IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 15, 2021, accepted March 3, 2021, date of publication March 8, 2021, date of current version March 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3064249

A Relational Abstraction of Structure and Behavior for Cyber-Physical System Design

CHAO WANG[1], LI WAN[1], TIFAN XIONG[1], YUANLONG XIE[1,2], (Member, IEEE), AND SHUTING WANG[1,2]

[1]School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Guangdong Intelligent Robotics Institute, Dongguan 523808, China

Corresponding author: Tifan Xiong (xiongtf@hust.edu.cn)

This work was supported in part by the National High Technology Research and Development Program of China under Grant 2018YFB-17008, in part by the China Postdoctoral Science Foundation under Grant 2019M650179, in part by the Guangdong Innovative and Entrepreneurial Research Team Program under Grant 2019ZT08Z780, in part by the Guangdong Huazhong University of Science and Technology (HUST) Industrial Technology Research Institute, in part by the Guangdong Provincial Key Laboratory of Digital Manufacturing Equipment under Grant 2020B1212060014, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515110464.

ABSTRACT Model-based approaches are essential for designing cyber-physical systems, which adopt the formal models to simultaneously form the specifications and enable the verification at an early stage. Aimed to model the complex structure and continuous-discrete hybrid behavior of cyber-physical systems, this paper mathematically defines a dynamic relational system so that the cyber-physical system can be regarded as dynamic relational systems in a hierarchical structure and each dynamical relational system is a triple of dynamic attributes, subsystems, and hybrid relations between attributes and subsystems. Every hybrid relation contains a tuple and a predicate to govern the system behaviors. By utilizing the dynamic relational system, a parametric abstraction is then performed to specify the design requirements and schemes. It can represent the structure and behaviors of multiple cyber-physical system design schemes in an integrated manner. With a mathematical foundation, the constructed relational models are beneficial for structural analysis and behavior verification. An implementation case of a friction-driven plate conveyor is presented to illustrate the design specification with relational models, and the connectivity analysis and behavior verifications are carried out to show the effectiveness and engineering practicability of the achieved models.

INDEX TERMS Formal specification, cyber-physical system, cyber-physical system modeling, system design, dynamic relational system, dynamic structure and behavior.

I. INTRODUCTION

The expanding demands for intelligent services and network interconnections in modern life have made industrial products increasingly present with the characteristics of the cyber-physical system (CPS). Compared with industrial concept such as Internet of Things (IoT), Industrial Internet, and smart products, CPS is an academic term emphasizing the interconnection and synergy between the cyber and physical worlds [1]. By applying information technologies (e.g., embedded computing, network communication, and distributed control) to physical systems, a CPS could adaptively change its behavior logic according to the applied environment, thus moving towards building trusted autonomy. The notable features of a CPS can be summarized as follows.

- *Multidomain and heterogeneous*. As the integrations of the cyber and physical worlds, CPS involves multiple domains of computation, communication, control, electric, and machine [2]–[4]. The sequential computation in virtual space and the concurrent evolution in physical world result in heterogeneous behaviors, interactions, and architectures [3]–[6].
- *Continuous-discrete hybrid dynamic*. The discrete computation steps in software, and continuous evolution of physical behaviors take time and are coordinated temporally in a hybrid manner [1], [4], [7].
- *Complex structure and large-scale*. The various heterogeneous subsystems in a CPS are associated with the compositions and interactions between them,

The associate editor coordinating the review of this manuscript and approving it for publication was Engang Tian.

leading to a complex, large-scale, and hierarchical structure [1], [3], [6].
- *Interactive*. Through control and feedback, the physical and cyber subsystems are tightly coupled, and synergy is produced by the close interactions [5], [6], [8].
- *Free to join and exit*. A subsystem is free to join or exit, and the CPS can adjust its behavior according to the participating subsystems [3], [5].

These intrinsic complexities impose new challenges for the design of CPS, which is always addressed with model-based approaches [6]. In model-based design, formal models are used to precisely specify system schemes and enable early verification and simulation [4]. Subsystems in a CPS often have domain-specific modeling techniques due to their multidomain and heterogeneous nature [6], [9]. The capacity of rigorously modeling the heterogeneous subsystems is important for guaranteeing an efficient CPS design. Furthermore, these heterogeneous subsystems are closely coupled in a complex (even dynamic) structure. In this regard, a separate design and analyses manner are not sufficient for a practical implementation of these subsystems. To enable the systemic design, their structure and behavior must be properly represented in an integrated way.

A natural idea to describe the structure and hybrid behaviors of CPS is to integrate the heterogeneous models into a framework. Some works achieve this goal by sharing data [10], [11] or transforming [12] between different models. With the abstraction of interactions between them, the composition of multiple heterogeneous models can be realized under different integration frameworks [13]–[15]. Then, various modeling languages [16]–[21] and toolsets [4], [16], [20] are proposed and demonstrated for the integration-based design of CPS. Most of these languages are variants of XML [22].

Applying the established modeling methods or their variants is another major approach for CPS representation during design. In the past decade, lots of studies have extended the existing modeling languages with new features to represent more aspects of CPS. Note that some contributions [23]–[26] attempt to achieve system-level architecture modeling based on domain-specific languages, such as SystemJ [27], Petri net [10], and AADL+ [26]. More attention is paid to extend the behavior expression ability of the model in cyber and physical spaces. For example, physical processes are incorporated into cyber models with discrete abstraction of hybrid behavior or vice versa [9], [27]–[30]. Transition systems [9], [31] and hybrid systems [11], [29], [32] are the most common semantic fundamentals for modeling the continuous-discrete hybrid behavior in the CPS. There are also several extension efforts for multi-paradigm modeling languages in engineering, such as Modelica, to model the structure and behavior of CPS [33]–[35].

The works above can practically represent the structure and behavior of the CPS during design. However, the integration approaches are always too complicated since the system designers must be familiar with multiple languages and the coordination between different tools. The extension approaches of supplementary features are defective in most situations due to the difficulty in expressing all intrinsic characteristics of a CPS. For example, existing approaches can hardly model the dynamic structures of a bouncing ball and cellular net, such as Modelica, due to the discrete interaction and dynamic structure of the system of ball and ground, mobile and station. Considering the complex structure and interactions of heterogeneous subsystems, the design of a CPS requires a layered abstraction for systemic and unifying specifications and behavior at different levels.

This paper represents a CPS with a triple of attributes, subsystems, and relations between the attributes and subsystems. The subsystems at different abstraction levels form a hierarchical structure, and their behavior is implied by the hierarchical models abstracting the structure and interactions of these subsystems. In this perspective, we rigorously define a formal abstraction of a CPS, called dynamic relational system, to model its dynamic behavior and complex structure. A parametric form of the dynamic relational system is proposed to specify the requirements and schemes during design. The proposed abstraction provides an integrated manner to express and analyze the complex structure of CPS, as well as its discrete-continuous behaviors.

The rest of this paper is governed as follows. Section II offers the related work. Section III defines a dynamic relational system and explains how it expresses the structure and behaviors of a CPS. The parametric abstraction of the dynamic relational system is proposed in Section IV. Section V provides a case study of a friction-drive plate conveyor (FDPC) to illustrate the CPS design with relational representation and to verify the achieved model. Finally, Section VI gives the conclusions and presents future research directions.

II. RELATED WORK

The main challenge to implement CPS design is how to achieve the unified formal representation of such a multidomain, heterogeneous, and continuous-discrete hybrid system [36]. Over the past decade, there have been many theoretical and practical efforts relating to this research area. Depending on the main modeling interests, there are three categories of methods to model the CPS during design: structure-oriented, behavior-oriented, and comprehensive methods. In the following, we introduce researches and typical methods in each category, and give a brief critical appraisal of these methods.

A. STRUCTURE-ORIENTED METHODS

Structure-oriented methods focus on the CPS structure and interactions between heterogeneous subsystems. Under an integration framework, multiple heterogeneous models are composed into one based on the formal description of their structures and interactions. Several attempts [14], [37] have been made for modeling the heterogeneous subsystems under well-defined interaction semantics, such as the

behavior composition semantics [37], and Behavior Interaction Priority [14]. There also exist architecture description languages [18], [25], [26], most of which are based on XML or its variants [22], including UML [19], [20], CyPhyML [16], and SysML [4], [17], [21], [38].

As a typical structure-oriented method, SysML has many features potential for CPS modeling [4]. It provides useful extensions for representing the CPS architectures [39] and an alternative approach to facilitate integrating multidomain models into a unifying one. Sangiovanni *et al.* adopt SysML for the inspired CPS design to represent the architectural decomposition [40]. With the composite structure diagram and the flow ports, the internal block diagram notation of SysML has great potential for modeling the interactions in CPS [4], [39]. Then, Passerone *et al.* propose a framework called SPEEDS for metamodel building, simulation, and analysis of system models [41]. This framework forms an important part of the metamodel-based interoperability concepts. Palachi *et al.* present a simulation framework for models of CPS based on SysML and numerical solvers [38]. Based on the formal specification from requirements to architectural, Derler *et al.* interlink a family of tools using SysML to co-model and co-simulation via FMI interface definitions [4], [17].

Structure-oriented methods can well represent the complicated structure and interactions in the integration of heterogeneous subsystems. Heterogeneous subsystems are often separately modeled in different views by various tools. For a consistent implementation in various tools, it demands a strictly defined and well-considered semantics of interactions between heterogeneous models in multiple forms. In addition, the deepening of multi-domain integration in a CPS makes the structure and interactions of the subsystems more and more complex, demanding the collaborative design process and data exchange of different toolsets. The absence of explicit/standard semantics and its complexity limit its ability to address the CPS design problems.

B. BEHAVIOR-ORIENTED METHODS

The behavior-oriented methods focus on the representation of continuous-discrete hybrid behavior in CPS. Typically, the unifying representation of the hybrid behavior is achieved by using discrete modeling methods [23], [24], [32], [33], [42], such as Petri net [23], [32], [42], Simulink [33], transition system [9], based on the discrete abstraction of the underlying continuous evolution, or by modeling method for hybrid system, such as timed automata [34], [43], hybrid automata [36], and hybrid program [29], [30], [44].

