Paradigms of Denotational Mathematics for Cognitive Informatics and Cognitive Computing

Yingxu Wang*

Visiting Professor, Dept. of Computer Science, Stanford University
Stanford, CA 94305-9010, USA
yingxuw@stanford.edu

International Center for Cognitive Informatics (ICfCI)
Theoretical and Empirical Software Engineering Research Centre (TESERC)
Dept. of Electrical and Computer Engineering, Schulich School of Engineering
University of Calgary, 2500 University Drive, NW, Calgary, Alberta, Canada T2N 1N4
yingxu@ucalgary.ca

Abstract. The abstract, rigorous, and expressive needs in cognitive informatics, intelligence science, software science, and knowledge science lead to new forms of mathematics collectively known as denotational mathematics. Denotational mathematics is a category of expressive mathematical structures that deals with high level mathematical entities beyond numbers and sets, such as abstract objects, complex relations, behavioral information, concepts, knowledge, processes, and systems. Denotational mathematics is usually in the form of abstract algebra that is a branch of mathematics in which a system of abstract notations is adopted to denote relations of abstract mathematical entities and their algebraic operations based on given axioms and laws. Four paradigms of denotational mathematics, known as concept algebra, system algebra, Real-Time Process Algebra (RTPA), and Visual Semantic Algebra (VSA), are introduced in this paper. Applications of denotational mathematics in cognitive informatics and computational intelligence are elaborated. Denotational mathematics is widely applicable to model and manipulate complex architectures and behaviors of both humans and intelligent systems, as well as long chains of inference processes.
Keywords: Cognitive informatics, denotational mathematics, concept algebra, system algebra, process algebra, RTPA, visual semantic algebra, intelligence science, AI, computational intelligence, software engineering, knowledge engineering, cognitive computing, case studies

1. Introduction

The history of sciences and engineering shows that many branches of mathematics have been created in order to meet their abstract, rigorous, and expressive needs. These phenomena may be conceived as that new problems require new forms of mathematics [3, 43]. It also indicates that the maturity of a new discipline is characterized by the maturity of its theories denoted in rigorous and efficient mathematical means [37, 38, 41, 43]. The entire computing theory, as Lewis and Papadimitriou perceived, is about mathematical models of computers and algorithms [17]. Hence, the entire theory of cognitive informatics and intelligence science is about mathematical models of the natural and machine intelligence and efficient mathematical means.

A great extent of effort has been put on extending the capacity of sets and mathematical logic in dealing with the problems in cognitive informatics and computational intelligence. The former are represented by the proposals of fuzzy sets [58] and rough sets [25, 26, 29]. The latter are represented by the development of fuzzy logic [59] and temporal logic [27]. New mathematical structures are created such as embedded relations, incremental relations, and the big-R calculus [43, 49]. More systematically, a set of new denotational mathematical forms [37, 38, 41, 43] are developed in recent years.

Definition 1. Denotational mathematics is a category of expressive mathematical structures that deals with high level mathematical entities beyond numbers and sets, such as abstract objects, complex relations, behavioral information, concepts, knowledge, processes, and systems.

The emergence of denotational mathematics is driven by the practical needs in cognitive informatics, intelligence science, software science, and knowledge science, because all of these modern disciplines study complex human and machine behaviors and their rigorous treatments [43, 50]. The utility of denotational mathematics serves as the means and rules to rigorously and explicitly express design notions and conceptual models of abstract architectures and interactive behaviors of complex systems at the highest level of abstraction, in order to deal with problems of cognitive informatics and computational intelligence, which are often characterized with large scales, complex architectures, and long chains of computing behaviors. Therefore, denotational mathematics is a system level mathematics, in which detailed individual computing behaviors may still be modeled by conventional analytical mathematics at lower levels. Typical paradigms of denotational mathematics are known as concept algebra [44], system algebra [45], Real-Time Process Algebra (RTPA) [32, 36, 38, 46, 47], and Visual Semantic Algebra (VSA) [48].

This paper presents the contemporary denotational mathematical structures in cognitive informatics, computational intelligence, and cognitive computing beyond classic mathematical entities, such as information, concepts, knowledge, processes, behaviors, systems, complex relations, and distributed objects. The paradigms of denotational mathematics, such as concept algebra, system algebra, RTPA, and VSA, and their applications in cognitive informatics and intelligence science are elaborated throughout this paper. Section 2 presents concept algebra for knowledge and software system manipulations. Section 3 introduces system algebra for abstract system modeling. Section 4 describes RTPA for intelligent and
computational behavior modeling and manipulations, followed by Section 5 on VSA for visual semantic objects and patterns modeling and manipulations.

