Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8015896, 17 pages
https://doi.org/10.1155/2022/8015896

Research Article

Proving Simulink Block Diagrams Correct via Refinement

Wei Zhang,[1] **Quan Sun**,[2] **Chao Wang**,[1] **and Zhiming Liu**[1,3]

[1]*College of Computer and Information Science, Southwest University, Chongqing 400715, China*
[2]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*
[3]*School of Software, Northwestern Polytechnical University, Xi'an 710129, China*

Correspondence should be addressed to Quan Sun; quansun@nuaa.edu.cn and Zhiming Liu; zliu@nwpu.edu.cn

Received 22 June 2022; Accepted 30 July 2022; Published 19 September 2022

Academic Editor: Chenglu Jin

Simulink is a well-known block diagram-based tool for modular design and multidomain simulation of Cyber-Physical Systems (CPS). However, the simulation by Simulink cannot completely cover the state space or behavior of a target system, which would not ensure the correctness of the developed block diagrams in Simulink. In this work, we present a contract-based method, which supports compositional reasoning and refinement, for proving the correctness of Simulink block diagrams with *discrete-time* and *continuous-time* dynamic behavior. We use the assume-guarantee contract as a specification language. The Simulink block diagrams are correct in the sense that if the block diagrams satisfy the formal specifications of the system being modeled. To prove the correctness of a block diagram, we first define semantics for Simulink block diagrams. We study three composition operators, i.e., serial, parallel, and algebraic loop-free feedback with multistep delays. We present a satisfaction relation between the block diagram and contract and present a refinement relation between the contracts. We prove that if the Simulink block diagram satisfies the composition contract and the composition contract refines the system specifications, the block diagram is correct relative to the system specifications. Furthermore, we demonstrate the effectiveness of our method via a real-world case study originating from the control system of a reservoir. Our method can also provide an idea to verify whether the designed CPS is planted with a logic bomb by attackers.

1. Introduction

Cyber-Physical Systems (CPS) are engineering systems where functionalities emerge from the network interaction of physical and computational processes [1]. Designing CPS correctly and efficiently is a critical challenge for computer science and industry. Model-based design (MBD) [2] provides virtual system integration and a visual approach to develop models for CPS. Bugs in the model can be identified and corrected at an early stage of the design process when no hardware is available. Such a method is considered as an effective solution to design CPS correctly.

Simulink [3] is a graphical modeling language for model-based design (MBD). Currently, Simulink greatly appeals to CPS engineers since it captures the dynamic behavior of the modeled system. A Simulink block diagram consists of blocks connected via wires. The blocks (from Simulink library, a set of predefined blocks that can assemble block diagrams of systems with drag-and-drop mouse operations) represent different parts of a system being modeled, and wires indicate the communication between the blocks. The blocks have input and output ports that receive the input signals and send the output signals. The signals are the functions of time that can be *continuous-time* or *discrete-time*. Hence, the Simulink block diagrams can be classified based on the time: contain only *discrete-time* blocks, *continuous-time* blocks, or a mixture of *discrete-time* and *continuous-time* blocks. Our work focuses on the Simulink block diagrams containing only *discrete-time* blocks and *continuous-time* blocks, which we call *discrete-time* Simulink block diagrams or *continuous-time* Simulink block diagrams.

Simulink supports the design, modeling, simulation, and test of CPS. The test for Simulink block diagrams is based on numerical simulation. One of the drawbacks of numerical simulation is that it does not completely cover a target system's state space or behavior. In addition, a logic bomb [4] maliciously inserted into Simulink by attackers can persistently change the behavior. In safety-critical systems, an error could lead to incorrect analysis results and thus result in property damage, even significant injury or death. Formal methods can rigorously prove that all possible behaviors satisfy a specific formal specification, thus ensuring correctness. By "correctness," we mean that all possible behaviors of the Simulink block diagram satisfy the given formal specification of a system to be modeled.

A number of methods have been reported in the literature. To the best of our knowledge, some existing solutions only focus on Simulink block diagrams with *discrete-time* behaviors, e.g. [5–11]. A common approach for tackling *continuous-time* Simulink block diagrams is to discretize the *continuous-time* dynamical behavior [12, 13]. However, the discretization of continuous systems reduces the accuracy of the verification of continuous dynamics. The contract supports compositional reasoning and refinement, enabling hierarchical design and verification of complex systems by decomposed system-level specification into the block-level specification to provide implementations correctly. Based on this advantage, we employ contract as formal specification to specify the observable trajectory of *discrete-time* and *continuous-time* blocks and present a contract-based refinement technique for proving Simulink block diagrams' correctness.

To prove the correctness of Simulink block diagrams, we first define formal semantics for Simulink block diagrams. We consider the blocks in the library to be units and call them elementary blocks. The elementary block expresses the time-dependent (*continuous-time* or *discrete-time*) relationships between the inputs, internal states, and outputs. Thus, we define the *elementary block* as a dynamic system that can model both *continuous-time* and *discrete-time* blocks. Based on this definition, we define the observable trajectory for the blocks, i.e., the evolution of the value of input-output variables over time. We formulate the wire as *unilateral connection* (i.e., the relations of output and input between connected blocks) for communication. Then, to construct the Simulink block diagrams, we define three basic composition operators, namely, serial, parallel, and algebraic loop-free feedback composition. Moreover, we study the algebraic loop-free feedback composition containing multistep delays. Similar works [13, 14] only considered the ones with unit delay.

We then present a contract-based refinement technique to prove the correctness of the Simulink block diagrams, as shown in Figure 1. Our purpose is to prove that the block diagram satisfies the system specification. To this end, we introduce a mid-level called composition contract (i.e., the composition of contracts corresponding to the blocks that construct the Simulink block diagrams) between low-level block diagrams and high-level system specifications for composition. The approach is divided into two stages. Firstly, we define the satisfaction relation that relates block to contract and verify that the satisfaction relation is preserved by composition, i.e., if blocks satisfy their contract, respectively, their composition satisfies the contract composition. Secondly, we define the refinement relation between contracts and prove that the composition of contracts refines the system specifications. The block diagram is correct as long as we prove that the block diagram satisfies the composition contract and the composition contract refines the system specification to imply that the block diagram satisfies the system specification.

1.1. Contributions. Contributions of this paper are summarized as follows:

(i) We define the formal semantics for Simulink block diagrams from the viewpoint of dynamic systems to precisely express the trajectory of the Simulink block diagrams with *discrete-time* and *continuous-time*

(ii) Under the semantics, we propose a contract-based refinement technique mentioned above for proving the correctness of the Simulink block diagrams with *discrete-time* and *continuous-time* blocks

(iii) We demonstrate the effectiveness of our method through a case study of the control system of a reservoir that is modelled with Simulink block diagrams

1.2. Organization. The remainder of this paper is organized as follows. Section 2 reviews the related works. Section 3 introduces the notations and notions used in our work. Section 4 presents the semantics of the Simulink block diagrams. Section 5 proposes the contract-based refinement method for proving the correctness of Simulink diagram blocks. Section 6 demonstrates the effectiveness of our method with a case study. Section 7 concludes the paper and proposes future works.

2. Related Works

In recent years, there are a range of methods to analyze and verify Simulink.

There exist some works that translate Simulink into other formal modeling languages. For example, Tripakis et al. [5] translated the Simulink block diagrams to a synchronous dataflow language, Lustre. Since the Lustre has a *discrete-time* semantics, the work [5] only handled *discrete-time* Simulink block diagrams. In [6], Cavalcanti et al. presented a semantics for *discrete-time* Simulink blocks diagrams called Circus. The work [6] was based on existing tools that generate CSP and Z specifications from *discrete-time* block diagrams. It only translated *discrete-time* Simulink blocks diagrams, Simulink block diagrams with *continuous-time* were not considered.

Chen and Dong [15, 16] presented the method to automatically transformed Simulink diagrams with *discrete-time* and *continuous-time* into Timed Interval Calculus (TIC) models. This method applied the Prototype Verification System (PVS) to validate that TIC fulfils requirements. These works were the first attempt to model Simulink block diagrams with *continuous-time*. The work [17] presented an operational semantics for Simulink's simulation engine that formally defines the numerical simulation result, including *discrete-time* and *continuous-time*. Zou et al. [18]

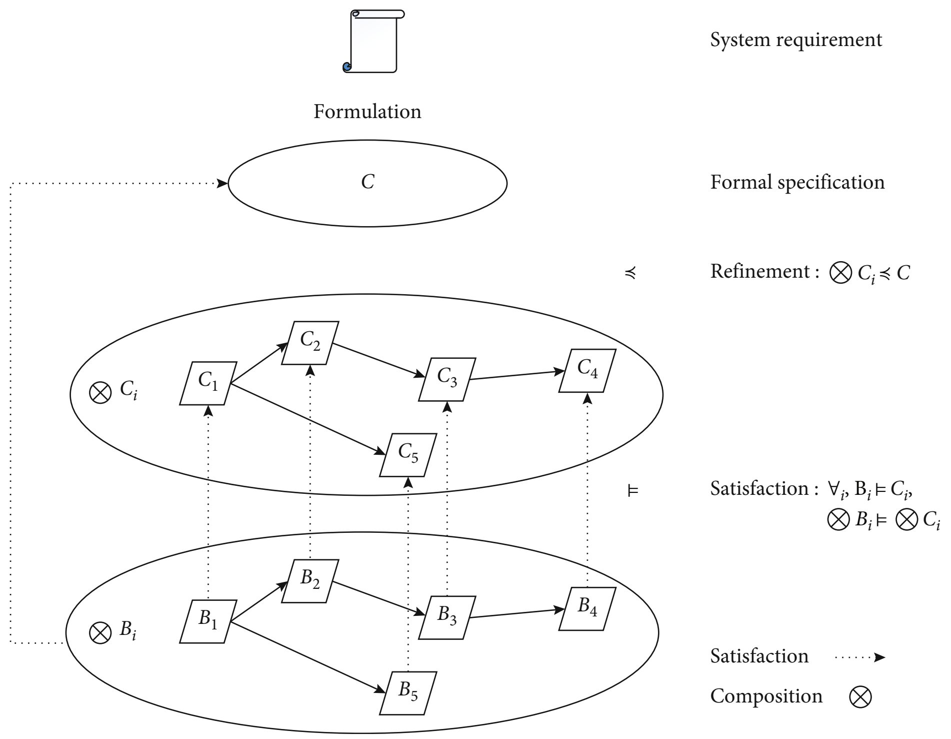


FIGURE 1: An overview of contract-based correctness proof methods for Simulink block diagrams, where the dashed line and $\vDash$: satisfaction; $\preccurlyeq$: refinement; $\otimes$: composition.

automatically translated Simulink block diagrams into Hybrid Communicating Sequential Processes (HCSP) and showed how the translated HCSP models are verified using the Hybrid Hoare Logic Verifier. In [19], it showed how different Simulink blocks can be expressed in the synchronous language Ze$'$lus, which extends a language Lustre with ODEs and zero-crossing events. The main difference between the articles [15–19] and our work is intentions. Our main goal here is not to translate the Simulink block diagrams to other formalisms, nor to define the semantics of Simulink's engine, but to directly define the trajectory of the Simulink block diagrams from the point of view of dynamic systems and provide a compositional and refinement technology to prove the correctness of Simulink block diagrams with *discrete-time* and *continuous-time*. On the other hand, we have three basic composition operators, i.e., serial, parallel, and algebraic loop-free feedback composition with multistep delays, which more facilitate the composition in construction. However, the work [18] could verify hybrid system, which is not considered in our work.

The contract-based approaches for verifying the correctness of Simulink block diagrams were also widely studied in the literature. Bostro et al. [7] showed definitions of contract and refinement using the action systems for Simulink models, while refinement provides a framework for reasoning about implementation correctness. However, this work only focuses on *discrete-time* Simulink block diagrams. Ye et al. [10, 11] defined a theoretical reasoning framework for Simulink block diagrams using Unifying Theories of Programming (UTP). The main idea of these papers is to translate each block or subsystem to a design, and the hierarchical connections of blocks are mapped to a variety of compositions of designs, and verify some properties. However, these papers only handled *discrete-time* Simulink block diagrams. In our work, we provide a compositional and refinement technology to prove the correctness of Simulink block diagrams with *discrete-time* and *continuous-time*. Dragomir et al. [13] recently presented a Refinement Calculus of Reactive Systems (RCRS) toolset for compositional formal modeling and reasoning about discrete and continuous reaction systems. RCRS is a *discrete-time* framework. The continuous systems can be modeled by discretizing time. However, the discretization of continuous systems reduces the accuracy of the verification of continuous dynamics. Our approach differs because we can directly represent and theoretically verify the correctness of *discrete-time* and *continuous-time* Simulink block diagrams. Moreover, we study the algebraic loop-free feedback composition with multistep delays.

3. Preliminary

In this section, we introduce some notations and notions that will be used in our work. We denote the set of natural numbers by $\mathcal{N}$, i.e., $\{0, 1, 2, \cdots\}$, the set of positive integers by $\mathcal{N}^+$, i.e., $\{1, 2, \cdots\}$, the set of positive real numbers $\mathcal{R}^+$, and the set of nonnegative real numbers by $\mathcal{R}_0^+$. We denote the set of integers between 1 and n by $[n] = \{1, \cdots, n\} \subset \mathcal{N}^+$. We denote vectors by bold fonts, and their components are indexed from 1 to n; for example, $\mathbf{x} = (x_1, \cdots, x_n)$, and x_i is the *i-th* component of $\mathbf{x}$, $i \in [n]$.

Following the definitions of vector addition and scalar multiplication in vector spaces, for two vectors $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$, the vector addition can be expressed as $\mathbf{x} + \mathbf{y} = (x_1 + y_1, x_2 + y_2)$ and the scalar multiplication can be expressed as $k(x_1, x_2) = (kx_1, kx_2)$, where $k \in F$ and F is a field.

We fix a *time axis* $\mathscr{R}_0^+$. The *continuous-time* domain T_c is a subset of $\mathscr{R}_0^+$ with a left endpoint equal to 0. The right endpoint may be open or closed. For any $\tau \in \mathscr{R}^+$, the *discrete-time* domain is a set $T_\tau = \{t_k | t_k = k\tau, k \in \mathscr{N}\}$, where the time instant t_k is called *sample time point*, $t_0 = 0$ is the initial time, and t_k keeps increasing at every iteration, i.e., $\forall k \in \mathscr{N}, t_k < t_{k+1}$.

