

An Operational Semantics of Real-Time Process Algebra (RTPA)

Yingxu Wang, University of Calgary, Canada

*Cyprian F. Ngolah, University of Calgary, Canada and
University of Buea, Republic of Cameroon*

ABSTRACT

The need for new forms of mathematics to express software engineering concepts and entities has been widely recognized. Real-time process algebra (RTPA) is a denotational mathematical structure and a system modeling methodology for describing the architectures and behaviors of real-time and nonreal-time software systems. This article presents an operational semantics of RTPA, which explains how syntactic constructs in RTPA can be reduced to values on an abstract reduction machine. The operational semantics of RTPA provides a comprehensive paradigm of formal semantics that establishes an entire set of operational semantic rules of software. RTPA has been successfully applied in real-world system modeling and code generation for software systems, human cognitive processes, and intelligent systems.

Keywords: cognitive informatics; operational semantics; RTPA; real-time process algebra; real-time systems; software engineering; reduction machine;

INTRODUCTION

Real-time process algebra (RTPA) is a denotational mathematical structure and a system modeling methodology for describing the architectures and behaviors of real-time and nonreal-time software systems (Wang, 2002, 2003, 2006a, 2006b, 2007a, 2008a-c). RTPA provides a coherent notation system and a rigorous mathematical structure for modeling software and intelligent systems. RTPA can

be used to describe both *logical* and *physical* models of systems, where logic views of the architecture of a software system and its operational platform can be described using the same set of notations. When the system architecture is formally modelled, the static and dynamic behaviors that perform on the system architectural model, can be specified by a three-level refinement scheme at the system, class, and object levels in a top-down approach.

Although CSP (Hoare, 1978, 1985), the timed CSP (Boucher & Gerth, 1987; Fecher, 2001; Nicollin & Sifakis, 1991), and other process algebra (Baeten & Bergstra, 1991; Milner, 1980, 1989) treated a computational operation as a process, RTPA distinguishes the concepts of meta processes from complex and derived processes by algebraic process operations.

Definition 1: *Operational semantics of a programming language or a formal notation system is the semantics perceived on a given virtual machine, known as the abstract reduction machine, that denotes the semantics of programs or formal system models by its equivalent behaviors implemented on the reduction machine.*

One way to define an operational semantics for a language or formal notation system is to provide a **state transition system** for the language, which allows a formal analysis of the language and permits the study of **relations** between programs (Jones, 2003; Plotkin, 1981; Schneider, 1995). An alternative way is to describe the operations of the language on an abstract deductive machine whose operations are precisely defined (Sloneger & Barry, 1995; Winskel, 1993). In operational semantics, the reduction machine is a virtual machine that is adopted for reducing a given program to values of identifiers modeled in the machine by a finite set of permissible operations (Louden, 1993; McDermid, 1995).

This article presents a comprehensive operational semantics for RTPA on the basis of an abstract reduction machine, which defines inference rules for repetitively reducing a system model in RTPA into the computational values of identifiers and data objects. The abstract syntaxes of RTPA are introduced, and the reduction machine of RTPA is elaborated. Based on these, the operational semantics of 17 RTPA meta processes and 17 RTPA process relations are systematically developed. A comparative analysis of a set of comprehensive formal semantics for RTPA may be referred to (Wang, 2007a). The *deductive semantics* of RTPA is

presented in Wang (2006a, 2008b). The *denotational semantics* of RTPA is reported in Tan and Wang (2008).

THE ABSTRACT SYNTAX OF RTPA

On the basis of the process metaphor of software systems, abstract processes can be rigorously treated as a mathematical entity beyond sets, relations, functions, and abstract concepts. RTPA is a denotational mathematical structure for denoting and manipulating system behavioral processes (Wang, 2002, 2003, 2006a, 2006b, 2008a-c). RTPA is designed as a coherent algebraic system for software and intelligent system modeling, specification, refinement, and implementation. RTPA encompasses 17 meta processes and 17 relational process operations.

Definition 2: *RTPA is a denotational mathematical structure for algebraically denoting and manipulating system behavioural processes and their attributes by a triple, that is:*

$$RTPA \triangleq (\mathcal{T}, \mathfrak{P}, \mathfrak{R}) \quad (1)$$

where

- \mathcal{T} is a set of 17 primitive types for modeling system architectures and data objects;
- \mathfrak{P} a set of 17 meta processes for modeling fundamental system behaviors;
- \mathfrak{R} a set of 17 relational process operations for constructing complex system behaviors.

Detailed descriptions of \mathcal{T} , \mathfrak{P} , and \mathfrak{R} in RTPA will be extended in the following subsections (Wang, 2007a).

The Meta Processes of Software Behaviors in RTPA

RTPA adopts the foundationalism in order to elicit the most primitive computational processes known as the *meta processes*. In this

approach, complex processes are treated as derived processes from these meta processes based on a set of algebraic process composition rules known as the *process relations*.

Definition 3: *A meta process in RTPA is a primitive computational operation that cannot be broken down to further individual actions or behaviors.*

A meta process is an elementary process that serves as a basic building block for modeling software behaviors. *Complex processes* can be composed from meta processes using

process relations. In RTPA, a set of 17 meta processes has been elicited from essential and primary computational operations commonly identified in existing formal methods and modern programming languages (Aho, Sethi, & Ullman, 1985; Higman, 1977; Hoare et al., 1986; Louden, 1993; Wilson & Clark, 1988; Woodcock & Davies, 1996). Mathematical notations and syntaxes of the meta processes are formally described in Table 1.

Lemma 1: *The essential computing behaviours state that the RTPA meta process system \mathfrak{P} encompasses 17 fundamental computational*

Table 1. The meta processes of RTPA

No.	Meta Process	Notation	Syntax
1	Assignment	$:=$	$y\mathbb{T} := exp\mathbb{T}$
2	Evaluation	\blacklozenge	$\blacklozenge_{\mathbb{T}} exp\mathbb{T} \rightarrow \mathbb{T}$
3	Addressing	\Rightarrow	$id\mathbb{T} \Rightarrow MEM[ptr\mathbf{P}] \mathbb{T}$
4	Memory allocation	\Leftarrow	$id\mathbb{T} \Leftarrow MEM[ptr\mathbf{P}] \mathbb{T}$
5	Memory release	$\Leftarrow\Leftarrow$	$id\mathbb{T} \Leftarrow\Leftarrow MEM[\perp]\mathbb{T}$
6	Read	\triangleright	$MEM[ptr\mathbf{P}]\mathbb{T} \triangleright x\mathbb{T}$
7	Write	\triangleleft	$x\mathbb{T} \triangleleft MEM[ptr\mathbf{P}]\mathbb{T}$
8	Input	$ \triangleright$	$PORT[ptr\mathbf{P}]\mathbb{T} \triangleright x\mathbb{T}$
9	Output	$ \triangleleft$	$x\mathbb{T} \triangleleft PORT[ptr\mathbf{P}]\mathbb{T}$
10	Timing	$\underline{\underline{@}}$	$@t\mathbf{TM} @\$t\mathbf{TM}$ $\mathbf{TM} = \mathbf{yy:MM:dd}$ $\quad \mathbf{hh:mm:ss:ms}$ $\quad \mathbf{yy:MM:dd:hh:mm:ss:ms}$
11	Duration	\triangleq	$@t_n\mathbf{TM} \triangleq \$t_n\mathbf{TM} + \Delta n\mathbf{TM}$
12	Increase	\uparrow	$\uparrow(n\mathbb{T})$
13	Decrease	\downarrow	$\downarrow(n\mathbb{T})$
14	Exception detection	$!$	$! (@e\mathbf{S})$
15	Skip	\otimes	\otimes
16	Stop	\boxtimes	\boxtimes
17	System	\S	$\S(\text{SysID}\mathbf{ST})$

operations elicited from the most basic computing, that is:

$$\mathfrak{P} = \{ :=, \blacklozenge, \Rightarrow, \Leftarrow, \nabla, \triangleright, \triangleleft, | \triangleright, | \triangleleft, @, \hat{=}, \hat{\uparrow}, \hat{\downarrow}, !, \otimes, \boxtimes, \S \} \tag{2}$$

As shown in Lemma 1 and Table 1, each meta process is a basic operation on one or more operands such as variables, memory elements, or I/O ports. Structures of the operands and their allowable operations are constrained by their types as described in previous sections. It is noteworthy that not all generally important and fundamental computational operations, as shown in Table 1, had been explicitly identified in conventional formal methods. For instances, the evaluation, addressing, memory allocation/release, timing/duration, and the system processes. However, all these are found necessary and essential in modeling system architectures and behaviors.