2. Concept Algebra

Concepts are the basic unit of both knowledge and reasoning [2, 6, 8, 9, 16, 20, 23, 37, 40, 41]. The rigorous modeling and formal treatment of concepts are at the center of theories for knowledge presentation and manipulation [5, 23, 30, 54]. A concept in linguistics is a noun or noun-phrase that serves as the subject of a to-be statement [16, 33, 38]. A new mathematical structure known as concept algebra is introduced below for a formal treatment of abstract concepts.

2.1. The Mathematical Model of Formal Concepts

Definition 2. A concept is a cognitive unit to identify and/or model a real-world concrete entity and a perceived-world abstract object.

Before an abstract concept is defined, the semantic environment or context [8, 9, 16, 21] in a given language, is introduced.

Definition 3. Let $O$ denote a finite nonempty set of objects, and $A$ be a finite nonempty set of attributes, then a semantic environment or context $\Theta_C$ is denoted as a triple, i.e.:

$$\Theta_C = (O, A, R)$$

where $R$ is a set of relations between $O$ and $A$, and $|$ denotes alternative relations.

Concepts in denotational mathematics [38, 44] are an abstract structure that carries certain meaning in almost all cognitive processes such as thinking, learning, and reasoning.

Definition 4. An abstract concept $c$ on the semantic environment $\Theta_C$ is a 5-tuple, i.e.:

$$c = (O, A, R^e, R^i, R^o)$$

where

- $O$ is a nonempty set of objects of the concept, $O = \{o_1, o_2, \ldots, o_m\} \subseteq \mathcal{P}O$, where $\mathcal{P}O$ denotes a power set of $O$.
- $A$ is a nonempty set of attributes, $A = \{a_1, a_2, \ldots, a_n\} \subseteq \mathcal{P}A$.
- $R^e = O \times A$ is a set of internal relations.
- $R^i \subseteq C' \times C$ is a set of input relations, where $C'$ is a set of external concepts, i.e., $C' \subseteq \Theta_C$.
- $R^o \subseteq C \times C'$ is a set of output relations.
2.2. Concept Algebra for Knowledge Manipulations

Concept algebra is an abstract mathematical structure for the formal treatment of concepts and their algebraic relations, operations, and associative rules for composing complex concepts.

Definition 5. A concept algebra $CA$ on a given semantic environment $\Theta_C$ is a triple, i.e.:

$$CA:= (C, OP, \Theta_C) = (\{O, A, R^c, R^i, R^o\}, \{\cdot_r, \cdot_c\}, \Theta_C)$$ (3)

where $OP = \{\cdot_r, \cdot_c\}$ are the sets of relational and compositional operations on abstract concepts.

Definition 6. The relational operations $\cdot_r$ in concept algebra encompass 8 comparative operators for manipulating the algebraic relations between concepts, i.e.:

$$\cdot_r = \{\leftrightarrow, \leftrightarrow^c, \prec, \succ, =, \equiv, \sim, \ldots\}$$ (4)

where the relational operators stand for related, independent, subconcept, superconcept, equivalent, consistent, comparison, and definition, respectively.

Definition 7. The compositional operations $\cdot_c$ in concept algebra encompass 9 associative operators for manipulating the algebraic compositions among concepts, i.e.:

$$\cdot_c = \{; , ; , \vdash, \triangleright, \triangleright^c, \triangleright^i, \triangleright^o\}$$ (5)

where the compositional operators stand for inheritance, tailoring, extension, substitute, composition, decomposition, aggregation, specification, and instantiation, respectively.

Detailed descriptions of the relational and compositional operations of concept algebra may be referred to [44]. Concept algebra provides a denotational mathematical means for algebraic manipulations of abstract concepts. Concept algebra can be used to model, specify, and manipulate generic “to be” type problems, particularly system architectures, knowledge bases, and detail-level system designs, in cognitive informatics, computational intelligence, cognitive computing, software engineering, and system engineering.

2.3. Applications of Concept Algebra in Machine Learning

In cognitive informatics [33, 34, 39, 50], learning is defined as a cognitive process at the higher cognitive function layer according to the Layered Reference Model of the Brain (LRMB) [52]. The learning process interacts with multiple fundamental cognitive processes such as object identification, abstraction, search, concept establishment, comprehension, memorization, and retrievably testing. Learning is closely related to other higher cognitive processes of inferences such as deduction, induction, abduction, analogy, analysis, synthesis, and problem solving [51, 52].

Definition 8. Learning is a higher cognitive process of the brain at the higher cognitive layer of LRMB that gains knowledge of something or acquires skills in some actions by updating the cognitive models in Long-Term Memory (LTM).
According to the Object-Attribute-Relation (OAR) model [40], results of learning can be embodied by the updating of the existing OAR in the brain as a concept network. In other words, learning is a dynamic composition of the currently created sub-OAR and the existing OAR in LTM.