We denote the n-dimensional real-valued vector space by $\mathscr{R}^n$, where $n \in \mathscr{N}^+$. A *signal* is a function from a time-domain T to $Z \subseteq \mathscr{R}^n$, i.e., $\mathbf{z}(\cdot)$: $T \mapsto Z$ (or $\mathbf{z}$ for short). We use Z^T to denote a set of all signals $\mathbf{z}$. A *continuous-time* signal is a signal defined over a *continuous-time* domain T_c. A *discrete-time* signal is a signal that is defined at the *discrete-time* domain T_τ. The values of the *discrete-time* signal update at each t_k and remain constant in the intervals $[k\tau, (k+1)\tau), k \in \mathscr{N}$.

4. Simulink Block Diagrams and Semantics

In this section, we present the semantics of Simulink block diagrams. We will start by giving a brief introduction to Simulink block diagrams, and we highlight the features relevant to our work. For more details, we refer the reader to [3].

4.1. Introduction of Simulink Block Diagrams. A Simulink block diagram is a graphical representation of a dynamic system. The Simulink block diagrams are composed of blocks and wires. The blocks can be either *elementary block*s provided by the Simulink library or composition blocks made up of *elementary blocks* or other composition blocks. An example that implements the relationship between the vehicle's power, resistance, and speed is shown in Figure 2. It consists of Constant, Subsystem, and Scope, where Constant and Scope are *elementary blocks*, and the composition block Subsystem comprises four *elementary blocks* Gain1, Subtract, Integrator, and Gain2.

In its most general form, the *elementary block* has *Inputports* and *Outputports* that receive the input signals and send the output signals. For some special blocks, the absence of *Inputports* or *Outputports* is also allowed. We will explain the details later. An *elementary block* is either *stateful* or *stateless*. We say a block is *stateful* if the output of this block depends on its inputs and internal states (i.e., memory). We say a block is *stateless* if the output depends only on its inputs. Wires transmit signals in the direction indicated by the arrow. It must transmit signals from *Outputports* of one block to *Inputports* of another block in terms of its sample times. An exception to this is that one wire can be drawn from another. This sends the original signal to two (or more) target blocks. Wire communication is instant. That is, when a block outputs the value to a wire, all blocks connected to that wire will see the new value simultaneously.

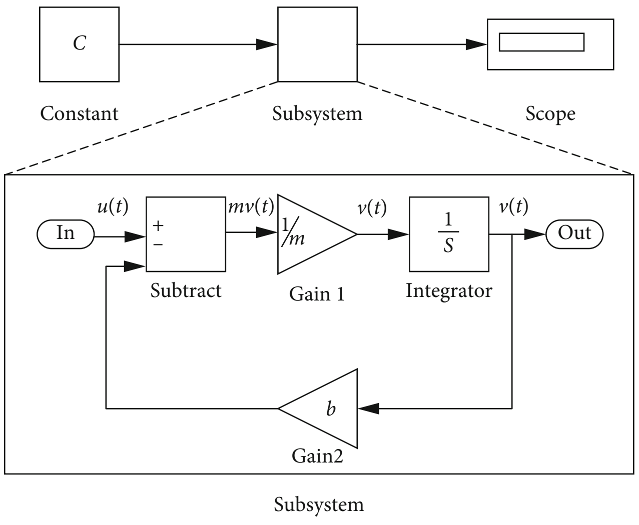


FIGURE 2: A Simulink block diagram.

To improve the modeling capabilities of Simulink, each *elementary block* contains some user-tunable parameters. One of the significant parameters is sample time that indicates the rate at which the block executes in simulation. According to the sample time, the blocks are divided into two main categories: continuous and discrete blocks. The sample time comprises two parameters: *sample time period* τ and *initial time offset* θ. For the continuous blocks, the *sample time period* $\tau = 0$. For the discrete blocks, the *sample time period* τ is always greater than zero and less than the simulation end time and θ less than or equal to τ. Since the default value of the initial offset is 0, unless otherwise mentioned, we let the initial offset θ be 0. As an example, suppose that the time unit is seconds, let the *sample time period* of a block be 0.02 s, and then the block updates methods (update, derivative or output) each 0.02 s. The Simulink block diagrams can be single rate where all blocks run with the same period or multirate where blocks run on different periods. This work considers single rate *discrete-time* Simulink block diagrams and *continuous-time* Simulink block diagrams with $\tau = 0$.

There are three basic composition operators in Simulink: (i) Serial composition is the composition that the output of the source block is connected to the input of the target block. (ii) Parallel composition is that two blocks are "stacked on top of each other" without any wires between the two blocks. (iii) Feedback composition is that the output of a block connects to one of its inputs. Other forms of composition can be assembled from these three basic composition operations and wires.

4.2. The Semantics of Simulink Block Diagrams. In this subsection, we formally define the semantics for Simulink block diagrams. We focus on the semantics of *elementary block*, composition semantics for composition operators, and the semantics of communication between blocks.

As introduced in 4.1, an *elementary block* has *Inputports* which receive the input signals, internal state, and *Outputports* that send the output signals. It describes a mathematical relationship between inputs, outputs, and internal states to capture dynamic behavior. We define an *elementary block* as a dynamical system.

Definition 1 (An elementary block). An *elementary block B* is a tuple $(\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$, where

(i) $\mathbf{x} : T \mapsto X$ is an input signal, $\mathbf{x}(t) \in X \subseteq \mathscr{R}^n$, $\mathbf{x}(t_0)$ is the initial value of the input signal

(ii) $\mathbf{y} : T \mapsto Y$ is an output signal, $\mathbf{y}(t) \in Y \subseteq \mathscr{R}^m$, $\mathbf{y}(t_0)$ is the initial value of the output signal

(iii) $\mathbf{s} : T \mapsto S$ is an internal state signal, $\mathbf{s}(t) \in S \subseteq \mathscr{R}^p$, $\mathbf{s}(t_0)$ is the initial value of the internal state signal

(iv) φ is the transition function of internal state

(v) f is the output function, i.e., $\mathbf{y}(t) = f(\mathbf{x}(t), \mathbf{s}(t))$

The above definition can express both *discrete-time* blocks and *continuous-time* blocks. We say that a block is continuous if it operates on *continuous-time* signals. We say a block is discrete if it operates on *discrete-time* signals. Every block must define its output function and may define initialize, update, or derivative function to realize the corresponding function. For the *discrete-time* blocks, we denote $\varphi = \varphi_u$, i.e., $\mathbf{s}(t_{k+1}) = \varphi_u(\mathbf{x}(t_k), \mathbf{s}(t_k))$, which refers to the update function of internal state. For the *continuous-time* blocks, we denote $\varphi = \varphi_d$, i.e., $\dot{\mathbf{s}}(t) = \varphi_d(\mathbf{x}(t), \mathbf{s}(t))$, which refers to the derivative function of internal state. The input or internal state of a block can be empty, respectively. In that case, we denote the input or internal state by $\mathbf{x} = \mathbf{0}$ or $\mathbf{s} = \mathbf{0}$ and use a symbol "-" to denote the transition function of the internal state. If a block has no internal state, we say the block is *stateless*.

Some *elementary blocks* mentioned in our article are shown in Table 1 Three simple examples are shown below.

Example 1 (Unitdelay). An example of a stateful *discrete-time* elementary block is the Unitdelay block. The Unitdelay expresses that the current output value of this block is equal to the value of the current internal state, and the value of the next internal state is equal to the current input value. It can be represented as $B_u = (\mathbf{x}_u, \mathbf{s}_u, \mathbf{y}_u, \mathbf{s}_u(t_{k+1}) = \mathbf{x}_u(t_k), \mathbf{y}_u(t_k) = \mathbf{s}_u(t_k))$, where $t_k \in T_\tau$.

Example 2 (Multistep delays). A block with multistep delays is denoted as $B = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$, where $\mathbf{y}(t_{n\cdot k}) = \mathbf{s}(t_{n\cdot k})$, and $\mathbf{s}(t_{n\cdot(k+1)}) = \mathbf{x}(t_{n\cdot k})$, where n means that the block outputs the input of this block after n sample periods, $n \in \mathscr{N}^+, k \in \mathscr{N}$.

Example 3 (Integrator). An example of a stateful *continuous-time* elementary block is the Integrator. The block models the relations, $\dot{\mathbf{s}}(t) = \mathbf{x}(t)$ with $y(t) = s(t)$. The Integrator can be represented as $B_I = (\mathbf{x}_I, \mathbf{s}_I, \mathbf{y}_I, \dot{\mathbf{s}}_I(t) = \mathbf{x}_I(t), \mathbf{y}_I(t) = \mathbf{s}_I(t))$, where $t \in \mathscr{R}_0^+$.

We use the observable trajectory to model the evolution of the input-output variables.

Definition 2 (The observable trajectory of the *discrete-time* elementary block). Let $B = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi_u, f)$ be a *discrete-time* elementary block. An observable trajectory Tr of B is a set $\{(\mathbf{x}, \mathbf{y}) : T_\tau \mapsto X \times Y | \exists \mathbf{s} \mathbf{y}(t_k) = f(\mathbf{s}(t_k), \mathbf{x}(t_k)) \wedge \mathbf{s}(t_{k+1}) = \varphi_u(\mathbf{x}(t_k), \mathbf{s}(t_k)), k \in \mathscr{N}\}$, where $\mathbf{x} : T_\tau \mapsto X \subseteq \mathscr{R}^n$ is the input trajectory and $\mathbf{y} : T_\tau \mapsto Y \subseteq \mathscr{R}^m$ *is* the output trajectory.

Definition 3 (The observable trajectory of the *continuous-time* elementary block). Let $B = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi_d, f)$ be a *continuous-time* elementary block. An observable trajectory Tr of B is a set $\{(\mathbf{x}, \mathbf{y}) : T_c \mapsto X \times Y | \exists \mathbf{s} \mathbf{y}(t) = f(\mathbf{s}(t), \mathbf{x}(t)) \wedge \dot{s}(t) = \varphi_d(\mathbf{x}(t), \mathbf{s}(t))\}$, where $\mathbf{x} : T_c \mapsto X \subseteq \mathscr{R}^n$ is the input trajectory and $\mathbf{y} : T_c \mapsto Y \subseteq \mathscr{R}^m$ is the output trajectory.

The wires connect some *Outputports* of a block to some *Inputports* of other block for communicating. For any blocks, $B_1 = (\mathbf{x}_1, \mathbf{s}_1, \mathbf{y}_1, \varphi_1, f_1)$ and $B_2 = (\mathbf{x}_2, \mathbf{s}_2, \mathbf{y}_2, \varphi_2, f_2)$, where $\mathbf{x}_1 = (x_{1,1}, \cdots, x_{1,n})$, $\mathbf{y}_1 = (y_{1,1}, \cdots, y_{1,m})$, $\mathbf{x}_2 = (x_{2,1}, \cdots, x_{2,n})$, and $\mathbf{y}_2 = (y_{2,1}, \cdots, y_{2,m})$. We model the wires between B_1 and B_2 as a relation, called unilateral connection. A unilateral connection is a set of variables pair $(y_{1,j}, x_{2,i})$, for $j \in [m]$ and $i \in [n]$, where the former of each pair is the output variable that comes from B_1, the latter of each pair is the input variable that comes from B_2. We define the unilateral connection as follows:

Definition 4 (Unilateral connection). Given two blocks $B_1 = (\mathbf{x}_1, \mathbf{s}_1, \mathbf{y}_1, \varphi_1, f_1)$ and $B_2 = (\mathbf{x}_2, \mathbf{s}_2, \mathbf{y}_2, \varphi_2, f_2)$, we define a unilateral connection from B_1 to B_2 (and vice versa) as a relation $\rho = \{(y_{1,j}, x_{2,i}) | j \in [m], i \in [n]\}$ satisfying that:

(i) $y_{1,j}$ is the *j-th* component of $\mathbf{y_1}$, for all $j \in [m]$

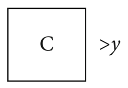
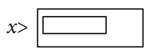
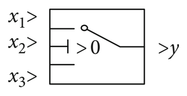
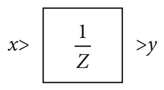
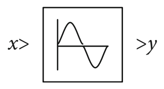
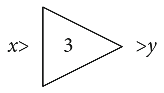
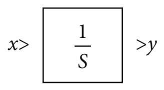
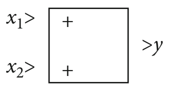
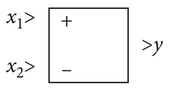
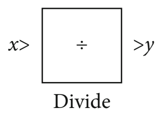
(ii) $x_{2,i}$ is the *i-th* component of $\mathbf{x_2}$, for all $i \in [n]$

(iii) $y_{1,j}(t) = x_{2,i}(t)$, for all $t \in T$

Note that not all the blocks can be connected. Given a source block $B_1 = (\mathbf{x}_1, \mathbf{s}_1, \mathbf{y}_1, \varphi_1, f_1)$ and a target block $B_2 = (\mathbf{x}_2, \mathbf{s}_2, \mathbf{y}_2, \varphi_2, f_2)$, if we connect the output variables of the source block with input variables of the target block, variable names and their types (the sets of values that a variable can take) impose some constraints: First, the names of the input and output variables should not conflict. Second, the types should match and $Y_1 \subseteq X_2$, where Y_1 is the type of $\mathbf{y}_1$ and X_2 is the type of $\mathbf{x}_2$. Third, one output port can connect to many input ports, but an input port can connect to at most one output ports.

For readability, we use $\{\rho_1, \cdots, \rho_n\} \subseteq \rho$ to denote a set of specific unilateral connection (determined by a relation as defined in Definition 4). Note that, since $\rho = \{(y_{1,j}, x_{2,i}) | j \in$

Table 1: The representation of some *elementary blocks* and their semantics.