Process Operations of RTPA

Definition 4: A process relation in RTPA is an algebraic operation and a compositional rule between two or more meta processes in order to construct a complex process.

A set of 17 fundamental process relations has been elicited from fundamental algebraic and relational operations in computing in order to build and compose complex processes in the context of real-time software systems. Syntaxes and usages of the 17 RTPA process relations are formally described in Table 2. Deductive semantics of these process relations may be referred to (Wang, 2006a, 2007a, 2008a).

Lemma 2: The software composing rules state that the RTPA process relation system \mathfrak{R} encompasses 17 fundamental algebraic and relational operations elicited from basic computing needs, that is:

$$\mathfrak{R} = \{ \rightarrow, \curvearrowright, |, | \dots | \dots, R^*, R^+, R^i, \odot, \times, \parallel, \S, \parallel, \gg, \ll, \ll_r, \ll_e, \ll_i \} \tag{3}$$

The Type System of RTPA

A type is a set in which all member data objects share a common logical property or attribute. The maximum range of values that a variable can assume is a type, which is associated with a set of predefined or allowable operations. A type can be classified as *primitive* and *derived* (complex) types. The former is the most elemental types that cannot further divided into simpler ones; the latter is a compound form of multiple primitive types based on given type rules. Most primitive types are provided by programming languages; while most user defined types are derived ones.

Definition 5: A type system specifies data object modeling and manipulation rules in computing.

The 17 RTPA primitive types in computing and human cognitive process modeling have been elicited from works in (Cardelli & Wegner, 1985; Martin-Lof, 1975; Mitchell, 1990; Stubbs & Webre, 1985; Wang, 2002, 2003, 2007a), which is summarized in the following lemma.

Lemma 3: The primary types of computational objects state that the RTPA type system \mathfrak{T} encompasses 17 primitive types elicited from fundamental computing needs, that is:

$$\mathfrak{T} \triangleq \{ \mathbf{N}, \mathbf{Z}, \mathbf{R}, \mathbf{S}, \mathbf{BL}, \mathbf{B}, \mathbf{H}, \mathbf{P}, \mathbf{TI}, \mathbf{D}, \mathbf{DT}, \mathbf{RT}, \mathbf{ST}, @e\mathbf{S}, @t\mathbf{TM}, @int\mathbf{O}, \mathbf{S}\mathbf{BL} \} \tag{4}$$

where the primitive types stand for natural number, integer, real, string, Boolean, byte, hexadecimal, pointer, time, date, date/Time, run-time determinable type, system architectural type, random event, time event, interrupt event, and status.

In Lemma 3, the first 11 primitive types are for mathematical and logical manipulation

Table 2. The process relations and algebraic operations of RTPA

No.	Process Relation	Notation	Syntax
1	Sequence	\rightarrow	$P \rightarrow Q$
2	Jump	\curvearrowright	$P \curvearrowright Q$
3	Branch	$ $	$\blacklozenge \text{exp} \mathbf{BL} = \mathbf{T} \rightarrow P$ $ \blacklozenge \sim \rightarrow Q$
4	Switch	$ $ \dots $ $	$\blacklozenge \text{exp} \mathbf{T} =$ $i \rightarrow P_i$ $ \sim \rightarrow \emptyset$ where $\mathbf{T} \in \{\mathbf{N}, \mathbf{Z}, \mathbf{B}, \mathbf{S}\}$
5	While-loop	R^*	$\overset{\mathbf{F}}{R}_{\text{exp} \mathbf{BL} = \mathbf{T}} P$
6	Repeat-loop	R^+	$P \rightarrow \overset{\mathbf{F}}{R}_{\text{exp} \mathbf{BL} = \mathbf{T}} P$
7	For-loop	R^i	$\overset{\mathbf{TM}}{R}_{i \mathbf{M} = 1} P(i \mathbf{M})$
8	Recursion	\circ	$\overset{0}{R}_{i \mathbf{M} = \mathbf{TM}} P^{\mathbf{M}} \circ P^{\mathbf{M}-i}$
9	Function call	\mapsto	$P \mapsto F$
10	Parallel	\parallel	$P \parallel Q$
11	Concurrency	$\text{\textcircled{H}}$	$P \text{\textcircled{H}} Q$
12	Interleave	$\text{\textcircled{I}}$	$P \text{\textcircled{I}} Q$
13	Pipeline	\gg	$P \gg Q$
14	Interrupt	\curvearrowleft	$P \curvearrowleft Q$
15	Time-driven dispatch	\hookrightarrow_t	$@t \mathbf{TM} \hookrightarrow_t P_i$
16	Event-driven dispatch	\hookrightarrow_e	$@e_i \mathbf{S} \hookrightarrow_e P_i$
17	Interrupt-driven dispatch	\hookrightarrow_i	$@int_j \text{\textcircled{O}} \hookrightarrow_i P_j$

of data objects in computing, and the remaining six are for system architectural modeling. More rigorous description of RTPA type rules may be referred to (Wang, 2007a).

THE REDUCTION MACHINE OF RTPA

A reduction machine is an abstract machine that defines inference rules for repetitively reducing language constructs until a solid value

or behavior is obtained. Reduction machines model the operational semantics of a given language or formal notation system in three components: *specification*, *control*, and *store*. In other words, an operational semantics describes how the *control* of the machine reduces a given *specification* to values of variables and how the *memory (store)* is changed during the execution of the specification.

In the operational semantics approach, the underlying target machine that operates

and implements the semantic rules is modeled by an abstract reduction machine. A program and its behavior space or the semantic environment are realized by the target computer. An abstract model of a generic reduction machine as the system platform for embodying software semantics can be modeled below.

Definition 6: *The reduction machine, \mathcal{S} , is an abstract logical model of the executing platform of a target machine denoted by a set of parallel or concurrent computing resources as shown in Figure 1.*

As shown in Figure 1, the reduction machine \mathcal{S} for operational semantics is the executing platform that controls all the computing resources of an abstract target machine. The system is logically abstracted as a set of processes and underlying resources, such as the memory, ports, variables, statuses, and the system clock. A process is dispatched and controlled by the system \mathcal{S} , which is triggered by various external, system timing, or interrupt events. The reduction machine of RTPA is not only the platform

of the computing resources such as processes, memory, ports, and system clocks, but also the implementation of the computing mechanisms such as system dispatches, timing, interrupt handling, and system event captures.