Due to the feature of continuous-discrete mixing, CPS is often modeled as hybrid systems with formal methods like hybrid automata [11]. Note that hybrid automata represent an extension of finite-state automata with continuous variables, which evolve dynamics specified at each discrete control mode. The ODEs in each location represent continuous dynamics, whereas the transitions between locations represent discrete dynamics [45]. They are generally described by using denotational semantics in terms of the piecewise continuous function. Hybrid automata support the specification and formal verification of the behavior of complex systems with a graphical modeling formalization [46]. Platzer *et al.* propose a hybrid program method to characterize interacting discrete and continuous dynamics of CPS [30], which can be verified with differential dynamic logic [29], [44]. Nuzzo *et al.* propose a design method for CPS by specifying the design with an A/G contract and analyzing the design at different abstraction levels with the differential dynamic logic and the hybrid automata [36].

Behavior-oriented modeling methods, such as transition systems, differential dynamic logic, and hybrid automata, have well-defined semantics to model the continuous-discrete hybrid behavior of CPS. Unfortunately, they provide limited support for model composition and have difficulty in modeling the complex structure of the CPS.

C. COMPREHENSIVE METHODS

The previous two categories of methods are involved in each other more or less, but with different concerns. The comprehensive methods pay relatively balanced attention to the structure and behavior of the CPS. Instead of applying established modeling techniques, these methods are mostly tailored-modeling techniques, such as Modelica and AADL+ [22]. This is the present trend of CPS modeling methods as the understanding of CPS becomes more and more profound.

As the first step to combine the multi-paradigm into one formalization, Amrani *et. al.* presents a comprehensive formal framework of multi-paradigm modeling for the CPS [35]. Engineering modeling languages, such as Modelica, and their variant may be potentially comprehensive methods that describes both the structure and behavior of the CPS [37]. They can express the dynamic behavior in different domains mathematically and achieve seamless integration of them [47] based on DAEs (differential algebraic equations) combined with an event-handling mechanism. Fritzson *et. al.* and Junjie *et. al.* discuss the abilities of Modelica in the unified expression of multidomain, continuous-discrete hybrid, and heterogeneous subsystems in CPS [48], [49]. By extending Modelica with temporal logic, Garro *et al.* presents a new modeling language called Form-L for CPS designs [37].

There is relatively little research on the comprehensive modeling languages and tools. Most of established comprehensive methods, such as Modelica, and Form-L, satisfy engineering needs of modeling in most situations. However, these methods lack systematic consideration of CPS design and a formal definition of behavior semantics. This may cause the difference in the tool implementation and defect of modeling features of CPS. For example, the system with dynamic structure, such as a bouncing ball and cellular net, is hard to be represented naturally with existing comprehensive methods due to the lack of discrete interactions modeling.

In this paper, we propose a relational abstraction to synthesize the advantages of SysML, hybrid automata, and

Modelica. Compared with the existing modeling methods, this approach can understand the CPS from a more general perspective, which has a mathematical semantic definition and realizes the unified formal representation of the dynamic structure and behavior of the CPS. Additionally, the relational abstraction is design-oriented and beneficial for supporting the specification of requirements and design schemes.

III. DYNAMIC RELATIONAL SYSTEM

Mathematical formalizations are often used to realize the unified representation of systems [50]. This paper designs a relational formalism from a systemic perspective on CPS, which naturally expresses the structure and the interactions between each part of the system. Inspired by the relational structure of general systems [51], it provides a mathematical foundation to develop the rigorous representation for CPS design. It has a layered structure, and in each layer, it contains a set of attributes, subsystems, and relations between the attributes and subsystems.

Definition: A *dynamic relational system* is formally defined as a triple $S = (A, M, R)$, where:

- A is a finite set of *dynamic attributes*, representing the continuous -discrete hybrid states;
- M is a finite set of subsystems at a specific abstraction level;
- and R is a finite set of *hybrid relations*, representing the interactions between the attributes and subsystems.

Both the attributes in A and the subsystems in M are elements constructing the system S. If the attributes set $A = \emptyset$, the system S is a composition of subsystems, which is called *composited dynamic relational system*. The subsystem set $M = \emptyset$ means that S has no subsystems. Such a system consisting of attributes and relations is called a *primary dynamic relational system*.

A subsystem $S^1 = \left(A^1, M^1, R^1\right) \in M$ is called a 1st-level subsystem of S. If there exists a sequence $S^i = \left(A^i, M^i, R^i\right) \in M^{i-1}$ for $i = 1 \ldots n$, the system $S^n = (A^n, M^n, R^n) \in M^{n-1}$ is called an nth-level subsystem of S. Correspondingly, if all nth-level subsystems of a system S are primary systems, S is called n level system. The subsystems in each level are associated with hybrid relations, forming a hierarchical structure.

The semantics of the dynamic attribute, hybrid relation and dynamic relational system are mathematically explicated in the remainder of this section.

A. DYNAMIC ATTRIBUTE

A dynamic attribute is a real variable representing a sequence of states. It is defined as the fundamental object to unify the representation of the continuous physical states and discrete cyber states in the hierarchical system structure. It evolutes on a hyper-real time domain [13], [43] $\mathbb{T} = [t_0, t_1] \subseteq \mathbb{R}^+$ of interest. For any instant $t \in \mathbb{T}$ and $t \neq t_0, t_1$, there exist a predecessor $t^- = \lim_{\epsilon \to 0} t - \epsilon \in \mathbb{T}$ and a successor $t^+ = \lim_{\epsilon \to 0} t + \epsilon \in \mathbb{T}$. The value of an attribute a at each instant

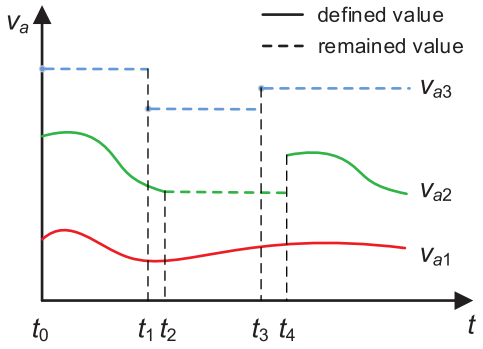

FIGURE 1. Dynamic attribute always has a value.

is given by an evaluation function $v_a : \mathbb{T} \to \mathbb{R}$. The function $v_a(t)$ always has a real value on time domain $\mathbb{T}$, as the value of attribute $a1$, $a2$, and $a3$ shown in Figure 1. In particular, an attribute representing a piecewise state $a2$ or discrete state $a3$ preserves its last defined value at the instants where the evaluation function is undefined. The value sequence V_a of a on time domain $\mathbb{T}$ is called the trajectory of attribute a, which is formally defined as $V_a = \{v_a(t) \mid t \in \mathbb{T}\}$.

Theorem 1: For an arbitrary dynamic relational system S, there exists exactly one set $\hat{A}(S)$ consisting of all attributes in S.

Proof: Consider an n level system $S = (A, M, R)$. Define $A_0^* = A$ and $M_0^* = M$. For indices $i \in \{1 \ldots n\}$, the sequence $\left\{A_i^*\right\}$ of attributes in each level and the sequence $\left\{M_i^*\right\}$ of subsystems in each level can be defined such that

$$A_i^* = \bigcup \left\{A_s \mid s = (A_s, M_s, R_s) \in M_{i-1}^*\right\},$$
$$M_i^* = \bigcup \left\{M_s \mid s = (A_s, M_s, R_s) \in M_{i-1}^*\right\}.$$

Let $\hat{A}(S) = \bigcup \left\{A_i^*\right\}$. For the finite level of S, $\hat{A}(S)$ is the set consisting of all attributes in S, and from the uniqueness of each A_i^*, it follows that $\hat{A}(S)$ is uniquely determined by S.

The behaviors of S can be described by the trajectory $V_{\hat{A}(S)}$, which is the union trajectories of its attributes and 1st-level subsystems, formally denoted as $V_{\hat{A}(S)} = V_A \cup \left(\bigcup \{V_{A(S_i)} | S_i \in M\}\right)$. Conversely, the behavior of a 1st-level subsystem S_i could be considered as the projection of $V_{\hat{A}(S)}$ onto attributes set $\hat{A}(S_i)$ of S_i.

B. HYBRID RELATION

A hybrid relation represents the interactions between objects in a CPS, such as a power transfer in physical systems or a signal connection in cyber systems. It extends the relation expressing coupling between objects in general system [51] with temporal characteristics to model the continuous-discrete hybrid interactions between objects in a CPS.

1) RELATION BETWEEN ATTRIBUTES

As the basis of the hybrid relation, the relation between attributes is discussed here. An *attribute relation* r on a non-empty attribute set A is a tuple of attributes semantically related to a mathematical predicate. An n-ary attribute

TABLE 1. Representation forms and semantics of attribute relations.

Type	Form	Semantics
Causal	$(X, y)\|y \coloneqq f(X)$	$\forall t \in \mathbb{T}\|\left(v_y(t) \coloneqq f\left(v_X(t)\right)\right)$
Acausal	$X\|f(X) = 0$	$\forall t \in \mathbb{T}\|\left(f\left(v_X(t)\right) = 0\right)$
Bound	$X\|f(X) \geq 0$	$\forall t \in \mathbb{T}\|\left(f\left(v_X(t)\right) \geq 0\right)$
Event	$(X, Z)\|e(Z)?$ $\mathcal{P}(X):\mathcal{P}'(X)$	$\forall t \in \mathbb{T}\|(e(v_Z(t)) \wedge \mathcal{P}(v_X(t)))$ $\vee\left(\neg e(v_Z(t) \wedge \mathcal{P}^{\wedge\prime}(v_X(t))\right)$

relation r is denoted as

$$(x_1, x_2, \ldots, x_n)|\mathcal{P}(x_1, x_2, \ldots, x_n),$$

where the n-ary tuple $(x_1, x_2, \ldots, x_n)$ is an element in A^n declaring the structure of attributes; $\mathcal{P}(x_1, x_2, \ldots, x_n)$ is a mathematical predicate interpreting the temporal characteristics of r.

