

A Denotational Semantics of Real-Time Process Algebra (RTPA)

Xinming Tan, University of Calgary, Canada and Wuhan University of Technology, China

Yingxu Wang, University of Calgary, Canada

ABSTRACT

Real-time process algebra (RTPA) is a form of denotational mathematics for dealing with fundamental system behaviors such as timing, interrupt, concurrency, and event/time/interrupt-driven system dispatching. Because some key RTPA processes cannot be described adequately in conventional denotational semantic paradigms, a new framework for modeling time and processes is sought in order to represent RTPA in denotational semantics. Within this framework, time is modeled by the elapse of process execution. The process environment encompasses states of all variables represented as mathematical maps, which project variables to their corresponding values. Duration is introduced as a pair of time intervals and the environment to represent the changes of the process environment during a time interval. Temporal ordered durations and operations on them are used to denote process executions. On the basis of these means, a comprehensive set of denotational semantics for RTPA are systematically developed and formally expressed.

Keywords: cognitive informatics; deductive semantics; denotational mathematics; denotational semantics; formal methods; formal semantics; RTPA; real-time systems; software engineering

INTRODUCTION

Real-time process algebra (RTPA) is a form of denotational mathematics for the modeling, specification, and refinement of real-time and safety-critical systems (Wang, 2002, 2003, 2006a, 2006b, 2007a, 2008a-c), as well as human cognitive behaviors and processes (Wang, 2003, 2006b, 2007b). RTPA elicits a rich set of process operations including timing, interrupt, concurrency, and event/time/inter-

rupt-driven dispatches with a rigorous algebraic structure.

The conventional forms of denotational semantics were designed to deal with simple and sequential computation behaviors (Louden, 1993; McDermid, 1991; Wang, 2007a; Winskel, 1993). Efforts to represent parallel and concurrent behaviors in the denotational semantic approach focused on communications between components (Schneider, 2000). Various

treatments for time and durations have been introduced (Baeten & Bergstra, 1991; Boucher & Gerth, 1987; Corsetti, Montanari, & Ratto, 1991; Dierks, 2000; Fecher, 2001; Milner, 1980; Schneider, 1995) in order to describe the real-time mechanisms in denotation semantics. Because the mathematical structure of RTPA cannot be described adequately in conventional denotational semantics paradigms (Hoare, 1978, 1985; Schneider, 2000; Wang, 2007a; Winskel, 1993), a new extension on conventional denotational semantics is introduced for modeling time and processes in RTPA, where relative time is modeled by the elapse of process execution, and a process is modeled by temporal ordered duration sequences.

The article attempts to handle the sequential, parallel, concurrent, and real-time behaviors of RTPA in a coherent system. The abstract syntax of RTPA is presented. The system environment is introduced to describe instantaneous behaviors of processes and durations. Temporal ordered duration sequences are introduced to present the semantics of RTPA meta processes and process operations. The denotational semantics for RTPA can be used as rules for correctness checking and system verification in system design and modeling using RTPA. It also facilitates the rigorous understanding of a comprehensive set of fundamental computing system behaviors as modeled in RTPA.

THE ACTIVITY DURATION CALCULUS

The activity duration calculus (ADC) is developed to describe system behaviors over time where each activity happens in a timely order and will last for a period of time or an interval. At any specific time moment, the state of the system is represented by the values of its variables at the moment (Wang, 2006a, 2008b). The time sequence of the instantaneous moments expresses the behaviors of a system. ADC uses a semantic environment, which is a map from variables to values, to record an instantaneous system state (Wang, 2006a, 2008b). It uses a duration adopted as a 2-tuple of an interval and an environment to represent a system state for a

given period of time. A temporal ordered duration sequence is therefore a semantics model of system behaviors.

Variables and Values

ADC uses variables with a universal type by default. When different types of variables, such as integer, real, and Boolean, are needed, it can be considered that the various variables are in the form of set sum (Winskel, 1993), as used in the denotational semantics of RTPA.

\mathbb{X} - The set of all variables. (1)

\mathbb{V} - Domain of all possible values. (2)

When multiple variables are modeled, a subscript will be used, such as $x_1, \dots, x_n, x_i \in \mathbb{X}$. So do the values $v_i \in \mathbb{V}$.

Environments

An environment, which records the current states and values of all variables of a system, is a map (RAISE, 1992) from variables to values. A map is a table structure similar to a function except in domain definition, which maps values of one type into values of another type. The domain of all environments is defined as shown in Equation 3.

$$\mathbb{EN} = \mathbb{X} \xrightarrow{m} \mathbb{V} \quad (3)$$

It is noteworthy that \mathbb{EN} encompasses an inexplicit subset \mathbb{EN}' known as the system's environment that is independent from the user's environment $\mathbb{EN} \setminus \mathbb{EN}'$.

For a given expression $\rho : \mathbb{X} \xrightarrow{m} \mathbb{V}$, m is a map type and ρ is a map of $\mathbb{X} \xrightarrow{m} \mathbb{V}$. An empty map is denoted by $[\]$, which is the bottom of \mathbb{EN} . If $x \mapsto v$ belongs to ρ , $\rho(x)$ is called the application of ρ to x , written as $\rho(x) = v$. Suppose $\rho = [x_1 \mapsto 1, x_2 \mapsto 3, x_3 \mapsto 5]$, the domain of ρ , written as $dom \rho$, is $\{x_1, x_2, x_3\}$; the range of ρ , written as $rng \rho$, is $\{1, 3, 5\}$.

The following three operations are needed

for the discussion in the latter part of the article, where \mathbb{R}^+ is the set of non-negative real numbers.