Library	Description	Elementary block	Semantics
Source	Constant Constant value		$B = (\mathbf{0}, \mathbf{0}, \mathbf{y}, -, \mathbf{y}(t) = c)$
Sinks	Display System output		$B = (\mathbf{x}, \mathbf{0}, \mathbf{y}, -, \mathbf{y}(t) = \mathbf{x}(t))$
Signal routing	Switch Conditional statement		$B = (\mathbf{x}, \mathbf{0}, \mathbf{y}, -, f)$ $\mathbf{y}(t) = \begin{cases} \mathbf{x}_1(t) & \mathbf{x}_2(t) > 0, \\ \mathbf{x}_3(t) & \mathbf{x}_2(t) \le 0 \end{cases}$
Discrete	Unitdelay Discrete-time delay		$B = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$ $\varphi : \mathbf{s}(t_{k+1}) = \mathbf{x}(t_k)$ $f : \mathbf{y}(t_k) = \mathbf{s}(t_k)$
Discrete/continuous	Sine wave Discrete-time		$B = (\mathbf{x}, \mathbf{0}, \mathbf{y}, -, y(t_k) = \sin x(t_k))$
Math operations	Gain Math operation		$B = (\mathbf{x}, \mathbf{0}, \mathbf{y}, -, \mathbf{y}(t) = 3\mathbf{x}(t))$
Continuous	Integrator Continuous-time		$B = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$ $\varphi : \dot{\mathbf{s}}(t) = \mathbf{x}(t), f : y(t) = s(t)$
Math operations	Sum Math operation		$B = (\mathbf{x}, \mathbf{0}, \mathbf{y}, -, f), \mathbf{x} = (x_1, x_2)$ $\mathbf{y}(t) = \mathbf{x_1}(t) + \mathbf{x_2}(t))$
Math operations	Subtraction Math operation		$B = (\mathbf{x}, \mathbf{0}, \mathbf{y}, -, f), \mathbf{x} = (x_1, x_2), \mathbf{y}(t) = x_1(t) - x_2(t))$
Math operations	Divide Math operation		$B = \left(\mathbf{x}, \mathbf{0}, \mathbf{y}, -, \mathbf{y}(t) = \dfrac{1}{x(t)}\right)$

$[m], i \in [n]\}$ is a relation, we can also have $\rho(y_{1,j}) = \{x_{2,i} | (y_{1,j}, x_{2,i}) \in \rho, j \in [m], i \in [n]\}$.

4.3. Composition. A Simulink block diagram is a composition block constructed by *elementary blocks* according to composition operators. We will define the semantics of three basic composition operators in the following, namely, serial, parallel, and algebraic loop-free feedback composition with multistep delays. We first consider the serial composition.

We can compose blocks B_1 and B_2 to form a serial composition when there is a *unilateral connection* between B_1 and B_2 and the *unilateral connection* satisfies the connection rules, as shown in Figure 3(a).

Definition 5 (Serial composition). Given blocks $B_1 = (\mathbf{x}_1, \mathbf{s}_1, \mathbf{y}_1, \varphi_1, f_1)$ and $B_2 = (\mathbf{x}_2, \mathbf{s}_2, \mathbf{y}_2, \varphi_2, f_2)$, a serial composition $B_1 ; B_2$ *of* B_1 and B_2 is defined as a block $(\mathbf{x}, \mathbf{s}, \mathbf{y}, f, \varphi)$ if there exists a unilateral connection and

$$\begin{aligned}
&\mathbf{x} \coloneqq \mathbf{x}_1, \\
&\mathbf{s} \coloneqq (\mathbf{s}_1, \mathbf{s}_2), \\
&\mathbf{y} \coloneqq \mathbf{y}_2, \\
&\varphi \coloneqq (\varphi_1(\mathbf{x}_1, \mathbf{s}_1), \varphi_2(\mathbf{x}_2, \mathbf{s}_2)), \\
&f \coloneqq (f_1 ; \rho ; f_2)(\mathbf{x}, \mathbf{s}) = f_2(\rho(f_1(\mathbf{x}_1, \mathbf{s}_1)), \mathbf{s}_2).
\end{aligned}$$

The definition of the serial composition of *elementary blocks* coincides with the definition of an *elementary block* (i.e., Definition 1), and therefore a composition can be considered a block.

Example 4 (Serial composition). In Figure 3(b), the stateless block Gain takes input $\boldsymbol{x}_1$, computes $k \in \mathscr{R}$ times of $\mathbf{x_1}$, and returns $\mathbf{y}_1$ as output, where $\mathbf{x}_1 : \mathscr{R} \mapsto \mathscr{R}$ and $\mathbf{y}_1 : \mathscr{R} \mapsto \mathscr{R}$.

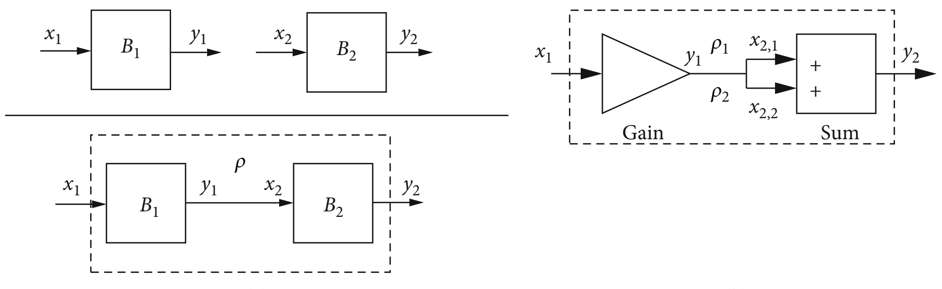


FIGURE 3: (a) Graphical representation of the serial composition of blocks, where $\rho = \{(y_1, x_2)\}$ is a *unilateral connection* between B_1 and B_2, the dashed box indicates a composition of blocks. (b) Graphical representation of an example of serial composition, where $\rho = \{(y_1, x_{2,1}), (y_1, x_{2,2})\}$ is a *unilateral connection* between Gain and Sum.

Another stateless block Sum has the input $\boldsymbol{x}_2 = (x_{2,1}, x_{2,2})$ and output $\boldsymbol{y}_2(t) = x_{2,1}(t) + x_{2,2}(t)$, where $\mathbf{x}_2 : \mathscr{R} \mapsto \mathscr{R} \times \mathscr{R}$ and $\mathrm{y}_2 : \mathscr{R} \mapsto \mathscr{R}$, $\forall t \in \mathscr{R}$. *Let* ρ be the unilateral connection between Gain and Sum. Then, we can express these blocks as follows:

$$\begin{aligned} \text{Gain} &:= (\mathbf{x}_1, \mathbf{0}, \mathbf{y}_1, -, \mathbf{y}_1(t) = \text{gain}(x_1(t), \mathbf{0})), \\ \rho &:= \{(y_1, x_{2,1}), (y_1, x_{2,2})\}, \\ \text{Sum} &:= (\mathbf{x}_2, \mathbf{0}, \mathbf{y}_2, -, y_2(t) = \text{add}(\mathbf{x}_2(t), \mathbf{0})), \end{aligned}$$

where $y_2(t) = \text{add}(\mathbf{x_2}(\mathbf{t}), \mathbf{0}) = x_{2,1}(t) + x_{2,2}(t)$, $y_1(t) = \text{gain}(x_1(t), \mathbf{0}) = k \cdot x_1(t)$, $y_1(t) = x_{2,1}(t), y_1(t) = x_{2,2}(t)$. According to Definition 5, we compute the output function of the serial composition as follows:

$$\begin{aligned} f(\mathbf{x}_1(t), \mathbf{0}) &:= \text{add}(\rho(\text{gain}(\mathbf{x}_1(t), \mathbf{0}))) \\ &= \text{add}(\rho(\mathbf{y}_1(t)), \mathbf{0}) = \text{add}(\mathbf{x}_2(t), \mathbf{0}) \\ &= x_{2,1}(t) + x_{2,2}(t) = 2k \cdot x_1(t). \end{aligned}$$

Therefore, we have $\text{Gain} ; \text{Adder} := (\mathbf{x}_1, \mathbf{0}, \mathbf{y}_2, -, \mathbf{y}_2(t) = 2k \cdot \mathbf{x_1}(t))$.

The serial composition of blocks satisfies associative laws.

Lemma 6 (Associativity). *Given blocks B_1, B_2, and B_3, we have $(B_1 ; B_2) ; B_3 = B_1 ; (B_2 ; B_3)$.*

Proof. It is easy to verify using Definition 5. Thus, if we compose multiple blocks in serial, we can first compose two blocks, compose the result with a third one, ⋯, and rearranging the brackets in the expression does not change the result as long as the block's position remains the same. □

Parallel composition is a particular case of composition with connection, where the *unilateral connection* between B_1 and B_2 is an empty set. We define parallel composition as follows.

Definition 7 (Parallel composition). Given blocks $B_1 = (\boldsymbol{x}_1, \boldsymbol{s}_1, \boldsymbol{y}_1, \varphi_1, f_1)$ and $B_2 = (\boldsymbol{x}_2, \boldsymbol{s}_2, \boldsymbol{y}_2, \varphi_2, f_2)$, we define the parallel composition $B_1 \| B_2$ of B_1 and B_2 as a block $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}, f, \varphi)$, where

$$\begin{aligned} \mathbf{x} &:= (\mathbf{x}_1, \mathbf{x}_2), \\ \mathbf{s} &:= (\mathbf{s}_1, \mathbf{s}_2), \\ \mathbf{y} &:= (\mathbf{y}_1, \mathbf{y}_2), \\ \varphi &:= (\varphi_1 \| \varphi_2)((\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2)) \\ &= (\varphi_1(\mathbf{x}_1, \mathbf{s}_1), \varphi_2(\mathbf{x}_2, \mathbf{s}_2)), \\ f &:= (f_1 \| f_2)((\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2)) \\ &= (f_1(\mathbf{x}_1, \mathbf{s}_1), f_2(\mathbf{x}_2, \mathbf{s}_2)). \end{aligned}$$

Example 5 (Parallel composition). Consider the parallel composition shown in Figure 4(a). The block Divide models the relation $\mathbf{y}_1(t) = 1/\mathbf{x}_1(t)$, where $\mathbf{x}_1$ is the input signal and requires $x_1(t) \neq 0$, $\mathbf{y}_1$ is the output signal, and t represents time. The Sine Wave block models the relation $\mathbf{y}_2(t) = \mathbf{x}_2(t)$, where $\mathbf{x}_2$ is the input variable and $\boldsymbol{y}_2$ is the output variable. Therefore, the Divide and Sine Wave can be represented as

$$\text{Divide} := \left(\mathbf{x}_1, \mathbf{0}, \mathbf{y}_1, -, \mathbf{y}_1(t) = \frac{1}{\mathbf{x}_1(t)}\right),$$

$$\text{Sine Wave} := (\mathbf{x}_2, \mathbf{0}, \mathbf{y}_2, -, \mathbf{y}_2(t) = \mathbf{x}_2(t)).$$

Following Definition 7, we write the parallel composition as follows:

$$\text{Divide}\|\text{Sine Wave} := \left((\mathbf{x}_1, \mathbf{x}_2), \mathbf{0}, (\mathbf{y}_1, \mathbf{y}_2), -, \left(\mathbf{y}_1(t) = \frac{1}{\mathbf{x}_1(t)}, \mathbf{y}_2(t) = \mathbf{x}_2(t)\right)\right).$$

Note that Definition 5 defines the case that all the *Outputports* of B_1 match the *Inputports* of B_2. However, not all *Outputports* of B_1 match the *Inputports* of B_2 and not all *Outputports* of B_1 are connected to all the *Inputports* of B_2. As an example, we consider Figure 4(b). To handle this composition, we introduce a particular *elementary block Id* representing its output is identical to its input. We model the wires that connect the $y_{1,1}$ and $x_{2,2}$ as Id_1 and Id_2, respectively. We then denote the composition by $B = ((B_1 \| \mathrm{Id}_1) ; \rho ; (B_1 \| \mathrm{Id}_2))$, where $\rho = \{(y_{1,1}, x_{Id_2}), (y_{1,2}, x_{2,1}), (y_{Id_1}, x_{2,2})\}$.

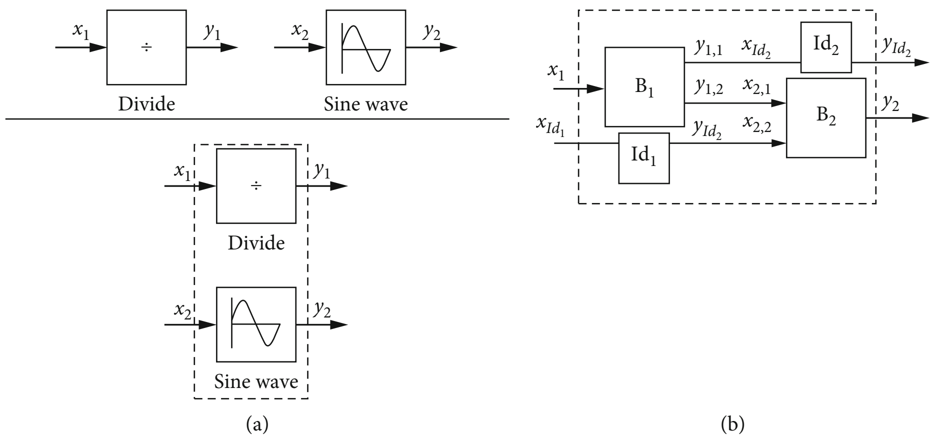


FIGURE 4: (a) Graphical representation of parallel composition, where the dashed box indicates a composition of blocks. (b) Graphical representation of a composition with serial and parallel, where the dashed box indicates a composition of blocks.

The parallel composition of blocks satisfies commutative law and associative law.

Lemma 8. *Given blocks B_1, B_2 and B_3, we have*

(i) $B_1 \| B_2 = B_2 \| B_1$

(ii) $(B_1 \| B_2) \| B_3 = B_1 \| (B_2 \| B_3)$

Proof. Straightforward from the Definition 7. □

We will next define the algebraic loop-free feedback composition with multistep delays. Before defining that, we first introduce the algebraic loop feedback to aid this definition. In Simulink, an algebraic loop occurs when an input port of a direct feedthrough block is driven by the output of the same block in the same time step. The direct feedthrough means that a stateless block computes its output only depending on the input value at the current time. In mathematics, an algebraic loop can be expressed as the algebraic equation $f(\mathbf{x}, \mathbf{y}) = 0$, where $\mathbf{x}$ is the input variable and $\mathbf{y}$ is the algebraic variable. Simulink solves the algebraic equation for $\mathbf{y}$ at each time instant.