For an attribute relation $r = X|\mathcal{P}(X)$, the predicate $\mathcal{P}(X)$ constrains the trajectory of attributes in X. The trajectory $V_X = (V_{x1}, V_{x2}, \ldots, V_{xn})$ of X is a projection of V_A onto X. It satisfies the predicate of r at any instant, i.e., each value $v_X(t), t \in \mathbb{T}$ in V_X holds the predicate $\mathcal{P}(X)$ true, denoted as $V_X \models r$.

The predicate $\mathcal{P}(X)$ is an arithmetic formula, such as an assignment function, equation, or inequality. It is built over attributes in tuple X and their first-order differences, real number, parentheses, binary operators (such as $+$, $-$, $*$, $/$, and exp), evaluation operator $:=$, and relation operators (such as $=$, $<$, $\leq$, $\geq$, $>$), with extension of event expressions. An event expression is a predicate indicating if $v_X(t)$ satisfies some conditions connected with logical connectives, such as $y > x - 3, x > 0 \wedge x < 1$.

Four types of attribute relations are defined in this paper according to the characteristics of interactions, as shown in Table 1. Causal relations often represent the input-output interactions in cyber space, whereas acausal relations represent bidirectional interactions in physical space. Beside them, bound relations represent the ambiguous interactions from human, random algorithm, or system cannot be predicted. Event relations are used to model the discrete characteristics in the event-triggered interactions. An event relation has two predicate branches, which are separated by colons. When the event expression is satisfied, the predicate governing attribute values switches from one branch to another.

2) RELATION BETWEEN ATTRIBUTES AND SUBSYSTEMS

Consider an attribute $a \in A$ and a 1st-level subsystem S^1 in the system S. A hybrid relation between them is a synthesis of attribute relations between a and the attributes in S^1. Similarly, a hybrid relation between two subsystems

S_1^1 and S_2^1 is a synthesis of attribute relations between their attributes. Without loss of generality, a hybrid relation can be defined as a synthesis of attribute relations between the attributes in them.

TABLE 2. Typical interactions between subsystems of CPS and their relational representation.

Interactions	Attributes in S	Attributes in S'	$\mathcal{P}(A_S, A'_{S'})$
Signal interaction	Output s_o	Input s_i	$s_i \coloneqq s_o$
Translational interaction	Position p, Force f	Position p', Force f'	$p - p' = 0$, $f + f' = 0$
Rotational interaction	Angle a, Torque t	Angle a', Torque t'	$a - a' = 0$, $t + t' = 0$
Electrical interaction	Voltage v, Current c	Voltage v', Current c'	$v - v' = 0$, $c + c' = 0$
Hydraulic interaction	Pressure p, Mass Flow f	Pressure p', Mass Flow f'	$p - p' = 0$, $f + f' = 0$
Thermal interaction	Temperature t, Heat Flow f	Temperature t', Heat Flow f'	$t - t' = 0$, $f + f' = 0$
...	...	...	...

Definition An n-ary *hybrid relation* r is defined as $X|\mathcal{P}(X)$, where $X = (x_1, x_2, \ldots, x_n) \in (A \cup M)^n$ is the tuple of r, in which each x_i can be an attribute or a subsystem; and $\mathcal{P}(X)$ is a set of predicates whose variables are from x_i or attributes in x_i.

It is intuitive that an n-ary hybrid relation is equivalent to a set of n-ary attribute relations. The tuple of each attribute relation is an element in $\prod A(x_i)$, satisfying that

- The attributes in tuple are from $x_i, i = 1 \ldots n$;
- For each x_i, there exists an attribute a in the tuple.

Consider a 1-D spring oscillator consisting of a spring s and a mass body m; the hybrid relation between s and m could be represented as $(s, m) \mid \{f_s + f_m = 0, p_{sr} - p_m = 0\}$, where f_s and f_m are the forces on the spring and mass body, respectively, and p_{sr} and p_m are the corresponding positions. This hybrid relation can be considered as two attribute relations $(f_s, f_m)|f_s + f_m = 0$ and $(p_{sr}, p_m)|p_{sr} - p_m = 0$. Typical hybrid relations in CPS, as well as attributes involved, are summarized in Table 2.

Like the construction of $\hat{A}(S)$, the set of attribute relations in an n level system S can be constructed.

Theorem 2: There exists a unique attribute relation set $\hat{R}(S)$ containing all the attribute relations in S.

Proof: According to the definition of hybrid relation, each hybrid relation is equivalent to a set of attribute relations. Define $R_0^* = \bigcup\{r_H \mid r_H \in R_0\}$ as the union of equivalent attribute relation sets of each hybrid relation in S, $M_0^* = M_0$ as the set of the 1st-level subsystems in S, and $\tilde{R}_0 = \bigcup\left\{R_s \mid s = (A_s, M_s, R_s) \in M_0^*\right\}$ as the union of hybrid relation set in the 1st-level subsystems. For $i \in \{1, \ldots, n\}$, three sequences $\{R_i^*\}$, $\{M_i^*\}$ and $\{\tilde{R}_i\}$ can be defined such that

$$R_i^* = \bigcup\left\{r \mid \forall r_H \in \tilde{R}_{i-1}(r \in r_H)\right\},$$
$$M_i^* = \bigcup\left\{M_s \mid s = (A_s, M_s, R_s) \in M_{i-1}^*\right\},$$
$$\tilde{R}_i = \bigcup\left\{R_s \mid s = (A_s, M_s, R_s) \in M_i^*\right\}.$$

Let $\hat{R}(S) = \bigcup \left\{ R_i^* | i \in \{0, \ldots, n\} \right\}$, then $\hat{R}(S)$ is the exactly unique attribute relation set equivalent to all the hybrid relations in S.

C. SEMANTICS OF DYNAMIC RELATIONAL SYSTEM

A dynamic relational system models a CPS in different abstraction levels, forming a hierarchical structure. The system $S = (A, M, R)$ is a triple of a dynamic attribute set, a lower-level subsystem set, and a hybrid relation set. Suppose that E is the event expression set of all event attribute relations in R. When an event expression $e \in E$ is satisfied, the predicate switches from one branch to another. This leads to the change in system structure and behavior. The discrete transition of the system structure and behavior could be described by the mapping $l_e : S\left(t_e^-\right) \rightarrow S\left(t_e\right)$, where t_e^- is the instant before the event is triggered. $S\left(t_e^-\right)$ and $S\left(t_e\right)$ represent different system modes of S. In each system mode, the structure are static, and the evolutions are constrained by continuous functions or equations from causal relations, acausal relations, bound relations, or activated branches of event relations.

A hierarchical dynamic relational system can be flattened into a primary system to interpret its continuous-discrete hybrid behavior.

Theorem 3: An arbitrary dynamic relational system $S = (A, M, R)$ can be written as a primary system without subsystems, denoted as $\hat{S} = \left(\hat{A}(S), \emptyset, \hat{R}(S)\right)$.

Proof: Theorem 1 and theorem 2 states that all the attributes and all the hybrid relations in S can be written as the attribute set $\hat{A}(S)$ and the attribute relation set $\hat{R}(S)$. By transfinite induction of layers in the hierarchical structure, S can be represented as the primary system $\hat{S} = \left(\hat{A}(S), \emptyset, \hat{R}(S)\right)$ without subsystems.

Based on theorem 3, the behavior of S is the same with the behavior of $\hat{S}$ due to the equivalent of relations governing the attributes. The attributes in $\hat{S}$ evolve dynamics specified at each discrete system mode. The transitions between different system modes and continuous behaviors in each system mode can be interpreted by a hybrid automaton

$$H = (Loc, Edge, \Sigma, X, Init, Inv, Flow, Jump),$$

where:

- $Loc \subseteq 2^{\Sigma}$ is the finite set of locations representing behavior modes of S, and $\Sigma = \hat{E}$ is the finite set of event expressions in $\hat{R}(S)$;
- $Edge \subseteq Loc \times \Sigma \times Loc$ is the finite set of edges representing the discrete transitions in the behavior modes, and each transition is a linkage mapping in $\{l_e \mid e \in \Sigma\}$;
- X is the finite set of real-valued variables representing attribute set $\hat{A}(S)$;
- $Init$, Inv, and $Flow$ are three sets of predicates in the initial relations, the bound relations, and the rest attribute relations included in $\hat{R}(S)$, respectively.
- $Jump$ is the finite set of event expressions related to events in Σ, stating when a transition is possible.

In this interpretation, the events in $\hat{E}$ divide S into a set $\mathcal{S} = \{S_0\} \cup \left\{ S_e \mid e \in \hat{E} \right\}$ of continuous systems with static structure, where S_0 is the initial mode of S, and S_e represents the mode just after event e is triggered. The system S evolves as the discrete transitions between these system modes and the continuous evolutions constrained by the predicates in relations.

D. EXAMPLE OF DYNAMIC RELATIONAL SYSTEM

A water heater, as shown in Figure 2(a), is a typical example to illustrate the semantics of a dynamic relational system. It maintains the water temperature in a range and mainly contains a controller, a heater, a tank filled by water, and a temperature sensor.

The controller switches the working mode of the heater between "ON" and "OFF" according to the temperature feedback from the sensor. It can be represented as the dynamic relational system S_C in (1), where s_{Cin} is the feedback signal from the temperature sensor, and s_{Cout} is the control signal with two values: "1" representing the "ON" state and "0" representing the "OFF" state. Note that the attribute set enclosed in parentheses and the attribute tuple enclosed in braces are different. The former indicates the set of attributes in the controller, whereas the latter means the attributes sequentially involving in the relation of the controller.

The power of the heater is 300 W. It can be represented as S_H in (2), where s_H, and f_H are the input signal, and the output heating flow, respectively.

The environment temperature is 20 °C, and the coefficient of heat dissipation is assumed as 0.2 in this example. The tank can be represented as (3), where tp_T is the temperature of the water, which starts at 20 °C and should not exceed 100 °C; and f_T is the input heating flow to the water.

The temperature sensor is represented as (4), where tp_S, and s_S are the temperature sensed and the feedback signal, respectively.