- a. The *override operation* \dagger : It is a standard map operation (RAISE, 1992), puts mappings unique to the first argument and the whole mappings of the second argument together into the resulting environment.

$$\begin{aligned} \dagger : \mathbb{E}\mathbb{N} \times \mathbb{E}\mathbb{N} &\rightarrow \mathbb{E}\mathbb{N} \\ \rho_1 \dagger \rho_2 &= \\ [x \mapsto v \mid (x \in \text{dom } \rho_1 \wedge x \notin \text{dom } \rho_2 \wedge \\ &\rho_1(x) = v) \vee (x \in \text{dom } \rho_2 \wedge x \notin \text{dom} \\ &\rho_1 \wedge \rho_2(x) = v)] \end{aligned} \quad (4)$$

- a. The *advance operation* $+$: It increases all the time variables in an environment by a given number r , that is:

$$\begin{aligned} + : \mathbb{E}\mathbb{N} \times \mathbb{R}^+ &\rightarrow \mathbb{E}\mathbb{N} \\ \rho + r &= \rho \dagger [t \mapsto \rho(t) + r \mid t \in \text{dom } \rho] \end{aligned} \quad (5)$$

- c. The *retreat operation* $-$: It decreases all the time variables in an environment by a given number r . If the given number is greater than the least value holding by any timing variable in the environment, the result of the operation is the same as decrease by that least value.

$$\begin{aligned} - : \mathbb{E}\mathbb{N} \times \mathbb{R}^+ &\rightarrow \mathbb{E}\mathbb{N} \\ \rho - r &= \rho \dagger [t \mapsto \rho(t) - r \mid r' \in \mathbb{R}^+ \wedge t \in \\ &\text{dom } \rho \wedge \\ t_1 \in \text{dom } \rho \wedge \forall t_2 \in \text{dom } \rho \bullet \rho(t_2) \geq \rho(t_1) \\ &\wedge \\ (r < \rho(t_1) \wedge r' = r \vee r \geq \rho(t_1) \wedge r' = \rho(t_1))] \end{aligned} \quad (6)$$

Durations

An interval $[a,b)$, where $a, b \in \mathbb{R}^+$ and $b > a$, denotes a period. The set of all intervals is denoted by \mathbb{I} where $i \in \mathbb{I}$.

A duration is a pair of an interval and an environment, which records the activities happening during the interval.

$$\mathbb{D} = \mathbb{I} \times \mathbb{E}\mathbb{N} \quad (7)$$

For a duration $d = (i, \rho) = ([a,b), \rho)$ and $r \in \mathbb{R}^+$, the following operations can be defined:

Get the interval of duration:

$$\text{intv}(d) = i = [a,b);$$

Get the beginning of duration:

$$\text{begin}(d) = a;$$

Get the end of duration:

$$\text{end}(d) = b;$$

Get the environment of duration:

$$\text{env}(d) = \rho;$$

Duration advance:

$$\begin{aligned} \text{advance}(d, r) &= d + r = (i+r, \rho+r) \\ &= ([a+r, b+r), \rho+r); \end{aligned}$$

Duration retreat:

$$\begin{aligned} \text{retreat}(d, r) &= d - r = (i-r, \rho-r) \\ &= ([a-r, b-r), \rho-r), a \geq r; \\ &\text{or } ([0, b-a), \rho-a), a < r \end{aligned} \quad (8)$$

Temporal Ordered Duration Sequences

\mathbb{D}^* is used to denote the set of finite duration sequences. An n -duration sequence consisting of durations d_1, d_2, \dots, d_n is denoted as:

$$\begin{aligned} \langle d_1 \wedge d_2 \wedge \dots \wedge d_n \rangle \\ \text{or shortly as } \langle d_1, d_2, \dots, d_n \rangle \end{aligned} \quad (9)$$

\diamond is an empty sequence.

A duration sequence $c = \langle d_1, d_2, \dots, d_n \rangle$ is in temporal order iff $\text{end}(d_k) = \text{begin}(d_{k+1})$ for all $k = 1, \dots, n-1$. The set of all temporal ordered duration sequences is denoted as:

$$\mathbb{C} \subseteq \mathbb{D}^* \quad (10)$$

The attributes of a temporal ordered duration sequence $c = \langle d_1, d_2, \dots, d_n \rangle \in \mathbb{C}$ can be defined as follows:

$$\begin{aligned} \text{head}(c) &= d_1; \\ \text{tail}(c) &= \langle d_2, \dots, d_n \rangle; \\ \text{front}(c) &= \langle d_1, d_2, \dots, d_{n-1} \rangle; \\ \text{back}(c) &= d_n; \end{aligned}$$

$$\begin{aligned} first(c) &= begin(d_1); \\ last(c) &= end(d_n). \end{aligned} \quad (11)$$

When $c = \diamond$, all of the above operations are unspecified with the value \perp .

Sequence Operations

The following operations on \mathbb{C} are introduced:

- a. The *advance operation* $+$: It moves a temporal ordered duration sequence forward with a given number of time units.

$$\begin{aligned} + : \mathbb{C} \times \mathbb{R}^+ &\rightarrow \mathbb{C} \\ \diamond + r &= \diamond \\ \langle d \rangle \wedge c + r &= \langle d + r \rangle \wedge (c + r) \end{aligned} \quad (12)$$

- b. The *retreat operation* $-$: It moves a temporal ordered duration sequence backward with a given number of time units.

$$\begin{aligned} - : \mathbb{C} \times \mathbb{R}^+ &\rightarrow \mathbb{C} \\ \diamond - r &= \diamond \\ \langle d \rangle \wedge c - r &= \langle d - r \rangle \wedge (c - r) \end{aligned} \quad (13)$$

- c. The *truncate-right operation* \triangleleft : It cuts off the right part of a temporal ordered duration sequence before a given time point.

$$\begin{aligned} \triangleleft : \mathbb{C} \times \mathbb{R}^+ &\rightarrow \mathbb{C} \\ \diamond \triangleleft r &= \diamond \\ c \triangleleft r &= c \\ first(c) &\geq r \\ c \triangleleft r &= c \\ last(c) &\leq r \\ c_1 \wedge c_2 \triangleleft r &= c_1 \\ last(c_1) &= r \end{aligned} \quad (14)$$

- d. The *truncate-left operation* \triangleright : It cuts off the left part of a temporal ordered duration sequence before a given time point.

$$\begin{aligned} \triangleright : \mathbb{C} \times \mathbb{R}^+ &\rightarrow \mathbb{C} \\ \diamond \triangleright r &= \diamond \\ c \triangleright r &= \diamond \\ last(c) &\leq r \\ c \triangleright r &= c \\ first(c) &\geq r \\ c_1 \wedge c_2 \triangleright r &= c_2 \\ first(c_2) &= r \end{aligned} \quad (15)$$