A simple example of an algebraic loop is the feedback that represents in Figure 5(a). The feedback is a Subtraction block with input $\mathbf{x} = (x_{1,1}, x_{1,2})$ and an output $\mathbf{y}$. The first element $x_{1,1}$ of input is used to communicate with the environment. The second element $x_{1,2}$ is used for feedback. The output $\mathbf{y}$ is split into two equal signals: one is for output, and another is to feed the output back into input $x_{1,2}$. Then, this loop implies that the output of the Subtraction block is an algebraic variable y that is constrained to equal the first input $x_{1,1}$ minus $x_{1,2}$, i.e., $x_{1,1}(t) - x_{1,2}(t) = y(t)$. Let $x_{1,2}(t) = y(t)$ and $x_{1,1}(t) = u(t)$, we have $u(t) - y(t) = y(t)$, and then the solution of this loop is $y(t) = u(t)/2$.

However, the algebraic loop has inherent difficulties in solving: (1) while Simulink solver solves the algebraic loop, the simulation can execute slowly. (2) Some algebraic loops have no solution (an example shown in Figure 5(b), the expression of this loop is $u(t) + y(t) = y(t)$, etc. These problems lead to algebraic loops that are undesirable. To remove the algebraic loop, according to Simulink, we connect $\mathbf{y}$ to $x_{1,2}$ by *stateful* blocks (such as Delay, Memory, or Integrator) in this feedback to break the algebraic loop. We refer to this kind of acyclic structure as algebraic loop-free feedback, as shown in Figure 6. The work [14] defined the feedback with Unitdelay. We will handle the case of multistep delays.

Definition 9 (Algebraic loop-free feedback composition with multistep delays). Let blocks $B_1 = (\mathbf{x}_1, \mathbf{s}_1, \mathbf{y}_1, \varphi_1, f_1)$ be feedback with an algebraic loop and $B_2 = (\mathbf{x}_2, \mathbf{s}_2, \mathbf{y}_2, \varphi_2, f_2)$ be a multistep delays block. Let $\rho_1 = (y_2, x_{1,2})$ and $\rho_2 = (y, x_2)$ be the unilateral connections between B_1 and B_2. We denote by $B_1 \otimes_f B_2$ the algebraic loop-free feedback composition of B_1 and B_2 with multistep delays and define $B_1 \otimes_f B_2 = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$, where

$$\begin{aligned}
\mathbf{x} &:= x_{1,1}, \\
\mathbf{s} &= (s_1, s_2), \\
\mathbf{y} &= y_1, \\
\varphi &:= (\varphi_1 \| \varphi_2)((x_1, s_1), (x_2, s_2)) \\
&= (\varphi_1(x_1, s_1), \varphi_2(x_2, s_2)), \\
f &:= \begin{cases} y(t_0) & t = t_0, \\ y(t_{n\cdot k}) & t = t_{n\cdot k}, n, k \in \mathcal{N}^+. \end{cases}
\end{aligned}$$

In the definition above, the calculation process starts with the multistep delays block. When $t = t_0$, let the initial state $s(t_0) = (0, s_2(t_0))$. The output's initial value of this algebraic loop-free feedback $y(t_0) = f_1((x_{1,1}(t_0), x_{1,2}(t_0)), s(t_0))$. There exists ρ_1, s.t. $x_{1,2}(t_0) = \rho_1(y_2(t_0))$, when $y_2(t_0) = f_2(x_2(t_0), s_2(t_0))$. Given a delay step $n \in \mathcal{N}^+$, when $t = t_{n\cdot k}$, where $k \in \mathcal{N}$, the output $y(t_{n\cdot k}) = f_1((x_{1,1}(t_{n\cdot k}), x_{1,2}(t_{n\cdot k})), \mathbf{s}(t_{n\cdot k}))$, where the internal state $\mathbf{s}(t_{n\cdot k}) = (0, s_2(t_{n\cdot k}))$, and $x_{1,2}(t_{n\cdot k}) = \rho_1(y_2($

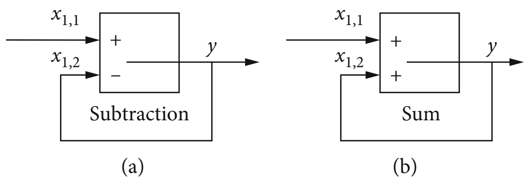


Figure 5: Graphical representation of feedback.

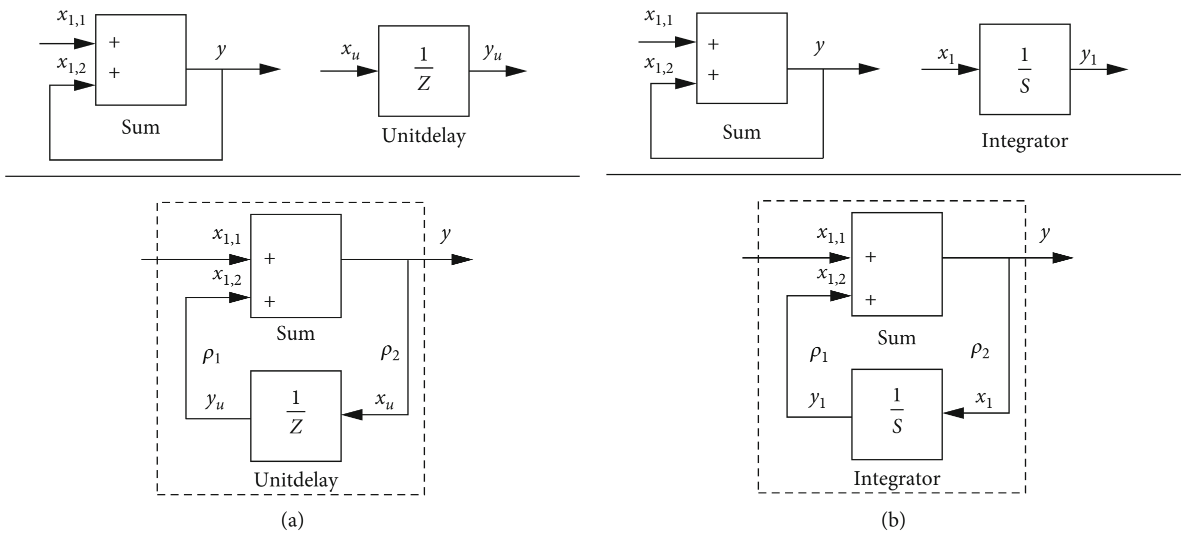


Figure 6: (a) Graphical representation of algebraic loop-free feedback composition with Unitdelay. (b) Graphical representation of algebraic loop-free feedback composition with Integrator.

$t_{n\cdot k}))$, where $y_2(t_{n\cdot k}) = f_2(x_2(t_{n\cdot k}), s_2(t_{n\cdot k}))$ is the output of B_2. There exists $\rho_2, s.t. x_2(t_{n\cdot k}) = \rho_2(y(t_{n\cdot(k-1)}))$ is the input of B_2, and $s_2(t_{n\cdot k}) = \varphi_2(x_2(t_{n\cdot(k-1)}), s_2(t_{n\cdot(k-1)}))$.

We next give an example to illustrate removing an algebraic loop by introducing Unitdelay, i.e., let the delay step $n = 1$.

Example 6 (An algebraic loop-free feedback composition with Unitdelay). In Figure 6(a), the Sum block $\text{Sum} = ((x_{1,1}, x_{1,2}), \mathbf{0}, y_1, -, y_1(t) = x_{1,1}(t) + x_{1,2}(t))$ is feedback with an algebraic loop. We connect $\mathbf{y}$ *to* $x_{1,2}$ in this feedback by the Unitdelay to break the algebraic loop. The Unitdelay block $B_u = (\mathbf{x}_u, \mathbf{s}_u, \mathbf{y}_u, \mathbf{s}_u(t_{k+1}) = \mathbf{x}_u(t_k), \mathbf{y}_u(t_k) = \mathbf{s}_u(t_k))$, where $t_k \in T_\tau$. Let $\rho_1 = (y_u, x_{1,2})$ and $\rho_2 = (y, x_u)$, the ρ_1 and ρ_2 satisfy the connection rules, and then an algebraic loop-free feedback composition with Unitdelay is a block $B_1 \otimes_f B_2 = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$, where $\mathbf{x} = x_{1,1}, \mathbf{s} = (\mathbf{0}, \mathbf{s}_u), \mathbf{y} = y$, and the output function

$$y(t) = \begin{cases} y(t_0) & t = t_0, \\ y(t_k) & t = t_k, k \in \mathcal{N}^+. \end{cases}$$

When $t = t_0$, let $s_u(t_0)$ be the initial value of the internal state of the Unitdelay. We have $y_u(t_0) = s_u(t_0)$ and $x_{1,2}(t_0) = y_u(t_0)$. Hence, we get the initial output value of this composition $y(t_0) = x_{1,1}(t_0) + s_u(t_0)$.

When $t = t_k, k \in \mathcal{N}^+$, we first consider Unitdelay. Because of $x_u(t_k) = y(t_k)$, $\mathbf{s}_u(t_{k+1}) = \mathbf{x}_u(t_k), \mathbf{y}_u(t_k) = \mathbf{s}_u(t_k)$, and $x_{1,2}(t_k) = y_u(t_k)$, we have $y(t_k) = x_{1,1}(t_k) + y(t_{k-1})$.

Next, we give an example about algebraic loop-free feedback composition with Integrator.

Example 7 (An algebraic loop-free feedback composition with Integrator). An example of continuous time algebraic loop-free feedback composition with Integrator is shown in Figure 6(b). As previously mentioned, the Sum block $\text{Sum} = ((x_{1,1}, x_{1,2}), \mathbf{0}, \mathbf{y}_1, -, y_1(t) = x_{1,1}(t) + x_{1,2}(t))$ is feedback with an algebraic loop. The Integrator can be represented as $B_I = (\mathbf{x}_I, \mathbf{s}_I, \mathbf{y}_I, \dot{\mathbf{s}}_I(t) = \mathbf{x}_I(t), y_I(t) = \mathbf{s}_I(t))$, where $t \in \mathcal{R}_0^+$. In this composition, there exist unilateral connections $\rho_1 = (y_I, x_{1,2})$ and $\rho_2 = (y, x_I)$, and the unilateral connections satisfy the connection rules, and then an algebraic loop-free feedback composition with Integrator is a block $B_1 \otimes_f B_2 = (\mathbf{x}, \mathbf{s}, \mathbf{y}, \varphi, f)$, where $\boldsymbol{x} = x_{1,1}, \boldsymbol{s} = (\mathbf{0}, \boldsymbol{s}_I)$. Let $s_I(0)$ be the initial value of the internal state of the Integrator. We have $y_I(0) = s_I(0)$, and there is a ρ_1, s.t. $x_{1,2}(0) = y_I(0)$. Hence, we get $y(0) = x_{1,1}(0) + x_{1,2}(0) = x_{1,1}(0) + y_I(0)$.

When $t \in \mathcal{R}^+$, similarly, we first consider Integrator. Because there is a ρ_2, s.t. $x_I(t) = y(t)$, and $\dot{s}_I(t) = x_I(t), y_I(t)$

$= s_I(t)$. There exists ρ_1, s.t. $x_{1,2}(t) = y_I(t)$, we get

$$y(t) = x_{1,1}(t) + x_{1,2}(t) = x_{1,1}(t) + \left(y_I(0) + \int_0^t x_I(\delta)d\delta \right)$$

5. A Contract-Based Refinement Approach

The correctness refers to the trajectories of Simulink block diagrams that should satisfy the requirement specifications of the system. Since contracts are centered around trajectories, they are expressive and versatile enough to specify *discrete-time* and *continuous-time* blocks. We use the contract as a specification language to formalize and prove system requirements and specify the trajectories of blocks. In this section, we present a generic contract-based method for proving the correctness of Simulink block diagrams with *discrete-time* and *continuous-time* blocks.

Because the requirement specifications of a system are informal, we first formalize system requirement specifications into system contract specifications. We construct the Simulink block diagram according to the contract specification. We aim to prove that the Simulink block diagrams satisfy the system contract specification. To this end, the approach is divided into two steps. First, we modularly verify that Simulink block diagrams satisfy the composition contract (i.e., the composition of contracts corresponding to the blocks that construct the Simulink block diagrams). We associate a contract for every elementary block as a specification. After that, we define the satisfaction relation that relates block to contract and verify that the satisfaction relation is preserved by composition. That is, if blocks satisfy their contract, respectively, then the composition of blocks satisfies the composition of contracts. Second, we define the refinement relation between contracts and verify that the composition of contracts refines the system contract specifications. If the block diagram satisfies the composition contracts and the composition contracts refine the system specification, then the block diagram also satisfies the system specification. Hence, the Simulink block diagram is correct with respect to the system specification.

We further elaborate on the concepts and properties mentioned above in the following. We first define the contract specification for the block.

Definition 10 (Contract). A contract is a tuple $C = (\boldsymbol{x}, \boldsymbol{y}, \phi_a, \phi_g)$, where

(i) $\mathbf{x}, \mathbf{y}$ are the input vectors and output vectors, respectively

(ii) $\phi_a \subseteq X^T$ represents the *assumption* for the input trajectories of the block, where $X \subseteq \mathscr{R}^n$

(iii) $\phi_g \subseteq (X \times Y)^T$ represents the *guarantee* for the input-output trajectories of the block, where $Y \subseteq \mathscr{R}^m$

The contract specifies the expected trajectory for each elementary block. An example of a contract for a block Gain is shown below.

