Compositional Properties of Sequential Processes

Naijun Zhan

Lehrstuhl für Praktische Informatik II
Facultät für Mathematik und Informatik
Mannheim Universität
D7,27, 68163 Mannheim, Deutschland
Email: {zhan}@pi2.informatik.uni-mannheim.de

Abstract

It is widely agreed that the modular method is one of the most effective methods to specify and verify complex systems in order to avoid combinatorial explosion. FLC (Fixpoint Logic with Chop) is an important modal logic because of its expressivity and logic properties, e.g., it is strictly more expressive than the μ-calculus. In this paper, we study the compositionality of FLC, namely, to investigate the connection between the connectives of the logic and the constructors of programs. To this end, we first extend FLC with a nondeterministic operator "+" (FLC$^+$ for the extension) and then establish a correspondence between the logic and the basic process algebra with deadlock and termination (abbreviated by BPA$^{\epsilon}_{\delta}$). Finally, we show that as a by-product of the correspondence characteristic formulae for processes of BPA$^{\epsilon}_{\delta}$ up to strong (observational) bisimulation can be constructed compositionally directly from the syntax of processes.

Key words: chop operator, modal logic, compositionality, verification, bisimulation, characteristic formula, process algebra

1 Introduction

There is a growing need for reliable methods in designing correct reactive systems [9] such as computer operating systems and air traffic control systems. These systems are characterized by ongoing, typically nonterminating and highly nondeterministic behavior. Such systems are often used to model "*safety critical systems*" like, e.g., air traffic control systems, nuclear reaction control systems and so on. As any faulty behavior of such systems might imply catastrophical consequences, proving the correctness of such systems with respect to the expected behavior is inevitable. There is a common agreement that formal methods, such as modal and temporal logics [17,24] and process algebra [3,10,19], are effective and reliable methods to design these systems.

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

Because the complexity of large systems is normally uncontrollable, it is necessary that a method for developing such systems is compositional (vertically or horizontally) in order to avoid combinatorial explosion in specifying and verifying them, e.g. [10,19,3,2,16,4,5,15]. The compositional method allows one to build up a large system by composing existing systems with the defined constructors and reduce the problem of correctness for a complex system to similar and simpler correctness problems for the subsystems.

FLC [18] is an extension of the μ-calculus [12] with the sequential composition operator — "chop" (denoted by ";"). [18] pointed out that FLC is strictly more expressive than the μ-calculus because [6,11] proved that only "regular" properties can be defined in the μ-calculus, but characteristic formulae of context-free processes can be defined in FLC. [13,14] investigated the issue of FLC model checking.

The compositionality was stated in [8] as one important requirement that should also be satisfied by specification logic used in a process algebraic setting, that is, any program constructor cons corresponds to an operator **cons** of the logic such that

(a) $P_i \models \phi_i$ for $i = 1, \ldots, n$ implies $cons(P_1, \ldots, P_n) \models \mathbf{cons}(\phi_1, \ldots, \phi_n)$;

(b) $cons(P_1, \ldots, P_n) \models \mathbf{cons}(\phi_1, \ldots, \phi_n)$ is the strongest assertion which can be deduced from $P_i \models \phi_i$ for $i = 1, \ldots, n$.

It is clear that FLC does not meet the above conditions since P meets ϕ and Q satisfies ψ, but we can not get any property that holds in the combined system $P + Q$ according to ϕ and ψ in FLC. In order to guarantee that a specification logic satisfies the above conditions, we have two alternatives: one is to show that for each constructor in process algebra, a corresponding connective can be defined in the logic. To our knowledge, until so far it is still an open problem if a suitable "+" is definable in classical modal logics; the other is directly to introduce a connective, which exactly corresponds to the constructor in process algebra, into the logic like, e.g., in [8,15] a nondeterministic choice "+" is introduced explicitly.

Besides, it is worth investigating the connection between the sequential composition of process algebra and the 'chop' operator of FLC, but it seems no people to do such a job up to now.

In this paper, we first extend FLC with a non-deterministic operator "+" (denoted by FLC$^+$). Intuitively, $P \models \phi + \psi$ means that P consists of two parts P_1 and P_2, which are executed nondeterministically such that $P_1 \models \phi$ and $P_2 \models \psi$. Then we show that the operators $+$, ; and $\nu X.$ in the logic relate to $+$, ; and $rec\, x$ in process algebraic settings such as the basic process algebra with termination and deadlock (BPA$^{\epsilon}_{\delta}$ for short) respectively. Thus, we can claim that the logic is compositional.

What's more, compositionality also makes many senses in practice for the following reasons:

i) It means one more step to the goal to exploit the structure of process terms

for model checking.

ii) It allows to give a precise and compact specification of certain nondeterministic systems.

iii) It is very easy to modify the specification of a system when additional alternatives for the behavior of the system should be admitted.

iv) It enhances the possibility of modularity in model checking which is useful in redesigning of systems.

i) depends on if it is possibly to work out a syntax-directed model checker for FLC$^+$. In fact, we believe that it may be done exploiting the connection between FLC$^+$ and BPA$^{\epsilon}_{\delta}$ that is presented in this paper. To explain the issues ii), iii) and iv), we present the following example: Consider a car factory that wants to establish an assembly line shown in the Figure 1,

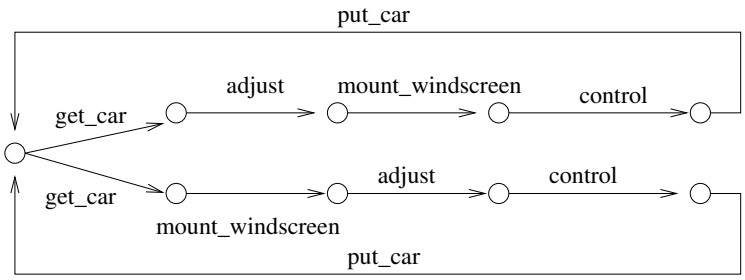

Fig. 1. The Process P

which we denote by the process P, for one production step. If there is a car available for P then P will either get the car, adjust the motor, mount the windscreen, control the car, and put the car on the conveyer band or P will get the car, mount the windscreen, adjust the motor, control the car, and put it back. Then P may start again. The first option can be specified by

$$\mathrm{Spec}_1 \mathrel{\widehat{=}} [\mathrm{get\_car}]; \langle \mathrm{adjust} \rangle; \langle \mathrm{mount\_windscreen} \rangle; \langle \mathrm{control} \rangle; \langle \mathrm{put\_car} \rangle \wedge \langle \mathrm{get\_car} \rangle; \mathrm{true},$$

whereas the second is described by

$$\mathrm{Spec}_2 \mathrel{\widehat{=}} [\mathrm{get\_car}]; \langle \mathrm{mount\_windscreen} \rangle; \langle \mathrm{adjust} \rangle; \langle \mathrm{control} \rangle; \langle \mathrm{put\_car} \rangle \wedge \langle \mathrm{get\_car} \rangle; \mathrm{true}.$$