- e. The *follow operation* \rightsquigarrow : It puts two temporal ordered duration sequences in a temporal order.

$$\begin{aligned} \rightsquigarrow : \mathbb{C} \times \mathbb{C} &\rightarrow \mathbb{C} \\ \diamond \rightsquigarrow c &= c \\ \langle d \rangle \wedge c \rightsquigarrow \diamond &= \langle d \rangle \wedge c \\ c_1 \wedge \langle d_1 \rangle \rightsquigarrow \langle d_2 \rangle \wedge c_2 &= \\ c_1 \wedge \langle d_1 \rangle \wedge (\langle d_2 \rangle \wedge c_2 + end(d_1) - begin(d_2)) \end{aligned} \quad (16)$$

The following sections present the denotational semantics of RTPA using ADC, which encompasses the abstract syntax of RTPA, the semantic domains of RTPA, and the semantic functions of RTPA.

THE ABSTRACT SYNTAX OF RTPA

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

$$RTPA \triangleq (\mathfrak{T}, \mathfrak{P}, \mathfrak{R}) \quad (17)$$

where \mathfrak{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, and \mathfrak{R} a set of 17 relational process operations for constructing complex system behaviors.

Further details of \mathfrak{T} , \mathfrak{P} , and \mathfrak{R} in RTPA may be referred to (Wang, 2002, 2007a, 2008a). The

abstract syntax of RTPA adopts three syntactic categories, namely variables, expressions, and processes.

Variables of RTPA

RTPA uses identifiers with a type suffix to declare and denote a variable. In the definition of the abstract syntax of RTPA, the following conventions are adopted:

1. When the type of a variable is not a concern, x is used to represent the variable;
2. $@e$ is used for event variable, ptr for pointer variable, t for time variable, and id for name (string) variable; that is:

$$x, @e, ptr, t, id \in \mathbb{X}. \quad (18)$$

For a pointer variable ptr , which takes value of memory addresses, an associated variable $addr$ is attached to denote the object to which the ptr points.

3. Integer subscripsts are added when multiple variables are discussed.

Expressions of RTPA

All kinds of typical arithmetical, Boolean, and relational expressions are a part of RTPA expressions. Special expressions of RTPA are explicitly defined below:

$$e \triangleq \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T} \mid \text{MEM}(\perp)\mathbb{T} \mid \text{PORT}(\text{ptr}\mathbf{P})\mathbb{T} \mid \mathbb{T}\mathbf{TM} \mid \mathbb{T}\mathbf{TM} + \Delta n\mathbf{N} \quad (19)$$

where $\mathbb{E}\mathbb{X}\mathbb{P}$ is the set of all expressions, $e \in \mathbb{E}\mathbb{X}\mathbb{P}$, \mathbb{T} is a type suffix, MEM indicates a memory location or area, PORT indicates an I/O port, and \mathbb{T} stands for the system time in RTPA.

A detailed discussion on formal syntaxes of RTPA is provided in (Tan, 2006; Tan, Wang, & Ngolah, 2004; Wang, 2002, 2007a, 2008a).

Processes of RTPA

The entire process system of RTPA, \mathbb{P} , models 17 meta processes \mathfrak{P} and 17 process relations \mathfrak{R}

as defined in Equations 20 and 21, respectively (Wang, 2002, 2007a, 2008a).

$$\begin{aligned} \mathfrak{P} \triangleq & x\mathbb{T} := e\mathbb{T} \mid \diamond e\mathbf{BL} \mid id\mathbf{S} \Rightarrow \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T} \mid \\ & id\mathbf{S} \Leftarrow \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T} \mid id\mathbf{S} \Leftarrow \text{MEM}(\perp)\mathbb{T} \mid \\ & \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T} \triangleright x\mathbb{T} \mid x\mathbb{T} \Leftarrow \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T} \mid \\ & \text{PORT}(\text{ptr}\mathbf{P})\mathbb{T} \triangleright x\mathbb{T} \mid x\mathbb{T} \Leftarrow \text{PORT}(\text{ptr}\mathbf{P})\mathbb{T} \mid \\ & @t\mathbf{TM} @ \mathbb{T} \mid @t\mathbf{TM} \triangleq \mathbb{T}\mathbf{TM} + \Delta n\mathbf{N} \mid \\ & (n\mathbf{N}) \mid \downarrow (n\mathbf{N}) \mid !(@e\mathbf{S}) \mid \otimes \mid \boxtimes \mid \mathbb{T}(\text{id}\mathbf{S}) \end{aligned} \quad (20)$$

$$\begin{aligned} \mathfrak{R} \triangleq & P \rightarrow Q \mid (\diamond e\mathbf{BL} = \mathbf{T} \rightarrow P \mid \sim \rightarrow Q) \mid \\ & (\diamond e\mathbf{N} = 1 \rightarrow P1 \mid 2 \rightarrow P2 \mid \dots \mid \sim \rightarrow Q) \mid \\ & P \rightarrow Q \mid P \circ P \mid P \parallel Q \mid P \boxplus Q \mid P \parallel\parallel Q \mid P \gg Q \mid \\ & @t\mathbf{TM} \hookrightarrow P \mid @e\mathbf{S} \hookrightarrow P \mid @i \odot \hookrightarrow P \mid P \downarrow Q \mid \\ & P \rightsquigarrow Q \mid \\ & \underset{e\mathbf{BL}=\mathbf{T}}{R} (P) \mid \underset{i\mathbf{N}=1}{R} (P(i)) \mid P \rightarrow \underset{e\mathbf{BL}=\mathbf{T}}{R} (P) \end{aligned} \quad (21)$$

where \mathbb{P} is the set of all RTPA processes, and $\mathfrak{P}, \mathfrak{R} \in \mathbb{P}$.

