

Deductive Semantics of RTPA

Yingxu Wang, University of Calgary, Canada

ABSTRACT

Deductive semantics is a novel software semantic theory that deduces the semantics of a program in a given programming language from a unique abstract semantic function to the concrete semantics embodied by the changes of status of a finite set of variables constituting the semantic environment of the program. There is a lack of a generic semantic function and its unified mathematical model in conventional semantics, which may be used to explain a comprehensive set of programming statements and computing behaviors. This article presents a complete paradigm of formal semantics that explains how deductive semantics is applied to specify the semantics of real-time process algebra (RTPA) and how RTPA challenges conventional formal semantic theories. Deductive semantics can be applied to define abstract and concrete semantics of programming languages, formal notation systems, and large-scale software systems, to facilitate software comprehension and recognition, to support tool development, to enable semantics-based software testing and verification, and to explore the semantic complexity of software systems. Deductive semantics may greatly simplify the description and analysis of the semantics of complicated software systems specified in formal notations and implemented in programming languages.

Keywords: computational linguistics; deductive semantics; denotational mathematics; formal semantics; mathematical model of semantics; RTPA; semantic analyses; semantic diagrams; semantic functions; semantic environment; semantics of RTPA; software engineering

INTRODUCTION

Semantics in linguistics is a domain that studies the interpretation of words and sentences, and analysis of their meanings. Semantics deals with how the meaning of a sentence in a language is obtained, hence the sentence is comprehended. Studies on semantics explore mechanisms in the understanding of languages and their meanings on the basis of syntactic structures (Chomsky, 1956, 1957, 1959, 1962, 1965, 1982; Tarski, 1944).

Software semantics in computing and computational linguistics have been recognized as one of the key areas in the development of

fundamental theories for computer science and software engineering (Bjoner, 2000; Gries, 1981; Hoare, 1969; McDermid, 1991; Slonneg & Kurts, 1995; Wang, 2006b, 2007c). The semantics of a programming language is the behavioral meaning that constitute what a syntactically correct instructional statement in the language is supposed to do during run time. The development of formal semantic theories of programming is one of the pinnacles of computing and software engineering (Gunter, 1992; Meyer, 1990; Loudon, 1993; Bjoner, 2000; Pagan, 1981).

Definition 1. *The semantics of a program in a given programming language is the logical consequences of an execution of the program that results in the changes of values of a finite set of variables and/or the embodiment of computing behaviors in the underpinning computing environment.*

A number of formal semantics, such as the *operational* (Marcotty & Ledgard, 1986; Ollongren, 1974; Wegner, 1972; Wikstrom, 1987), *denotational* (Bjorner and Jones, 1982; Jones, 1980; Schmidt, 1988, 1994, 1996; Scott, 1982; Scott & Strachey, 1971), *axiomatic* (Dijkstra, 1975, 1976; Gries, 1981; Hoare, 1969), and *algebraic* (Goguen, Thatcher, Wagner, & Wright, 1977; Gougen & Malcolm, 1996; Guttag & Horning, 1978), have been proposed in the last three decades for defining and interpreting the meanings of programs and programming languages. The classic software semantics are oriented on a certain set of software behaviors that are limited at the level of language statements rather than that of programs and software systems. There is a lack of a generic semantic function and its unified mathematical model in conventional semantics, which may be used to explain a comprehensive set of programming statements and computing behaviors. The mathematical models of the target machines and the semantic environments in conventional semantics seem to be inadequate to deal with the semantics of complex programming requirements, and to express some important instructions, complex control structures, and the real-time environments at run time. For supporting systematical and machine enabled semantic analysis and code generation in software engineering, the *deductive semantics* is developed that provides a systematic semantic analysis methodology.

Deduction is a reasoning process that discovers new knowledge or derives a specific conclusion based on generic premises such as abstract rules or principles (Wang, 2006b, 2007a, 2007c). The nature of semantics of a given programming language is its computational meanings or embodied behaviors expressed

by an instruction in the language. Because the carriers of software semantics are a finite set of variables declared in a given program, program semantics can be reduced onto the changes of values of these variables over time. In order to provide a rigorous mathematical treatment of both the abstract and concrete semantics of software, a new type of formal semantics known as the deductive semantics is presented.

Definition 2. *Deductive semantics is a formal semantics that deduces the semantics of a program in a given programming language from a generic abstract semantic function to the concrete semantics, which are embodied onto the changes of status of a finite set of variables constituting the semantic environment of computing.*

This article presents a comprehensive theory of deductive semantics of software systems. The mathematical models of deductive semantics and the fundamental properties are described. The deductive models of semantics, semantic function, and semantic environment at various composing levels of programs are introduced. Properties of software semantics and relationships between the software behavioral space and the semantic environment are studied. New methods such as the semantic differential and semantic matrix are developed to facilitate deductive semantic analyses from a generic semantic function to a specific semantic matrix, and from semantics of statements to those of processes and programs. The establishment of the deductive semantic rules of RTPA (Wang, 2002, 2003, 2006a, 2006b, 2007a, 2007b, 2008a, 2008b) is described, where the semantics of a comprehensive set of processes is systematically modeled.

THE THEORY OF DEDUCTIVE SEMANTICS

This section presents the theory of deductive semantics (Wang, 2006b, 2007c). A generic mathematical model of deductive semantics of software is developed, and the concepts of semantic environment and semantic func-

tion are rigorously defined. Based on them, deductive semantics of programs at different composition levels are rigorously modeled. Then, common properties of software semantics are analyzed.

The Semantic Environment and Semantic Function

Definition 3. A semantic environment Θ of a programming language is a logical model of a set of identifiers I and their values V bound in pairs, i.e.:

$$\begin{aligned} \Theta &\triangleq f : I \rightarrow V, V \subseteq \mathbb{R} \\ &\quad \#I \\ &= \{R(i_k, v_k)\} \\ &= \{(i_1, v_1), (i_2, v_2), \dots, (i_{\#I}, v_{\#I})\} \end{aligned} \tag{1}$$

where \mathbb{R} is the set of real numbers, $i_k \in I, v_k \in V \subseteq \mathbb{R}$, and $\#I$ the number of elements in I .

Note the big-R notation is adopted to denote a set of recurring structures or repetitive behaviors (Wang, 2002, 2007c, 2008a). The semantic environment constituting the behaviors of software is inherently a three dimensional structure known as those of operations, memory space, and time.

Definition 4. The behavioral space Ω of a program executed on a certain machine is a finite set of variables operated in a 3-D state space determined by a triple, i.e.:

$$\Omega \triangleq (OP, T, S) \tag{2}$$

where OP is a finite set of operations, T is a finite set of discrete time points of program execution, and S is a finite set of memory locations or their logical representations by identifiers of variables.

According to Definitions 3 and 4, the set of variables of a program, S , plays an important role in semantic modeling and analysis, because they are the objects of software behavioral operations and the carriers of program semantics.

Variables can be classified as *free* and *system* variables. The former are user defined and the latter are language provided. From a functional point of view, variables can be classified into *object representatives*, *control variables*, *result containers*, and *address locaters*. The life spans or scopes of variables can be categorized as *persistent*, *global*, *local*, and *temporal*. The persistent variables are those that their lifespan are longer than the program that generates them, such as data in a database or files in a distributed network.

A new calculus introduced in deductive semantics is the *partial differential of sets* (Wang, 2006b, 2007c), which is used to facilitate the instantiation of abstract semantics by concrete ones, as described below.

Definition 5. Given two sets X and $U, X \subseteq \mathcal{P}U$, a partial differential of X on U with elements $x, x \in X$, denoted by $\partial U / \partial x$, is an elicitation of interested elements from U as specified in X , i.e.:

$$\begin{aligned} \frac{\partial U}{\partial x} &\triangleq X \cap U, x \in X \\ &= X, X \subseteq \mathcal{P}U \end{aligned} \tag{3}$$

where $\mathcal{P}U$ denotes a power set of U .

The partial differential of sets can be easily extended to double, triple, or more generally, multiple partial differentials as defined below.

Definition 6. A multiple partial differential of $X_1, X_2, \dots,$ and X_n on $\mathcal{P}U$ with elements $x_1 \in X_1, x_2 \in X_2, \dots,$ and $x_n \in X_n$, denoted by

$$\frac{\partial^n}{\partial x_1 \partial x_2 \dots \partial x_n} U$$

is a Cartesian product of all partial differentials that select interested elements from U as specified in $X_1, X_2, \dots,$ and X_n , respectively, i.e.:

$$\frac{\partial^n}{\partial x_1 \partial x_2 \dots \partial x_n} U \triangleq X_1 \times X_2 \times \dots \times X_n \tag{4}$$

where $X_1, X_2, \dots, X_n \subseteq \wp U$ and $\forall i \neq j, 1 \leq i, j \leq n, X_i \cap X_j = \emptyset$.

For example,

$$\frac{\partial^2}{\partial x \partial y} U = X \times Y, x \in X, y \in Y, \text{ and } X, Y \subseteq \wp U$$

and

$$\frac{\partial^3}{\partial x \partial y \partial z} U = X \times Y \times Z, x \in X, y \in Y, z \in Z, \text{ and } X, Y, Z \subseteq \wp U.$$

On the basis of the definitions of software behavioral space and partial differential of sets, the semantic environment of software can be formally described.

Definition 7. *The semantic environment Θ of a program on a certain target machine is its run-time behavioral space Ω projected onto the Cartesian plane determined by T and S , i.e.:*

$$\begin{aligned} \Theta &= \frac{\partial^2 \Omega}{\partial t \partial s}, t \in T \wedge s \in S \\ &= \frac{\partial^2 \Omega}{\partial t \partial s} (OP, T, S) \\ &= T \times S \end{aligned} \tag{5}$$

As indicated in Definition 7, the semantic environment of a program is a dynamic space over time, because following each execution of a statement in the program, the semantic environment Θ , particularly the sets of variables S and their values V , may be changed.

In semantic analysis, the changed part of the semantic environment Θ is particularly interested, which is the embodiment of software semantics. A generic semantic function is developed below, which can be used to derive a specific and concrete semantic function for

a given statement, process, or program by mathematical deduction.

Definition 8. *A semantic function of a program $\wp, f_\theta(\wp)$, is a function that maps the semantic environment Θ into a finite set of values V determining by a Cartesian product on a finite set of executing steps T and a finite set of variables S , i.e.:*

$$f_\theta(\wp) \triangleq f : T \times S \rightarrow V = \begin{pmatrix} & s_1 & s_2 & \dots & s_m \\ t_0 & \perp & \perp & \dots & \perp \\ t_1 & v_{11} & v_{12} & & v_{1m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_n & v_{n1} & v_{n1} & \dots & v_{nm} \end{pmatrix} \tag{6}$$

where $T = \{t_0, t_1, \dots, t_n\}$, $S = \{s_1, s_2, \dots, s_m\}$, and V is a finite set of values $v(t_i, s_j)$, $0 \leq i \leq n$, and $1 \leq j \leq m$.

In Equation 6, all values of $v(t_i, s_j)$ at t_0 is undefined for a program as denoted by the bottom symbol \perp , i.e. $v(0, s_j) = \perp, 1 \leq j \leq m$. However, for a statement or a process, it is usually true that $v(0, s_j) \neq \perp$ dependent on the context of previous statement(s) or the initialization of the system.

According to Definitions 7 and 8, the semantic environment and the domain of a semantic function can be illustrated by a semantic diagram as described below (Wang, 2006b, 2007c).

Definition 9. *A semantic diagram is a sub Cartesian-plane in the semantic environment Θ that forms the domain of the semantic function for a given process P with $f_\theta(P) = f: T_p \times S_p \rightarrow V_p$*

For example, the semantic diagram of an abstract process $P, f_\theta(P)$, as defined in Definition 9 can be illustrated in Figure 1, where V_p is the domain of dynamic variable values of process P over time, i.e., $V_p = T_p \times S_p$. The semantic

diagram of two sequential processes, $P \rightarrow Q$, can be referred to Figure 3.

The semantic diagram can be used to analyze complex semantic relations, and to demonstrate semantic functions and their semantic environments. Observing Figures 1 and 3, the flowing properties of process relations can be derived.