We are now looking for a specification that admits only such systems that offer both alternatives and that can be easily constructed from Spec_1 and Spec_2. Obviously, $\mathrm{Spec}_1 \wedge \mathrm{Spec}_2$ is not suitable whereas $\mathrm{Spec}_1 \vee \mathrm{Spec}_2$ allows for implementations that exhibit only one of the behavior. $\mathrm{Spec}_1 + \mathrm{Spec}_2$ describes the behavior we have in mind and a system P that offers this behavior repeatedly is described by

$$\mathrm{Spec} \mathrel{\widehat{=}} \nu X.(\mathrm{Spec}_1 + \mathrm{Spec}_2); X.$$

It will be shown that $rec\, x.(P_1 + P_2); x \models \mathrm{Spec}$ in Example 4.9 in Section 4, where

$\mathrm{P}_1 \widehat{=} \mathrm{get\_car}; \mathrm{adjust}; \mathrm{mount\_windscreen}; \mathrm{control}; \mathrm{put\_car}$

$\mathrm{P}_2 \widehat{=} \mathrm{get\_car}; \mathrm{mount\_windscreen}; \mathrm{adjust}; \mathrm{control}; \mathrm{put\_car}.$

Let us now assume that the system specification should be modified to allow for a third alternative behavior Spec_3, then this specification may be simply "added" to form

$$\mathrm{Spec}' \widehat{=} \nu X.(\mathrm{Spec}_1 + \mathrm{Spec}_2 + \mathrm{Spec}_3); X.$$

If we establish $P_3 \models \mathrm{Spec}_3$ then we obtain immediately that

$$rec\ x.(P_1 + P_2 + P_3); x \models \mathrm{Spec}'.$$

In addition, if we have to modify Spec_1 to Spec_1' such that $P_1' \models \mathrm{Spec}_1'$, and obtain

$$rec\ x.(P_1' + P_2 + P_3); x \models \nu X.(\mathrm{Spec}_1' + \mathrm{Spec}_2 + \mathrm{Spec}_3); X.$$

The remainder of this paper is organized as follows: Some basic notions are briefly reviewed in Section 2; The syntax and semantics of FLC^+ are defined in Section 3; Section 4 establishes a connection between the constructors of $\mathrm{BPA}_\delta^\epsilon$ and the connectives of FLC^+; In Section 5, an algorithm to construct a formula Ψ_P for each process $P \in \mathrm{BPA}_\delta^\epsilon$ according to its syntax is presented and we show that $\Psi_P; \surd$ is the characteristic formula of P by the compositionality of FLC^+; Finally, a brief conclusion is provided in Section 6.

2 Preliminaries

Let *Act* be a finite set of (atomic) actions, ranged over by $a, b, c, \ldots$, and $\mathcal{X}$ be a countable set of process variables, ranged over by $x, y, z, ...$

Sequential process terms are those which do not involve parallelism and communication, which are generated by the following grammar:

$$E ::= \delta \ \mid\ \epsilon \ \mid\ x \ \mid\ a \ \mid\ E_1; E_2 \ \mid E_1 + E_2 \ \mid\ rec\ x.E$$

We denote the above language by $\mathcal{P}^s$. Intuitively, we consider that the elements of $\mathcal{P}^s$ represent programs; δ denotes a deadlocked process that cannot execute any action and stays there for ever; ϵ denotes a terminated process that can not proceed, but terminates at once; the other constructors can be conceived as usual ones.

In the presence of the sequential composition operator ";" it is common to use a special predicate $\mathcal{T}$ to evaluate the semantics of the sequential composition operator ";". Let $\mathcal{T} \subset \mathcal{P}^s$ be the least set which contains ϵ and is closed under the rules:

- if $\mathcal{T}(E_1)$ and $\mathcal{T}(E_2)$ then $\mathcal{T}(E_1; E_2)$ and $\mathcal{T}(E_1 + E_2)$; and
- if $\mathcal{T}(E)$ then $\mathcal{T}(rec\ x.E)$.

Convention: For the sake of simplicity, as a well-formedness condition, expressions like $E = E_1 + E_2$ in which $\mathcal{T}(E_1)$ and $\neg\mathcal{T}(E_2)$ or $\neg\mathcal{T}(E_1)$ and $\mathcal{T}(E_2)$ are prohibited.

An occurrence of a variable $x \in \mathcal{X}$ is called *free* in an expression E iff it does not occur within any sub-term of the form $rec\ x.E'$, otherwise called *bound*. We will use $fn(E)$ to stand for all variables which have some free occurrence in E, and $bn(E)$ for all bound variables which have some bound occurrence in E. A variable $x \in \mathcal{X}$ is called *guarded* within a term E iff every occurrence of x is within a sub-term F where F lies in a subexpression $F^*; F$ such that $\neg\mathcal{T}(F^*)$. A term E is called *guarded* iff all variables occurring in it are guarded. The set of all closed and guarded terms of $\mathcal{P}^s$ essentially corresponds to *basic process algebra (*BPA*)* with the terminated process ϵ and the deadlocked process δ, denoted by $\mathrm{BPA}^{\epsilon}_{\delta}$, ranged over by $P, Q, \ldots$. BPA is a sub-fragment of ACP [3].

Example 2.1 The expressions $a; x, a; (b + x), rec\ x.(a + b); x; (y + z), \epsilon, \delta$ are guarded whereas $x, a + x$ are unguarded.

An operational semantics of $\mathcal{P}^s$ is given by the following inference rules:

$$\text{Act}\quad \frac{}{a \xrightarrow{a} \epsilon} \qquad \text{Rec}\quad \frac{E_1[rec.xE_1/x] \xrightarrow{a} E_1'}{rec\ x.E_1 \xrightarrow{a} E_1'}$$

$$\text{Seq-1}\quad \frac{E_1 \xrightarrow{a} E_1'}{E_1; E_2 \xrightarrow{a} E_1'; E_2} \qquad \text{Seq-2}\quad \frac{E_2 \xrightarrow{a} E_2' \text{ and } \surd(E_1)}{E_1; E_2 \xrightarrow{a} E_2'}$$

$$\text{Nd}\quad \frac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1' \text{ and } E_2 + E_1 \xrightarrow{a} E_1'}$$

Definition 2.2 A binary relation $\mathcal{S} \subseteq \mathrm{BPA}^{\epsilon}_{\delta} \times \mathrm{BPA}^{\epsilon}_{\delta}$ is called a strong bisimulation if $(P, Q) \in \mathcal{S}$ implies

- $\surd(P)$ iff $\surd(Q)$; and
- whenever $P \xrightarrow{a} P'$ then, for some $Q', Q \xrightarrow{a} Q'$ and $(P', Q') \in \mathcal{S}$, for any $a \in Act$; and
- whenever $Q \xrightarrow{a} Q'$ then, for some $P', P \xrightarrow{a} P'$ and $(P', Q') \in \mathcal{S}$ for any $a \in Act$.

Given two processes $P, Q \in \mathrm{BPA}^{\epsilon}_{\delta}$, we say P and Q are strongly bisimilar, written $P \sim Q$, if $(P, Q) \in \mathcal{S}$ for some strong bisimulation $\mathcal{S}$.