Based on these models of subsystems, the water heater can be represented as a system shown in (5). When the events $s_{Cin} < 60$ and $s_{Cin} >$ *mathrm*90 are triggered, the water heater switches its behavior mode between "ON" and "OFF", as the hybrid automaton shown in Figure 2(b). In each system mode, the water heater continuously evolves following hybrid relations between attributes and subsystems. For example, the relation between the temperature and heating power is

$$(T_T, f_T) \mid \frac{d(T_T)}{dt} - \frac{f_T - 0.2 * (T_T - 20)}{4.2 * 10^3} = 0.$$

It constrains the temperature evolves following the differential equation in its predication, i.e., the water temperature $v_T(t)$ satisfies

$$\frac{dv_T(t)}{dt} = \frac{300 - 0.2 * (v_T(t) - 20)}{4.2 * 10^3}.$$

IV. DESIGN SPECIFICATION BASED ON DYNAMIC RELATIONAL SYSTEM

A dynamic relational system could be used to represent the design schemes of the CPS. In this section, a parametric abstraction of the dynamic relational system is defined for specifying a family of possible design schemes satisfying the design requirements.

Definition: A *design model* is defined as a tuple $D = (S_P, P, C_P)$, where:

- $S_P = (A, M, R_P)$ is a parametric dynamic relational system with R_P being the set of parametric relations;
- P is a finite set of parameters, which can affect the definition of relation;
- C_P is a finite set of predicates whose free variables are from P. C_P states the constraints to the values of P.

The relational design model abstracts the suppressed details that are not of interest, and can be used to predict the states and behaviors of the designed system. For a design model $D = (S_P, P, C_P)$, C_P states the design space of a system. Given a parameter value v_P, the evaluation $C_P(v_P) \in \{\text{True}, \text{False}\}$ indicates whether the design scheme at v_P is feasible. The notation $[\![C_P]\!]$ is the set of all possible values satisfying the predicates C_P. Each parameter value $v_P \in [\![C_P]\!]$ stating a dynamic relational system $S(v_P) = (A, M, R_P(v_P))$ specifying a feasible scheme of the putative system. To state it clear, a design can be denoted as $D = (A, M, R_P, P, C_P)$ by expanding the dynamic relational system S_P. For example, the design of the tank in the previous can be represented with relational design model as (6), where δ, T_e, and M are design parameters representing the coefficient of heat dissipation, the environment temperature, and the mass of the water, respectively. The predicate set "$\{\delta = 0.2, T_e = 20, M \leq 5, M \geq 0.1\}$" is the constraints to parameters. Each value $v_P \in [\![C_P]\!]$ decides a design scheme for the tank. The water tank in section III.D is a scheme of the model D_T at $v_P = \{0.2, 20, 1\}$, (1)–(6), as shown at the bottom of this page.

A complex CPS is always designed as the composition of some subsystems, and these subsystems are designed as the composition of some simpler subsystems. The hierarchical structure of the dynamic relational system can naturally express the composition of subsystem in CPS design. A set of relational models can compose the model of a more complex system with proper relations between them. Consider a finite set of relational models $\{D_i\}$, where $i \in I = \{1, 2, \ldots, n\}$, $D_i = (A_i, M_i, R_i, P_i, C_{Pi})$, the *composition* of them with a set of hybrid relations $R_{\{m_i\}}$ between them is a new design $D_\Sigma = \left(\emptyset, \{m_i\}, R_{\{m_i\}}, \emptyset, \emptyset\right)$. Therefore, the design of a CPS can be performed by composing the design of subsystems layer by layer.

The premise that a relational model can represent some design schemes of a system is that there exist some possible trajectories for each parameter value $v_P \in [\![C_P]\!]$. To ensure a design is concrete enough to implement, the relational model must be deterministic, which means it indicates a unique system behavior and all attributes of interest can be described by a certain trajectory at each parameter value $v_P \in [\![C_P]\!]$. For cyber processes described by cause-effect relations, the determinacy means that given an input, it will always produce an output and the same inputs cause same outputs. For physical interactions described by acausal relations, the determinacy means that the behavior is definitive and can be used to verify the safety of a system. A deterministic relational model states an implementable design scheme for each parameter value.

Verifying the compatibility and determinacy of a relational model is its mathematical analysis. According to theorem 3, the hierarchical model of FDPC can be written as a primary dynamic system over parameter set P by transfinite induction. The predicates of attribute relations in the primary dynamic system forms a mathematic system of inequations and equations extended with event expression. As the semantics states in section III.D, the mathematic system can be divided into different modes linked by event transitions. In each mode, the behavior is constrained by continuous DAEs and inequalities. The compatibility of a relational model is equivalent to that there exists some solutions satisfying constraints in each mode, and the determinacy means that there exists exactly one solution satisfying the constraints in each mode. Therefore, the verification of compatibility and determinacy equals the

$$S_C = \left(\{s_{Cin}, s_{Cout}\}, \emptyset, \left\{ (s_{Cin}, s_{Cout}) \mid s_{Cin} < 60 ? s_{Cout} := 1 : s_{Cin} > 90 ? s_{Cout} := 0 \right\}\right) \tag{1}$$

$$S_H = \left(\{s_H, f_H\}, \emptyset, \{(s_H, f_H) \mid f_H = s_H * 300\}\right) \tag{2}$$

$$S_T = \left(\{tp_T, f_T\}, \emptyset, \left\{ \begin{array}{l} (tp_T) \mid tp_T \leq 100, \\ (tp_T, f_T) \mid d(tp_T)/dt - \left(f_T - 0.2 * (tp_T - 20)\right) / \left(4.2 * 10^3\right) = 0 \end{array} \right\}\right) \tag{3}$$

$$S_S = \left(\{tp_S, s_S\}, \emptyset, \{(tp_S, s_S) \mid s_S := tp_S\}\right) \tag{4}$$

$$S = \left(\emptyset, \{S_C, S_H, S_T, S_S\}, \left\{ \begin{array}{l} (S_C, S_H) \mid s_H := s_{Cout}, \\ (S_H, S_T) \mid \{T_H - T_T = 0, f_H + f_T = 0\}, \\ (S_T, S_S) \mid T_S - T_T = 0, \\ (S_S, S_C) \mid s_{Cin} := s_S \end{array} \right\}\right) \tag{5}$$

$$D_T = \left(\begin{array}{l} \left\{tp_T, f_T\right\}, \emptyset, \left\{\left(tp_T, f_T\right) \mid f_T - \delta * \left(tp_T - T_e\right) / \left(4.2 \times 10^3 \times M\right) - \left(d\left(tp_T\right)\right) / (dt) = 0\right\}, \\ \{D, T_e, M\}, \{\delta = 0.2, T_e = 20, M \leq 5, M \geq 0.1\} \end{array} \right) \tag{6}$$

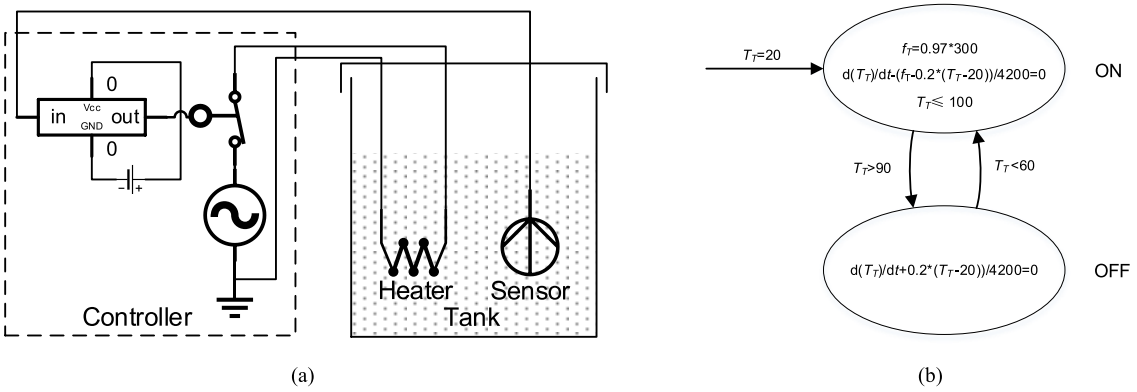

FIGURE 2. A water heater maintains the temperature in 60 ∼ 90 °C. The heating power is 300 W. There is 1 Kg of water in the tank. (a) The structure of the water heater. It consists of a controller, a heater, a temperature sensor, and a water tank. (b) The hybrid automata illustrating the behavior of the water heater.

analysis of the existence and uniqueness of the solution in each mode.

A basic idea of verifying such an equations and inequalities mixing system is solving the equation part and validating the inequality part. Structure analysis methods, such as consistence analysis [52], [53], and deduction methods, such as dynamic difference logic [29], [44], can be used to speed up the solving procedure of the equations. Alternatively, from the perspective of piecewise continuous systems the relation model could be verified with verification tools for hybrid automata, such as HYCOMP [54], by transforming it into hybrid automata.

V. AN ILLUSTRATIVE DESIGN CASE

The goal of CPS design is to obtain a proper scheme for system implementation. The design process starts with the requirements and ends with an established relational model. In this section, a FDPC is taken as an example to illustrate the CPS design based on the relational representation.

A. CONSTRAINTS TO FDPC

The FDPC is a common piece of equipment in the modern industry. It is widely used in manufacturing due to its large driving force, low noise, and the absence of oil pollution.

An FDPC in practice has multiple drive units, which would join or exit the motion system as the plate moves. As shown in Figure 3, the FDPC in this example has two driver units. It is used to deliver workpieces within a range (0∼1 m) in an assembly line. The weight of the workpiece is in the range 1500 ∼ 2000 kg. To ensure sufficient time for processing, the velocity of the plate should be slower than 0.3 $m{\cdot}s^{-1}$. When the plate moves too slowly, the "creeping" phenomenon may easily appear, i.e., the plate moves in the pattern of "fast-slow" or "jump-stop". Therefore, the velocity should be greater than 0.05 $m{\cdot}s^{-1}$. We assume that there is no slip between the drive wheel and the plate if the velocity satisfies the conditions above. The requirements of this FDPC can be summarized as Table 3.

