scientific reports

Check for updates

OPEN

Toward a modeling and analysis method of cyber-physical systems architecture evolution based on bigraph

Chaoze Lu[1]✉, Qifeng Zou[2]✉ & Jianchao Zhou[1]

The evolution of cyber-physical systems (CPS) is inevitable. Traditional graph and hypergraph modeling and analysis methods can only describe one-dimensional evolutionary information, making it difficult to directly apply them to the modeling and analysis of CPS evolution processes that involve two-dimensional space. To address this issue, this paper proposes a Bigraph model for CPS that incorporates positional constraints. This model adopts a divide-and-conquer strategy, utilizing the link graph and place graph of Bigraph to represent the two-dimensional relationships of connectivity and positional relationships among entities within the CPS, respectively. Based on this model, a set of dynamic evolution rules for CPS architecture is designed. Furthermore, by leveraging the concepts of conditional matching and state transition, a model for the dynamic evolution of CPS structure and information flow evolution is proposed. Algorithms for checking consistency, integrity, and reachability constraints during the dynamic evolution of CPS architecture are developed around this model. These algorithms ensure the correctness and reliability of the CPS system after its dynamic evolution. Finally, experiments are conducted using the evolution of a smart meeting system and a vehicular networking system as case studies, validating the effectiveness of the proposed model and algorithms.

Keywords Cyber-physical systems evolution, Architecture evolution modeling, Evolution reliability, Bigraph model, Software architecture

In recent years, embedded computing, wireless communication, sensor monitoring and large-scale data processing technologies have developed rapidly, which makes the computing process, communication process and physical process highly integrated. Computing ability, communication ability and perception ability are constantly integrated into the physical process, which makes the physical equipment be information and networked. This process forms a hybrid autonomous system, which is called Cyber-Physical System (CPS)[1–3]. At present, cyber-physical system is closely related to human life and social development, and has been applied to various fields, such as urban traffic flow forecasting[4], smart grid control[5], medical health management[6], smart manufacturing[7,] etc. Although they focus on different design goals and research contents, they essentially integrate and process the resources of physical space and information space through the network. It makes people's life more convenient and comfortable.

From the system point of view, CPS is a large-scale multi-dimensional complex system based on physics. It has the following basic characteristics: scale diversity, space-time, distribution, dynamic reconfiguration, high autonomy, high reliability, strong security, etc[8,9]. In order to understand and explore these characteristics of this complex system, a highly abstract description method is usually needed to describe the system itself. As a highly abstract top-level design method, software architecture has been widely used. It is also suitable for the establishment of CPS model, so as to avoid getting caught up in tedious details too early, which makes it difficult to analyze the essential characteristics of the above system. At present, many scholars and researchers have explored CPS from the perspective of software architecture. For example, Ivanov et al.[10] proposed an architecture for Cyber-Physical Systems (CPS) suitable for medical applications, which meets the requirements of rapid interrupt response and task switching. Vogel et al.[11] have proposed a software architecture definition for Cyber-Physical Production Systems (CPPS) that aims to satisfy characteristics such as flexibility, maintainability, and extensibility. Dotty et al.[12] have proposed a software architecture that can meet the requirements of intrusion

[1]School of Cyber Science and Engineering, Ningbo University of Technology, Ningbo 315211, Zhejiang, China. [2]School of Humanities, Ningbo University of Finance and Economics, Ningbo 315175, Zhejiang, China. ✉email: luchaoze@126.com; zouqifeng425@126.com

detection and protection for CPS, thereby enhancing the security of CPS systems, and so on. Thus, software architecture is indispensable in CPS system. Excellent software architecture can not only make the design of CPS easier to meet user requirements, but also make CPS easier to extend, modify and maintain in the later period. This is one of the motives of this study. However, CPS does not remain unchanged after running. Faced with the change of user requirements and environment, CPS also needs to be changed accordingly. In addition, compared with the traditional information system, the CPS has more vigorous user requirements and more complex environment, so the CPS changes quite frequently. From the point of view of software architecture, the change of system is naturally reflected in the change of software architecture. The change of software architecture has always been the focus of attention in the field of software engineering. How to make the software architecture change correctly according to the change of requirement and environment is the second motive of this study.

Motivated by the software architecture of the cyber-physical system, the core concern objects of cyber-physical fusion are abstracted, adhering to the top-level design principle. Building on this foundation, the dynamic evolution process of CPS is explored, leading to the presentation of the following main contributions:

(1) A unified mathematical model of a Bigraph for Cyber-Physical Systems (CPS) incorporating positional constraints is proposed. This model adopts a divide-and-conquer strategy, utilizing the link graph and the place graph of the Bigraph to represent the connectivity and positional relationships among various entities within the CPS, respectively.
(2) A set of dynamic evolution rules for the architecture of CPS is designed. Based on these rules, a model for the dynamic evolution of CPS structure and information flow evolution is proposed through the idea of conditional matching and state transition.
(3) Checking algorithms for consistency, integrity, and reachability constraints in the dynamic evolution of CPS architecture are designed based on the hypergraph characteristics of the link graph and the nested relationships of the place graph. These algorithms ensure the correctness and reliability of the CPS system after its dynamic evolution.

The rest of this paper is organized as follows: "Related work" section reviews the related work which includes traditional software system evolution and cyber-physical system modeling and evolution. "The basic concepts of CPS and Bigraph" section introductions the basic framework of CPS and Bigraph. "Architecture of cyber-physical systems" section details static and dynamic model of CPS. "Constraint analysis and checking algorithm design of architecture" section discusses constraint analysis and checking algorithm design of architecture. "Instance and experimental analysis" section carried out an instance and experiment to validate the proposed models and methods. "Conclusion" section gives the conclusion and the future directions of research.

Related work

Traditional software system evolution

Software system evolution is one of the research hotspots in the field of software engineering that has been continuously concerned. In recent years, it has been studied by many researchers from different perspectives and achieved a series of results, mainly focusing on three aspects: evolution rule description, software architecture evolution modeling, and evolution platform development.

In terms of evolution rule description, for instances, Ferchichi et al.[13] analyzed the evolution process of software product line (SPL), proposed to use ontology to manage software evolution knowledge method, thus giving an ontology-based evolution rule description. Chaturvedi et al.[14] analyzed system evolution through network graph model, and used subgraph transformation semantics to describe evolution rules, and implemented the corresponding prototype tool to verify the use of evolution rules. Berrio et al.[15] used architecture description language to model service-oriented software architecture models, and used ADL to understand the evolution rules of components and connectors in software architecture. Chaturvedi et al.[16] designed three types of labeled evolution rules for insertion, deletion, and modification within evolving cloud service systems. They conducted experiments on two cloud service systems to demonstrate the feasibility of these rules. Chaturvedi et al.[17] also proposed methods for mining stable network evolution rules and metrics for variability within evolving systems over time, etc.

In terms of modeling software architecture evolution, for instances, Haitzer et al.[18] used UML to model software architecture and discussed the consistency of software architecture evolution with source code evolution. Ivers et al.[19] designed a search-based approach to software architecture refactoring evolution using optimization theory to guide the optimal model for refactoring evolution. Zhong et al.[20] constructed an inverse model of software architecture using binary tree as a way to support the construction of software evolution. Chaturvedi et al.[21] have proposed an Evolution and Change Learning (ECL) method, which combines evolution with neural networks to develop a system structure learning algorithm, making it possible for intelligent evolutionary analysis, etc.

In terms of evolution platform development, for instances, Ali et al.[22] develop a Repast software dynamic evolution simulation environment, which is a dynamic evolution simulation model for agents that better reflects the real world of software evolution compared to the traditional system dynamic evolution simulation model SD. Cui et al.[23] designed a dynamic wearable computer software platform based on the cloud computing platform and developed a corresponding graphical management interface for wearable computers. Chaturvedi[24] generates evolution information of call graphs among various versions of a software system within a graph-based environment, thereby supporting software evolution analysis and facilitating the maintenance or upgrade process of the software system, etc.

Although the current field of software system evolution has achieved good research results in these areas, the special nature of the physical properties of cyber-physical systems makes it impossible to apply them directly to the evolution of such systems, and new evolutionary techniques are needed to support them.

Cyber-physical system modeling and evolution

Cyber-physical systems are an extension of traditional software systems to the physical world, which are different from the construction and composition of traditional software systems. The integration of information space and physical space makes the evolution of such systems more complex. Traditional software system evolution methods mainly focus on the modeling and analysis of information space evolution, which are not suitable for cyber-physical system evolution modeling and analysis. This is because they lack the capability of describing the physical space, that is, they do not have the ability to describe both the information space and the physical space at the same time. The modeling of cyber-physical systems is challenging due to their dual-space attribute. However, in recent years, many researchers have explored in two aspects: cyber-physical system modeling and cyber-physical system evolution analysis.

Cyber-physical system modeling aims to model the structure or behavior of the cyber-physical system, without considering the dynamics of its structural or behavioral changes. It employs formal modeling methods such as AADL, UML, Petri net, multi-paradigm, etc. For instance, Zhang et al.[25] developed an extended AADL based on Modelica and cellular automata to model cyber-physical systems, and demonstrated the effectiveness of this modeling approach with concrete examples. Ionita et al.[26] analyzed how to use UML for modeling Cyber-Physical Systems (CPS) and discussed UML along with two of its parts particularly relevant in the context of CPS applications: SysML and MARTE. Grobelna et al.[27] utilized interpreted Petri nets to describe cyber-physical systems and characterize the behavior of concurrent processes within these systems. Morozov et al.[28] applied a multi-paradigm modeling MPM method for modeling cyber-physical systems, thereby improving the interoperability of the system, etc.

The evolution analysis of cyber-physical systems mainly involves modeling and analyzing the changes in system structure and behavior. Some researchers have also begun to explore this topic in recent years. For instance, Lehnert et al.[29] proposed a method for describing automation software variants in Cyber-Physical Production Systems (CPPS) based on a Domain-Specific Language (DSL). This method models the evolution of system software by utilizing multiple levels of abstraction. Haubeck et al.[30] focused on the evolution of cyber-physical production systems CPPS, and emphasized the capability of machine evolution awareness. They introduced a concept of machine evolution community to support the identification and propagation of different evolution use cases. Yu et al.[31] adopted generalized stochastic Petri nets to present a dynamic model of malware system propagation. Based on this model, they developed a trustworthy evolution model of cyber-physical manufacturing system CPMS, and demonstrated the validity of their approach through simulation experiments. Dong et al.[32] developed a hierarchical evolution model of spatial power grid, and examined the fragility of the structure under global and local failures, etc. Moreover, some researchers utilized dynamic update techniques to the evolution update of cyber-physical systems. For instance, Gartziandia et al.[33] addressed the challenge of detecting runtime performance issues when deploying new software versions for microservices, and suggested a machine learning-based algorithm that forecasts the performance of new versions based on the knowledge of previous versions, to guarantee the reliability of cyber-physical system updates. Houssem et al.[34] present an evolutionary process for cyber-physical system product lines that leverages contract-based design (CBD) to enable reuse of components and incremental analysis and verification of their modules. Stadler et al.[35] introduce a runtime model GRuM that supports updating and adapting cyber-physical systems as they evolve. Vierhauser et al.[36] propose a novel framework for runtime monitoring of cyber-physical systems that combines model-driven and runtime monitoring techniques to enhance maintenance efficiency by tracking system evolution. Zimmermann et al.[37] proposed a graph-based Cyber-Physical System architecture that can improve change impact analysis and track the interrelationships between system elements and their digital representations, etc.

From the perspective of formal modeling methods, AADL, UML, Petri nets, multi-paradigm, graphs, hypergraphs, and other formal modeling methods currently focus on modeling the single information space of cyber-physical systems, and these methods only have the ability to model in a single dimension. However, the physical space of cyber-physical systems is also an extremely important part, and the loss of modeling for this part of the evolution makes it difficult to effectively analyze the evolution of the entire system. From the perspective of structural and behavioral evolutionary modeling analysis methods, these methods also tend to model the structure and behavior of information space, without simultaneously conducting structural and behavioral modeling analysis research in both information space and physical space. In other words, these studies have only explored one aspect of evolution, and have not provided a systematic and in-depth approach to modeling and analysis from both perspectives. This is the motivation behind this study. The proposed Bigraph method can tackle the challenge of concurrently modeling both the information space and the physical space.

The basic concepts of CPS and bigraph

The basic framework of CPS

Cyber-physical systems highlight the interaction between the physical world and the information world, as shown in Fig. 1. The system consists of a sensor network that can perceive the physical environment, an information world platform that processes the physical attribute information obtained by the sensor network, a controller network that obtains the relevant control information after computation, and an actuator network that receives the control information and performs various actions to change the related attributes of the physical world. Through this iterative process, the system meets the desired needs of people. Thus, cyber-physical systems emphasize the fusion of physical and information, enabling the integration and coordination of the physical world and the virtual world, and creating the next generation of intelligent systems that can autonomously sense, predict, plan and adjust. Traditional software systems are mainly described by means such as ADL, process algebra, graphs, Petri nets, etc. However, they only account for the information world and neglect the physical world, making them inadequate for describing and characterizing cyber-physical systems. Fortunately, Turing Award winner Milner[38] proposed a Bigraph model that can describe both the physical world and the

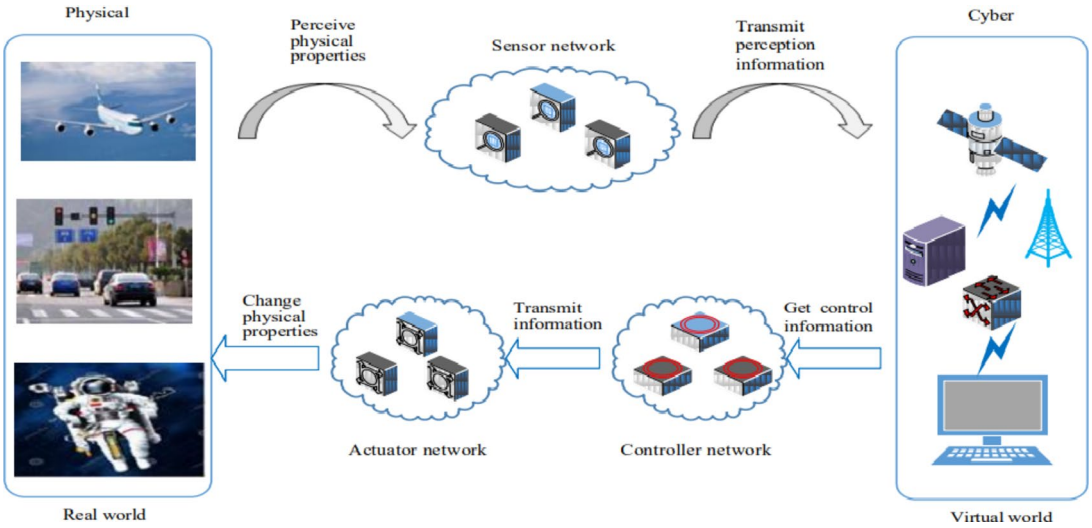

Fig. 1. The basic framework of cyber-physical systems.

information world for ubiquitous computing. Bigraph is well suited for representing cyber-physical systems and will be explained in more detail below.

Bigraph

(1) Bigraph static semantics.

Bigraph is composed of a place graph and a link graph, denoted as $B=(B^P, B^L)= (V, E, Ctrl, Prnt, Link)$: $< m,X>\rightarrow<n,Y>$. The place graph is $B^P=(V, Ctrl, Prnt)$: $m\rightarrow n$. The link graph is $B^L=(V, E, Ctrl, Link)$: $X\rightarrow Y$. V denotes a finite set of nodes. E denotes a finite set of edges. *Ctrl:* $V\rightarrow K$ denotes a mapping graph from node V to control K, that is, the type of node V is K, and K has three states: active, inactive and atomic. *Prnt:* $m \uplus v\rightarrow v \uplus n$ denotes a parent mapping, which is a non-cyclic graph, consisting of n trees forming a forest, where the names of the n outer domains indicate the roots of the n trees. *Link:* $X \uplus P\rightarrow E \uplus Y$ denotes a link mapping graph, $P=\{(v, i)|i \in mp(Ctrl(v))\}$ denotes the set of ports of all nodes of B, that is, (v,i) represents the i-th port of node v. The m denotes the number of inner sites. X denotes the set of inner names. The n denotes the number of outer domains. Y denotes the set of outer names. Figure 2 shows a Bigraph, and Figs. 3 and 4 show its place graph and link graph respectively.

As shown above, Bigraph is a graphical modeling method that allows observation, but it is not suitable for computer systems to understand and process. Therefore, Minler et al.[38] also introduced a term language in its algebraic form to formally reason and deduce the relevant properties of the system. The meanings of the terms and operations in the term language are shown in Table 1.

(2) Bigraph dynamic semantics.

Bigraph can also characterize the dynamics of the model. Its main idea is to use a reaction rule to transform Bigraph B into Bigraph B'. This transformation process is also known as the reaction system of Bigraph. Figure 5 shows an example of a reaction system. In a local service center *L-Center*, there is data D on a data server *LocalS*. For security purposes, the data D on the local server *LocalS* is remotely backed up to a cloud server *CloudS1* in the cloud service. This operation is achieved by a reaction rule (r, r'), which is a substitution process of a pair of subgraphs, as shown in Fig. 6. The r is the left-hand side subgraph, and r' is the right-hand side subgraph. That is, in the main graph, find and replace the matching left-hand side subgraph r with the right-hand side subgraph r'. The rest of the main graph remains unchanged after the reaction. For more details about Bigraph and its dynamic semantics, please refer to literature[38].

Architecture of cyber-physical systems

Static model of CPS

System architecture is a high-level abstraction that captures the essence of a system, typically composed of three parts: components, connectors, and constraints. It helps to avoid getting bogged down in tedious details too early when conducting evolutionary analysis on a system, and facilitates studying the system's structure and behavior from a macro perspective. Cyber-physical systems are an extension of traditional information systems, which link the physical world with the information world through networks to enable interaction. However, due to the openness, freedom, and difficulty of controlling the physical environment, such systems require more design and analysis of their structure and behavior essence from the system architecture level. From the system architecture perspective, the first step is to abstract out the essential components in cyber-physical systems,

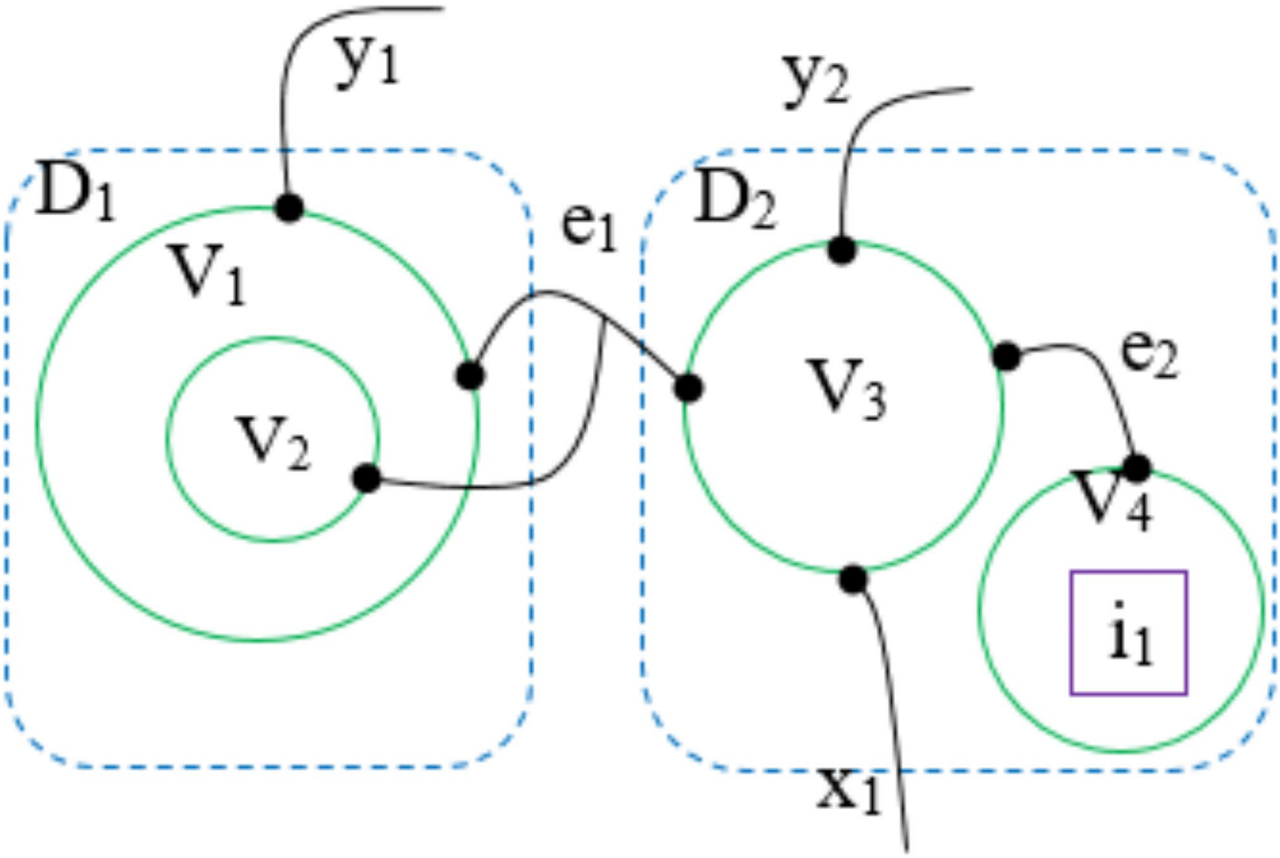

Fig. 2. Bigraph.

mainly including environment entities, sensor entities, controller entities, actuator entities, information entities, and environmental places, and provide their definitions.