**Definition 9.** A composition of concept $c$ from $n$ subconcepts $c_1, c_2, \ldots, c_n$, denoted by $\psi$, is an integration of them that creates the new super concept $c$ via concept conjunction, and establishes new associations between them, i.e.:

\[
c(\bar{O}, A, R^c, R^c, R^a) \equiv \bigcup_{i=1}^{n} c_i 
\]

\[
c(\bar{O}, A, R^c, R^c, R^a) \mid O = \bigcup_{i=1}^{n} O_i, A = \bigcup_{i=1}^{n} A_i.
\]

\[
R^c = \bigcup_{i=1}^{n} (R^c_{O_i} \cup \{(c, c_i), (c_i, c)\}), R^i = \bigcup_{i=1}^{n} R^c_{O_i}, R^a = \bigcup_{i=1}^{n} R^a_{O_i}
\]

\[
\mid R^c_{O_i} = R^c_{O_i} \cup \{(c, c_i), (c_i, c)\}
\]

\[
(6)
\]

As specified in Eq. 6, the composition operation results in the generation of new internal relations $\Delta R^c = \bigcup_{i=1}^{n} \{(c, c_i), (c_i, c)\}$ that is not belongs to any of its subconcepts. It is also noteworthy that, during learning by concept composition, the existing knowledge in forms of the individual $n$ concepts is concurrently changed and updated via the newly created input/output relations with the newly generated concept.

**Example 1.** The learning process is a cognitive composition of a piece of newly acquired information and the existing knowledge in LTM in the form of the OAR-based knowledge networks. The cognitive process of learning can be formally modeled using concept algebra and Real-Time Process Algebra (RTPA) as given in Fig. 1. The center of the cognitive process of learning is that knowledge about the learn objects and intermediate results are represented internally in the brain as a sub-OAR model. According to the LRMB model [53] and the OAR model [40] of internal knowledge representation in the brain, the temporal result of learning in Short-Term Memory (STM) is a new sub-OAR model, which will be used to update the entire OAR model of knowledge in LTM as permanent learning result.

According to the formal model of the learning process, autonomous machine learning can be carried out by the following steps: 1) **Identify object:** This step identifies the learning object $O$; 2) **Concept establishment:** This step establish a concept model for the learning object $O$, $c(A, R, O)$, by searching related attributes $A$, Relations $R$, and instances $O$; 3) **Comprehension:** This step comprehends the concept and represents the concept with a sub-OAR model in STM; 4) **Memorization:** This step associates the learnt sub-OAR of the learning object with the entire OAR knowledge, and retains it in LTM; 5) **Rehearsal test:** This step checks if the learning result needs to be rehearsed. If yes, it continues to parallel execution of Steps (6) and (7); otherwise, it exit; 6) **Re-establishment of concept:** This step recalls the concept establishment process to rehearse the learning result; 7) **Re-comprehension:** This step recalls the comprehension process to rehearse the learning result.
The formalization of the cognitive process of learning by concept algebra and RTPA does not only reveal the mechanisms of human learning, but also explain how machine may gain the capability of autonomic learning. Based on the rigorous syntaxes and semantics of RTPA, the formal learning process can be implemented by computers in order to form the core of machine intelligence [42].

3. System Algebra

Systems are the most complicated entities and phenomena in abstract, physical, information, and social worlds across all science and engineering disciplines. The system concept can be traced back to the 17th Century when R. Descartes (1596-1650) noticed the interrelationships among scientific disciplines as a system, followed by the proposal of the general system notion by Ludwig von Bertalanffy in the 1920s [7, 31]. Systems may be treated rigorously as a new mathematical structure beyond conventional mathematical entities known as the abstract systems [38, 45]. Based on this view, the concept of abstract systems and system algebra are elaborated in this section.

3.1. The Mathematical Model of Formal Systems

**Definition 10.** An abstract system is a collection of coherent and interactive entities that has stable functions and a clear boundary with the external environment.

An abstract system forms the generic model of various real-world systems and represents the most common characteristics and properties of them [45]. For instance, the theory of granular computing can be explained by the abstract system theory in Subsection 3.3 [47].
**Definition 11.** Let \( E \) be a finite nonempty set of *entities*, \( F \) a finite nonempty set of *functions*, \( V_E \) a finite nonempty set of *domains* of \( E \), and \( V_F \) a finite nonempty set of *domains* of \( F \), then the *universal system* \( \mathcal{U} \), which forms the discourse of abstract systems, is denoted as a 4-tuple, i.e.:

\[
\mathcal{U} = (E, F, V_E, V_F)
\]  

(7)

Abstract systems can be classified into two categories known as the *closed* and *open* systems. Most practical and useful systems in nature are open systems in which there are interactions between the system and its environment.