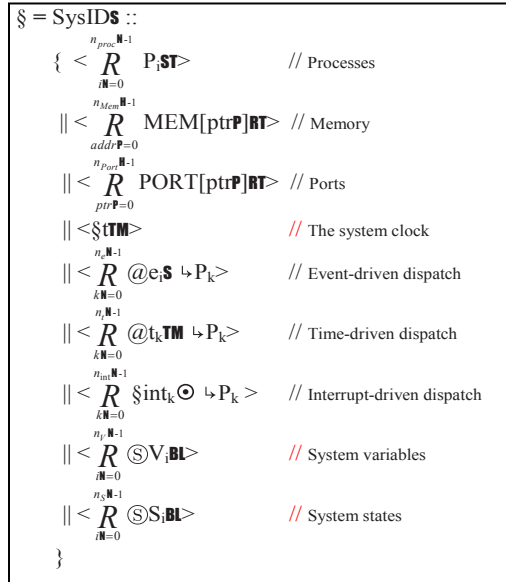
Definition 7. *The semantic environment of the deduction machine, Θ , is the entire set of identifiers and their combinations declared and constrained in the abstract reduction machine, that is:*

$$\Theta \triangleq (I, T, V, A) = I \times T \times V \times A \tag{5}$$

where I is a nonempty set of identifies, T is a set of types corresponding to each identifier in I , V is a set of values corresponding to each identifier in I , and A is a set of addresses corresponding to each identifier in I .

The semantic environment Θ can be divided into three subcategories known as the *operational environment* Θ_{op} , *system environment* Θ_{sys} ,

Figure 1. The abstract model of the RTPA deduction machine



and interrupt environment Θ_{int} , that is:

$$\Theta \triangleq \Theta_{\text{op}} \cup \Theta_{\text{sys}} \cup \Theta_{\text{int}} \quad (6)$$

where only the operational environment Θ_{op} is controllable by users and applications.

Definition 8. An inference rule in operational semantics, R_{os} , is a formal structure in which a propositional conclusion C is derived based on a set of given true premises P , that is:

$$R_{\text{os}} \triangleq \frac{\text{Premise(s)}}{\text{Conclusion}} = P \vdash C \quad (7)$$

where P and C are usually Boolean propositions. However, C can be a sequence of behavioral processes.

The semantic environment Θ forms the context of inference rules in operational semantics. Therefore, the following convention is adopted in all notations of inference rules:

$$\langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle \quad (8)$$

where \parallel denotes a parallel relationship between a process or an expression P and its underlying semantic environment Θ , \Rightarrow denotes a transition between a pair of semantic items, \emptyset denotes an empty operation and/or the completion of a preceding process, and Θ' denotes an updated semantic environment as a result of the operation or effect of a process.

Box 1.

$$\frac{\text{expRT}, \text{yRT} \in \Theta, T(\text{expRT} \parallel \Theta) = T(\text{yRT} \parallel \Theta)}{\langle \text{yRT} := \text{expRT} \rangle \parallel \Theta \Rightarrow \langle \text{V}(\text{yRT}) = \text{V}(\text{expRT}) \rangle \parallel \Theta'} \quad (\text{Rule 1})$$

Definition 9. The evaluations of an identifier id on its type t , value v , and address $addr$ in Θ can be denoted as follows:

$$\begin{aligned} t_{id} &\triangleq T(id \parallel \Theta) \\ v_{id} &\triangleq V(id \parallel \Theta) \\ addr_{id} &\triangleq A(id \parallel \Theta) \end{aligned} \quad (9.a-c)$$

where $id \in I \sqsubseteq \Theta$, or simply denoted $id \in \Theta$ when there is no confusion.

OPERATIONAL SEMANTICS OF RTPA META-PROCESSES

Using the reduction machine as defined in preceding section, the operational semantics for the 17 meta processes of RTPA can be described as follows.

Definition 10. The reduction rule for the assignment process of RTPA, $\text{yRT} := \text{expRT}$, in operational semantics is shown in Box 1. The assignment rule indicates that an assignment transfers the value of expRT into yRT whenever both variables' types are identical or equivalent.

Definition 11. The reduction rule for the Boolean evaluation process of RTPA, $\diamond \text{expBL} \rightarrow \text{BL}$, in operational semantics is shown in Box 2, where $|$ denotes a pair of alternative rules.

The above rule for Boolean evaluation can be extended to more general cases where numerical evaluations are needed. The reduction rule for the numerical evaluation process, $\diamond \text{expT} \rightarrow \text{T}$, in operational semantics is shown

Box 2.

$$\frac{\frac{\frac{exp\mathbf{BL}, \mathbf{T}, \mathbf{F} \in \Theta, \langle \diamond exp\mathbf{BL} \parallel \Theta \rangle \Rightarrow V(exp\mathbf{BL}) = \mathbf{T} \parallel \Theta'_{sys} >}{\langle \diamond exp\mathbf{BL} \rightarrow \mathbf{BL} \rangle \parallel \Theta > \mid \Rightarrow \langle exp\mathbf{BL} = \mathbf{T} \parallel \Theta' >}}{exp\mathbf{BL}, \mathbf{T}, \mathbf{F} \in \Theta, \langle \diamond exp\mathbf{BL} \parallel \Theta \rangle \Rightarrow V(exp\mathbf{BL}) = \mathbf{F} \parallel \Theta'_{sys} >}}{\langle \diamond exp\mathbf{BL} \rightarrow \mathbf{BL} \rangle \parallel \Theta > \mid \Rightarrow \langle exp\mathbf{BL} = \mathbf{F} \parallel \Theta' >}} \quad (\text{Rule 2.a})$$

in Box 3, where \mathbb{T} is a numerical type, i.e., $\mathbb{T} = \{\mathbf{N}, \mathbf{Z}, \mathbf{R}, \mathbf{B}\} \subset \mathbb{T} \sqsubseteq \Theta$.

It is noteworthy that, although the evaluation process does not affect Θ_{op} , but it changes Θ_{sys} .

Definition 12. The reduction rule for the addressing process of RTPA, $id\mathbf{S} \Rightarrow ptr\mathbf{P}$, in operational semantics is shown in Box 4.

Definition 13. The reduction rule for the memory allocations process of RTPA, $id\mathbf{S} \leftarrow MEM[ptr\mathbf{P}]\mathbf{RT}$, in operational semantics is shown in Box 5, where $size(\mathbf{RT})$ is the length of a certain type of variable, \mathbf{RT} , in bytes, which is implementation specific.

Definition 14. The reduction rule for the memory release process of RTPA, $id\mathbf{S} \neq MEM[\perp]\mathbf{RT}$, in operational semantics is shown in Box 6, where \perp denotes an empty or unassigned value.

Definition 15. The reduction rule for the read process of RTPA, $MEM[ptr\mathbf{P}]\mathbf{RT} \triangleright x\mathbf{RT}$, in operational semantics is shown in Box 7.

Definition 16. The reduction rule for the write process of RTPA, $MEM[ptr\mathbf{P}]\mathbf{RT} \triangleleft x\mathbf{RT}$, in operational semantics is shown in Box 8.

Definition 17. The reduction rule for the input process of RTPA, $PORT[ptr\mathbf{P}]\mathbf{RT} \triangleright x\mathbf{RT}$, in operational semantics is shown in Box 9.