Definition 1 (Environment Entity): An environment entity is an object or person that can be perceived in the physical world, with its own unique identity and physical attributes. Formally, it can be represented by a triple $M = (ID_M, K_M, Attr_M)$, where ID_M denotes the identity identifier of the environment entity, and K_M denotes the control type of the environment entity, $Attr_M$ represents the control attributes of environment entities. It usually has three types: active, inactive and atomic. Active representations can use reaction rules for it, inactive representations cannot use reaction rules for it, and atomic representations cannot be nested.

Definition 2 (Sensor Entity): A sensor entity is an entity that perceives the information of environment entities in a cyber-physical system. Formally, it can be represented by a triple $S = (ID_S, K_S, Attr_S)$, where ID_S denotes the identity identifier of sensor entities, K_S denotes the control type of sensor entities, $Attr_S$ denotes the control attributes of sensor entities. It usually has three types: active, inactive and atomic. Active means that the sensor is in sleep mode and can apply reaction rules. Inactive means that the sensor is in work mode and cannot apply reaction rules. Atomic means that the sensor cannot be nested but can apply reaction rules.

Definition 3 (Controller Entity): A controller entity is an entity that controls the information flow between environment entities and sensors in a cyber-physical system. Formally, it can be represented by a triple $C = (ID_C, K_C, Attr_C)$, where ID_C denotes the identity identifier of the controller entity, K_C denotes the control type of the controller entity, $Attr_C$ denotes the control attribute of the controller entity. It usually has three types: active, inactive and atomic. Active means that the controller is in sleep mode and can apply reaction rules. Inactive means that the controller is in work mode and cannot apply reaction rules. Atomic means that the controller cannot be nested but can apply reaction rules.

Definition 4 (Actuator Entity): An actuator entity is an entity that executes the information from controller entities a cyber-physical system. Formally, it can be represented by a triple $A = (ID_A, K_A, Attr_A)$, where ID_A denotes the identity identifier of the actuator entity, K_A denotes the type of control that the actuator entity belongs to, $Attr_A$ denotes the control attributes of the actuator entity. It usually has three types: active, inactive and atomic. Active means that the actuator is in sleep mode and can apply reaction rules. Inactive means that the actuator is in work mode and cannot apply reaction rules. Atomic means that the actuator cannot be nested but can apply reaction rules.

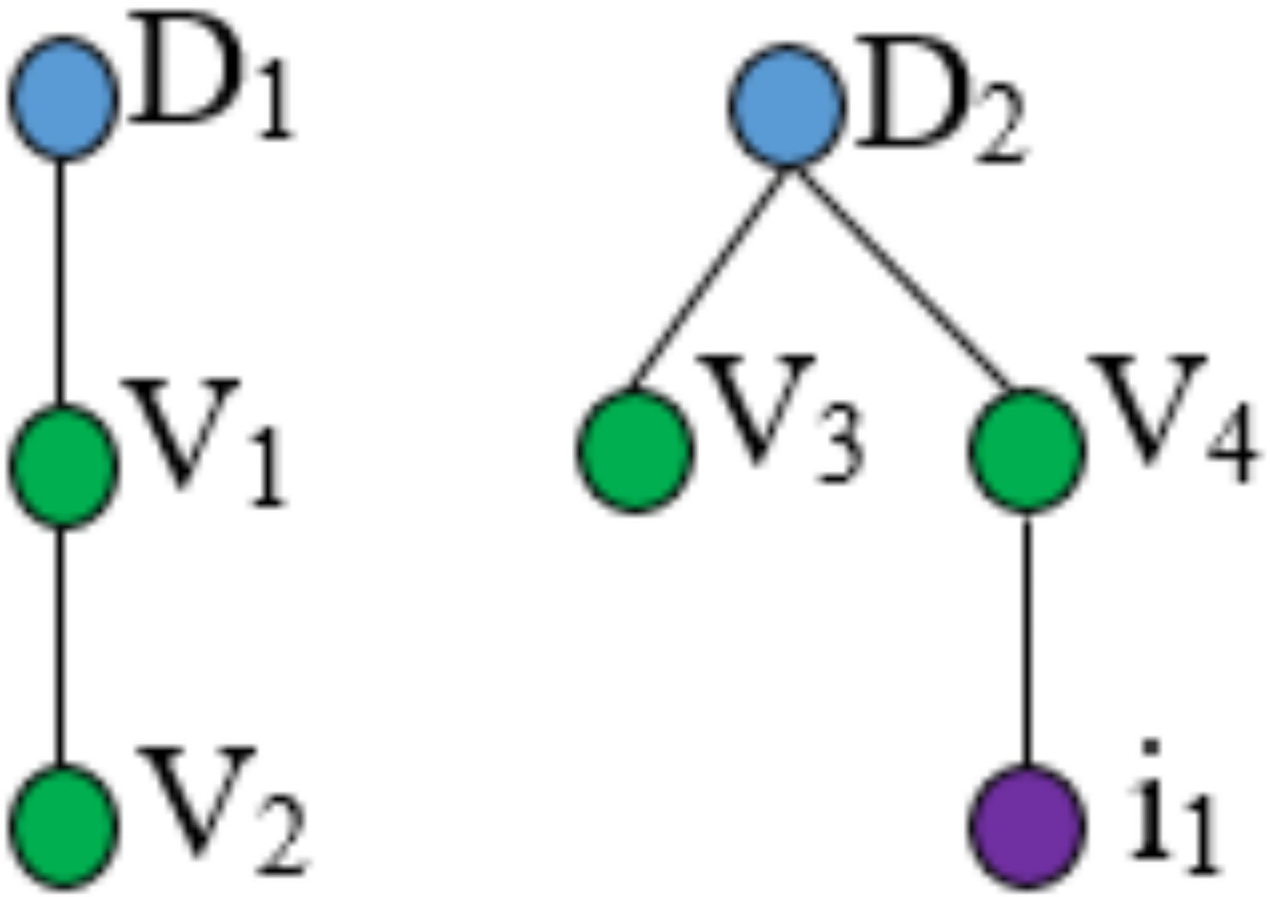

Fig. 3. Place graph.

Definition 5 (Information Entity): An information entity is a data entity in a sensor entity, a controller entity, or an actuator entity of a cyber-physical system. Formally, it can be represented by a triple $I = (ID_I, K_I, Attr_I)$, where ID_I denotes the identity identifier of the information entity, K_I denotes the control type of the information entity, $Attr_I$ denotes the control attribute of the information entity, which is usually atomic.

Definition 6 (Environment Place Entity): In cyber-physical system, the entity that is used to assign the location area of environment entity, sensor entity, controller entity, actuator entity and information entity is called environment place entity. Formally, it can be represented by a triple $D = (ID_D, K_D, Attr_D)$, where ID_D denotes the identity identifier of the environment place, K_D denotes the control type of the place, $Attr_D$ denotes the control attribute of the environment place. It usually has three types: active, inactive and atomic. Active means that reaction rules can be applied to the environment place entity. Inactive means that reaction rules cannot be applied to the environment place entity. Atomic means that it cannot be nested but reaction rules can be applied to it.

In order to more clearly demonstrate the control over various entities within the system structure, Table 2 summarizes the control types and control attributes corresponding to the entities defined above.

Definition 7 (Bigraph Model of Architecture of Cyber-Physical System): It is a Bigraph structure composed of environment entities, sensor entities, controller entities, actuator entities, information entities and environment place entities according to some constraint rules. Formally, it can be represented by a six tuples $CPSSA = (V, D, I, E, Prnt, Link)$, where $V=(M, S, C, A)$, M is environment entity set, S is sensor entity set, C is controller entity set, A is actuator entity set. D is the entity set of environment place. I is the information entity set. E is the edge set of V. $Prnt$ is a set of mapping relations between D and I to entity V, indicating the location area of V and the dependent entity of I, i.e. $Prnt$: $D \rightarrow V$; $Link$ is a set of mapping relations between V entity and V entity, i.e. $Link$: $V \rightarrow V$.

Specifically, as shown in Table 3, there is a well-established mapping relationship between the architecture of CPS and Bigraph.

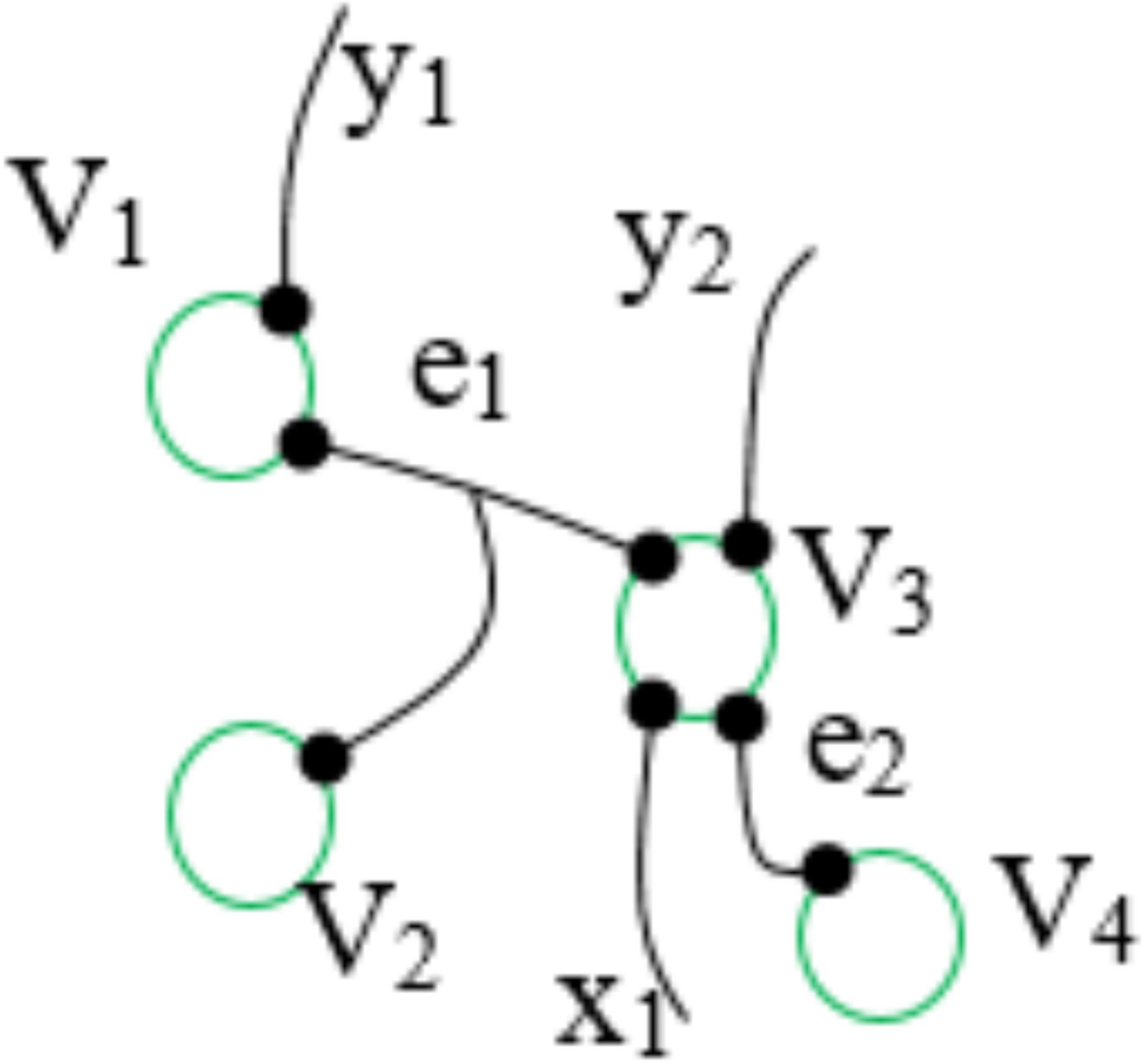

Fig. 4. Link graph.

Term	Meaning
U ° V	Composition operation, which puts the domain of V into the site of U, and then links the outer names of V with the inner names of U
U⊗V	Extension product, which puts U and V side by side from left to right
U||V	Parallel domains
U|V	Parallel nodes under a domain
U(V)	Nesting, U contains V
ϕ	Empty domain
—*i*	Site index is i
/x.U	The outer name x of U is replaced by an edge
x/y	Renaming, using the outer name x to replace the inner name y
e/x, y	One edge links ports x and y

Table 1. The term meaning of bigraph.

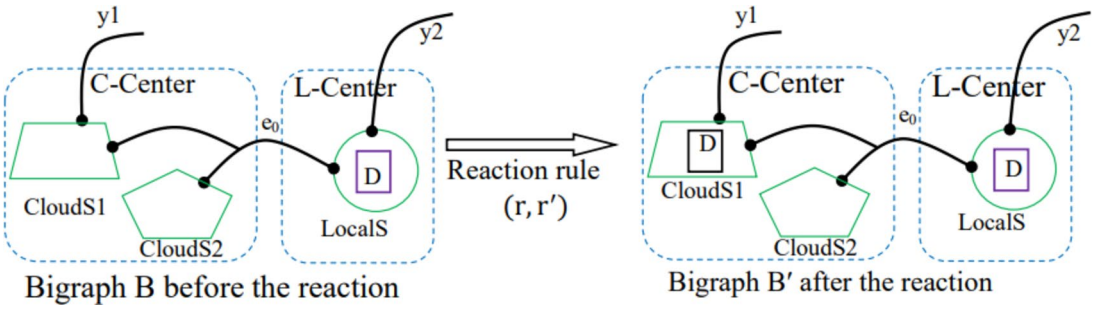

Fig. 5. Reaction system.

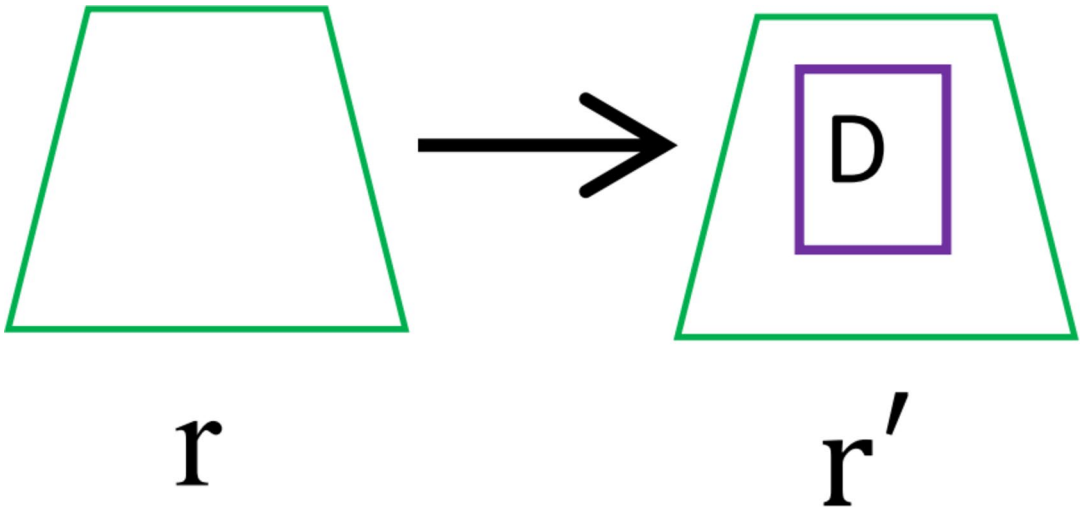

Fig. 6. Reaction rule.

Entity description	Control type	Control attributes
Environment entity	K_M	Active, inactive, atomic
Sensor entity	K_S	Active, inactive, atomic
Controller entity	K_C	Active, inactive, atomic
Actuator entity	K_A	Active, inactive, atomic
Information entity	K_I	Atomic
Environment place entity	K_D	Active, inactive, atomic

Table 2. Control set of entities.

Dynamic model of CPS

Computer information systems are constantly evolving in response to the changing requirements and environment. Cyber-physical system is a kind of extension of traditional computer information system, and it also undergoes evolution under the influence of requirements and environment. This process of change is called system evolution. System evolution is not a disorderly process, it usually follows certain rules. Evolution rules usually have three operations: adding, replacing and deleting. However, the three traditional evolutionary rules only describe the link relations of adding, replacing and deleting entities, and do not pay attention to the place relations of these entities. This paper focuses on the simultaneous link and place relationships involving the addition, replacement, and deletion of entities, with the relevant rules described below.

Definition 8 (Entity Adding Rule): In CPS, when the adding entity is one of environment entity M, sensor entity S, controller entity C and actuator entity A, the rule is $r1$: $\begin{cases} D(U_X) \rightarrow D(U_X|V_Y) \\ e/_{\varnothing,\varnothing} \rightarrow e/_{X,Y} \end{cases}$, as shown in Fig. 7.

Architecture of CPS	Bigraph
Environment entity	Node
Sensor entity	Node
Controller entity	Node
Actuator entity	Node
Information entity	Site
Connector	Edge
Port	Port
Environment place entity	Node
Virtual space of environment place entity	Domain
Entity type	Control
Constraint	Palace and link relationship

Table 3. The mapping relationship between architecture of CPS and bigraph.

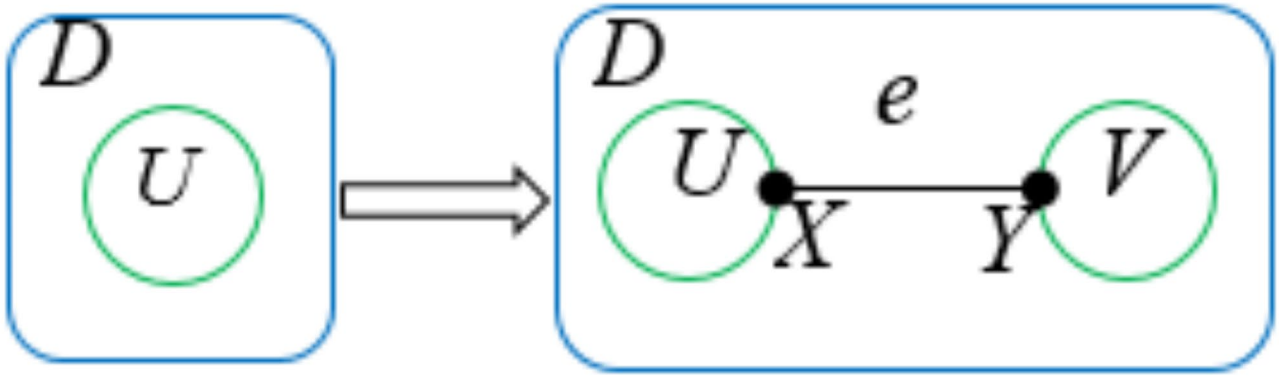

Fig. 7. Rule r1.

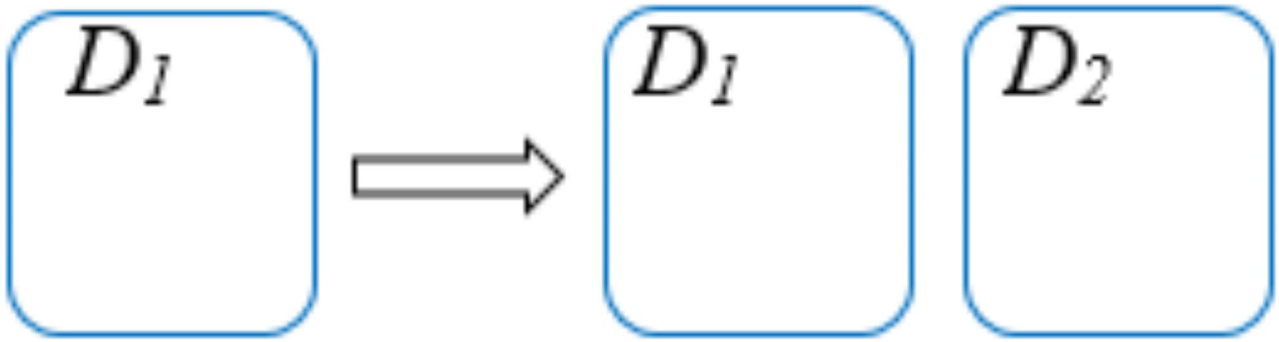

Fig. 8. Rule r2.

D denotes an environment place entity, which is used to determine the specific place of the adding entity. $U \in \{M, S, C, A\}$. V denotes an adding entity and $V \in \{M, S, C, A\}$. The e denotes the edge. X and Y denote ports. $\varnothing$ denotes empty port. The left part of $\rightarrow$ denotes the entity place and link relation before evolution. The right part of $\rightarrow$ denotes the entity place and link relation after evolution. When the adding entity is environment place entity D, and the rule is $r2$: $D_1 \rightarrow D_1 \,||\, D_2$, as shown in Fig. 8. Where D_2 is an adding environment place entity, without considering edges. When the adding entity is information entity I, and the rule is $r3$: $U \rightarrow U\,(V)$, as shown in Fig. 9. Where $U \in \{M, S, C, A\}$, V denotes an adding information entity I, without considering edges.

