can be checked by the satisfiability of the conjunction specification $S_1 \wedge \cdots \wedge S_2$. $\mu\text{LOTOS}^{\text{R}}$ does not need the complex consistency check used in μLOTOS.

key words: process algebra, process logic, process synthesis, least fixpoint, greatest fixpoint, disjunction, conjunction

1. Introduction

The design of concurrent processes is known to be a complex task. In order to support the design, formal description techniques (FDTs) are used for verifying that a realized process conforms to its specification. *Process algebras* such as CCS[12], CSP[4], and LOTOS[18] are examples of FDTs to describe concurrent processes.

In practice, a number of (incomplete) specifications are often given to a process instead of its complete specification in the first design step, because requirements for the process have not been always completely determined yet. Such specifications which contain incomplete requirements can be formally described by *process logics* such as $\mathcal{PL}$[12] and μ-calculus[16]. In a process logic, incomplete requirements can be expressed by a possibility operator or a disjunction operator. The satisfaction of process algebraic expressions (i.e. processes) for process logical expressions (i.e. specifications) is formally defined, and if a process and a specification are given, then the satisfaction can be verified[1], [16].

However it is still a complex task to design a process, if many incomplete specifications are given. Then, methods to synthesize a process from specifications are necessary. For the synthesis, it is useful to describe processes and specifications in the same language. But incomplete requirements can not be expressed by ordinary process algebras whose semantics is given by a labelled transition system (LTS).

Therefore extended process algebras with process logical properties have been proposed. For example, Larsen proposed an extended LTS[9] which has a *required transition* $\longrightarrow_{\Box}$ and an *allowed transition* $\longrightarrow_{\diamond}$ to express loose specifications, and presented a process algebra called *modal CCS*[10] based on the extended LTS. In modal CCS, possibility and necessity of actions can be expressed, and a conjunction operator $\wedge$ is defined in modal CCS. As the other examples, Steen et al. proposed an extended LTS[14] which has a disjunctive transition $\mapsto$ to express a disjunction operator $\vee$, and we proposed to use a special action[5] for the same purpose, in the same time. And thereafter, Steen proposed a LOTOS-like language PSL[15] based on an extended LTS which has $\mapsto$ and Larsen's $\longrightarrow_{\Box}$ and $\longrightarrow_{\diamond}$. On the other hand, we proposed a language μLOTOS[6] to express least fixpoints and greatest fixpoints. Least fixpoints are necessary for describing eventuality of actions, namely liveness properties.

In [6], we presented a method to transform each specification to a standard formed specification, a method to check the consistency of standard formed specifications, and a method to synthesize a conjunction specification from them. In this paper, we present $\mu\text{LOTOS}^{\text{R}}$, which is a refined μLOTOS. The improvement is that $\mu\text{LOTOS}^{\text{R}}$ has a conjunction operator to express the conjunction specification. $\mu\text{LOTOS}^{\text{R}}$ does not need the complex transformation, the consistency check, and the synthesis used in μLOTOS.

The outline of this paper is as follows. In Section 2, the syntax and the semantics of $\mu\text{LOTOS}^{\text{R}}$ are defined. In Section 3, the satisfaction of a process for a specification is defined. In Section 4, we explain how to express greatest fixpoints and least fixpoints in $\mu\text{LOTOS}^{\text{R}}$. In Section 5, we present an inductive characterization of the satisfiability and a method to synthesize a process from a specification. In Section 6, we state related works, and in Section 7, we conclude this paper. In Appendix, a lemma is given.

2. Definition of specifications

In order to describe specifications which contains incomplete requirements, Steen et al.[14] proposed to extend LOTOS with a disjunction operator $\vee$. This op-

Manuscript received June 28, 1999.
Manuscript revised September 30, 1999.

[†]The authors are with Computer Science Division, Electrotechnical Laboratory, 1-1-4, Umezono, Tsukuba, Ibaraki 305-8568, Japan

erator is similar to a disjunction operator in (process) logic, and if P_1 is a process which satisfies a specification S_1 and P_2 is a process which satisfies a specification S_2, then the specification $S_1 \vee S_2$ can be implemented by either P_1 or P_2, where each process is formally an ordinary LOTOS-expression.

In addition to the disjunction operator, for the purpose to express least fixpoints and greatest fixpoints, we proposed to use two kinds of states[6] : *stable states* and *unstable states*, and to impose a condition that every state must reach either a stable state or a stop state. The stable states and unstable states are described by two state operators $\{\circ, \triangleleft\}$, where $\circ$ and $\triangleleft$ are called *Stabilizer* and *Instabilizer*. μLOTOS extends Steen's LOTOS with the state operators.

For example, the following specification AB is satisfied by processes which iteratively perform the action a or stop after the action b.

$$AB := \circ a; AB \vee \circ b; \mathbf{stop}$$

where ; is a prefix operator, thus $\circ a; AB$ requires processes that they can perform a and thereafter conform to the specification AB. The symbol := is used for defining the left *Constant* AB as the right specification, thus it is a recursive definition, because the right specification can contain the Constant. In this case, the disjunction $\vee$ is recursively resolved. For example, all the following processes satisfy the specification AB.

$$A_\infty := a; A_\infty, \quad AB_n := \underbrace{a; \cdots; a}_{n \text{ times}}; b; \mathbf{stop}$$

In this case, the action b can not be always performed in processes satisfying AB, because A_∞ satisfies AB.

On the other hand, the following specification AB' is satisfied by processes which can perform a zero or more times and *must eventually* stop after b, because the state $(\triangleleft a; AB')$ is unstable.

$$AB' := \triangleleft a; AB' \vee \circ b; \mathbf{stop}$$

For example, the process AB_n satisfies AB', but A_∞ does not satisfy AB'.

In addition, μLOTOS$^{\mathrm{R}}$ has a conjunction operator $\wedge$. Intuitively, if P is a process which satisfies both specifications S_1 and S_2, then the specification $S_1 \wedge S_2$ can be implemented by P. In Subsection 2.1 and Subsection 2.2, the syntax and the semantics of μLOTOS$^{\mathrm{R}}$ are defined, respectively.

2.1 Syntax

We assume that a finite set of *names* $\mathcal{N}$ is given. The set of actions Act is defined as $Act = \mathcal{N} \cup \{i\}$ and $\alpha, \beta, \cdots$ are used to range over Act, where i is the *internal action* ($i \notin \mathcal{N}$). We give a set of *state operators* $Stt = \{\circ, \triangleleft\}$, where $\circ$ is called *Stabilizer* and $\triangleleft$ is called *Instabilizer*. The set Stt is ranged over by $\psi, \varphi, \cdots$.

We also assume that a set of *specification constants* $\mathcal{K}$ (also called *Constants*), a set of *stable specification variables* $\mathcal{X}_\circ$ (also called *Stable Variables*), and a set of *unstable specification variables* $\mathcal{X}_\triangleleft$ (also called *Unstable Variables*), are given. The set $\mathcal{K}$ is ranged over by $A, B, \cdots$, and the set $\mathcal{X}_\circ \cup \mathcal{X}_\triangleleft$ is ranged over by $X, Y, \cdots$.

Then, the syntax of μLOTOS$^{\mathrm{R}}$ is defined.

Definition 2.1: The set Spx of *specification expressions* is the smallest set which includes $\mathcal{K} \cup \mathcal{X}_\circ \cup \mathcal{X}_\triangleleft \cup \{\mathbf{stop}\}$, and contains the following expressions, where M, M_i is already in Spx.

$\psi\alpha; M$	:	Prefix ($\psi \in Stt, \alpha \in Act$)
$M_1 \,[]\, M_2$	:	Choice
$M_1 \,\|[G]\|\, M_2$	:	Parallel composition ($G \subseteq \mathcal{N}$)
$M_1 \wedge M_2$	:	Conjunction
$\bigvee_{i \in I} M_i$	:	Disjunction (I an indexing set)

The set Spx is ranged over by $M, N, \cdots$. ∎

The special specification expression **stop** is satisfied by processes which have no action. The Disjunction $\bigvee_{i \in I} M_i$ is also written $\bigvee\{M : C(M)\}$, where C is a condition such that the set of specification expressions which satisfy C is $\{M_i : i \in I\}$. And, a binary expression $M_1 \vee M_2$ is also used for $\bigvee_{i \in \{1,2\}} M_i$. Another special case is $I = \emptyset$, and $\bigvee_{i \in \emptyset} M_i$ is denoted by **F** which is a specification satisfied by no process as explained in Section 3.

In order to avoid too many parentheses, operators have binding power such that: Prefix>Parallel composition>Choice >Conjunction >Disjunction.

The difference between the Disjunction $\vee$ and the Choice [] is intuitively explained as follows: designers *statically* decide whether $M \vee N$ is implemented by either M or N in specification phase, while users *dynamically* decide whether $M \,[]\, N$ behaves like either M or N at run time. Thus, Disjunctions are used only in specifications, and they do not remain in implemented processes.

The Parallel composition $|[G]|$ with a subset G of $\mathcal{N}$ synchronizes actions contained by G and independently performs the other actions. This can synchronize three or more actions.