**Definition 12.** An *open system* \( S \) on the universal system environment \( \mathcal{U} \), \( S \subseteq \mathcal{U} \), is a 7-tuple, i.e.:

\[
S = (C, R^c, R^i, R^o, B, \Omega, \Theta)
\]  

(8)

where

- \( \subseteq \) denotes that a set or structure (tuple) is a substructure or derivation of a super structure.
- \( C \) is a finite nonempty set of *components* of system \( S \), \( C \subseteq \mathcal{P}E \subseteq \mathcal{U} \) and \( C \subseteq S \), where \( \mathcal{P} \) denotes a power set.
- \( R^c \subseteq C \times C \) is a finite nonempty set of *internal relations* between pairs of the components \( C \subseteq S \)
- \( R^i \subseteq C_\Theta \times C \) is a finite nonempty set of input relations, where \( C_\Theta \) is a finite nonempty set of external component, \( C_\Theta \subseteq \mathcal{P}E \subseteq \mathcal{U} \) and \( C_\Theta \nsubseteq S \)
- \( R^o \subseteq C \times C_\Theta \) is a finite nonempty set of *output relations*.
- \( B \) is a finite nonempty set of *behaviors* of \( C \subseteq S \), \( B \subseteq \mathcal{P}F \subseteq \mathcal{U} \) and \( B \subseteq S \).
- \( \Omega \) is a finite nonempty set of *constraints* of \( C \) and \( B \), \( \Omega \subseteq \mathcal{P}V_E \cup \mathcal{P}V_F \subseteq \mathcal{U} \) and \( \Omega \subseteq S \).
- \( \Theta \) is the *environment* of \( S \) with a finite nonempty set of external components outside \( S \), i.e., \( \Theta = C_\Theta \subseteq \mathcal{P}E \subseteq \mathcal{U} \) and \( \Theta = C_\Theta \nsubseteq S \).

A closed system \( \tilde{S} \) on \( \mathcal{U} \) is a special case of an open system \( S \), which can be derived based on Definition 12 as a 4-tuple, i.e., \( \tilde{S} = (C, R, B, \Omega) \).

### 3.2. System Algebra for System Manipulations

System algebra is an abstract mathematical structure for the formal treatment of abstract and general systems as well as their algebraic relations, operations, and associative rules for composing and manipulating complex systems [45].

**Definition 13.** A *system algebra* \( SA \) on a given universal system environment \( \mathcal{U} \) is a triple, i.e.:

\[
SA = (S, OP, \mathcal{U})
\]

\[
= (\{C, R^c, R^i, R^o, B, \Omega\}, \{\bullet_r, \bullet_c\}, \mathcal{U})
\]  

(9)

where \( OP = \{\bullet_r, \bullet_c\} \) are the sets of *relational* and *compositional* operations, respectively, on abstract systems.
Definition 14. The relational operations $\bullet_r$ in system algebra encompass 6 comparative operators for manipulating the algebraic relations between abstract systems, i.e.:

$$\bullet_r \hat{=} \{ \nleftrightarrow, \leftrightarrow, \Pi, =, \subseteq, \supseteq \}$$

where the relational operators stand for independent, related, overlapped, equivalent, subsystem, and supersystem, respectively.

Definition 15. The compositional operations $\bullet_c$ in system algebra encompass 9 associative operators for manipulating the algebraic compositions among abstract systems, i.e.:

$$\bullet_c \hat{=} \{ :iden, +, \sim, \ominus, \oplus, \lhd, \triangleright, \vdash \}$$

where the compositional operators stand for system inheritance, tailoring, extension, substitute, difference, composition, decomposition, aggregation, and specification, respectively.

Detailed description of the relational and compositional operations on abstract systems may be referred to [45].

3.3. Applications of System Algebra in Granular Computing

System algebra provides a denotational mathematical means for algebraic manipulations of all forms of abstract systems. System algebra can be used to model, specify, and manipulate generic “to be” and “to have” type problems, particularly system architectures and high-level system designs, in cognitive informatics, computational intelligence, cognitive computing, software engineering, and system engineering.

This subsection presents a case study of applications of system algebra for formally modeling and manipulating granular computing problems. Granular computing is a new computational methodology that models and implements computational structures and functions by a granular system, where each granule in the system carries out a predefined function or behavior by interacting to other granules in the system [18, 25, 28, 57, 60].

It is recognized that any abstract or concrete granule can be formally modeled by an abstract system in system algebra. On the basis of Definition 12, an abstract granule can be formally described as follows.