Lemma 1. *The variables of two arbitrary processes P and Q , S_p and S_q in the semantic environment Θ possess the following properties:*

a. *The entire set of variables:*

$$S = S_p \cup S_q \tag{7}$$

b. *Global variables:*

$$S_G \subseteq S_p \cap S_q \tag{8}$$

c. *Local variables:*

$$S_L = S - S_G, S_L \subseteq S_p \oplus S_q,$$

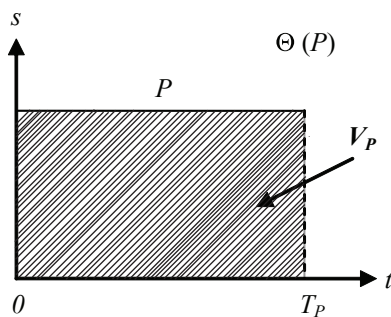
where $S_{Lp} = S_L \setminus S_q$ and $S_{Lq} = S_L \setminus S_p$

$$\tag{9}$$

Deductive Semantics of Programs at Different Levels of Compositions

According to the generic model and the hierarchical architecture of programs, the semantics of a program in a given programming language can be described and analyzed at various composition levels, such as those of *statement*,

Figure 1. The semantic diagram of a process



process, and *system* from the bottom-up (Wang, 2007c, 2008b).

Definition 10. *The semantics of a statement p , $\theta(p)$, on a given semantic environment Θ is a double partial differential of the semantic function $f_\theta(p)$ on executing steps T and the set of variables S , i.e.:*

$$\begin{aligned} \theta(P) &\triangleq \frac{\partial^2}{\partial t \partial s} f_\theta(p) \\ &= \underset{\#T(p)}{\mathbf{R}} \underset{\#S(p)}{\mathbf{R}} (v_p, t_i, s_j) \\ &= \underset{1}{\mathbf{R}} \underset{\#\{s_1, s_2, \dots, s_m\}}{\mathbf{R}} (v_p, t_i, s_j) \\ &= \begin{pmatrix} & \mathbf{s}_1 & \mathbf{s}_2 & \dots & \mathbf{s}_m \\ \mathbf{t}_0 & v_{01} & v_{02} & \dots & v_{0m} \\ (\mathbf{t}_0, \mathbf{t}_1] & v_{11} & v_{12} & \dots & v_{1m} \end{pmatrix} \end{aligned} \tag{10}$$

where t denotes the discrete time immediately before and after the execution of p during (t_0, t_1) , and $\#$ is the cardinal calculus that counts the number of elements in a given set, i.e. $n = \#T(p)$ and $m = \#S(p)$.

In Definition 10, the first partial differential selects all related variable $S(p)$ of the statement p from Θ . The second partial differential selects a set of discrete steps of p 's execution $T(p)$ from Θ . According to Definition 10, the semantics of a statement can be reduced onto a semantic function that results in a 2-D matrix with the changes of values of all variables over time of program execution.

On the basis of Definitions 8 and 10, semantics of individual statements can be analyzed using Equation 10 in a deductive process.

Example 1. *Analyze the semantics of Statement 3, $\theta(p_3)$, in the following program entitled sum:*

```
void sum;
{
```

```

(0) int x, y, z;
(1) x = 8;
(2) y = 2;
(3) z := x + y;
}
    
```

According to Definition 10, the semantics of Statement p_3 is as follows:

$$\begin{aligned}
 \theta(p_3) &= \frac{\partial^2}{\partial t \partial s} f_0(p_3) \\
 &= \mathop{R}_{i=2}^3 \left(\mathop{R}_{j=1}^{\#S(p_3)} v_{p_3}(t_i, s_j) \right) \\
 &= \mathop{R}_{i=2}^3 \left(\mathop{R}_{j=1}^{\#\{x,y,z\}} v_{p_3}(t_i, s_j) \right) \\
 &= \left(\begin{array}{ccc|ccc}
 & \mathbf{x} & \mathbf{y} & \mathbf{z} & & \\
 \mathbf{t}_2 & 8 & 2 & \perp & & \\
 \mathbf{(t_2, t_3)} & 8 & 2 & 10 & &
 \end{array} \right)
 \end{aligned}
 \tag{11}$$

This example shows how the concrete semantics of a statement can be derived on the basis of the generic and abstract semantic function as given in Definition 10.

Definition 11. The semantic effect of a statement p , $\theta^*(p)$, is the resulted changes of values of variables by its semantic function $\theta(p)$ during the time interval immediately before and after the execution of p , $\Delta t = (t_p, t_{i+1})$, see Box 1, where \rightarrow denotes a transition of values for a given variable.

Example 2. For the same statement p_3 as given in Example 1, determine its semantic effect $\theta^*(p_3)$.

According to Equation 12, the semantic effect $\theta^*(p_3)$ is seen in Box 2.

It is noteworthy in Examples 1 and 2 that deductive semantics can be used not only to describe the *abstract* and *concrete* semantics of programs, but also to elicit and highlight their semantic effects.

Considering that a program or a process is composed by individual statements with given rules of compositions, the definition and mathematical model of deductive semantics at the statement level can be extended onto the higher levels of program hierarchy.

Box 1.

$$\begin{aligned}
 \theta^*(p) &= \mathop{R}_{j=1}^{\#S(p)} (v_p(t_i, s_j) \oplus v_p(t_{i+1}, s_j)) \\
 &= \mathop{R}_{j=1}^{\#S(p)} \langle v_p(t_i, s_j) \rightarrow v_p(t_{i+1}, s_j) \mid v_p(t_i, s_j) \neq v_p(t_{i+1}, s_j) \rangle
 \end{aligned}
 \tag{12}$$

Box 2.

$$\begin{aligned}
 \theta^*(p_3) &= \mathop{R}_{j=1}^{\#S(p_3)} \langle v_{p_3}(t_2, s_j) \rightarrow v_{p_3}(t_3, s_j) \mid v_{p_3}(t_2, s_j) \neq v_{p_3}(t_3, s_j) \rangle \\
 &= \mathop{R}_{j=1}^{\#\{x,y,z\}} \langle v_{p_3}(t_2, s_j) \rightarrow v_{p_3}(t_3, s_j) \mid v_{p_3}(t_2, s_j) \neq v_{p_3}(t_3, s_j) \rangle \\
 &= \{ \langle v_{p_3}(t_2, z) = \perp \rightarrow v_{p_3}(t_3, z) = 10 \rangle \}
 \end{aligned}$$

Definition 12. The semantics of a process $P, \theta(P)$, on a given semantic environment Θ is a double partial differential of the semantic function $f_\theta(P)$ on the sets of variables S and executing steps T , see Box 3, where $V_{p_k}, 1 \leq k \leq n-1$, is a set of values of local variables that belongs to processes P_k and V_G is a finite set of values of global variables.

On the basis of Definition 12, the semantics of a program at the top-level composition can be deduced to the combination of the semantics of a set of processes, each of which can be further deduced to the composition of all statements' semantics as described below.

Definition 13. The semantics of a program \wp , $\theta(\wp)$, on a given semantic environment Θ , is a combination of the semantic functions of all processes $\theta(P_k), 1 \leq k \leq n$, i.e.:

$$\begin{aligned} \theta(\wp) &= \prod_{k=1}^{\#K(\wp)} \frac{\partial^2}{\partial t \partial S} f_\theta(\wp) \\ &= \prod_{k=1}^{\#K(\wp)} \theta(P_k) \\ &= \prod_{k=1}^{\#K(\wp)} \left[\prod_{i=0}^{\#T(P_k)} \left(\prod_{j=1}^{\#S(P_k)} v_{P_k}(t_i, S_j) \right) \right] \end{aligned} \tag{14}$$

Box 3.

$$\begin{aligned} \theta(P) &= \frac{\partial^2}{\partial t \partial S} f_\theta(P) \\ &= \prod_{k=1}^{n-1} \left\{ \left[\frac{\partial^2}{\partial t \partial S} f_\theta(P_k) \right] r_{kl} \left[\frac{\partial^2}{\partial t \partial S} f_\theta(P_l) \right] \right\}, l = k + 1 \\ &= \prod_{k=1}^{n-1} \left\{ \left[\prod_{i=0}^{\#T(P_k)} \left(\prod_{j=1}^{\#S(P_k)} v_{P_k}(t_i, S_j) \right) \right] r_{kl} \left[\prod_{i=0}^{\#T(P_l)} \left(\prod_{j=1}^{\#S(P_l)} v_{P_l}(t_i, S_j) \right) \right] \right\} \\ &= \begin{pmatrix} \mathbf{V}_{P_1} & & & \mathbf{V}_G \\ & \mathbf{V}_{P_2} & & \mathbf{V}_G \\ & & \ddots & \vdots \\ & & & \mathbf{V}_{P_{n-1}} & \mathbf{V}_G \end{pmatrix} \end{aligned} \tag{13}$$

where $\#K(\wp)$ is the number of processes or components encompassed in the program.

It is noteworthy that Equation 14 will usually result in a very large matrix of semantic space, which can be quantitatively predicated as follows.

Definition 14. The semantic space of a program $S_\theta(\wp)$ is a product of $\#S(\wp)$ variables and $\#T(\wp)$ executing steps, i.e.:

$$\begin{aligned} S_\theta(\wp) &= \#S(\wp) \bullet \#T(\wp) \\ &= \sum_{k=1}^{\#K(\wp)} \#S(\wp_k) \bullet \sum_{k=1}^{\#K(\wp)} \#T(\wp_k) \end{aligned} \tag{15}$$

The semantic space of programs provides a useful measure of software complexity. Due to the tremendous size of the semantic space, both program composition and comprehension are innately a hard problem in terms of complexity and cognitive difficulty.

Properties of Software Semantics

Observing the formal definitions and mathematical models of deductive semantics developed in previous subsections, a number of common properties of software semantics may be elicited,

which are useful for explaining the fundamental characteristics of software semantics.

One of the most interesting characteristics of program semantics is its invariance against different executing speeds as described in the following theorem.

Theorem 1. *The asynchronicity of program semantics states that the semantics of a relatively timed program is invariant with the changes of executing speed, as long as any absolute time constraint is met.*

Theorem 1 asserts that, for most non real-time or relatively timed programs, different executing speeds or simulation paces will not alter the semantics of the software system. This explains why a programmer may simulate the run-time behaviors of a given program executing at a speed of up to 10^9 times faster than that of human beings. It also explains why computers with different system clock frequencies may correctly run the same program and obtain the same behavior.

Definition 15. *The behavior of a computational statement is a set of observable actions or changes of status of objects operated by the statement.*

According to Definition 4, the behavioral space of software, Ω , is three dimensional, while as given in Definition 7, the semantic

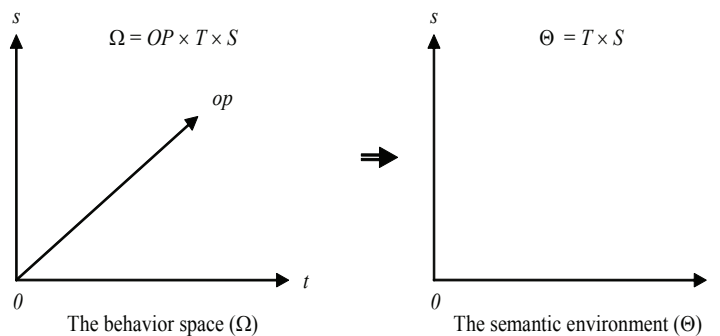
environment Θ is two dimensional. Therefore, to a certain extent, semantic analysis is a projection of the 3-D software behaviors into the 2-D semantic environment Θ as shown in Figure 2.

The theory of deductive semantics can be systematically applied to formally and rigorously model and describe the semantics of the RTPA metaprocesses and the process relations (operations). On the basis of the mathematical models and properties of deductive semantics, the following sections formally describe a comprehensive set of RTPA semantics, particularly the 17 metaprocesses and the 17 process relations (Wang, 2002, 2003, 2007c, 2008a, 2008b). This work extends the coverage of semantic rules of programming languages to a complete set of features that encompasses both basic computing operations and their algebraic composition rules. Because RTPA is a denotational mathematical structure based on process algebra that covers a comprehensive set of computing and programming requirements, any formal semantics that is capable to process RTPA is powerful enough to express the semantics of any programming language.