THE SEMANTIC DOMAINS OF RTPA

Apart from the three syntactic domains, five semantic domains, namely \mathbb{V} - values, $\mathbb{E}\mathbb{N}$ - environments, \mathbb{D} - durations, \mathbb{C} - temporal ordered duration sequences, and COM - prefix closure of temporal ordered duration sequences, are adopted in this section. Some other discrete domains, such as Boolean values and natural numbers, are also needed in the discussions.

The least observable interval used in defining the semantics of a process is a unit interval $[a, b)$ such that $b - a = 1$ (time unit), which corresponds to the relative system clock model, $\mathbb{T}\mathbf{N}$, of RTPA. Therefore, the set of non-negative real numbers \mathbb{R}^+ used in ADC may be replaced by the natural numbers \mathbb{N} when giving a denotational semantics for RTPA.

Variables and Values in RTPA

A specific value domain is needed for each type of variables, which is called the logical value domain in order to distinguish from the domain \mathbb{V} . Typical value domains are as follows:

\mathbb{B} - Boolean values, $\mathbb{B} = \{true, false, \perp\}$.
 \mathbb{N} - Natural numbers with bottom \perp .
 \mathbb{R} - Real numbers with bottom \perp .
 \mathbb{E} - Values for event variables, $\mathbb{E} = \{event, reset, \perp\}$.
 \mathbb{PT} - Values for pointer variables.
 \mathbb{O} - Values for pointer associated variables.
 \mathbb{V} - Domain of all possible values of all variables.
 $\mathbb{V} = \mathbb{B} + \mathbb{N} + \mathbb{R} + \mathbb{E} + \mathbb{PT} + \mathbb{O}$

(22)

Therefore, the domain \mathbb{V} is a sum of different logical domains (Winskel, 1993). It easily can be understood from the context which value domains are in use. Therefore, we do not need to explicitly present the values in \mathbb{V} in domain sum format as a pair of an index in the sum and a value of the corresponding value domains.

Process Executions in RTPA

A temporal ordered duration sequence can be considered as a denotation of one possible execution path of a process, or a specific computation. All the specified computations performed by a process describe its behaviors as the semantics of the process.

A set $\mathbb{PC} \subseteq \mathbb{C}$ is called a prefix closure if it satisfies:

1. $\diamond \in \mathbb{PC}$, and
2. $c_1 \wedge c_2 \in \mathbb{PC} \Rightarrow c_1 \in \mathbb{PC}$.

(23)

\mathbb{COM} denotes all prefix closures with set inclusion relation \subseteq , which has the bottom \diamond and close at operation \cup .

The semantics of both expressions and processes can be described by three semantic functions, which specify what values in the domain \mathbb{V} serve as the denotations for expressions and which objects in the domain \mathbb{COM} serve as the denotations for processes. The semantic function for Boolean expressions is treated separately from the general expression evaluation function, because it is used quite often in the semantic definitions.

The Boolean expression evaluation function \mathcal{B} evaluates logical expressions to Boolean values.

$$\mathcal{B} : \mathbb{EXP} \rightarrow \mathbb{EN} \rightarrow \mathbb{B} \quad (24)$$

The general expression evaluation function \mathcal{E} evaluates all kinds of expressions to its values in \mathbb{V} , that is:

$$\mathcal{E} : \mathbb{EXP} \rightarrow \mathbb{EN} \rightarrow \mathbb{V}$$

$$\mathcal{E}[\mathbb{xT}](\rho) = \rho(\mathbb{x});$$

$\mathcal{E}[\mathbb{e}](\rho)$ follows the traditional convention if \mathbb{e} is regular expression, such as **arithmetic, Boolean, relational expression, etc;**

$\mathcal{E}[\mathbb{MEM}(\text{ptr}\mathbb{P})\mathbb{T}](\rho)$ gives an object of type \mathbb{T} located at the \mathbb{MEM} position where a ptr points to;

$\mathcal{E}[\mathbb{PORT}(\text{ptr}\mathbb{P})\mathbb{T}](\rho)$ gives an object of type \mathbb{T} located at the \mathbb{PORT} pointed to by ptr ;

$\mathcal{E}[\mathbb{\$tTM}](\rho) = \rho(\mathbb{\$t})$, the system time is determined by durations, for a given **duration**

$d, \rho(\mathbb{\$t}) = \text{begin}(d)$, which will be discussed in the following section;

$$\mathcal{E}[\mathbb{\$TMM} + \Delta n\mathbb{N}](\rho) = \rho(\mathbb{\$t}) + \rho(n). \quad (25)$$

THE DENOTATIONAL SEMANTIC FUNCTIONS OF RTPA META PROCESSES

This section presents the denotational semantic functions of the 17 RTPA meta processes.

A process function describes what the behaviors are when a process runs under a given environment, that is:

$$\mathcal{C} : \mathbb{P} \rightarrow \mathbb{EN} \rightarrow \mathbb{COM} \quad (26)$$

where $\mathbb{P} \rightarrow \mathbb{EN} \rightarrow \mathbb{COM}$ is a function space as proposed in (Winskel, 1993). The operator \rightarrow is right-associative, which means $\mathbb{P} \rightarrow \mathbb{EN} \rightarrow \mathbb{COM}$ is an abbreviation of $\mathbb{P} \rightarrow (\mathbb{EN} \rightarrow \mathbb{COM})$. Therefore, \mathcal{C} is a high order function,

which returns a function of type $\mathbb{EN} \rightarrow \text{COM}$ as the result.

The semantic functions of RTPA processes are defined deductively on different process syntactic formations below:

1. **The assignment process:** The semantic function of the assignment process in RTPA is specified as follows.

$$\mathcal{C}[\![x\mathbb{T}:=e\mathbb{T}]\!] (\rho) \triangleq \{ \langle (i, \rho^\dagger[x \mapsto \mathcal{E}[e]](\rho)) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(27)

The assignment process changes the value of the left-side variable of the assignment.

2. **The Evaluation Process:** The semantic function of the evaluation process in RTPA is specified as follows.

$$\mathcal{C}[\![\blacklozenge e\mathbf{BL}]\!] (\rho) \triangleq \{ \langle (i, \rho^\dagger[r \mapsto \mathcal{E}[e\mathbf{BL}]](\rho)) \rangle \mid r \in \mathbb{EN}' \wedge \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(28)

where \mathbb{EN}' is the system's environment, and the evaluation result r is set to true or false based on $\mathcal{E}[e\mathbf{BL}]](\rho)$.