Box 3.

$$\frac{\frac{exp\mathbf{T}, n\mathbf{T} \in \Theta, \mathbb{T} = \{\mathbf{N}, \mathbf{Z}, \mathbf{R}, \mathbf{B}\} \subset \Theta, \langle \diamond exp\mathbf{T} \rightarrow \mathbb{T} \rangle \parallel \Theta > \Rightarrow}{\langle V(exp\mathbf{T}) = n\mathbf{T} \parallel \Theta'_{sys} >}}{\langle \diamond exp\mathbf{T} \rightarrow \mathbb{T} \rangle \parallel \Theta' > \Rightarrow \langle exp\mathbf{T} = n\mathbf{T} \parallel \Theta' >}} \quad (\text{Rule 2.b})$$

Box 4.

$$\frac{id\mathbf{S}, ptr\mathbf{P} \in \Theta, \langle id\mathbf{S} \Rightarrow ptr\mathbf{P} \rangle \parallel \Theta > \Rightarrow \langle \emptyset \parallel \Theta' >}{\langle id\mathbf{S} \Rightarrow ptr\mathbf{P} \rangle \parallel \Theta > \text{A} \Rightarrow \langle id\mathbf{S} = ptr\mathbf{P} \parallel \Theta' >}} \quad (\text{Rule 3})$$

Box 5.

$$\frac{id\mathbf{S}, ptr\mathbf{P}, n\mathbf{N} \in \Theta, \langle id\mathbf{S} \leftarrow MEM[ptr\mathbf{P}]\mathbf{RT} \rangle \Rightarrow}{\langle \emptyset \parallel \Theta' >}}{\langle id\mathbf{S} \leftarrow MEM[ptr\mathbf{P}]\mathbf{RT} \rangle \parallel \Theta > \Rightarrow \langle A(id\mathbf{S}) = [ptr\mathbf{P}ptr + size(\mathbf{RT})-1] \parallel \Theta >}} \quad (\text{Rule 4})$$

Box 6.

$$\frac{id\mathbf{S}, ptr\mathbf{P} \in \Theta, \langle id\mathbf{S} \neq MEM[\perp] \mathbf{RT} \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle id\mathbf{S} \neq MEM[\perp] \mathbf{RT} \rangle \parallel \Theta \Rightarrow \{ \langle MEM_{free} \cup MEM[A(id\mathbf{S}), A(id\mathbf{S}) + size(\mathbf{RT}) - 1] \parallel \Theta' \rangle \rightarrow \langle (A(id\mathbf{S}) = \perp) \parallel \Theta' \rangle \}} \quad (\text{Rule 5})$$

Box 7.

$$\frac{x\mathbf{RT}, ptr\mathbf{P}, MEM\mathbf{RT} \in \Theta, \langle (MEM[ptr\mathbf{P}] \mathbf{RT} \triangleright x\mathbf{RT}) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (MEM[ptr\mathbf{P}] \mathbf{RT} \triangleright x\mathbf{RT}) \parallel \Theta \rangle \Rightarrow \{ \langle (x\mathbf{RT} = MEM[ptr\mathbf{P}] \mathbf{RT}) \parallel \Theta' \rangle \}} \quad (\text{Rule 6})$$

Box 8.

$$\frac{x\mathbf{RT}, ptr\mathbf{P}, MEM\mathbf{RT} \in \Theta, \langle (MEM[ptr\mathbf{P}] \mathbf{RT} \triangleleft x\mathbf{RT}) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (MEM[ptr\mathbf{P}] \mathbf{RT} \triangleleft x\mathbf{RT}) \parallel \Theta \rangle \Rightarrow \{ \langle (MEM[ptr\mathbf{P}] \mathbf{RT} = x\mathbf{RT}) \parallel \Theta' \rangle \}} \quad (\text{Rule 7})$$

Definition 18. The reduction rule for the output process of RTPA, $x\mathbf{RT} \mid \triangleleft \text{PORT}[ptr\mathbf{P}] \mathbf{RT}$, in operational semantics is shown in Box 10.

Definition 19. The reduction rule for the timing process of RTPA, $@_i \mathbf{TM} @_{\$} i \mathbf{TM}$, in operational semantics is shown in Box 11.

Definition 20. The reduction rule for the duration process of RTPA, $@_i \mathbf{TM} \triangle_{\$} i \mathbf{TM} + \Delta d \mathbf{L}$, in operational semantics is shown in Box 12.

Definition 21. The reduction rule for the increase process of RTPA, $\uparrow(x\mathbf{RT})$, in operational semantics is shown in Box 13.

Definition 22. The reduction rule for the decrease process of RTPA, $\downarrow(x\mathbf{RT})$, in operational semantics is shown in Box 14.

Definition 23. The reduction rule for the exceptional detection process of RTPA, $!(@e\mathbf{S})$, in operational semantics is shown in Box 15, where $CRT\mathbf{ST}$ is the standard output device of the reduction machine for displaying system information in the type of strings.

Definition 24. The reduction rule for the skip process of RTPA, \otimes , in operational semantics is shown is:

Box 9.

$$\frac{x\mathbf{RT}, ptr\mathbf{P}, \text{PORT}\mathbf{RT} \in \Theta, \langle (\text{PORT}[ptr\mathbf{P}] \mathbf{RT} \triangleright x\mathbf{RT}) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (\text{PORT}[ptr\mathbf{P}] \mathbf{RT} \triangleright x\mathbf{RT}) \parallel \Theta \rangle \Rightarrow \{ \langle (x\mathbf{RT} = \text{PORT}[ptr\mathbf{P}] \mathbf{RT}) \parallel \Theta' \rangle \}} \quad (\text{Rule 8})$$

Box 10.

$$\frac{xRT, ptrP, PORTRT \in \Theta, \langle (PORT[ptrP]RT \leq xRT) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (PORT[ptrP]RT \leq xRT) \parallel \Theta \rangle \Rightarrow \{ \langle (PORT[ptrP]RT = xRT) \parallel \Theta' \rangle \}} \quad (\text{Rule 9})$$

Box 11.

$$\frac{tTM, \xi tTM, \Delta dZ \in \Theta, \langle (@tTM = \xi tTM) \parallel \Theta' \rangle, \langle (@tTM @ \xi tTM) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (@tTM @ \xi tTM) \parallel \Theta \rangle \Rightarrow \{ \langle (@tTM = \xi tTM) \parallel \Theta' \rangle \rightarrow \langle @tTM \hookrightarrow P \parallel \Theta'' \rangle \}} \quad (\text{Rule 10})$$

Box 12.

$$\frac{tTM, \xi tTM, \Delta dZ \in \Theta, \langle (tTM = \xi tTM + \Delta dZ) \parallel \Theta' \rangle, \langle (@tTM \Delta \xi tTM + \Delta dZ) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (@tTM \Delta \xi tTM + \Delta dZ) \parallel \Theta \rangle \Rightarrow \{ \langle (@tTM = \xi tTM + \Delta dZ) \parallel \Theta' \rangle \rightarrow \langle @tTM \hookrightarrow P \parallel \Theta'' \rangle \}} \quad (\text{Rule 11})$$

Box 13.

$$\frac{xRT \in \Theta, RT \in \{N, B, H, Z, P, TM\} \subset \Theta, \langle (\uparrow(xRT)) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (\uparrow(xRT)) \parallel \Theta \rangle \Rightarrow \{ \langle (xRT = xRT + 1) \parallel \Theta' \rangle \}} \quad (\text{Rule 12})$$

Box 14.

$$\frac{xRT \in \Theta, RT \in \{N, B, H, Z, P, TM\} \subset \Theta, \langle (\downarrow(xRT)) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (\downarrow(xRT)) \parallel \Theta \rangle \Rightarrow \{ \langle (xRT = xRT - 1) \parallel \Theta' \rangle \}} \quad (\text{Rule 13})$$

Box 15.

$$\frac{eS, ptrP, CRTP \in \Theta, \langle ptrP = A(CRTST) \parallel \Theta' \rangle, \langle !(@eS) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle !(@eS) \parallel \Theta \rangle \Rightarrow \{ \langle (PORT[ptrP]S = @eS) \parallel \Theta' \rangle \}} \quad (\text{Rule 14})$$

$$\frac{\langle \otimes \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta'_{sys} \rangle}{(\text{Rule 15})}$$

As defined above, the rule for the skip process of RTPA is an axiom, which does not

ing functionally from users' point of view, but jumps to a new point of process by changing the control variables of program execution sequence in Θ'_{sys} .