DEDUCTIVE SEMANTICS OF RTPA METAPROCESSES

Metaprocesses of RTPA are elicited from basic computational requirements. Complex

Figure 2. Relationship between software behavior space and the semantic environment



processes can be composed with multiple metaprocesses. RTPA identified 17 metaprocesses, \mathfrak{P} , on fundamental computing operations such as assignment, system control, event/time handling, memory and I/O manipulation, i.e., $\mathfrak{P} = \{:=, \blacklozenge, \Rightarrow, \Leftarrow, \neq, >, <, |>, |<, @, \hat{=}, \uparrow, \downarrow, !, \otimes, \boxtimes, \S\}$. Detailed descriptions of the metaprocesses of RTPA and their syntaxes may be referred to (Wang, 2002, 2007c, 2008b), where each metaprocess is a basic operation on one or more operands such as variables, memory elements, or I/O ports. Based on Definitions 8 and 12, the deductive semantics of the set of RTPA metaprocesses can be defined in the following subsections.

The Assignment Process

Definition 16. *The semantics of the assignment process on a given semantic environment Θ , $\theta(\mathbf{yRT} := \mathbf{xRT})$, is a double partial differential of the semantic function $f_\theta(\mathbf{yRT} := \mathbf{xRT})$ on the sets of variables S and executing steps T , i.e.:*

$$\begin{aligned} \theta(\mathbf{yRT} := \mathbf{xRT}) &\triangleq \frac{\partial^2}{\partial t \partial s} f_\theta(\mathbf{yRT} := \mathbf{xRT}) \\ &= \prod_{i=0}^{\#T(\mathbf{yRT} := \mathbf{xRT})} \left(\prod_{j=1}^{\#S(\mathbf{yRT} := \mathbf{xRT})} v_{t_i, s_j} \right) \\ &= \prod_{i=0}^1 \prod_{j=1}^2 v_{t_i, s_j} \\ &= \begin{pmatrix} & \mathbf{xRT} & \mathbf{yRT} \\ \mathbf{t}_0 & \mathbf{xRT} & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \mathbf{xRT} & \mathbf{xRT} \end{pmatrix} \end{aligned} \quad (16)$$

where the size of the matrix is $\#T \bullet \#S$.

The Evaluation Process

Definition 17. *The semantics of the evaluation process on Θ , $\theta(\blacklozenge \exp \mathbb{T} \rightarrow \mathbb{T})$, is a double partial differential of the semantic function $f_\theta(\blacklozenge \exp \mathbb{T} \rightarrow \mathbb{T})$ on the sets of variables S and executing steps T , i.e.:*

$\rightarrow \mathbb{T})$ on the sets of variables S and executing steps T in the following two forms, i.e.:

$$\begin{aligned} \theta(\blacklozenge \exp \mathbf{BL} \rightarrow \mathbf{BL}) &\triangleq \frac{\partial^2}{\partial t \partial s} f_\theta(\blacklozenge \exp \mathbf{BL} \rightarrow \mathbf{BL}) \\ &= \prod_{i=0}^{\#T(\blacklozenge \exp \mathbf{BL} \rightarrow \mathbf{BL})} \left(\prod_{j=1}^{\#S(\blacklozenge \exp \mathbf{BL} \rightarrow \mathbf{BL})} v_{t_i, s_j} \right) \\ &= \prod_{i=0}^1 \prod_{j=1}^2 v_{t_i, s_j} \\ &= \begin{pmatrix} & \exp \mathbf{B} & \blacklozenge(\exp \mathbf{BL}) \mathbf{BL} \\ (\mathbf{t}_0, \mathbf{t}_1] & \delta(\exp \mathbf{BL}) \mathbf{BL} & \perp \\ (\mathbf{t}_1, \mathbf{t}_2] & \mathbf{T} & \mathbf{T} \\ (\mathbf{t}_1, \mathbf{t}_2] & \mathbf{F} & \mathbf{F} \end{pmatrix} \end{aligned} \quad (17a)$$

or

$$\begin{aligned} \theta(\blacklozenge \exp \mathbb{T} \rightarrow \mathbb{T}) &\triangleq \frac{\partial^2}{\partial t \partial s} f_\theta(\blacklozenge \exp \mathbb{T} \rightarrow \mathbb{T}) \\ &= \prod_{i=0}^{\#T(\blacklozenge \exp \mathbb{T} \rightarrow \mathbb{T})} \left(\prod_{j=1}^{\#S(\blacklozenge \exp \mathbb{T} \rightarrow \mathbb{T})} v_{t_i, s_j} \right) \\ &= \prod_{i=0}^1 \prod_{j=1}^2 v_{t_i, s_j} \\ &= \begin{pmatrix} & \exp \mathbb{T} & \blacklozenge(\exp \mathbb{T}) \mathbb{T} \\ (\mathbf{t}_0, \mathbf{t}_1] & \delta(\exp \mathbb{T}) \mathbb{T} & \perp \\ (\mathbf{t}_1, \mathbf{t}_2] & n\mathbb{T} & n\mathbb{T} \end{pmatrix} \end{aligned} \quad (17b)$$

where $\blacklozenge(\exp \mathbf{BL})$ is the Boolean evaluation function on $\exp \mathbf{BL}$ that results in \mathbf{T} or \mathbf{F} . $\blacklozenge(\exp \mathbb{T})$ is a more general cardinal or numerical evaluation function on $\exp \mathbb{T}$ that results in $\mathbb{T} = \{\mathbf{N}, \mathbf{Z}, \mathbf{R}, \mathbf{B}\}$, i.e., in types of nature number, integer, real number, and byte, respectively (Wang, 2002).

The Addressing Process

Definition 18. *The semantics of the addressing process on Θ , $\theta(\text{id} \mathbf{S} \Rightarrow \text{ptr} \mathbf{P})$, is a double partial differential of the semantic function $f_\theta(\text{id} \mathbf{S} \Rightarrow \text{ptr} \mathbf{P})$ on the sets of variables S and executing steps T , i.e.:*

$$\begin{aligned}
 \theta(id\mathbf{S} \Rightarrow ptr\mathbf{P}) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(id\mathbf{S} \Rightarrow ptr\mathbf{P}) \\
 &= \overset{\#T(id\mathbf{S} \Rightarrow ptr\mathbf{P})}{R}_{i=0} \left(\overset{\#S(id\mathbf{S} \Rightarrow ptr\mathbf{P})}{R}_{j=1} v t_i, s_j \right) \\
 &= \overset{1}{R}_{i=0} \overset{2}{R}_{j=1} v(t_i, s_j) \\
 &= \begin{pmatrix} & id\mathbf{S} & ptr\mathbf{P} \\ \mathbf{t}_0 & id\mathbf{S} & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & id\mathbf{S} & \pi(id\mathbf{S})\mathbf{H} \end{pmatrix}
 \end{aligned} \tag{18}$$

where $\pi(id\mathbf{S})\mathbf{H}$ is a function that associates a declared identifier $id\mathbf{S}$ to its hexadecimal memory address located by the pointed $ptr\mathbf{P}$.

The Memory Allocation Process

Definition 19. The semantics of the memory allocation process on Θ , $\theta(id\mathbf{S} \Leftarrow MEM(ptr\mathbf{P})\mathbf{RT})$, is a double partial differential of the semantic function $f_0(id\mathbf{S} \Leftarrow MEM(ptr\mathbf{P})\mathbf{RT})$ on the sets of variables S and executing steps T , see Box 4. Where $\pi(id\mathbf{S})\mathbf{H}$ is a mapping function that associates an identifier $id\mathbf{S}$ to a memory block starting at a hexadecimal address located by the pointed $ptr\mathbf{P}$. The ending address of the allocated memory block, $ptr\mathbf{P} + size(\mathbf{RT}) - 1$, is

dependent on a machine implementation of the size of a given variable in type \mathbf{RT} .

The Memory Release Process

Definition 20. The semantics of the memory release process on Θ , $\theta(id\mathbf{S} \Leftarrow MEM(\perp)\mathbf{RT})$, is a double partial differential of the semantic function $f_0(id\mathbf{S} \Leftarrow MEM(\perp)\mathbf{RT})$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(id\mathbf{S} \Leftarrow MEM[\perp]\mathbf{RT}) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(id\mathbf{S} \Leftarrow MEM[\perp]\mathbf{RT}) \\
 &= \overset{\#T(id\mathbf{S} \Leftarrow MEM[\perp]\mathbf{RT})}{R}_{i=0} \left(\overset{\#S(id\mathbf{S} \Leftarrow MEM[\perp]\mathbf{RT})}{R}_{j=1} v t_i, s_j \right) \\
 &= \overset{1}{R}_{i=0} \overset{3}{R}_{j=1} v(t_i, s_j) \\
 &= \begin{pmatrix} & id\mathbf{RT} & ptr\mathbf{P} & MEM\mathbf{RT} \\ \mathbf{t}_0 & (id\mathbf{S}) & \pi(id\mathbf{S})\mathbf{H} & MEM(ptr\mathbf{P})\mathbf{RT} \\ (\mathbf{t}_0, \mathbf{t}_1] & \perp & \perp & \perp \end{pmatrix}
 \end{aligned} \tag{20}$$

The Read Process

Definition 21. The semantics of the read process on Θ , $\theta(MEM(ptr\mathbf{P})\mathbf{RT} \triangleright x\mathbf{RT})$, is a double partial differential of the semantic function $f_0(MEM(ptr\mathbf{P})\mathbf{RT} \triangleright x\mathbf{RT})$ on the sets of variables S and executing steps T , see Box 5.

Box 4.

$$\begin{aligned}
 \theta(id\mathbf{S} \Leftarrow MEM[ptr\mathbf{P}]\mathbf{RT}) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(id\mathbf{S} \Leftarrow MEM[ptr\mathbf{P}]\mathbf{RT}) \\
 &= \overset{\#T(id\mathbf{S} \Leftarrow MEM[ptr\mathbf{P}]\mathbf{RT})}{R}_{i=0} \left(\overset{\#S(id\mathbf{S} \Leftarrow MEM[ptr\mathbf{P}]\mathbf{RT})}{R}_{j=1} v t_i, s_j \right) \\
 &= \overset{1}{R}_{i=0} \overset{3}{R}_{j=1} v(t_i, s_j) \\
 &= \begin{pmatrix} & id\mathbf{S} & ptr\mathbf{P} & MEM\mathbf{RT} \\ \mathbf{t}_0 & id\mathbf{S} & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & id\mathbf{S} & \pi(id\mathbf{S})\mathbf{H} & MEM[ptr\mathbf{P}]\mathbf{RT} \end{pmatrix}
 \end{aligned} \tag{19}$$

The Write Process

Definition 22. The semantics of the write process on Θ , $\theta(\text{MEM}(\text{ptr}\mathbf{P}\mathbf{RT} \leftarrow \mathbf{xRT}))$, is a double partial differential of the semantic function $f_0(\text{MEM}(\text{ptr}\mathbf{P}\mathbf{RT} \leftarrow \mathbf{xRT}))$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} &\theta(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \leftarrow \mathbf{xRT}]) \\ &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \leftarrow \mathbf{xRT}]) \\ &= \prod_{i=0}^{\#T(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \leftarrow \mathbf{xRT}])} \left(\prod_{j=1}^{\#S(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \leftarrow \mathbf{xRT}])} v_{t_i, s_j} \right) \\ &= \prod_{i=0}^1 \prod_{j=1}^3 v_{t_i, s_j} \\ &= \left(\begin{array}{cccc} & \mathbf{xRT} & \text{ptr}\mathbf{P} & \text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT}] \\ \mathbf{t}_0 & \mathbf{xRT} & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \mathbf{xRT} & \text{ptr}\mathbf{P} & \mathbf{xRT} \end{array} \right) \end{aligned} \tag{22}$$

The Input Process

Definition 23. The semantics of the input process on Θ , $\theta(\text{PORT}(\text{ptr}\mathbf{P}\mathbf{RT} | \gg \mathbf{xRT}))$, is a double partial differential of the semantic function