The notation $Var(M)$ represents the set of Variables occurring in the specification expression M, and it is inductively defined as follows :

$$\begin{aligned} Var(A) &= \emptyset, \\ Var(X) &= \{X\}, \\ Var(\mathbf{stop}) &= \emptyset, \\ Var(\psi\alpha; M) &= Var(M), \\ Var(M \ op \ N) &= Var(M) \cup Var(N), \\ Var(\textstyle\bigvee_{i \in I} M_i) &= \textstyle\bigcup_{i \in I} Var(M_i) \end{aligned}$$

where op is [] or $|[G]|$ or $\wedge$. Then, a specification expression M is a *specification*, if it contains no Variable ($Var(M) = \emptyset$). The set of specifications is denoted by Sp, and it is ranged over by $S, T, \cdots$. A Constant is a

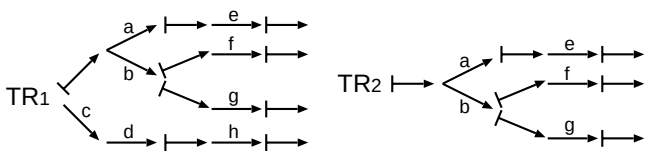

Fig. 1 Transition systems with disjunctive transitions $\mapsto$

Name	Hypothesis	$\vdash$	Conclusion
$\mathbf{Var}_S$	$X \in \mathcal{X}_\circ$	$\vdash$	$X \in Stb$
$\mathbf{Act}_S$		$\vdash$	$\circ\alpha; M \in Stb$
$\mathbf{Stop}_S$		$\vdash$	$\mathbf{stop} \in Stb$
$\mathbf{Ch}_S$	$M \in Stb,\ N \in Stb$	$\vdash$	$M \,[]\, N \in Stb$
$\mathbf{Par}_{S1}$	$M \in Stb$	$\vdash$	$M \,\|[G]\|\, N \in Stb$
$\mathbf{Par}_{S2}$	$N \in Stb$	$\vdash$	$M \,\|[G]\|\, N \in Stb$
$\mathbf{Con}_S$	$M \in Stb$	$\vdash$	$M \wedge N \in Stb$

Fig. 2 The inference rules for the set Stb

specification whose meaning is given by a defining equation. We assume that for every Constant $A \in \mathcal{K}$, there is a defining equation of the form $A := S$, where S is a specification which can contain Constants again. Thus, it is a recursive definition. We assume that every recursion must be guarded by Prefixes, such as $A := \circ a; A$. For example, we do not consider $A := A \,[]\, \circ a; \mathbf{stop}$.

Processes are also described in μLOTOS$^{\mathrm{R}}$. They are μLOTOS$^{\mathrm{R}}$-expressions which neither contain Disjunctions nor Instabilizers nor Variables, and they correspond to ordinary (basic) LOTOS-expressions. Thus, the set of processes Pr is a subset of Sp, and the syntax is defined in terms of the following BNF expression:

$$P ::= A \ \mid\ \mathbf{stop} \ \mid\ \circ\alpha; P \ \mid\ P \,[]\, P \ \mid\ P \,|[G]|\, P$$

where $A \in \mathcal{K}_P \subseteqq \mathcal{K}$, $\alpha \in Act$, and $G \subseteqq \mathcal{N}$. The set Pr is ranged over by $P, Q, \cdots$. We assume that for every $A \in \mathcal{K}_P$, there is a defining equation $A := P$ ($P \in Pr$). *Note that Stabilizers $\circ$ are often omitted.* For example, $\circ\alpha; P$ is written as $\alpha; P$.

2.2 Semantics

We proposed an extended labelled transition system μLTS [6] which distinguishes between stable states and unstable states, to give the semantics of μLOTOS. The semantics of μLOTOS$^{\mathrm{R}}$ is defined by the following μLTS$^{\mathrm{R}}$ which is a refined μLTS.

Definition 2.2: A μLTS$^{\mathrm{R}}$ is a structure

$$\langle St_1, St_2, St_3, Lb, \mapsto, \rightarrow \rangle,$$

where St_1 is a set of *states*, St_2 is a set of *intermediate states* ($St_1 \cap St_2 = \emptyset$), St_3 is a set of *stable intermediate states* ($St_3 \subseteqq St_2$), Lb is a set of labels, $\mapsto \subseteqq St_1 \times St_2$ is a set of *disjunctive transitions*, $\rightarrow \subseteqq St_2 \times Lb \times St_1$ is a set of *labelled transitions*. We often write $s_1 \mapsto s_2$ for $(s_1, s_2) \in \mapsto$ and $s_2 \xrightarrow{\alpha} s_1$ for $(s_2, \alpha, s_1) \in \rightarrow$. ∎

The difference between μLTS$^{\mathrm{R}}$ and μLTS is that μLTS$^{\mathrm{R}}$ separates intermediate states St_2 from states St_1. By the separation, disjunctive transitions and labelled transitions are always *alternately* performed, and the semantics can be more explicitly defined. For example, the transition system TR_1 in Fig.1 is equated to TR_2 by μLTS (and also by the other LTSs with disjunctive transitions such as ALTS[14] and DMLTS[15]). This example shows that disjunctive transitions and labelled transitions should not arise from the same state. This is the reason why μLTS$^{\mathrm{R}}$ distinguishes between intermediate states for labelled transitions and states for disjunctive transitions.

The meaning of disjunctive transitions and labelled transitions is intuitively explained as follows: *one* of disjunctive transitions for each state must be performed, and *all* of labelled transitions for each intermediate state must be performed. The stability is checked in intermediate states ($St_3 \subseteqq St_2$).

The semantics of μLOTOS$^{\mathrm{R}}$ is given by the μLTS$^{\mathrm{R}}$ $\langle Spx, \lceil Spx \rfloor, \lceil Stb \rfloor, Act, \mapsto, \rightarrow \rangle$, where the function $\lceil \cdot \rfloor : (set \rightarrow set)$ is defined for any set Set as

$$\lceil Set \rfloor = \{\lceil e \rfloor : e \in Set\},$$

Stb is defined in Definition 2.3, $\mapsto$ is defined in Definition 2.4, and $\rightarrow$ is defined in Definition 2.5. The function $\lceil \cdot \rfloor$ is used to separate intermediate states from states. In this paper, *we consider only specification expressions with finite states.* For example, $A := \circ a; A$ has finite states, but $A_i := \circ a; A_{i+1}$ has infinite states.

Definition 2.3: The set of stable specification expressions $Stb \subseteqq Spx$ is the smallest set satisfying the inference rules in Fig.2. ∎

Definition 2.4: The disjunctive transition relation $\mapsto \subseteqq Spx \times \lceil Spx \rfloor$ is the smallest relation satisfying the inference rules in Fig.3. ∎

Definition 2.5: The labelled transition relation $\rightarrow \subseteqq \lceil Spx \rfloor \times Act \times Spx$ is the smallest relation satisfying the inference rules in Fig.4, where the relation $\equiv$ represents syntactic identity. ∎

Fig.2 shows that unstable states are described by using Instabilizers, because $\triangleleft\alpha; M \notin Stb$. For example, $\triangleleft a; \mathbf{stop} \,[]\, \circ b; \mathbf{stop}$ is unstable. It is important to note that the stability of $M \wedge N$ depends *only* on M by $\mathbf{Con}_S$. This *asymmetry* is the key idea of μLOTOS$^{\mathrm{R}}$. The stability of N is considered after M reaches a stable state, as shown in the rules $\mathbf{Con}_{1,2}$ in Fig.4. The rules show that if M is stable then the order of M' and N' is exchanged after a labelled transition. Therefore, the stability of M and N in $M \wedge N$ is *alternately* considered.

The Disjunction $\bigvee\{N'' : \lceil N \rfloor \xrightarrow{\alpha} N''\}$ of $\mathbf{Con}_1$ is also important. The rule $\mathbf{Con}_1$ intuitively requires that if $\lceil M \rfloor \xrightarrow{\alpha} M'$, then for *some* N' such that $\lceil N \rfloor \xrightarrow{\alpha} N'$, $N' \wedge M'$ is satisfied after the action α. It is helpful to compare $\mathbf{Con}_1$ with the following simple rule $\mathbf{Con}'_1$:

Name	Hypothesis	$\vdash$	Conclusion
$\mathbf{Var}_\vee$		$\vdash$	$X \mapsto \lceil X \rfloor$
$\mathbf{Stop}_\vee$		$\vdash$	$\mathbf{stop} \mapsto \lceil \mathbf{stop} \rfloor$
$\mathbf{Act}_\vee$		$\vdash$	$\psi\alpha; M \mapsto \lceil \psi\alpha; M \rfloor$
$\mathbf{Rec}_\vee$	$S \mapsto \lceil S' \rfloor,\ A := S$	$\vdash$	$A \mapsto \lceil S' \rfloor$
$\mathbf{Ch}_\vee$	$M \mapsto \lceil M' \rfloor,\ N \mapsto \lceil N' \rfloor$	$\vdash$	$M \,[]\, N \mapsto \lceil M' \,[]\, N' \rfloor$
$\mathbf{Par}_\vee$	$M \mapsto \lceil M' \rfloor,\ N \mapsto \lceil N' \rfloor$	$\vdash$	$M \,\|[G]\|\, N \mapsto \lceil M' \,\|[G]\|\, N' \rfloor$
$\mathbf{Con}_\vee$	$M \mapsto \lceil M' \rfloor,\ N \mapsto \lceil N' \rfloor$	$\vdash$	$M \wedge N \mapsto \lceil M' \wedge N' \rfloor$
$\mathbf{Dis}_\vee$	$M_i \mapsto \lceil M' \rfloor,\ i \in I$	$\vdash$	$\bigvee_{i\in I} M_i \mapsto \lceil M' \rfloor$