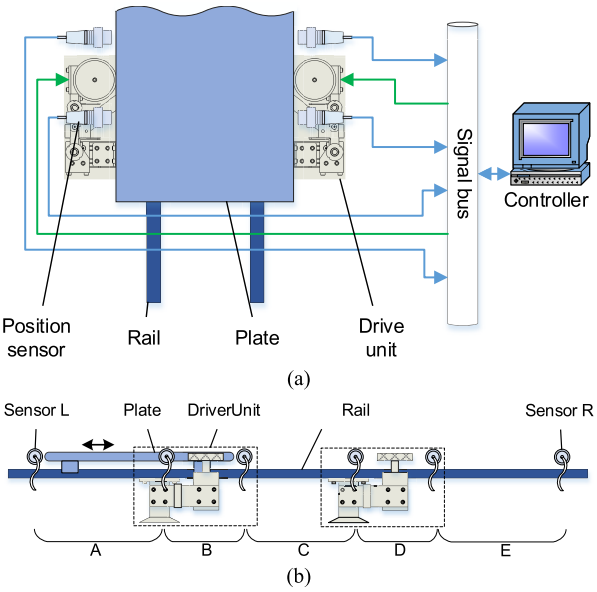

FIGURE 3. The FDPC in the design case. (a) The structure of the FDPC. The considered FDPC mainly consists of a controller running on a computer, a plate carrying the workpiece, a rail that the plate moves on, and some drive units arranged along the rail. The controller receives feedback signals from the sensors at two ends of the rail and sends control signals to drive units to change their rotate direction in modes of 'clockwise', 'anticlockwise', or 'stop'. A driver unit consists of a rotor, two proximate sensors, a switch decide "rotate" or "stop", and a wheel pressed to the plate by an elastic arm. It drives the plate to move along the rail by the friction. (b) The motion of the plate. The plate could move forward or backward. When the plate arrives at the first sensor (the left sensor when moves forward or the right sensor when moves backward) of a driver unit, the rotor joins the system and rotates. Here, "arrive" means that one part of the plate enters the region between two sensors, such as B and D in the figure. When it leaves the second sensor of a driver unit, the rotor stop rotation and exits the system. If the plate arrives at the sensor near the ends of the rail, i.e. the plate exceeds region A or E, all rotors switch their rotation direction. The plate starts slow down until the direction is reversed.

B. SPECIFICATION OF FDPC DESIGN

The design process of FDPC starts with the formal specification of ambiguous requirements in Table 1. As the model shown in the (7), the model D_{FDPC} formally specifies the requirements of the FDPC, where M and k_R are design param-

TABLE 3. Requirements of an FDPC design.

Requirement items	Constraints
Fraction factor between the rail and the plate	0.15, 0.2
Load of the plate	1500~2000 kg
Position of the plate	0~1 m
Velocity of the plate	0.05~0.3 $m \cdot s^{-1}$

eters representing the mass of the load, and the friction factor between the plate and the rail; v_P and p_P are the attributes representing the velocity and position of the plate. (8)–(15) specify the design scheme of the FDPC, as shown at the bottom of the page.

As shown in Figure 3, the FDPC consists of a controller, a set of driver units, a rail, two end sensors of the rail, and a plate moving on the rail. The controller decides the rotate direction of the driver units. When the plate reaches a driver unit, it is subjected to the friction force between them. The requirements of the FDPC can be partitioned into design constraints to these subsystems. The subsystems and hybrid relations between them can be represented as the model D_{FDPC} in (8). The structure of these subsystems is shown as the 1st-level of the FDPC in Figure 4.

For the plate, its acceleration a_P is the difference in its velocity v_P. Let fl and fr represent the forces from the left and right driver unit, respectively. According to Newton's second law, a_P satisfies $M * a_P = fl + fr + fb - M * g * k_R$, where g is the gravity, and $M * g * k_R$ is the frictional force from the rail. In addition, the safety of the FDPC requires that the plate remains in contact with at least one drive unit, which means that the position satisfies $p_P \geq p_{DL} - l/2$ and $p_P \leq p_{DR} + l/2$. The design of the plate can be represented as D_P in (9), where p_{DL}, p_{DR} are the positions of two driver units.

$$D_{FDPC} = \left(\{v_P, a_P\}, \emptyset, \left\{ \begin{array}{l} (v_P) \mid v_P \geq 0.05, \\ (v_P) \mid v_P \leq 0.3, \\ (p_P) | p_P \geq 0, \\ (p_P) | p_P \leq 1 \end{array} \right\}, \{M, k_R\}, \left\{ \begin{array}{l} M \geq 1500 \wedge M \leq 2000, \\ k_R = 0.15 \vee k_R = 0.2 \end{array} \right\} \right) \tag{7}$$

$$D_{FDPC} = \left(\begin{array}{l} \emptyset, \{S_P, S_{DL}, S_{DR}, S_C, S_{SL}, S_{SR}\} \\ \left\{ \begin{array}{l} (S_P, S_{SL}) \mid p_{S_L} = p_{S_P}, \\ (S_P, S_{SR}) \mid p_{S_R} = p_{S_P}, \\ (S_C, S_{SL}) \mid si_{CL} := so_{SL}, \\ (S_C, S_{SR}) \mid si_{CR} := so_{SR}, \\ (S_{DL}, S_C) \mid s_{dir} := so_C, \\ (S_{DR}, S_C) \mid s_{dir} := so_C, \\ (S_P, S_{DL}) \mid p_P - l/2 < p_{DL} \wedge p_P + l/2 > p_{DL} ? fl + f_{DL} = 0 : fl = 0 \\ (S_P, S_{DR}) \mid p_P - l/2 < p_{DR} \wedge p_P + l/2 > p_{DR} ? fr + f_{DR} = 0 : fr = 0 \end{array} \right\} \\ \{M, k_R, p_{DL}, p_{DR}, l\}, \{M \geq 1500 \wedge M \leq 2000, k_R = 0.15 \vee k_R = 0.2\} \end{array} \right) \tag{8}$$

$$D_P = \left(\begin{array}{l} \{p_P, v_P, a_P, fl, fr\}, \emptyset, \\ \left\{ \begin{array}{l} (p_P, v_P) \mid dp_P/dt - v_P = 0, \\ (v_P, a_P) \mid dv_P/dt - a_P = 0, \\ (fl, fr, fb, a_P) \mid fl + fr - M * g * k_R - M * a_P = 0, \\ (p_P) | p_P \geq 0, \\ (p_P) | p_P \leq 1, \\ (v_P) \mid v_P \geq 0.05, \\ (v_P) \mid v_P \leq 0.3 \end{array} \right\}, \\ \{M, g, k_R, p_{DL}, p_{DR}, l\}, \{M \geq 1500 \wedge M \leq 2000, g = 9.8, k_R = 0.15 \vee k_R = 0.2\} \end{array} \right) \tag{9}$$

$$D_{SENSOR} = \left(\{p_P, so_S\}, \emptyset, \left\{ (p_P, so_S) \mid p_P - l/2 > p_S \wedge p_P + l/2 > p_S ? so_S := 1 : so_S := 0 \right\}, \{p_S, l\}, \emptyset \right) \tag{10}$$

$$D_{CONTR} = \left(\{sil_C, sir_C, so_C\}, \emptyset, \{(sil_C, sir_C, so_C) \mid sil_C > 0 ? so_C = -1 : sir_C > 0 ? so_C = 1\}, \emptyset, \emptyset \right) \tag{11}$$

$$D_{DRIVERUNIT} = \left(\{s_{dir}\}, \{S_S, S_{SL}, S_{SR}, S_R, S_W\}, \left\{ \begin{array}{l} (S_S, S_{SL}) \mid si_{SL} := so_{SL}, \\ (S_S, S_{SR}) \mid si_{SR} := so_{SR}, \\ (S_S, S_R) \mid \{s_{dir} := so_S\}, \\ (S_R, S_W) \mid \{t_R + t_W = 0, w_R - w_W = 0\} \end{array} \right\}, \{p_D\}, \emptyset \right) \tag{12}$$

$$D_{SWITCH} = \left(\{s_L, s_R, s_{dir}, so\}, \emptyset, \{(s_L, s_R, s_{dir}, so) \mid s_L = 1 \vee s_R = 1 ? so := s_{dir} : so := 0\}, \emptyset, \emptyset \right) \tag{13}$$

$$D_{ROTATOR} = \left(\{s_{dir}, t_R, w_R\}, \emptyset, \{(s_{dir}, t_R, w_R) \mid \eta * P_e - s_{dir} * t_R * w_R = 0\}, \{\eta, P_e\}, \emptyset \right) \tag{14}$$

$$D_{WHEEL} = \left(\{f_W, v_W, t_W, w_W\}, \emptyset, \left\{ \begin{array}{l} (v_W, w_W) \mid v_W - w_W * R_W = 0, \\ (f_W, v_W, w, t_W) \mid t_W * w - f_W * v_W = 0 \end{array} \right\}, \{R_W\}, \emptyset \right) \tag{15}$$