$f_0(\text{PORT}(\text{ptr}\mathbf{P}\mathbf{RT} | \gg \mathbf{xRT}))$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} &\theta(\text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT} | \gg \mathbf{xRT}]) \\ &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(\text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT} | \gg \mathbf{xRT}]) \\ &= \prod_{i=0}^{\#T(\text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT} | \gg \mathbf{xRT}])} \left(\prod_{j=1}^{\#S(\text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT} | \gg \mathbf{xRT}])} v_{t_i, s_j} \right) \\ &= \prod_{i=0}^1 \prod_{j=1}^3 v_{t_i, s_j} \\ &= \left(\begin{array}{cccc} & \text{ptr}\mathbf{P} & \text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT}] & \mathbf{xRT} \\ \mathbf{t}_0 & \text{ptr}\mathbf{P} & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \text{ptr}\mathbf{P} & \text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT}] & \text{PORT}[\text{ptr}\mathbf{P}\mathbf{RT}] \end{array} \right) \end{aligned} \tag{23}$$

The Output Process

Definition 24. The semantics of the output process on Θ , $\theta(\mathbf{xRT} | \leftarrow \text{PORT}(\text{ptr}\mathbf{P}\mathbf{RT}))$, is a double partial differential of the semantic function $f_0(\mathbf{xRT} | \leftarrow \text{PORT}(\text{ptr}\mathbf{P}\mathbf{RT}))$ on the sets of variables S and executing steps T , i.e.:

Box 5.

$$\begin{aligned} &\theta(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \gg \mathbf{xRT}]) \\ &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \gg \mathbf{xRT}]) \\ &= \prod_{i=0}^{\#T(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \gg \mathbf{xRT}])} \left(\prod_{j=1}^{\#S(\text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT} \gg \mathbf{xRT}])} v_{t_i, s_j} \right) \\ &= \prod_{i=0}^1 \prod_{j=1}^3 v_{t_i, s_j} \\ &= \left(\begin{array}{cccc} & \text{ptr}\mathbf{P} & \text{MEM}(\text{ptr}\mathbf{P}\mathbf{RT}) & \mathbf{xRT} \\ \mathbf{t}_0 & \text{ptr}\mathbf{P} & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \text{ptr}\mathbf{P} & \text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT}] & \text{MEM}[\text{ptr}\mathbf{P}\mathbf{RT}] \end{array} \right) \end{aligned} \tag{21}$$

$$\begin{aligned}
 & \theta(\mathbf{xRT} \mid \llcorner \text{PORT}[\text{ptrP}]\text{RT}) \\
 & \triangleq \frac{\partial^2}{\partial t \partial s} f_0(\mathbf{xRT} \mid \llcorner \text{PORT}[\text{ptrP}]\text{RT}) \\
 & = \prod_{i=0}^{\#T(\mathbf{xRT} \mid \llcorner \text{PORT}[\text{ptrP}]\text{RT})} \left(\prod_{j=1}^{\#S(\mathbf{xRT} \mid \llcorner \text{PORT}[\text{ptrP}]\text{RT})} v(t_i, s_j) \right) \\
 & = \prod_{i=0}^1 \prod_{j=1}^3 v(t_i, s_j) \\
 & = \left(\begin{array}{cccc} \mathbf{xRT} & \text{ptrP} & \text{PORT}[\text{ptrP}]\text{RT} & \\ \mathbf{t}_0 & \mathbf{xRT} & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \mathbf{xRT} & \text{ptrP} & \mathbf{xRT} \end{array} \right)
 \end{aligned} \tag{24}$$

The Timing Process

Definition 25. The semantics of the timing process on Θ , $\theta(@i\mathbf{TM} @ \S t\mathbf{TM})$, is a double partial differential of the semantic function $f_0(@i\mathbf{TM} @ \S t\mathbf{TM})$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(@i\mathbf{TM} @ \S t\mathbf{TM}) & \triangleq \frac{\partial^2}{\partial t \partial s} f_0(@i\mathbf{TM} @ \S t\mathbf{TM}) \\
 & = \prod_{i=0}^{\#T(@i\mathbf{TM} @ \S t\mathbf{TM})} \left(\prod_{j=1}^{\#S(@i\mathbf{TM} @ \S t\mathbf{TM})} v(t_i, s_j) \right) \\
 & = \prod_{i=0}^1 \prod_{j=1}^2 v(t_i, s_j) \\
 & = \left(\begin{array}{ccc} \S t\mathbf{TM} & @i\mathbf{TM} & \\ \mathbf{t}_0 & \S t\mathbf{TM} & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \S t\mathbf{TM} & \S t\mathbf{TM} \end{array} \right)
 \end{aligned} \tag{25}$$

where \mathbf{TM} represents the three timing types, i.e., $\mathbf{TM} = \{\mathbf{yy:MM:dd}, \mathbf{hh:mm:ss:ms}, \mathbf{yy:MM:dd:hh:mm:ss:ms}\}$.

The Duration Process

Definition 26. The semantics of the duration process on Θ , $\theta(@i\mathbf{TM} \triangleq \S t\mathbf{TM} + \Delta d\mathbf{Z})$, is a double partial differential of the semantic function $f_0(@i\mathbf{TM} \triangleq \S t\mathbf{TM} + \Delta d\mathbf{N})$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 & \theta(@i\mathbf{TM} \triangleq \S t\mathbf{TM} + \Delta d\mathbf{Z}) \\
 & \triangleq \frac{\partial^2}{\partial t \partial s} f_0(@i\mathbf{TM} \triangleq \S t\mathbf{TM} + \Delta d\mathbf{Z}) \\
 & = \prod_{i=0}^{\#T(@i\mathbf{TM} \triangleq \S t\mathbf{TM})} \left(\prod_{j=1}^{\#S(@i\mathbf{TM} \triangleq \S t\mathbf{TM})} v(t_i, s_j) \right) \\
 & = \prod_{i=0}^1 \prod_{j=1}^3 v(t_i, s_j) \\
 & = \left(\begin{array}{cccc} \S t\mathbf{TM} & \Delta d\mathbf{N} & @i\mathbf{TM} & \\ \mathbf{t}_0 & \S t\mathbf{TM} & \Delta d\mathbf{N} & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & \S t\mathbf{TM} & \Delta d\mathbf{N} & \S t\mathbf{TM} + \Delta d\mathbf{N} \end{array} \right)
 \end{aligned} \tag{26}$$

where $\mathbf{TM} = \{\mathbf{yy:MM:dd}, \mathbf{hh:mm:ss:ms}, \mathbf{yy:MM:dd:hh:mm:ss:ms}\}$.

The Increase Process

Definition 27. The semantics of the increase process on Θ , $\theta(\uparrow(\mathbf{xRT}))$, is a double partial differential of the semantic function $f_0(\uparrow(\mathbf{xRT}))$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(\uparrow(\mathbf{xRT})) & \triangleq \frac{\partial^2}{\partial t \partial s} f_0(\uparrow(\mathbf{xRT})) \\
 & = \prod_{i=0}^{\#T(\uparrow(\mathbf{xRT}))} \left(\prod_{j=1}^{\#S(\uparrow(\mathbf{xRT}))} v(t_i, s_j) \right) \\
 & = \prod_{i=0}^1 \prod_{j=1}^1 v(t_i, s_j) \\
 & = \left(\begin{array}{cc} \mathbf{xRT} & \\ \mathbf{t}_0 & \mathbf{xRT} \\ (\mathbf{t}_0, \mathbf{t}_1] & \mathbf{xRT} + 1 \end{array} \right)
 \end{aligned} \tag{27}$$

where the run-time type $\mathbf{RT} = \{\mathbf{N}, \mathbf{Z}, \mathbf{B}, \mathbf{H}, \mathbf{P}, \mathbf{TM}\}$

The Decrease Process

Definition 28. The semantics of the decrease process on Θ , $\theta(\downarrow(\mathbf{xRT}))$, is a double partial differential of the semantic function $f_0(\downarrow(\mathbf{xRT}))$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(\downarrow(x\mathbf{RT})) &\triangleq \frac{\partial^2}{\partial t \partial S} f_0(\downarrow(x\mathbf{RT})) \\
 &= \prod_{i=0}^{\#T(\downarrow(x\mathbf{RT}))} \prod_{j=1}^{\#S(\downarrow(x\mathbf{RT}))} v(t_i, s_j) \\
 &= \prod_{i=0}^1 \prod_{j=1}^1 v(t_i, s_j) \\
 &= \begin{pmatrix} & \mathbf{xRT} \\ \mathbf{t}_0 & \mathbf{xRT} \\ (\mathbf{t}_0, \mathbf{t}_1] & \mathbf{xRT}-1 \end{pmatrix}
 \end{aligned} \tag{28}$$

where the run-time type $\mathbf{RT} = \{\mathbf{N}, \mathbf{Z}, \mathbf{B}, \mathbf{H}, \mathbf{P}, \mathbf{TM}\}$

The Exception Detection Process

Definition 29. The semantics of the exception detection process on Θ , $\theta(!(@)e\mathbf{S})$, is a double partial differential of the semantic function $f_0(!(@)e\mathbf{S})$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(!(@)e\mathbf{S}) &\triangleq \frac{\partial^2}{\partial t \partial S} f_0(!(@)e\mathbf{S}) \\
 &= \prod_{i=0}^{\#T(!(@)e\mathbf{S})} \left(\prod_{j=1}^{\#S(!(@)e\mathbf{S})} v(t_i, s_j) \right) \\
 &= \prod_{i=0}^1 \prod_{j=1}^3 v(t_i, s_j) \\
 &= \begin{pmatrix} & @e\mathbf{S} & \mathbf{ptrP} & \mathbf{PORT}(\mathbf{ptrP}\mathbf{S}) \\ \mathbf{t}_0 & @e\mathbf{S} & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & @e\mathbf{S} & \mathbf{ptrP} & @e\mathbf{S} \end{pmatrix}
 \end{aligned} \tag{29}$$

Equation 29 indicates that the semantics of exception detection is the output of a string $@e\mathbf{S}$ to a designated port $\mathbf{PORT}[\mathbf{ptrP}\mathbf{S}]$, where the pointer \mathbf{ptrP} points to a CRT or a printer. Therefore, the semantics of exception detection can be described based on the semantics of the output process as defined in Equation 24, i.e.:

$$\theta(!(@)e\mathbf{S}) = \theta(@e\mathbf{S} | \leftarrow \mathbf{PORT}[\mathbf{ptrP}\mathbf{S}]) \tag{30}$$

The Skip Process

Definition 30. The semantics of the skip process on Θ , $\theta(\otimes)$, is a double partial differential of the semantic function $f_0(\otimes)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(\otimes) &\triangleq \theta(P^k \curvearrowright P^{k-1}) \\
 &= \frac{\partial^2}{\partial t \partial S} f_0(P^k \curvearrowright P^{k-1}) \\
 &= \prod_{i=0}^{\#T(P^k \curvearrowright P^{k-1})} \left(\prod_{j=1}^{\#S(P^k \curvearrowright P^{k-1})} v(t_i, s_j) \right) \\
 &= \prod_{i=0}^1 \prod_{j=1}^2 v(t_i, s_j) \\
 &= \begin{pmatrix} & \mathbf{S}_{P^{k-1}} & \mathbf{S}_{P^k} \\ \mathbf{t}_0 & \mathbf{S}_{P^{k-1}} & \mathbf{S}_{P^k} \\ (\mathbf{t}_0, \mathbf{t}_1] & \mathbf{S}_{P^{k-1}} \setminus \mathbf{S}_{P^k} & \perp \end{pmatrix}
 \end{aligned} \tag{31}$$

where P^k is a process P at a given embedded layer k in a program with P^0 at the uttermost layer; and \curvearrowright denotes the jump process relation where its semantics will be formally defined in the next section.

According to Definition 30, the skip process \otimes has no semantic effect on the current process P^k at the given embedded layer k in a program, such as a branch, loop, or function. However, it redirects the system to jump to execute an upper-layer process P^{k-1} in the embedded hierarchy. Therefore, skip is also known as *exit* or *break* in programming languages.

