Denotational and Algebraic Semantics for the CaIT calculus

Ningning Chen and Huibiao Zhu

Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
hbzhu@sei.ecnu.edu.cn

Abstract

The practical applications of the Internet of Things (IoT) have sprung up worldwide, making IoT systems more complex. Therefore, it is desirable to model and reason about IoT systems theoretically, especially from the perspective of formal methods, to ensure their quality, reliability, and security. Thus, the Calculus of the Internet of Things (CaIT) has been proposed to model the interactions among components and verify the network deployment to ensure the quality and reliability of IoT systems. However, the CaIT calculus can only support point-to-point communication, while broadcast communication is more common in IoT systems. Hence, this paper updates the CaIT calculus by replacing its communication primitive with the broadcast. Based on the Unifying Theories of Programming (UTP), we further explore its denotational semantics and algebraic semantics, with a special focus on broadcast communication, actions with the timeout (e.g. input actions and migration actions), and channel restriction. To facilitate the algebraic exploration of parallel expansion laws, we further extend the CaIT calculus with a new concept called guarded choice, which allows us to transform each program into a guarded choice form.

1 Introduction

With the increasing demand for applications and technologies of IoT, a variety of promising technologies (such as 5G, high speed, low latency networks) have been applied to the IoT paradigm, meeting the demands of a series of advanced technologies, such as machine learning, edge computing, and Industry 4.0. However, most of the existing studies on IoT focus on its practical applications, and little work has been done to model interactions between components in IoT systems and check IoT network deployments before practical implementation.

Lanese et al. presented the first calculus for IoT, named IoT-calculus, designed to capture some fundamental characteristics of IoT systems. Subsequently, Bodei et al. proposed a secure untimed process calculus, called IoT-LYSA, which employs static analysis technologies to track the sources and paths of IoT data and detect how they influence smart objects. Neither IoT-calculus nor IoT-LYSA considers the effect of time on process actions. Thus, Lanotte et al. presented the CaIT calculus to model discrete timed behaviors with consistency and fairness properties [1]. However, the CaIT calculus only supports point-to-point communication, not broadcast communication. According to b$\pi-$calculus, broadcast primitives have the following advantages: processes may interact without explicit knowledge of each other and receivers can be dynamically added or removed without modifying the sender.

As described in Hoare and He's Unifying Theories of Programming (UTP) [2], three methods can be used to represent the semantics of a programing: operational semantics, denotational semantics, and algebraic semantics. The operational semantics provides a set of transition rules to simulate how a program works. The denotational semantics explains what a program does from a purely mathematical point of view. The algebraic semantics includes a series of algebraic laws, which is well suited to the symbolic calculation of parameters and structures of an optimal design. The operational semantics of the CaIT calculus has been explored in [1].

This paper has been accepted by ICTAC 2022 [3], and its main contributions are as follows:

• We enrich the CaIT calculus by replacing point-to-point communication with broadcast communication.

• We explore the denotational semantics of the CaIT calculus, involving the basic commands, the guarded choice, the parallel composition, and channel restriction.

• We investigate the algebraic semantics of the CaIT calculus, especially the algebraic laws of channel restriction. By establishing the algebraic laws for the parallel composition of guarded choice components, we can describe any program as a guarded choice form.

2 The CaIT Calculus

2.1 Syntax

In this subsection, we only introduce some vital process actions (also called basic commands). Please refer to [3] for a more complete description of the CaIT calculus.

Network Level:

(1) $(vc')M$ indicates that channel c' is private to network M.

(2) ${}_n[\Gamma \bowtie P]^u_l$ denotes a network node, where n is the node ID, P is the process modelling the logic of this node, l records its current location, and Γ is its physical interface. u is given to differentiate between stationary nodes (if $u = s$) and mobile nodes (if $u = m$).

Process Level:

(1) $\lfloor \pi; P \rfloor Q$ stands for some actions which execute with the timeout, where $\pi \in \{?(x)^c,\ move\_k\}$. For $\lfloor ?(x)^c; P \rfloor Q$, if a value can be received via channel c within one time unit, it continues as process P after that. Otherwise, Q runs after one time unit. $\lfloor move\_k; P \rfloor Q$ illustrates the node mobility, which is similar to the input statement.

2.2 Guarded Choice

To support our investigation of the algebraic parallel expansion laws, we extend the CaIT calculus with the following three types of guarded choices. By using our algebraic parallel expansion laws, we aim to transform every program into the form of guarded choices.

- **Instantaneous Guarded Choice:** $[\!]_{i\in I}\{g_i \to N_i\}$,
  where, $g_i \in \{!\langle v\rangle^c@l,\ ?(x)^c@l,\ c.[v/x]@(l, l_1),\ s?y@l,\ a!v@l,\ move\_k@l,\ b_i\&\tau@l\}$.
- **Delay Guarded Choice:** $\#t \to N$
- **Hybrid Guarded Choice:** $[\!]_{i\in I}\{g_i \to N_i\}$
  $\oplus \exists t' \in (0 \ldots 1) \bullet \#t' \to N'$
  $\oplus \#1 \to N''$

3 Denotational Semantics and Algebraic Semantics

3.1 Denotational Semantics

We present the denotational semantics of the CaIT calculus by taking the input statement, the parallel composition, and channel restriction as examples. We use $\mathbf{beh}(N)$ to stand for the denotational semantics of the network N.