Definition 16. A computing granule $G$ on the universal system environment $\Omega$ is a 7-tuple, i.e.:

$$G \equiv S = (C, R^c, R^i, R^o, B, \Omega, \Theta)$$

where $C$ is a nonempty set of cell or component of the system, $C = \{c_1, c_2, \cdots, c_n\} \subseteq \mathbb{C} \subseteq \Omega$. The other object sets are the same as those of abstract system $S$.

Definition 17. A granular system $S_G$ is a composition of multiple granules in a system where all granules interact with each other for a common goal of system functionality.

Properties of granular systems obey the properties of generic abstract systems [47]. The set of relational and compositional operations on granules and granular systems towards granular computing are identical as that modeled in system algebra.
Example 2. A digital clock granule, *Clock*, is a granular system *G*₁ as follows:

\[ \text{Clock} \hat{=} G_1(C_1, R'_1, R''_1, R'''_1, B_1, \Omega_1, \Theta_1) \]  \hspace{1cm} (13)

where the configuration of the Clock granule is given in Fig. 2.

In Fig. 2, the behaviors of the clock granule defined in *B*₁ can be further refined by a set of processes in RTPA.

Example 3. The alarm subsystem for the digital clock, *Alarm*, can be treated as another granule *G*₂ as follows:

\[ \text{Alarm} \hat{=} G_2(C_2, R'_2, R''_2, R'''_2, B_2, \Omega_2, \Theta_2) \]  \hspace{1cm} (14)

where the configuration of the Alarm granule is given in Fig. 3.

The composition of two concrete granules in real-world system design for granular computing are described in Example 4, which illustrates how the generic abstract granule compositional operation may be implemented in real-world granule-based system design and modeling.

Example 4. According to system algebra, the composition of the two granules *G*₁(*Clock*) and *G*₂(*Alarm*) as given in Examples 2 and 3 results in a new supergranule *G*(*Alarm_Clock*) as follows:
The Alarm Granule $G(C, R, R^1, R^2, B, \Omega, \Theta)$

- The set of cells: $C_1 = \{\text{Processor}, \text{Keypad}, \text{LEDs}, \text{Bell}\}$
- The set of internal relations:
  - $R^1 \subseteq C_1 \times C_2$, $(\text{Keypad, Processor})$
  - $R^2 \subseteq C_2 \times C_3$, $(\text{AlarmClock, Ring, Alarm})$
  - $\text{Output (Processor, LEDs), Ring (Processor, Bell)}$
- The set of input relations:
  - $R_1 \subseteq C_1 \times C_3$, $(\text{SetAlarm (User, Keypad)})$
- The set of output relations:
  - $R_1 \subseteq C_1 \times C_3$, $(\text{ShowAlarm (LEDs, User)})$
- The set of behaviors:
  - $B = \{\text{SetAlarm, ShowAlarm, CheckAlarm, Ring, AlarmRelease}\}$
- The set of constraints:
  - $\Omega_1 = \{\text{alarm = hh \times mm}\}$
- The environment:
  - $\Theta_1 = \{\text{User}\}$

Figure 3. The Alarm granule

\[
G_1(C_1, R^1, R^2, B_1, \Omega_1, \Theta_1) \cup G_2(C_2, R^1, R^2, B_2, \Omega_2, \Theta_2) \triangleq G(C, R^1, R^2, B, \Omega, \Theta) \mid C = C_1 \cup C_2, R^1 = R^1_1 \cup R^1_2, R^2 = R^2_1 \cup R^2_2, B = B_1 \cup B_2, \Omega = \Omega_1 \cup \Omega_2, \Theta = \Theta_1 \cup \Theta_2
\]

\[
\begin{align*}
&\big| \prod_{i=1}^{2} G_i(C_1, R^1, R^2, B_1, \Omega_1, \Theta_1) \mid R^i_1 = R^i_1 \cup (C \times C_1), \\
&R^i_2 = R^i_2 \cup (C \times C), \Theta_1 = \Theta_1 \cup C
\end{align*}
\]

where the configuration of the Alarm_Clock granule is given in Fig. 4.

Note that the newly generated relations `Select(Clock, Alarm)`, as well as the new behaviors `SelectClock` and `SelectAlarm` in the granule system $G(Alarm\_Clock)$ are the results of the incremental unions of systems as defined in Eq. 15. Therefore, they do not belong to either subgranule $G_1(Clock)$ or $G_2(Alarm)$ rather than purely the properties of the supergranule $G(Alarm\_Clock)$.