We can extend the definition of $\sim$ over $\mathcal{P}^s$ as: Let $E_1, E_2 \in \mathcal{P}^s$, and $fn(E_1) \cup fn(E_2) \subseteq \{x_1, \ldots, x_n\}$. If $E_1\{P_1/x_1, \ldots, P_n/x_n\} \sim E_2\{P_1/x_1, \ldots, P_n/x_n\}$ for any $P_1, \ldots, P_n \in \mathrm{BPA}^{\epsilon}_{\delta}$, then $E_1 \sim E_2$.

[1] proved that the following proof system for $\mathrm{BPA}^{\epsilon}_{\delta}$ is complete [1] :

A0	$E_1 + E_2 = E_2 + E_1$	A1	$(E_1 + E_2) + E_3 = E_1 + (E_2 + E_3)$
A2	$E + E = E$	A3	$(E_1 + E_2); E_3 = (E_1; E_3) + (E_2; E_3)$

[1] The proof system for $\mathrm{BPA}^{\epsilon}_{\delta}$ presented in this paper is a little different from the one in [1]. But [1] pointed out that the variant is still complete.

A4 $(E_1;E_2);E_3 = E_1;(E_2;E_3)$ A5 $rec\ x.E = E\{rec\ x.E/x\}$

A6 $E+\delta = E$ A7 $\delta;E = \delta$

A8 $E;\epsilon = E$ A9 $\epsilon;E = E$

Also in [1], the following lemma was shown.

Lemma 2.3 *For any* $P \in \mathrm{BPA}^{\epsilon}_{\delta}$, $P \sim \sum_{a\in Act}\sum_{j=1}^{i_a} a;Q_{a,j}$ *or* $P \sim \delta$, *where* $Q_{a,j} \in \mathrm{BPA}^{\epsilon}_{\delta}$. *Note that if for all* $a \in Act$, $i_a < 1$ *then* $\sum_{a\in Act}\sum_{j=1}^{i_a} a;Q_{a,j}$ *will be abbreviated as* ϵ.

Notations: From now on, we use $\mathcal{A}$ *op* $\mathcal{B}$ to stand for $\{E_1\ op\ E_2 \mid E_1 \in \mathcal{A}$ and $E_2 \in \mathcal{B}\}$, $\mathcal{A}$ *op* E for $\mathcal{A}$ *op* $\{E\}$, where $E \in \mathcal{P}^s, \mathcal{A} \subseteq \mathcal{P}^s, \mathcal{B} \subseteq \mathcal{P}^s$, and $op \in \{+,;\}$.

3 FLC with Nondeterministic Operator "+" (FLC$^+$)

FLC is due to Markus Müller-Olm [18], which is an extension of the modal μ-calculus, can express non-regular properties. It is therefore strictly more expressive than the μ-calculus, since [6,11] proved that only regular properties can be defined in the μ-calculus. Here, we extend FLC with a nondeterministic operator "+", which is similar to the one presented in [7,15], in order to define the composed property according to the ones for the two components which can be executed nondeterministically. Intuitively, $P \models \phi_1 + \phi_2$ means that P consists of P_1 and P_2 which are executed nondeterministically such that P_1 satisfies ϕ_1 and P_2 meets ϕ_2.

3.1 Syntax and Semantics

Let $X, Y, Z, \ldots$ range over an infinite set *Var* of *variables*, *true* and *false* be propositional constants as usual, and $\surd$ for another special propositional constant that is used to indicate if a process is terminated.

Formulae of FLC$^+$ are generated by the following rules:

$$\phi ::= true \mid false \mid \surd \mid term \mid X \mid [a] \mid \langle a\rangle \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid$$
$$\phi_1;\phi_2 \mid \phi_1 + \phi_2 \mid \mu X.\phi \mid \nu X.\phi$$

where $X \in Var$ and $a \in Act$.

In the sequel, we use @ to stand for $\langle a\rangle$ or $[a]$, p for *true*, *false* or $\surd$, and σ for ν or μ.

As in the modal μ-calculus, the two *fixpoint operators* μX and νX bind the respective variable X and we will apply the usual terminology of *free* and *bound occurrences* of variables in a formula, *closed* and *open* formulae etc. We will use $fn(\phi)$ to stand for all variables which have some free occurrence in ϕ and $bf(\phi)$ for all variables that have some bound occurrence in ϕ. We say that X is guarded in ϕ if each occurrence of X is preceded with @ or p via ";". If all variables in ϕ are guarded, then ϕ is called *guarded*.

Example 3.1 Formulae $\langle a\rangle; X, \nu X.(\langle a\rangle \vee \langle b\rangle); X; (Y+Z), \mathit{false}; X$ are guarded, but $X, \langle a\rangle \wedge X, \mu X.(X+Y) \vee [a]$ are not guarded.

As in [18], formulae are interpreted as a *monotonic predicate transformer* which is simply a mapping $f : 2^{\mathrm{BPA}^{\epsilon}_{\delta}} \to 2^{\mathrm{BPA}^{\epsilon}_{\delta}}$ which is monotonic w.r.t. the inclusion ordering on $2^{\mathrm{BPA}^{\epsilon}_{\delta}}$. We use $\mathrm{MPT_T}$ to represent all these monotonic predicate transformers over $\mathrm{BPA}^{\epsilon}_{\delta}$. $\mathrm{MPT_T}$ together with the inclusion ordering defined by

$$f \subseteq f' \text{ iff } f(\mathcal{A}) \subseteq f'(\mathcal{A}) \text{ for all } \mathcal{A} \subseteq \mathrm{BPA}^{\epsilon}_{\delta}$$

forms a complete lattice. We denote the join and meet operators by $\sqcup$ and $\sqcap$. By Tarski-Knaster Theorem, the least and greatest fixed points of monotonic functions:

$$(2^{\mathrm{BPA}^{\epsilon}_{\delta}} \to 2^{\mathrm{BPA}^{\epsilon}_{\delta}}) \to (2^{\mathrm{BPA}^{\epsilon}_{\delta}} \to 2^{\mathrm{BPA}^{\epsilon}_{\delta}})$$

exist. They are used to interpret the fixed point formulae of FLC^{+}.

The meaning of true and false are interpreted in the standard way, i.e. by $\mathrm{BPA}^{\epsilon}_{\delta}$ and $\emptyset$ respectively. The meaning of $\surd$ is to map any set of processes to the subset of $\mathrm{BPA}^{\epsilon}_{\delta}$ in which all terminated processes are contained. So, a process P meets $\surd$ iff $\mathcal{T}(P)$. term is interpreted as an identity. Because ϵ and δ have different behavior in the presence of ;, they should be distinguished by FLC^{+}. To this end, $[a]$ of FLC^{+} is interpreted as a function that maps any set of processes $\mathcal{A}$ to the set in which each process is not terminated and every a-successor of the process must be contained in $\mathcal{A}$. This is different from the meaning of $[a]$ given in FLC. Therefore, according to our interpretation, $P \models [a]$ only if $\neg\mathcal{T}(P)$. Whereas, P always meets $[a]$ for any $P \in \mathrm{BPA}^{\epsilon}_{\delta}$ in FLC. So, it is easy to show that ϵ does not satisfy $\bigwedge_{a\in Act}[a]; \mathit{false}$, while $\bigwedge_{a\in Act}[a]; \mathit{false}$ is the characteristic formula of δ. The meaning of variables is given by an *environment* $\rho : \mathit{var} \to (2^{\mathrm{BPA}^{\epsilon}_{\delta}} \to 2^{\mathrm{BPA}^{\epsilon}_{\delta}})$ that assigns variables to monotonic functions of sets to sets. $\rho[X \rightsquigarrow f]$ agrees with ρ except for associating f with X.