The Stop Process

Definition 31. The semantics of the stop process on Θ , $\theta(\boxtimes)$, is a double partial differential of the semantic function $f_0(\boxtimes)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(\boxtimes) &\triangleq \theta(P \curvearrowright S) \\
 &= \frac{\partial^2}{\partial t \partial S} f_{\theta}(P \curvearrowright S) \\
 &= \underset{\#T(P \curvearrowright S)}{R} \left(\underset{\#S(P \curvearrowright S)}{R} v t_i, s_j \right) \\
 &= \underset{i=0}{R} \underset{j=1}{R} v(t_i, s_j) \\
 &= \begin{pmatrix} & \mathbf{S}_S & \mathbf{S}_P \\ \mathbf{t}_0 & S_S & S_P \\ (\mathbf{t}_0, \mathbf{t}_1] & S_S \setminus S_P & \perp \end{pmatrix} \tag{32}
 \end{aligned}$$

where the stop process \boxtimes does nothing but returns the control of execution to the system.

DEDUCTIVE SEMANTICS OF RTPA PROCESS RELATIONS

The preceding section provides formal definitions of metaprocesses of RTPA for software system modeling. Via the composition of multiple metaprocesses by the 17 process relations, $R = \{\rightarrow, \curvearrowright, |, | \dots |, \dots, R^*, R^+, R^i, \odot, \succ, \parallel, \{\!\!\}, \|\!\!\}, \gg, \ll, \llcorner, \lrcorner, \lrcorner, \lrcorner\}$, complex architectures and behaviors of software systems, in the most complicated case, a real-time system, can be sufficiently described (Wang, 2002, 2006a, 2007c, 2008b). On the basis of Definitions 8 and 12, the semantics of the RTPA process relations can be formally defined and analyzed as follows.

The Sequential Process Relation

Definition 32. *The semantics of the sequential relation of processes on Θ , $\theta(P \rightarrow Q)$, is a double partial differential of the semantic function $f_{\theta}(P \rightarrow Q)$ on the sets of variables S and executing steps T , i.e.:*

$$\begin{aligned}
 \theta(P \rightarrow Q) &\triangleq \frac{\partial^2}{\partial t \partial S} f_{\theta}(P \rightarrow Q) \\
 &= \frac{\partial^2}{\partial t \partial S} f_{\theta}(P) \rightarrow \frac{\partial^2}{\partial t \partial S} f_{\theta}(Q) \\
 &= \underset{\#T(P)}{R} \left(\underset{\#S(P)}{R} v_P t_i, s_j \right) \rightarrow \underset{\#T(Q)}{R} \left(\underset{\#S(Q)}{R} v_Q t_i, s_j \right) \\
 &= \underset{\#T(P \cup Q)}{R} \left(\underset{\#S(P \cup Q)}{R} v t_i, s_j \right) \\
 &= \begin{pmatrix} & \mathbf{S}_P & \mathbf{S}_Q & \mathbf{S}_{PQ} \\ \mathbf{t}_0 & \perp & \perp & \perp \\ (\mathbf{t}_0, \mathbf{t}_1] & V_{1P} & - & V_{1PQ} \\ (\mathbf{t}_1, \mathbf{t}_2] & - & V_{2Q} & V_{2PQ} \end{pmatrix} \\
 &= \begin{pmatrix} \mathbf{V}_P & \mathbf{V}_{PQ} \\ & \mathbf{V}_Q & \mathbf{V}_{PQ} \end{pmatrix} \tag{33}
 \end{aligned}$$

where $P \curvearrowright Q$ indicates a concatenation of these two processes over time, and in the simplified notation of the matrix, $V_P = v(t_P, s_P)$, $0 \leq t_P \leq n_P$, $1 \leq s_P \leq m_P$; $V_Q = v(t_Q, s_Q)$, $0 \leq t_Q \leq n_Q$, $1 \leq s_Q \leq m_Q$; and $V_{PQ} = v(t_{PQ}, s_{PQ})$, $0 \leq t_{PQ} \leq n_{PQ}$, $1 \leq s_{PQ} \leq m_{PQ}$.

In Equation 33, the first partial differential selects a set of related variables in the sequential processes P and Q , $S(P \cup Q)$. The second partial differential selects a set of time moments $T(P \cup Q)$. The semantic diagram of the sequential process relation as defined in Equation 33 is illustrated in Figure 3 on Θ .

The following example shows the physical meaning of Equation 33 and how the abstract syntaxes and their implied meanings are embodied onto the objects (variables) and their dynamic values in order to obtain the concrete semantics in deductive semantics.

Example 3. *Analyze the semantics of the sequential processes P_0 through P_5 in the following program:*

```

void sequential_sum;
{
    int x, y, z; // P0
}
    
```

```

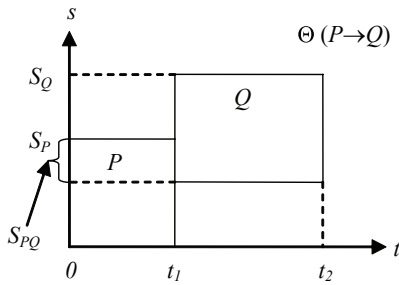
x = 2;           // P1
y = 8;           // P2
z := x + y; // P3
z := x + y + z; // P4
print z;        // P5
}
    
```

According to Definition 32, the semantics of the above program can be analyzed as seen in Box 6. Where $\text{PORT}[\text{CRTP}]\mathbf{N}$ denotes a system monitor of type \mathbf{N} located by the pointer CRTP .

The Jump Process Relation

Definition 33. The semantics of the jump relations of processes on Θ , $\theta(P \curvearrowright Q)$, is a double partial differential of the semantic function $f_\theta(P \curvearrowright Q)$ on the sets of variables S and executing steps T , i.e.:

Figure 3. The semantic diagram of the sequential process relation



Box 6.

$$\begin{aligned}
 \theta(P_0 \rightarrow P_1 \rightarrow \dots \rightarrow P_5) &= \frac{\partial^2}{\partial t \partial s} f_\theta(P_0 \rightarrow P_1 \rightarrow \dots \rightarrow P_5) \\
 &= \frac{\partial^2}{\partial t \partial s} f_\theta(P_0) \rightarrow \frac{\partial^2}{\partial t \partial s} f_\theta(P_1) \rightarrow \dots \rightarrow \frac{\partial^2}{\partial t \partial s} f_\theta(P_5) \\
 &= \underset{\#T(P_0) \#S(P_0)}{R} \left(\underset{j=1}{R} v_{P_0} t_i, s_j \right) \rightarrow \underset{\#T(P_1) \#S(P_1)}{R} \left(\underset{j=1}{R} v_{P_1} t_i, s_j \right) \rightarrow \dots \rightarrow \underset{\#T(P_5) \#S(P_5)}{R} \left(\underset{j=1}{R} v_{P_5} t_i, s_j \right) \\
 &= \underset{\#T(\widehat{P_0 P_1 \dots P_5}) \#S(P_0 \cup P_1 \cup \dots \cup P_5)}{R} \left(\underset{j=1}{R} v t_i, s_j \right) \\
 &= \underset{i=0}{R} \underset{j=1}{R} v(t_i, s_j) \\
 &= \left(\begin{array}{ccccc}
 & \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{PORT}[\mathbf{CRTP}]\mathbf{N} \\
 \mathbf{t}_0 & \perp & \perp & \perp & \perp \\
 (\mathbf{t}_0, \mathbf{t}_1) & 2 & \perp & \perp & \perp \\
 (\mathbf{t}_1, \mathbf{t}_2) & 2 & 8 & \perp & \perp \\
 (\mathbf{t}_2, \mathbf{t}_3) & 2 & 8 & 10 & \perp \\
 (\mathbf{t}_3, \mathbf{t}_4) & 2 & 8 & 20 & \perp \\
 (\mathbf{t}_4, \mathbf{t}_5) & 2 & 8 & 20 & 20
 \end{array} \right)
 \end{aligned}
 \tag{34}$$

Box 7.

$$\begin{aligned}
 \Theta(\diamond \text{expRT} \rightarrow P \mid \diamond \sim \rightarrow Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(\diamond \text{expRT} \rightarrow P \mid \diamond \sim \rightarrow Q) \\
 &= \diamond \text{expBL} \rightarrow \frac{\partial^2}{\partial t \partial s} f_0(P) \\
 &\quad \mid \diamond \rightarrow \frac{\partial^2}{\partial t \partial s} f_0(Q) \\
 &= \diamond \text{expBL} \rightarrow \overset{\#T(P) \#S(P)}{R} \left(R v_P t_i, s_j \right) \\
 &\quad \mid \diamond \rightarrow \overset{\#T(Q) \#S(Q)}{R} \left(R v_Q t_i, s_j \right) \\
 &= \begin{pmatrix} \text{expBL} & S_P & S_Q & S_{PQ} \\ (\mathbf{t}_0, \mathbf{t}_1] & \delta(\text{expBL}) & \perp & \perp & \perp \\ (\mathbf{t}_1, \mathbf{t}_2] & \mathbf{T} & V_{2P} & - & V_{2PQ} \\ (\mathbf{t}_1, \mathbf{t}_2'] & \mathbf{F} & - & V_{3Q} & V_{3PQ} \end{pmatrix} \tag{36}
 \end{aligned}$$

Figure 5. The semantic diagram of the branch process relation

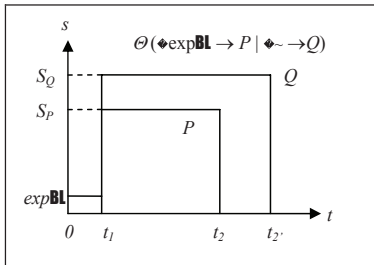
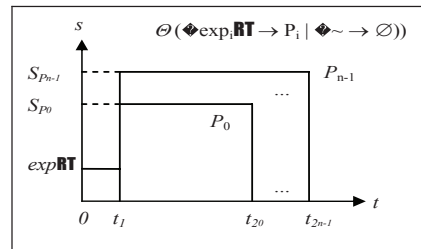


Figure 6. The semantic diagram of the switch process relation



where V_G is a set of global variables shared by P_0 , P_P and P_{n-1} .

The semantic diagram of the switch process relation as defined in Equation 37 is illustrated in Figure 6 on $\Theta(\diamond \text{exp}_i \text{RT} \rightarrow P_i \mid \diamond \sim \rightarrow \emptyset)$.

is a double partial differential of the semantic function

$$f_0 \left(\overset{\mathbf{F}}{R}_{\text{expBL}=\mathbf{T}}^*(P) \right)$$

on the sets of variables S and executing steps T , i.e.:

The While-Loop Process Relation

Definition 36. The semantics of the while-loop relations of processes on Θ ,

$$\Theta \left(\overset{\mathbf{F}}{R}_{\text{expBL}=\mathbf{T}}^*(P) \right),$$

$$\begin{aligned}
 \theta(\underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^*}}(P)) &\triangleq \frac{\partial^2}{\partial t \partial s} f_{\theta}(\underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^*}}(P)) \\
 &= \underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^*}}\left(\frac{\partial^2}{\partial t \partial s} f_{\theta}(P)\right) \\
 &= \underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^*}}\left(\underset{i=0}{\overset{\#T(P)}{R}} \underset{j=1}{\overset{\#S(P)}{R}} v_p(t_i, s_j)\right) \\
 &= \left(\begin{array}{ccc} & \text{expBL} & S_p \\ [t_0, t_1] & \delta(\text{expBL}) & \perp \\ (t_1, t_2] & \mathbf{T} & V_p \\ (t_1, t_2] & \mathbf{F} & \otimes \\ \vdots & \vdots & \vdots \\ (t_3, t_4] & \delta(\text{expBL}) & - \\ (t_4, t_5] & \mathbf{T} & V_p \\ (t_4, t_5] & \mathbf{F} & \otimes \end{array} \right) \quad (38)
 \end{aligned}$$

where \emptyset denotes exit, and $\delta(\text{expBL})$ is the evaluation function on the Boolean expression, $\delta(\text{expBL}) \in \{\mathbf{T}, \mathbf{F}\}$.

The semantic diagram of the while-loop process relation as defined in Equation 38 is illustrated in Figure 7 on Θ .