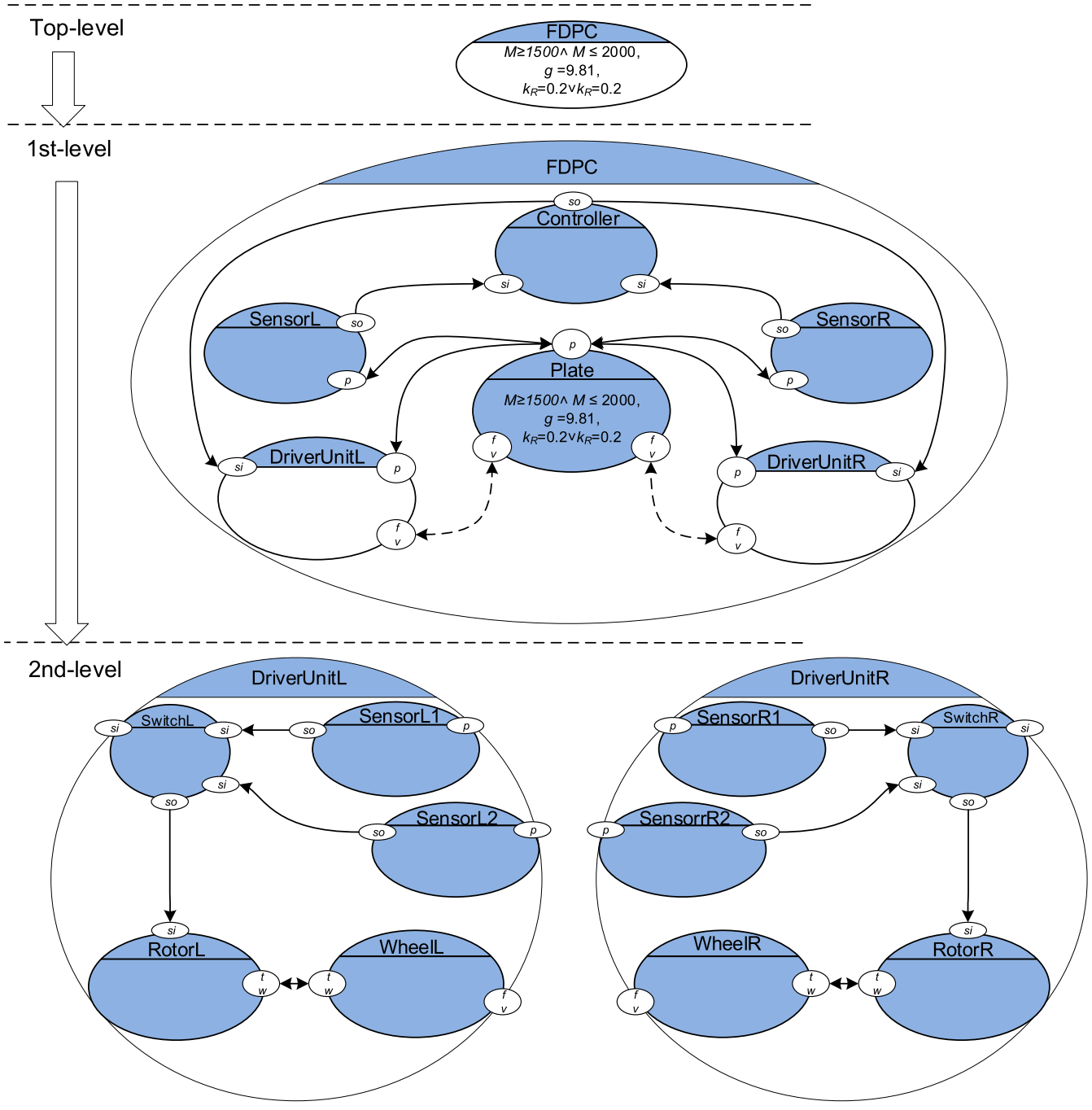

FIGURE 4. The hierarchical structure of FDPC design. Eclipses with name represent systems in different level. Lines with arrows represent relations between systems. The direction and style of a line states causality and continuity. The attributes connected by a relation can identity which interaction in Table 2 the relation represents. The predicates in eclipse are constraints to the design parameters.

A sensor's output switches to "1" when the plate reaches, and switches to "0" when the plate leaves. The design of a sensor can be represented by D_S in (10), where p_P is the attribute representing the plate's position, and p_S and l are the parameters representing the sensor's position and the length of plate, respectively. Each sensor in the FDPC can be represented as an instance of D_{SENSOR} by reconfiguring the parameter p_S.

The controller sends signals "1", "0", and "−1" to switch the system mode between "clockwise", "stop", and "anti-clockwise". The design of the controller can be specified as the model D_{CONTR} in (11), where sil_C and sir_C are the feedback signals from two end sensors; and so_C is the control signal.

Each driver unit starts to work when the plate reaches and stops when the plate leaves. It can be decomposed into

a switch, a rotor, a wheel, and two proximate sensors to reduce the complexity of design. The switch turns on/off (represented as "1"/ "0") the rotor according to the feedback signals from sensor, which indicates whether the plate is near the sensor. The rotor drives the plate moving with the friction between the wheel and the plate. The composition of the subsystems in a driver unit can be represented as $D_{DRIVERUNIT}$ shown in (12).

As shown in Figure 4, the subsystems in a driver unit are the second-level subsystems of the FDPC. The switch turns on the rotor and makes it rotate in the direction decided by the controller when any of the two sensors gives "1" signal. Otherwise, it turns off the rotor. The design of the switch can be represented as D_{SWITCH} in (13), where s_L, s_R, s_{dir}, and s_o are the signal from the left sensor, the signal from the right sensor, the direction signal from the controller, and the output signal, respectively. Assume that the power-torque relation is $t_R = \eta * P_e/w_R$, where t_R, P_e, w_R are the torque, nominal power and rotate speed. The design of rotor can be specified as D_{ROTOR} in (14), where s_{dir} is the signal stating rotate direction; $\eta < 1$ is the efficiency of the rotor. The wheel is driven by the rotor. It is designed as D_{WHEEL} in (15), where f_W, v_W, t_W, w_W are the frictions from the plate, the velocity, the drive torque, and the rotate speed, respectively. The sensors in a driver unit can be represented with the design model of the end sensors by reconfiguring the parameter p_S.

C. RESULTS ANALYSES

The design of the FDPC is specified with the relational models in the last section as our expectation. It is a 2-level system with a hierarchical structure in Figure 4. Its 1st-level consists of six subsystems, where the driver units can be decomposed into simpler subsystems in the 2nd-level.

Based on the relational model obtained in last section, some conclusions in general system science [51] can be adopted to analyze the structure of the designed FDPC, such as the connectivity of subsystems. Define the *connect degree* of two objects $O_1, O_2 \in A \cup M$ as $|R_c|$. $R_c = \left\{ r \in \hat{R}_P \mid \exists \alpha, \beta < n(r)\, r(\alpha) = O_1 \wedge r(\beta) = O_2 \right\}$ is the set of attribute relations between O_1 and O_2, where $r(\alpha)$, $r(\beta)$ is the αth, βth element in tuple of r. The connect degree measures the coupling of two subsystems, guiding how to decompose a system and assign tasks in a multi-designer design. With a dynamic structure, the connection degree between the 1st-level subsystems of the FDPC can be evaluated by counting attribute relations of each pair of subsystems in each system mode. The connect degree of each pair system (shown in Figure 6) indicates that the left/right driver unit and the plate couple more closely than other subsystems.

The behavior of the designed system D_{FDPC} can be illustrated by the hybrid automaton in Figure 5. Let E be the set of event expressions in D_{FDPC}. When an event in E is triggered, e.g., the plate arrives at driver units or ends of the rail, the hybrid automaton transmits from one behavior mode to another. Each behavior mode equals to a continuous system

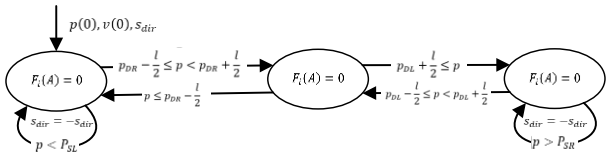

FIGURE 5. The hybrid automata illustrating behavior of designed FDPC. Given the initial conditions $p(0)$, $v(0)$, s_{dir}, the system evolves under the constraints of DEAs $F_i(A) = 0$, where $i \in 2^E$ is one mode of FDPC, E is the event set. When event $p < P_{SL}$ ($p > P_{SR}$) is triggered, the rotor starts to rotate clockwise (anticlockwise). When event $p_{DL} - l/2 \le p < p_{DL} + l/2$ ($p_{DR} - l/2 \le p < p_{DR} + l/2$) is triggered, the plate begins to receive friction from the left (right) drive wheel. When event $p_{DL} + l/2 \le p$ ($p \le p_{DR} - l/2$) is triggered, the plate is no longer subject to the friction from the left (right) drive wheel.

	A	B	C	D	E	F
A	-	1	1	1	1	0
B	1	-	0	0	0	1
C	1	0	-	0	0	3
D	1	0	0	-	0	1
E	1	0	0	0	-	1
F	0	1	3	1	1	-

	A	B	C	D	E	F
A	-	1	1	1	1	0
B	1	-	0	0	0	1
C	1	0	-	0	0	3
D	1	0	0	-	0	1
E	1	0	0	0	-	3
F	0	1	3	1	3	-

	A	B	C	D	E	F
A	-	1	1	1	1	0
B	1	-	0	0	0	1
C	1	0	-	0	0	1
D	1	0	0	-	0	1
E	1	0	0	0	-	3
F	0	1	1	1	3	-

FIGURE 6. Connect degree between 1st-level subsystems of FDPC. The map between the table headers and subsystems in S is as follows: A - "Controller", B - "SensorL", C - "DriverUnitL", D - "SensorR", E - "DriverUnitR", F - "Plate". The number in a cell represents the count of attribute relations between the row object and column object in a behavior mode. These three tables correspond to the three behavior modes in Figure 5.

following the DAEs $F_i(A)$, corresponding to a system mode of D_{FDPC}. Equations (16) to (18) are the DAEs at the modes when the plate is driven by the left driver unit, by both driver units, and by the right driver unit. Given an initial state $p(0)$, $v(0)$, and s_{dir}, the designed FDPC evolves as the hybrid automata states.

$$\begin{cases} \dot{p} = v \\ \dot{v} = a \\ fl - M * g * k_R - M * a = 0 \\ \eta * P_e - s_{dir} * t * w = 0 \\ t * w - f_{DL} * v = 0 \\ fl = -f_{DL} \end{cases} \tag{16}$$

$$\begin{cases} \dot{p} = v \\ \dot{v} = a \\ fl + fr - M * g * k_R - M * a = 0 \\ \eta * P_e - s_{dir} * t * w = 0 \\ t * w - f_{DL} * v = 0 \\ fl + f_{DL} = 0 \\ t * w - f_{DR} * v = 0 \\ fr + f_{DR} = 0 \end{cases} \tag{17}$$

$$\begin{cases} \dot{p} = v \\ \dot{v} = a \\ fr - M * g * k_R - M * a = 0 \\ \eta * P_e - s_{dir} * t * w = 0 \\ t * w - f_{DR} * v = 0 \\ fl = -f_{DR} \end{cases} \tag{18}$$