Definition 9 (Entity Replacing Rule): In CPS, when the replacing entity is one of environment entity M, sensor entity S, controller entity C and actuator entity A, the rule is $r4$: $\begin{cases} D(U_X|V_Y) \rightarrow D(U_X|V'_Z) \\ e/_{X,Y} \rightarrow e/_{X,Z} \end{cases}$, as shown

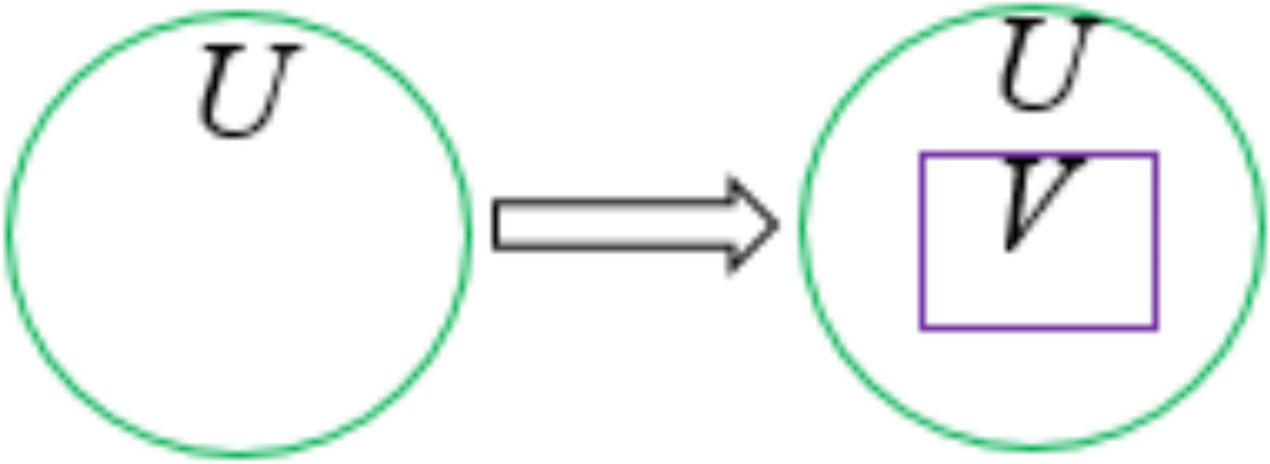

Fig. 9. Rule r3.

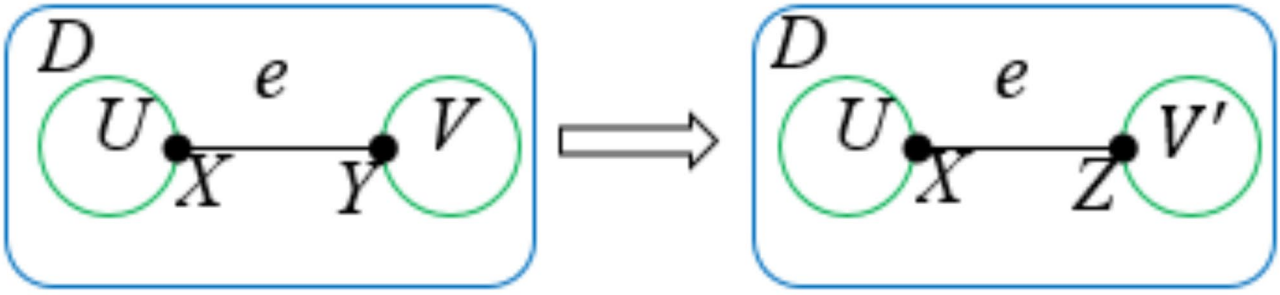

Fig. 10. Rule r4.

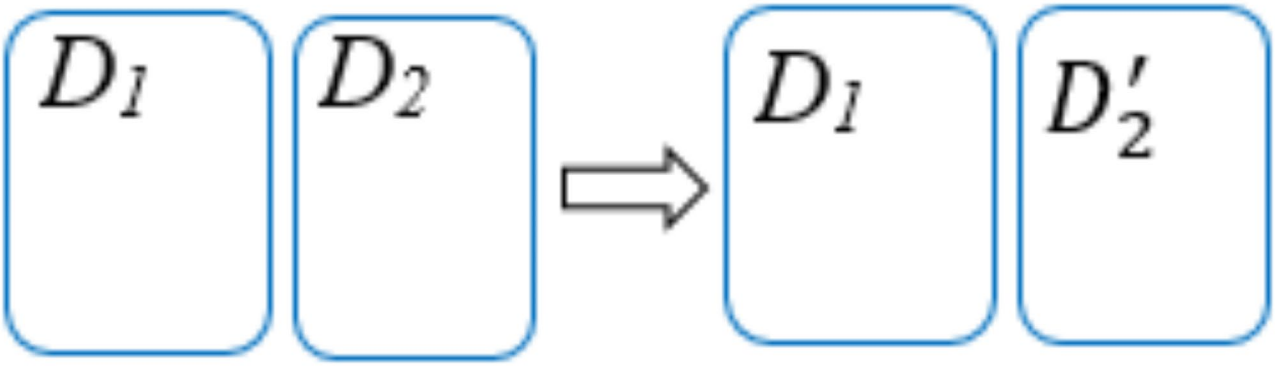

Fig. 11. Rule r5.

in Fig. 10. D denotes an environment place entity, which is used to determine the specific place of the replacing entity. $U \in \{M, S, C, A\}$. V denotes an before replacing entity and $V \in \{M, S, C, A\}$. V' denotes an after replacing entity and $V' \in \{M, S, C, A\}$. The e denotes the edge. X, Y and Z denote ports. The left part of $\rightarrow$ denotes the entity place and link relation before evolution. The right part of $\rightarrow$ denotes the entity place and link relation after evolution. When the replacing entity is environment place entity D, and the rule is *r5*: $D_1 \,||\, D_2 \rightarrow D_1 \,||\, D_2'$, as shown in Fig. 11. Where D_1 is a before replacing environment place entity, D_2 is an after replacing environment place entity, without considering edges. When the replacing entity is information entity I, and the rule is *r6*: $U(V) \rightarrow U(V')$, as shown in Fig. 12. Where $U \in \{M, S, C, A\}$, V is a before replacing information entity, V' is an after replacing information entity, without considering edges.

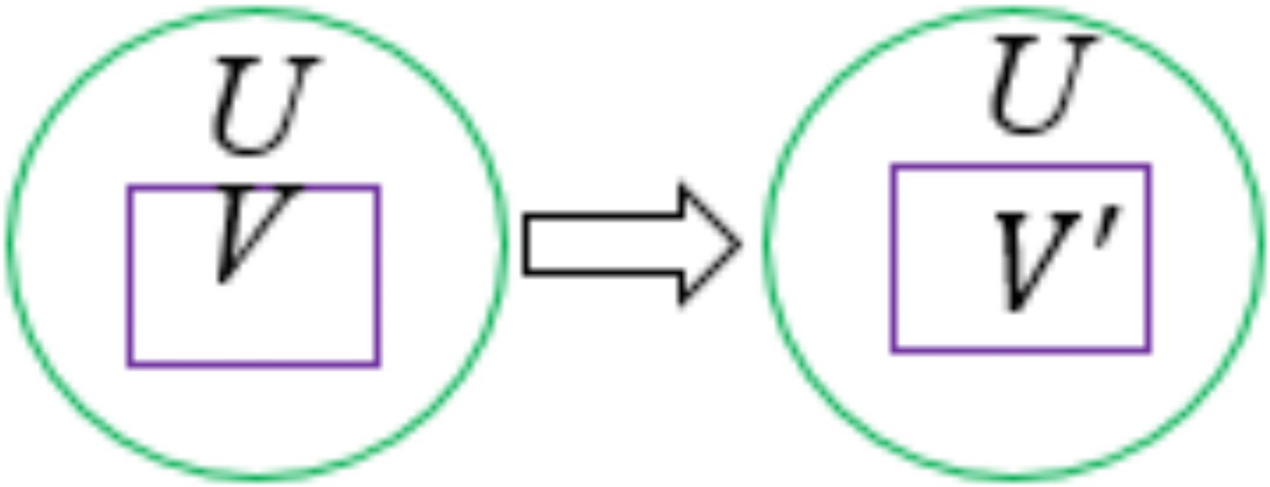

Fig. 12. Rule r6.

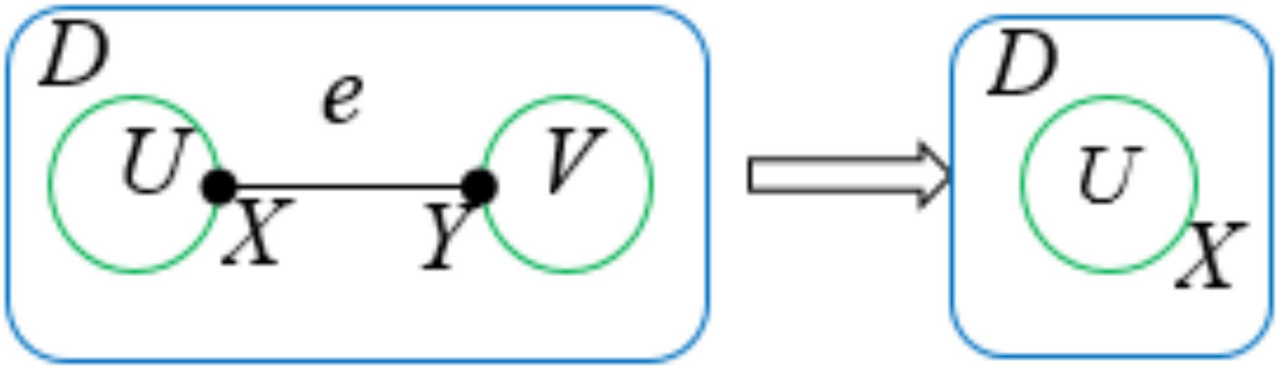

Fig. 13. Rule r7.

Definition 10 (Entity Deleting Rule): In CPS, when the deleting entity is one of environment entity *M*, sensor entity *S*, controller entity *C* and actuator entity *A*, the rule is *r7*: $\begin{cases} D(U_X|V_Y) \rightarrow D(U_X) \\ e/_{X,Y} \rightarrow e/_{\varnothing,\varnothing} \end{cases}$, as shown in Fig. 13. *D* denotes an environment place entity, which is used to determine the specific place of the deleting entity. $U \in \{M, S, C, A\}$. *V* denotes an deleting entity and $V \in \{M, S, C, A\}$. *e* denotes the edge. *X* and *Y* denote ports. $\varnothing$ denotes empty port. The left part of $\rightarrow$ denotes the entity place and link relation before evolution. The right part of $\rightarrow$ denotes the entity place and link relation after evolution. When the deleting entity is environment place entity *D*, and the rule is *r8*: $D_1 \,||\, D_2 \rightarrow D_1$, as shown in Fig. 14. Where D_2 is a deleting environment place entity, without considering edges. When the deleting entity is information entity *I*, and the rule is *r9*: $U(V) \rightarrow U$, as shown in Fig. 15.Where $U \in \{M, S, C, A\}$, *V* denotes an deleting information entity *I*, without considering edges.

Definition 11 (CPS entity evolution): CPS entity evolution is the process of changing the system to adapt to the evolving requirements by adding, replacing and deleting rules in CPS. Formally, suppose there are Bigraph B_0 and target Bigraph B_n, and the set of evolution rules $R = \{r1, r2, r4, r5, r7, r8\}$, the process of B_0 evolving into B_n is $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_{n-1} \xrightarrow{r_n} B_n$, where $r_i \in R$, $n \in N^*$, B_i is the intermediate state.

Proposition 1 *Suppose there are an initial Bigraph B_0 of CPS and a target Bigraph B_n. In target Bigraph, only environment entity M, sensor entity S, controller entity C, actuator entity A and environment place entity D are evolved, so initial Bigraph B_0 can always be transformed into target Bigraph B_n by using rule R = {r1, r2, r4, r5, r7, r8} for a limited number of times.*

Proof 1 (1) When $n = 1$, the target Bigraph is B_1. It is known that the transformation from B_0 to B_1 involves only the addition, deletion, or replacement of entities *M*, *S*, *C*, *A*, and any $r_i \in R$ represents a Bigraph transformation

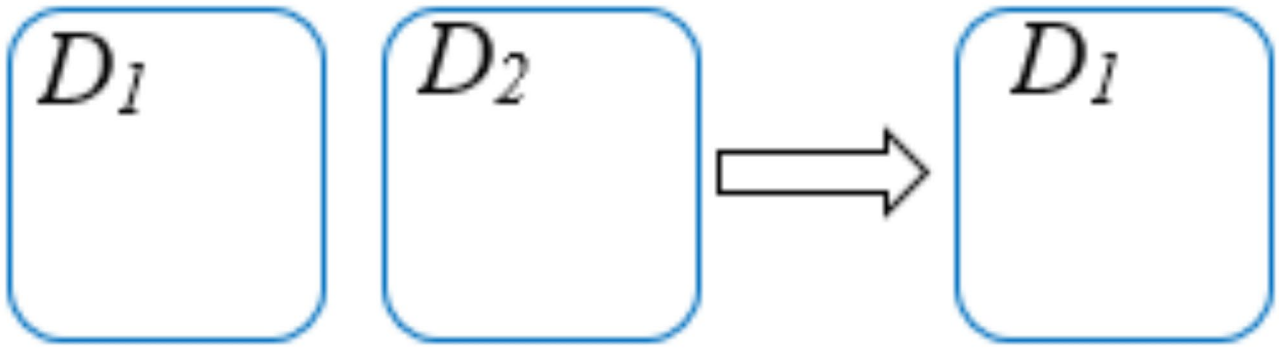

Fig. 14. Rule r8.

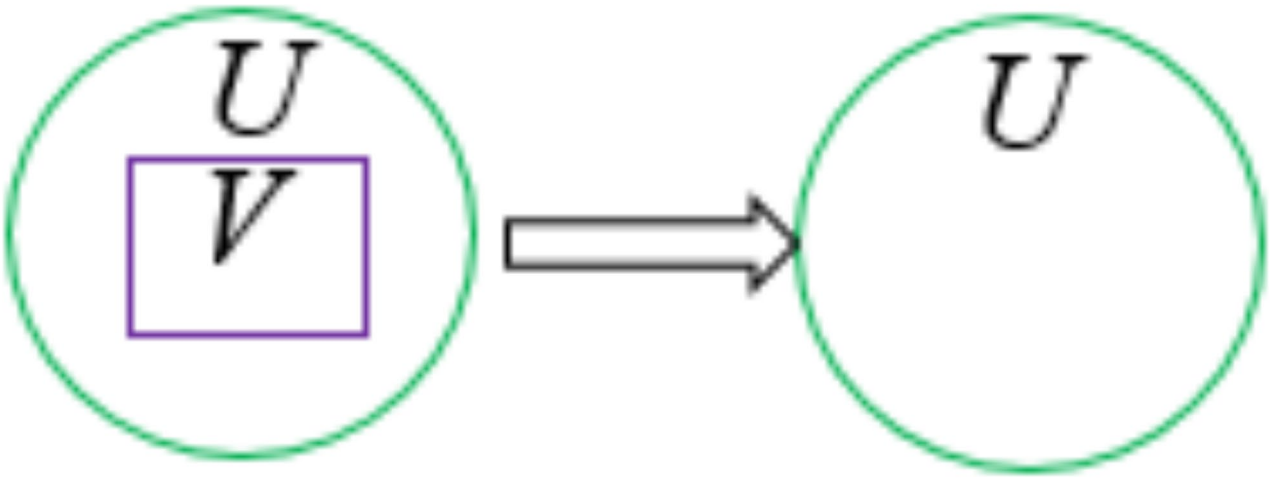

Fig. 15. Rule r9.

rule, which is an atomic operation for the addition, deletion, or replacement of *M*, *S*, *C*, *A* in Bigraph transformation. B_1 is obviously obtained from B_0 using one of these r_i transformations, i.e., $B_0 \xrightarrow{r_i} B_1$ holds.

(2) Suppose that when $n=k$, $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_{k-1} \xrightarrow{r_k} B_k$ holds. Similarly to $B_0 \xrightarrow{r_i} B_1$, any $B_{i-1} \xrightarrow{r_i} B_i$ also holds. When $n=k+1$, $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_k \xrightarrow{r_{k+1}} B_{k+1}$ can be transformed into $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_{(k+1)-1} \xrightarrow{r_{k+1}} B_{k+1}$.Therefore, when $n=k+1$, $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_k \xrightarrow{r_{k+1}} B_{k+1}$ holds, meaning that any $B_{i-1} \xrightarrow{r_i} B_i$ also holds.
From (1) and (2), it can be concluded that the proposition holds for all positive integers $n \in N^*$.

Definition 12 (Evolution of CPS Information Flow): In CPS, information is constantly added, replaced and deleted in the environment entity *M*, the sensor entity *S*, the controller entity *C* and the actuator entity *A*. This process is called information flow evolution. Formally, suppose there are Bigraph B_0 and target Bigraph B_n, and the set of evolution rules $R' = \{r3, r6, r9\}$,the process of B_0 g into B_n by information flow evolved is $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_{n-1} \xrightarrow{r_n} B_n$, where $r_i \in R'$, $n \in N^*$, B_i is the intermediate state.

Proposition 2 *Suppose there are an initial Bigraph B_0 and a target Bigraph B_n of CPS. In target Bigraph, only information entity I is evolved, so initial Bigraph B_0 can always be transformed into target Bigraph B_n by using rule $R' = \{r3, r6, r9\}$ for a limited number of times.*

Proof 2 (1) When $n=1$, the target Bigraph is B_1. It is known that the transformation from B_0 to B_1 involves only the addition, deletion, or replacement of information entities *I*, and any $r_i \in R'$ represents a Bigraph transformation rule, which is an atomic operation for the addition, deletion, or replacement of *I* in Bigraph transformation. B_1 is obviously obtained from B_0 using one of these r_i transformations, i.e., $B_0 \xrightarrow{r_i} B_1$ holds.

(2) Suppose that when $n=k$, $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_{k-1} \xrightarrow{r_k} B_k$ holds. Similarly to $B_0 \xrightarrow{r_i} B_1$, any $B_{i-1} \xrightarrow{r_i} B_i$ also holds. When $n=k+1$, $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_k \xrightarrow{r_{k+1}} B_{k+1}$ can be transformed into $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow$

$B_{(k+1)-1} \xrightarrow{r_{k+1}} B_{k+1}$.Therefore, when $n=k+1$, $B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_2} B_2 \rightarrow \cdots \rightarrow B_k \xrightarrow{r_{k+1}} B_{k+1}$ holds, meaning that any $B_{i-1} \xrightarrow{r_i} B_i$ also holds.
From (1) and (2), it can be concluded that the proposition holds for all positive integers $n \in N^*$.

Proposition 3 *Suppose there are an initial Bigraph B_0 and a target Bigraph B_n of CPS. Regardless of which entity is evolved in B_n, the initial Bigraph B_0 can always be transformed into the target Bigraph B_n by using the rules R = { r1, r2, r4, r5, r7, r8} and R' = { r3, r6, r9 } for a limited number of times.*

Proof 3 Based on propositions 1 and 2, and with reference to proof 1 and proof 2, proposition 3 is true.

Definition 13 (CPS evolution Model): Assuming a CPS (Cyber-Physical System) architecture is described as *CPSSA=(V, D, I, E, Prnt, Link)* using Bigraph, and nine evolution rules, *r1* to *r9*, are defined on this structure, when *CPSSA* evolves into a desired *CPSSA* driven by this set of rules, then the comprehensive model of CPS evolution is described as $CPSSA_0 \xrightarrow{r_1} CPSSA_1 \xrightarrow{r_2} CPSSA_2 \rightarrow \cdots \rightarrow CPSSA_{n-1} \xrightarrow{r_n} CPSSA_n$. Where $V=(M, S, C, A)$, M is environment entity set, S is sensor entity set, C is controller entity set, A is actuator entity set. D is the entity set of environment place. I is the information entity set. E is the edge set of V. *Prnt* is a set of mapping relations between D and I to entity V, indicating the location area of V and the dependent entity of I, i.e. *Prnt*: $D \rightarrow V$; *Link* is a set of mapping relations between V entity and V entity, i.e. *Link*: $V \rightarrow V$. Evolution rule *ri* is one of the evolution rules specified in Definition 8, Definition 9, and Definition 10.

Constraint analysis and checking algorithm design of architecture

Architecture as a high-level abstract description of a system, its changes are not arbitrary, it always has to conform to pre-agreed constraints. From the overall structure, these constraints mainly involve consistency, integrity and accessibility constraints.

Definition 14 (Consistency of CPS Dynamic Evolution): After the dynamic evolution of CPS, if all environment entities M are linked to sensor entities S, all sensor entities S are linked to controller entities C, all controller entities C are linked to actuator entities A, all actuator entities A are linked to environment entities M, and all information entities I are placed only one of sensor entities S, controller entities C or actuator entities A, the dynamic evolution of CPS is consistent. Formally, in *CPSSA*, if $\forall M_X, \exists S_Y, \exists e/_{X,Y}$, and if $\forall S_X, \exists C_Y, \exists e/_{X,Y}$, and if $\forall C_X, \exists A_Y, \exists e/_{X,Y}$, and if $\forall A_X, \exists M_Y, \exists e/_{X,Y}$, and if $\forall I, \exists S(I)$, or $\exists C(I)$, or $\exists A(I)$, then *CPSSA* ?*Const*, where X and Y denote port, *Const* denotes consistency, ?denotes that it satisfies some property.