Examples 2 through 4 demonstrate that the generality and expressive power of system algebra and the generic abstract systems. Using system algebra, complex granule modeling and complicated granule manipulations in granular computing can be formally and rigorously conducted. The rigorous treatment of the granular computing paradigm by system algebra has established a solid foundation for granular computing and granule-based systems modeling.
4. Real-Time Process Algebra (RTPA)

A key metaphor in system modeling, specification, and description is that a software and intelligent system can be perceived and described as the composition of a set of interacting processes. Hoare in 1978 [11, 12], Milner in 1980 [22], and others developed various algebraic approaches to represent communicating and concurrent systems, known as process algebra. A process algebra is a set of formal
notations and rules for describing algebraic relations of software engineering processes. RTPA [32, 33, 36, 38, 47] is a real-time process algebra that can be used to formally and precisely describe and specify architectures and behaviors of human and software systems.

4.1. The Mathematical Model of Abstract Behavioral Processes

Definition 18. A process $P$ is a composed listing and a logical combination of $n$ meta-statements $p_i$ and $p_j$, $1 \leq i < n$, $1 < j \leq m = n + 1$, according to certain composing relations $r_{ij}$, i.e.:

$$P = \prod_{i=1}^{n-1} r_{ij} (p_i, r_j p_j), f = i + 1$$

$$= (\cdots (((p_1) r_2 p_2) r_3 p_3) \cdots r_{n-1,n} p_n)$$

(16)

where the big-R notation [49] is adopted that describes the nature of processes as the building blocks of programs.

Definition 18 indicates that the mathematical model of a process is a cumulative relational structure among computing operations. The simplest process is a single computational statement.

4.2. Real-Time Process Algebra (RTPA) for Process Manipulations

Definition 19. RTPA is a denotational mathematical structure for algebraically denoting and manipulating system behavioural processes and their attributes by a triple, i.e.:

$$RTPA \equiv (\Sigma, \Psi, \Re)$$

(17)

where $\Sigma$ is a set of 17 primitive types for modeling system architectures and data objects, $\Psi$ a set of 17 meta-processes for modeling fundamental system behaviors, and $\Re$ a set of 17 relational process operations for constructing complex system behaviors.

Definition 20. The primary types of computational objects of RTPA, $\Sigma$, encompass 17 primitive types elicited from fundamental computing needs, i.e.:

$$\Sigma \equiv \{N, Z, R, S, BL, B, H, P, TI, D, DT, RT, ST, \odot, \otimes, TM, \ominus, \ominus, SBL\}$$

(18)

Details of the RTPA type system may be referred to [46].

Definition 21. A meta-process in RTPA is a primitive computational operation that cannot be broken down to further individual actions or behaviors.

A meta-process serves as a basic building block for modeling software behaviors. Complex processes can be composed from meta-processes using process relational operations. In RTPA, a set of 17 meta-processes has been elicited from essential and primary computational operations commonly identified in existing formal methods and modern programming languages [1, 10, 19, 55, 56].
Definition 22. The RTPA meta-process system encompasses 17 fundamental computational operations elicited from the most basic computing needs, i.e.:

\[ \mathcal{P} \doteq \{ :=, \ddot{\cdot}, \lll, \llll, \ggg, \gggg, \lll, \llll, \ggg, \gggg, \lll, \llll, \ggg, \gggg, \lll, \llll, \ggg, \gggg \} \]

where the meta-processes of RTPA stand for assignment, evaluation, addressing, memory allocation, memory release, read, write, input, output, timing, duration, increase, decrease, exception detection, skip, stop, and system, respectively.

Definition 23. A process relation in RTPA is an algebraic operation and a compositional rule between two or more meta-processes in order to construct a complex process.

A set of 17 fundamental process relations has been elicited from fundamental algebraic and relational operations in computing in order to build and compose complex processes in the context of real-time software systems.

Definition 24. The software composing operations of RTPA, \( \mathcal{R} \), encompass 17 fundamental algebraic and relational operations elicited from basic computing needs, i.e.:

\[ \mathcal{R} \doteq \{ \rightarrow, \leftarrow, |, \cdots |, \cdots, R', R^+, \mathcal{O}, \leftarrow, \lll, \llll \} \]

where the relational operators of RTPA stand for sequence, jump, branch, while-loop, repeat-loop, for-loop, recursion, function call, parallel, concurrence, interleave, pipeline, interrupt, time-driven dispatch, event-driven dispatch, and interrupt-driven dispatch, respectively.

RTPA provides a coherent notation system and a formal engineering methodology for modeling both software and intelligent systems. RTPA can be used to describe both logical and physical models of systems, where logic views of the architecture of a software system and its operational platform can be described using the same set of notations. When the system architecture is formally modelled, the static and dynamic behaviours, which perform on the system architectural model, can be specified by a three-level refinement scheme at the system, class, and object levels in a top-down approach. Detailed syntaxes and formal semantics of RTPA meta-processes and process relations may be referred to [32, 38, 46, 47].