Example 8 (Contract for Gain). As an example, we consider the Gain block that represented in Example 4. A contract specification for Gain is a tuple: $C = (\boldsymbol{x}, \boldsymbol{y}, \phi_a, \phi_g)$, where

$$\phi_a \coloneqq \left\{ \mathbf{x} \in \mathscr{R}^T \middle| \forall t \in T, x(t) \in \mathscr{R} \right\},$$

$$\phi_g \coloneqq \left\{ (\mathbf{x}, \mathbf{y}) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T, y(t) = k \cdot x(t) \right\}.$$

Next, we will define the satisfaction relation, which relates a block to a contract by determining when a given block's trajectories satisfy the specified specification. To define satisfaction relation, we first define a projection that expresses the input trajectory to aid that definition. A projection of a set $Tr \subseteq (X \times Y)^T$ into $\mathscr{X} \subseteq X^T$ is defined as the set $Tr\downarrow_{\mathscr{X}} = \{\mathbf{x} \in \mathscr{X} | \exists \mathbf{y}(\mathbf{x}, \mathbf{y}) \in Tr\}$, where T can be either the *discrete-time* domain or the *continuous-time* domain. We are now in a position to define satisfaction relation.

Definition 11 (Satisfaction). Let $B = (\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}, \varphi, f)$ be a block and $C = (\boldsymbol{x}, \boldsymbol{y}, \phi_a, \phi_g)$ be a contract. We say B satisfies C, denoted by $B \vDash C$, if $Tr\downarrow_{\mathscr{X}} \subseteq \phi_a$ and $Tr \subseteq \phi_g$.

We say that B is a correct implementation of C if $B \vDash C$. We have defined the notion of contract and satisfaction relation. The primary task of the proof method is to state that the satisfaction relation is preserved by composition. In the following, we study the composition operators of contracts according to the composition operators of blocks, i.e., serial, parallel, and algebraic loop-free feedback with multistep delays. We first define the serial composition of contracts.

Definition 12 (The serial composition of contracts). Let $C_i = (\mathbf{x}_i, \mathbf{y}_i, \phi_a^i, \phi_g^i)$ be contracts for $i = 1, 2$. The serial composition of C_1 and C_2, written $C_1 ; C_2$, is a contract $(\boldsymbol{x}, \boldsymbol{y}, \phi_a, \phi_g)$, where

$$\mathbf{x} = \mathbf{x}_1,$$

$$\mathbf{y} = \mathbf{y}_2,$$

$$\phi_a \coloneqq \left\{ \mathbf{x}_1 \in X^T \middle| \mathbf{x}_1 \in \phi_a^1 \wedge \left((\exists y_1) \left((\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge \phi_\rho \longrightarrow \mathbf{y}_1 \in \phi_a^2 \right) \right) \right\},$$

$$\phi_g \coloneqq \left\{ (\mathbf{x}_1, \mathbf{y}_2) \in \left(X \times Y \right)^T \middle| (\exists y_1 \exists x_2) \left((\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge \phi_\rho \wedge (\mathbf{x}_2, \mathbf{y}_2) \in \phi_g^2 \right) \right\},$$

$$\phi_\rho \coloneqq \bigwedge_{(y_{1,j}, x_{2,i}) \in \rho} y_{1,j}(t) = x_{2,i}(t), \forall t \in T.$$

The serial composition preserves the satisfaction relation. That is, if blocks satisfy their contracts, respectively, then the composition of blocks satisfies the composition of contracts.

Theorem 13 (Serial composition preserves satisfaction). *If$B_1 \vDash C_1$ and $B_2 \vDash C_2$, then $B_1 ; B_2 \vDash C_1 ; C_2$.*

Proof. We show a proof here for blocks with *continuous-time* trajectories, and the proof for blocks with *discrete-time* trajectories is similar. Let $B_1 = (\mathbf{x}_1, \mathbf{s}_1, \mathbf{y}_1, \varphi_1, f_1)$ and $B_2 = (\mathbf{x}_2, \mathbf{s}_2, \mathbf{y}_2, \varphi_2, f_2)$ be *continuous-time* blocks. The observable trajectories of B_1 and B_2 are denoted as $Tr_1 = \{(\mathbf{x}_1, \mathbf{y}_1): T \mapsto X_1 \times Y_1 | \exists \mathbf{s}_1 \mathbf{y}_1 = f(\mathbf{x}_1, \mathbf{s}_1) \wedge \dot{\mathbf{s}}_1 = \varphi_{d1}(\mathbf{x}_1, \mathbf{s}_1)\}$ and $Tr_2 = \{(\mathbf{x}_2, \mathbf{y}_2): T \mapsto X_2 \times Y_2 | \exists \mathbf{s}_2 \mathbf{y}_2 = f(\mathbf{s}_2, \mathbf{x}_2) \wedge \dot{\mathbf{s}}_2 = \varphi_{d2}(\mathbf{x}_2, \mathbf{s}_2)\}$, respectively, where $X_1, X_2 \subseteq \mathscr{R}^n$, $Y_1, Y_2 \subseteq \mathscr{R}^m$, $n, m \in \mathscr{N}^+$. The projection of Tr_1 into $\mathscr{X}_1$ is $Tr_1 \downarrow_{\mathscr{X}_1} = \{\mathbf{x}_1 | \exists y_1 (\mathbf{x}_1, \mathbf{y}_1) \in Tr_1\}$. The projection of Tr_2 into $\mathscr{X}_2$ is $Tr_2 \downarrow_{\mathscr{X}_2} = \{\mathbf{x}_2 | \exists y_2 (\mathbf{x}_2, \mathbf{y}_2) \in Tr_2\}$.

According to Definition 5, we write $B_1 ; B_2 = (\mathbf{x}_1, \mathbf{y}_2, \mathbf{s}, f, \varphi)$. We denote the observable trajectories of $B_1 ; B_2$ by Tr. Then$Tr = \{(\mathbf{x}_1, \mathbf{y}_2) | \exists \mathbf{s} \mathbf{y}_2 = f_2(\rho(f_1(\mathbf{x}_1, \mathbf{s}_1), \mathbf{s}_2)) \wedge \dot{\mathbf{s}} = \varphi_d(\mathbf{x}, \mathbf{s})\}$. The projection of Tr into $\mathscr{X}_1$ is$Tr \downarrow_{\mathscr{X}_1} = \{\mathbf{x}_1 | \exists y_1 \exists x_2 \exists y_2 (\mathbf{x}_1, \mathbf{y}_1) \in Tr_1 \wedge (\mathbf{y}_1, \mathbf{x}_2) \in \rho \wedge (\mathbf{x}_2, \mathbf{y}_2) \in Tr_2\}$. So, $Tr \downarrow_{\mathscr{X}_1} \subseteq Tr_1 \downarrow_{\mathscr{X}_1}$. □

Given two contracts $C_1 = (\mathbf{x}_1, \mathbf{y}_1, \phi_a^1, \phi_g^1)$ and $C_2 = (\mathbf{x}_2, \mathbf{y}_2, \phi_a^2, \phi_g^2)$, by Definition 12, we have $C_1 ; C_2 = (\mathbf{x}_1, \mathbf{y}_2, \phi_a, \phi_g)$, where $\phi_a = \{\mathbf{x}_1 \in X^T | \mathbf{x}_1 \in \phi_a^1 \wedge ((\exists y_1)((\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge \phi_\rho \longrightarrow \mathbf{y}_1 \in \phi_a^2))\}$ and $\phi_g = \{(\mathbf{x}_1, \mathbf{y}_2) \in (X \times Y)^T | (\exists y_1 \exists x_2)((\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge \phi_\rho \wedge (\mathbf{x}_2, \mathbf{y}_2) \in \phi_g^2)\}$.

In order to proof $B_1 ; B_2 \vDash C_1 ; C_2$, according to Definition 11, we only need to show that: if $Tr \downarrow_{\mathscr{X}_1} \subseteq \phi_a$ implies $Tr \subseteq \phi_g$. We first prove that $Tr \downarrow_{\mathscr{X}_1} \subseteq \phi_a$.

Since $B_1 \vDash C_1$, according to Definition 11, we have $Tr_1 \downarrow_{\mathscr{X}_1} \subseteq \phi_a^1$ implies $Tr_1 \subseteq \phi_g^1$. That is, for all $x_1 \in Tr \downarrow_{\mathscr{X}_1} \subseteq Tr_1 \downarrow_{\mathscr{X}_1} \subseteq \phi_a^1$, then $(\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1$. Since $B_2 \vDash C_2$, according to Definition 11, we have for all $x_2 \in Tr_2 \downarrow_{\mathscr{X}_2} \subseteq \phi_a^2$, then $Tr_2 \subseteq \phi_g^2$. There exists a connection $\rho = (\mathbf{y}_1, \mathbf{x}_2)$, according to Definition 4, we have $y_1(t) = x_2(t)$. Then, $\mathbf{y}_1 \in \phi_a^2$, and by the expression of ϕ_a, we have $\mathbf{x}_1 \in \phi_a$. Thus, $Tr \downarrow_{\mathscr{X}_1} \subseteq \phi_a$. For all $(\mathbf{x}_1, \mathbf{y}_2) \in Tr$, we have $(\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1$, $\mathbf{y}_1 = \mathbf{x}_2$, and $(\mathbf{x}_2, \mathbf{y}_2) \in \phi_g^2$. According to the expression of ϕ_g, we have $(\mathbf{x}_1, \mathbf{y}_2) \in \phi_g$. Thus, $Tr \subseteq \phi_g$. Hence, $B_1 ; B_2 \vDash C_1 ; C_2$.

We now define the parallel composition of contracts.

Definition 14 (The parallel composition of contracts). Given contracts $C_1 = (\boldsymbol{x}_1, \boldsymbol{y}_1, \phi_a^1, \phi_g^1)$ and $C_2 = (\boldsymbol{x}_2, \boldsymbol{y}_2, \phi_a^2, \phi_g^2)$, we define the parallel composition of contracts as $C_1 \| C_2 = (\boldsymbol{x}, \boldsymbol{y}, \phi_a, \phi_g)$, where $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{y} = (y_1, y_2)$, and

$$\phi_a \coloneqq \{\mathbf{x} \in X^T | x_1 \in \phi_a^1 \wedge x_2 \in \phi_a^2\},$$
$$\phi_g \coloneqq \left\{(\mathbf{x}, \mathbf{y}) \in (X \times Y)^T | (x_1, y_1) \in \phi_g^1 \wedge (x_2, y_2) \in \phi_g^2\right\}.$$

The parallel composition also preserves the satisfaction relation. That is, if blocks satisfy their contracts, respectively, then the parallel composition of blocks satisfies the parallel composition of contracts.

Theorem 15 (Parallel composition preserves satisfaction). *If$B_1 \vDash C_1$ and $B_2 \vDash C_2$, then $B_1 \| B_2 \vDash C_1 \| C_2$.*

Proof. We show here a proof for blocks with *continuous-time* trajectories, and the proof for blocks with *discrete-time* trajectories is similar. Let $B_1 = (\mathbf{x}_1, \mathbf{s}_1, y_1, \varphi_1, f_1)$ and $B_2 = (\mathbf{x}_2, \mathbf{s}_2, y_2, \varphi_2, f_2)$ be blocks with *continuous-time* trajectories. The observable trajectories of B_1 and B_2 are denoted as $Tr_1 = \{(\mathbf{x}_1, \mathbf{y}_1): T \mapsto X_1 \times Y_1 | \exists \mathbf{s}_1 \mathbf{y}_1 = f_1(\mathbf{x}_1, \mathbf{s}_1) \wedge \dot{\mathbf{s}}_1 = \varphi_{d1}(\mathbf{x}_1, \mathbf{s}_1)\}$ and $Tr_2 = \{(\mathbf{x}_2, \mathbf{y}_2): T \mapsto X_2 \times Y_2 | \exists \mathbf{s}_2 \mathbf{y}_2 = f_2(\mathbf{s}_2, \mathbf{x}_2) \wedge \dot{\mathbf{s}}_2 = \varphi_{d2}(\mathbf{x}_2, \mathbf{s}_2)\}$, respectively, $X_1, X_2 \subseteq \mathscr{R}^n$, $Y_1, Y_2 \subseteq \mathscr{R}^m$, $n, m \in \mathscr{N}^+$. The projection of Tr_1 into $\mathscr{X}_1$ is $Tr_1 \downarrow_{\mathscr{X}_1} = \{\mathbf{x}_1 | \exists \mathbf{y}_1 (\mathbf{x}_1, \mathbf{y}_1) \in Tr_1\}$. The projection of Tr_2 into $\mathscr{X}_2$ is the set $Tr_2 \downarrow_{\mathscr{X}_2} = \{\mathbf{x}_2 | \exists \mathbf{y}_2 (\mathbf{x}_2, \mathbf{y}_2) \in Tr_2\}$. Following Definition 7, we write $B_1 \| B_2 = (\mathbf{x}, \mathbf{y}, \mathbf{s}, f, \varphi)$. We denote the observable trajectories of $B_1 \| B_2$ by Tr. Then $Tr = \{(\mathbf{x}, \mathbf{y}) | \exists \mathbf{s} = (s_1, s_2) \mathbf{y}_1 = f_1(\mathbf{x}_1, \mathbf{s}_1) \wedge \dot{\mathbf{s}}_1 = \varphi_{d1}(\mathbf{x}_1, \mathbf{s}_1) \wedge \mathbf{y}_2 = f_2(\mathbf{x}_2, \mathbf{s}_2) \wedge \dot{\mathbf{s}}_2 = \varphi_{d2}(\mathbf{x}_2, \mathbf{s}_2)\}$. The projection of Tr into $\mathscr{X}$ is $Tr \downarrow_{\mathscr{X}} = \{\mathbf{x} | \exists \mathbf{y}_1 \mathbf{y}_2 (\mathbf{x}_1, \mathbf{y}_1) \in Tr_1 \wedge (\mathbf{x}_2, \mathbf{y}_2) \in Tr_2\}$, where $\mathscr{X} \subseteq (X_1 \times X_2)^T$.