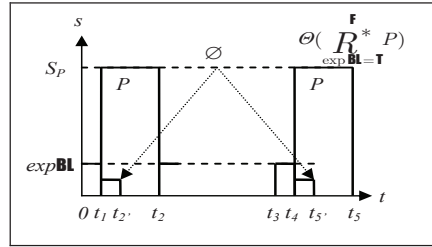
The Repeat-Loop Process Relation

Definition 37. The semantics of the repeat-loop relations of processes on Θ ,

$\theta(\underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^+}}(P))$,
 is a double partial differential of the semantic function

$f_{\theta}(\underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^+}}(P))$
 on the sets of variables S and executing steps T , i.e.:

Figure 7. The semantic diagram of the while-loop process relation



$$\begin{aligned}
 \theta(\underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^+}}(P)) &\triangleq \frac{\partial^2}{\partial t \partial s} f_{\theta}(\underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^+}}(P)) \\
 &= \underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^+}}\left(\frac{\partial^2}{\partial t \partial s} f_{\theta}(P)\right) \\
 &= P \rightarrow \underset{\text{expBL}=\mathbf{T}}{\overset{\mathbf{F}}{R^*}}\left(\underset{i=0}{\overset{\#T(P)}{R}} \underset{j=1}{\overset{\#S(P)}{R}} v_p(t_i, s_j)\right) \\
 &= \left(\begin{array}{ccc} & \text{expBL} & S_p \\ [t_0, t_1] & \perp & V_p \\ (t_1, t_2] & \delta(\text{expBL}) & - \\ (t_2, t_3] & \mathbf{T} & V_p \\ (t_2, t_3] & \mathbf{F} & \otimes \\ \vdots & \vdots & \vdots \\ (t_4, t_5] & \delta(\text{expBL}) & - \\ (t_5, t_6] & \mathbf{T} & V_p \\ (t_5, t_6] & \mathbf{F} & \otimes \end{array} \right) \quad (39)
 \end{aligned}$$

The semantic diagram of the repeat-loop process relation as defined in Equation 39 is illustrated in Figure 8 on Θ .

The For-Loop Process Relation

Definition 38. The semantics of the for-loop relations of processes on Θ ,

$$\theta(\underset{i=1}{\overset{n}{R}}P(i)),$$

Figure 8. The semantic diagram of the repeat-loop process relation

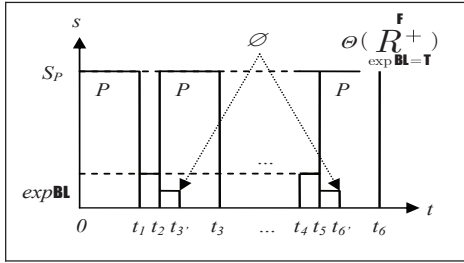
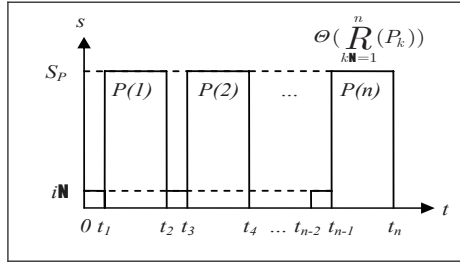


Figure 9. The semantic diagram of the for-loop process relation



is a double partial differential of the semantic function

$$f_0(\overset{n}{R}_{\text{iN}=1}^P(i))$$

on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} \Theta(\overset{n}{R}_{\text{iN}=1}^P(i)) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(\overset{n}{R}_{\text{iN}=1}^P(i)) \\ &= \overset{n}{R}_{\text{kN}=1} \left(\frac{\partial^2}{\partial t \partial s} f_0(P_i) \right) \\ &= \overset{n}{R}_{\text{kN}=1} \left(\overset{\#T(P_k)}{R}_{i=0} \overset{\#S(P_k)}{R}_{j=1} v_{P_k}(t_i, s_j) \right) \\ &= \begin{pmatrix} & \text{kN} & S_P \\ [t_0, t_1] & 1 & \perp \\ (t_1, t_2] & 1 & V_P \\ \vdots & \vdots & \vdots \\ (t_{n-2}, t_{n-1}] & n & - \\ (t_{n-1}, t_n] & n & V_P \end{pmatrix} \end{aligned} \tag{40}$$

The semantic diagram of the for-loop process relation as defined in Equation 40 is illustrated in Figure 9 on Θ .

The Function Call Process Relation

Definition 39. The semantics of the function call relations of processes on Θ , $\theta(P \mapsto Q)$,

is a double partial differential of the semantic function $f_0(P \mapsto Q)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} \theta(P \mapsto Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(P \mapsto Q) \\ &= \frac{\partial^2}{\partial t \partial s} f_0(P) \mapsto \frac{\partial^2}{\partial t \partial s} f_0(Q) \\ &= \overset{\#T(P)}{R}_{i=0} \overset{\#S(P)}{R}_{j=1} v_P(t_i, s_j) \mapsto \overset{\#T(Q)}{R}_{i=0} \overset{\#S(Q)}{R}_{j=1} v_Q(t_i, s_j) \\ &= \overset{\#T([t_0, t_1] \widehat{(t_1, t_2]} \widehat{(t_2, t_3)})}{R}_{i=0} \overset{\#S(P \cup Q)}{R}_{j=1} (v_{t_i, s_j}) \\ &= \begin{pmatrix} & S_P & S_Q & S_{PQ} \\ t_0 & \perp & \perp & \perp \\ (t_0, t_1] & V_{IP} & - & V_{IPQ} \\ (t_1, t_2] & - & v_{2Q} & V_{2PQ} \\ (t_2, t_3] & V_{3P} & - & V_{3PQ} \end{pmatrix} \end{aligned} \tag{41}$$

The semantic diagram of the procedure call process relation as defined in Equation 41 is illustrated in Figure 10 on $\Theta(P \mapsto Q)$.

The Recursive Process Relation

Definition 40. The semantics of the recursive relations of processes on Θ , $\theta(P \circ P)$, is a double partial differential of the semantic function $f_0(P \circ P)$ on the sets of variables S and executing steps T , i.e.:

Figure 10. The semantic diagram of the function call process relation

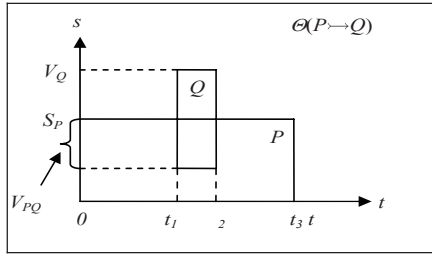
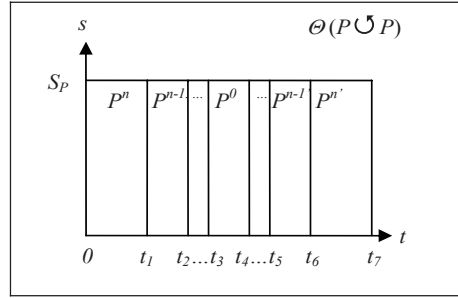


Figure 11. The semantic diagram of the recursive call process relation



$$\begin{aligned}
 \theta(P \cup P) &\triangleq \frac{\partial^2}{\partial t \partial s} f_{\theta}(P \cup P) \\
 &= \frac{\partial^2}{\partial t \partial s} f_{\theta}(P) \cup \frac{\partial^2}{\partial t \partial s} f_{\theta}(P) \\
 &= \mathbf{R}_{i=0}^{\#T(P)} \left(\mathbf{R}_{j=1}^{\#S(P)} v_{t_i, s_j} \right) \cup \mathbf{R}_{i=0}^{\#T(P)} \left(\mathbf{R}_{j=1}^{\#S(P)} v_{t_i, s_j} \right) \\
 &= \mathbf{R}_{i=0}^{\#T(P)} \left(\mathbf{R}_{j=1}^{\#S(P)} v_{t_i, s_j} \right) \\
 &= \begin{pmatrix} \mathbf{S}_P \\ [t_0, t_1] & V_{P^n} \\ (t_1, t_2] & V_{P^{n-1}} \\ \vdots & \vdots \\ (t_3, t_4] & V_{P^0} \\ \vdots & \vdots \\ (t_5, t_6] & V_{P^{n-1}} \\ (t_6, t_7] & V_{P^n} \end{pmatrix}
 \end{aligned} \tag{42}$$

$$\begin{aligned}
 \theta(P \parallel Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_{\theta}(P \parallel Q) \\
 &= \frac{\partial^2}{\partial t \partial s} f_{\theta}(P) \parallel \frac{\partial^2}{\partial t \partial s} f_{\theta}(Q) \\
 &= \mathbf{R}_{i=0}^{\#T(P)} \left(\mathbf{R}_{j=1}^{\#S(P)} v_{t_i, s_j} \right) \parallel \mathbf{R}_{i=0}^{\#T(Q)} \left(\mathbf{R}_{j=1}^{\#S(Q)} v_{t_i, s_j} \right) \\
 &= \mathbf{R}_{i=0}^{\max(\#T(P), \#T(Q))} \left(\mathbf{R}_{j=1}^{\#S(P \cup Q)} v_{t, s} \right) \\
 &= \begin{pmatrix} \mathbf{S}_P & \mathbf{S}_Q & \mathbf{S}_{P \cup Q} \\ \mathbf{t}_0 & V_{0P} & V_{0Q} & V_{0P \cup Q} \\ (t_0, t_1] & V_{1P} & V_{1Q} & V_{1P \cup Q} \\ (t_1, t_2] & - & V_{2Q} & V_{2P \cup Q} \end{pmatrix}
 \end{aligned} \tag{43}$$

where $t_2 = \max(\#T(P), \#T(Q))$ is the synchronization point between two parallel processes.

The semantic diagram of the recursive process relation as defined in Equation 42 is illustrated in Figure 11 on $\Theta(P \cup P)$.

The Parallel Process Relation

Definition 41. The semantics of the parallel relations of processes on Θ , $\theta(P \parallel Q)$, is a double partial differential of the semantic function $f_{\theta}(P \parallel Q)$ on the sets of variables S and executing steps T , i.e.:

The semantic diagram of the parallel process relation as defined in Equation 43 is illustrated in Figure 12 on $\Theta(P \parallel Q)$.

It is noteworthy that parallel processes P and Q are interlocked. That is, they should start and end at the same time. In case $t_1 \neq t_2$, the process completed earlier, should wait for the completion of the other. The second condition between parallel processes is that the shared resources, in particular variables, memory space, ports, and devices should be protected. That is, when a process operates on a shared resource, it is locked to the other process until

Figure 12. The semantic diagram of the parallel process relation

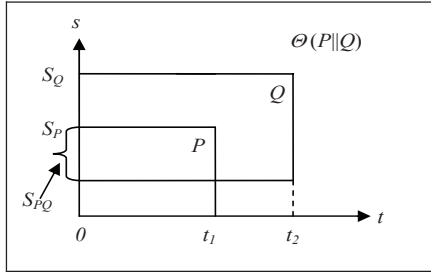
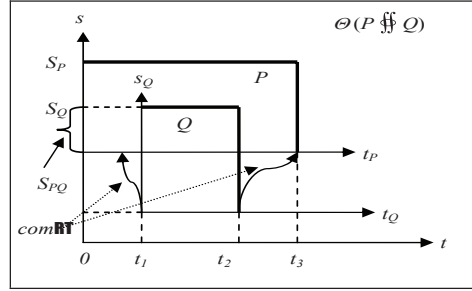


Figure 13. The semantic diagram of the concurrent process relation



the operation is completed. A variety of interlocking and synchronization techniques, such as *semaphores*, *mutual exclusions*, and *critical regions*, have been proposed in real-time system techniques (McDermid, 1991).

The Concurrent Process Relation

Definition 42. The semantics of the concurrent relations of processes on Θ , $\Theta(P \bowtie Q)$, is a double partial differential of the semantic function $f_0(P \bowtie Q)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} \Theta(P \bowtie Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(P \bowtie Q) \\ &= \frac{\partial^2}{\partial t \partial s} f_0(P) \bowtie \frac{\partial^2}{\partial t \partial s} f_0(Q) \\ &= \underset{\#T(P) \#S(P)}{R} \left(\underset{j=1}{R} v_P t_i, s_j \right) \bowtie \underset{\#T(Q) \#S(Q)}{R} \left(\underset{j=1}{R} v_Q t_i, s_j \right) \\ &= \underset{\max(\#T(P), \#T(Q)) \#S(P \cup Q)}{R} \left(\underset{j=1}{R} v t_i, s_j \right) \\ &= \begin{pmatrix} & S_P & S_Q & S_{PQ} & \text{comRT} \\ \mathbf{t}_0 & V_{0P} & V_{0P} & V_{0P} & V_{0com} \\ (\mathbf{t}_0, \mathbf{t}_1] & V_{1P} & - & V_{1PQ} & V_{1com} \\ (\mathbf{t}_1, \mathbf{t}_2] & V_{2P} & V_{2Q} & V_{2PQ} & V_{2com} \\ (\mathbf{t}_2, \mathbf{t}_3] & V_{3P} & - & V_{3PQ} & V_{3com} \end{pmatrix} \end{aligned} \quad (44)$$

where *comRT* is a set of interprocess communication variables that are used to synchronize P and Q executing on different machines based on independent system clocks.