Fig. 3 The inference rules for the disjunctive transition relation $\mapsto$

Name	Hypothesis	$\vdash$	Conclusion
Act		$\vdash$	$\lceil \psi\alpha; M \rfloor \xrightarrow{\alpha} M$
$\mathbf{Ch}_1$	$\lceil M \rfloor \xrightarrow{\alpha} M'$	$\vdash$	$\lceil M \,[]\, N \rfloor \xrightarrow{\alpha} M'$
$\mathbf{Ch}_2$	$\lceil N \rfloor \xrightarrow{\alpha} N'$	$\vdash$	$\lceil M \,[]\, N \rfloor \xrightarrow{\alpha} N'$
$\mathbf{Par}_1$	$\lceil M \rfloor \xrightarrow{\alpha} M',\ \alpha \notin G$	$\vdash$	$\lceil M \,\|[G]\|\, N \rfloor \xrightarrow{\alpha} M' \,\|[G]\|\, N$
$\mathbf{Par}_2$	$\lceil N \rfloor \xrightarrow{\alpha} N',\ \alpha \notin G$	$\vdash$	$\lceil M \,\|[G]\|\, N \rfloor \xrightarrow{\alpha} M \,\|[G]\|\, N'$
$\mathbf{Par}_3$	$\lceil M \rfloor \xrightarrow{\alpha} M',\ \lceil N \rfloor \xrightarrow{\alpha} N',\ \alpha \in G$	$\vdash$	$\lceil M \,\|[G]\|\, N \rfloor \xrightarrow{\alpha} M' \,\|[G]\|\, N'$
$\mathbf{Con}_1$	$\lceil M \rfloor \xrightarrow{\alpha} M',\ N' \equiv \bigvee\{N'' : \lceil N \rfloor \xrightarrow{\alpha} N''\},\ M \in Stb$	$\vdash$	$\lceil M \wedge N \rfloor \xrightarrow{\alpha} N' \wedge M'$
$\mathbf{Con}_2$	$\lceil N \rfloor \xrightarrow{\alpha} N',\ M' \equiv \bigvee\{M'' : \lceil M \rfloor \xrightarrow{\alpha} M''\},\ M \in Stb$	$\vdash$	$\lceil M \wedge N \rfloor \xrightarrow{\alpha} N' \wedge M'$
$\mathbf{Con}_3$	$\lceil M \rfloor \xrightarrow{\alpha} M',\ N' \equiv \bigvee\{N'' : \lceil N \rfloor \xrightarrow{\alpha} N''\},\ M \notin Stb$	$\vdash$	$\lceil M \wedge N \rfloor \xrightarrow{\alpha} M' \wedge N'$
$\mathbf{Con}_4$	$\lceil N \rfloor \xrightarrow{\alpha} N',\ M' \equiv \bigvee\{M'' : \lceil M \rfloor \xrightarrow{\alpha} M''\},\ M \notin Stb$	$\vdash$	$\lceil M \wedge N \rfloor \xrightarrow{\alpha} M' \wedge N'$

Fig. 4 The inference rules for the labelled transition relation $\rightarrow$

$$\mathbf{Con}'_1 \quad \lceil M \rfloor \xrightarrow{\alpha} M',\ \lceil N \rfloor \xrightarrow{\alpha} N',\ M \in Stb \vdash \lceil M \wedge N \rfloor \xrightarrow{\alpha} N' \wedge M'$$

This rule $\mathbf{Con}'_1$ intuitively requires that if $\lceil M \rfloor \xrightarrow{\alpha} M'$, then for *every* N' such that $\lceil N \rfloor \xrightarrow{\alpha} N'$, $N' \wedge M'$ is satisfied after the action α. This requirement fails to define conjunction, when either N' such that $\lceil N \rfloor \xrightarrow{\alpha} N'$ does not exist or such N' is not uniquely determined.

Fig.3 shows that Disjunctions are resolved by $\mathbf{Dis}_\vee$, and Constants are unwound by $\mathbf{Rec}_\vee$. The other operators are preserved by disjunctive transitions. For example, the rules $\mathbf{Act}_\vee$, $\mathbf{Con}_\vee$, $\mathbf{Dis}_\vee$, and $\mathbf{Rec}_\vee$ infer the disjunctive transition $A \mapsto \lceil \circ a; A \wedge \triangleleft b; \mathbf{stop} \rfloor$, where $A := (\circ a; A \wedge \triangleleft b; \mathbf{stop}) \vee \circ c; A \vee \triangleleft d; \mathbf{stop}$. Since processes Pr have no Disjunction and no Conjunction, every process has only one disjunctive transition.

3. Satisfaction

In this section, we define a satisfaction $P \models S$ of a process P for a specification S as an extension of the satisfaction $P \models_{[14]} S$ in [14]. The satisfaction $\models_{[14]}$ has been defined as follows[†]: the satisfaction $\models_{[14]}$ is the largest relation such that, $P \models_{[14]} S$ implies that for some S', $S \mapsto \lceil S' \rfloor$ and for every $\alpha \in Act$,

$(i.[14])$ if $P \mapsto \xrightarrow{\alpha} P'$ then, for some S'', $\lceil S' \rfloor \xrightarrow{\alpha} S''$ and $P' \models_{[14]} S''$,

$(ii.[14])$ if $\lceil S' \rfloor \xrightarrow{\alpha} S''$ then, for some P', $P \mapsto \xrightarrow{\alpha} P'$ and $P' \models_{[14]} S''$.

[†]This is slightly changed from [14], because our definition of $\mapsto$ is different from [14]. But the essence remains.

This requires that there *exists* an S' such that $S \mapsto \lceil S' \rfloor$ and the pair (P, S') satisfies $(i.[14])$ and $(ii.[14])$. This makes it possible that a specification can be satisfied by two or more processes which may be different from each other.

In the definition of $\models_{[14]}$, the specification S' can be freely selected from $\{S' : S \mapsto \lceil S' \rfloor\}$. On the other hand, we can control the selection by state operators $\circ$ and $\triangleleft$. The key point is that S must eventually reach either a stable state or a stop state. Then, our satisfaction is defined as follows.

Definition 3.1: A relation $\mathcal{R} \subseteqq Pr \times Sp$ is a *satisfaction relation*, if $\mathcal{R} \subseteqq \theta(\mathcal{R})$, where $\theta(\mathcal{R}) \subseteqq Pr \times Sp$ is inductively defined for any relation $\mathcal{R}$, as follows:

1. $(P, S) \in \theta^{(0)}(\mathcal{R})$ iff for some S', $S \mapsto \lceil S' \rfloor \in \lceil Stb \rfloor$ and for every $\alpha \in Act$,

   (i_0) if $P \mapsto \xrightarrow{\alpha} P'$ then, for some S'', $\lceil S' \rfloor \xrightarrow{\alpha} S''$ and $(P', S'') \in \mathcal{R}$,

   (ii_0) if $\lceil S' \rfloor \xrightarrow{\alpha} S''$ then, for some P', $P \mapsto \xrightarrow{\alpha} P'$ and $(P', S'') \in \mathcal{R}$,

2. $(P, S) \in \theta^{(n+1)}(\mathcal{R})$ iff for some S', $S \mapsto \lceil S' \rfloor \notin \lceil Stb \rfloor$ and for every $\alpha \in Act$,

   (i_{n+1}) if $P \mapsto \xrightarrow{\alpha} P'$ then, for some (m, S''), $\lceil S' \rfloor \xrightarrow{\alpha} S''$, $(P', S'') \in \theta^{(m)}(\mathcal{R})$, $m \leqq n$,

   (ii_{n+1}) if $\lceil S' \rfloor \xrightarrow{\alpha} S''$ then, for some (m, P'), $P \mapsto \xrightarrow{\alpha} P'$, $(P', S'') \in \theta^{(m)}(\mathcal{R})$, $m \leqq n$,

3. $(P,S) \in \theta(\mathcal{R})$ iff $(P,S) \in \theta^{(n)}(\mathcal{R})$, for some n.

∎

Definition 3.2: P *satisfies* S, written $P \models S$, if $(P,S) \in \mathcal{R}$, for some satisfaction relation $\mathcal{R}$. (i.e. $\models = \bigcup\{\mathcal{R} : \mathcal{R}$ is a satisfaction relation$\}$). We use the notation $Proc(S)$ for the set of all the processes which satisfy the specification S (i.e. $Proc(S) = \{P : P \models S\}$). ∎

This satisfaction is expected to be automatically checked by a similar algorithm to one for bisimilarity [7]. Furthermore, it is useful to use efficient verification techniques such as compositional verification [17], ρ-bisimulation [1], and so on.