3. **The addressing process:** The semantic function of the addressing process in RTPA is specified as follows.

$$\mathcal{C}[\![id\mathbf{S} \Rightarrow \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]\!] (\rho) \triangleq \{ \langle (i, \rho^\dagger[id \mapsto \mathcal{E}[\text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]](\rho)) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(29)

The addressing process lets id associate to the object of type \mathbb{T} located at the MEM position where ptr points to.

4. **The memory allocation process:** The semantic function of the memory allocation process in RTPA is specified as follows:

$$\mathcal{C}[\![id\mathbf{S} \Leftarrow \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]\!] (\rho) \triangleq \{ \langle (i, \rho_2) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+1] \wedge$$

$$\begin{aligned} \rho_1 &= \rho^\dagger[\text{ptr} \mapsto \mathcal{E}[\text{MEM}()\mathbb{T}]](\rho) \wedge \\ \rho_2 &= \rho_1^\dagger[id \mapsto \mathcal{E}[\text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]](\rho_1) \} \end{aligned}$$
(30)

The memory allocation process allocates memory for an object of type \mathbb{T} identified by id with the address of ptr .

5. **The memory release process:** The semantic function of the memory release process in RTPA is specified as follows.

$$\mathcal{C}[\![id\mathbf{S} \Leftarrow \text{MEM}(\perp)\mathbb{T}]\!] (\rho) \triangleq \{ \langle (i, \rho^\dagger[\text{ptr} \mapsto \perp, id \mapsto \perp]) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(31)

The memory release process releases the memory for the object identified by id , and as a result, id points to \perp .

6. **The Read Process:** The semantic function of the read process in RTPA is specified as follows:

$$\mathcal{C}[\![\text{MEM}(\text{ptr}\mathbf{P})\mathbb{T} \triangleright x\mathbb{T}]\!] (\rho) \triangleq \{ \langle (i, \rho^\dagger[x \mapsto \mathcal{E}[\text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]](\rho)) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(32)

The read process reads a object prescribed by $\mathcal{E}[\text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]](\rho)$ into variable x .

7. **The write process:** The semantic function of the write process in RTPA is specified as follows:

$$\mathcal{C}[\![x\mathbb{T} \Leftarrow \text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]\!] (\rho) \triangleq \{ \langle (i, \rho^\dagger[\text{addr} \mapsto \rho(x)]) \rangle \mid \text{addr} \in \mathbb{EN}' \wedge \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(33)

The write process writes the value of x into the object prescribed by $\text{addr} = \mathcal{E}[\text{MEM}(\text{ptr}\mathbf{P})\mathbb{T}]](\rho)$.

8. **The input process:** The semantic function of the input process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\text{PORT}(\text{ptr}\mathbf{P})\mathbb{T}]_{>x\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [x \mapsto \mathcal{E}[\text{PORT}(\text{ptr}\mathbf{P})\mathbb{T}](\rho)]) \rangle \mid \\ & \exists n \in \mathbb{N} \bullet i = [n, n+1] \} \end{aligned} \quad (34)$$

The input process reads a object prescribed by $\mathcal{E}[\text{PORT}(\text{ptr}\mathbf{P})\mathbb{T}](\rho)$ into variable x .

9. **The Output Process:** The semantic function of the output process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbb{x}\mathbb{T}]_{<\text{PORT}(\text{ptr}\mathbf{P})\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [\text{addr} \\ & \mapsto \rho(x)]) \rangle \mid \\ & \text{addr} \in \mathbb{E}\mathbb{N}' \wedge \exists n \in \mathbb{N} \bullet i = [n, \\ & n+1] \} \end{aligned} \quad (35)$$

The output process outputs the value of x to the object prescribed by $\text{addr} = \mathcal{E}[\text{PORT}(\text{ptr}\mathbf{P})\mathbb{T}](\rho)$.

10. **The timing process:** The semantic function of the timing process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\text{@t}\mathbf{TM} \text{ @ } \mathbb{T}]_{\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [t \mapsto \\ & \mathcal{E}[\mathbb{T}](\rho)]) \rangle \mid \\ & \exists n \in \mathbb{N} \bullet i = [n, n+1] \} \end{aligned} \quad (36)$$

The timing process sets the value of variable t to the system time.

11. **The duration process:** The semantic function of the duration process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\text{@t}\mathbf{TM} \triangleq \mathbb{T} + \Delta n \mathbf{N}]_{\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [t \mapsto \\ & \mathcal{E}[\mathbb{T} + \Delta n \mathbf{N}](\rho)]) \rangle \mid \exists m \in \mathbb{N} \bullet i = \\ & [m, m+\rho(n)] \} \end{aligned} \quad (37)$$

The system time is specified by the intervals in the temporal ordered duration sequences. The value of t is the start of the interval plus the increment. Hence, for a duration $\langle [a, b), \rho \dagger [t \mapsto \mathcal{E}[\mathbb{T} + \Delta n \mathbf{N}](\rho)] \rangle$, the system time \mathbb{T} is the beginning of the interval and t will be eventually evaluated to $a + \rho(n) = \text{begin}([a, b)) + \rho(n)$.

12. **The increase process:** The semantic function of the increase process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbb{n}\mathbf{N}]_{\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [n \mapsto \mathcal{E}[\mathbb{n}\mathbf{N}](\rho)]) \rangle \mid \\ & \exists m \in \mathbb{N} \bullet i = [m, m+1] \} \end{aligned} \quad (38)$$

The increase process adds one to the value of variable n .

13. **The decrease process:** The semantic function of the decrease process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbb{d}\mathbf{N}]_{\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [n \mapsto \mathcal{E}[\mathbb{d}\mathbf{N}](\rho)]) \rangle \mid \\ & \exists m \in \mathbb{N} \bullet i = [m, m+1] \} \end{aligned} \quad (39)$$

The decrease process subtracts one from the value of variable n .