The semantic diagram of the concurrent process relation as defined in Equation 44 is illustrated in Figure 13 on $\Theta(P \bowtie Q)$.

The Interleave Process Relation

Definition 43. The semantics of the interleave relations of processes on Θ , $\Theta(P \parallel Q)$, is a double partial differential of the semantic function $f_0(P \parallel Q)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} \Theta(P \parallel Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(P \parallel Q) \\ &= \frac{\partial^2}{\partial t \partial s} f_0(P) \parallel \frac{\partial^2}{\partial t \partial s} f_0(Q) \\ &= \underset{\#T(P) \#S(P)}{R} \left(\underset{j=1}{R} v_P t_i, s_j \right) \parallel \underset{\#T(Q) \#S(Q)}{R} \left(\underset{j=1}{R} v_Q t_i, s_j \right) \\ &= \underset{\#T(\mathbf{t}_0, \mathbf{t}_1] \widehat{(\mathbf{t}_1, \mathbf{t}_2]} \widehat{(\mathbf{t}_2, \mathbf{t}_3]} \widehat{(\mathbf{t}_3, \mathbf{t}_4]} \widehat{(\mathbf{t}_4, \mathbf{t}_5]}) \#S(P \cup Q)}{R} \left(\underset{j=1}{R} v t_i, s_j \right) \\ &= \begin{pmatrix} & S_P & S_Q & S_{PQ} \\ \mathbf{t}_0 & V_{0P} & V_{0Q} & V_{0PQ} \\ (\mathbf{t}_0, \mathbf{t}_1] & V_{1P} & - & V_{1PQ} \\ (\mathbf{t}_1, \mathbf{t}_2] & - & V_{2Q} & V_{2PQ} \\ (\mathbf{t}_2, \mathbf{t}_3] & V_{3P} & - & V_{3PQ} \\ (\mathbf{t}_3, \mathbf{t}_4] & - & V_{4Q} & V_{4PQ} \\ (\mathbf{t}_4, \mathbf{t}_5] & V_{5P} & - & V_{5PQ} \end{pmatrix} \end{aligned} \quad (45)$$

The semantic diagram of the interleave process relation as defined in Equation 45 is illustrated in Figure 14 on $\Theta(P \parallel Q)$.

The Pipeline Process Relation

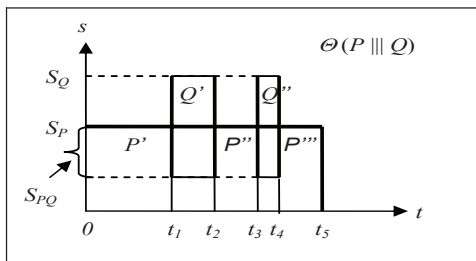
Definition 44. The semantics of the pipeline relations of processes on Θ , $\theta(P \gg Q)$, is a double partial differential of the semantic function $f_\theta(P \gg Q)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} \theta(P \gg Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_\theta(P \gg Q) \\ &= \frac{\partial^2}{\partial t \partial s} f_\theta(P) \gg \frac{\partial^2}{\partial t \partial s} f_\theta(Q) \\ &= \underset{\#T(P) \#S(P)}{\mathbf{R}} \underset{\#S(Q)}{\mathbf{R}} v_P(t_i, s_j) \gg \underset{\#T(Q) \#S(Q)}{\mathbf{R}} \underset{\#S(Q)}{\mathbf{R}} v_Q(t_i, s_j) \\ &= \underset{\#T(PQ) \#S(P \cup Q)}{\mathbf{R}} \left(\underset{j=1}{\mathbf{R}} v_{t_i, s_j} \right) \\ &= \begin{pmatrix} \mathbf{S}_P & \mathbf{S}_{P_o} = \mathbf{S}_{Q_i} & \mathbf{S}_Q \\ \mathbf{t}_0 & V_{0P} & V_{0PQ} & V_{0Q} \\ (t_0, t_1] & V_{1P} & V_{1PQ} & - \\ (t_1, t_2] & - & V_{2PQ} & V_{2Q} \end{pmatrix} \end{aligned} \tag{46}$$

where \mathbf{S}_{P_o} and \mathbf{S}_{Q_i} denote a set of n one-to-one connections between the outputs of P and inputs of Q , respectively, as follows:

$$\underset{k=0}{\mathbf{R}}^{n-1} (P_o(i) = Q_i(i)) \tag{47}$$

Figure 14. The semantic diagram of the interleave process relation



The semantic diagram of the pipeline process relation as defined in Equation 46 is illustrated in Figure 15 on $\Theta(P \gg Q)$.

The Interrupt Process Relation

Definition 45. The semantics of the interrupt relations of processes, $\theta(P \not\llcorner Q)$, on a given semantic environment Θ is a double partial differential of the semantic function $f_\theta(P \not\llcorner Q)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned} \theta(P \not\llcorner Q) &\triangleq \frac{\partial^2}{\partial t \partial s} f_\theta(P \not\llcorner Q) \\ &= \frac{\partial^2}{\partial t \partial s} f_\theta(P) \not\llcorner \frac{\partial^2}{\partial t \partial s} f_\theta(Q) \\ &= \underset{\#T(P) \#S(P)}{\mathbf{R}} \underset{\#S(P)}{\mathbf{R}} v_P(t_i, s_j) \not\llcorner \underset{\#T(Q) \#S(Q)}{\mathbf{R}} \underset{\#S(Q)}{\mathbf{R}} v_Q(t_i, s_j) \\ &= \underset{\#T(P \hat{\llcorner} Q \hat{\llcorner} P^*) \#S(P \cup Q)}{\mathbf{R}} \left(\underset{j=1}{\mathbf{R}} v_{t_i, s_j} \right) \\ &= \begin{pmatrix} \mathbf{S}_P & \mathbf{S}_Q & \mathbf{S}_{PQ} & \mathbf{int} \odot \\ [t_0, t_1] & V_{1P^*} & \perp & V_{1PQ} & \perp \\ (t_1, t_2] & - & - & V_{2PQ} & V_{2int \odot} \\ (t_2, t_3] & - & V_{3Q} & V_{3PQ} & - \\ (t_3, t_4] & - & - & V_{4PQ} & V_{4int \odot} \\ (t_4, t_5] & V_{5P^*} & - & V_{5PQ} & - \end{pmatrix} \end{aligned} \tag{48}$$

The semantic diagram of the interrupt process relation as defined in Equation 48 is illustrated in Figure 16 on $\Theta(P \not\llcorner Q)$, where

Figure 15. The semantic diagram of the pipeline process relation

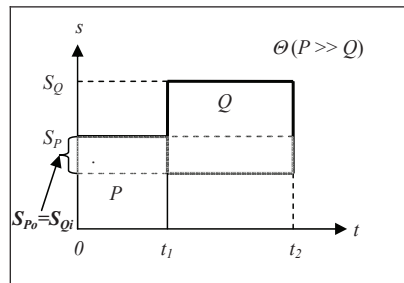
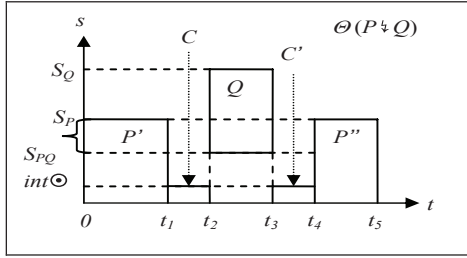


Figure 16. The semantic diagram of the interrupt process relation



$C(int'⊙)$ and $C'(int'⊙)$ are the interrupt and interrupt-return points, respectively.

The deductive semantics of the three system dispatch process relations will be presented in the next section.

DEDUCTIVE SEMANTICS OF SYSTEM-LEVEL PROCESSES OF RTPA

The deductive semantics of systems at the top level of programs can be reduced onto a dispatch mechanism of a finite set of processes based on the mechanisms known as time, event, and interrupt. This section first describes the deductive semantics of the system process. Then, the three system dispatching processes will be formally modeled.

The System Process

Definition 46. The semantics of the system process ξ on Θ , $\theta(\xi)$, is an abstract logical model of the executing platform with a set of parallel dispatched processes based on internal system clock, external events, and system interrupts, i.e.:

$$\begin{aligned} \theta(\xi) &\triangleq \frac{\partial^2}{\partial t \partial s} f_0(\xi) \\ &= \frac{\partial^2}{\partial t \partial s} f_0 \left\{ \begin{aligned} &R_{i\mathbf{N}=0}^{n_i\mathbf{N}-1} (@ e_i \mathbf{s} \mapsto P_i) \\ &\parallel (R_{j\mathbf{N}=0}^{n_j\mathbf{N}-1} @ t_j \mathbf{TM} \mapsto P_j) \\ &\parallel (R_{k\mathbf{N}=0}^{n_{int}\mathbf{N}-1} @ int_k \mathbf{s} \mapsto P_k) \end{aligned} \right\} \\ &= \overset{\mathbf{T}}{R}_{SysShutDown\mathbf{BL}=\mathbf{F}} \left\{ \begin{aligned} &R_{i\mathbf{N}=0}^{n_i\mathbf{N}-1} (@ e_i \mathbf{s} \mapsto P_i) \\ &\parallel (R_{j\mathbf{N}=0}^{n_j\mathbf{N}-1} @ t_j \mathbf{TM} \mapsto P_j) \\ &\parallel (R_{k\mathbf{N}=0}^{n_{int}\mathbf{N}-1} @ int_k \mathbf{s} \mapsto P_k) \end{aligned} \right\} \end{aligned} \tag{49}$$

where the semantics of the parallel relations has been given in Definition 41, and those of the system dispatch processes will be described in the following subsections.

The Time-Driven Dispatching Process Relation

Definition 47. The semantics of the time-driven dispatching relations of processes on Θ , $\theta(@ t_k \mathbf{TM} \mapsto P_k)$, is a double partial differential of the semantic function $f_0(@ t_k \mathbf{TM} \mapsto P_k)$ on the sets of variables S and executing steps T , i.e.:

$$\begin{aligned}
 \theta(@t_k \mathbf{TM} \hookrightarrow_k P_k) &\triangleq \frac{\partial^2}{\partial t \partial S} f_\theta(@t_k \mathbf{TM} \hookrightarrow_k P_k) \\
 &= \prod_{k=1}^n (@t_k \mathbf{TM} \rightarrow \frac{\partial^2}{\partial t \partial S} f_\theta(P_k)) \\
 &= \prod_{k=1}^n (@t_k \mathbf{TM} \rightarrow \prod_{i=0}^{\#T(P_k)} \prod_{j=1}^{\#S(P_k)} v_{P_k}(t_i, s_j)) \\
 &= \diamond @t_1 \mathbf{TM} \rightarrow \prod_{i=0}^{\#T(P_1)} \prod_{j=1}^{\#S(P_1)} v_{P_1}(t_i, s_j) \\
 &\quad | \dots \\
 &\quad | \diamond @t_n \mathbf{TM} \rightarrow \prod_{i=0}^{\#T(P_n)} \prod_{j=1}^{\#S(P_n)} v_{P_n}(t_i, s_j) \\
 &= \begin{pmatrix} @t_k \mathbf{TM} & S_{P_1} & \dots & S_{P_n} \\ [t_0, t_1] & \delta(@t_k \mathbf{TM}) & \perp & \dots & \perp \\ (t_1, t_2] & @t_l & V_{P_1} & \dots & - \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (t_1, t_n] & @t_n & - & \dots & V_{P_n} \end{pmatrix} \\
 &\hspace{10em} (50)
 \end{aligned}$$

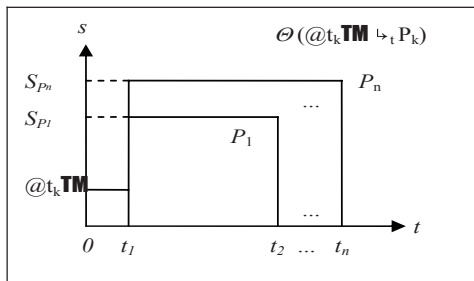
where $\diamond(@t_k \mathbf{TM}) = \diamond(@t_k \mathbf{N})$ is the evaluation function as defined in Equation 17b.