By transfinite induction, the hierarchical model of FDPC can be written as a primary dynamic system over

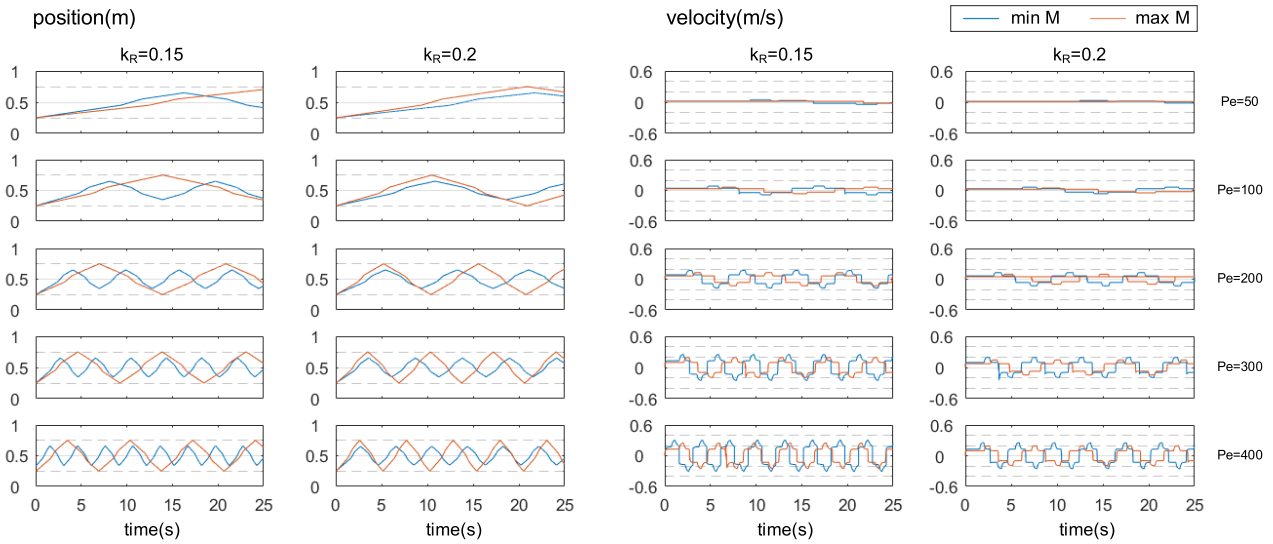

FIGURE 7. The dynamic position and velocity of the plate at different parameter values. The left two columns and the right two columns are the plots of the position p_P and velocity v_P at different value of parameter P_e and k_R. Each plot includes two lines indicates the dynamics at minimum load $M = 1500$ kg and maximum load $M = 2000$ kg. These plots are generated at the values of other parameters as, $= 9.8m \bullet s^{-2}$, $\eta = 0.95$, $p_{DL} = 0.3$ m, $p_{DR} = 0.7$ m, $p_{SL} = 0.15$ m, $p_{SR} = 0.85$ m, $l = 0.5$ m.

parameter set P. After eliminating simple equations of two attributes, the equations in attribute relations form three DAEs stitched by (16) to (18), with a parameter set $P = \{M, g, k_R, \eta, P_e, p_{DL}, p_{DR}, P_{SL}, P_{SR}, l\}$. The dynamics of each DAEs should satisfies the inequalities $0 \leq \mathrm{p} \leq 1$ and $0.05 \leq \mathrm{v} \leq 0.2$ constraining variables p and v. We can find the feasible parameter scheme by solving the DAEs with numerical solvers, such as Matlab and SciPy, and verify if it satisfies the constraints. A better feasible scheme can be obtained by comparing state performance. Generated by Matlab,[1] Figure 7 shows the dynamic position and velocity of the plate in designed FDPC at different values of parameter M, k_R, P_e. According the simulation result, when $(k_R, P_e) \in \{(0.15, 50), (0.15, 100), (0.2, 50), (0.2, 100), (0.2, 200), (0.2, 300)\}$, the position and velocity at both minimum and maximum loads satisfy the pre-defined constraints. Comparing these feasible parameter schemes, when $k_R = 0.2$, $P_e = 300$ W, the position and velocity have a stabler and more efficient performance. Therefore, the scheme of friction rate at 0.2 and the rotor with 300 W rate power is a better choice for practical implementation

VI. CONCLUSION

This paper proposed a relational abstraction named dynamic relational system to model the structure and behaviors of CPS during design. It has a mathematical definition and synthesizes the specification of complex structure and continuous-discrete hybrid behavior in an integrated manner. The dynamic relational model can imply the discrete processes in cyber space with the sequences of triggered events, and describe the continuous evolution in physical space with DAEs. A parametric representation is defined to specify the design schemes of CPS. A design case of FDPC is given to suggest that relational models can be used to specify the requirements and schemes at different abstraction level.

With well-defined semantics, the constructed models can be used as an abstraction of CPS design for structure analyses and behavior verification. The relational model has advantages of structure-oriented approach such as SysML on structure modeling, and hybrid approaches such as hybrid automata or hybrid program on hybrid behavior modeling. In other words, it is more expressive in hybrid behavior than SysML, and more expressive in system structure than hybrid automata and hybrid program. Compared with comprehensive modeling methods, such as Modelica, the relational abstraction has a rigorously defined mathematical semantics of structure and behavior, leading a good expression of discrete interactions in dynamic structure systems, such as the bouncing ball, and cellular net mentioned previously. In addition, the mathematical form discards many linguistic features in engineering, making it easier to understand and apply mathematical methods to structure analyses and behavior verification.

The relational representation provides a systemic perspective for the CPS design specification and leads a semantics foundation for the language and tool implementations. However, we still need further work to apply the relational models to achieve a practical CPS design. Future studies will focus on two aspects, i.e., how to analyze CPS structure with relational models and use it in practical CPS design (such as, defining a homeomorphism relation based on the dynamic relational system to realize model transformation and design case reasoning); and how to verify the proposed

[1]Simulation scripts and data are available at https://github.com/wangchustcad/FDPCSimulation.git

relational models and fit established verification tools to the models.