14. **The exception detection process:** The semantic function of the exception detection process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\text{!}(\text{@e}\mathbf{S})]_{\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [\text{addr} \mapsto \rho(e)]) \rangle \mid \\ & \text{addr} \in \mathbb{E}\mathbb{N}' \wedge \exists n \in \mathbb{N} \bullet i = [n, n+1] \} \end{aligned} \quad (40)$$

The exception detection process is a special output process where its contents are warning messages in string type and its port is the standard system device such as CRT or printer.

15. **The skip process:** The semantic function of the skip process in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\text{⊗}]_{\mathbb{T}}(\rho) \triangleq & \{ \langle (i, \rho \dagger [r \mapsto \mathcal{E}[\text{⊗}](\rho)]) \rangle \mid \\ & r \in \mathbb{E}\mathbb{N}' \wedge \exists n \in \mathbb{N} \bullet i = [n, n+1] \} \end{aligned} \quad (41)$$

where the skip process results in a new executing address r , which replaces the usual system dispatching mechanism, that is, $r = r+1$.

16. **The stop process:** The semantic function of the stop process in RTPA is specified as follows:

$$\mathcal{C}[\boxtimes](\rho) \triangleq \bigcup_{k=1}^{\infty} \{ \langle (i, \rho) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+k] \}$$
(42)

It indicates that no activities can be observed from the stop process, but the semantic environment keeps the same as that of the duration before the stop process.

17. **The system process:** The semantic function of the system process in RTPA is specified as follows:

$$\mathcal{C}[\$(idS)](\rho) \triangleq \{ \langle (i, \rho \dagger [id \mapsto \perp]) \rangle \mid \exists n \in \mathbb{N} \bullet i = [n, n+1] \}$$
(43)

The system process returns the control back to the host system, such as an operating system, so the application identified by an id is released.

THE DENOTATIONAL SEMANTIC FUNCTIONS OF RTPA PROCESS RELATIONS

This section presents the denotational semantic functions of the 17 RTPA process relations.

1. **The sequential process:** The semantic function of the sequential process relation in RTPA is specified as follows:

$$\mathcal{C}[P \rightarrow Q](\rho) \triangleq \{ c_1 \rightsquigarrow c_2 \mid c_1 \in \mathcal{C}[P](\rho) \wedge c_2 \in \mathcal{C}[Q](env(back(c_1))) \}$$
(44)

It is noteworthy that the system time accumulates as temporal ordered duration sequences extend. By introducing the follow operator, as shown in Equation 45, the sequential process relation can be defined as shown in Equation 46.

$$\begin{aligned} & ; : (\mathbb{E}N \rightarrow \text{COM})^2 \rightarrow \mathbb{E}N \rightarrow \text{COM} \\ (f;g)(\rho) & = \{ c_1 \rightsquigarrow c_2 \mid c_1 \in f(\rho) \wedge c_2 \in g(env(back(c_1))) \} \end{aligned}$$
(45)

where $(\mathbb{E}N \rightarrow \text{COM})^2$ is an abbreviation of $(\mathbb{E}N \rightarrow \text{COM}) \times (\mathbb{E}N \rightarrow \text{COM})$.

$$\mathcal{C}[P \rightarrow Q](\rho) \triangleq (\mathcal{C}[P];\mathcal{C}[Q])(\rho)$$
(46)

2. **The jump process:** The semantic function of the jump process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[P \curvearrowright Q](\rho) & \triangleq \{ c \mid \exists n \in \mathbb{N} \bullet \\ & (c \triangleleft n) \in \mathcal{C}[P](\rho) \wedge (c \triangleright n) \in \\ & \mathcal{C}[Q](env(back(c \triangleleft n))) \} \end{aligned}$$
(47)

The definition shows that the execution control transfers from process P to process Q.

3. **The branch process:** In order to define the semantics of the branch process, the condition operator is introduced first.

$$\begin{aligned} cond : (\mathbb{E}N \rightarrow \mathbb{B}) \times (\mathbb{E}N \rightarrow \text{COM})^2 & \rightarrow \\ & \mathbb{E}N \rightarrow \text{COM} \\ cond(b, f, g)(\rho) & = f(\rho), \\ & b(\rho) = \text{true}; \\ cond(b, f, g)(\rho) & = g(\rho), \\ & b(\rho) = \text{false}; \\ cond(b, f, g)(\rho) & = \{ \diamond \}, \\ & b(\rho) = \perp. \end{aligned}$$
(48)

The semantic function of the branch process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\blacklozenge e \mathbf{BL} = \mathbf{T} \rightarrow P \mid \blacklozenge \sim \rightarrow Q](\rho) & \triangleq \\ cond(\mathcal{B}[e], \mathcal{C}[P], \mathcal{C}[Q])(\rho) \end{aligned}$$
(49)

4. **The switch process:** The semantic function of the switch process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\diamond e \mathbf{N}=1 \rightarrow P_1 | 2 \rightarrow P_2 | \dots | \sim \rightarrow Q](\rho) \triangleq \\ \mathcal{C}[Q](\rho) \cup \bigcup_{k=1}^n \text{cond}(\mathcal{B}[e=k], \mathcal{C}[P_k]), \\ \mathcal{C}[\emptyset](\rho) \end{aligned} \quad (50)$$

The definition shows that if expression e is evaluated to be k , process P_k will be executed; otherwise process Q will be executed as default.

5. **The while-loop process:** With the condition and follow operators, the semantic function of the while-loop process relation in RTPA can be described recursively as follows:

$$\begin{aligned} \mathcal{C}[\overset{F}{\mathbf{R}}(P)](\rho) \triangleq \text{cond}(\mathcal{B}[e], \mathcal{C}[P]; \mathcal{C}[\overset{F}{\mathbf{R}} \\ (P)], \mathcal{C}[\emptyset](\rho)) \end{aligned} \quad (51)$$

Since it is a recursive definition, the least fixed point operator fix is used to define the semantics (Winskel, 1993).

$$\begin{aligned} W : (\mathbb{EN} \rightarrow \text{COM}) \rightarrow \mathbb{EN} \rightarrow \text{COM} \\ W(\theta) = \text{cond}(\mathcal{B}[e], \mathcal{C}[P]; \theta, \mathcal{C}[\emptyset]) \\ \mathcal{C}[\overset{F}{\mathbf{R}}(P)](\rho) \triangleq \text{fix}(W)(\rho) \end{aligned} \quad (52)$$

The use of the least fixed point operator can be justified by the fact that all functions over $\mathbb{EN} \rightarrow \text{COM}$ defined here are continuous functions.