By Definition 3.1, if a specification S has no disjunctive transition $\mapsto$, then S is satisfied by no process. It is important to note that $\mathbf{F} \equiv \bigvee_{i\in\emptyset} M_i$ has no disjunctive transition by the inference rule $\mathbf{Dis}_\vee$ in Fig.3. This implies that $S \,[]\, \mathbf{F}$ and $S \wedge \mathbf{F}$ have no disjunctive transition. On the other hand, a Constant $\mathbf{T}$ which is satisfied by all the processes is defined as follows.

$$\mathbf{T} := \bigvee\{\sum\{\circ\alpha; \mathbf{T} \,:\, \alpha \in \mathcal{A}\} \,:\, \mathcal{A} \subseteqq Act\}$$

where if $n \geqq 1$, then $\sum\{M_1, \cdots, M_n\} \equiv M_1 \,[]\, \cdots \,[]\, M_n$, otherwise $\sum\{\,\} \equiv \mathbf{stop}$.

Example 3.1: Consider the following process PR and the specification SAB.

$$PR := a; a; b; PR\,, \quad SAB := \triangleleft a; SAB \vee \circ b; SAB$$

In the specification SAB, only $(\circ b; SAB)$ is stable. Thus, SAB requires that the action b must be always eventually performed, although the action a may be performed zero or more times before b. In this case, we can show that $PR \models SAB$, because the set

$$\mathcal{R} = \{(PR, SAB), (a; b; PR, SAB), (b; PR, SAB)\}$$

is a satisfaction relation, because

$$\begin{array}{c}(PR, SAB) \in \theta^{(2)}(\mathcal{R}),\\ (a; b; PR, SAB) \in \theta^{(1)}(\mathcal{R}), \text{ and}\\ (b; PR, SAB) \in \theta^{(0)}(\mathcal{R}).\end{array}$$

∎

The satisfaction $\models$ is preserved by process operators: Prefix, Choice, Parallel composition, as shown in the following proposition.

Proposition 3.1: Let $\psi \in Stt$, $\alpha \in Act$, $G \subseteqq \mathcal{N}$, and for each $i \in \{1,2\}$, $P_i \in Pr$, $S_i \in Sp$, $P_i \models S_i$. Then,

1. $\alpha; P_1 \models \psi\alpha; S_1\,,$
2. $P_1 \,[]\, P_2 \models S_1 \,[]\, S_2\,,$
3. $P_1 \,|[G]|\, P_2 \models S_1 \,|[G]|\, S_2\,.$

Proof The proofs for 1 and 2 are omitted, because they are easier than the proof for 3. We show that the following $\mathcal{R}$ is a satisfaction relation for 3.

$$\mathcal{R} = \{(P_1 \,|[G]|\, P_2, S_1 \,|[G]|\, S_2) : P_1 \models S_1, P_2 \models S_2\}$$

Let $(P_1 \,|[G]|\, P_2, S_1 \,|[G]|\, S_2) \in \mathcal{R}$, thus $P_1 \models S_1$ and $P_2 \models S_2$. For each $i \in \{1,2\}$, since $P_i \models S_i$, for some n_i, $(P_i, S_i) \in \theta^{(n_i)}(\models)$. Then, we can derive $(P_1 \,|[G]|\, P_2, S_1 \,|[G]|\, S_2) \in \theta(\mathcal{R})$ by induction on $n_1 + n_2$. This proof is easier than Lemma Appendix A.1, and the proof technique of the lemma is also useful here. ∎

The next proposition shows that logical operators have expected properties. Here, it is important that the property of the Conjunction is symmetric, although the definition ($\mathbf{Con}_{S,1,2,3,4}$) is asymmetric.

Proposition 3.2: Let $P \in Pr$, $S_i \in Sp$. Then,

1. $P \models \mathbf{T}$ and $P \not\models \mathbf{F}$
2. $P \models S_1 \wedge S_2 \iff P \models S_1$ and $P \models S_2$
3. $P \models S_1 \vee S_2 \iff P \models S_1$ or $P \models S_2$

Proof The proofs for 1 and 3 are omitted, because they are easier than the proof for 2. The case $(\Rightarrow)$ for 2 : We show that the following $\mathcal{R}$ is a satisfaction relation:

$$\begin{array}{rl}\mathcal{R} = \{(P, S_1) : \exists S_2, & P \models S_1 \wedge S_2\} \cup\\ \{(P, S_1) : \exists S_2, & P \models S_2 \wedge S_1\}\end{array}$$

Let $(P, S_1) \in \mathcal{R}$, thus for some S_2, $P \models T$, where either $T \equiv S_1 \wedge S_2$ or $T \equiv S_2 \wedge S_1$. By the definition of $\models$, for some n, $(P, T) \in \theta^{(n)}(\models)$. If $T \equiv S_1 \wedge S_2$, then we have that $(P, S_1) \in \theta(\mathcal{R})$ by Lemma Appendix A.1.

Let $T \equiv S_2 \wedge S_1$. In this case, T eventually reaches $T' \equiv S_2' \wedge S_1'$ which is a stable state or a stop state, because T is satisfiable. If T' is a stable state, then the order of S_2' and S_1' is exchanged by $\mathbf{Con}_{1,2}$, then Lemma Appendix A.1 is also useful for this case. Otherwise (i.e. T' is a stop state) S_1' must be also a stop state. Consequently, we can obtain that $(P, S_1) \in \theta(\mathcal{R})$.

The case $(\Leftarrow)$ for 2 : We show that the following $\mathcal{R}$ is a satisfaction relation:

$$\mathcal{R} = \{(P, S_1 \wedge S_2) : P \models S_1, P \models S_2\}$$

This proof is similar to the case $(\Rightarrow)$ above. ∎

Proposition 3.2 is applied to the following example.

Example 3.2: Consider the following two symmetrical specifications SAB and SBA.

$$\begin{array}{l}SAB := \triangleleft a; SAB \vee \circ b; SAB,\\ SBA := \circ a; SBA \vee \triangleleft b; SBA\end{array}$$

The specification SAB have been used in Example 3.1, and SBA requires that the action a must be always eventually performed, although the action b may be performed zero or more times before a. Then, the process PR of Example 3.1 also satisfies SBA. Thus, by Proposition 3.2, PR satisfies $SAB \wedge SBA$.

The transition graph of $SAB \wedge SBA$ is shown in Fig.5. In this graph, the transitions which go out from the dotted box can be ignored, because $\mathbf{F}$ can not be satisfied. In Fig.5, each circled state represents a stable state. This graph shows that the eventuality of a and b is alternately considered. ∎

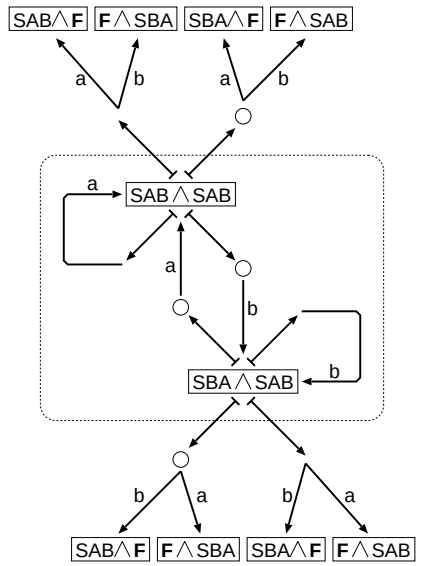

Fig. 5 The transition graph of $SAB \wedge SBA$ ($\bigcirc$: a stable state)

Next, a partial order $\sqsubseteq$ and an equivalence relation $\cong$ over specifications are defined. The order $S \sqsubseteq T$ means that if a process satisfies S, then it satisfies T, thus S is a *refined* specification from T.

Definition 3.3: Let $S, T \in Sp$.

1. $S \sqsubseteq T \iff Proc(S) \subseteqq Proc(T)$.
2. $S \cong T \iff Proc(S) = Proc(T)$.

The following proposition shows that $\sqsubseteq$ is preserved by sequential operators : Prefix, Choice, Conjunction, and Disjunction. This implies that $\cong$ is also preserved by the sequential operators, because $S \cong T$ if and only if $S \sqsubseteq T$ and $T \sqsubseteq S$.