Definition 15 (Integrity of CPS Dynamic Evolution): After the dynamic evolution of CPS, if the number of sensor entities S in an environment place entity D is required to be no less than m, the number of one controller entity C link sensor entity S is required to be no more than n, the number of one controller entity C link actuator entity S is required to be no more than k, the dynamic evolution of CPS owns integrity. Formally, in *CPSSA*, if $\forall D, \exists \{D(S_1), D(S_2), \ldots, D(S_p)\}, p \geq m$, and if $\forall C_X, \exists \{S^1_{Y1}, S^2_{Y2}, \ldots, S^p_{Yp}\}, \{e/_{X,Y1}, e/_{X,Y2}, \ldots, e/_{X,Yp}\}, p \leq n$, and if $\forall C_X, \exists \{A^1_{Y1}, A^2_{Y2}, \ldots, A^p_{Yt}\}, \{e/_{X,Y1}, e/_{X,Y2}, \ldots, e/_{X,Yt}\}, t \leq k$, then *CPSSA* ?*Inte*, where X and Y denote port, *Inte* denotes integrity, ?denotes that it satisfies some property.

Definition 16 (Reachability of CPS Dynamic Evolution): After the dynamic evolution of CPS, if they start from any environment entity M, it can always reach the actuator entity A through sensor entity S and controller entity C, the dynamic evolution of *CPS* is reachable. Formally, in *CPSSA*, if $\forall M, \exists$ *Path*: $M_{X1} \xrightarrow{e/_{X1,X2}} S_{X2} \xrightarrow{e/_{X2,X3}} C_{X3} \xrightarrow{e/_{X3,X4}} A_{X4}$, then *CPSSA* ?*Reac*, where *Xi* denotes port, *Reac* denotes Completeness, ?denotes that it satisfies some property.

Proposition 4 Evolution rules of information flow can guarantee consistency, completeness and reachability. Formally, suppose $CPSSA_0 = (V, D, I, E, Prnt, Link)$, where $V=(M, S, C, A)$, $CPSSA_0$?*Const*, $CPSSA_0$?*Inte*, $CPSSA_0$?*Reac*, if the evolution rule of information flow is R' ={r3, r6, r9}, $CPSSA_0 \xrightarrow{r_1} CPSSA_1 \xrightarrow{r_2} CPSSA_2 \rightarrow \cdots \rightarrow CPSSA_{n-1} \xrightarrow{r_n} CPSSA_n$, $r_i \in R'$, $n \in N^*$, then $CPSSA_n$?*Const*, $CPSSA_n$?*Comp*, $CPSSA_n$?*Reac*, $CPSSA_i$ is the intermediate state.

Proof 4 (1) When $n=1$, the target Bigraph is $CPSSA_1$. It is known that the transformation from $CPSSA_0$ to $CPSSA_1$ involves only the addition, deletion, or replacement of one entity I. For any $r_i \in R'$, it satisfies $\forall M_X, \exists S_Y, \exists e/_{X,Y}$, and $\forall S_X, \exists C_Y, \exists e/_{X,Y}$, and $\forall C_X, \exists A_Y, \exists e/_{X,Y}$, and $\forall A_X, \exists M_Y, \exists e/_{X,Y}$, $\forall I, \exists S(I)$, or $\exists C(I)$, or $\exists A(I)$, i.e., *CPSSA* ?*Const*. It can also satisfy $\forall D, \exists \{D(S_1), D(S_2), \ldots, D(S_p)\}, p \geq m$, and $\forall C_X, \exists \{S^1_{Y1}, S^2_{Y2}, \ldots, S^p_{Yp}\}, \{e/_{X,Y1}, e/_{X,Y2}, \ldots, e/_{X,Yp}\}, p \leq n$, and $\forall C_X, \exists \{A^1_{Y1}, A^2_{Y2}, \ldots, A^p_{Yt}\}, \{e/_{X,Y1}, e/_{X,Y2}, \ldots, e/_{X,Yt}\}, t \leq k$, i.e., *CPSSA* ?*Inte*. It can also satisfy $\forall M, \exists$ *Path*: $M_{X1} \xrightarrow{e/_{X1,X2}} S_{X2} \xrightarrow{e/_{X2,X3}} C_{X3} \xrightarrow{e/_{X3,X4}} A_{X4}$, i.e., *CPSSA* ?*Reac*. Therefore, for $CPSSA_0 \xrightarrow{r_i} CPSSA_1$, the $CPSSA_1$?*Const*, and $CPSSA_1$?*Comp*, and $CPSSA_1$?*Reac* hold.

(2) Suppose when $n=k$, $CPSSA_0 \xrightarrow{r_1} CPSSA_1 \xrightarrow{r_2} CPSSA_2 \rightarrow \cdots \rightarrow CPSSA_{k-1} \xrightarrow{r_k} CPSSA_k$, holds. Similarly to $CPSSA_0 \xrightarrow{r_i} CPSSA_1$, the $CPSSA_1$?*Const*, and $CPSSA_1$?*Comp*, and $CPSSA_1$?*Reac* hold. For any $B_{i-1} \xrightarrow{r_i} B_i$, the $CPSSA_i$?*Const*, and $CPSSA_i$?*Comp*,and $CPSSA_i$?*Reac* hold.When $n=k+1$, $CPSSA_0 \xrightarrow{r_1} CPSSA_1 \xrightarrow{r_2} CPSSA_2 \rightarrow \cdots \rightarrow CPSSA_k \xrightarrow{r_{k+1}} CPSSA_{k+1}$ can be transformed into $CPSSA_0 \xrightarrow{r_1} CPSSA_1 \xrightarrow{r_2} CPSSA_2 \rightarrow \cdots \rightarrow CPSSA_{(k+1)-1} \xrightarrow{r_{k+1}}$

$CPSSA_{k+1}$.Therefore, when $n=k+1$, for $CPSSA_0 \xrightarrow{r_1} CPSSA_1 \xrightarrow{r_2} CPSSA_2 \rightarrow \cdots \rightarrow CPSSA_k \xrightarrow{r_{k+1}} CPSSA_{k+1}$, the $CPSSA_{k+1}$?*Const*, and $CPSSA_{k+1}$?*Comp*, and $CPSSA_{k+1}$?*Reac* hold.

From (1) and (2), we can conclude that the proposition holds true for all positive integers $n \in N^*$.

In CPS, consistency checking of the system is required to avoid system insecurity, as the system consistency may be destroyed by the use of evolution rules. The main idea of the consistency checking method is illustrated in Fig. 16. To address various application scenarios, I further categorize it into three types of consistency methods: (1) Perform a global consistency check only once after all the evolution rules are executed, its specific process is shown in Algorithm 1. (2) When every evolution rule is used, a global consistency check is performed. This allows early checking of inconsistencies without global rollbacks. When the number of evolution rules is less, the execution efficiency is very good. However, as the number of evolutionary rules increases, the evolutionary execution time will be too long and the execution efficiency will be low, its specific process is shown in Algorithm 2. (3) When every evolution rule is used, a local consistency check is performed. In other words, the scope of influence of an evolution rule is knew, and then only the scope of influence is checked. When the scale of the architecture is getting larger and larger, the efficiency of this method is quite high, its specific process is shown in Algorithm 3.

Input: After Evolved CPS architecture $SA=(V, D, I, E, Prnt, Link)$, $SA^P=(V, D, I, Prnt)$, $SA^L=(V, E, Link)$, where $V=(M, S, C, A)$
Output: *Cons*
1: $Cons \leftarrow \emptyset$, $Cons^P \leftarrow \emptyset$, $Cons^L \leftarrow \emptyset$
2: **for** $\forall I \in SA^P=(V, D, I, Prnt)$
//Checking consistency for *SA* place graph
3: **if** $M(I)$ or $S(I)$ or $C(I)$ or $A(I)$
4: $Cons^P \leftarrow 1$;
5: **else** $Cons^P \leftarrow 0$;
6: **end if**
7: **end for**
8: **for** $\forall M_{X1}, \forall S_{X2}, \forall C_{X3}, \forall A_{X4} \in SA^L=(V, E, Link)$
//Checking consistency for SA Link graph
9: **if** $(\exists S_{Y1}, e/_{X1,Y1})$ and $(\exists C_{Y2}, e/_{X2,Y2})$ and $(\exists A_{Y3}, e/_{X3,Y3})$ and $(\exists M_{Y4}, e/_{X4,Y4})$=True
10: $Cons^L \leftarrow 1$
11: **else** $Cons^L \leftarrow 0$
12: **end if**
13: : **end for**
14: $Cons \leftarrow Cons^P \wedge Cons^L$
15: **return** *Cons*

Algorithm 1. Checking Architecture Consistency, Abbreviated CSC1.

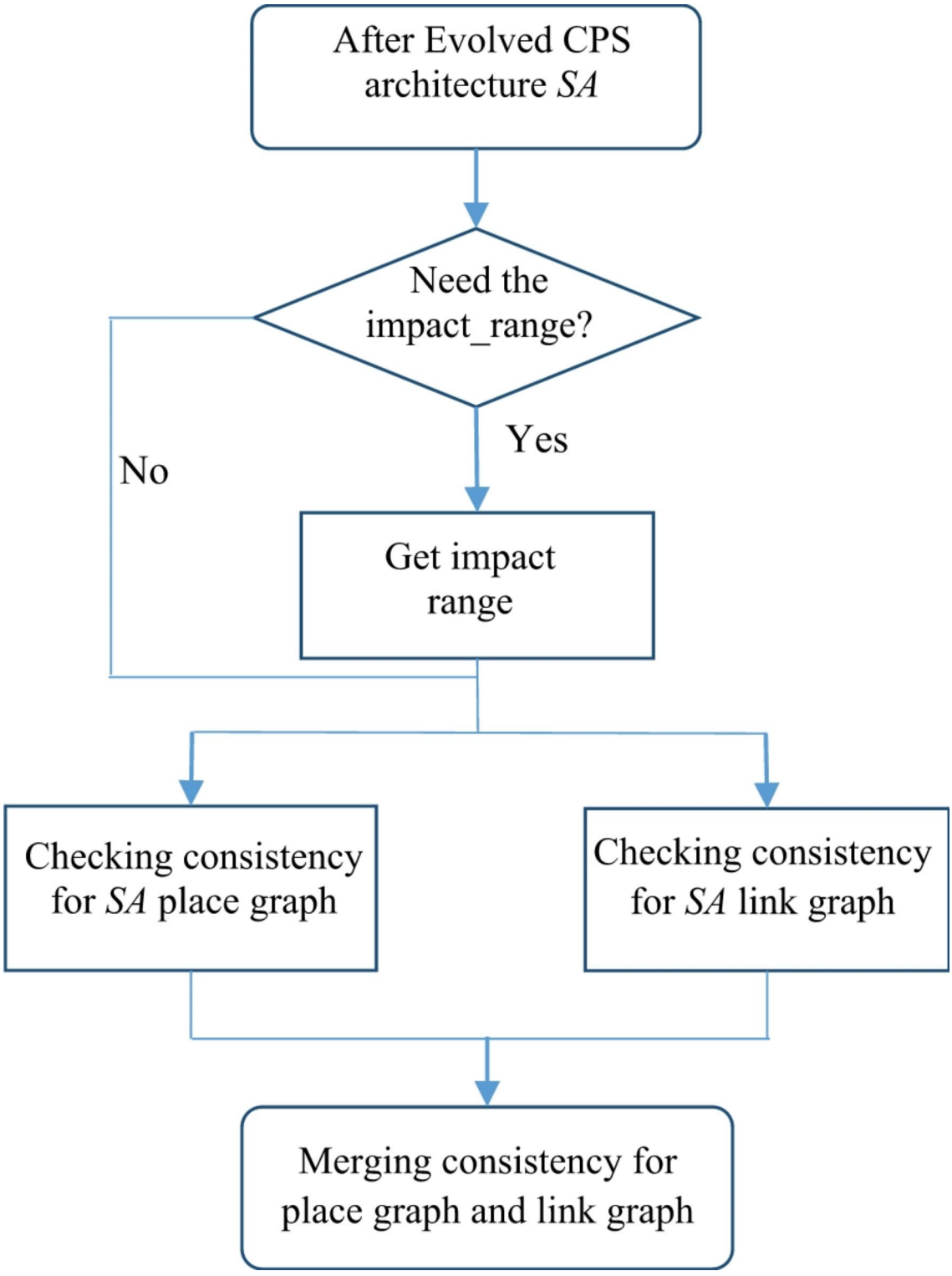

Fig. 16. Consistency checking flowchart.

Input: Initial CPS architecture $SA_0=(V, D, I, E, Prnt, Link)$, $SA_0^P=(V, D, I, Prnt)$, $SA_0^L=(V, E, Link)$, where $V=(M, S, C, A)$, $R=\{r1, r2, r3, r4, r5, r6, r7, r8, r9\}$

Output: *Cons*

$Cons \leftarrow \emptyset$, $Cons^P \leftarrow \emptyset$, $Cons^L \leftarrow \emptyset$
for SA_0 to use $r_i \in R$
  for $\forall I \in SA^P=(V, D, I, Prnt)$
    // *SA* is the evolved CPS architecture
    //Checking consistency for *SA* place graph
    if $M(I)$ or $S(I)$ or $C(I)$ or $A(I)$
      $Cons^P \leftarrow 1$
    else $Cons^P \leftarrow 0$
    end if
  end for
  for $\forall M_{X1}, \forall S_{X2}, \forall C_{X3}, \forall A_{X4} \in SA^L=(V, E, Link)$
    //Checking consistency for *SA* Link graph
    if $(\exists S_{Y1}, e/_{X1,Y1})$ and $(\exists C_{Y2}, e/_{X2,Y2})$ and $(\exists A_{Y3}, e/_{X3,Y3})$ and $(\exists M_{Y4}, e/_{X4,Y4})$=True
      $Cons^L \leftarrow 1$
    else $Cons^L \leftarrow 0$
    end if
  end for
  $Cons \leftarrow Cons^P \wedge Cons^L$
  return *Cons*
end for

Algorithm 2. Checking Architecture Consistency, Abbreviated CSC2.

Input: Initial CPS architecture $SA_0=(V, D, I, E, Prnt, Link)$, $SA_0^P=(V, D, I, Prnt)$, $SA_0^L=(V, E, Link)$, where $V=(M, S, C, A)$, $R=\{r1, r2, r3, r4, r5, r6, r7, r8, r9\}$
Output: *Cons*
1: $Cons \leftarrow \emptyset$, $Cons^P \leftarrow \emptyset$, $Cons^L \leftarrow \emptyset$
2: **for** SA_0 to use $r_i \in R$ // *SA* is the evolved CPS architecture
3: $SA^{P'} \leftarrow$get_impact_range(r_i, SA^P)
// $SA^{P'}$ is sub graph of SA^P
4: $SA^{L'} \leftarrow$get_impact_range(r_i, SA^L)
// $SA^{L'}$ is sub graph of SA^L
5: **for** $\forall I \in SA^{P'}=(V, D, I, Prnt)$
//Checking consistency for *SA* place graph
6: **if** $M(I)$ or $S(I)$ or $C(I)$ or $A(I)$
7: $Cons^P \leftarrow 1$
8: **else** $Cons^P \leftarrow 0$
9: **end if**
10: **end for**
11: **for** $\forall M_{X1}, \forall S_{X2}, \forall C_{X3}, \forall A_{X4} \in SA^{L'}=(V, E, Link)$
//Checking consistency for *SA* Link graph
12: **if** $(\exists S_{Y1}, e/_{X1,Y1})$ and $(\exists C_{Y2}, e/_{X2,Y2})$ and $(\exists A_{Y3}, e/_{X3,Y3})$ and $(\exists M_{Y4}, e/_{X4,Y4})$=True
13: $Cons^L \leftarrow 1$
14: **else** $Cons^L \leftarrow 0$
15: **end if**
16: **end for**
17: $Cons \leftarrow Cons^P \wedge Cons^L$
18: **return** *Cons*
19: **end for**

Algorithm 3. Checking Architecture Consistency, Abbreviated CSC3.

In CPS, since the system integrity may be destroyed by using evolutionary rules, the system needs to be checked for integrity to avoid system insecurity. In terms of integrity checking, when all evolution rules are executed, only one global integrity checking is performed. Because integrity mainly involves the number of entities, it is not necessary to check every evolution rule after it is executed. If that is done, the execution efficiency is too low. Its specific process is shown in Fig. 17 and Algorithm 4.

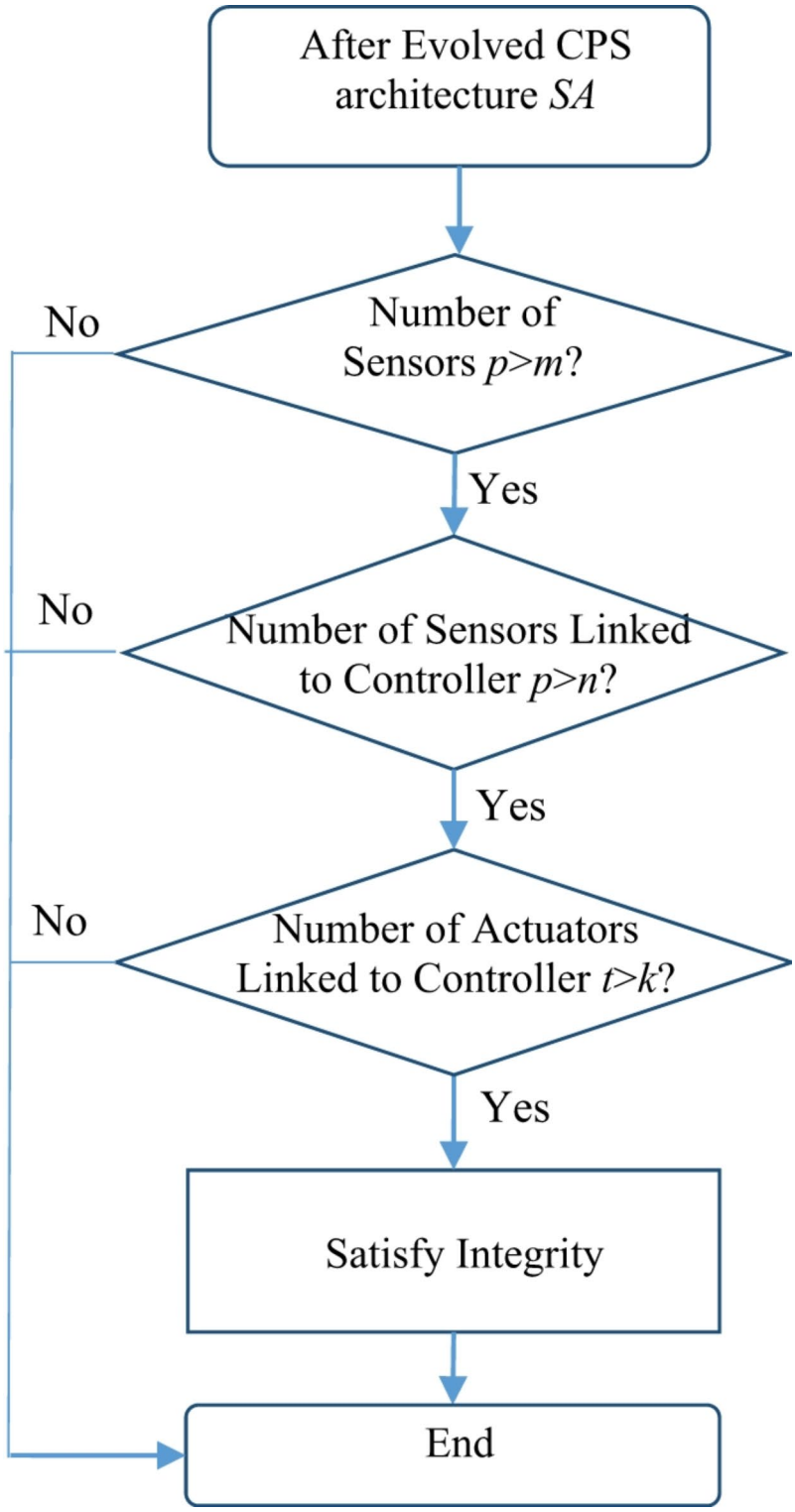

Fig. 17. Integrity checking flowchart.

Input: After Evolved CPS architecture $SA=(V, D, I, E, Prnt, Link)$, $SA^P=(V, D, I, Prnt)$, $SA^L=(V, E, Link)$, where $V=(M, S, C, A)$
Output: *Inte*
1: $Inte \leftarrow \emptyset$,
2: **for** $\forall D \in SA^P=(V, D, I, Prnt)$
3: **if** $\exists\{D(S_1), D(S_2), \ldots, D(S_p)\}$ and $p \geq m$
4: $S^{Num} \leftarrow 1$
5: **else** $S^{Num} \leftarrow 0$;
6: **end if**
7: **end for**
8: **for** $\forall C_X \in SA^L=(V, E, Link)$
9: **if** $\exists\{S^1_{Y1}, S^2_{Y2}, \ldots, S^p_{Yp}\}$ and $\{e/_{X,Y1}, e/_{X,Y2}, \ldots, e/_{X,Yp}\}$ and $p \leq n$
10: $C^{MdrcLinkSensor} \leftarrow 1$
11: **else** $Cons^L \leftarrow 0$
12: **end if**
13: **for** $\forall C_X \in SA^L=(V, E, Link)$
14: **if** $\exists\{A^1_{Y1}, A^2_{Y2}, \ldots, A^p_{Yt}\}$ and $\{e/_{X,Y1}, e/_{X,Y2}, \ldots, e/_{X,Yt}\}$ and $t \leq k$
15: $C^{MdrcLinkActuator} \leftarrow 1$
16: **else** $Cons^L \leftarrow 0$
17: **end if**
18: **end for**
19: $Inte \leftarrow S^{Num} \wedge C^{MdrcLinkSensor} \wedge C^{MdrcLinkActuator}$
20: **return** *Inte*

Algorithm 4. Checking Architecture Integrity, Abbreviated CAI

In CPS, since the system reachability may be destroyed by using evolutionary rules, the system needs to be checked for reachability to avoid system insecurity. In terms of reachability checking, when all evolution rules are executed, only one global reachability checking is performed. Because reachability mainly involves the global entities, it is not necessary to check every evolution rule after it is executed. If that is done, the execution efficiency is too low. Its specific process is shown in Fig. 18 and Algorithm 5.