Definition 25. The reduction rule for the stop process of RTPA, \boxtimes , in operational semantics is as follows:

$$\frac{\langle \boxtimes \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' = \emptyset \rangle}{\text{(Rule 16)}}$$

The rule for the stop process in RTPA is an axiom, which terminates the current system and releases all existing identifiers and their values in the environment.

Definition 26. The reduction rule for the system process of RTPA, $\S(\text{SysIDS})$, in operational semantics is shown in Box 16.

The rule for the system process in RTPA is an axiom, which does nothing functionally from users' point of view, but it creates the system identifier and allocates necessary resources to the newly created system.

OPERATIONAL SEMANTICS OF PROCESS RELATIONS

RTPA process relations are rules of algebraic operations of processes, which describe how the

meta processes can be combined to form complex processes. The operational semantics of the 17 process relations of RTPA is elaborated in this section on the platform of the reduction machine.

Definition 27. The reduction rule for the sequential process of RTPA, $P \rightarrow Q$, in operational semantics is shown in Box 17.

Definition 28. The reduction rule for the jump process of RTPA, $P \curvearrowright Q$, in operational semantics is shown in Box 18.

Definition 29. The reduction rule for the branch process of RTPA, $\diamond \text{exp} \mathbf{RT} \rightarrow P \mid \diamond \rightsquigarrow Q$, in operational semantics is shown in Box 19, where \mid denotes a pair of alternative sub-rules dependent on given conditions.

Definition 30. The reduction rule for the switch process of RTPA, $\diamond \text{exp} \mathbf{RT} \rightarrow P_i \mid \diamond \rightsquigarrow \otimes$, in operational semantics is shown in Box 20, where

$$\mathbf{R}_{i \in \mathbb{N}}^{\mathbb{N}}$$

denotes a set of recurrent structures.

Box 16.

$$\frac{\langle \S(\text{SysIDS}) \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' = (I \cup \{\text{SysID}\}) \rangle,}{\begin{aligned} t(\text{SysID}) &= \mathbf{S}, v(\text{SysID}) = \S = (\perp, \perp, \perp, \perp), \\ a(\text{SysID}) &= A(\text{SysID}) \end{aligned}}{\text{(Rule 17)}}$$

Box 17.

$$\frac{\langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle Q \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (P \rightarrow Q) \parallel \Theta \rangle \Rightarrow \{ \langle P \parallel \Theta' \rangle \rightarrow \langle Q \parallel \Theta' \rangle \}}{\text{(Rule 18)}}$$

Box 18.

$$\frac{\langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle Q \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle (P \curvearrowright Q) \parallel \Theta \rangle \Rightarrow \{ \langle P \parallel \Theta' \rangle \rightarrow \langle \emptyset \parallel \Theta' \cup \Theta_{\text{sys}} \rangle \rightarrow \langle Q \parallel \Theta' \cup \Theta'_{\text{sys}} \rangle \}}{\text{(Rule 19)}}$$

Box 19.

$$\frac{\frac{\text{expBL} \in \Theta, \langle \text{expBL} \parallel \Theta \rangle \Rightarrow \text{expBL} = \mathbf{T}, \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle \blacklozenge \text{expRT} \rightarrow P \mid \blacklozenge \sim \rightarrow Q \rangle \Rightarrow \langle P \parallel \Theta' \rangle}}{\frac{\text{expBL} \in \Theta, \langle \text{expBL} \parallel \Theta \rangle \Rightarrow \text{expBL} = \mathbf{F}, \langle Q \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle \blacklozenge \text{expRT} \rightarrow P \mid \blacklozenge \sim \rightarrow Q \rangle \Rightarrow \langle Q \parallel \Theta' \rangle}} \quad (\text{Rule 20})$$

Box 20.

$$\frac{\begin{array}{l} i\mathbf{N}, n\mathbf{N} \in \Theta, \langle \text{expRT} \parallel \Theta \rangle \Rightarrow \text{expRT} = i\mathbf{N}, 1 \leq i\mathbf{N} \leq n\mathbf{N}, \\ \langle \mathbf{R}_{i\mathbf{N}=1}^{n\mathbf{N}} P \parallel \Theta \rangle \Rightarrow \langle P_i \parallel \Theta' \rangle \\ \langle \blacklozenge \text{exp}_i \text{RT} \rightarrow P_i \mid \blacklozenge \sim \rightarrow \otimes \rangle \Rightarrow \langle P_i \parallel \Theta' \rangle \\ i\mathbf{N}, n\mathbf{N} \in \Theta, \langle \text{expRT} \parallel \Theta \rangle \Rightarrow \text{expRT} = i\mathbf{N}, i\mathbf{N} \notin [1, n\mathbf{N}], \\ \langle \mathbf{R}_{i\mathbf{N}=1}^{n\mathbf{N}} P \parallel \Theta \rangle \Rightarrow \langle \otimes \parallel \Theta \rangle \end{array}}{\langle \blacklozenge \text{exp}_i \text{RT} \rightarrow P_i \mid \blacklozenge \sim \rightarrow \otimes \rangle \Rightarrow \langle \otimes \parallel \Theta'_{\text{sys}} \rangle} \quad (\text{Rule 21})$$

The operational semantics of iterations are presented in the following definitions. It is noteworthy that iterations were diversely interpreted in literature (Louden, 1993; McDermid, 1991). Although the decision point may be denoted by branch constructs, most existing operational semantic rules failed to express the key semantics of “while” and the rewinding action of loops. Further, the semantics for more complicated types of iterations, such as the repeat-loop and for-loop, are rarely found in the literature.

Definition 31. *The reduction rule for the while-loop process of RTPA,*

$$\mathbf{R}_{\text{expBL}=\mathbf{T}}^{\mathbf{F}} P,$$

in operational semantics is shown in Box 21, where the while-loop is defined recursively on Θ .

The above rule indicates that, when a Boolean expression expBL in the environment

Θ is true, the execution of the loop body P as a process for an iteration under Θ can be reduced to the same loop under an updated environment Θ' , which is resulted by the last execution of P ; When $\text{expBL} = \mathbf{F}$ in Θ , the loop is reduced to a termination or exit \otimes .

Definition 32. *The reduction rule for the repeat-loop process of RTPA,*

$$P \rightarrow \mathbf{R}_{\text{expBL}=\mathbf{T}}^{\mathbf{F}} P,$$

in operational semantics is shown in Box 22.

The above rule indicates that the semantics of a repeat-loop process is semantically equivalent to the sequential composition of P and a while-loop.

Definition 33. *The reduction rule for the for-loop process of RTPA,*

$$\mathbf{R}_{i\mathbf{N}=1}^{n\mathbf{N}} P(i\mathbf{N}),$$

in operational semantics is shown in Box 23.