Definition 3.2 Given an environment ρ, the meaning of a formula ϕ, denoted by $\mathcal{C}_\rho(\phi)$, is inductively defined as follows:

$$\begin{aligned}
\mathcal{C}_\rho(\mathit{true})(\mathcal{A}) &= \mathrm{BPA}^{\epsilon}_{\delta}\\
\mathcal{C}_\rho(\mathit{false})(\mathcal{A}) &= \emptyset\\
\mathcal{C}_\rho(\surd)(\mathcal{A}) &= \{P \in \mathrm{BPA}^{\epsilon}_{\delta} \mid \mathcal{T}(P)\}\\
\mathcal{C}_\rho(\mathit{term})(\mathcal{A}) &= \mathcal{A}\\
\mathcal{C}_\rho(X) &= \rho(X)\\
\mathcal{C}_\rho([a])(\mathcal{A}) &= \{P \in \mathrm{BPA}^{\epsilon}_{\delta} \mid \neg\mathcal{T}(P) \wedge \forall P' \in \mathrm{BPA}^{\epsilon}_{\delta}.P \xrightarrow{a} P' \Rightarrow P' \in \mathcal{A}\}\\
\mathcal{C}_\rho(\langle a\rangle)(\mathcal{A}) &= \{P \in \mathrm{BPA}^{\epsilon}_{\delta} \mid \exists P' \in \mathrm{BPA}^{\epsilon}_{\delta}.P \xrightarrow{a} P' \wedge P' \in \mathcal{A}\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{C}_\rho(\phi_1 \wedge \phi_2)(\mathcal{A}) &= \mathcal{C}_\rho(\phi_1)(\mathcal{A}) \cap \mathcal{C}_\rho(\phi_2)(\mathcal{A}) \\
\mathcal{C}_\rho(\phi_1 \vee \phi_2)(\mathcal{A}) &= \mathcal{C}_\rho(\phi_1)(\mathcal{A}) \cup \mathcal{C}_\rho(\phi_2)(\mathcal{A}) \\
\mathcal{C}_\rho(\phi_1; \phi_2) &= \mathcal{C}_\rho(\phi_1) \cdot \mathcal{C}_\rho(\phi_2) \\
\mathcal{C}_\rho(\phi_1 + \phi_2)(\mathcal{A}) &= \{P \in \mathrm{BPA}^\epsilon_\delta \mid \exists P_1, P_2 \in \mathrm{BPA}^\epsilon_\delta . P \sim P_1 + P_2 \wedge \\
&\qquad P_1 \in \mathcal{C}_\rho(\phi_1)(\mathcal{A}) \wedge P_2 \in \mathcal{C}_\rho(\phi_2)(\mathcal{A})\} \\
\mathcal{C}_\rho(\mu X.\phi) &= \sqcap\{f \in \mathrm{MPT_T} \mid \mathcal{C}_{\rho[X \rightsquigarrow f]}(\phi) \subseteq f\} \\
\mathcal{C}_\rho(\nu X.\phi) &= \sqcup\{f \in \mathrm{MPT_T} \mid \mathcal{C}_{\rho[X \rightsquigarrow f]}(\phi) \supseteq f\}
\end{aligned}$$

where $\mathcal{A} \subseteq \mathrm{BPA}^\epsilon_\delta$, and $\cdot$ stands for the compositional operator over functions.

As the meaning of a closed formula ϕ is independent of any environment, we sometimes write $\mathcal{C}(\phi)$ for $\mathcal{C}_\rho(\phi)$, where ρ is an arbitrary environment. We also abuse $\phi(\mathcal{A})$ to stand for $\mathcal{C}_\rho(\phi)(\mathcal{A})$ if ρ is clear from the context.

The set of processes *satisfying* a given closed formula ϕ is $\phi(\mathrm{BPA}^\epsilon_\delta)$. A process P is said to satisfy ϕ iff $P \in \mathcal{C}_\rho(\phi)(\mathrm{BPA}^\epsilon_\delta)$ for some environment ρ, denoted by $P \models_\rho \phi$. If ρ is clear from the context, we directly write $P \models \phi$. $\phi \Rightarrow \psi$ means that $\mathcal{C}_\rho(\phi)(\mathcal{A}) \subseteq \mathcal{C}_\rho(\psi)(\mathcal{A})$ for any $\mathcal{A} \subset \mathrm{BPA}^\epsilon_\delta$ and any ρ. $\phi \Leftrightarrow \psi$ means $(\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$. The other notations can be defined in a standard way.

Given a formula ϕ, the set of sub-formulae at the end of ϕ, denoted by $\mathrm{ESub}(\phi)$, is: $\{\phi\}$ if $\phi = p, term, X$ or $@$; $\mathrm{ESub}(\phi_1) \cup \mathrm{ESub}(\phi_2)$ if $\phi = \phi_1 \; op \; \phi_2$ where $op \in \{\wedge, \vee, +\}$; if $\phi = \phi_1; \phi_2$ then if $\tau \notin \mathrm{ESub}(\phi_2)$ then $\mathrm{ESub}(\phi_2)$ else $(\mathrm{ESub}(\phi_2) - \{\tau\}) \cup \mathrm{ESub}(\phi_1)$; $\mathrm{ESub}(\phi')$ if $\phi = \sigma X.\phi'$. When we say that $\surd$ only occurs at the end of ϕ it means that $\surd$ only can be in $\mathrm{ESub}(\phi)$ as a sub-formula of ϕ.

Convention: In the sequel, we assume the binding precedence among the operators of the logic as follows: "νX" = "μX" > ";" > "$+$" > "$\vee$" = "$\wedge$" > "$\Rightarrow$" = "$\Leftrightarrow$".

3.2 Some Theorems for FLC^+

In this subsection, we prove some theorems for FLC^+ which will be used in the later.

In fact, we can prove that all properties on FLC shown in [18] are still true for FLC^+, e.g., FLC^+ is strictly more expressive than the μ-calculus, FLC^+ is decidable for finite-state processes and undecidable for context-free processes and so on.