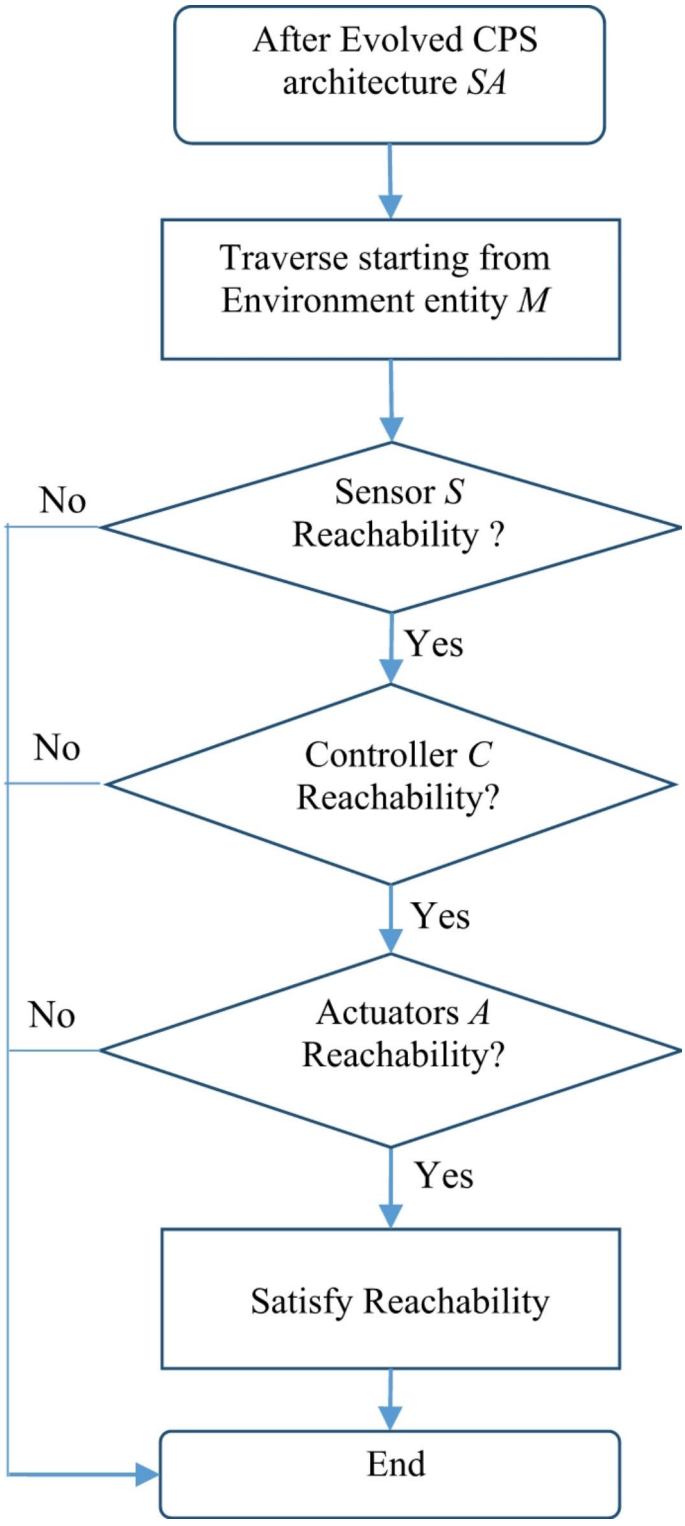

Fig. 18. Reachability checking flowchart.

Input: After Evolved CPS architecture $SA=(V, D, I, E, Prnt, Link)$, $SA^P=(V, D, I, Prnt)$, $SA^L=(V, E, Link)$, where $V=(M, S, C, A)$
Output: $Reac$
1: $Reac \leftarrow \emptyset$
2: **for** $\forall M \in SA^L=(V, E, Link)$
3: **if** $\exists Path: M_{X1} \xrightarrow{e/X1,X2} S_{X2} \xrightarrow{e/X2,X3} C_{X3} \xrightarrow{e/X3,X4} A_{X4} \xrightarrow{e/X4,X5} M_{X5}$
4: $S^{Num} \leftarrow 1$
5: **else** $S^{Num} \leftarrow 0$;
6: **end if**
7: **end for**
8: **return** $Reac$

Algorithm 5. Checking Architecture Reachability, Abbreviated CAR

Instance and experimental analysis

Experiment 1: Evolution modeling and analysis for smart meeting room system.

Smart meeting room system modeling

A smart meeting room system is an office system that enables companies to enhance the comfort and efficiency of their meetings. It is a representative example of cyber-physical systems. It involves various entities. Initially, Table 4 presents the fundamental control set for the smart meeting room system. Subsequently, the Bigraph modeling tool, BigRed[39], is utilized to establish the initial Bigraph model of the smart meeting room system, as illustrated in Fig. 19.

In Fig. 19, the smart meeting room system mainly consists of one environmental place entity *smartmeetingroom*, two environmental entities *air* and *light*, one temperature sensor *tempsensor*, one light sensor *lumsensor*, one global *controller*, and four actuators: *aircd*, *curtain*, *lamp*, and *projector*. The system assumes that these actuators

Control	Attribute	Description
MROOM	Atomic	Smart meeting room
AIR	Atomic	Air
LIGHT	Atomic	Lamplight
TS	Active	Temperature sensor
LS	Active	Light sensor
CR	Active	Global controller
AC	Active	Air conditioner
CT	Active	Curtain actuator
LP	Active	Lamp
PJ	Active	Projector
OFF	Atomic	Open
ON	Atomic	Close
HOT	Atomic	Temperature hot information
COLD	Atomic	Temperature cold information
STRONG	Atomic	Light strong information
WEAK	Atomic	Light weak information

Table 4. Control set of smart meeting room system.

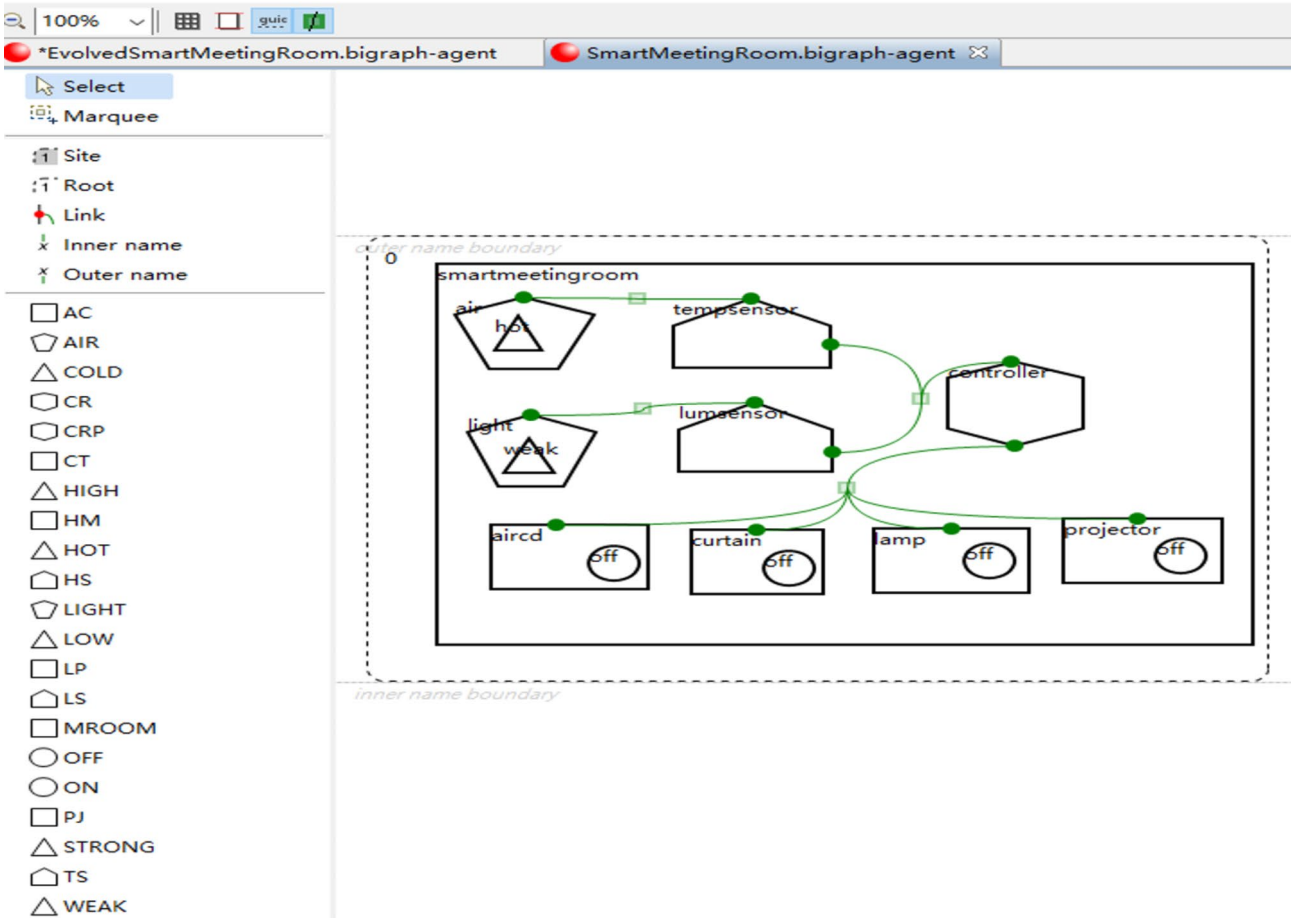

Fig. 19. Initial Bigraph model of the smart meeting room system.

are all off in the initial state, represented by the closed entity off, which is embedded in each of these actuators. It is also assumed that the temperature sensor and the light sensor perceive hot information and weak light intensity respectively in the initial state, represented by the entities *hot* and *weak*. The connection relationships of these entities are shown in Fig. 19. *Air* is connected to *tempsensor*, *light* is connected to *lumsensor*, *tempsensor* and *lumsensor* are both connected to *controller*, and *controller* is connected to *aircd*, *curtain*, *lamp*, and *projector* respectively.

Modeling of evolution rules

The meeting room is often used for long meetings and is closed most of the time, which makes people feel dry and uncomfortable. Therefore, people want to upgrade the smart meeting room facilities to add a humidification function, without affecting the normal operation of the existing system. To achieve this, a humidifier, a humidity sensor, and two indicators of high and low humidity are needed. However, the original controller has a limited capacity and cannot control more than four actuators, so it needs to be replaced with a more powerful controllerplus. To achieve these evolution objectives, as shown in Figs. 20, 21, 22 and 23, and Fig. 24, four add evolution rules, namely r11, r12, r13, and r14, as well as one replace evolution rule r15 are designed.

(1) Add evolution rules.

(2) Replace evolution rule.

(3) Information flow evolution rule.

The smart meeting system has an information flow feature, which means that information can move between different entities in the system. This movement needs to be captured by evolution rules in the model, called information flow evolution rules. Information is an atomic entity that can be embedded in various other entities. For example, when the humidity sensor detects low humidity in the air, the low information will transfer from the air entity to the humidity sensor entity. This can be expressed by the information flow evolution rule shown in Fig. 25.

The term language representation of the evolution rules above is as follows:

r11:smartmeetingroom(air_x|$controller_{yz}$) → $e1/_{x,u}.e2/_{v,y}$.smartmeetingroom(air_x|$humisensor_{uv}$|$controller_{yz}$).

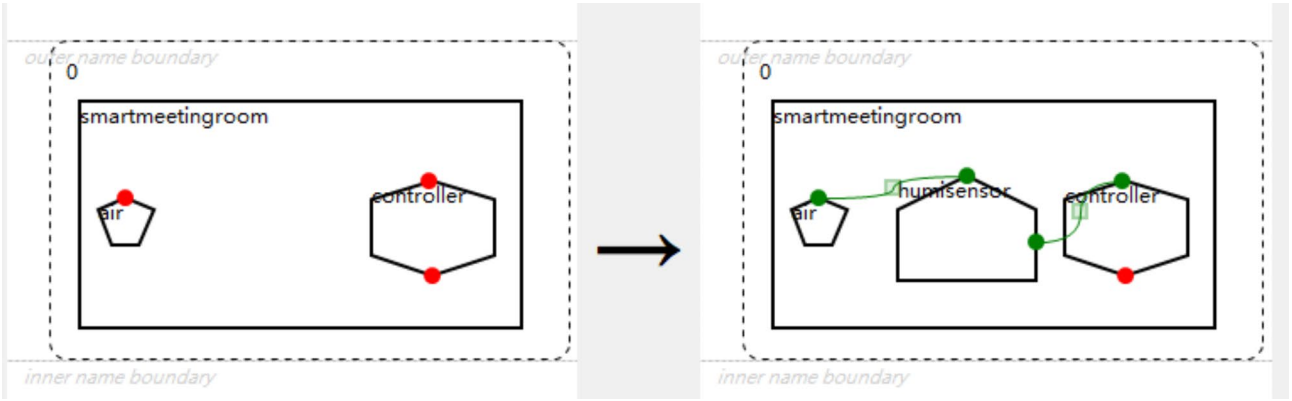

Fig. 20. Add evolution rule for humidity sensor r11.

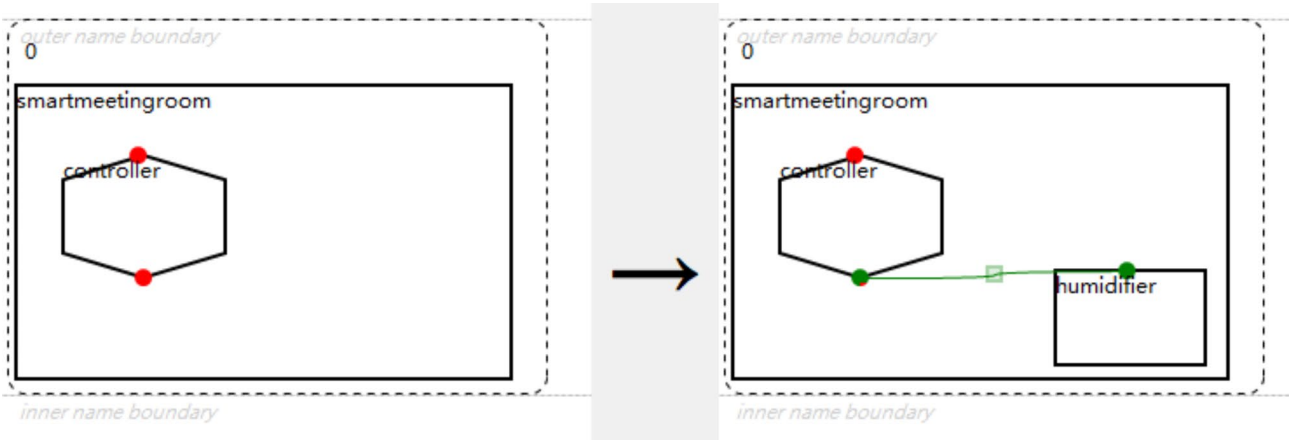

Fig. 21. Add evolution rule for humidifier r12.

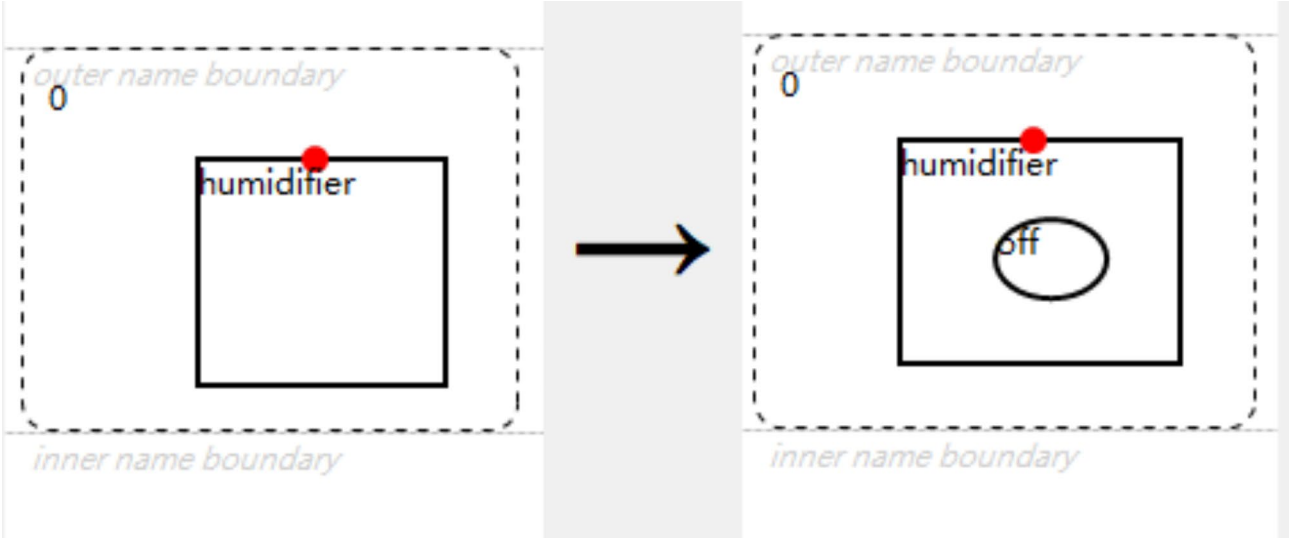

Fig. 22. Add evolution rule for initial state of humidifier r13.

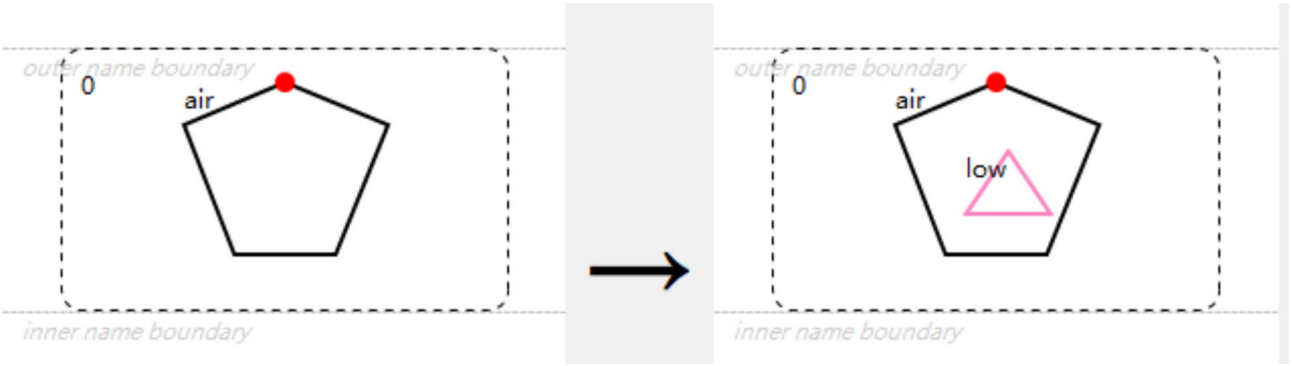

Fig. 23. Add evolution rule for initial state of air humidity r14.

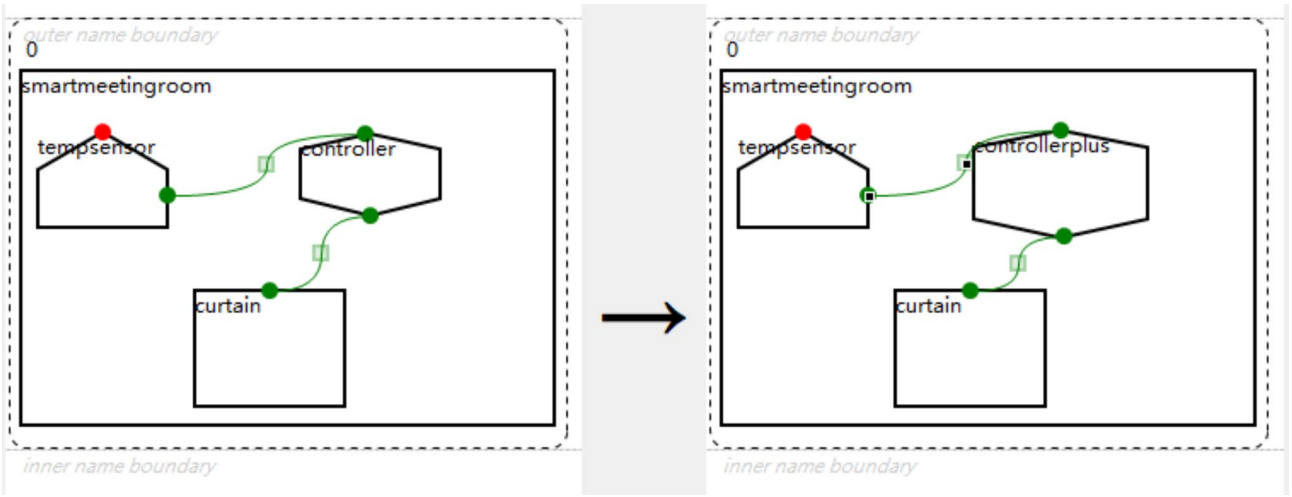

Fig. 24. Replace controller evolution rule r15.