Let $C_1 = (\mathbf{x}_1, \mathbf{y}_1, \phi_a^1, \phi_g^1)$ and $C_2 = (\mathbf{x}_2, \mathbf{y}_2, \phi_a^2, \phi_g^2)$ be contracts. By Definition 14, we write $C_1 \| C_2 = (\mathbf{x}, \mathbf{y}, \phi_a, \phi_g)$, where $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$, and

$$\begin{aligned} \phi_a &\coloneqq \{\mathbf{x} \in X^T | \mathbf{x}_1 \in \phi_a^1 \wedge \mathbf{x}_2 \in \phi_a^2\}, \\ \phi_g &\coloneqq \left\{(\mathbf{x}, \mathbf{y}) \in (X \times Y)^T | (\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge (\mathbf{x}_2, \mathbf{y}_2) \in \phi_g^2\right\}. \end{aligned} \tag{1}$$

□

To proof $B_1 \| B_2 \vDash C_1 \| C_2$, according to Definition 11, we only need to show that if $(Tr \downarrow_{\mathscr{X}}) \subseteq \phi_a$ implies $Tr \subseteq \phi_g$. We first prove that $Tr \downarrow_{\mathscr{X}} \subseteq \phi_a$. Since $B_1 \vDash C_1$, according to Definition 11, we have if $(Tr_1 \downarrow_{\mathscr{X}_1}) \subseteq \phi_a^1$ implies $Tr_1 \subseteq \phi_g^1$. That is for all $x_1 \in (Tr_1 \downarrow_{\mathscr{X}_1}) \in \phi_a^1$, for all $(x_1, y_1) \in Tr_1$, $(x_1, y_1) \in \phi_g^1$. Since $B_2 \vDash C_2$, we have $(Tr_2 \downarrow_{\mathscr{X}_2}) \subseteq \phi_a^2$ implies $Tr_2 \subseteq \phi_g^2$. That is, for all $x_2 \in (Tr_2 \downarrow_{\mathscr{X}_2})$, then $x_2 \in \phi_a^2$, for all $(x_2, y_2) \in Tr_2$, then $(x_2, y_2) \in \phi_g^2$. Hence, by (1), we have $\mathbf{x} = (x_1, x_2) \in \phi_a$, $(\mathbf{x}, \mathbf{y}) \in \phi_g$. Hence, $B_1 \| B_2 \vDash C_1 \| C_2$.

The parallel composition of contract is also associative and commutative.

Lemma 16 (Associativity, commutativity). *Let C_1, C_2, and C_3 be contracts. Then*

(i) $C_1 \| C_2 = C_2 \| C_1$

(ii) $(C_1 \| C_2) \| C_3 = C_1 \| (C_2 \| C_3)$

Proof. Immediately follows from the Definition 14. □

We now define the algebraic loop-free feedback composition of contracts.

Definition 17 (The algebraic loop-free feedback composition of contracts). Let $C_1 = (\mathbf{x}_1, \mathbf{y}_1, \phi_a^1, \phi_g^1)$ and $C_2 = (\mathbf{x}_2, \mathbf{y}_2, \phi_a^2,$

$\phi_g^2)$ be the contracts. The algebraic loop-free feedback composition of C_1 and C_2 be defined as $C_1 \otimes_f C_2 = (\mathbf{x}, \mathbf{y}, \phi_a, \phi_g)$,

$$\phi_a := \left\{ \mathbf{x}_{1,1} \in X^T \middle| \mathbf{x}_{1,1} \in \phi_a^1 \wedge \left((\exists y_1)(\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge \phi_{\rho_2} \longrightarrow y_1 \in \phi_a^2 \right) \right) \wedge \left((\exists \mathbf{y}_2)(\mathbf{x}_2, \mathbf{y}_2) \in \phi_g^2 \wedge \phi_{\rho_1} \longrightarrow \mathbf{y}_2 \in \phi_a^1 \right) \right) \right\},$$

$$\phi_g := \left\{ (\mathbf{x}_{1,1}, \mathbf{y}_1) \in (X \times Y)^T \middle| (\exists x_{1,2} \exists y_2 \exists x_2) \left((\mathbf{x}_1, \mathbf{y}_1) \in \phi_g^1 \wedge \phi_\rho \wedge (\mathbf{x}_2, \mathbf{y}_2) \in \phi_g^2 \right) \right\},$$

$$\phi_\rho := \phi_{\rho_1} \wedge \phi_{\rho_2}.$$

where $\phi_{\rho_2} := \wedge_{(y_{1,j}, x_{2,i}) \in \rho_2} y_{1,j}(t) = x_{2,i}(t)$, $\phi_{\rho_1} := \wedge_{(y_{2,j}, x_{1,i}) \in \rho_1} y_{2,j}(t) = x_{1,i}(t)$, and $\mathbf{x}_1 = (x_{1,1}, x_{1,2})$.

The algebraic loop-free feedback composition also preserves the satisfaction relation.

Theorem 18 (Algebraic loop-free feedback composition preserves satisfaction). *Let B_1 be the feedback with an algebraic loop and B_2 be a block with multistep delays. Let $C_1 = (\mathbf{x}_1, \mathbf{y}_1, \phi_a^1, \phi_g^1)$ and $C_2 = (\mathbf{x}_2, \mathbf{y}_2, \phi_a^2, \phi_g^2)$ be the contracts. If $B_1 \vDash C_1$ and $B_2 \vDash C_2$, then $B_1 \otimes_f B_2 \vDash C_1 \otimes_f C_2$.*

Proof. It easily follows from Theorem 13 to Theorem 15. □

We next turn to the refinement relation between the contract. We follow a standard notion inspired by [20].

Definition 19 (Refinement of contracts). Let $C_1 = (\mathbf{x}, \mathbf{y}, \phi_a^1, \phi_g^1)$ and $C_2 = (\mathbf{x}, \mathbf{y}, \phi_a^2, \phi_g^2)$ be two contracts. We say C_2 refines C_1, denoted by $C_2 \preccurlyeq C_1$, *if* $\phi_a^1 \subseteq \phi_a^2$ and $\phi_g^2 \subseteq \phi_g^1$.

Refinement relaxes assumptions and reinforces guarantees, therefore, strengthening the contract. Obviously, the following refinement rule holds.

Lemma 20. *Let $C_1 = (\mathbf{x}, \mathbf{y}, \phi_a^1, \phi_g^1)$ and $C_2 = (\mathbf{x}, \mathbf{y}, \phi_a^2, \phi_g^2)$ be two contracts. Then*

(i) $C_i \preccurlyeq C_i$, for $i = 1, 2$

(ii) If $\phi_a^1 \subseteq \phi_a^2$ and $\phi_g^2 = \phi_g^1$, then $C_2 \preccurlyeq C_1$

(iii) If $\phi_g^2 \subseteq \phi_g^1$ and $\phi_a^1 = \phi_a^2$, then $C_2 \preccurlyeq C_1$

The refinement implies that every Simulink block diagram satisfies the C_2 also satisfies the C_1. This gives us the following property of correctness.

where $\mathbf{x} = x_{1,1}, \mathbf{y} = y_1$, and

Theorem 21 (Correctness). *Let $B = (\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}, \varphi, f)$ be a block. Let $C_1 = (\boldsymbol{x}, \boldsymbol{y}, \phi_a^1, \phi_g^1)$ and $C_2 = (\boldsymbol{x}, \boldsymbol{y}, \phi_a^2, \phi_g^2)$ be contracts. The B is said to be correct for C_2, if $(B \vDash C_1 \wedge C_1 \preccurlyeq C_2) \Rightarrow B \vDash C_2$.*

Proof. Suppose $C_1 \preccurlyeq C_2$, in terms of Definition 19, then $\phi_a^2 \subseteq \phi_a^1$, and $\phi_g^1 \subseteq \phi_g^2$. Suppose $B \vDash C_1$, then, for all $x \in Tr\downarrow_{\mathcal{X}} \subseteq \phi_a^1$, and $Tr \subseteq \phi_g^1 \subseteq \phi_g^2$. Hence, if $x \in Tr\downarrow_{\mathcal{X}} \subseteq \phi_a^2$, then $B \vDash C_2$.

Theorem 21 is essential in our contract theory since it relates the Simulink block diagram (composition block) to a system contract specification. That is, if a Simulink block diagram satisfies the composition contract, and the composition contract refines the system contract, then it also satisfies the system contract. This ensures that the Simulink block diagram is correct.

6. Case Study

We have discussed the proof approach in the previous section, and in this section, the approach that we proposed is being implemented through a real-world case study.

6.1. Problem Statement.

6.1. Problem Statement. We examine a case, a safety-critical water level control system of reservoir, to illustrate how to verify that the controller model of a reservoir originating from the farmland irrigation satisfy the safety requirements.

This reservoir mainly is used for irrigation and considers the comprehensive utilization of flood control and aquaculture. An inlet water pipe and a spillway exist on the top and bottom of the dam, respectively. The maximum dam height is 40m, the maximum is 30m, and the minimum water level is 10m. The management facilities of the reservoir are very backward. There are no special management agencies to observe and safety check the water level and no timely flood control.

We will use Simulink to model the water level control system of the reservoir and simulate the water level trajectories to monitor the water level control operation. Then we adopt the method that we presented to verify the Simulink block diagram's correctness for ensuring the water level's safety according to the reservoir's actual situation. The water level controls the opening or closing of the valves. That is, the water level is neither higher than the highest water level,

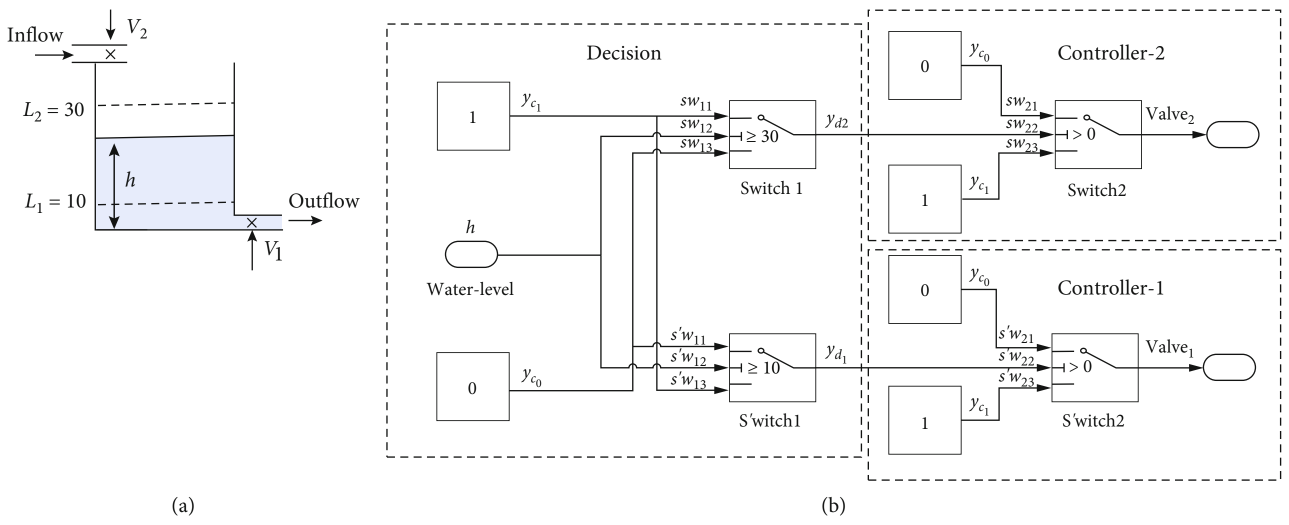


Figure 7: (a) An overview of reservoir system. (b) The model of controller.

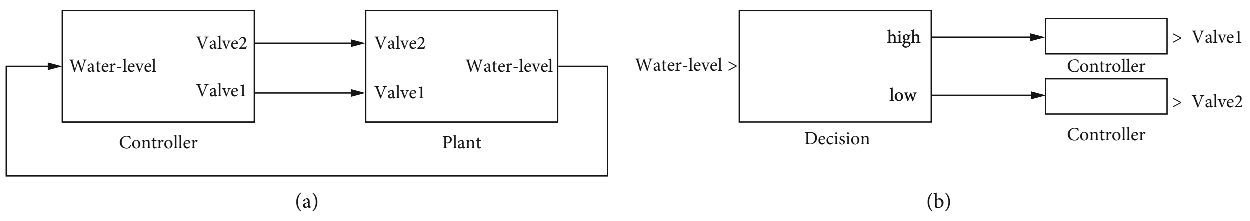


Figure 8: (a) A Simulink block diagram of reservoir system. (b) The block diagram of controller.

which causes the dam to be overloaded, nor can the aquaculture industry and farmland irrigation be affected by the water level being lower than the lowest water level.

6.2. Architectural Overview of Control System. The control system of water level consists of the following two major components: Sensor and Plant.

(i) *Sensor.* The *Sensor* is a water level sensor. Inputs about the water level of the reservoir from its corresponding level sensor, and we suppose that the *Sensor* works properly

(ii) *Plant.* The *Plant* is made up of four principal components, as shown in Figure 7(a): Reservoir, Valves, Controller, and Water inlet and outlet pipe

(a) *Reservoir.* The *Reservoir* has parameters giving the maximum water lever L_2, the minimum water lever L_1, and the water level of the reservoir $h \geq 0$

(b) *Water valves.* The *Reservoir* has two water valves. The inlet valve V_2 is at the top of the *Reservoir*, and the outlet valve V_1 is at the bottom. We suppose that the valves are either fully opened or fully closed immediately

(c) *Controller.* The *Controller* is able to read the current water level provided by the sensor, and the output is the command signal *open* or *closed*. The controller's target is to keep the water level between the minimum water lever L_1 and the maximum water lever L_2

(d) *Water inlet and outlet.* The inlet pipe $F_{\text{in}} \geq 0$ and $F_{\text{out}} \geq 0$, and $F_{\text{in}} \neq F_{\text{out}}$.

As the system is a closed-loop system, the controller must work with the reservoir. In Figure 8(a), a system including both controller and plant is given. We only focus on the controller.

6.3. The Safety Requirements of Water Level. The safety requirements for water level h in the controller are given below, and the purpose is to avoid the water empty and filling of reservoir.

Requirement 1 When the h of reservoir is above L_1, the V_1 is opened, and when the h of reservoir is below L_1, the V_1 is closed.

Requirement 2 When the h of reservoir is above L_2, the V_2 is closed, and when the h of reservoir is below L_2, the V_2 is opened.