Besides, we can show that FLC^+ has the tree model property, viz

Theorem 3.3 *Given $P, Q \in \mathrm{BPA}^\epsilon_\delta$, and $P \sim Q$, then for any closed formula ϕ of FLC^+, $P \models \phi$ iff $Q \models \phi$.*

Definition 3.4 We define a subclass of FLC^+, denoted by $\mathcal{N}$, as follows:

$$\phi ::= p \mid @ \mid \phi \wedge \phi \mid \phi + \phi \mid @; \psi$$

where $a \in Act, \psi \in \mathrm{FLC}^+$.

By the definition of the semantics of FLC$^+$, $\phi; true \Rightarrow \psi; true$ means that for any process P and environment ρ, $P \models_\rho \phi$ implies $P \models_\rho \psi$. For convenience, we abbreviate $\phi; true \Rightarrow \psi; true$ by $\phi \rightarrow \psi$, and $\phi; true \Leftrightarrow \psi; true$ by $\phi \leftrightarrow \psi$.

The following theorem indicates the relationships among the propositional letters, connectives of FLC$^+$.

Theorem 3.5

N1	$term; \phi \Leftrightarrow \phi; term \Leftrightarrow \phi$	P1	$p; \phi \Leftrightarrow p$
P2	$\phi + false \Leftrightarrow false$	P3	$p + p \Leftrightarrow p$
P4	$\langle a \rangle; false \Leftrightarrow false$	T1	$\surd \vee [a]; true \Leftrightarrow true$
T2	$\surd \wedge [a]; true \Leftrightarrow false$	S1	$\phi + \psi \Leftrightarrow \psi + \phi$
S2	$(\phi + \psi) + \varphi \Leftrightarrow \phi + (\psi + \varphi)$	S3	$\phi + \phi \leftrightarrow \phi$ *if* $\phi \in \mathcal{N}$
SI	$\phi + (\psi \wedge \varphi) \Rightarrow (\phi + \psi) \wedge (\phi + \varphi)$	SD	$\phi + (\psi \vee \varphi) \Leftrightarrow (\phi + \psi) \vee (\phi + \varphi)$
SC	$(\phi + \psi); \varphi \Leftrightarrow (\phi; \varphi) + (\psi; \varphi)$	C1	$\phi; \psi \rightarrow \phi$
C2	$(\phi; \psi); \varphi \Leftrightarrow \phi; (\psi; \varphi)$	IC	$(\phi \wedge \psi); \varphi \Leftrightarrow (\phi; \varphi) \wedge (\psi; \varphi)$
DC	$(\phi \vee \psi); \varphi \Leftrightarrow (\phi; \varphi) \vee (\psi; \varphi)$	SD	$[a]; \phi_1 + [a]; \phi_2 \leftrightarrow [a]; (\phi_1 \vee \phi_2)$

The following theorem says that ";" and "+" both are monotonic.

Theorem 3.6 ***(i)*** *If $\phi \Rightarrow \psi$ then $\phi; \varphi \Rightarrow \psi; \varphi$ and $\varphi; \phi \Rightarrow \varphi; \psi$; and* ***(ii)*** *If $\phi_1 \Rightarrow \phi_2$ and $\psi_1 \Rightarrow \psi_2$ then $\phi_1 + \psi_1 \Rightarrow \phi_2 + \psi_2$.*

By extending the tableau based model checker for FLC given in [13] with a rule for "+", namely

$$(+) \quad \frac{(\mathcal{E}, \mathcal{F}) \vdash \phi_1 + \phi_2}{(\mathcal{E}_1, \mathcal{F}) \vdash \phi_1 \quad (\mathcal{E}_2, \mathcal{F}) \vdash \phi_2} \quad \mathcal{E} = \mathcal{E}_1 + \mathcal{E}_2$$

where $\mathcal{E}, \mathcal{E}_1, \mathcal{E}_2, \mathcal{F} \subseteq \mathrm{BPA}^\epsilon_\delta$, we can get a model checker for FLC$^+$, whose complexity is same as the one of the model-checker of FLC presented in [13].

4 Correspondence between FLC$^+$ and BPA$^\epsilon_\delta$

In this section, we discuss how to derive composite properties for a sequential process from those of its components.

4.1 Nondeterminism

It is clear that the connection between "+" of BPA$^\epsilon_\delta$ and "+" of FLC$^+$ can be expressed as follows:

Theorem 4.1 ***(i)*** *For any $P, Q \in \mathrm{BPA}^{\epsilon}_{\delta}$, if $P \models \phi$ and $Q \models \psi$ then $P+Q \models \phi + \psi$;*

(ii) *For any $R \in \mathrm{BPA}^{\epsilon}_{\delta}$, if $R \models \phi+\psi$ then there exist $P, Q \in \mathrm{BPA}^{\epsilon}_{\delta}$ such that $R \sim P+Q$, $P \models \phi$ and $Q \models \psi$.*

4.2 Sequential Composition

In this subsection, we show that the sequential composition of process algebra can be characterized by the chop operator.

Normally, $P \models \phi$ and $Q \models \psi$, but $P;Q \not\models \phi;\psi$ because possibly ϕ only describes some partial executions of P. For example, let $P = a;b$, $Q = c;d$. It's obvious that $P \models \langle a\rangle$ and $Q \models \langle c\rangle$, but $P;Q \not\models \langle a\rangle;\langle c\rangle$. So, we require that the property of the first operand of ; must specify full executions of the corresponding process, that is, $P \models \phi;\surd$. This is similar to the premise of the rule Seq-2 that only after the first segment P of ; finishes the executing, then the second one can be performed.

Another issue is that by the definition of the semantics of $\mathcal{P}^s$, $\epsilon;P \sim P$. Therefore, the properties concerning intermediate terminations should be omitted in the resulting formula. Otherwise, the composed property does not hold in the combined system. E.g., let $P = a;\epsilon$ and $Q = b;\delta$, $\phi = \langle a\rangle;\surd$, and $\psi = \langle b\rangle$. It's obvious that $P \models \phi;\surd$ and $Q \models \psi$ but $P;Q \not\models \phi;\psi$. So, we will replace $\surd$ occurring in ϕ with *term* since *term* is a neutral element of the chop. This is in accordance with that ϵ is a neutral element of the sequential composition ";" (See the axioms A8 and A9).

Besides, since $\phi\{term/\surd\};\psi$ will be as a specification of $P;Q$, it is possible that $\surd$ as a sub-formula of ϕ makes the sub-formulae of ϕ followed it by ; no sense during calculating the meaning of ϕ by P1, but the sub-formulae play a nontrivial role during calculating the meaning of $\phi\{term/\surd\};\psi$. E.g. $\epsilon \models \surd;[a];\langle b\rangle$ and $a;c \models \langle a\rangle;\langle c\rangle$, but $\epsilon;(a;c) \not\models (term;[a];\langle b\rangle);(\langle a\rangle;\langle c\rangle)$. So, we require that $\surd$ only can appear at the end of ϕ as a sub-formula. In fact, such a requirement is reasonable because all formulae can be transform to such kind of the form equivalently by Theorem 3.5.