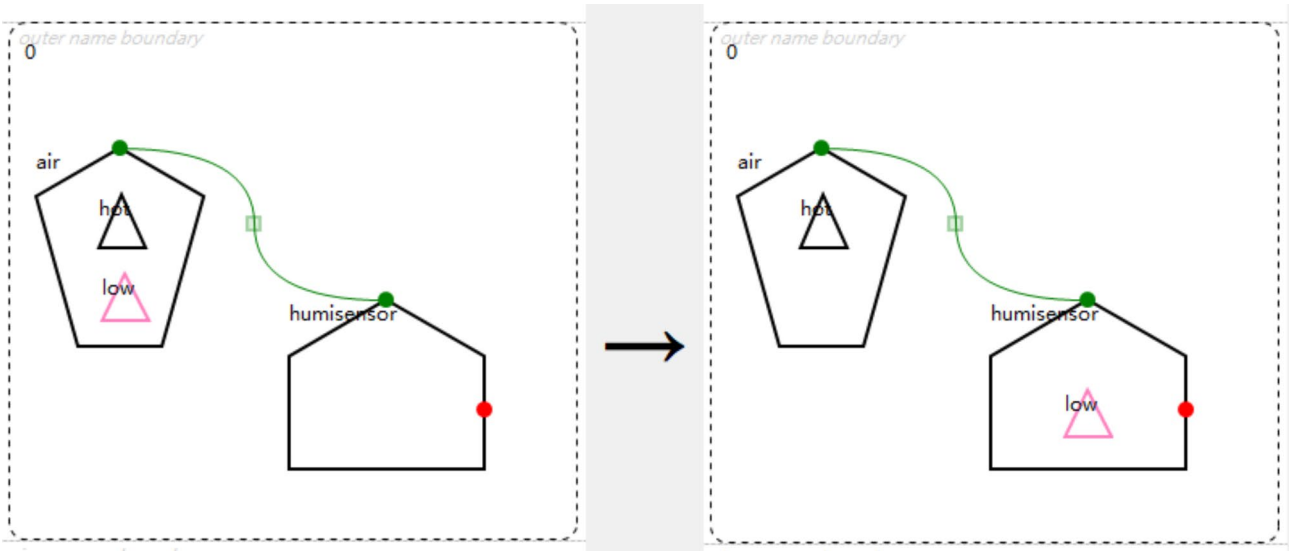

Fig. 25. Evolution rule for humidity sensor perception and acquisition of humidity information flow r16.

r12:smartmeetingroom(controller$_{yz}$) → e3/$_{z,\ w}$.smartmeetingroom(controller$_{yz}$|humidifier$_{w}$).
r13: humidifier$_{w}$ → humidifier$_{w}$(off).
r14: air$_{o}$ → air$_{o}$(low).
r15 :e4/$_{p,\ y}$.e5/$_{z,\ q}$.smartmeetingroom(tempsensor$_{p}$|controller$_{yz}$|curtain$_{q}$) → e6/$_{p,\ r}$.e7/$_{s,\ q}$.smartmeetingroom(tempsensor$_{p}$|controllerplus$_{rs}$|curtain$_{q}$).
r16: e6/$_{o,\ u}$.air$_{o}$(hot|low)|humisensor$_{uv}$ → e6/$_{o,\ u}$.air$_{o}$(hot)| humi sensor$_{uv}$(low).

It can be seen that our method not only has an intuitive graphical expression of evolution rules, but also has a strict formalized algebraic semantic expression of evolution rules.

Rule-driven system function evolution

In order to achieve the evolution goal and meet the user's evolution requirements, a software engineer first used the evolution rule *r11* to add the *humisensor*, then used r12 to add the *humidifier*, then used r13 to set the *humidifier* to the initial state *off*, then used r14 to add the initial humidity state *low* in the *air*, and finally used r15 to replace the *controller* with a more powerful *controllerplus*. The final evolution result is shown in Fig. 26. As can be seen from Fig. 26, the original functions and function logic of the software system are not affected, only the automatic humidification function is extended, which meets the user's evolution requirements. This shows that as long as a clear user evolution requirement goal is given, it is always possible to use a finite number of evolution rules to achieve the target requirement, which proves the correctness of *Proposition 1* from the actual system. In order to ensure the reliability of the system after evolution, it is also necessary to further check the consistency, integrity and reachability of the system after evolution, which will be discussed later.

Rule-driven system information flow evolution

Rule *r16* can express that the humidity *low* in the air is perceived and acquired by the *humisensor*. The Bigraph model of the system evolution after applying the rule is shown in Fig. 27. This intuitive way of expression is conducive to the visualization and understanding of the model. The Bigraph model also has a rigorous mathematical formal semantics, which can be used for the global representation of system states.

Reliability checking of system evolution

(1) Consistency checking.

Figure 28 shows the Bigraph model of the smart meeting system after evolution using the evolution rules by the software engineer, which leads to the lack of connection between *humisensor* and *controllerplus* due to negligence. The algorithms CSC1, CSC2, and CSC3 are used to check the consistency of the evolved Bigraph

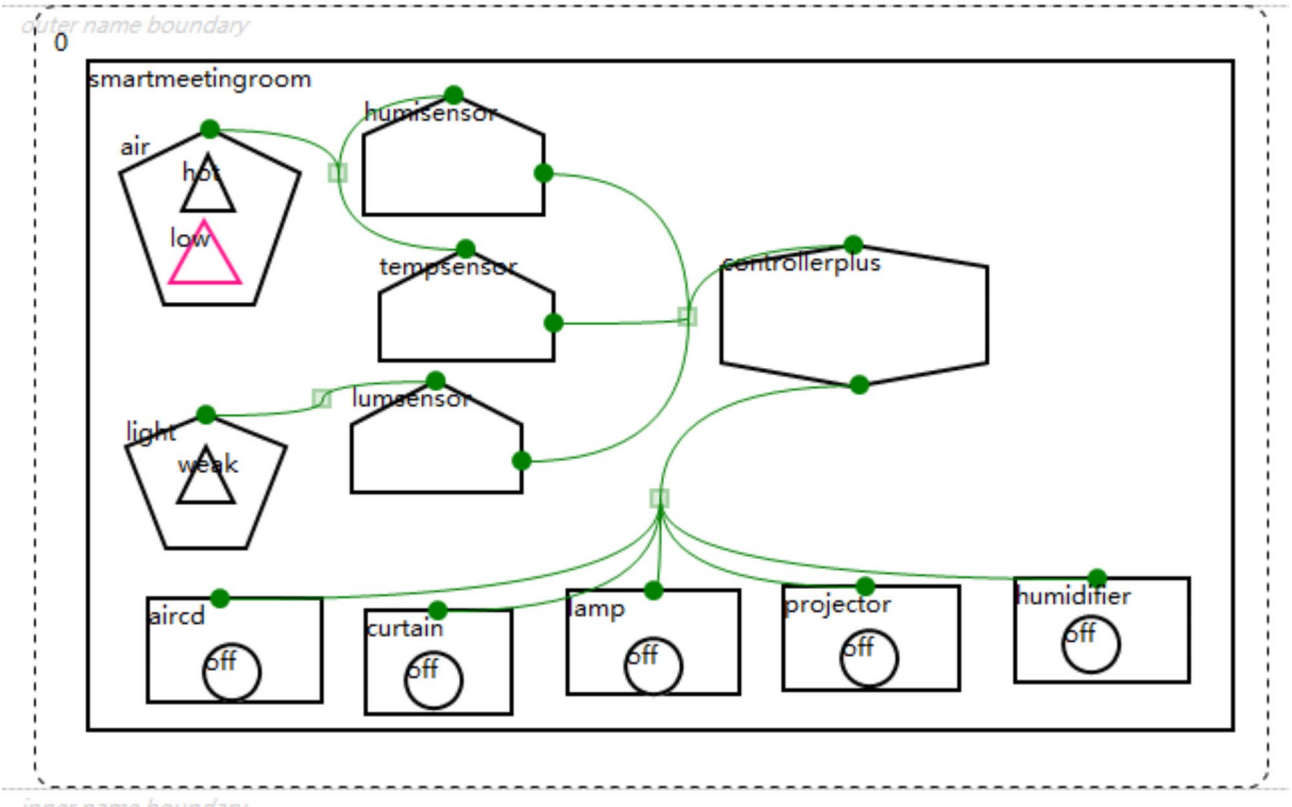

Fig. 26. Bigraph model of smart meeting system after evolution.

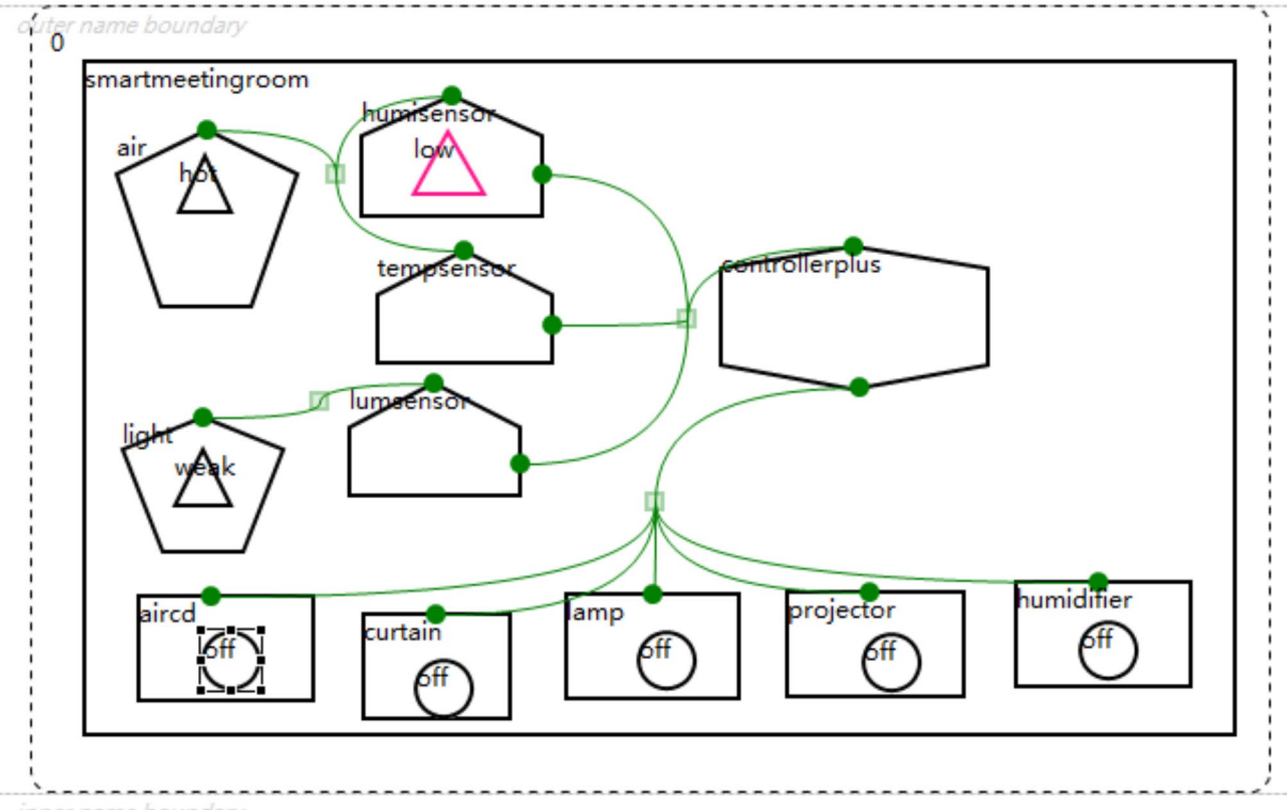

Fig. 27. Bigraph model of smart meeting system after information flow evolution.

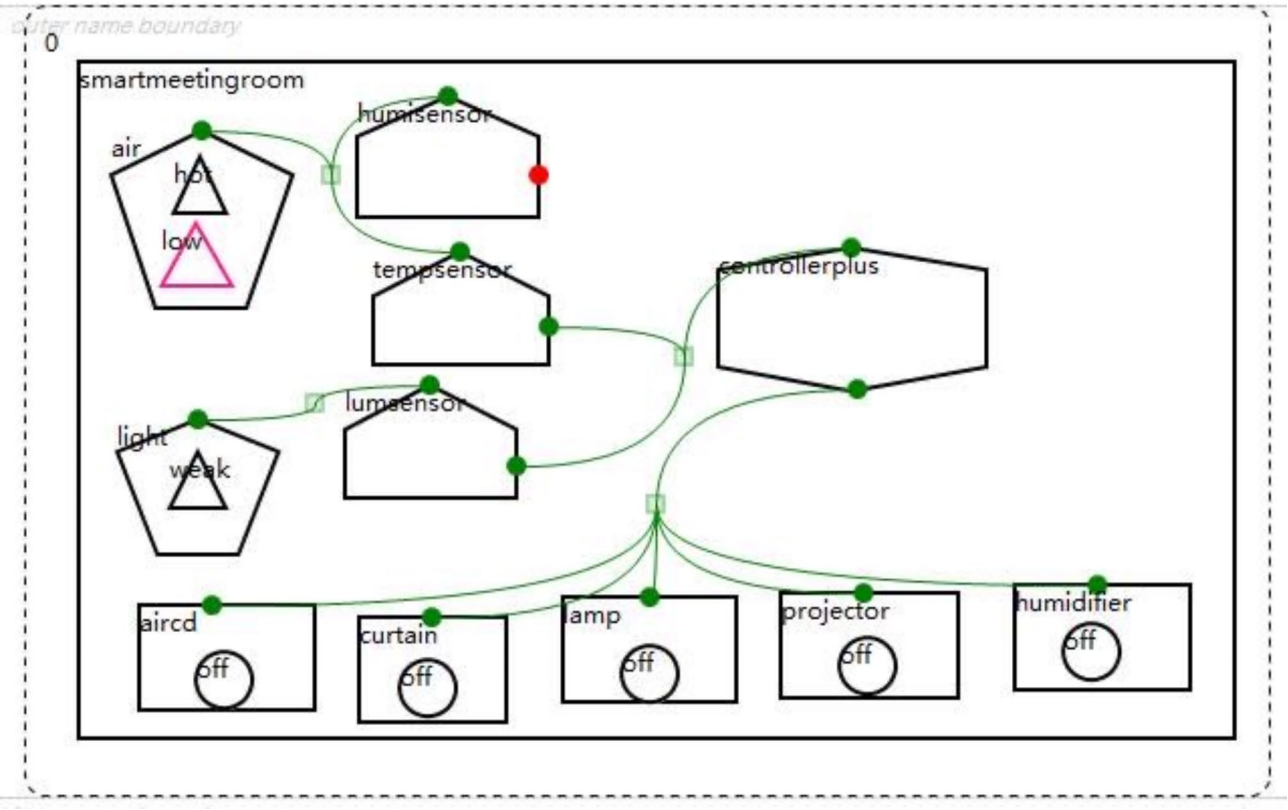

Fig. 28. the Bigraph model with lacking the connection after evolution.

Interval	Counts									
	20	40	60	80	100	120	140	160	180	200
1	100.40	200.87	514.34	786.61	799.98	989.11	1163.05	1200.10	2200.31	2399.78
2	103.73	300.05	748.55	796.02	999.91	983.39	1210.51	1499.78	1806.12	1900.26
3	151.89	400.76	612.64	627.80	900.02	985.78	1283.61	1500.42	1337.03	2275.79
4	163.79	299.82	699.88	900.03	956.13	1014.12	1157.87	1500.44	1574.29	1820.57
5	305.28	500.90	799.71	799.98	1207.57	1345.51	1308.00	1500.05	2046.02	2105.02
Average time (ns)	165.02	340.48	675.02	782.09	972.72	1063.58	1224.61	1440.16	1792.75	2100.28

Table 5. The average checking time of the CSC1 consistency algorithm under different entity sizes.

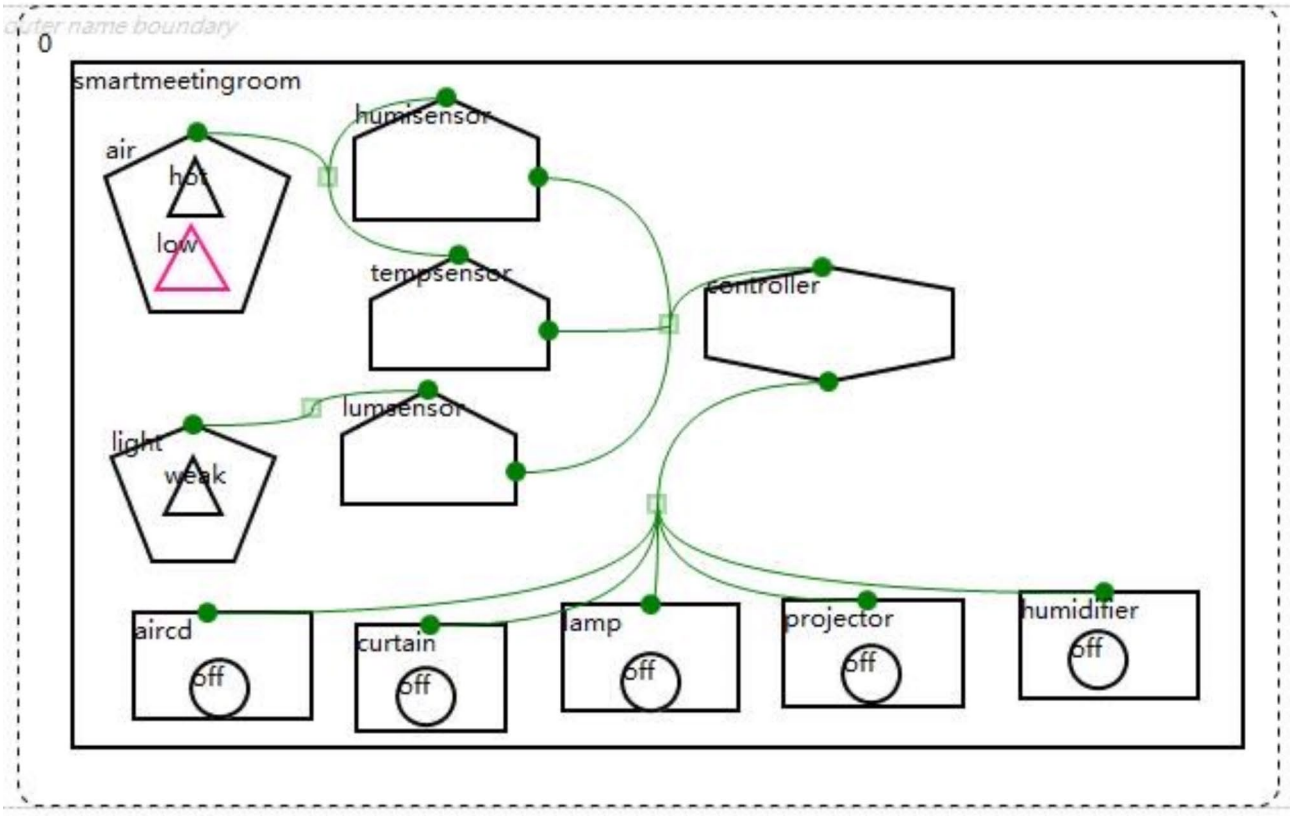

Fig. 29. Bigraph model with integrity errors after evolution.

model, respectively, and can detect that the lack of connection between *humisensor* and *controllerplus* causes the inconsistency of the system. This also illustrates the necessity of consistency checking on the one hand, and verifies the correctness of our algorithm design on the other hand.

In order to verify the detection efficiency of the consistency detection algorithm, five equal-interval inconsistency test points were set up as the number of entities in the smart meeting system model increased, and the average detection time of these test points was calculated and used to measure the detection time at a certain scale. Firstly, the CSC1 algorithm are used for test point detection, and the corresponding average detection time is shown in Table 5. By observing the average detection time, it can be seen that the average detection time generally increases with the increase in scale, but it is not proportional to the number of scales. This also reflects that as the number of entities increases, the relationship between entities becomes more complex and requires more detection time, but overall, it is at the nanosecond level, and the detection time efficiency is very high. These detection times are relatively small compared to the time spent on system evolution, so it is feasible to add consistency detection during system evolution, which can also ensure consistent system evolution. The detection time of the CSC2 algorithm under equivalent conditions is approximately twice that of the CSC1 algorithm. When the influence range of the CSC3 algorithm is 1/k of the overall range, the detection time of the CSC3 algorithm is approximately 1/k of that of the CSC1 algorithm. Since the detection methods are similar, they will not be repeated.

(2) Integrity checking.

In the integrity check, the assumption is made that the old controller can connect up to two sensors, and the Bigraph model of the smart meeting system after evolution is shown in Fig. 29. The model is the software engineer using the evolution rules to add *humisensor* and *humidifier*, and forgetting to replace the old *controller* with *controllerplus*. The detection algorithm CAI is used to perform integrity check on this model, and can detect that *controller* connects three sensors: *lumsensor*, *tempsensor*, and *humisensor*, which violates the integrity constraint that the old *controller* can connect up to two sensors. Through such an integrity check, the satisfaction of the system model with relevant constraints and requirements is ensured, thereby further guaranteeing the system's reliability.