• **(Input)** The denotational semantics of the input statement is classified into three branches. The first branch indicates that the input action happens at the triggering time. In the second branch, the input command occurs after t' time units. The last branch shows that this input command does not happen within one time unit.

$$\mathbf{beh}({}_n[\Gamma \bowtie \lfloor ?(x)^c; P \rfloor Q]^u_l) =_{df} \left(\begin{array}{ll} \exists m \in Type(c) \bullet \mathbf{beh}(?(m)^c@l); \mathbf{beh}({}_n[\Gamma \bowtie P[m/x]]^u_l) \vee & \textbf{(1.1)} \\ \exists t' \in (0 \ldots 1) \bullet \mathbf{beh}(\#t'); \exists m \in Type(c) \bullet \mathbf{beh}(?(m)^c@l); & \\ \qquad \mathbf{beh}({}_n[\Gamma \bowtie P[m/x]]^u_l) \vee & \textbf{(1.2)} \\ \mathbf{beh}(\#1); \mathbf{beh}({}_n[\Gamma \bowtie Q]^u_l) & \textbf{(1.3)} \end{array} \right)$$

• **(Parallel Composition)** We discuss the denotational semantics of the parallel compositions. Predicate $Merge$ merges states, termination time, and the traces contributed by networks N_1 and N_2 [3].

$$\mathbf{beh}(N_1 \| N_2) =_{df} \left(\begin{array}{ll} \exists st_1, st_1', st_2, st_2', time_1, time_1', time_2, time_2', tr_1, tr_1', tr_2, tr_2' \bullet & \\ \quad st_1 = st_2 = st \wedge time_1 = time_2 = time \wedge tr_1 = tr_2 = tr \wedge & \textbf{(1)} \\ \mathbf{beh}(N_1)[st_1, st_1', time_1, time_1', tr_1, tr_1'/st, st', time, time', tr, tr'] \wedge & \textbf{(2)} \\ \mathbf{beh}(N_2)[st_2, st_2', time_2, time_2', tr_2, tr_2'/st, st', time, time', tr, tr'] \wedge & \textbf{(3)} \\ Merge & \textbf{(4)} \end{array} \right)$$

• **(Channel Restriction)** Based on the denotational semantics of network M, we can further gain the denotational semantics of $(vc')M$ (Page 2). We need to remove the snapshots which record output actions and synchronous communication actions occurring on c' from the trace of M.

$$\mathbf{beh}((vc')M) =_{df} \left(\begin{array}{l} \mathbf{beh}(M)[\mathbf{Re}(tr' - tr, c')/tr' - tr, div/str'] \\ \lhd \mathbf{Diverge}(\mathbf{beh}(M), c') \rhd \mathbf{beh}(M)[\mathbf{Re}(tr' - tr, c')/tr' - tr] \end{array} \right),$$

The condition $\mathbf{Diverge}(\mathbf{beh}(M), c')$ is used to check whether the concealment of channel c' leads to the behaviour of M (i.e., $\mathbf{beh}(M)$) diverge. If it is true, the final state is a divergence state (i.e., div). The function $\mathbf{Re}(s, c')$ is defined to remove snapshots involving c' from trace s.

3.2 Algebraic Laws

In this subsection, we explore the algebraic laws of the CaIT calculus, involving basic commands (using the input statement as an example), parallel compositions, and channel restriction.

• **(Input)** $$\begin{aligned} {}_n[\Gamma \bowtie \lfloor ?(x)^c; P \rfloor Q]_l^u =& ?(x)^c @ l \rightarrow_n [\Gamma \bowtie P]_l^u \\ & \oplus \exists t' \in (0 \ldots 1) \bullet \#t' \rightarrow ?(x)^c @ l \rightarrow_n [\Gamma \bowtie P]_l^u \\ & \oplus \#1 \rightarrow_n [\Gamma \bowtie Q]_l^u \end{aligned}$$

• **(Parallel Composition)** Now we explore the algebraic laws for parallel compositions, especially the parallel compositions of guarded choices. To support the parallel expansion laws, two components of a parallel process can both be one of the three types of guarded choices, as shown in Table 1.

Table 1: Parallel composition of two guarded choices

	Instantaneous	**Delay**	**Hybrid**
Instantaneous	**(par-4-1), (par-4-2)**	**(par-5)**	**(par-6)**
Delay		**(par-7)**	**(par-8)**
Hybrid			**(par-9)**

• **(par-5)** $[\![_{i \in I}\{g_i \rightarrow N_i\} \| \#t \rightarrow M = [\![_{i \in I}\{g_i \rightarrow (N_i \| \#t \rightarrow M)\}$

Here, we take **(par-5)** as an example to illustrate the details of Table 1. **(par-5)** describes the parallel composition of the instantaneous guarded choice and the delay guarded choice.

• **(Channel Restriction)** Based on the above algebraic laws, we can transform any program (e.g., M) without restricted channels into a guarded choice form. To further deal with the channel restriction (i.e., $(vc')M$), we give a law to replace output actions and synchronous communication actions occurring on channel c' with the silent action (i.e., $true \& \tau @ l$). For other actions occurring in M, such as delaying actions, communication actions that do not involve c', etc., $(vc')M$ is consistent with M.

References

[1] Ruggero Lanotte, Massimo Merro: A semantic theory of the Internet of Things. Inf. Comput. 259(1): 72-101 (2018).

[2] C. A. R. Hoare and Jifeng He. Unifying Theories of Programming. Prentics Hall International Series in Computer Science, 1998.

[3] https://github.com/Cnn-c/Denotational-and-Algebraic-Semantics-for-the-CaIT-calculus.git