6. **The repeat-loop process:** The semantic function of the repeat-loop process relation in RTPA is specified as follows.

$$\mathcal{C}[P \rightarrow \overset{F}{\mathbf{R}}(P)](\rho) \triangleq (\mathcal{C}[P]; \mathcal{C}[\overset{F}{\mathbf{R}}(P)])(\rho) \quad (53)$$

It shows that the loop of process P will be carried out at least once.

7. **The for-loop process:** The semantic function of the for-loop process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\overset{n}{\mathbf{R}}(P)](\rho) \triangleq \text{fix}(W)(\rho) \\ W : (\mathbb{EN} \rightarrow \text{COM}) \rightarrow \mathbb{EN} \rightarrow \text{COM} \\ W(\theta) = \text{cond}(\mathcal{B}[i \leq n], \mathcal{C}[P]; \theta, \mathcal{C}[\emptyset]) \end{aligned} \quad (54)$$

The for-loop process is similar to the while-loop process, only using a more specific condition $i \leq n$.

8. **The recursion process:** The semantic function of the recursion process relation in RTPA is specified as follows.

$$\begin{aligned} \mathcal{C}[P \circ P](\rho) \triangleq \text{fix}(W)(\rho) \\ W : (\mathbb{EN} \rightarrow \text{COM}) \rightarrow \mathbb{EN} \rightarrow \text{COM} \\ W(\theta) = \mathcal{C}[P]; \theta \end{aligned} \quad (55)$$

Similarly, the technique for defining recursion is used.

9. **The function call process:** The semantic function of the function call process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[P \rightarrow Q](\rho) \triangleq \{ c \mid \exists n_1, n_2 \in \mathbb{N} \bullet \\ n_1 < n_2 \wedge (c \triangleleft n_1) \in \mathcal{C}[P](\rho) \wedge \\ ((c \triangleleft n_2) \triangleright n_1) \in \mathcal{C}[Q](\text{env}(\text{back}(c \\ \triangleleft n_1))) \wedge \\ (c \triangleright n_2) \in \mathcal{C}[P](\text{env}(\text{back}((c \triangleleft n_2) \\ \triangleright n_1))) \} \end{aligned} \quad (56)$$

The definition shows the execution control will return to process P after process Q is completed.

10. **The parallel process:** The semantic function of the parallel process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbf{P} \parallel \mathbf{Q}](\rho) \triangleq & \{c \mid \exists n \in \mathbb{N} \bullet c \triangleleft n \in \\ & \mathcal{C}[\mathbf{P}](\rho) \wedge \\ & c \in \mathcal{C}[\mathbf{Q}](\rho) \vee c \in \mathcal{C}[\mathbf{P}](\rho) \wedge c \triangleleft n \\ & \in \mathcal{C}[\mathbf{Q}](\rho)\} \end{aligned} \quad (57)$$

The parallel process models the single-clock multi-processor (SCMP) mechanism for the semantic environment.

11. **The concurrent process:** The semantic function of the concurrent process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbf{P} \parallel \mathbf{Q}](\rho) \triangleq & \{c \mid \exists n \in \mathbb{N} \bullet c \triangleleft n \in \\ & \mathcal{C}[\mathbf{P}](\rho) \wedge \\ & c \in \mathcal{C}[\mathbf{Q}](\rho) \vee c \in \mathcal{C}[\mathbf{P}](\rho) \wedge c \triangleleft n \\ & \in \mathcal{C}[\mathbf{Q}](\rho)\} \end{aligned} \quad (58)$$

The concurrent process models the multi-clock multi-processor (MCMP) mechanism for the semantic environment.

12. **The interleave process:** The semantic function of the interleave process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbf{P} \parallel \mathbf{Q}](\rho) \triangleq & \{c \mid \exists n \in \mathbb{N} \bullet c \triangleleft n \in \\ & \mathcal{C}[\mathbf{P}](\rho) \wedge \\ & c \in \mathcal{C}[\mathbf{Q}](\rho) \vee c \in \mathcal{C}[\mathbf{P}](\rho) \wedge c \triangleleft n \\ & \in \mathcal{C}[\mathbf{Q}](\rho)\} \end{aligned} \quad (59)$$

The parallel process models the single-clock single-processor (SCSP) mechanism for the semantic environment.

13. **The pipeline process:** The semantic function of the pipeline process relation in RTPA is specified as follows:

$$\begin{aligned} \mathcal{C}[\mathbf{P} \gg \mathbf{Q}](\rho) \triangleq & \{c \mid \exists n \in \mathbb{N} \bullet c \triangleleft n \in \\ & \mathcal{C}[\mathbf{P}](\rho) \wedge \\ & c \in \mathcal{C}[\mathbf{Q}](\rho) \vee c \in \mathcal{C}[\mathbf{P}](\rho) \wedge c \triangleleft n \\ & \in \mathcal{C}[\mathbf{Q}](\rho)\} \end{aligned} \quad (60)$$

In the pipeline process, process P and Q run concurrently while the outputs of process P should be the inputs of process

Q. However, the input/output relation can be checked statically.

14. **The interrupt process:** The semantic function of the interrupt process relation in RTPA is specified as follows.

$$\begin{aligned} \mathcal{C}[\mathbf{P} \downarrow \mathbf{Q}](\rho) \triangleq & \{c \mid \exists n_1, n_2 \in \mathbb{N} \bullet n_1 \leq \\ & n_2 \wedge \\ & (c \triangleleft n_1) \rightsquigarrow (c \triangleright n_2) \in \mathcal{C}[\mathbf{P}](\rho) \wedge ((c \\ & \triangleleft n_2) \triangleright n_1) - \\ & n_1 \in \mathcal{C}[\mathbf{Q}](env(back(c \triangleleft n_1)))\} \end{aligned} \quad (61)$$

The above definition shows that process P is exempted by process Q at the time point n1 and resumes execution after Q ends at the time point n2.