To evaluate the efficiency of algorithm CAI for integrity checking, we design the smart meeting system to record the time consumption of detecting the controller with integrity constraint errors under different entity numbers. As shown in Table 6, the total detection time does not increase proportionally when the entity number doubles. The main reason is that the controller is in the middle position of the model, and the time to search for the controller entity is similar, and the time difference mainly relies on the subsequent judgment of the number of other entities connected by the controller. The experimental results indicate that the detection time of integrity constraints is at the nanosecond (ns) level, which is very low compared to the other operations of the whole evolution. From the perspective of model reliability, it is worthwhile to perform corresponding integrity check after evolution, and the cost is very small.

(3) Reachability checking.

In the reachability check, assuming that the humidifier is added by using the evolution rule, but it is not connected to *controllerplus* due to negligence, and the Bigraph model of the smart meeting system after evolution is as shown in Fig. 30. The checking algorithm CAR are applied to perform reachability check on this model from two environment entities air and light, and detect that humidifier is not connected to *controllerplus*, which violates the reachability constraint. Through such reachability check, it can ensure that the system model meets the constraints of relevant requirements, and thus further enhance the reliability of the system.

To evaluate the efficiency of algorithm CAR for reachability checking, the smart meeting system to measure the time consumption of detecting the reachability of all entities under different entity numbers are designed. As shown in Table 7, the total detection time increases linearly when the entity number doubles. The main reason is that the reachability check has to traverse all paths, and the increase of entity number leads to almost linear growth of reachable paths. The experimental results show that the detection time of reachability constraints is at the nanosecond (ns) level, which is very low compared to the other operations of the whole evolution. From the perspective of model reliability, it is worthwhile to perform corresponding reachability check after evolution, and the cost is very small.

The above experimental analyses shows that our proposed modeling method can effectively address the evolution modeling problem of cyber-physical system, and can provide graphical and algebraic representations of the evolution process of cyber-physical system, laying a solid mathematical semantic foundation for the evolution analysis of the system.

Experiment 2: evolution modeling and analysis for internet of vehicles system

Modeling for internet of vehicles system

Internet of vehicles system is a typical application of cyber-physical system, which is an important carrier of the next generation intelligent vehicle system. It consists of various vehicular systems that are organized in a distributed way. Each vehicular system is composed of a series of functional components, and they communicate with each other mainly through wireless communication. The system involves many entities, and the basic control set of the Internet of vehicles system are presented, as shown in Table 8. Based on this, the Bigraph modeling tool BigRed to used to construct the initial Bigraph model of the Internet of vehicles system, as shown in Fig. 31.

Figure 31 shows an Internet of vehicles system composed of two cars, which is mainly formed by the cyber-physical vehicular systems of each car through a wireless signal network in a distributed way. Each cyber-physical vehicular system mainly consists of environmental place entity road, and environmental entities barrier, people and car. The environmental entity car includes roadsensor, barriersensor, personsensor, carsensor, controller, accelerator, brake, and warning. And the environmental entities car communicate with each other through wireless network entity signaltower. The connection relationships of these entities are shown in Fig. 31. The sensor connects to both the corresponding perceived entities road, barrier, people and car, and the global controller. The controller connects to all the control objects accelerator, brake, warning.

Entity	Counts									
	20	40	60	80	100	120	140	160	180	200
Lumsensor	207.88	318.27	513.08	699.90	844.97	1098.36	1204.61	1502.57	1602.11	1813.60
Tempsensor	174.25	404.13	605.25	700.91	1030.79	1109.48	1400.02	1499.99	1706.78	1900.03
Humisensor	165.29	302.56	499.98	700.01	917.83	1108.60	1197.02	1500.02	1608.34	1808.03
Controller	159.59	336.08	504.02	601.11	830.40	1011.97	1106.12	1317.46	1603.40	1601.30
Total tims (ns)	707.01	1361.04	2122.33	2701.93	3623.99	4328.41	4907.77	5820.04	6520.63	7122.96

Table 6. The total checking time of the CAI integrity algorithm under different entity sizes.

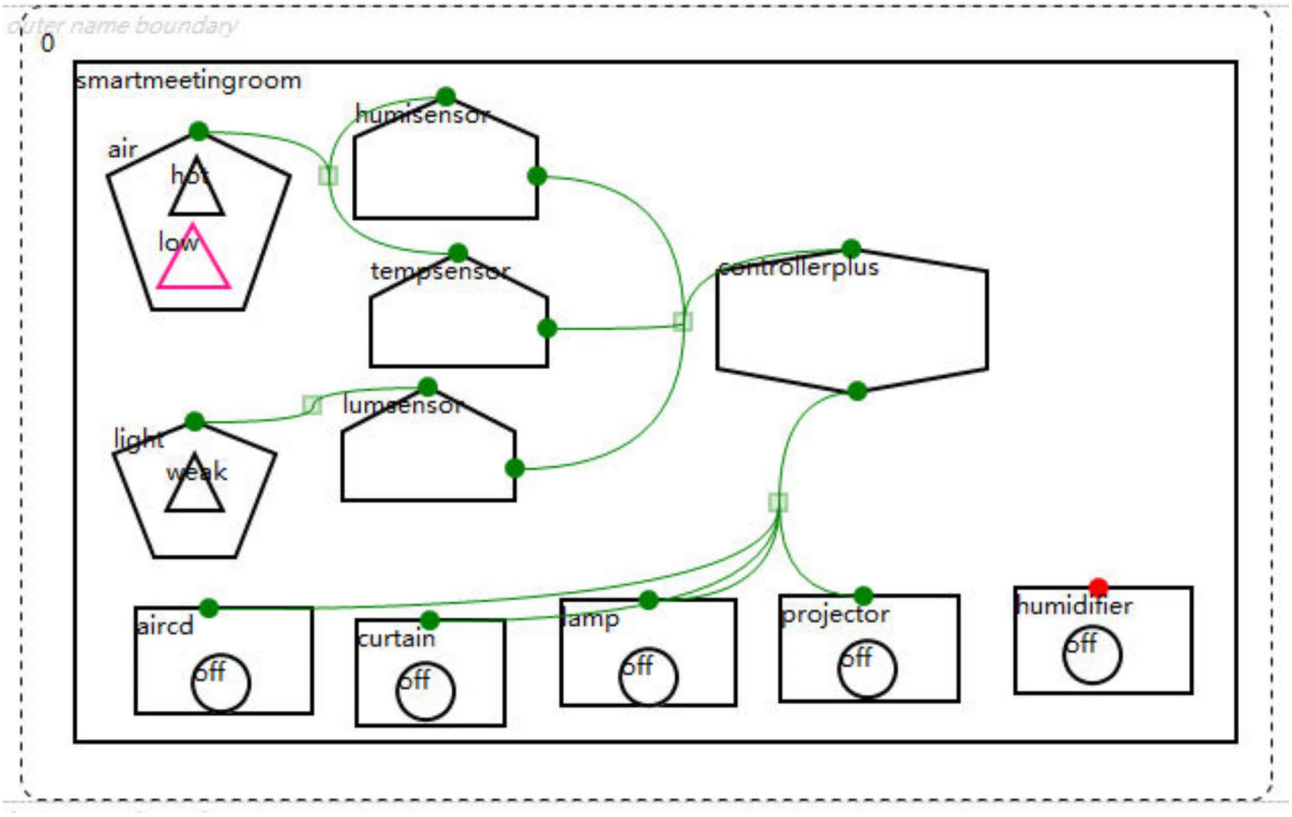

Fig. 30. Bigraph model with reachability errors after evolution.

Entity	Counts									
	20	40	60	80	100	120	140	160	180	200
Air	172.18	308.52	511.56	800.01	832.08	1012.06	1206.02	1402.46	1496.02	1703.17
Humisensor	149.30	406.81	500.01	704.46	900.79	1104.39	1200.67	1404.11	1701.02	1712.02
Controllerplus	166.12	301.99	410.88	600.17	811.73	909.30	1200.20	1400.02	1408.02	1619.33
Humidifer	98.25	403.34	416.47	800.55	827.89	901.44	1100.02	1326.02	1809.04	1590.42
Total time (ns)	585.85	1420.66	1838.92	2905.19	3372.49	3927.19	4706.91	5532.61	6414.1	6624.94

Table 7. The total checking time of the CAR reachability algorithm under different entity sizes.

Evolution rule

The primary goal of vehicular networking is to connect a car to the system, so the first step is to design a networking evolution rule for a car. Secondly, since the vehicular network system currently lacks visualization of various sensor data, people want to equip each car in the system with a visual display, so a functional component entity displayer for visualization is also required. In addition, since the original controller cannot control the functional component displayer, it needs to be replaced by a controller1 that can support this function. Moreover, since the displayer functional component has a warning function already, the warning functional component warning needs to be removed to avoid redundancy. Therefore, the following evolution rules need to be designed, as shown in Fig. 32 for a car networking evolution rule, as shown in Fig. 33 for an add displayer evolution rule, as shown in Fig. 34 for a replace controller evolution rule, and as shown in Fig. 35 for a delete warning evolution rule. Due to the limited space, the term language expression forms of the evolution rules are not given here, and their expression forms are the same as those of experiment 1.

(1) A car networking evolution rule.

(2) Add displayer evolution rule.

(3) Replace controller evolution rule.

(4) Delete warning evolution rule.

Control	Attribute	Description
AR	Active	Accelerator
BK	Active	Brake
BR	Active	Barrier
BS	Active	Barrier sensor
CAR	Active	Car
CR	Active	Controller
CS	Active	Car sensor
HAVEB	Atomic	Barrier information
NONEB	Atomic	People information
PE	Active	People
PS	Active	People sensor
RD	Active	Road
RS	Active	Road sensor
ST	Active	Signal tower
WR	Active	Warning

Table 8. Control set for internet of vehicles system.

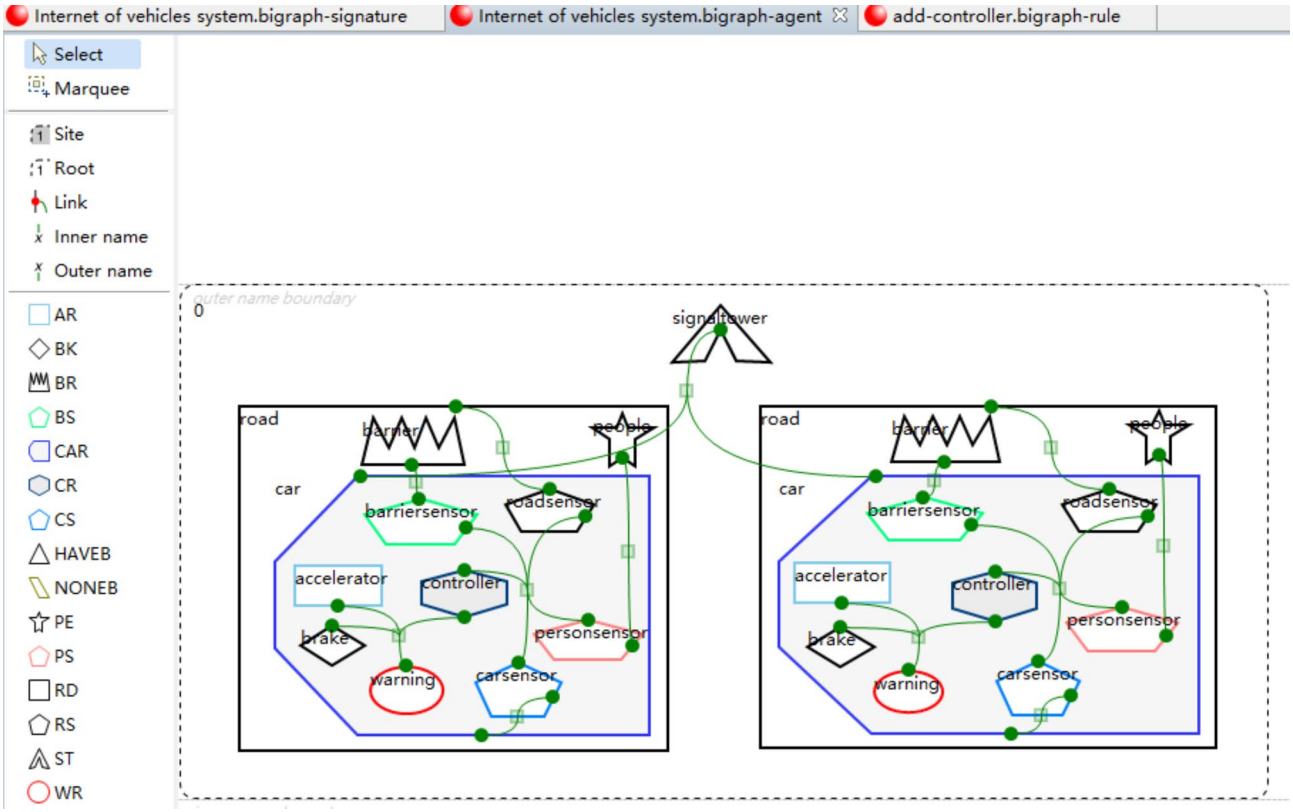

Fig. 31. Initial Bigraph model of Internet of vehicles system.

System evolution

In order to achieve the evolution goal of adding a car to the current system, and adding a displayer component, replacing a stronger controller component, and deleting a redundant warning component. A software engineer first uses the evolution rule r21 to add a car to the system, then uses r22 to globally add the displayer component, then uses r23 to globally replace the old controller component with the new component controller1, and finally uses r24 to delete the redundant warning component. The final evolution result is shown in Fig. 36. From Fig. 36, it can be seen that the original functions and function logic of the vehicular network system are not affected, but only one car is added to the network, and the displayer function is added, the stronger controller1 is replaced, and the redundant warning function is deleted, which meets the user's evolution requirements. This shows that as long as a clear user requirement goal is given, it can always be achieved with a finite number of evolution rules,

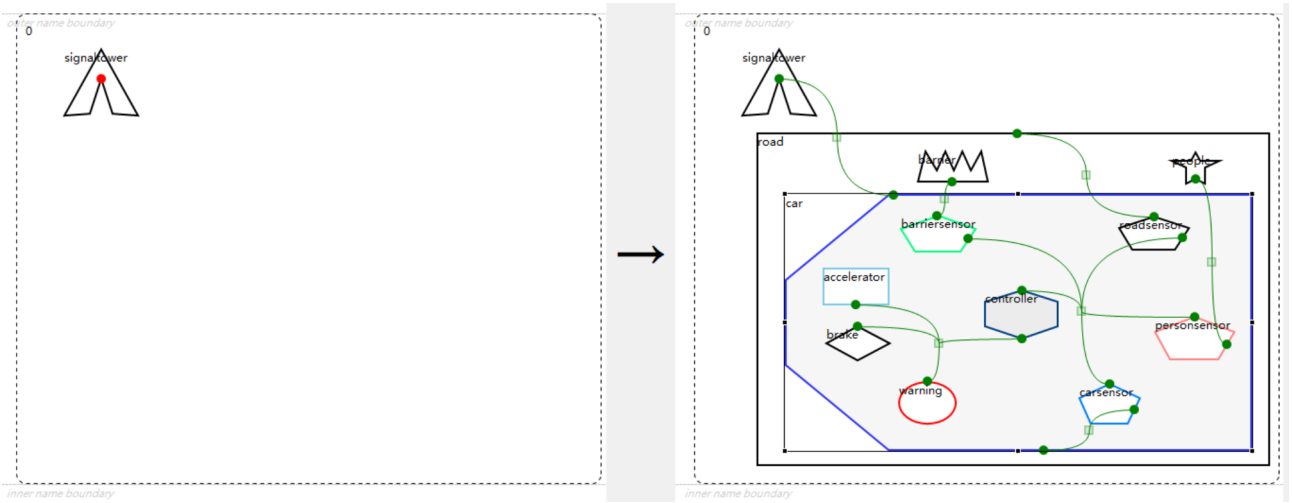

Fig. 32. A car networking evolution rule *r21*.

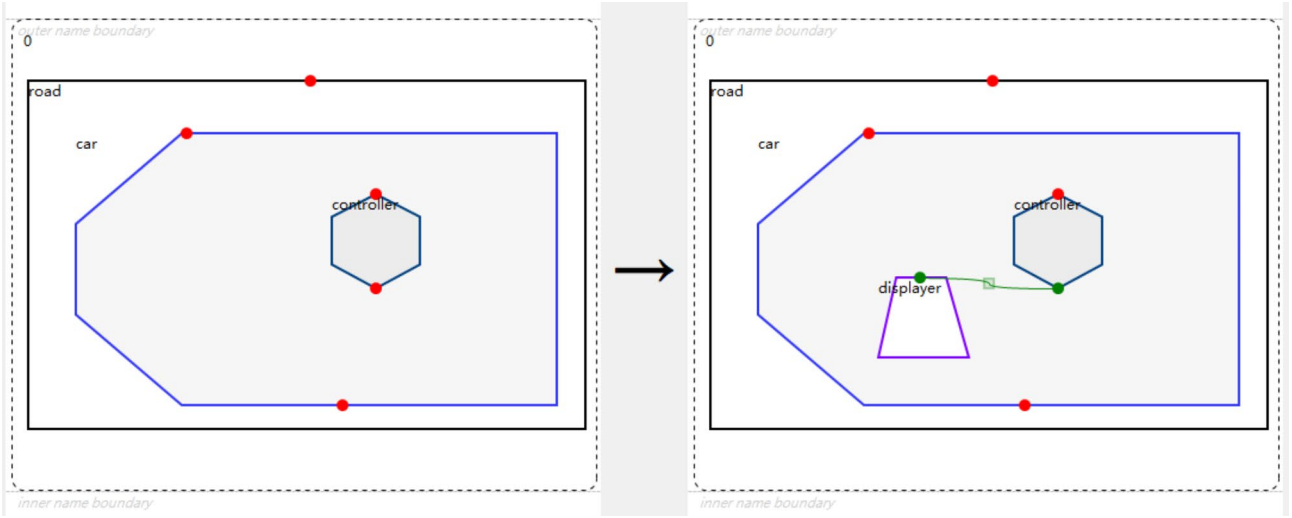

Fig. 33. Add displayer evolution rule *r22*.

which proves the correctness of Proposition 1 from the actual system again. After evolution, further verification of the system's consistency, integrity and reachability is needed, which will be discussed later.

Performance of checking algorithm

To verify the checking efficiency of the three algorithms CSC1, CAI, and CAR, the time consumption of checking all entities for consistency, integrity, and reachability after evolution with different entity numbers are measured. Similar to Experiment 1, the detection times corresponding to the three algorithms under the scenario of increasing entity counts in Tables 9 and 10, and 11 are presented, respectively. Based on the experimental results, as the scale of entities continues to increase, the detection times for all three algorithms also show an upward trend. Overall, the detection time for consistency is at the nanosecond level, while the detection times for integrity and reachability are both in the tens of microseconds. Compared to the minute-level durations of other operations in the entire evolution process, these detection times remain relatively low. From the perspective of model reliability, it is absolutely necessary to conduct corresponding checks for consistency, integrity, and reachability after evolution, and the cost involved is minimal.

Discussion

Through the analysis of the above two experiments, it can be seen that the Bigraph modeling method can effectively address the modeling issue of both the connectivity and positional relationships of entities in the evolution of cyber-physical systems (CPS). Compared to traditional single-dimensional modeling approaches such as graphs and hypergraphs, it better aligns with the two-dimensional spatial modeling of CPS. Due to the difference in dimensions, it is not feasible to directly compare our method's performance in terms of consistency,

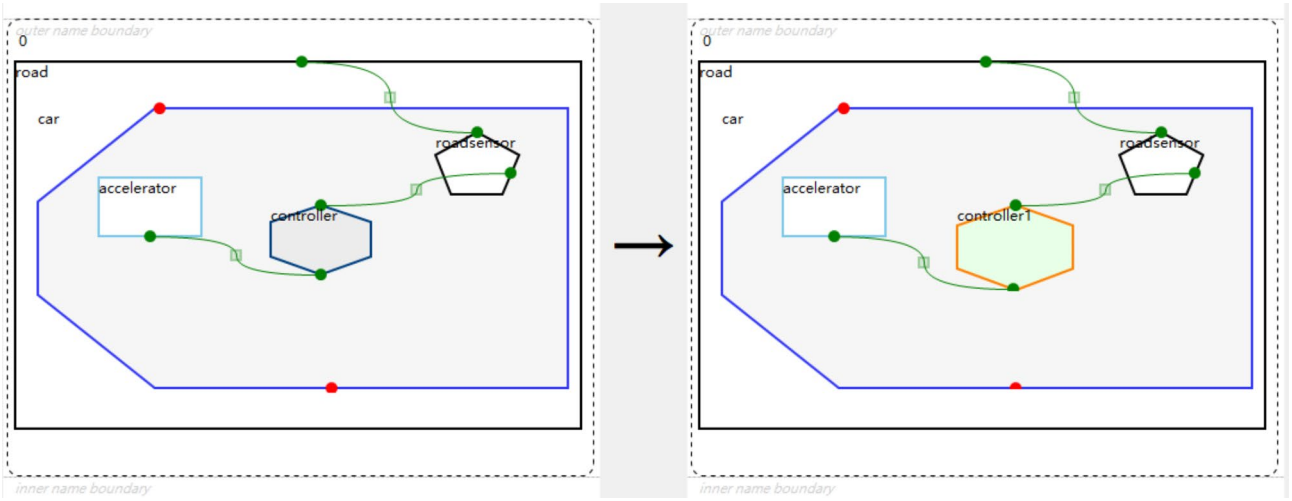

Fig. 34. Replace controller evolution rule *r23*.