Proposition 3.3: Let $S_1 \sqsubseteq T_1$ and $S_2 \sqsubseteq T_2$. Then,

1. $\psi\alpha; S_1 \sqsubseteq \psi\alpha; T_1$
2. $S_1 \,[]\, S_2 \sqsubseteq T_1 \,[]\, T_2$
3. $\mathbf{F} \sqsubseteq S_1 \sqsubseteq \mathbf{T}$
4. $S_1 \wedge S_2 \sqsubseteq T_1 \wedge T_2$
5. $S_1 \vee S_2 \sqsubseteq T_1 \vee T_2$

Proof The proof of 1 is easy. For 2, we show that if $P \models S_1 \,[]\, S_2$, then $P \models T_1 \,[]\, T_2$, by using Proposition 3.1. At first, P is divided into P_1 and P_2 such that P and $P_1 \,[]\, P_2$ are strongly bisimilar[12], $P_1 \models S_1$, and $P_2 \models S_2$. The process P_i is defined as follows:

$$P_i \equiv \sum\{\alpha; P' : \exists S'', \lceil S_i' \rfloor \xrightarrow{\alpha} S'', P \mapsto \xrightarrow{\alpha} P', P' \models S''\}$$

where S_i' is a specification such that $S_1 \,[]\, S_2 \mapsto \lceil S_1' \,[]\, S_2' \rfloor$ and $P \models S_1' \,[]\, S_2'$. The details are omitted.

The cases 3, 4, and 5, are easily shown by Proposition 3.2. For example, the proof of 3 is as follows:

$$\begin{aligned} Proc(S_1 \wedge S_2) &= Proc(S_1) \cap Proc(S_2) \\ &\subseteqq Proc(T_1) \cap Proc(T_2) = Proc(T_1 \wedge T_2). \end{aligned}$$

4. Fixpoint

In this section, we give important properties for fixpoints. The greatest fixpoint is the maximum solution of $X \cong M$, and the least fixpoint is the minimum solution of $X \cong M$. They are defined as follows, where the notation $M\{N/X\}$ indicates the substitution of N for every occurrence of the Variable X in M.

Definition 4.1: Let $Var(M) \subseteqq \{X\}$.

1. T is the greatest fixpoint of M with respect to X, $\iff$ for every S such that $S \cong M\{S/X\}$, $S \sqsubseteq T$ and $T \cong M\{T/X\}$.
2. T is the least fixpoint of M with respect to X, $\iff$ for every S such that $S \cong M\{S/X\}$, $T \sqsubseteq S$ and $T \cong M\{T/X\}$.

At first, we define a set $Path_\psi(M) \subseteqq \mathcal{X}_\circ \cup \mathcal{X}_\triangleleft$. Intuitively, if $X \in Path_\circ(M)$ (or $X \in Path_\triangleleft(M)$), then all the states on the paths from M to X is stable (or unstable). For example, if $X \in \mathcal{X}_\circ$ and $Y \in \mathcal{X}_\triangleleft$, then

$$X \in Path_\circ(\circ a; \circ b; X \,[]\, \circ c; \triangleleft d; \mathbf{stop}),$$
$$Y \in Path_\triangleleft(\triangleleft a; \triangleleft b; Y \,[]\, \circ c; \circ d; \mathbf{stop}).$$

Definition 4.2: Let $M \in Spx$.

1. $X \in Path_\circ(M) \iff$ for every M', if $M \rightsquigarrow \lceil M' \rfloor$ and $X \in Var(M')$, then $M' \in Stb$.
2. $X \in Path_\triangleleft(M) \iff$ for every M', if $M \rightsquigarrow \lceil M' \rfloor$ and $X \in Var(M')$, then $M' \notin Stb$.

where $\rightsquigarrow$ is the smallest relation satisfying the following inference rules, thus it gives all the reachable intermediate states.

Hypothesis	$\vdash$	Conclusion
$M \mapsto \lceil M' \rfloor$	$\vdash$	$M \rightsquigarrow \lceil M' \rfloor$
$M \rightsquigarrow \lceil M' \rfloor \xrightarrow{\alpha} M'' \mapsto \lceil M''' \rfloor$	$\vdash$	$M \rightsquigarrow \lceil M''' \rfloor$

Secondly, the definitions of sequential expressions and guarded expressions of CCS[12] are extended for μLOTOS$^{\mathrm{R}}$.

Definition 4.3: Let $X \in \mathcal{X}_\circ \cup \mathcal{X}_\triangleleft$ and $M \in Spx$.

1. X is sequential in $M \iff$ for every subexpression M' of M such that $X \in Var(M')$, M' is of the form either $\psi\alpha; M''$ or $M_1'' \,[]\, M_2''$ or $M_1'' \wedge M_2''$ or $\bigvee_{i \in I} M_i''$ or X.
2. X is guarded in $M \iff$ each occurrence of X is

within some subexpression M' of M such that M' is of the form $\psi\alpha; M''$.

∎

For example, the variable X is sequential and guarded in $(\circ a; X) \vee (\circ c; \mathbf{stop} \,|[\emptyset]|\, \circ d; \mathbf{stop}) \,[]\, (\circ b; X)$.

Then, Proposition 4.1 holds. Intuitively, if $X \in Path_\circ(M)$, then the loop defined by M is infinite, and if $X \in Path_\lhd(M)$, then the loop defined by M is finite.

Proposition 4.1: Assume that X is sequential and guarded in M, $Var(M) \subseteqq \{X\}$, and $A := M\{A/X\}$.

1. Let $X \in Path_\circ(M)$. Then
$$P \models A \iff P \models M^{\langle n \rangle}\{\mathbf{T}/X\} \text{ for any } n \geqq 0.$$
2. Let $X \in Path_\lhd(M)$. Then
$$P \models A \iff P \models M^{\langle n \rangle}\{\mathbf{F}/X\} \text{ for some } n \geqq 0.$$

where $M^{\langle n \rangle}$ is inductively defined from M as follows:

$$M^{\langle 0 \rangle} \equiv X, \quad M^{\langle n+1 \rangle} \equiv M\{M^{\langle n \rangle}/X\}.$$

Proof (outline) 1. ($\Rightarrow$): This is easily shown by Proposition 3.3 and $P \models \mathbf{T}$. ($\Leftarrow$): At first, inductively define $\models_{(n)}$ as follows:

$$\models_{(0)} = Pr \times Sp, \quad \models_{(n+1)} = \theta(\models_{(n)}).$$

Then, $\bigcap_{i \geqq 0} \models_{(i)}$ is a satisfaction relation. Hence, we show $P \models_{(n)} A$ for any n. In fact, we can show that $P \models_{(n)} A$ if $P \models M^{\langle n \rangle}\{\mathbf{T}/X\}$.

2. ($\Leftarrow$): This is easily shown by Proposition 3.3 and $P \not\models \mathbf{F}$. ($\Rightarrow$): It can be shown by induction on n that if $(P, N\{A/X\}) \in \theta^{(n)}(\models)$, then $P \models N\{M^{\langle n+1 \rangle}\{\mathbf{F}/X\}/X\}$, where $Var(N) \subseteqq \{X\}$, X is sequential in N, and $X \in Path_\lhd(N)$. The key points are (1) if $N\{A/X\}$ is stable then $Var(N) = \emptyset$, because $X \in Path_\lhd(N)$, and (2) X is guarded in $N\{M/X\}$ and $Proc(N\{A/X\}) = Proc(N\{M/X\}\{A/X\})$. The initial transitions of $N\{M/X\}\{A/X\}$ do not depend on A, because A is guarded. This proof technique by guarded expressions is used in the proof of unique solution for equality in [12](p.158). ∎

We explain Proposition 4.1 by using the following example.

$$\begin{aligned} M_1 &\equiv \circ a; X_1 \vee \circ b; \mathbf{stop} \quad (X_1 \in \mathcal{X}_\circ) \\ M_2 &\equiv \lhd a; X_2 \vee \circ b; \mathbf{stop} \quad (X_2 \in \mathcal{X}_\lhd) \\ A_i &:= M_i\{A_i/X_i\} \quad \text{for each } i \in \{1, 2\} \\ P &:= a; P \end{aligned}$$

By Definition 4.2, $X_1 \in Path_\circ(M_1)$, $X_2 \in Path_\lhd(M_2)$. In this case, $P \models A_1$, because $P \models M_1^{\langle n \rangle}\{\mathbf{T}/X\}$ for any n. On the other hand, $P \not\models A_2$, because there is no integer n such that $P \models M_2^{\langle n \rangle}\{\mathbf{F}/X\}$. Intuitively, A_2 requires that b must be *eventually* performed. For example, if the process Q is defined as

$$Q := a; a; b; \mathbf{stop},$$

then $Q \models A_2$, because $Q \models M_2^{\langle 3 \rangle}\{\mathbf{F}/X\}$.

Finally, Theorem 4.2 is presented. This theorem shows that fixpoints can be expressed by Constants.

Theorem 4.2: Assume that X is sequential and guarded in M, $Var(M) \subseteqq \{X\}$, and $A := M\{A/X\}$.

1. If $X \in Path_\circ(M)$, then
A is the greatest fixpoint of M with respect to X.
2. If $X \in Path_\lhd(M)$, then
A is the least fixpoint of M with respect to X.

Proof By Proposition 3.3, every sequential specification expression M is monotonic. For example, if $S \sqsubseteq T$ then $M\{S/X\} \sqsubseteq M\{T/X\}$. And by $\mathbf{Rec}_\vee$, it can be easily shown that $A \cong M\{A/X\}$.