4.3. Applications of RTPA in Computational Intelligence and Software Engineering

The most generic and fundamental operations in system and human behavioral modeling are iterations and recursions. Because a variety of iterative constructs are provided in different programming languages, the notation for repetitive, cyclic, recursive behaviors and architectures in computing need to be unified. The big-R notation of RTPA can be used to denote not only repetitive operational behaviors in computing, but also recurring constructs of architectures and data objects as shown below.

Example 5. The architecture of a two-dimensional array with \( n \times m \) integer elements, \( A_{nm} \), can be denoted by the big-R notation as follows:
Because the big-R notation provides a powerful and expressive means for denoting iterative and recursive behaviors, as well as architectures of systems or human beings, it is a universal mathematical means for system modeling in terms of repetitive behaviors and recurring architectures, respectively.

Example 6. The while-, repeat-, and for-iteration of process P can be denoted by the big-R notation in RTPA as:

\begin{align*}
A_{\text{iter}} &= \bigcup_{i=0}^{n-1} \bigcup_{j=0}^{m-1} A[i, j] N \\
\end{align*}

Example 7. The cognitive process of learning [42] can be modeled in RTPA as shown in Fig. 1.

A number of real-world software systems have been formally modeled in RTPA [38], such as the telephone switching system, the lift dispatching system, the real-time operating system, and the ATM system. RTPA are not only useful for rigorously modeling and manipulating software systems, but also widely applicable to human cognitive process modeling and computational intelligence [35, 38, 51].

5. Visual Semantic Algebra

It is recognized that human visual knowledge about the real-world is mainly represented by abstract or semantic objects [14, 15, 48]. In order to efficiently model the abstract visual objects, their semantic representations, and their rigorous compositions and manipulations, a new denotational mathematics known as visual semantic algebra is introduced in this section for visual object and pattern recognition and processing.

5.1. The Mathematical Model of Abstract Visual Objects and Patterns

The basic geometric shapes (2-D) and solids (3-D), known collectively geons, have been studies in cognitive psychology [4], computational intelligence, and robotics [13, 24, 48].
Definition 25. A set of 26 typical visual semantic objects are modeled in four categories known as those of plane geometry $H$, solid geometry $S$, generic figures $F$, and abstract spatial limits $L$ as defined below:

$$
H \triangleq \{ \bullet, -, \angle, \bigcirc, \bigtriangleup, \bigtriangledown, \diamond, \circ, \sqsubset \}
$$

$$
S \triangleq \{ C_y, R_y, C_y', S_p, C_p, P_y \}
$$

$$
F \triangleq \{ \uparrow, \Diamond, \bigcirc \}
$$

$$
L \triangleq \{ \uparrow, \downarrow, \leftarrow, \rightarrow \}
$$

(22)

Each abstract visual object in $\{H, S, F, L\}$ can be rigorously modeled by a specific mathematical model [48 ].

5.2. Visual Semantic Algebra (VSA) for Visual Object Manipulations

A set of 11 algebraic operations, as described in Table 1, is elicited from relational compositions of the 26 abstract visual objects. In Table 1, any 2-D or 3-D geometric structure can be analyzed or composed semantically using VSA.

Definition 26. The set of 11 algebraic operations as modeled in Table 1 can be applied to relational compositions of the 26 abstract visual objects given in Definition 25, i.e.:

Table 1. Algebraic Operations on Abstract Visual Objects in VSA

<table>
<thead>
<tr>
<th>Relational operations</th>
<th>Symbol</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Above</td>
<td>↑</td>
<td>$S_i \uparrow S_j$</td>
<td>$S_i$ is above $S_j$.</td>
</tr>
<tr>
<td>Below</td>
<td>↓</td>
<td>$S_i \downarrow S_j$</td>
<td>$S_i$ is below $S_j$.</td>
</tr>
<tr>
<td>Left</td>
<td>←</td>
<td>$S_i \leftarrow S_j$</td>
<td>$S_i$ is on the left of $S_j$.</td>
</tr>
<tr>
<td>Right</td>
<td>→</td>
<td>$S_i \rightarrow S_j$</td>
<td>$S_i$ is on the right of $S_j$.</td>
</tr>
<tr>
<td>Front</td>
<td>⊙</td>
<td>$S_i \odot S_j$</td>
<td>$S_i$ is in front of $S_j$.</td>
</tr>
<tr>
<td>Behind</td>
<td>⊗</td>
<td>$S_i \otimes S_j$</td>
<td>$S_i$ is behind $S_j$.</td>
</tr>
<tr>
<td>Inside</td>
<td>⊕</td>
<td>$S_i \oplus S_j$</td>
<td>$S_i$ is inside $S_j$.</td>
</tr>
<tr>
<td>Angle</td>
<td>$\angle { x^\circ }$</td>
<td>$S_i \angle { x^\circ } S_j$</td>
<td>$S_i$ is at an $x^\circ$ angle position related to $S_j$, $\angle O^\circ$ is defined at the right position.</td>
</tr>
<tr>
<td>Relative position</td>
<td>$\hat{(p)}$</td>
<td>$S \hat{(p)}$</td>
<td>$S$ is allocated at the position $p$.</td>
</tr>
<tr>
<td>Absolute position</td>
<td>$\hat{(x, y, z)}$</td>
<td>$S \hat{(x,y,z)}$</td>
<td>$S$ is allocated at the position $(x, y, z)$.</td>
</tr>
<tr>
<td>Move</td>
<td>$\bigcirc \hat{(p)}$</td>
<td>$S \hat{(p)} \bigcirc \hat{(p_2)}$</td>
<td>$S$ is moved from position $p_1$ to $p_2$.</td>
</tr>
</tbody>
</table>
\[ \cdot_{\text{VSA}} = \{ \uparrow, \downarrow, \leftarrow, \circ, \otimes, \angle, \circlearrowleft, (\mathbb{p}), @((x, y, z), \curvearrowright) \} \]  

Using the algebraic operations of semantic objects, any 2-D or 3-D geometric structure can be analyzed or composed semantically. Details of the set of abstract visual objects and their algebraic operations in VSA may be referred to [48].

**Definition 27.** The *Visual Semantic Algebra* (VSA) is a denotational algebraic system that formally manipulates visual objects by algebraic operations on symbolic or semantic objects in geometric analyses and compositions, i.e.:

\[
\text{VSA} \cong (O, \cdot_{\text{VSA}}) = ( \{ H, S, F, L \}, \cdot_{\text{VSA}} )
\]

5.3. Applications of VSA

VSA provides a neat and powerful algebraic system for rigorously manipulating visual objects and patterns. Any 2-D or 3-D visual semantic structure or system can be analyzed or composed using VSA. A set of case studies on applications of VSA is presented below.

**Example 8.** The visual semantic model of the solid structures of \( A \) and \( B \) as given in Fig. 5 can be expressed in VSA as follows:

\[
A \cong C_o \uparrow C_y \uparrow @(\text{center}) R_s \uparrow \perp \\
B \cong P_y \uparrow @(\text{center}) R_s \uparrow (C_{y1} \leftarrow C_{y2}) \uparrow \perp
\]

where \( O \uparrow \perp \) denotes that the object \( O \) is on the top of the ground or a reference plane.

**Example 9.** The tower of Hanoi problem as shown in Fig. 6 can be described using VSA from the point of view of visual object processing by humans or robots. The algorithm of the tower of Hanoi described in VSA is given in Fig. 7, where it encompasses the architecture (such as its layout, initial and final states) and computational behaviors.
The theory and case studies demonstrate that VSA provides a new paradigm of denotational mathematics for relational visual object manipulation. VSA can be applied not only in machine visual and spatial reasoning, but also in intelligent system designs as a man-machine language in representing and
dealing with the high-level inferences in complex visual patterns and systems. On the basis of VSA, computational intelligence systems such as robots and cognitive computers can process and inference visual and image object rigorously and efficiently at the conceptual level.

6. Conclusions

Denotational mathematics has been introduced as a category of expressive mathematical structures that deals with high level mathematical entities such as abstract objects, complex relations, behavioral information, concepts, knowledge, processes, and systems. Extensions of conventional algebra onto more complicated mathematical entities beyond numbers lead to the contemporary denotational mathematics. Four paradigms of denotational mathematics, such as concept algebra, system algebra, Real-Time Process Algebra (RTPA), and Visual Semantic Algebra (VSA), have bee introduced. Among the new forms of denotational mathematics, concept algebra has been designed to deal with the mathematical structure of abstract concepts and their representation and manipulations in knowledge engineering. System algebra has been created for the rigorous treatment of abstract systems and their algebraic relations and operations. RTPA has been developed as an expressive, easy-to-comprehend, and language-independent mathematical structure, as well as a modeling method for software system behaviors description and specification. VSA has been created for modeling and manipulating visual semantic objects and patterns. Applications of denotational mathematics in cognitive informatics, computational intelligence, and cognitive computing have been elaborated with a set of case studies and examples. This work has demonstrated that denotational mathematics is an ideal mathematical means for a set of emerging modern disciplines that deal with concepts, knowledge, behavioral processes, and human/machine intelligence.

Acknowledgement

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 the valuable comments and suggestions of the anonymous reviewers.

References