REFERENCES

[1] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1421–1434, Sep. 2017.
[2] P. J. Mosterman and J. Zander, "Cyber-physical systems challenges: A needs analysis for collaborating embedded software systems," *Softw. Syst. Model.*, vol. 15, no. 1, pp. 5–16, Feb. 2016.
[3] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A survey on concepts, applications, and challenges in cyber-physical systems," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 12, pp. 4242–4268, 2014.
[4] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 13–28, Jan. 2012.
[5] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyber physical systems: A survey," *IEEE Syst. J.*, vol. 9, no. 2, pp. 350–365, Jun. 2015.
[6] S. S. Bhattacharyya and M. C. Wolf, "Research challenges for heterogeneous cyberphysical system design," *Computer*, vol. 53, no. 7, pp. 71–75, Jul. 2020.
[7] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE Int. Symp. Object Component-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2008, pp. 363–369.
[8] P. Nuzzo, A. Iannopollo, S. Tripakis, and A. Sangiovanni-Vincentelli, "Are interface theories equivalent to contract theories?" in *Proc. 12th ACM/IEEE Conf. Formal Methods Models Codesign*, Oct. 2014, pp. 104–113.
[9] G. Pola and M. D. Di Benedetto, "Control of cyber-physical-systems with logic specifications: A formal methods approach," *Annu. Rev. Control*, vol. 47, pp. 178–192, 2019.
[10] R. Wisniewski, G. Bazydlo, P. Szczesniak, and I. Grobelna, "Design and verification of cyber-physical systems specified by Petri nets—A case study of a direct matrix converter," *Mathematics*, vol. 7, no. 9, p. 812, Sep. 2019.
[11] S. Bak, O. A. Beg, S. Bogomolov, T. T. Johnson, L. V. Nguyen, and C. Schilling, "Hybrid automata: From verification to implementation," *Int. J. Softw. Tools Technol. Transf.*, vol. 21, no. 1, pp. 87–104, Feb. 2019.
[12] H. S. Son, W. Y. Kim, R. Y. Kim, and H.-G. Min, "Metamodel design for model transformation from Simulink to ECML in cyber physical systems," in *Computer Applications for Graphics, Grid Computing, and Industrial Environment*. Berlin, Germany: Springer, 2012, pp. 56–60.
[13] G. Simko, T. Levendovszky, M. Maroti, and J. Sztipanovits, "Towards a theory for cyber-physical systems modeling," in *Proc. 4th ACM SIGBED Int. Workshop Desing, Modeling, Eval. Cyber-Phys. Syst.*, New York, NY, USA, 2014, pp. 56–61.
[14] S. Bliudze and J. Sifakis, "The algebra of connectors-structuring interaction in BIP," *IEEE Trans. Comput.*, vol. 57, no. 10, pp. 1315–1330, Oct. 2008.
[15] J. C. Willems, "The behavioral approach to open and interconnected systems," *IEEE Control Syst.*, vol. 27, no. 6, pp. 46–99, Dec. 2007.
[16] J. Fitzgerald, C. Gamble, P. G. Larsen, K. Pierce, and J. Woodcock, "Cyber-physical systems design: Formal foundations, methods and integrated tool chains," in *Proc. 3rd FME Workshop Formal Methods Softw. Eng.*, Piscataway, NJ, USA, 2015, pp. 40–46.
[17] P. Derler, E. A. Lee, S. Tripakis, and M. Törngren, "Cyber-physical system design contracts," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2013, pp. 109–118.
[18] Z. Cheng, C. Fulong, L. Chao, and Q. Xuemei, "XML-based component modeling and stimulation of cyber physical system," *J. Comput. Appl.*, vol. 39, no. 6, pp. 1842–1848, 2019.
[19] M. Hu, W. Duan, M. Zhang, T. Wei, and M. Chen, "Quantitative timing analysis for cyber-physical systems using uncertainty-aware scenario-based specifications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 4006–4017, Nov. 2020.
[20] L. Ordinez, G. Eggly, M. Micheletto, and R. Santos, "Using UML for learning how to design and model cyber-physical systems," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 15, no. 1, pp. 50–60, Feb. 2020.
[21] N. Bombieri, E. Ebeid, F. Fummi, and M. Lora, "On the reuse of heterogeneous IPs into SysML models for integration validation," *J. Electron. Test.*, vol. 29, no. 5, pp. 647–667, Oct. 2013.
[22] A. Wortmann, O. Barais, B. Combemale, and M. Wimmer, "Modeling languages in industry 4.0: An extended systematic mapping study," *Softw. Syst. Model.*, vol. 19, no. 1, pp. 67–94, Jan. 2020.
[23] J. Zeng, L. T. Yang, and J. Ma, "A system-level modeling and design for cyber-physical-social systems," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 2, pp. 1–26, Jun. 2016.
[24] S. S. Setty, H. Yaqoob, A. Malik, K. I.-K. Wang, Z. Salcic, H. Park, and U. D. Atmojo, "A unified framework for the design of distributed cyber-physical systems–industrial automation example," in *Proc. IEEE 10th Conf. Ind. Electron. Appl. (ICIEA)*, New York, NY, USA, Jun. 2015, pp. 1001–1007.
[25] N. Kaminski, E. Kusmenko, and B. Rumpe, "Modeling dynamic architectures of self-adaptive cooperative Systems.," *J. Object Technol.*, vol. 18, no. 2, p. 21, 2019.
[26] J. Liu, T. Li, Z. Ding, Y. Qian, H. Sun, and J. He, "AADL+: A simulation-based methodology for cyber-physical systems," *Frontiers Comput. Sci.*, vol. 13, no. 3, pp. 516–538, Jun. 2019.
[27] A. Malik, Z. Salcic, P. S. Roop, and A. Girault, "SystemJ: A GALS language for system level design," *Comput. Lang., Syst. Struct.*, vol. 36, no. 4, pp. 317–344, Dec. 2010.
[28] W. Dai, V. Vyatkin, C. Chen, and X. Guan, "Modeling distributed automation systems in cyber-physical view," in *Proc. IEEE 10th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2015, pp. 984–989.
[29] A. Platzer, "Differential-algebraic dynamic logic for differential-algebraic programs," *J. Log. Comput.*, vol. 20, no. 1, pp. 309–352, Feb. 2010.
[30] A. Platzer, "Choice & Control," in *Logical Foundations of Cyber-Physical Systems*. Cham, Switzerland: Springer, 2018.
[31] S. Zhou, Z. Yu, E. S. A. Nasr, H. A. Mahmoud, E. M. Awwad, and N. Wu, "Homomorphic encryption of supervisory control systems using automata," *IEEE Access*, vol. 8, pp. 147185–147198, 2020.
[32] Z. Yu, J. Ouyang, S. Li, and X. Peng, "Formal modeling and control of cyber-physical manufacturing systems," *Adv. Mech. Eng.*, vol. 9, no. 10, Oct. 2017, Art. no. 168781401772547.
[33] W. Dai, C. Pang, V. Vyatkin, J. H. Christensen, and X. Guan, "Discrete-Event-Based deterministic execution semantics with timestamps for industrial cyber-physical systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 3, pp. 851–862, Mar. 2020.
[34] N. Canadas, J. Machado, F. Soares, C. Barros, and L. Varela, "Simulation of cyber physical systems behaviour using timed plant models," *Mechatronics*, vol. 54, pp. 175–185, Oct. 2018.
[35] M. Amrani, D. Blouin, R. Heinrich, A. Rensink, H. Vangheluwe, and A. Wortmann, "Towards a formal specification of multi-paradigm modelling," in *Proc. ACM/IEEE 22nd Int. Conf. Model Driven Eng. Lang. Syst. Companion (MODELS-C)*, Sep. 2019, pp. 419–424.
[36] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proc. IEEE*, vol. 103, no. 11, pp. 2104–2132, Nov. 2015.
[37] A. Garro, A. Tundis, D. Bouskela, A. Jardin, N. Thuy, and M. Otter, "On formal cyber physical system properties modeling: A new temporal logic language and a Modelica-based solution," in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, Oct. 2016, pp. 1–8.
[38] E. Palachi, C. Cohen, and S. Takashi, "Simulation of cyber physical models using SysML and numerical solvers," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2013, pp. 671–675.
[39] M. Wolf and E. Feron, "What don't we know about CPS architectures?" in *Proc. 52nd Annu. Design Autom. Conf.*, Jun. 2015, p. 80.
[40] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-based design for cyber-physical systems," *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, Jan. 2012.
[41] R. Passerone, I. B. Hafaiedh, S. Graf, A. Benveniste, D. Cancila, A. Cuccuru, S. Gerard, F. Terrier, W. Damm, A. Ferrari, L. Mangeruca, B. Josko, T. Peikenkamp, and A. Sangiovanni-Vincentelli, "Metamodels in Europe: Languages, tools, and applications," *IEEE Des. Test. Comput.*, vol. 26, no. 3, pp. 38–53, May 2009.
[42] L. V. Nguyen, K. A. Hoque, S. Bak, S. Drager, and T. T. Johnson, "Cyber-Physical Specification Mismatches," *ACM Trans. Cyber-Phys. Syst.*, vol. 2, no. 4, p. 23, Sep. 2018.
[43] P. Bouyer, N. Markey, N. Perrin, and P. Schlehuber-Caissier, "Timed-automata abstraction of switched dynamical systems using control invariants," *Real-Time Syst.*, vol. 53, no. 3, pp. 327–353, May 2017.
[44] A. Platzer, "Differential dynamic logic for hybrid systems," *J. Automated Reasoning*, vol. 41, no. 2, pp. 143–189, Aug. 2008.

[45] I. Hasuo, "Metamathematics for systems design: Comprehensive transfer of formal methods techniques to cyber-physical systems," *New Gener. Comput.*, vol. 35, no. 3, pp. 271–305, Jul. 2017.
[46] H. Fang, H. Zhu, and J. Shi, "An object-oriented language for modeling of hybrid systems," in *Proc. IEEE 16th Int. Symp. High Assurance Syst. Eng. (HASE)*, Jan. 2015, pp. 1–9.
[47] A. Elsheikh, M. U. Awais, E. Widl, and P. Palensky, "Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface," in *Proc. Workshop Modeling Simulation Cyber-Phys. Energy Syst. (MSCPES)*, May 2013, pp. 1–6.
[48] P. Fritzson, "Modelica—A cyber-physical modeling language and the OpenModelica environment," in *Proc. 7th Int. Wireless Commun. Mobile Comput. Conf.*, Jul. 2011, pp. 1648–1653.
[49] T. Junjie, Z. Jianjun, D. Jianwan, C. Liping, X. Gang, G. Bin, and Y. Mengfei, "Cyber-physical systems modeling method based on modelica," in *Proc. IEEE 6th Int. Conf. Softw. Secur. Rel. Companion*, Jun. 2012, pp. 188–191.
[50] H. Ye, K. Liu, Q. Mou, and Y. Liu, "Modeling and formulation of delayed cyber-physical power system for small-signal stability analysis and control," *IEEE Trans. Power Syst.*, vol. 34, no. 3, pp. 2419–2432, May 2019.
[51] L. D. Xu, "The contribution of systems science to industry 4.0," *Syst. Res. Behav. Sci.*, vol. 37, no. 4, pp. 618–631, Jul. 2020.
[52] R. P. de Soares and A. R. Secchi, "Structural analysis for static and dynamic models," *Math. Comput. Model.*, vol. 55, nos. 3–4, pp. 1051–1067, Feb. 2012.
[53] P. Bunus and P. Fritzson, "Automated static analysis of equation-based components," *SIMULATION*, vol. 80, nos. 7–8, pp. 321–345, Jul. 2004.
[54] A. Cimatti, S. Mover, and S. Tonetta, "SMT-based scenario verification for hybrid systems," *Formal Methods Syst. Design*, vol. 42, no. 1, pp. 46–66, Feb. 2013.

CHAO WANG received the B.S. degree in mechanical design, manufacture, and automation, and the M.S. degree in mechanical design theory from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2011 and 2014, respectively, where he is currently pursuing the Ph.D. degree with the School of Mechanical Science and Engineering.

He has published two academic articles. His research interests include intelligent system design and knowledge-based design.

LI WAN received the B.S. degree in mechanical design, manufacture, and automation, and the Ph.D. degree in mechanical design theory from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1985 and 1995, respectively.

He is currently a Full Professor with the School of Mechanical Science and Engineering, HUST, where he is also the Deputy Director of the National CAD Support Software Engineering Research Center, Wuhan. He has published more than 50 articles. His research interests include advanced manufacture technology, collaborative design, and digital design and manufacture.

TIFAN XIONG received the B.S. degree in mechanical from the Dalian University of Science and Technology (DUST), Dalian, China, in 1995, and the M.S. degree in mechanical and the Ph.D. degree in mechanical design theory from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1998 and 2007, respectively.

He is currently a Lecturer with the School of Mechanical Science and Engineering, HUST. He has published more than 20 articles. His research interests include product lifecycle management (PLM), knowledge service in crowdsourcing design, resource planning and optimization, and industrial application of AR/VR.

YUANLONG XIE (Member, IEEE) received the B.S. degree in electrical engineering and the Ph.D. degree in mechanical engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2014 and 2018, respectively.

From 2017 to 2018, he was an Academic Visitor with the School of Electronic and Electrical Engineering, University of Leeds, Leeds, U.K. Since November 2018, he has been a Postdoctoral Fellow with HUST. He has published more than 60 academic journal and conference papers, one book, and held more than 15 patents. His research interests include robot control, servo control, filed-bus technology, and networked control systems.

Dr. Xie received the Best Student Paper Award at the 2020 IEEE Region 10 Conference.

SHUTING WANG received the B.S. and M.S. degrees from the School of Energy and Power Engineering, Wuhan University of Technology, Wuhan, China, in 1991 and 1994, respectively, and the Ph.D. degree from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, in 2002.

He is currently a Full Professor and the Vice-President with the School of Mechanical Science and Engineering, HUST, where he is also the Deputy Director of the National CAD Support Software Engineering Research Center, HUST. He has published more than 90 articles. His research interests include mobile robot, mechanical design, and numerical control.