In a word, we have the following theorem on the sequential composition ";":

Theorem 4.2 *If $\surd$ only occurs at the end of ϕ as a sub-formula, $P \models \phi;\surd$ and $Q \models \psi$ then $P;Q \models \phi\{term/\surd\};\psi$.*

Remark 4.3 Generally speaking, the converse of Theorem 4.2 is not valid, e.g. $(a;b;c+a);(c;d) \models \langle a\rangle;(\langle b\rangle;\langle c\rangle;\langle c\rangle)$, but we can not conclude that $a;b;c+a \models \langle a\rangle;\surd$ and $c;d \models \langle b\rangle;\langle c\rangle;\langle c\rangle$.

4.3 Recursion

In this subsection, we will study how to relate $rec\ x$ to νX. So, in this subsection, all fixed point operators occurring in formulae will refer to ν if no

otherwise stated. To this end, we firstly define a relation called syntactical confirmation between processes and formulae, with the type $\mathcal{P}^s \times FLC^+ \mapsto \{\text{true}, \text{false}\}$, denoted by $\models_{sc}$.

Definition 4.4 Given a formula ϕ, we associate a map from $2^{\mathcal{P}^s}$ to $2^{\mathcal{P}^s}$ with it, denoted by $\widehat{\phi}$, constructed by the following rules:

$$\begin{aligned}
\widehat{\surd}(\mathcal{E}) &\widehat{=} \{E \mid E \in \mathcal{P}^s \wedge \mathcal{T}(E)\}\\
\widehat{true}(\mathcal{E}) &\widehat{=} \mathcal{P}^s\\
\widehat{false}(\mathcal{E}) &\widehat{=} \emptyset\\
\widehat{term}(\mathcal{E}) &\widehat{=} \mathcal{E}\\
\widehat{X}(\mathcal{E}) &\widehat{=} \{x; E \mid E \in \mathcal{E}\}\\
\widehat{\langle a\rangle}(\mathcal{E}) &\widehat{=} \{E \mid \exists E' \in \mathcal{E}.E \xrightarrow{a} E'\}\\
\widehat{[a]}(\mathcal{E}) &\widehat{=} \{E \mid \neg\mathcal{T}(E) \wedge E \text{ is guarded } \wedge \forall E'.E \xrightarrow{a} E' \Rightarrow E' \in \mathcal{E}\}\\
\widehat{\phi_1 \wedge \phi_2}(\mathcal{E}) &\widehat{=} \widehat{\phi_1}(\mathcal{E}) \cap \widehat{\phi_2}(\mathcal{E})\\
\widehat{\phi_1 \vee \phi_2}(\mathcal{E}) &\widehat{=} \widehat{\phi_1}(\mathcal{E}) \cup \widehat{\phi_2}(\mathcal{E})\\
\widehat{\phi_1 + \phi_2}(\mathcal{E}) &\widehat{=} \{E \mid \exists E_1, E_2.E = E_1 + E_2 \wedge E_1 \in \widehat{\phi_1}(\mathcal{E}) \wedge E_2 \in \widehat{\phi_2}(\mathcal{E})\}\\
\widehat{\phi_1; \phi_2}(\mathcal{E}) &\widehat{=} \widehat{\phi_1} \cdot \widehat{\phi_1}(\mathcal{E})\\
\widehat{\sigma X.\phi}(\mathcal{E}) &\widehat{=} \{(rec\ x.E_1); E_2 \mid E_1 \in \widehat{\phi}(\{\epsilon\}) \wedge E_2 \in \mathcal{E}\}
\end{aligned}$$

where $\mathcal{E} \subseteq \mathcal{P}^s$.

$\models_{sc} (E, \phi) = \text{true}$ iff $E \in \widehat{\phi}(\{\epsilon\})$; otherwise, $\models_{sc} (E, \phi) = \text{false}$. Hereinafter, we denote $\models_{sc} (E, \phi) = \text{true}$ by $E \models_{sc} \phi$ and $\models_{sc} (E, \phi) = \text{false}$ by $E \not\models_{sc} \phi$.

Informally, $P \models_{sc} \phi$ means that P and ϕ have a similar syntax, e.g.

$$rec\ x.[(a; x; x; b) + c] \models_{sc} \nu X.[(\langle a\rangle; X; X; \langle b\rangle) \wedge \langle c\rangle].$$

The following theorem states that $\models_{sc}$ itself is compositional too.

Theorem 4.5 *Let $\surd$ only appear at the end of ϕ_1, ϕ_2 and ϕ as a sub-formula. Then,*

i) if $E_1 \models_{sc} \phi_1$ and $E_2 \models_{sc} \phi_2$ then $E_1 + E_2 \models_{sc} \phi_1 + \phi_2$;

ii) if $E_1 \models_{sc} \phi_1$ and $E_2 \models_{sc} \phi_2$ then $E_1; E_2 \models_{sc} \phi_1\{term/\surd\}; \phi_2$;

iii) if $E \models_{sc} \phi$ then $rec\ x.E \models_{sc} \sigma X.\phi\{term/\surd\}$.

Example 4.6 Let $E_1 \widehat{=} (a; x; x) + d$, $E_2 \widehat{=} x; (b + c); y$, $E_3 \widehat{=} a; b; c$, $\phi \widehat{=} \langle a\rangle; X; X$, $\psi \widehat{=} X; \langle b\rangle; Y$ and $\varphi \widehat{=} [a]; \langle b\rangle; \langle c\rangle$. It's obvious that $E_1 \models_{sc} \phi$, $E_2 \models_{sc} \psi$, $E_3 \models_{sc} \varphi$, $E_1 + E_3 \models_{sc} \phi + \psi$, $E_3; (E_1 + E_3) \models_{sc} \varphi; (\phi + \psi)$ and $rec\ x.\ rec\ y.E_3; (E_1 + E_3) \models_{sc} \nu X.\nu Y.(\varphi; (\phi + \psi))$ by Definition 4.4.

The following lemma indicates that replacing $\models_{sc}$ with $\models$, the relation between processes and formulae is preserved by substitution, i.e.,

Lemma 4.7 *Let* $fn(E) \subseteq \{x_1, \ldots, x_n\}$, $fn(\psi) \subseteq \{X_1, \ldots, X_n\}$. *If* $E \models_{sc} \psi$ *and* $P_i \models \phi_i; \surd$ *where* $\surd$ *does not occur neither in* ϕ_i *for* $i \in \{1, \ldots, n\}$ *nor in* ψ, *then*

$$E\{P_1/x_1, \ldots, P_n/X_n\} \models \psi\{\phi_1/X_1, \ldots, \phi_n/X_n\}; \surd.$$

Theorem 4.8 establishes a connection between $\models_{sc}$ and $\models$, so that we can relate $rec\ x$ to νX. For instance, in the above example, we get

$$rec\ x.\ rec\ y.E_3; (E_1 + E_3) \models \nu X.\nu Y.(\varphi; (\phi + \psi)).$$

Theorem 4.8 *If* $P \in \mathrm{BPA}^{\epsilon}_{\delta}$, $\surd$ *only occurs at the end of* ϕ *and* $P \models_{sc} \phi$, *then* $P \models \phi; \surd$.