Box 21.

$$\begin{array}{c}
 \frac{exp\mathbf{BL} \in \Theta, \langle exp\mathbf{BL} \parallel \Theta \rangle \Rightarrow exp\mathbf{BL} = \mathbf{T}, \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta \rangle \Rightarrow \{ \langle P \parallel \Theta' \rangle \rightarrow \langle \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta' \rangle \}} \\
 \left| \frac{exp\mathbf{BL} \in \Theta, \langle exp\mathbf{BL} \parallel \Theta \rangle \Rightarrow exp\mathbf{BL} = \mathbf{F}, \langle \otimes \parallel \Theta'_{sys} \rangle}{\langle \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta \rangle \Rightarrow \langle \otimes \parallel \Theta'_{sys} \rangle} \right. \\
 \end{array}$$

(Rule 22)

Box 22.

$$\begin{array}{c}
 \frac{exp\mathbf{BL} \in \Theta, \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta \rangle}{\langle P \rightarrow \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta \rangle \Rightarrow \{ \langle P \parallel \Theta' \rangle \rightarrow} \\
 (\frac{\langle exp\mathbf{BL} \parallel \Theta' \rangle \Rightarrow exp\mathbf{BL} = \mathbf{T}, \langle P \parallel \Theta' \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta' \rangle \Rightarrow \{ \langle P \parallel \Theta'' \rangle \rightarrow \langle \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta'' \rangle \}} \\
 \left| \frac{\langle exp\mathbf{BL} \parallel \Theta' \rangle \Rightarrow exp\mathbf{BL} = \mathbf{F}, \langle \otimes \parallel \Theta'_{sys} \rangle}{\langle \overset{\mathbf{F}}{\underset{exp\mathbf{BL}=\mathbf{T}}{\mathbf{R}}} P \parallel \Theta' \rangle \Rightarrow \langle \otimes \parallel \Theta'_{sys} \rangle} \right. \\
) \\
 \} \\
 \end{array}$$

(Rule 23)

Box 23.

$$\begin{array}{c}
 \frac{i\mathbf{N}, n\mathbf{N} \in \Theta, 1 \leq i\mathbf{N} \leq n\mathbf{N}, \langle P_i \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle \overset{\mathbf{F}}{\underset{i\mathbf{N}=1}{\mathbf{R}}} P(i\mathbf{N}) \parallel \Theta \rangle \Rightarrow \{ \langle P(i\mathbf{N} = 1) \parallel \Theta' \rangle \rightarrow} \\
 \langle P(i\mathbf{N} = 2) \parallel \Theta'' \rangle \rightarrow \dots \rightarrow \\
 \langle P(i\mathbf{N} = n\mathbf{N}) \parallel \Theta^{n'} \rangle \} \\
 \end{array}$$

(Rule 24)

The above rule indicates that the semantics of a for-loop process is semantically equivalent to a sequence of n serial processes. The semantic rule of for-loop processes may also be defined recursively as that of the while-loop rule as shown in Box 24.

Definition 34. The reduction rule for the function call process of RTPA, $P \rightarrow F$, in operational semantics is shown in Box 25.

Definition 35. The reduction rule for the recursion process of RTPA, $P \circ P$, in operational semantics is shown in Box 26.

The semantic rule of recursion processes may also be defined iteratively as that of the for-loop rule as shown in Box 27, where the base process P^0 should be able to be reduced to a constant.

Box 24.

$$\begin{array}{c}
\frac{\langle \blacklozenge(1 \leq i\mathbf{N} \leq n)\mathbf{BL} \parallel \Theta \rangle \Rightarrow \mathbf{T}, \langle P_i \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle \prod_{i=1}^n P \parallel \Theta \rangle \Rightarrow \{ \langle P_i \parallel \Theta' \rangle \rightarrow \langle i\mathbf{N} = i\mathbf{N} + 1 \rangle \rightarrow \langle \prod_{i=1}^n P \parallel \Theta' \rangle \}} \\
\frac{\langle \blacklozenge(1 \leq i\mathbf{N} \leq n)\mathbf{BL} \parallel \Theta \rangle \Rightarrow \mathbf{F}, \langle \otimes \parallel \Theta'_{sys} \rangle}{\langle \prod_{i=1}^n P \parallel \Theta \rangle \Rightarrow \langle \otimes \parallel \Theta'_{sys} \rangle}
\end{array}$$

(Rule 25)

Box 25.

$$\frac{\langle P \parallel \Theta \rangle \Rightarrow \{ \langle P' \parallel \Theta' \rangle \rightarrow \langle P'' \parallel \Theta'' \rangle \}, \langle F \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle P \rightarrow F \parallel \Theta \rangle \Rightarrow \{ \langle P' \parallel \Theta' \rangle \rightarrow \langle F \parallel \Theta'' \rangle \rightarrow \langle P'' \parallel \Theta'' \rangle \}}$$

(Rule 26)

Box 26.

$$\frac{\langle P \parallel \Theta \rangle \Rightarrow \{ \langle P' \parallel \Theta' \rangle \rightarrow \langle P'' \parallel \Theta'' \rangle \}, \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle}{\langle P \cup P \parallel \Theta \rangle \Leftrightarrow \langle P' \parallel \Theta' \rangle \rightarrow \langle P \cup P \parallel \Theta'' \rangle \rightarrow \langle P'' \parallel \Theta'' \rangle}$$

(Rule 27)

Box 27.

$$\frac{i\mathbf{N}, n\mathbf{N} \in \Theta, 1 \leq i\mathbf{N} \leq n\mathbf{N}, \langle P^{i\mathbf{N}} \parallel \Theta \rangle \Rightarrow \langle P^{i\mathbf{N}-1} \parallel \Theta' \rangle, P^0 = \text{const}\mathbf{N}}{\langle P \cup P \parallel \Theta \rangle (\Rightarrow \prod_{i=n\mathbf{N}}^1 \langle P^{i\mathbf{N}} \rightarrow P^{i\mathbf{N}-1} \rangle \parallel \Theta')}$$

(Rule 28)

Definition 36. The reduction rule for the parallel process of RTPA, $P \parallel Q$, in operational semantics is shown in Box 28, where S_1 and S_2 are two additional synchronization processes introduced by the system.

The parallel process rule models the process relation with the single-clock multi-processor (SCMP) structure. In other words, it behaves in a synchronized system or a common environment.

Definition 37. The reduction rule for the concurrent process of RTPA, $P \text{ \textcircled{f} } Q$, in operational semantics is shown in Box 29, where C_1 and C_2 are two additional communication processes introduced in two separated system environments Θ_p and Θ_q .

The concurrent processes model the process relation with the multi-clock multi-processor (MCMP) structure. In other words, it behaves in an asynchronized system or a separated environments linked by the communication means.

Definition 38. The reduction rule for the interleave process of RTPA, $P \parallel\parallel Q$, in operational semantics is shown in Box 30.

The rule of interleave processes models the process relation with the single-clock single-processor (SCSP) structure in a synchronized environment.

Definition 39. The reduction rule for the pipeline process of RTPA, $P \gg Q$, in operational semantics is shown in Box 31.

Box 28.

$$\begin{array}{c}
 \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle Q \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle S_1 \parallel \Theta \rangle \Rightarrow \\
 \langle \emptyset \parallel \Theta \rangle, \langle S_2 \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta \rangle \\
 \hline
 \langle (P \parallel Q) \parallel \Theta \rangle \Rightarrow \{ \langle S_1 \parallel \Theta_{sys} \rangle \rightarrow \left\langle \begin{array}{c} \rightarrow \langle P \parallel \Theta' \rangle \rightarrow \\ \rightarrow \langle Q \parallel \Theta'' \rangle \rightarrow \end{array} \right\rangle \rightarrow \\
 \langle S_2 \parallel \Theta' \cup \Theta'' \cup \Theta'_{sys} \rangle \}
 \end{array}
 \quad (\text{Rule 29})$$

Box 29.

$$\begin{array}{c}
 \langle P \parallel \Theta_P \rangle \Rightarrow \langle \emptyset \parallel \Theta'_P \rangle, \langle Q \parallel \Theta_Q \rangle \Rightarrow \langle \emptyset \parallel \Theta'_Q \rangle, \\
 \langle C_1 \parallel \{ \Theta_P \parallel \Theta_Q \} \rangle \Rightarrow \langle \emptyset \parallel \{ \Theta_P \parallel \Theta_Q \} \rangle, \\
 \langle C_2 \parallel \{ \Theta_P \parallel \Theta_Q \} \rangle \Rightarrow \langle \emptyset \parallel \{ \Theta_P \parallel \Theta_Q \} \rangle \\
 \hline
 \langle (P \parallel\!\!\parallel Q) \parallel \{ \Theta_P \parallel \Theta_Q \} \rangle \Rightarrow \{ \langle C_1 \parallel \{ \Theta_P \parallel \Theta_Q \} \rangle \\
 \Leftrightarrow \left\langle \begin{array}{c} \rightarrow \langle P \parallel \Theta'_P \rangle \rightarrow \\ \rightarrow \langle Q \parallel \Theta'_Q \rangle \rightarrow \end{array} \right\rangle \rightarrow C_2 \parallel \{ \Theta'_P \parallel \Theta'_Q \} \}
 \end{array}
 \quad (\text{Rule 30})$$