1. Assume that $X \in Path_\circ(M)$ and $S \cong M\{S/X\}$. Then, let $P \models S$. Since M is monotonic, for any $n \geqq 0$,

$$S \cong M\{S/X\} \cong \cdots \cong M^{\langle n \rangle}\{S/X\} \sqsubseteq M^{\langle n \rangle}\{\mathbf{T}/X\}.$$

Thus, for any n, $P \models M^{\langle n \rangle}\{\mathbf{T}/X\}$. Finally, by Proposition 4.1, $P \models A$. This means that for every S such that $S \cong M\{S/X\}$, $S \sqsubseteq A$.

2. Assume that $X \in Path_\lhd(M)$ and $S \cong M\{S/X\}$. Then, let $P \models A$. By Proposition 4.1, for some n, $P \models M^{\langle n \rangle}\{\mathbf{F}/X\}$. Since M is monotonic,

$$M^{\langle n \rangle}\{\mathbf{F}/X\} \sqsubseteq M^{\langle n \rangle}\{S/X\} \cong \cdots \cong M\{S/X\} \cong S$$

Hence, $P \models S$. This means that for every S such that $S \cong M\{S/X\}$, $A \sqsubseteq S$. ∎

In the rest of this section, we formally explain the liveness property that the action β must be eventually performed for every execution path. If a process P satisfies the liveness property, then we write $P \overset{\beta}{\rightsquigarrow}$, and it is formally defined as follows.

Definition 4.4: Let $\beta \in Act$. The set $\overset{\beta}{\rightsquigarrow} \subseteqq Pr$ is inductively defined as follows:

1. $P \overset{\beta}{\rightsquigarrow}_{(0)}$ iff $act(P) = \{\beta\}$,
2. $P \overset{\beta}{\rightsquigarrow}_{(n+1)}$ iff for every $P' \in deri_\beta(P) \neq \emptyset$, for some m, $P' \overset{\beta}{\rightsquigarrow}_{(m)}$ and $m \leqq n$,
3. $P \overset{\beta}{\rightsquigarrow}$ iff $P \overset{\beta}{\rightsquigarrow}_{(n)}$, for some n,

where $deri_\beta(P) = \{P' : \exists \alpha, P \mapsto \xrightarrow{\alpha} P', \alpha \neq \beta\}$ and $act(P) = \{\alpha : \exists P' : P \mapsto \xrightarrow{\alpha} P'\}$. ∎

The condition $deri_\beta(P) \neq \emptyset$ guarantees that processes do not stop before that β is performed. For the following examples PE_1 and PE_2, $PE_1 \overset{e}{\rightsquigarrow}$ and $PE_2 \overset{e}{\not\rightsquigarrow}$, because an execution path abf which does not contain the action e exists in PE_2.

$$\begin{aligned} PE_1 &\equiv a; (b; e; f; \mathbf{stop} \,[]\, e; \mathbf{stop}) \,[]\, c; d; e; \mathbf{stop}, \\ PE_2 &\equiv a; (b; f; \mathbf{stop} \,[]\, e; \mathbf{stop}) \,[]\, c; d; e; \mathbf{stop} \end{aligned}$$

This liveness property can be expressed by the Constant $LIVE_\beta$, as shown in the following proposition. The Constant $LIVE_\beta$ is the least fixpoint of M_β with respect to X, because $X \in Path_\lhd(M_\beta)$.

Proposition 4.3: Let $\beta \in Act$. Define the Constant $LIVE_\beta$ as $LIVE_\beta := M_\beta\{LIVE_\beta/X\}$, where $X \in \mathcal{X}_\lhd$ and M_β is defined as follows :

$$M_\beta \equiv \bigvee\{N_{(\beta,\mathcal{A})} : \mathcal{A} \subseteqq Act, \mathcal{A} \neq \emptyset\},$$
$$N_{(\beta,\mathcal{A})} \equiv \sum\{\lhd\alpha; X : \beta \neq \alpha \in \mathcal{A}\} [] \sum\{\lhd\beta; \mathbf{T} : \beta \in \mathcal{A}\}.$$

Then, $P \models LIVE_\beta \iff P \overset{\beta}{\rightsquigarrow}$.

Proof ($\Rightarrow$) If $(P, LIVE_\beta) \in \theta^{(n)}(\models)$, then $n \geqq 1$, because if $LIVE_\beta \mapsto \lceil S' \rfloor$, then $S' \notin Stb$. We prove that if $(P, LIVE_\beta) \in \theta^{(n+1)}(\models)$ then $P \overset{\beta}{\rightsquigarrow}_{(n)}$ by induction.

For the base case $n = 0$, assume that $(P, LIVE_\beta) \in \theta^{(1)}(\models)$. By Definition 3.1, for some S', $LIVE_\beta \mapsto \lceil S' \rfloor$ and the pair (P, S') satisfies (i_1) and (ii_1). By $\mathbf{Act}_\vee$, $\mathbf{Ch}_\vee$, and $\mathbf{Dis}_\vee$, for some $\mathcal{A}$, $S' \equiv N_{(\beta,\mathcal{A})}\{LIVE_\beta/X\}$ and $\mathcal{A} \neq \emptyset$. Here, if $\mathcal{A} \neq \{\beta\}$, then for some α and P', $\beta \neq \alpha \in \mathcal{A}$, $P \mapsto \overset{\alpha}{\longrightarrow} P'$ and $(P', LIVE_\beta) \in \theta^{(0)}(\mathcal{R})$, because (P, S') satisfies (ii_1). But this is impossible as mentioned above. Therefore, $\mathcal{A} = \{\beta\}$. This implies that $act(P) = \{\beta\}$. Hence, $P \overset{\beta}{\rightsquigarrow}_{(0)}$.

For the induction case, assume that $(P, LIVE_\beta) \in \theta^{(n+2)}(\models)$. By a similar argument to the base case, $LIVE_\beta \mapsto \lceil S' \rfloor$, $S' \equiv N_{(\beta,\mathcal{A})}\{LIVE_\beta/X\}$, $\mathcal{A} = act(P) \neq \emptyset$, and the pair (P, S') satisfies (i_{n+2}) and (ii_{n+2}). The case $\mathcal{A} = \{\beta\}$ is easy. Then, assume that $\mathcal{A} - \{\beta\} \neq \emptyset$. This implies that $deri_\beta(P) \neq \emptyset$. Let $P' \in deri_\beta(P)$, thus for some α and P', $P \mapsto \overset{\alpha}{\longrightarrow} P'$ and $\alpha \neq \beta$. Since (P, S') satisfies (i_{n+2}), for some m, $\lceil S' \rfloor \overset{\alpha}{\longrightarrow} LIVE_\beta$, $(P', LIVE_\beta) \in \theta^{(m+1)}(\mathcal{R})$, and $m \leqq n$. Hence, by induction, $P' \overset{\beta}{\rightsquigarrow}_{(m)}$. Finally, $P \overset{\beta}{\rightsquigarrow}_{(n+1)}$, because for every $P' \in deri_\beta(P) \neq \emptyset$, for some m, $P' \overset{\beta}{\rightsquigarrow}_{(m)}$ and $m \leqq n$.

($\Leftarrow$) By a similar way to the case ($\Rightarrow$), it can be proven that if $P \overset{\beta}{\rightsquigarrow}_{(n)}$ then $(P, LIVE_\beta) \in \theta^{(n+1)}(\models)$. ∎

For the previous examples PE_1 and PE_2, by Proposition 4.3, $PE_1 \models LIVE_e$ and $PE_2 \not\models LIVE_e$, because $PE_1 \overset{e}{\rightsquigarrow}$ and $PE_2 \overset{e}{\not\rightsquigarrow}$.

5. Satisfiability

A number of specifications are sometimes given to a large system instead of its complete specification, because many designers work on the same system design in parallel, and it is not easy for each designer to know the whole system. Such design method decreases responsibility of each designer, but it raises two important issues: *consistency check* of the specifications and *synthesis* of a system to satisfy them.

Since μLOTOS$^{\mathrm{R}}$ has a Conjunction, the consistency of specifications $S_1, \cdots, S_n$ can be checked by a *satisfiability* of the specification $S_1 \wedge \cdots \wedge S_n$, where S is satisfiable if and only if $Proc(S) \neq \emptyset$. Thus, if a process P such that $P \models S$ is found, then S is satisfiable. But it is impossible to check that $P \models S$ for every $P \in Pr$, because Pr is an infinite set. Then, we present an inductive characterization of satisfiability as follows.

Definition 5.1: The set Sat is defined as

$$Sat = \bigcup\{\mathcal{S} \ : \ \mathcal{S} \text{ is a satisfiable set}\} \subseteqq Sp$$

where a set $\mathcal{S}$ is a *satisfiable set*, if $\mathcal{S} \subseteqq \Theta(\mathcal{S}) \subseteqq Sp$, where $\Theta(\mathcal{S})$ is inductively defined for any set $\mathcal{S}$, as follows:

1. $S \in \Theta^{(0)}(\mathcal{S})$ iff for some S', $S \mapsto \lceil S' \rfloor \in \lceil Stb \rfloor$, and for every $\alpha \in Act$, if $\lceil S' \rfloor \overset{\alpha}{\longrightarrow} S''$ then, $S'' \in \mathcal{S}$,
2. $S \in \Theta^{(n+1)}(\mathcal{S})$ iff for some S', $S \mapsto \lceil S' \rfloor \notin \lceil Stb \rfloor$ and for every $\alpha \in Act$, if $\lceil S' \rfloor \overset{\alpha}{\longrightarrow} S''$ then, for some m, $S'' \in \Theta^{(m)}(\mathcal{S})$ and $m \leqq n$,
3. $S \in \Theta(\mathcal{S})$ iff $S \in \Theta^{(n)}(\mathcal{S})$, for some n.