In order to demonstrate how to apply the compositionality of FLC^{+} to simplify the verification of reactive systems, we continue the example of a car factory given in Section 1.

Example 4.9 In the example, by Definition 4.4, it is easy to show that $\mathrm{P}_1 \models_{sc} \mathrm{Spec}_1$ and $\mathrm{P}_2 \models_{sc} \mathrm{Spec}_2$. Therefore, we have

$$rec\ x.(\mathrm{P}_1 + \mathrm{P}_2); x \models_{sc} \nu X.(\mathrm{Spec}_1 + \mathrm{Spec}_2); X$$

by Theorem 4.5. Furthermore, it follows that

$$rec\ x.(\mathrm{P}_1 + \mathrm{P}_2); x \models \nu X.(\mathrm{Spec}_1 + \mathrm{Spec}_2); X$$

by Theorem 4.8. That is,

$$P \models \mathrm{Spec}.$$

5 Constructing Characteristic Formulae of Sequential Processes

Given an equivalence or preorder $\preceq$ over processes, the characteristic formula for a process P up to $\preceq$ is a formula ϕ_P such that given a process Q, $Q \models \phi_P$ if and only if $Q \preceq P$. Characteristic formulae can be used to relate equational reasoning about processes to reasoning in a modal logic, and therefore to allow proofs about processes to be carried out in a logical framework. [7] proposed an approach to deriving the characteristic formula of a finite CCS-term up to strong bisimulation and observational congruence by defining a translation function from finite CCS-terms to formulae of a modal logic. [23] proposed a method to define characteristic formulae for preorders from the transition graphs of finite-state processes, and applied their approach to finite-state CCS-terms up to strong (weak) bisimulation. [18] gave a method to derive the characteristic formula of a process in BPA up to some preorder by solving equation systems induced by the semantics of the process (the rewrite system of the process). In this section, we present an algorithm to construct the characteristic formula of a process in $\mathrm{BPA}^{\epsilon}_{\delta}$ up to strong bisimulation directly from its syntax in a compositional manner based on the above results.

It is easy to show that $\bigwedge_{a\in Act}[a];false$ (Φ_δ for short) is the characteristic formula for δ, and $\surd$ for ϵ.

For convenience, $\bigwedge_{a\in Act-A}[a];false$ will be abbreviated by Φ_{-A} from now on.

Definition 5.1 Given a process term $E\in\mathcal{P}^s$, we associate with it a formula denoted by Ψ_E derived by the following rules:

$$\begin{aligned}\Psi_\delta &\mathrel{\widehat{=}} \Phi_\delta\\ \Psi_\epsilon &\mathrel{\widehat{=}} \surd\\ \Psi_x &\mathrel{\widehat{=}} X\\ \Psi_a &\mathrel{\widehat{=}} \Phi_{-\{a\}}\wedge(\langle a\rangle\wedge[a])\\ \Psi_{E_1;E_2} &\mathrel{\widehat{=}} \Psi_{E_1}\{term/\surd\};\Psi_{E_2}\\ \Psi_{E_1+E_2} &\mathrel{\widehat{=}} \Psi_{E_1}+\Psi_{E_2}\\ \Psi_{rec\,x.E} &\mathrel{\widehat{=}} \nu X.\Psi_E\{term/\surd\}\end{aligned}$$

Regarding Ψ_E, we have

Lemma 5.2 *For any $E\in\mathcal{P}^s$, $\surd$ only occurs at the end of Ψ_E as a subformula.*

Lemma 5.3 *For any $E\in\mathcal{P}^s$, $E\models_{sc}\Psi_E$ and $E\models_{sc}\Psi_E;\surd$.*

The rest of this section is devoted to prove that $\Psi_P;\surd$ is the characteristic formula of P up to $\sim$ for each $P\in\mathrm{BPA}^\epsilon_\delta$.

The following lemma says that the proof system for $\mathrm{BPA}^\epsilon_\delta$ (See Section 2) will be valid in FLC^+ if P is substituted by $\Psi_P;\surd$ and $=$ by $\Leftrightarrow$. That is,

Lemma 5.4

A0	$\Psi_{E_1+E_2};\surd\Leftrightarrow\Psi_{E_2+E_1};\surd$	A1	$\Psi_{(E_1+E_2)+E_3};\surd\Leftrightarrow\Psi_{E_1+(E_2+E_3)};\surd$
A2	$\Psi_{E+E};\surd\Leftrightarrow\Psi_E;\surd$	A3	$\Psi_{(E_1+E_2);E_3};\surd\Leftrightarrow\Psi_{(E_1;E_3)+(E_2;E_3)};\surd$
A4	$\Psi_{(E_1;E_2);E_3};\surd\Leftrightarrow\Psi_{E_1;(E_2;E_3)};\surd$	*A5*	$\Psi_{rec\,x.E};\surd\Leftrightarrow\Psi_{E\{rec\,x.E/x\}};\surd$
A6	$\Psi_{E+\delta};\surd\Leftrightarrow\Psi_E;\surd$	A7	$\Psi_{\delta;E};\surd\Leftrightarrow\Psi_\delta;\surd\ \Leftrightarrow\ \Psi_\delta$
A8	$\Psi_{E;\epsilon};\surd\Leftrightarrow\Psi_E;\surd$	A9	$\Psi_{\epsilon;E};\surd\Leftrightarrow\Psi_E;\surd$

By the above lemma and the fact that the proof system for $\mathrm{BPA}^\epsilon_\delta$ is complete [1], it is easy to show the following theorem:

Theorem 5.5 (Completeness) *If $E_1\sim E_2$, then $\Psi_{E_1}\Leftrightarrow\Psi_{E_2}$.*

The following lemma plays a key role in the proof for Theorem 5.7.

Lemma 5.6 *For any $P\in\mathrm{BPA}^\epsilon_\delta$, if $\neg\mathcal{T}(P)$ and there exists no P' such that $P\xrightarrow{a}P'$ then $\Psi_P\Rightarrow[a];false$.*

Theorem 5.7 *For any $P\in\mathrm{BPA}^\epsilon_\delta$, $\Psi_P;\surd$ is the characteristic formula of P.*

Remark 5.8 In Theorem 5.7, the condition P is guarded is essential. Otherwise, the theorem is not true any more because Lemma 5.6 will not be valid without the condition. For instance, $\nu X.(X + (\langle a\rangle \wedge [a] \wedge \Phi_{-\{a\}}))$ is equivalent to $\Psi_{rec\ x.(x+a)}$, nevertheless, $(\nu X.(X + (\langle a\rangle \wedge [a] \wedge \Phi_{-\{a\}})))$; $\surd$ is not the characteristic formula of $rec\ x.(x + a)$, since $rec\ x.(x + b + a)$ meets the formula as well, but $rec\ x.(x + b + a) \not\sim rec\ x.(x + a)$.

Example 5.9 Let $P \widehat{=} a; \epsilon$ and $Q \widehat{=} b; \delta$. Then,

$$\Psi_P \widehat{=} (\langle a\rangle \wedge [a] \wedge \Phi_{-\{a\}}); \surd, \text{ and}$$
$$\Psi_Q \widehat{=} (\langle b\rangle \wedge [b] \wedge \Phi_{-\{b\}}); \Phi_\delta$$

by Definition 5.1.