Box 30.

$$\begin{array}{c}
 i\mathbf{N}, n\mathbf{N} \in \Theta, 1 \leq i\mathbf{N} \leq n\mathbf{N}, \langle P \parallel \Theta \rangle \Rightarrow \prod_{i\mathbf{N}=1}^{n\mathbf{N}} \langle P_i \parallel \Theta_i \rangle, \\
 \langle Q \parallel \Theta \rangle \Rightarrow \prod_{i\mathbf{N}=1}^{n\mathbf{N}} \langle Q_i \parallel \Theta_i \rangle \\
 \hline
 \langle (P \parallel\!\!\parallel Q) \parallel \Theta \rangle \Rightarrow \prod_{i\mathbf{N}=1}^{n\mathbf{N}} (\langle P_i \parallel \Theta_i \rangle \rightarrow \langle Q_i \parallel \Theta_i \rangle)
 \end{array}
 \quad (\text{Rule 31})$$

Box 31.

$$\begin{array}{c}
 O_P, I_Q, i\mathbf{N}, n\mathbf{N} \in \Theta, \prod_{i\mathbf{N}=1}^{n\mathbf{N}} o_p = i_q, o_p \in O_P, i_q \in I_Q, \#O_P = \#I_Q, \\
 \langle P \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle, \langle Q \parallel \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta' \rangle \\
 \hline
 \langle P \gg Q \parallel \Theta \rangle \Rightarrow \{ \langle P \parallel \Theta' \rangle \rightarrow \langle Q \parallel \Theta'' \rangle \}
 \end{array}
 \quad (\text{Rule 32})$$

The rule of pipeline processes shows that from the functional point view, a pipeline process relation is equivalent to a sequential relation as long as the corresponding outputs O_p and inputs I_q are one-to-one coupled between the two processes.

Definition 40. *The reduction rule for the interrupt process of RTPA, $P \not\prec Q$, in operational semantics is shown in Box 32, where the interrupt semantic environment Θ_{int} is a subset of Θ as defined in Equation 6.*

Box 32.

$$\begin{array}{c}
\langle P \parallel \Theta \rangle \Rightarrow \{ \langle P' \parallel \Theta' \rangle \rightarrow \langle P'' \parallel \Theta'' \rangle \}, \\
\frac{\langle Q \parallel \Theta_{int} \subset \Theta \rangle \Rightarrow \langle \emptyset \parallel \Theta'_{int} \rangle}{\langle (P \not\Leftarrow Q) \parallel \Theta \rangle \Rightarrow \{ \langle P' \parallel \Theta' \rangle \rightarrow \\
\langle Q \parallel \Theta'_{int} \rangle \rightarrow \\
\langle P'' \parallel \Theta'' \cup \Theta'_{int} \rangle \}}
\end{array}
\quad (\text{Rule 33})$$

The rule of interrupt processes shows that the main environment Θ is protected when an interrupt occurs. However, the interrupt subroutine Q may affect Θ via global or shared variables and data structures after its completion.

Definition 41. The reduction rule for the time-driven process of RTPA, $@t_k \mathbf{TM} \mapsto_{P_k}$ in operational semantics is shown in Box 33.

The rule of time-driven processes models a top level system dispatching behavior where the system transfers control to a process P_k after capturing a corresponding timing event $@t_k \mathbf{TM}$; upon its completion, it returns the control of system resources and the environment to the system.

Definition 42. The reduction rule for the event-driven process of RTPA, $@e_k \mathbf{TM} \mapsto_{P_k}$ in operational semantics is shown in Box 34.

The rule of event-driven processes models the second type of top level system dispatching behaviors, where the system transfers control to a process P_k after capturing a corresponding event $@e_k \mathbf{S}$; upon its completion, it returns the control of system resources and the environment to the system.

Definition 43. The reduction rule for the interrupt-driven process of RTPA, $@int_k \odot \mapsto_{P_k}$ in operational semantics is shown in Box 35.

The rule of interrupt-driven processes models the third type of top level system dis-

Box 33.

$$\begin{array}{c}
k\mathbf{N}, n\mathbf{N}, t_k \mathbf{TM} \in \Theta, 1 \leq k\mathbf{N} \leq n\mathbf{N}, \\
\frac{\langle P \parallel \Theta \rangle \Rightarrow \prod_{k\mathbf{N}=1}^{n\mathbf{N}} \langle P_k \parallel \Theta' \rangle, \langle \S \parallel \Theta \rangle \Rightarrow \langle \S \parallel \Theta' \rangle}{\langle (@t_k \mathbf{TM} \mapsto P_k) \parallel \Theta \rangle \Rightarrow \{ \langle \S \parallel \Theta \rangle \rightarrow \langle P_k \parallel \Theta' \rangle \rightarrow \langle \S \parallel \Theta'' \rangle \}}
\end{array}
\quad (\text{Rule 34})$$

Box 34.

$$\begin{array}{c}
k\mathbf{N}, n\mathbf{N}, e_k \mathbf{S} \in \Theta, 1 \leq k\mathbf{N} \leq n\mathbf{N}, \\
\frac{\langle P \parallel \Theta \rangle \Rightarrow \prod_{k\mathbf{N}=1}^{n\mathbf{N}} \langle P_k \parallel \Theta' \rangle, \langle \S \parallel \Theta \rangle \Rightarrow \langle \S \parallel \Theta' \rangle}{\langle (@e_k \mathbf{S} \mapsto P_k) \parallel \Theta \rangle \Rightarrow \{ \langle \S \parallel \Theta \rangle \rightarrow \langle P_k \parallel \Theta \rangle \rightarrow \langle \S \parallel \Theta'' \rangle \}}
\end{array}
\quad (\text{Rule 35})$$

Box 35.

$$\begin{array}{c}
 k\mathbf{N}, n\mathbf{N}, \mathbf{Int}_k \odot \in \Theta, 1 \leq k \leq n\mathbf{N}, \\
 \hline
 \langle P \parallel \Theta_{\text{int}} \rangle \Rightarrow \prod_{k=1}^{n\mathbf{N}} \langle P_k \parallel \Theta'_{\text{int}} \rangle, \langle \S \parallel \Theta \rangle \Rightarrow \langle \S \parallel \Theta' \rangle \\
 \hline
 \langle (@\text{int}_k \odot \hookrightarrow P_k) \parallel \Theta \rangle \Rightarrow \{ \langle \S \parallel \Theta \rangle \rightarrow \langle P_k \parallel \Theta'_{\text{int}} \rangle \rightarrow \langle \S \parallel \Theta \cup \Theta'_{\text{int}} \rangle \} \\
 \text{(Rule 36)}
 \end{array}$$

patching behaviors, where the system transfers the control to an interrupt subroutine P_k after capturing a corresponding interrupt event $@\text{int}_k \odot$; upon its completion, it returns the control of system resources and the environment to the system.