∎

As similar to the definition of the satisfaction $\models$ (Definition 3.1), $S \in Sat$ if and only if S must eventually reach either a stable state or a stop state. In fact, the following expected proposition holds.

Proposition 5.1: Let $S \in Sp$. Then,

$$S \in Sat \iff \text{for some } P,\ P \models S$$

Proof The case ($\Rightarrow$): By Definition 5.1 and Definition 2.4, we can show that if $S \in Sat$, then for some S', $S \mapsto \lceil S' \rfloor \in \lceil Sat \rfloor$. Hence, we show that the following $\mathcal{R}$ is a satisfaction relation:

$$\mathcal{R} = \{(\mathbf{Pr}(S'), S) : S \mapsto \lceil S' \rfloor \in \lceil Sat \rfloor\}$$

where the process $\mathbf{Pr}(S)$ is a Constant defined from the specification S as follows.

$$\mathbf{Pr}(S) := \sum\{\alpha; \mathbf{Pr}(S'') : \exists S', \lceil S \rfloor \overset{\alpha}{\longrightarrow} S', S'' \in Min(S')\}$$

$$Min(S) = \{S' : S \mapsto \lceil S' \rfloor \in \lceil Sat \rfloor, |S'| = |S|\}$$
$$|S| = \min\{n : S \in \Theta^{(n)}(Sat)\}$$

The depth $|S|$ represents the minimum number of transitions to reach either a stable state or a stop state. Let $(\mathbf{Pr}(S'), S) \in \mathcal{R}$, thus $S \mapsto \lceil S' \rfloor \in \lceil Sat \rfloor$. By the definition of Sat, for some n, $S' \in \Theta^{(n)}(Sat)$. Then, we can derive $(\mathbf{Pr}(S'), S) \in \theta^{(n)}(\mathcal{R})$ from $S' \in \Theta^{(n)}(Sat)$ by induction on n, like Lemma Appendix A.1.

The case ($\Leftarrow$): It is easily shown that $S \in \Theta^{(n)}(\mathcal{S})$ from $(P, S) \in \theta^{(n)}(\models)$, where $\mathcal{S} = \{S : \exists P, P \models S\}$. ∎

In the proof of Proposition 5.1, the method to synthesize a process from a specification is shown as the definition of a Constant $\mathbf{Pr}(S)$. This means that when a number of specifications $S_1, \cdots, S_2$ are given, the consistency of them can be checked by $S_1 \wedge \cdots \wedge S_2 \in Sat$, and a process to satisfy them is synthesized by $\mathbf{Pr}(S')$, where $S_1 \wedge \cdots \wedge S_2 \mapsto \lceil S' \rfloor \in \lceil Sat \rfloor$.

Example 5.1: The specifications *SAB* and *SBA* of Example 3.2 are used again. As shown in Example 3.2, $SAB \wedge SBA$ can be satisfied by the process *PR*, thus it is satisfiable. In fact, $SAB \wedge SBA \in Sat$, because the following set $\mathcal{S}$ is a satisfiable set.

$$\mathcal{S} = \{(SAB \wedge SBA), (SBA \wedge SAB)\}$$

Furthermore, the depth of each satisfiable specification is estimated as follows:

$$|SAB \wedge SBA| = 0, \quad |SBA \wedge SAB| = 0,$$
$$|\circ b; SAB \wedge \lhd b; SBA| = 0, \quad |\circ a; SBA \wedge \lhd a; SAB| = 0,$$
$$|\lhd a; SAB \wedge \circ a; SBA| = 1, \quad |\lhd b; SBA \wedge \circ b; SAB| = 1.$$

Then the following process is defined.

$$PAB \equiv \mathbf{Pr}(\circ b; SAB \wedge \lhd b; SBA) := b; PBA$$
$$PBA \equiv \mathbf{Pr}(\circ a; SBA \wedge \lhd a; SAB) := a; PAB$$

The process *PAB* (also written $PAB := b; a; PAB$) alternately performs *a* and *b*. By the proof of Proposition 5.1, $PAB \models SAB \wedge SBA$, because $SAB \wedge SBA \mapsto \lceil \circ b; SAB \wedge \lhd b; SBA \rfloor \in \lceil Sat \rfloor$. Furthermore, by Proposition 3.2, $PAB \models SAB$ and $PAB \models SBA$. Thus, *PAB* is a common process of *SAB* and *SBA*. ∎

6. Related work

For integration or refinement of specifications, a number of approaches were proposed, for example [3], [10], [13]. Brinksma[3] proposed a refined parallel operator with multiple labels. This operator is used to effectively implement logical conjunction in LOTOS. Steen et al.[13] proposed a conjunction operator and a join operator in order to yield a common reduction and a common extension, respectively, in LOTOS. Larsen et al.[10] defined a conjunction operator $\wedge$ for loose specifications in modal CCS. However, these approaches do not consider Disjunction and least fixpoints. Therefore, these are not directly useful for μLOTOS$^{\mathrm{R}}$.

For logical requirements, synthesis algorithms of processes were proposed in [8] and [11]. Kimura et al.[8] presented a synthesis algorithm for CCS-processes by subcalculus of μ-calculus, but the subcalculus does not contain Disjunction. Manna et al.[11] presented an algorithm to synthesize a graph from requirements described in Propositional Temporal Logic (PTL). In PTL, eventualities can be expressed by an operator $\diamond$, but the synthesized graph from PTLs does *not* always represent *all* the processes which satisfy the PTLs, thus it is not conjunction.

7. Conclusion

In this paper, we have presented a process algebra μLOTOS$^{\mathrm{R}}$ which has logical properties : conjunction, disjunction, least fixpoint, and greatest fixpoint, as shown in Proposition 3.2 and Theorem 4.2. Furthermore, the consistency of specifications can be checked and a process to satisfy them can be synthesized by Proposition 5.1.

μLOTOS$^{\mathrm{R}}$ contains a Parallel composition, but several expected properties do not always hold for the Parallel composition as follows:

1. It may be expected that every satisfiable specification must eventually reach a stable state. But, we have defined the satisfiability such that every satisfiable specification must eventually reach *either* a stable state *or a stop state*, because unstable stop states are described by the Parallel composition. For example, if $S_{12} \equiv \lhd a; S_1 \,|[a,b]|\, \lhd b; S_2$, then $S_{12} \in \Theta^{(1)}(Sat)$, because $S_{12} \mapsto \lceil S_{12} \rfloor \not\longrightarrow$ and $\lceil S_{12} \rfloor \notin \lceil Stb \rfloor$. In fact, S_{12} is satisfiable, for example, $\mathbf{stop} \models S_{12}$.

2. Even if $S_1 \,|[G]|\, S_2$ is satisfiable, S_1 and S_2 are not always satisfiable. For example, if $S_1 := \circ a; S_1$ and $S_2 := \lhd b; S_2$, then $S_1 \,|[\emptyset]|\, S_2$ is satisfiable, but S_2 is not satisfiable. It is difficult to completely and soundly define the stability of $S_1 \,|[G]|\, S_2$.

3. The refinement $\sqsubseteq$ is not preserved by the Parallel composition. For example, if $S_1 \equiv \circ a; (\circ b; \mathbf{stop} \vee \circ c; \mathbf{stop})$, $S_2 \equiv \circ a; \circ b; \mathbf{stop} \vee \circ a; \circ c; \mathbf{stop} \vee S_3$, and $S_3 \equiv \circ a; \circ b; \mathbf{stop} \,[]\, \circ a; \circ c; \mathbf{stop}$, then $S_1 \sqsubseteq S_2$, but $S_{13} \equiv S_1 \,|[a,b,c]|\, S_3 \not\sqsubseteq S_2 \,|[a,b,c]|\, S_3 \equiv S_{23}$, because $P \in Proc(S_{13})$ and $P \notin Proc(S_{23})$, where $P \equiv a; b; \mathbf{stop} \,[]\, a; c; \mathbf{stop}$.

It is not easy to redefine a Parallel composition to hold the expected properties, even though Instabilizers $\lhd$ are removed (note 3). The notion of locality[2] may be useful for solving these problems. It is a future work.

References

[1] R.Barbuti,"Selective mu-calculus: new modal operators for proving properties on reduced transition systems," Formal Description Techniques and Protocol Specification, Testing and Verification, FORTE X/PSTV XVII, pp.519–534, 1997.

[2] G.Boudol, I.Castellani, M. Hennessy, and A.Kiehn: Observing localities, Theoretical Computer Science, Vol.114, pp.31-61, 1993.

[3] E.Brinksma, "Constraint-oriented specification in a constructive formal description technique," LNCS 430, Springer-Verlag, pp.130–152, 1989.