15. **The time-driven dispatch process:** The semantic function of the time-driven process relation in RTPA is specified as follows.

$$\begin{aligned} \mathcal{C}[\mathbf{@t} \mathbf{TM} \downarrow \mathbf{P}](\rho) \triangleq & \{c_1 \rightsquigarrow c_2 \mid \exists n \in \mathbb{N} \bullet \\ & c_1 \in \bigcup_{k=1}^{\rho(t)} \{<[n, n+k], \rho>\} \wedge \\ & c_2 \in \mathcal{C}[\mathbf{P}](env(back(c_1)))\} \end{aligned} \quad (62)$$

The above definition shows that the time-driven process will wait, which means keep the semantic environment unchanged, until the timeout to perform process P. Then, multi-processes wait on different timeouts becomes:

$$\begin{aligned} \mathcal{C}[\mathbf{@t}_1 \mathbf{TM} \downarrow \mathbf{P}_1 \mid \mathbf{@t}_2 \mathbf{TM} \downarrow \mathbf{P}_2 \mid \dots \mid \mathbf{@t}_n \mathbf{TM} \downarrow \mathbf{P}_n](\rho) \\ = \bigcup_{k=1}^n \mathcal{C}[\mathbf{@t}_k \mathbf{TM} \downarrow \mathbf{P}_k](\rho) \end{aligned} \quad (63)$$

16. **The event-driven dispatch process:** The semantic function of the event-driven process relation in RTPA is specified as follows:

$$\mathcal{C}[\mathbf{@eS} \downarrow \mathbf{P}](\rho) \triangleq (\mathcal{C}[\mathbf{@eS}]; \mathcal{C}[\mathbf{P}])(\rho) \quad (64)$$

The above definition indicates that, after the occurrence of event @e captured by the system, P is executed. Then, the waiting on multiple events becomes:

$$\mathcal{C}[\text{@}e_1 \mathbf{S} \mapsto P_1 | e_2 \mathbf{S} \mapsto P_2 | \dots | e_n \mathbf{S} \mapsto P_n](\rho) \triangleq \bigcup_{k=1}^n \mathcal{C}[\text{@}e_k \mathbf{S} \mapsto P_k](\rho) \quad (65)$$

17. **The interrupt-driven process:** The semantic function of the interrupt-driven process relation in RTPA is specified as follows.

$$\mathcal{C}[\text{@}i \odot \mapsto P](\rho) \triangleq (\mathcal{C}[\text{@}i \odot]; \mathcal{C}[P])(\rho) \quad (66)$$

It is noteworthy that an interrupt is considered as a special kind of events in the semantics of RTPA, where the interrupt variables are defined in the system's environment \mathbb{EN} and they are transparent to the environment of applications.

CONCLUSION

The denotational semantics of RTPA has been elaborated on the basis of the temporal ordered duration sequences. According to the denotational semantics of RTPA, instantaneous behaviors of processes at a given time moment have been captured by the changes of the environment. This work has provided a formal model for denoting the semantics of both concurrent and sequential processes required by RTPA in the denotational approach. An RTPA type checker and an RTPA code generator have been developed based on the formal models of deductive, denotational, and operational semantics. The denotational semantics for RTPA can be used as rules for correctness checking and system verification in system design and modeling using RTPA.

The denotational semantic framework is still not powerful enough to express some powerful features of RTPA processes. The denotational semantics of RTPA at the program

level where multiple processes are composed to describe the architectures and behaviors of a given system needs to be studied further. The latest development on the *deductive semantics* of RTPA (Wang, 2006a, 2008b) and the *mathematic laws* of RTPA (Wang, 2008d) has addressed the above problems with more rigorous treatment. A comparative analysis of a set of comprehensive formal semantics for RTPA has been carried out (Wang, 2007a). The deductive semantics of RTPA is presented in (Wang, 2008b). The operational semantics of RTPA is reported in (Wang & Ngolah, 2008). RTPA has been used not only in software system specifications, but also in human and intelligent system modeling (Wang, 2007b, 2007c). 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.

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

- 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.
- Corsetti E., Montanari, A., & Ratto, E. (1991). Dealing with different time granularities in formal specifications of real-time systems. *The Journal of Real-Time Systems*, 3(2), 191-215.
- Dierks, H. (2000). A process algebra for real-time programs. *LNCS #1783*, Springer, Berlin, pp. 66-76.
- Fecher, H. (2001). A real-time process algebra with open intervals and maximal progress. *Nordic Journal*

- of *Computing*, 8(3), 346-360.
- 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.
- Louden K.C. (1993). *Programming languages: Principles and practice*. Boston: PWS-Kent Publishing Co.
- 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.
- RAISE (1992). *The RAISE specification language*. London: Prentice Hall.
- Schneider, S. (1995). An operational semantics for timed CSP. *Information and Computation*, 116(2), 193-213.
- Schneider, S. (2000). *Concurrent and real-time systems: The CSP approach*. Wiley.
- Tan, X. (2006). Toward automatic code generation based on real-time process algebra (RTPA). PhD Thesis, University of Calgary, Canada.
- Tan, X., Wang, Y. & Ngolah, C.F. (2004). Specification of the RTPA grammar and its recognition. *Proceedings of the 2004 IEEE International Conference on Cognitive Informatics (ICCI'04)*, IEEE CS Press, Victoria, Canada, August, pp. 54-63.
- Wang, Y. (2002). The real-time process algebra (RTPA). *Annals of Software Engineering: A 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)*, 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. (2008d). Mathematical laws of software. *Transactions of Computational Science*, 2(2).
- Wang, Y. & Ngolah, C.F. (2008). An operational semantics of real-time process algebra (RTPA). *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(3), July.
- Winskel, G. (1993). *The formal semantics of programming languages*. MIT Press.

Xinming Tan is a professor at School of Computer Science and Technology, Wuhan University of Technology, China. He received a PhD in software engineering at University of Calgary, Canada in 2007. His major research interests are in formal methods, real-time systems, and cognitive informatics.

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.