CONCLUSION

The operational semantics of Real-Time Process Algebra (RTPA) has been developed in this article, which explains how syntactic constructs in RTPA can be reduced to values on an abstract reduction machine. The operational semantics of RTPA has provided a comprehensive paradigm of formal semantics, which extends the conventional express power of operational semantics to an entire set of semantic rules for complicated RTPA process structures and their algebraic operations. Especially, the operational semantics of the parallel, concurrent, and system dispatches have been formally and systematically elaborated.

RTPA has been presented as both a denotational mathematical structure and a system modeling methodology for describing the architectures and behaviors of real-time and nonreal-time software systems. The formal semantics of RTPA has helped to the comprehension and understanding of the RTPA syntactical and semantic rules as well as its expressive power in software engineering, cognitive informatics, and computational intelligence. RTPA has been used not only in software system specifications, but also in human and intelligent system modeling.

ACKNOWLEDGMENT

The authors would like to acknowledge the Natural Science and Engineering Council of Canada (NSERC) for its partial support to this work. We would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

- Aho, A.V., Sethi, R., & Ullman, J.D. (1985). *Compilers: Principles, techniques, and tools*. New York: Addison-Wesley Publication Co.
- Baeten, J.C.M. & Bergstra, J.A. (1991). Real time process algebra. *Formal Aspects of Computing*, 3, 142-188.
- Boucher, A. & Gerth, R. (1987). A timed model for extended communicating sequential processes. *Proceedings of ICALP'87*, Springer LNCS, 267.
- Cardelli, L. & Wegner, P. (1985). On understanding types, data abstraction and polymorphism. *ACM Computing Surveys*, 17(4), 471-522.
- Fecher, H. (2001). A real-time process algebra with open intervals and maximal progress. *Nordic Journal of Computing*, 8(3), 346-360.
- Higman, B. (1977). *A comparative study of programming languages*, 2nd ed. MacDonald.
- Hoare, C.A.R. (1978). Communicating sequential processes. *Communications of the ACM*, 21(8), 666-677.
- Hoare, C.A.R. (1985). *Communicating sequential processes*. London: Prentice-Hall International.

- Jones, C. B. (2003). Operational semantics: Concepts and their expression. *Information Processing Letters*, 88(1-2), 27 – 32.
- Louden K.C. (1993). *Programming languages: Principles and practice*. Boston: PWS-Kent Publishing Co.
- Martin-Lof, P. (1975). *An intuitionistic theory of types: Predicative part*. In H. Rose & J. C. Shepherdson (Eds.), *Logic Colloquium 1973*, North-Holland.
- McDermid, J. (Ed.) (1991). *Software engineer's reference book*. Oxford, UK: Butterworth Heinemann Ltd.
- Milner, R. (1980). *A calculus of communicating systems*, LNCS #92. Springer-Verlag.
- Milner, R. (1989). *Communication and concurrency*. Englewood Cliffs, NJ: Prentice-Hall
- Mitchell, J.C. (1990). Type systems for programming languages. In J. van Leeuwen (Ed.), *Handbook of theoretical computer science* (pp. 365-458). North Holland.
- Nicollin, X. & Sifakis, J. (1991). An overview and synthesis on timed process algebras. *Proceedings of the 3rd International Computer Aided Verification Conference*, pp. 376-398.
- Plotkin, G. (1981). A structural approach to operational semantics. *Technical Report DAIMI FN-19*, Aarhus University, Denmark.
- Schneider, S. (1995). An operational semantics for timed CSP. *Information and Computation*, 116(2), 193-213.
- Slonneger, K., & Barry, L.K. (1995). *Formal syntax and semantics of programming languages: A laboratory based approach*, (Chapter 8), Reading, MA: Addison-Wesley Publishing Company.
- Stubbs, D.F. & Webre, N.W. (1985). *Data structures with abstract data types and Pascal*. Monterey, CA: Brooks/Cole Publishing Co.
- Tan, X. & Wang, Y. (2008). A denotational semantics of real-time process algebra (RTPA). *The International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(3).
- Tan, X., Wang, Y., & Ngolah, C.F. (2006). Design and implementation of an automatic RTPA code generator. *Proceedings of the 19th Canadian Conference on Electrical and Computer Engineering (CCECE'06)*, Ottawa, ON, Canada, May, pp. 1605-1608.
- Wang, Y. (2002). The real-time process algebra (RTPA). *Annals of Software Engineering: An International Journal*, 14, 235-274.
- Wang, Y. (2003). Using process algebra to describe human and software system behaviors. *Brain and Mind*, 4(2), 199-213.
- Wang, Y. (2006a). On the informatics laws and deductive semantics of software. *IEEE Transactions on Systems, Man, and Cybernetics (C)*, 36(2), 161-171.
- Wang, Y. (2006b). Cognitive informatics and contemporary mathematics for knowledge representation and manipulation, invited plenary talk. *Proceedings of the 1st International Conference on Rough Set and Knowledge Technology (RSKT'06)*, Lecture Notes in Artificial Intelligence, LNAI #4062, Springer, Chongqing, China, July, pp. 69-78.
- Wang, Y. (2007a). *Software engineering foundations: A software science perspective*. New York: Auerbach Publications.
- Wang, Y. (2007b). The theoretical framework of cognitive informatics. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 1(1), 1-27.
- Wang, Y. (2007c). On theoretical foundations of software engineering and denotational mathematics, keynote speech. *Proceedings of the 5th Asian Workshop on Foundations of Software*, BHU Press, Xiamen, China, pp. 99-102.
- Wang, Y. (2008a). RTPA: A denotational mathematics for manipulating intelligent and computational behaviors. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(2), 44-62.
- Wang, Y. (2008b). Deductive semantics of RTPA. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(2), 95-121.
- Wang, Y. (2008c). On the Big-R notation for describing iterative and recursive behaviors. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(1), 17-28.
- Wang, Y. & King, G. (2000). *Software engineering processes: Principles and applications*, CRC series in software engineering, Vol. I. CRC Press.

Wilson, L.B. & Clark, R.G. (1988). *Comparative programming language*. Wokingham, UK: Addison-Wesley Publishing Co.

Woodcock, J. & Davies, J. (1996). *Using Z: Specification, refinement, and proof*. London: Prentice Hall International.

Winskel, G. (1993). *The formal semantics of programming languages*. MIT Press.

Yingxu Wang is professor of cognitive informatics and software engineering, director of the International Center for Cognitive Informatics (ICfCI), and director of Theoretical and Empirical Software Engineering Research Center (TESERC) at the University of Calgary. He received a PhD in software engineering from The Nottingham Trent University, UK, in 1997, and a BSc in Electrical Engineering from Shanghai Tiedao University in 1983. He was a visiting professor in the Computing Laboratory at Oxford University and Dept. of Computer Science at Stanford University during 1995 and 2008, respectively, and has been a full professor since 1994. He is founding editor-in-chief of International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), and editor-in-chief of CRC Book Series in Software Engineering. He has published over 300 journal and conference papers and 11 books in software engineering and cognitive informatics, and won dozens of research achievement, best paper, and teaching awards in the last 28 years, particularly the IBC 21st Century Award for Achievement "in recognition of outstanding contribution in the field of Cognitive Informatics and Software Science," and the 2007 groundbreaking book on Software Engineering Foundations: A Software Science Perspective.

Cyprian F. Ngolah holds a PhD in Software Engineering from the University of Calgary, Canada in 2006. He obtained a BSc degree in Mathematics and Computer Science from the University of Essex, England in 1988 and an MSc degree in Computer Control Systems from the University of Bradford, England in 1989. He has taught various computer sciences courses at both the undergraduate and graduate levels. He is currently a lecturer of Software Engineering at University of Buea, Republic of Cameroon. His main research interests are in real-time process algebra and its applications, tool support for formal specification languages, real-time software systems, formal methods in software engineering, and real-time operating systems.