Driven by requirement, we now refine system-level requirements into block-level implementations. When the system is refined, subsystem details are added. We first decompose the controller into three different controllers: *Decision* controller, *Controller*$_1$ controller, and *Controller*$_2$

controller. The block diagrams of these controllers can be depicted graphically as in Figure 8(b). In the block diagram, the *Decision* decides that the water level is high or low. The $Controller_1$ computes whether the $Valve_1$ should be open, while the $Controller_2$ computes if the $Valve_2$ should be open.

For the *Decision*, if the water level is above L_2, it is high. If the water level is below L_1, it is low. For the $Valve_2$, if the water level is high, then $Valve_2$ should be close(0). Otherwise, the $Valve_2$ should be open(1). For the $Valve_1$, if the water level is low, then $Valve_1$ should be close(0). Otherwise, then $Valve_1$ should be open(1).

We use the contract as a formal specification language. We suppose $L_1 = 10\text{m}$, $L_2 = 30\text{m}$, and $0\text{m} \le h(t) \le 40\text{m}$. The system-level safety requirement specification 1 is denoted as $C_{\text{req1}} = (\mathbf{h}, \mathbf{y}_{\mathbf{v}_1}, \phi_a^{\text{req1}}, \phi_g^{\text{req1}})$.

$$\phi_a^{\text{req1}} \coloneqq \{\mathbf{h} \in \mathscr{R}^T | \forall t \in T, 0 \le h(t) \le 40\},$$
$$\phi_g^{\text{req1}} \coloneqq \left\{ \left(\mathbf{h}, \mathbf{y}_{\mathbf{v}_1}\right) \in (\mathscr{R} \times \mathscr{R})^T | \forall t \in T, \mathbf{y}_{\mathbf{v}_1} = \begin{cases} \text{open}(1), & h(t) \ge 10, \\ \text{close}(0), & 0 \le h(t) < 10. \end{cases} \right\}. \tag{2}$$

The system-level safety requirement specification 2 is denoted as $C_{\text{req2}} = (\mathbf{h}, \mathbf{y}_{\mathbf{v}_2}, \phi_a^{\text{req2}}, \phi_g^{\text{req2}})$, where

$$\phi_a^{\text{req2}} \coloneqq \{\mathbf{h} \in \mathscr{R}^T | \forall t \in T, 0 \le h(t) \le 40\},$$
$$\phi_g^{\text{req2}} \coloneqq \left\{ \left(\mathbf{h}, \mathbf{y}_{\mathbf{v}_2}\right) \in (\mathscr{R} \times \mathscr{R})^T | \forall t \in T, \mathbf{y}_{\mathbf{v}_2} = \begin{cases} \text{close}(0), & h(t) \ge 30, \\ \text{open}(1), & 0 \le h(t) < 30. \end{cases} \right\}. \tag{3}$$

6.4. Verification of Correctness for the Controller System. We construct a Simulink block diagram for the control system according to requirements shown in Figure 7(b). Following the previous technical route, the verification method applied in this case consists of two stages.

Stage 1: verifying the Simulink block diagram satisfies the corresponding composition of contract.

The *Decision* controller has two controllers: $Decision_1$ and $Decision_2$. The $Decision_2$ determines whether the water level is higher than the highest water level. The composition of Simulink block diagrams of $Decision_2$ is denoted as $\otimes B_{d_2} = (B_{c_1} \| B_{c_0})\, ;\rho_1\, ; B_{sw_1}$, where $\rho_1 = \{(y_{c_1}, sw_{11}), (y_{c_0}, sw_{13})\}$. We denote the h by sw_{12}, and then $\otimes B_{d_2} \coloneqq (sw_{12}, \mathbf{0}, y_{d_2}, -, f_{d_2}(sw_{12}(t), \mathbf{0}))$, where,

$$f_{d_2}(sw_{12}, 0) = \begin{cases} 1(\text{high}) & sw_{12}(t) \ge 30, \\ 0(\neg\text{high}) & sw_{12}(t) < 30. \end{cases} \tag{4}$$

Therefore, the observable trajectories of $\otimes B_{d_2}$ are denoted as $Tr_2 = \{(sw_{12}, y_{d_2}) \colon T \mapsto \mathscr{R} \times \mathscr{R} | \forall t \in T, y_{d_2}(t) = f_{d_2}(sw_{12}(t), \mathbf{0})\}$, where

$$y_{d_2}(t) = \begin{cases} 1(\text{high}) & sw_{12}(t) \ge 30, \\ 0(\neg\text{high}) & sw_{12}(t) < 30. \end{cases} \tag{5}$$

The main task in the next step is to apply the composition rules to calculate the composition contract and prove that the composition of blocks satisfies the composition of the corresponding contracts. We first specify a contract for each elementary block following Table 1. Then we apply the composition rules to calculate the composition contract and prove that the composition of blocks satisfies the composition contracts.

The composition contract of the $Decision_2$ is stated as $\otimes C_{d_2} = (sw_{12}, y_{d_2}, \phi_a^{d_2}, \phi_g^{d_2})$, where

$$\begin{aligned}
\phi_a^{d_2} &\coloneqq \left\{ sw_{12} \in \mathscr{R}^T | \forall t \in T, \left(\left(\exists \mathbf{y}_{B_1}\right) \colon \left(\mathbf{0}, \mathbf{y}_{B_1}\right) \in \phi_g^{B_1} \land \phi_{\rho_1} \longrightarrow \mathbf{y}_{B_1} \in \phi_a^{B_{sw_1}} \right) \right\}, \\
\phi_g^{d_2} &\coloneqq \left\{ \left(\mathbf{0}, y_{d_2}\right) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T, \left(\mathbf{0}, \mathbf{y}_{B_1}\right) \in \phi_g^{B_1} \land \phi_{\rho_1} \land \left(sw_{12}, y_{d_2}\right) \in \phi_g^{sw_1} \right\}, \\
\phi_{\rho_1} &\coloneqq \left(y_{c_1}(t) = sw_{11}(t)\right) \land \left(y_{c_0}(t) = sw_{13}(t)\right).
\end{aligned} \tag{6}$$

Since

$$\phi_g^{B_1} \coloneqq \left\{ \left(\mathbf{0}, \mathbf{y}_{\mathbf{B}_1}\right) \in \left(\mathscr{R}^2 \times \mathscr{R}^2\right)^T \middle| \forall t \in T, y_{c_1}(t) = 1 \land y_{c_0}(t) = 0 \right\},$$
$$\phi_g^{B_{sw_1}} \coloneqq \left\{ \left(sw_{12}, y_{d_2}\right) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T, y_{d_2}(t) = f_{sw_1}(sw_{12}(t), \mathbf{0}) \right\}, \tag{7}$$

where

$$y_{d_2}(t) = \begin{cases} sw_{11} & sw_{12} \ge 30, \\ sw_{13} & sw_{12} < 30. \end{cases} \tag{8}$$

We use quantifier elimination law to simplify the expression. Therefore, the composition contract $(\otimes C_{d_2})$ of the D

ecision$_2$ controller was written in the following form.

$$\phi_a^{d_2} := \{sw_{12} \in \mathscr{R}^T | \forall t \in T, sw_{12}(t) \in \mathscr{R}\},$$
$$\phi_g^{d_2} := \left\{ \left(sw_{12}, y_{d_2}\right) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T, y_{d_2}(t) = \begin{cases} \text{high}(1) & sw_{12}(t) \geq 30, \\ \neg\text{high}(0) & sw_{12}(t) < 30. \end{cases} \right\}. \tag{9}$$

Similarly, the *Decision*$_1$ determines whether the water level is below the lowest water level. The composition of Simulink block diagrams of *Decision*$_1$ is denoted as $\otimes B_{d_1} = (B_{c_1} \| B_{c_0})\,;\rho_2\,;B_{s'w_1}$, where $\rho_2 = \{(y_{c_0}, s'w_{11}), (y_{c_1}, s'w_{13})\}$. Let $B_1 = B_{c_1} \| B_{c_0}$ and $\mathbf{y}_{B1} = (y_{c_0}, y_{c_1})$. We denote the Simulink block diagrams of *Decision*$_1$ by $\otimes B_{d_1} := (s'w_{12}, \mathbf{0}, y_{d_1}, -, f_{d_1}(s'w_{12}(t), \mathbf{0}))$, where

$$f_{d_1}\left(s'w_{12}(t), \mathbf{0}\right) = \begin{cases} 0(\neg\text{low}) & s'w_{12}(t) \geq 10, \\ 1(\text{low}) & s'w_{12}(t) < 10. \end{cases} \tag{10}$$

The observable trajectories of $\otimes B_{d_1}$ is denoted as $Tr_1 = \{(s'w_{12}, y_{d_1}): T \mapsto \mathscr{R} \times \mathscr{R} | \forall t \in T, y_{d_1}(t) = f_{d_1} s'w_{12}(t), \mathbf{0})\}$, where

$$y_{d_1}(t) = \begin{cases} 0(\neg\text{low}) & s'w_{12}(t) \geq 10, \\ 1(\text{low}) & s'w_{12}(t) < 10. \end{cases} \tag{11}$$

The composition contract of the *Decision*$_1$ controller for $\otimes B_{d_1}$ is stated as $\otimes C_{d_1} = (s'w_{12}, y_{d_1}, \phi_a^{d_1}, \phi_g^{d_1})$, where

$$\phi_a^{d_1} := \left\{s'w_{12} \in \mathscr{R}^T \middle| \forall t \in T, s'w_{12}(t) \in \mathscr{R}\right\},$$
$$\phi_g^{d_1} := \left\{ \left(s'w_{12}, y_{d_1}\right) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T, [l]@l@y_{d_1}(t) = \begin{pmatrix} \neg\text{low}(0) & s'w_{12}(t) \geq 10, \\ \text{low}(1) & s'w_{12}(t) < 10. \end{pmatrix} \right\}. \tag{12}$$

Since variables sw_{12} and $s'w_{12}$ are used to read the values of water level, so $sw_{12}(t) = h(t)$ and $s'w_{12}(t) = h(t)$. According to Definition 11, $\forall t \in T$, we have $\{sw_{12} \in \mathscr{R}^T | sw_{12}(t) \geq 0\} \subseteq \phi_a^{d_2}$, $Tr_2 \subseteq \phi_g^{d_2}$, and $\{s'w_{12} \in \mathscr{R}^T | s'w_{12}(t) \geq 0\} \subseteq \phi_a^{d_1}$, $Tr_1 \subseteq \phi_g^{d_1}$. Hence, $\otimes B_{d_2} \vDash \otimes C_{d_2}$ and $\otimes B_{d_1} \vDash \otimes C_{d_1}$.

We next turn to research the controller *Controller*$_1$. It can be represented as a composition block $\otimes B_{v_1} = ((B_{c_1} \| B_{c_0})\,;\rho_3\,;B_{s'w_2})$, where $\rho_3 = \{(y_{c_1}, s'w_{23}), (y_{c_0}, s'w_{21})\}$. Let $\otimes B_{v_1} = (s'w_{22}, \mathbf{0}, y_{v_1}, -, y_{v_1}(t) = f_{v_1}(s'w_{22}(t), \mathbf{0}))$, where

$$f_{v_1}\left(s'w_{22}(t), \mathbf{0}\right) = \begin{cases} 0(\text{close}) & s'w_{22}(t) > 0, \\ 1(\text{open}) & s'w_{22}(t) \leq 0. \end{cases} \tag{13}$$

Therefore, the observable trajectories of $\otimes B_{v_1}$ is denoted as $Tr_3 = \{(s'w_{22}, y_{v_1}): T \mapsto \mathscr{R} \times \mathscr{R} | \forall t \in T, y_{v_1}(t) = f_{v_1}(s'w_{22}(t), \mathbf{0})\}$, where

$$y_{v_1}(t) = \begin{cases} 0(\text{close}) & s'w_{22}(t) > 0, \\ 1(\text{open}) & s'w_{22}(t) \leq 0. \end{cases} \tag{14}$$

The composition contract of *Controller*$_1$ is stated as $\otimes C_{v_1} = (s'w_{22}, y_{v_1}, \phi_a^{v_1}, \phi_g^{v_1})$, where

$$\phi_a^{v_1} := \left\{s'w_{22} \in \mathscr{R}^T \middle| \forall t \in T, s'w_{22}(t) \in \mathscr{R}\right\},$$
$$\phi_g^{v_1} := \left\{ \left(s'w_{22}, y_{v_1}\right) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T', y_{v_1}\left(s'w_{22}(t), \mathbf{0}\right) = \begin{cases} 0(\text{close}) & s'w_{22}(t) > 0, \\ 1(\text{open}) & s'w_{22}(t) \leq 0. \end{cases} \right\}. \tag{15}$$

According to Definition 11, $\forall t \in T$, we have $\{s'w_{22} : T \mapsto \mathscr{R} | s'w_{22}(t) \in \mathscr{R}\} \subseteq \phi_a^{v_1}$, and $Tr_3 \subseteq \phi_g^{v_1}$. Hence, $\otimes B_{v_1} \vDash \otimes C_{v_1}$.