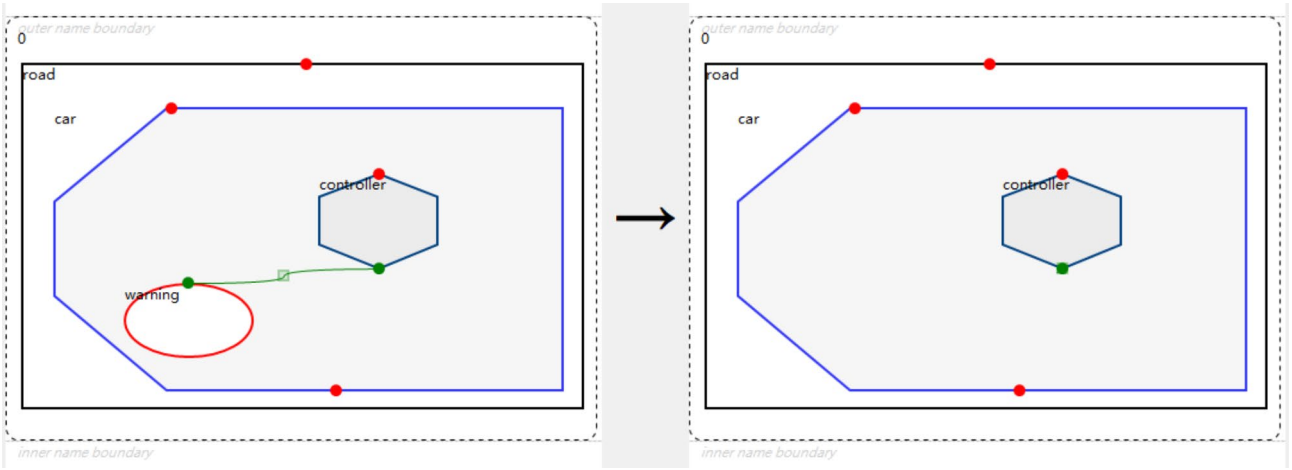

Fig. 35. Delete warning evolution rule *r24*.

integrity, and reachability with traditional single-dimensional modeling methods like graphs and hypergraphs. However, based on the performance analysis results from Experiment 1 and Experiment 2, when the scale of entities reaches 370, the most time-consuming integrity check only takes 30,589 nanoseconds, which is negligible compared to the minute-level duration of evolution operations. It is evident that our method can be extended to larger-scale CPS.

Conclusion

Oriented towards the need for a rigorous mathematical semantic description of the update and evolution processes of cyber-physical systems (CPS), this paper proposes a modeling method for the evolution of CPS architecture based on Bigraph. Essentially, this research not only presents an effective method for CPS modeling, but also provides a solution to the problem of simultaneously modeling connections and locations. The innovative aspects of this research mainly consist of the following three parts:

(1) The Bigraph model for Cyber-Physical Systems (CPS) proposed in this study can not only represent one-to-one relationships between entities within CPS but also accommodate one-to-many and many-to-many relationships, thereby fulfilling the need to express complex relationships among entities in the cyberspace.
(2) A set of dynamic evolution rules tailored for CPS architecture has been designed, and a model for the dynamic evolution of CPS structure and information flow has been proposed, drawing on the concepts of conditional matching and state transition.
(3) Algorithms for checking consistency, integrity, and reachability constraints in the dynamic evolution of CPS architecture have been designed, thereby ensuring the correctness and reliability of the CPS system after its dynamic evolution.

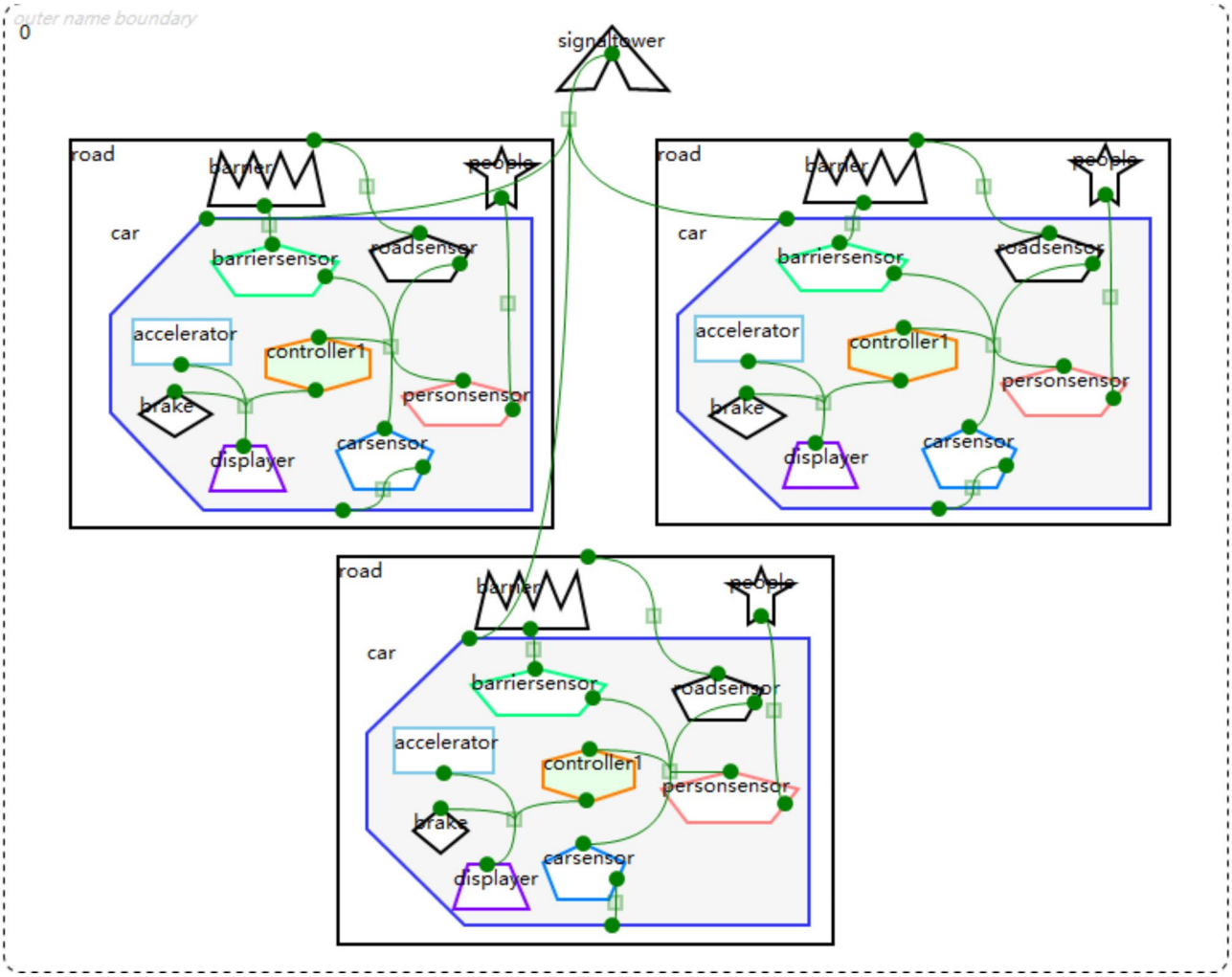

Fig. 36. Bigraph model of Internet of vehicles system after evolution.

Interval	Counts									
	37	74	111	148	185	222	259	296	333	370
1	593.70	1628.43	2372.60	2667.59	3392.19	4557.78	5332.97	6570.25	6829.31	7302.58
2	399.99	1357.24	1392.37	1296.32	1728.79	2727.47	3092.06	3500.22	3664.01	3810.02
3	300.11	1013.30	1359.73	1503.38	1883.78	3008.25	2987.34	3100.01	3292.82	3716.48
4	400.05	699.90	1081.97	1265.89	2214.27	2846.35	2430.04	2956.94	2899.94	3133.98
5	199.94	600.36	1183.58	1407.32	2083.56	2485.11	2613.06	2900.08	3100.04	3205.54
6	400.84	703.81	1083.43	1300.17	1974.52	2525.81	2590.68	3300.10	2600.09	3106.92
7	200.76	581.31	1297.06	1400.49	2106.53	2375.75	2840.31	3000.21	3177.15	3392.15
Average time (ns)	356.48	940.62	1395.82	1548.74	2197.66	2932.36	3126.64	3618.26	3651.91	3952.52

Table 9. The average checking time of the CSC1 consistency algorithm under different entity sizes.

However, the research in this paper has some limitations.

(1) Our current work focuses more on the evolution of system entities and does not address the issue of whether entity evolution is safe. For instance, if a particular entity is a crucial part of the system, evolving it may affect and propagate changes to other entities. The scope of its impact needs to be assessed, and an evaluation of evolution safety is required. Such evolution safety assessment can be further analyzed using our proposed Bigraph model. For example, the link graph of Bigraph can be used for safety assessment analysis of the impact scope of entity relationships, while the place graph can be used for safety scope analysis of the physical location impact of entities.

Entity	Counts									
	37	74	111	148	185	222	259	296	333	370
Road	900.77	1401.34	2142.60	3002.12	3026.09	5000.02	5005.57	5028.09	5704.85	7455.01
Controller	500.01	900.01	851.52	2153.58	2000.02	3000.01	3000.01	4000.08	4048.49	4486.41
Signaltower	603.73	900.01	2592.65	1914.14	2000.02	3000.03	4002.57	4027.74	3618.73	5967.28
Barriersensor	600.84	810.52	565.44	1559.84	2000.02	3000.04	2997.55	3002.45	4206.59	4027.31
Total time (ns)	2605.35	4011.88	6152.21	8629.68	9026.15	14000.1	15005.7	16058.36	17578.66	21936.01

Table 10. The total checking time of the CAI integrity algorithm under different entity sizes.

Entity	Counts									
	37	74	111	148	185	222	259	296	333	370
Road	999.18	1400.00	1862.43	3289.39	4000.05	5000.07	5000.06	5005.10	6113.22	8266.55
Barriersensor	500.00	800.01	1461.66	1685.41	2000.03	3000.04	2000.06	2966.17	4021.35	3941.56
Controller	501.63	900.01	1793.64	1629.86	3000.04	2190.95	2980.90	4000.05	4080.38	4420.52
Displayer	500.01	900.01	1591.97	1822.09	2000.03	3002.86	3000.05	4000.06	4794.16	4595.86
Barrier	601.05	804.57	966.03	1145.55	3013.22	3033.86	2998.79	3000.04	4045.94	5098.19
People	598.97	802.69	499.24	1002.70	2056.77	2002.48	3000.00	3000.11	3635.24	4267.11
Total time (ns)	3700.84	5607.29	8174.97	10,575	16070.14	18230.26	18979.86	21971.53	26690.29	30589.79

Table 11. The total checking time of the CAR reachability algorithm under different entity sizes.

(2) Furthermore, there is a need to delve deeper into the scalability of Bigraph within Cyber-Physical Systems (CPS), such as its extension in modeling the evolution of large-scale unmanned aerial vehicle (UAV) swarm systems. Its ability to extend when modeling the evolution of large-scale cooperative unmanned vehicle systems. The expansion needs of different systems vary, and analysis is needed on whether Bigraph possesses the capability for adaptive expansion.
(3) The efficiency of detecting various properties during the evolution of the cyber-physical system Bigraph model also requires further research. Particularly in scenarios with non-stop conditions, if the detection time occupies a significant portion of the entire evolution time, it may affect the continuity of system service provision, making it difficult to meet the requirements of high-quality service-oriented cyber-physical systems.
(4) Further research is also needed on the Bigraph model for real-time cyber-physical systems. The Bigraph model proposed in this paper does not address temporal attributes. However, in real-world cyber-physical systems, there are many real-time scenarios where evolution might compromise the system's real-time performance. It is necessary to extend the Bigraph model to incorporate temporal attributes, enabling the analysis of these attributes and ensuring their satisfiability.

Therefore, our future work will focus more on the above four areas research of CPS evolution.

Data availability

All data generated or analyzed during this study are included in this published article.

Received: 11 September 2024; Accepted: 28 February 2025
Published online: 13 March 2025

References

1. Fischer, S., Klammer, C., Fernández, A. M. G., Rabiser, R. & Ramler, R. Testing of highly configurable cyber–physical systems-Results from a two-phase multiple case study. *J. Syst. Softw.* **199**, 1–19 (2023).
2. Alshalalfah, A., Mohamed, O. A. & Ouchani, S. A framework for modeling and analyzing cyber-physical systems using statistical model checking. *Inter Thin.* **22**, 1–24 (2023).
3. Ahmad, I., Islam, Z. U. & Riaz, S. Complexity analysis of industrial scale cyber physical systems. In *Proc. IEEE 7th Inter. Conf. on ICPS*, 1–6 (2024).
4. Chen, C., Liu, X., Qiu, T. & Sangaiah, A. K. A short-term traffic prediction model in the vehicular cyber-physical systems. *Future Gener. Comput. Syst.* **105**, 894–903 (2020).
5. Habib, A. A. et al. False data injection attack in smart grid cyber physical system: issues, challenges, and future direction. *Comp. Electr. Eng.* **107**, 1–16 (2023).
6. Kankanamge, M. W., Hasan, S. M., Shahid, A. R. & Yang, N. Large language model integrated healthcare cyber-physical systems architecture. In *Proc. IEEE 48th Ann. Comp. Soft. Appl. Conf.*, 1540–1541 (2024).
7. Keung, K. L. et al. A cyber-physical robotic mobile fulfillment system in smart manufacturing: the simulation aspect. *Rob. Comp. Integ. Manuf.* **83**, 1–14 (2023).
8. Chen, L., Hu, F., Wang, S. & Chen, J. Cyber-physical system fusion modeling and robustness evaluation. *Electr. Power Syst. Res.* **213**, 1–9 (2022).

9. Wang, S., Gu, X., Chen, J., Chen, C. & Huang, X. Robustness improvement strategy of cyber-physical systems with weak interdependency. *Rel. Eng. Syst. Sa*. **229**, 1–14 (2023).
10. Ivanov, B. I., Hotmar, A., Ivanov, I. E. & Georgieva, D. A software architecture selection for cyber-physical system. In *Proc. Inter. Sci. Conf. Comp. Sci.*, 1–5 (2023).
11. Vogel–Heuser, B. et al. Automation software architecture in CPPS-definition, challenges and research potentials. In *Proc. IEEE 5th Inter. Conf. ICPS*, 1–8 (2022).
12. Dotty, F. A. & Asnar, Y. Intrusion detection system architecture for cyber-physical system. In *Proc. Inter. Conf. Elec. Eng. Inf.*, 1–6 (2023).
13. Ferchichi, O., Beltaifa, R. & Jilani, L. L. An ontological rule-based approach for software product lines evolution. In *Proc. Organ. Know. Advan. Tech.*, 1–9 (2020).
14. Chaturvedi, A. & Tiwari, A. System evolution analytics: evolution and change pattern mining of inter-connected entities. In *Proc. IEEE Inter. Con. on Syst. Man. Cyber.*, 3877–3882 (2018).
15. Berrio-Charry, E., Vergara-Vargas, J. & Umana-Acosta, H. A component-based evolution model for service-based software architectures. In *Proc. IEEE 11th Inter. Conf. Softw. Engi. Ser. Scie.*, 1–5 (2020).
16. Chaturvedi, A., Tiwari, A., Binkley, D. & Chaturvedi, S. Service evolution analytics: change and evolution mining of a distributed system. *IEEE Trans. Eng. Manag*. **68** (1), 137–148 (2021).
17. Chaturvedi, A., Tiwari, A. & Spyratos, N. MinStab: stable network evolution rule mining for system changeability analysis. *IEEE Trans. Emerg. Top. Comp. Intel*. **5** (2), 274–283 (2021).
18. Haitzer, T., Navarro, E. & Zdun, U. Reconciling software architecture and source code in support of software evolution. *J. Syst. Softw.* **123** (1), 119–144 (2017).
19. Ivers, J., Seifried, C. & Ozkaya, I. Untangling the knot: enabling architecture evolution with search-based refactoring. In *Proc. IEEE 19th Inter. Conf. on Softw. Arch.*, 101–111 (2022).
20. Zhong, L., Ye, H. & Xia, J. Research on software evolution reconstruction based on architecture recovery. In *Proc. IEEE 9th Inter. Conf. on Softw. Engi. and Ser. Scie.*, 68–71 (2018).
21. Chaturvedi, A., Tiwari, A., Chaturvedi, S. & Liò, P. System neural network: evolution and change based structure learning. *IEEE Trans. Artif. Intell*. **3** (3), 426–435 (2022).
22. Ali, S. M., Doolan, M., Wernick, P. & Wakelam, E. Developing an agent-based simulation model of software evolution. *Inf. Softw. Tech.* **96** (4), 126–140 (2018).
23. Cui, L. & Zhang, J. Design and implementation of a cloud-based software platform for dynamically reconfigurable wearable computers. *Asia-Eur. Conf. Electr. Dat. Proc. Inf.*, 507–512 (2023).
24. Chaturvedi, A. Call graph evolution analytics over a version series of an evolving software system. In *Proc. IEEE/ACM 37th Inter. Conf. Auto. Soft. Eng.*, 1–5 (2023).
25. Zhang, L. Specification and design of cyber physical systems based on system of systems engineering approach. In *Proc.The 17th Inter. Symp. on Distri. Comp. and Appl. for Busi. Engi. and Scie.*, 300–303 (2018).
26. Ionita, A. D., Meier, J., Schönberg, C. & Winter, A. Analysis of viewpoints for modeling cyber-physical systems. In *Proc. 24th Inter. Conf. Cont. Syst. Comp. Sci.*, 319–324 (2023).
27. Grobelna, I. Model checking of concurrency in cyber-physical systems specified with interpreted petri nets*. In *Proc. 23rd Inter. Symp. INFOTEH*, 1–4 (2024).
28. Morozov, D., Lezoche, M. & Panetto, H. Multi-paradigm modelling of cyber-physical systems. *IFAC-Pap OnL*. **51** (11), 1385–1390 (2018).
29. Lehnert, C., Engel, G. & Greiner, T. A hierarchical domain-specific language supporting variants of CPPS software and integrating asset administration shells. In *Proc. IEEE 17th Conference on Industrial Electronics and Applications*, 572–577 (2022).
30. Haubeck, C., Pokahr, A., Lamersdorf, W., Chakraborty, A. & Ladiges, J. A. Fay. Evolution of cyber-physical production systems supported by community-enabled experiences. In *Proc. IEEE 15th Ind. Conf. of Ind. Infor., Emden*, 867–874 (2017).
31. Yu, Z., Zhou, L., Ma, Z. & Meligy, M. A. E. Trustworthi-Ness modeling and analysis of cyber-physical manufacturing systems. *IEEE Access*. **5**, 26076–26085 (2017).
32. Dong, Z., Tian, M. & Ding, L. A framework for modeling and structural vulnerability analysis of Spatial cyber-physical power systems from an attack–defense perspective. *IEEE Syst. J.* **15** (1), 1369–1380 (2021).
33. Gartziandia, A. Microservice-based performance problem detection in cyber-physical system software update. In *Proc. IEEE/ACM 43rd Inter. Conf. on Softw. Engi.*, 147–149 (2021).
34. Houssem, G. & Janis, K. Sebastian V. M: Extension of contracts for variability modeling and incremental update checks of cyber physical systems. In *Proc. IEEE Inter. Symp. on Syst. Engi.*, 1–8 (2021).
35. Stadler, M., Vierhauser, M., Garmendia, A., Wimmer, M. & Huang, J. C. Flexible model-driven runtime monitoring support for cyber-physical systems. In *Proc. IEEE/ACM 44th Inter. Conf. on Softw. Engi.*, 350–351 (2022).
36. Vierhauser, M., Marah, H., Garmendia, A., Huang, J. C. & Wimmer, M. Towards a model-integrated runtime monitoring infrastructure for cyber-physical systems. In *Proc. IEEE/ACM 43rd Inter. Conf. on Softw. Engi.*, 96–100 (2021).
37. Zimmermann, T. C., Konietzko, E. & Lindow, K. Graph-based parameter management for configuration controlled multi-level modeling of cyber-physical systems. In *Proc.Ann. Syst. Sys. Eng. Conf.*, 270–274 (2024).
38. Milner, R. *The Space and Motion of Communicating Agents* (Cambridge University Press, 2009).
39. https://github.com/ale-f/big-red

Author contributions

All authors contributed to the study conception and design. C.L.: Conceptualization, methodology, validation, writing—review and editing, funding acquisition, supervision. Q.Z.: investigation, formal analysis, writing—original draf preparation, supervision. J.Z.: Conceptualization, formal analysis, data curation, supervision. All authors reviewed the manuscript. All the authors approve the submission to this journal.

Funding

This study was funded by "This research was supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LQ24F020023", and "This Project is Supported by Ningbo Natural Science Foundation under Grant No. 2023J180", and "A Project Supported by Scientific Research Fund of Zhejiang Provincial Education Department under Grant No. Y202351645", and "Scientific Research Project Funded by Ningbo University of Technology under Grant No. 2040011540019", and "Scientific Research Incubation Program of Ningbo University of Technology under Grant No. 2022TS23".

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to C.L. or Q.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.