The semantic diagram of the time-driven dispatching process relation as defined in Equation 50 is illustrated in Figure 17 on Θ .

The Event-Driven Dispatching Process Relation

Definition 48. The semantics of the event-driven dispatching relations of processes on Θ , $\theta(@e_k \mathbf{S} \hookrightarrow_e P_k)$, is a double partial differential of the semantic function $f_\theta(@e_k \mathbf{S} \hookrightarrow_e P_k)$ on the sets of variables S and executing steps T , i.e.:

Figure 17. The semantic diagram of time-driven dispatch relation



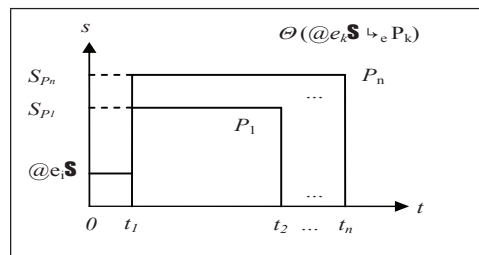
$$\begin{aligned}
 \theta(@e_k \mathbf{S} \hookrightarrow_e P_k) &\triangleq \frac{\partial^2}{\partial t \partial S} f_\theta(@e_k \mathbf{S} \hookrightarrow_e P_k) \\
 &= \prod_{k=1}^n (@e_k \mathbf{S} \rightarrow \frac{\partial^2}{\partial t \partial S} f_\theta(P_k)) \\
 &= \prod_{k=1}^n (@e_k \mathbf{S} \rightarrow \prod_{i=0}^{\#T(P_k)} \prod_{j=1}^{\#S(P_k)} v_{P_k}(t_i, s_j)) \\
 &= \diamond @e_1 \mathbf{S} \rightarrow \prod_{i=0}^{\#T(P_1)} \prod_{j=1}^{\#S(P_1)} v_{P_1}(t_i, s_j) \\
 &\quad | \dots \\
 &\quad | \diamond @e_n \mathbf{S} \rightarrow \prod_{i=0}^{\#T(P_n)} \prod_{j=1}^{\#S(P_n)} v_{P_n}(t_i, s_j) \\
 &= \begin{pmatrix} @e_k \mathbf{S} & S_{P_1} & \dots & S_{P_n} \\ [t_0, t_1] & \delta(@e_k \mathbf{S}) & \perp & \dots & \perp \\ (t_1, t_2] & @e_l & V_{P_1} & \dots & - \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (t_1, t_n] & @e_n & - & \dots & V_{P_n} \end{pmatrix} \\
 &\hspace{10em} (51)
 \end{aligned}$$

The semantic diagram of the event-driven process relation as defined in Equation 51 is illustrated in Figure 18 on Θ .

The Interrupt-Driven Dispatching Process Relation

Definition 49. The semantics of the interrupt-driven dispatching relations of processes on Θ , $\theta(@int_k \mathbf{S} \hookrightarrow_i P_k)$, is a double partial differential of the semantic function $f_\theta(@int_k \mathbf{S} \hookrightarrow_i P_k)$ on the sets of variables S and executing steps T , i.e.:

Figure 18. The semantic diagram of the event-driven dispatch relation



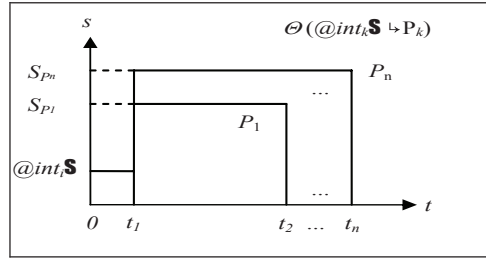
$$\begin{aligned}
 \Theta(@int_k \mathbf{S} \vdash P_k) &\triangleq \frac{\partial^2}{\partial t \partial S} f_0(@int_k \mathbf{S} \vdash P_k) \\
 &= \prod_{k=1}^n (@int_k \mathbf{S} \rightarrow \frac{\partial^2}{\partial t(P_k) \partial S(P_k)} f_0(P_k)) \\
 &= \prod_{k=1}^n (@int_k \mathbf{S} \rightarrow \prod_{i=0}^{\#T(P_k)} \prod_{j=1}^{\#S(P_k)} R v_{P_k}(t_i, s_j)) \\
 &= @int_1 \mathbf{S} \rightarrow \prod_{i=0}^{\#T(P_1)} \prod_{j=1}^{\#S(P_1)} (R v_{P_1} t_i, s_j) \\
 &| \dots \\
 &| @int_n \mathbf{S} \rightarrow \prod_{i=0}^{\#T(P_n)} \prod_{j=1}^{\#S(P_n)} R v_{P_n}(t_i, s_j) \\
 &= \left(\begin{array}{c|ccc} @int_k \mathbf{S} & S_{P_1} & \dots & S_{P_n} \\ \hline [t_0, t_1] & \delta(@int_k \mathbf{S}) & \perp & \dots & \perp \\ (t_1, t_2] & @e_j & V_{P_1} & \dots & - \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (t_1, t_n] & @e_n & - & \dots & V_{P_n} \end{array} \right) \tag{52}
 \end{aligned}$$

The semantic diagram of the interrupt-driven process relation as defined in Equation 52 is illustrated in Figure 19 on Θ .

CONCLUSION

Semantics plays an important role in cognitive informatics, computational linguistics, computing, and software engineering theories. Deductive semantics is a formal software semantics that deduces the semantics of a program in a given programming language from a generic abstract semantic function to the concrete semantics, which are embodied by the changes of statuses of a finite set of variables constituting the semantic environment of computing. Based on the mathematical models and architectural properties of programs at different composing levels, deductive models of software semantics, semantic environment, and semantic matrix have been formally defined. Properties of software semantics and relations between the software behavioral space and semantic environment have been discussed. Case studies on the deductive semantic rules of RTPA have been presented, which serve not only as a comprehensive paradigm, but also the verification of the expressive and analytic capacity of deductive semantics.

Figure 19. The semantic diagram of the interrupt-driven dispatch relation



A rigorous treatment of deductive semantics of RTPA has been presented in this article, which enables a new approach towards deductive reasoning of software semantics at all composing levels of the program hierarchy (i.e., statements, processes, and programs) from the bottom up. Deductive semantics has greatly simplified the description and analysis of the semantics of complicated software systems implemented in programming languages or specified in formal notations. Deductive semantics can be used to define both abstract and concrete semantics of large-scale software systems, facilitate software comprehension and recognition, support tool development, enable semantics-based software testing and verification, and explore the semantic complexity of software systems.

ACKNOWLEDGMENT

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

REFERENCES

Bjorner, D. (2000, November). Pinnacles of software engineering: 25 years of formal methods. In Y. Wang & D. Patel (Eds.), *Annals of software engineering: An international journal*, 10, 11-66.

Bjorner, D., & Jones, C. B. (1982). *Formal specification and software development*. Englewood Cliffs, NJ: Prentice Hall.

- Chomsky, N. (1956). Three models for the description of languages. *I.R.E. Transactions on Information Theory*, 2(3), 113-124.
- Chomsky, N. (1957). *Syntactic structures*. The Hague, The Netherlands: Mouton.
- Chomsky, N. (1982). *Some concepts and consequences of the theory of government and binding*. Cambridge, MA: MIT Press
- Dijkstra, E. W. (1975). Guarded commands, nondeterminacy, and the formal derivation of programs. *Communications of the ACM*, 18(8), 453-457.
- Dijkstra, E. W. (1976). *A discipline of programming*. Englewood Cliffs, NJ: Prentice Hall.
- Goguen, J. A., & Malcolm, G. (1996). *Algebraic semantics of imperative programming*. Cambridge, MA: MIT Press.
- Goguen, J.A., Thatcher, J. W., Wagner, E. G., & Wright, J. B. (1977). Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24(1), 68-59.
- Gries, D. (1981). *The science of programming*. Berlin, Germany: Spinger Verlag
- Gunter, C. A. (1992). Semantics of programming languages: Structures and techniques. In M. Garey & A. Meyer (Eds.), *Foundations of computing*. Cambridge, MA: MIT Press.
- Guttag, J. V., & Horning, J. J. (1978). The algebraic specification of abstract data types. *Acta Informatica*, 10, 27-52.
- Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10), 576-580.
- Jones, C.B. (1980). *Software development: A rigorous approach*. London: Prentice Hall International.
- Louden, K. C. (1993). *Programming languages: Principles and practice*. Boston: PWS-Kent.
- Marcotty, M., & Ledgard, H. (1986). *Programming language landscape* (2nd ed.). Chicago: SRA.
- McDermid, J. A. (Ed.). (1991). *Software engineer's reference book*. Oxford, England: Butterworth-Heinemann.
- Meyer, B. (1990). *Introduction to the theory of programming languages*. Englewood Cliffs, NJ: Prentice Hall
- Ollongren, A. (1974). *Definition of programming languages by interpreting automata*. New York: Academic Press.
- Pagan, F. G. (1981). *Semantics of programming languages: A panoramic primer*. Englewood Cliffs, NJ: Prentice Hall.
- Schmidt, D. (1988). *Denotational semantics: A methodology for language development*. Dubuque, IA: Brown.
- Schmidt, D. (1996, March). Programming language semantics. *ACM Computing Surveys*, 28(1).
- Schmidt, D. A. (1994). *The structure of typed programming languages*. Cambridge, MA: MIT Press.
- Scott, D. (1982). Domains for denotational semantics. In *Automata, languages and programming IX* (pp. 577-613). Berlin, Germany: Springer Verlag.
- Scott, D. S., & Strachey, C. (1971). *Towards a mathematical semantics for computer languages*. (Programming Research Group Tech. Rep. PRG-1-6). Oxford University.
- Slonneg, K., & Kurts, B. (1995). *Formal syntax and semantics of programming languages*. Addison-Wesley.
- Tarski, A. (1944). The semantic conception of truth. *Philosophic Phenomenological Research*, 4, 13-47.
- 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). Cognitive informatics and contemporary mathematics for knowledge representation and manipulation (Invited plenary talk). In *Proceedings of the First International Conference on Rough Set and Knowledge Technology (RSKT'06)* (LNAI 4062, pp. 69-78). Chongqing, China: Springer.
- Wang, Y. (2006b). On the informatics laws and deductive semantics of software, *IEEE Transac-*

tions on Systems, Man, and Cybernetics (C), 36(2), 161-171.

Wang, Y. (2007a). The Cognitive Processes of Formal Inferences. *The International Journal of Cognitive Informatics and Natural Intelligence*, 1(4), 75-86.

Wang, Y. (2007b). Keynote speech on theoretical foundations of software engineering and denotational mathematics. In *Proceedings of the Fifth Asian Workshop on Foundations of Software* (pp. 99-102). Xiamen, China:

Wang, Y. (2007c). *Software engineering foundations: A software science perspective*. Auerbach Publications, NY., July.

Wang, Y. (2008a). On the big-R notation for describing iterative and recursive behaviors. *The International Journal of Cognitive Informatics and Natural Intelligence*, 2(1), 17-28.

Wang, Y. (2008b, April). RTPA: A denotational mathematics for manipulating intelligent and computational behaviors. *The International Journal of Cognitive Informatics and Natural Intelligence*, 2(2), 44-62.

Wegner, P. (1972). The Vienna definition language. *ACM Computing Surveys*, 4(1), 5-63.

Wikstrom, A. (1987). *Functional programming using standard ML*. Englewood Cliffs, NJ: Prentice Hall.

Yingxu Wang is professor of cognitive informatics and software engineering, director of 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 Department of Computer Science at Stanford University during 1995 and 2008, respectively, and has been a full professor since 1994. He is the editor-in-chief of the International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), and editor-in-chief of the 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 groundbreaking book on Software Engineering Foundations: A Software Science Perspective.