For requirement 1 Next, we will compose the *Decision*$_1$ controller and *Controller*$_1$ controller for verifying requirement 1. We denote the composition of *Decision*$_1$ and *Controller*$_1$ by $\otimes B_{\text{Im }1}$. We denote the h by $s'w_{12}$. We write $\otimes B_{\text{Im }1} = \otimes B_{d_1}\,;\otimes B_{v_1} = (s'w_{12}, \mathbf{0}, y_{v_1}, -, y_{v_1} = f_{\text{Im }1}(s'w_{12}(t), \mathbf{0}))$, where

$$f_{\text{Im }1}\left(\mathbf{s'w_{12}}(t), \mathbf{0}\right) = \begin{cases} 1(\text{open}) & s'w_{12}(t) \geq 10, \\ 0(\text{close}) & s'w_{12}(t) < 10. \end{cases} \tag{16}$$

The observable trajectories of $\otimes B_{\text{Im }1}$ is denoted as $Tr_{\text{Im }1} = \{(s'w_{12}, y_{v_2}): T \mapsto \mathscr{R} \times \mathscr{R} | \forall t \in T, y_{\text{Im }1}(t) = f_{\text{Im }1}(s'w_{12}(t), \mathbf{0}))\}$, where

$$y_{\text{Im }1}(t) = \begin{cases} 1(\text{open}) & s'w_{12}(t) \geq 10, \\ 0(\text{close}) & s'w_{12}(t) < 10. \end{cases} \tag{17}$$

The composition contract of $\otimes B_{\text{Im }1}$ is defined as $\otimes C_{\text{Im }1} = (s'w_{12}, y_{v_1}, \phi_a^{\text{Im }1}, \phi_g^{\text{Im }1})$, and

$$\phi_a^{\text{Im }1} := \left\{s'w_{12} \in \mathscr{R}^T \middle| \forall t \in T, s'w_{12}(t) \in \mathscr{R}\right\},$$
$$\phi_g^{\text{Im }1} := \left\{ \left(s'w_{12}, y_{v_1}\right) \in (\mathscr{R} \times \mathscr{R})^T \middle| \forall t \in T, y_{v_1}(t) = \begin{cases} 1(\text{open}) & s'w_{12}(t) \geq 10, \\ 0(\text{close}) & s'w_{12}(t) < 10. \end{cases} \right\}. \tag{18}$$

$\forall t \in T$, we have $\{s'w_{12}(t): T \mapsto \mathscr{R} | s'w_{12}(t) \in \mathscr{R}\} \subseteq \phi_a^{\text{Im }1}$, and $Tr_{\text{Im }1} \subseteq \phi_g^{\text{Im }1}$. Hence, $B_{\text{Im }1} \vDash C_{\text{Im }1}$.

Stage 2: correctness verification. Our primary goal in this step is to verify that the Simulink block diagram ($\otimes B_{v_1}$) is correct for system contract C_{req1}. In terms of Definition 19, we get $\phi_a^{\text{req1}} \subseteq \phi_a^{\text{Im 1}}$ and $\phi_g^{\text{Im 1}} \subseteq \phi_g^{req1}$, then, $C_{\text{Im 1}} \preccurlyeq C_{\text{req1}}$. According to Theorem 21, we have $B_{\text{Im 1}} \vDash C_{\text{Im 1}} \wedge C_{\text{Im 1}} \preccurlyeq C_{\text{req1}} \Rightarrow B_{\text{Im 1}} \vDash C_{\text{req1}}$.

For requirement 2

We next turn to verify the correctness of the Simulink block diagrams for requirement 2. Similar to the method above, the *Controller*$_2$ controller can be represented as a composition block $\otimes B_{v_2} = (B_{c_1} \| B_{c_0}) \,;\, \rho_4 \,;\, B_{sw_2}$, where $\rho_4 = \{(y_{c_1}, sw_{23}), (y_{c_0}, sw_{21})\}$. Then $\otimes B_{v_2} = (sw_{22}, \mathbf{0}, y_{v_2}, -, y_{v_2}(t) = f_{v_2}(sw_{22}(t), \mathbf{0}))$, where

$$y_{v_2}(t) = \begin{cases} 0(\text{close}), & sw_{22}(t) > 0, \\ 1(\text{open}), & sw_{22}(t) \leq 0. \end{cases} \tag{19}$$

Therefore, the observable trajectories of the subsystem $\otimes B_{v_2}$ is denoted as $Tr_4 = \{(sw_{22}, y_{v_2}): T \mapsto \mathscr{R} \times \mathscr{R} | \forall t \in T, y_{v_2}(t) = f_{v_2}(sw_{22}(t), \mathbf{0})\}$, where

$$y_{v_2}(t) = \begin{cases} 0(\text{close}), & sw_{22}(t) > 0, \\ 1(\text{open}), & sw_{22}(t) \leq 0. \end{cases} \tag{20}$$

Stage 1: verifying the Simulink block diagram satisfies the corresponding composition of contract.

The composition contract of *Controller*$_2$ is defined as $\otimes C_{v_2} = (sw_{22}, y_{v_2}, \phi_a^{v_2}, \phi_g^{v_2})$, where

$$\phi_a^{v_2} \coloneqq \{sw_{22} \in \mathscr{R}^T | \forall t \in T, sw_{22}(t) \in \mathscr{R}\},$$
$$\phi_g^{v_2} \coloneqq \left\{ \left(sw_{22}, \mathbf{y}_{\mathbf{v_2}}\right) \in (\mathscr{R} \times \mathscr{R})^T | \forall t \in T, y_{v_2}(t) = \begin{cases} 0(\text{close}), & sw_{22}(t) > 0, \\ 1(\text{open}), & sw_{22}(t) \leq 0. \end{cases} \right\}. \tag{21}$$

According to Definition 11, $\forall t \in T$, we have $\{sw_{22}(t) \in \mathscr{R}^T | sw_{22}(t) \in \mathscr{R}\} \subseteq \phi_a^{v_2}$, and $Tr_4 \subseteq \phi_g^{v_2}$. Hence, $\otimes B_{v_2} \vDash \otimes C_{v_2}$.

Next, we will compose the *Decision*$_2$ and *Controller*$_2$ to verify requirement 2. Let $\otimes B_{\text{Im 2}} = \otimes B_{d_2} \,;\, \otimes B_{v_2} = (sw_{12}, \mathbf{0}, y_{v_2}, -, y_{v_2} = f_{v_2}(sw_{12}(t), \mathbf{0}))$, where

$$f_{\text{Im 2}}(\mathbf{sw_{12}}(t), \mathbf{0}) = \begin{cases} 0(\text{close}), & sw_{12}(t) \geq 30, \\ 1(\text{open}), & sw_{12}(t) > 30. \end{cases} \tag{22}$$

The observable trajectories of $\otimes B_{\text{Im 2}}$ are denoted as $Tr_{\text{Im 2}} = \{(sw_{12}, y_{v_2}): T \mapsto \mathscr{R} \times \mathscr{R} | \forall t \in T, y_{\text{Im 2}}(t) = f_{\text{Im 2}} sw_{12}(t), \mathbf{0})\}$, where

$$y_{\text{Im 2}}(t) = \begin{cases} 0(\text{close}), & sw_{12}(t) \geq 30, \\ 1(\text{open}), & sw_{12}(t) < 30. \end{cases} \tag{23}$$

The composition contract is defined as $\otimes C_{\text{Im 2}} = (sw_{22}, y_{v_2}, \phi_a^{\text{Im 2}}, \phi_g^{\text{Im 2}})$, and

$$\phi_a^{\text{Im 2}} \coloneqq \{sw_{12} \in \mathscr{R}^T | \forall t \in T, sw_{12}(t) \in \mathscr{R}\},$$
$$\phi_g^{\text{Im 2}} \coloneqq \left\{ \left(sw_{12}, \mathbf{y}_{\mathbf{v_2}}\right) \in (\mathscr{R} \times \mathscr{R})^T | \forall t \in T', y_{v_2}(sw_{12}(t), \mathbf{0}) \right.$$
$$= \left. \begin{cases} 0(\text{close}), & sw_{12}(t) \geq 30, \\ 1(\text{open}), & sw_{12}(t) < 30. \end{cases} \right\}. \tag{24}$$

Stage 2: correctness verification. Our main goal in this step is to verify that the Simulink block diagram ($\otimes B_{v_2}$) is a correct implementation of system contract C_{req2}. $\forall t \in T$, we have $\{sw_{12}(t): T \mapsto \mathscr{R} | sw_{12}(t) \in \mathscr{R}\} \subseteq \phi_a^{\text{Im 2}}$, and $Tr_{\text{Im 2}} \subseteq \phi_g^{\text{Im 2}}$. Hence, $B_{\text{Im 2}} \vDash C_{\text{Im 2}}$. In terms of Definition 19, we get $\phi_a^{req2} \subseteq \phi_a^{\text{Im 2}}$, $\phi_g^{\text{Im 2}} \subseteq \phi_g^{req2}$. Then, $C_{\text{Im 2}} \preccurlyeq C_{req2}$. According to Theorem 21, we have $B_{\text{Im 2}} \vDash C_{\text{Im 2}} \wedge C_{\text{Im 2}} \preccurlyeq C_{req2} \Rightarrow B_{\text{Im 2}} \vDash C_{req2}$.

In the example above, when an attacker adds a malicious logic bomb to Simulink, it can maliciously manipulate the input and output behaviors of any block in the block diagrams and the water level values read from the Sensor. This may lead to the block diagram's behavior not satisfying the system's formal specification. Considering from this perspective, our approach can identify and verify whether the designed CPS is planted with the logic bomb.

7. Conclusion and Future Work

In this paper, we presented a method to prove the correctness of Simulink block diagrams with *discrete-time* or *continuous-time* blocks using contract. This approach addressed the problem of proving that the system formal specifications are satisfied by composition and refinement of trajectories. We showed the usability of our proposals via a use case (as an example) which models the control system of a reservoir. Our method can improve the reliability of Simulink and reduce the development costs by performing early safety verification on verification of the target system.

In this work, we made the first step towards the contract-based verification of cyber-physical models in Simulink. Future work includes extending the work along several dimensions. First, this work considers single rate Simulink block diagrams. We will extend this work to consider the multiple sample times (multirate systems) diagrams. Another nontrivial extension involves applying our idea to prove the correctness of Simulink block diagrams mixture of *discrete-time* and *continuous-time* blocks. More challenging would be to develop a contract-based refinement approach to handle Stateflow. Second, this work only considers the blocks whose inputs-outputs can be explicitly represented through the mathematical relation. It is interesting to prove the correctness of the Simulink block diagrams, whose input-output behavior be expressed implicitly by a mathematical relation. Third, we plan to automatically verify the correctness of the Simulink block diagrams based on our method.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

Acknowledgments

This work was funded in part by the Chongqing Graduate Research and Innovation Project (grant No. CYB20098), the National Natural Science Foundation of China (Nos. 62032019, 61732019, 61672435, and 62002298), and the Capacity Development Grant of Southwest University (SWU116007). We are grateful for the valuable comments from Prof. Xinxin Liu and Prof. Zheng Yang on the earlier versions of this paper. We thank the anonymous referees whose suggestions improved the presentation.

References

[1] E. A. Lee, "Cyber physical systems: design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 363–369, Orlando, FL, USA, 2008.

[2] G. Nicolescu and P. J. Mosterman, *Model-Based Design for Embedded Systems*, Crc Press, 2010.

[3] Mathworks, "Design cyber-physical systems with MATLAB and Simulink," *Mathworks*, 2022, September 2021, https://www.mathworks.com/discovery/cyber-physical-systems.html.

[4] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security*, pp. 110–126, Springer International Publishing, 2018.

[5] S. Tripakis, C. Sofronis, P. Caspi, and A. Curic, "Translating discrete-time Simulink to Lustre," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 4, no. 4, pp. 779–818, 2005.

[6] A. Cavalcanti, P. Clayton, and C. O'Halloran, "Control law diagrams in Circus," in *FM 2005: Formal Methods, International Symposium of Formal Methods Europe, Newcastle, UK, July 18-22, 2005*, vol. 3582 of Proceedings, ser. Lecture notes in computer science, pp. 253–268, Springer, Berlin, Heidelberg, 2005.

[7] M. Bostro, L. Morel, and M. Wald'en, "Stepwise development of Simulink models using the refinement calculus framework," in *International Colloquium on Theoretical Aspects of Computing*, C. B. Jones, Z. M. Liu, and J. Woodcock, Eds., vol. 4711 of Lecture Notes in Computer Science, pp. 79–93, Springer, Berlin, Heidelberg, 2007.

[8] S. K. Muduli, "Contract based development and refinement in Simulink [MS Thesis]," International Institute of Information Technology, Bangalore, 2017.

[9] M. Bostro, "Contract-based verification of Simulink models," in *Formal Methods and Software Engineering -13th International Conference on Formal Engineering Methods, ICFEM 2011*, S. C. Qin and Z. Y. Qiu, Eds., vol. 6991 of Lecture Notes in Computer Science, pp. 291–306, Springer, Berlin, Heidelberg, 2011.

[10] K. Ye, S. Foster, and J. Woodcock, "Compositional assume-guarantee reasoning of control law diagrams using UTP," in *From Astrophysics to Unconventional Computation*, pp. 215–254, Springer, Cham, 2020.

[11] J. W. K. Ye and S. Foster, "Compositional assume-guarantee reasoning of control law diagrams using UTP," 2018, April 2022, https://eprints.whiterose.ac.uk/129640/15/Compositional.

[12] C. Zhou and R. Kumar, "Semantic translation of Simulink diagrams to input/output extended finite automata," *Discrete Event Dynamic Systems*, vol. 22, no. 2, pp. 223–247, 2012.

[13] I. Dragomir, V. Preoteasa, and S. Tripakis, "The refinement calculus of reactive systems toolset," *International Journal on Software Tools for Technology Transfer*, vol. 22, no. 6, pp. 689–708, 2020.

[14] V. Preoteasa, I. Dragomir, and S. Tripakis, "Compositional semantics and analysis of hierarchical block diagrams," in *International Symposium on Model Checking Software*, vol. 9641 of Lecture notes in computer science, pp. 38–56, Springer, Cham, 2016.

[15] C. Chen and J. S. Dong, "Applying timed interval calculus to Simulink diagrams," in *Formal Methods and Software Engineering, 8th International Conference on Formal Engineering Methods, ICFEM*, Z. Liu and J. He, Eds., vol. 4260 of Lecture Notes in Computer Science, pp. 74–93, Springer, Berlin, Heidelberg, 2006.

[16] C. Q. Chen, J. S. Dong, and J. Sun, "A formal framework for modelling and validating Simulink diagrams," *Formal Aspects of Computing*, vol. 21, no. 5, pp. 451–483, 2009.

[17] O. Bouissou and A. Chapoutot, "An operational semantics for Simulink's simulation engine," in *SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2012, LCTES '12*, pp. 129–138, Beijing, China, 2012.

[18] L. Zou, N. J. Zhan, S. L. Wang, F. Martin, and S. C. Qin, "Verifying Simulink diagrams via a hybrid hoare logic prover," in *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)*, pp. 1–10, Montreal, QC, Canada, 2013.

[19] T. Bourke, F. Carcenac, B. Pagano, C. Pasteur, and M. Pouzet, "A synchronous look at the Simulink standard library," vol. 16, no. 5, pp. 176:1–176:24, 2017.

[20] A. Benveniste, B. Caillaud, D. Nickovic et al., "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.