[4] C.A.R.Hoare, "*Communicating sequential processes*," Prentice-Hall, 1985.

[5] Y.Isobe, H.Nakada, Y.Sato, and K.Ohmaki, "Stepwise synthesis of multi-specifications using static choice actions (in Japanese)," Foundation of Software Engineering IV (FOSE'97), Lecture Notes 19, Kindaikagaku-sha, pp.12–19, 1997.

[6] Y.Isobe, Y.Sato, and, K.Ohmaki, "Eventuality in LOTOS with a disjunction operator," ASIAN'98, Asian Computing Science Conference, LNCS 1538, Springer-Verlag, pp.263–281, 1998.

[7] P.C.Kanellakis and S.A.Smolka,"CCS expressions, finite state processes, and three problems of equivalence," *Information and Computation*, Vol.86, pp.43–68, 1990.

[8] S.Kimura, A.Togashi and N.Shiratori, "Synthesis algorithm

for recursive processes by μ-calculus," Algorithmic Learning Theory, LNCS 872, Springer-Verlag, pp.379–394, 1994.

[9] K.G.Larsen, "Modal specifications, automatic verification methods for finite state systems", LNCS 407, Springer-Verlag, pp.232–246, 1989.

[10] K.G.Larsen, B.Steffen, and C.Weise, "A constraint oriented proof methodology based on modal transition systems", Tools and Algorithms for the Construction and Analysis of Systems, LNCS 1019, Springer-Verlag, pp.17–40, 1995.

[11] Z.Manna and P.Wolper, "Synthesis of communicating processes from temporal logic specifications," *ACM Trans. on Programming Languages and Systems*, Vol.6, No.1, pp.67-93, 1984.

[12] R.Milner,"*Communication and Concurrency*", Prentice-Hall, 1989.

[13] M.W.A.Steen, H.Bowman, and J.Derrick, "Composition of LOTOS specification", PSTV XV, pp.73–88, 1995.

[14] M.W.A.Steen, H.Bowman, J.Derrick, and E.A.Boiten, "Disjunction of LOTOS specification", FORTE X/PSTV XVII, pp.177–192, 1997.

[15] M.W.A.Steen, "Consistency and composition of process specifications (Chap.5)," Ph.D Thesis, University of Kent at Canterbury, pp.97–136, 1998, (http ://129.12.4.225/people/staff/mwas).

[16] C.Stirling, "An introduction to modal and temporal logics for CCS," Concurrency: Theory, Language, and Architecture, LNCS 491, Springer-Verlag, pp.2–20, 1989.

[17] N.Uchihira, "PQL : modal logic for compositional verification of concurrent programs (in Japanese)," Trans.IEICE, Vol.J75-D-I, No.2, pp.76–87, 1992.

[18] ISO 8807: Information Processing Systems–Open System Interconnection–LOTOS–A formal description technique based on the temporal ordering of observational behavior, 1989.

Appendix A: Lemma

The following lemma is used in the proof of Proposition 3.2.

Lemma Appendix A.1: Let $T \equiv S_1 \wedge S_2$ and $\mathcal{R} = \{(P, S_1) : P \models S_1 \wedge S_2\} \cup \{(P, S_1) : P \models S_2 \wedge S_1\}$. If $(P, T) \in \theta^{(n)}(\models)$, then $(P, S_1) \in \theta^{(n)}(\mathcal{R})$.

Proof This proceeds by induction on n. The base case $n = 0$ is similar to and easier than the induction case. We show the induction case $n + 1$ ($n \geqq 0$). By Definition 3.1, since $(P, T) \in \theta^{(n+1)}(\models)$, for some T', $T \mapsto \lceil T' \rfloor \notin \lceil Stb \rfloor$ and the pair (P, T') satisfies (i_{n+1}) and (ii_{n+1}). By $\mathbf{Con}_\vee$, the disjunctive transition $T \equiv S_1 \wedge S_2 \mapsto \lceil T' \rfloor$ implies that $S_1 \mapsto \lceil S_1' \rfloor$, $S_2 \mapsto \lceil S_2' \rfloor$, and $T' \equiv S_1' \wedge S_2'$. Here, by $\mathbf{Con}_S$, $S_1' \notin Stb$, because $S_1' \wedge S_2' \notin Stb$.

(i_{n+1}) Let $P \mapsto\!\xrightarrow{\alpha} P'$. Since (P, T') satisfies (i_{n+1}), for some (m, T_1''), $\lceil T' \rfloor \xrightarrow{\alpha} T_1''$ and $(P', T_1'') \in \theta^{(m)}(\models)$, $m \leqq n$. This transition must be inferred by either $\mathbf{Con}_3$ or $\mathbf{Con}_4$, because $T' \notin Stb$. The case by $\mathbf{Con}_3$ is omitted, because it is easier than the case by $\mathbf{Con}_4$. By $\mathbf{Con}_4$, the transition $\lceil T' \rfloor \xrightarrow{\alpha} T_1''$ implies that for some S_2'', $\lceil S_2' \rfloor \xrightarrow{\alpha} S_2''$ and $T_1'' \equiv S_{11}'' \wedge S_2''$, where $S_{11}'' \equiv \bigvee\{S_{11}''' : \lceil S_1' \rfloor \xrightarrow{\alpha} S_{11}'''\}$.

Furthermore, since $(P', T_1'') \in \theta^{(m)}(\models)$, it can be shown that for some T''', $T_1'' \mapsto \lceil T''' \rfloor$ and $(P', T''') \in \theta^{(m)}(\models)$. By $\mathbf{Con}_\vee$, this implies that for some S_1''' and S_2''', $S_{11}'' \mapsto S_1'''$, $S_2'' \mapsto S_2'''$, $T''' \equiv S_1''' \wedge S_2'''$. By $\mathbf{Dis}_\vee$, the transition $S_{11}'' \mapsto S_1'''$ implies that for some S_1'', $\lceil S_1' \rfloor \xrightarrow{\alpha} S_1'' \mapsto S_1'''$. Then, by $\mathbf{Con}_\vee$, $T'' \mapsto \lceil T''' \rfloor$, because $S_1'' \mapsto S_1'''$ and $S_2'' \mapsto S_2'''$, where $T'' \equiv S_1'' \wedge S_2''$. Thus, $(P', T'') \in \theta^{(m)}(\models)$ by Definition 3.1, because $T'' \mapsto \lceil T''' \rfloor$ and $(P', T''') \in \theta^{(m)}(\models)$. Finally, by induction, $(P', S_1'') \in \theta^{(m)}(\mathcal{R})$, because $T'' \equiv S_1'' \wedge S_2''$, $(P', T'') \in \theta^{(m)}(\models)$, and $m \leqq n$. These are summarized as : $\lceil S_1' \rfloor \xrightarrow{\alpha} S_1''$, $(P', S_1'') \in \theta^{(m)}(\mathcal{R})$, and $m \leqq n$.

(ii_{n+1}) Let $\lceil S_1' \rfloor \xrightarrow{\alpha} S_1''$. This case is easier than the case (i_{n+1}), and we can obtain that $P \mapsto\!\xrightarrow{\alpha} P'$, $(P', S_1'') \in \theta^{(m)}(\mathcal{R})$, and $m \leqq n$.

Consequently, $(P, S_1) \in \theta^{(n+1)}(\mathcal{R})$, because $S_1 \mapsto \lceil S_1' \rfloor \notin \lceil Stb \rfloor$ and for every $\alpha \in Act$, (i_{n+1}) and (ii_{n+1}) hold for the pair (P, S_1'). ∎

Yoshinao Isobe received his B.E. degree and the M.S. degree in Electrical Engineering from Shibaura Institute of Technology, Japan in 1990 and 1992 respectively. In 1992, he joined Electrotechnical Laboratory (ETL), AIST, MITI. His research interests theoretical aspects for concurrent systems, especially process algebras. He is a member of IEICE, JSSST, and IPSJ.

Yutaka Sato has received his B.E. degree and the Dr.Eng. degree in computer engineering, from Tsukuba University, Tsukuba, Japan, in 1982 and 1987 respectively. In 1987, Dr.Sato entered ETL, AIST, MITI. His research interests include flexible and provable software architecture, practical application of formal specification, especially for user interface management system. He is a member of ACM, JSSST, and IPSJ.

Kazuhito Ohmaki received his B.E. degree of Electrical Engineering from Iwate University, Iwate, Japan in 1974. He received the M.S. and Ph.D. degree of Information Science from Tohoku University, Sendai, Japan in 1976 and 1979, respectively. In 1979, Dr.Ohmaki joined Electrotechnical Laboratory (ETL) of the Agency of Industrial Science and Technology, the Ministry of International Trade and Industry. He was a visiting researcher of ETH (Swiss Federal Institute of Technology) for one year in 1985. From 1992 to 1995, he was a chief of Information Base Section, Computer Science Division, ETL. From 1995 to 1997, he

was a director of Research Planning Office of ETL. Since 1997, he is a director of Computer Science Division, ETL. His research interests include theoretical aspects for software constructions. He is a member of ACM, IEEE, JSSST, IEICE, and IPSJ.