It's obvious that $\Psi_P; \surd$ is the characteristic formula of P and $\Psi_Q; \surd$ is the one of Q. Furthermore, by Definition 5.1,

$$\begin{aligned}&\Psi_{rec\ x.(P;x;x;Q+P)}; \surd \\ \widehat{=}\ &\nu X. \left(\begin{array}{l} (\langle a\rangle \wedge [a] \wedge \Phi_{-\{a\}}); X; X; ((\langle b\rangle \wedge [b] \wedge \Phi_{-\{b\}}); \Phi_\delta) \\ + (\langle a\rangle \wedge [a] \wedge \Phi_{-\{a\}}) \end{array} \right); \surd,\end{aligned}$$

which is exactly the characteristic formula of $rec\ x.(a; x; x; b; \delta + a; \epsilon)$.

6 Related Work and Concluding Remarks

Since modular method plays a key role in developing large systems, many efforts have been done on this topic. For example, [20] investigated such topic from a logic point of view, i.e. specification and implementation of a system both are represented as a formula. While [10,19,3] studied the topic from an algebraic point of view, that is, specification and implementation of a system are represented as a process, and the relation between the specification and the implementation is described by several kinds of equivalences on processes. The advantage of logics lies in abstractness, but it is not easy to implement. In contrast to logics, it is easy to implement a process algebra, even, a process algebra itself can be looked as a programming language like, e.g. [10], but it lacks of abstractness. Although the established equivalences make process algebra itself to be used as a specification language, it is not ideal candidate as argued in [15]. Therefore, an appropriate method to develop a large system is to use a logic as a specification language and a process algebra as a modeling language.

[4,5] studied the compositionality of a choppy logic [22], which is an extension of classic propositional temporal logic [21] (PTL for short) by introducing a chop operator. [15] defined a modal process logic which has compositionality. But the logic can only express regular properties of processes. [18] extended the μ-calculus with the chop operator (FLC), which can define non-regular properties, and therefore is strictly more expressive than the μ-calculus. [13,14] discussed the model-checking problem of FLC. Nevertheless,

FLC has no compositionality, e.g., $Q \models \nu X.[a];\langle b\rangle;X;X;\langle c\rangle \wedge \langle d\rangle$ (ϕ for short) and $P \models \nu X.[a];\langle c\rangle;X;X;\langle c\rangle \wedge \langle d\rangle$ (ψ for short), but it is difficult to find a formula constructed based on ϕ and ψ which is satisfied by $P+Q$. In order to establish a connection between $\mathrm{BPA}_\delta^\epsilon$ and FLC, we introduce a nondeterministic operator "+" into FLC. So, FLC^+ has compositionality. E.g. in the above example, we will see that $P+Q \models \phi+\psi$.

In order to derive invariant properties of recursive processes based on the properties of the subsystems, we define a notion named "syntactical confirmation", denoted by $\models_{sc}$, between $\mathrm{BPA}_\delta^\epsilon$ and FLC^+ and show that $\models_{sc}$ itself is compositional too and if $P \models_{sc} \phi$, then $P \models \phi$, where all fixed point operators occurring in ϕ only refer to the greatest one.

A by-product of the compositionality of FLC^+ is that we present an algorithm to construct the characteristic formula of each process of $\mathrm{BPA}_\delta^\epsilon$ directly according to its syntax in contrast to that [18] derives the characteristic formula for a process by solving the equation system induced by the semantics of the process (the rewrite system of the process).

In this paper, we do not consider the compositionality for parallel operator "|".

Besides, if we re-interpret the modalities $\langle a\rangle$ and $[a]$, characteristic formulae of observable equivalence can also be constructed compositionally in the logic.

Acknowledgement

Special thanks are due to Prof. Mila Majster-Cederbaum for her supervision and many fruitful discussions on the topic related to this paper. The author also thanks the anonymous referees for their useful comments on this paper.

References

[1] L. Aceto and M. Hennessy. Termination, deadlock, and divergence. *Journal of ACM,* Vol. 39, No.1: 147-187. January, 1992.

[2] L. Aceto and M. Hennessy. Towards action-refinement in process algebra. *Information and Computation,* 103:204-269. 1993.

[3] J.A. Bergstra and J.W. Klop. Algebra of communication processes with abstraction. *Theoretical Computer Science,* 37:77-121. 1985.

[4] H. Barringer, R. Kuiper, A. Pnueli. Now you may compose temporal logic specifications. In Proc. 16^{th} STOC, pp. 51-63. 1984.

[5] H. Barringer, R. Kuiper, A. Pnueli. A compositional temporal approach to a CSP-like language. In Proc. IFIP conference, The Role of Abstract Models in Information Processing, pp. 207-227. 1985.

[6] E.A. Emerson and C.S. Jutla. Tree automata, μ-calculus, and determinacy. In proc. 33rd IEEE Symp. on Found. of Comp. Sci., 1991.

[7] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control,* 68:125-145. 1986.

[8] S. Graf and J. Sifakis. A logic for the description of non-deterministic programs and their properties. *Information and Control,* 68:254-270. 1986.

[9] D. Harel and A. Pnueli. On the development of reactive systems. In K.R. Apt, editor, Logics and Models of Concurrent Systems, volume 13 of NATO, ASI Series, pp. 447-498. Springer-Verlag, 1985.

[10] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[11] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. CONCUR'96, LNCS 1119, pp.263-277. Springer-Verlag, 1996.

[12] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science,* 27:333-354. 1983.

[13] M. Lange and C. Stirling. Model checking fixed point logic with chop. FOSSACS'02, LNCS 2303, pp. 250-263. 2002.

[14] M. Lange. Local model checking games for fixed point logic with chop. CONCUR'02, LNCS 2421, pp. 240-254. 2002.

[15] K.G. Larsen and B. Thomsen. A modal process logic. In the proc. of LICS'88, pp.203-210. IEEE Computer Science Society, 1988.

[16] M. Majster-Cederbaum and F. Salger. Towards the hierarchical verification of reactive systems. To appear in *Theoretical Computer Science.*

[17] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems:Specification.* Springer-Verlag, 1992.

[18] M. Müller-Olm. A modal fixpoint logic with chop, STACS'99, LNCS 1563, pp.510-520. 1999.

[19] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[20] B. Moszkowski. *Executing Temporal Logic Programs.* Cambridge University Press, Cambridge, 1986.

[21] A. Pnueli. The temporal logic of programs. In the proc. of 18^{th} STOC, pp:232-239. 1977.

[22] R. Rosner and A. Pnueli. A choppy logic. In the proc. of LICS'86, pp.306-313. IEEE Computer Science Society, 1986.

[23] B. Steffen, A. Ingólfsdóttir. Characteristic formulae for processes with divergence, *Information and Computation,* 110:149-163. 1994.

[24] C. Stirling. *Modal and Temporal Logics for Processes.* Springer-Verlag, 2001.