An Adaptive Discrete Event Model for Cyber-Physical System

Ke Yue, Li Wang, Shangping Ren
Department of CS
Illinois Institute of Technology
Chicago, IL 60616
{kyue,lwang64,ren}@iit.edu

Xufei Mao
Beijing Key Laboratory of Intelligent
Telecommunications Software and Multimedia
Beijing University of Posts and Telecommunications
maoxufei@bupt.edu.cn

Xiangyang Li
Institute of Computer
Application Technology
Hangzhou Dianzi University
xli@cs.iit.edu

***Abstract*—Cyber-Physical Systems (CPS) often involve a wide spectrum of events, ranging from lower-level signals to higher-level abstract events. In order to compose different levels of events, a Concept Lattice-based Event (CLE) model was developed. In the CLE model, the traditional first order logic is used in specifying rules composition. However, it is possible that the rules specified by the first order logic may have inconsistency. Furthermore, unanticipated events that are not considered in the initial design may affect systems' performance, or even lead to system failure. In order to address these issues, an Adaptive Discrete Event (ADE) model is proposed in this paper. ADE model uses Discrete Event Calculus (DEC) to overcome possible inherent inconsistencies in composition rules that are specified by first order logic. In addition, we define abnormal event rules as an adaptive part in the CPS event model to handle unanticipated events. Finally, a CPS application "iLight" is developed based on the ADE model, results show that "iLight" adapts to the new environment successfully.**

***Index Terms*—Cyber-Physical Systems; discrete event model; adaptive model**

I. Introduction

Cyber-Physical Systems (CPS) often involve a large spectrum of events ranging from lower-level physical signals to higher-level abstract events. Correct system functioning relies on the understanding of these events and the ability to infer or reason about information from these events. However, due to the wide variety of events across different abstraction levels, defining an event model that not only reflects the abstraction diversity of events, but also, at the same time, provides an effective mechanism to reason about the events at different levels, becomes a challenging issue.

Commonsense reasoning [1] is an approach that enables a system to make inferences by utilizing events obtained from different levels. The functionality of commonsense reasoning is to infer, or reason about, what event will happen next, and how to react to such event. The foundation of the reasoning process lies in the rules designed in advance which are based on human everyday knowledge. For example, a human being cannot cross the wall and a broken bulb cannot light again. Both examples belong to the everyday knowledge used in people's daily commonsense reasoning process. Adopting such reasoning process into the CPS event model will make the system more responsive to the happenings of events, and we need to take it into consideration in our design procedure.

However, commonsense reasoning using first order logic [2] may contribute to inherent inconsistency. The crux of the inconsistency problem is that multiple event composition rules may be related to the same event. In other words, the same event may result in two different conclusions. To see a simple scenario where the reasoning result of an event is not consistent due to manifold reasoning rules, consider a smart home system with two lights. One rule may state that all the lights must be on at time point $t + 1$ when a person comes into the room at time point t. However, at the same time, to save energy, another rule states that if two lights are on at the same time at time point t, one of the light must be turned off at time point $t + 1$. Imagine a case that a person comes into the room at time point t, clearly, the two rules will result in contradiction with each other.

In a real world, environment states may not hold forever. Therefore, we need a mechanism to model environment changes and rules to specify how CPS application responds to such changes. Many existing event models, such as OWL-DL ontology based event model [3], OWL-S ontology based event model [4], CLE model [5], did not focus on the dynamic nature of environment in which a CPS usually operates. Environment changes will introduce a myriad of unanticipated events which may not be taken into consideration at the beginning of the design. These unconsidered events may impair the ability of a CPS to work properly and may even lead to system breakdown.

To overcome the issues discussed above, we have developed an Adaptive Discrete Event (ADE) model. The ADE model incorporates the Discrete Event Calculus (DEC) [6], [7], rather than first order logic, to compose events. DEC has been proved successful in solving inherent inconsistency problem [1]. Moreover, the abnormal reasoning set is adopted to dynamically have the unanticipated events. By providing corresponding mechanism for the unanticipated events, the

*The research of Shangping Ren is supported by National Science Foundation (NSF) under grant CNS-0746643 and CNS-1035894. The research of Xufei Mao is supported by the National Natural Science Foundation of China under Grant No. 60833009, the National 973 Project of China under Grant No.2011CB302700, the National Science Fund for Distinguished Young Scholars under Grant No.60925010. The research of Xiang-Yang Li is partially supported by NSF CNS-0832120, NSF CNS-1035894, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program).

CPS can be easily adapted to the new environment.

The main contributions of this paper are twofold: first, we introduce DEC in our Adaptive Discrete Event (ADE) model to define composition rules across vertical and horizontal domains. Predicates in DEC form the basis of solution to the inherent inconsistency problem existed in the traditional first order logic that was used in our previous concept-lattice based event model [8]. Second, the adaptability of CPS application is improved by introducing abnormal reasoning set which enables the CPS to handle unanticipated events.

The rest of this paper is organized as follows: Section II compares the related work. Section III formally defines the ADE model. The developed model is applied in a case study called "iLight", which is illustrated in Section IV. Finally, we conclude this paper and point out future work in Section V.

II. Related Work

The event concept has been investigated from different aspects. For instance, Mikhail et al. [9] define an event as a detectable action performed during the program execution, such as a statement execution, a procedure call, or a message transmission, etc. Luckham et al. [10] refer component interactions as event. Auguston's definition of event [11] is similar to the one in [10], but it is also used to form a behavior model for debugging and testing automation tools. As CPS applications are different from the applications above, they not only involve actions performed at the physical level, but also at network and computer level, even possibly at human level, the general event concepts mentioned above cannot accommodate such needs. One of our work in this paper is to develop a generalized event structure that can accurately specify events at different levels.

In order to handle events, work in [12] [13] uses first order logic to reason about the CPS events acquired by the system. However, first order logic cannot fully represent the semantic meaning of the commonsense reasoning. Mueller et al. [14] propose Discrete Event Calculus (DEC) to specify complex relationships among events. Although other computational approaches, such as situation calculus [15] and fluent calculus [16], are able to specify event relationships, DEC is superior because it addresses the rule inconsistency problem more efficiently. However, the mathematical definition of an event is not given in DEC.

Knowledge graph systems [17] are able to infer boolean values about the system state based on specific inputs and rules. However, knowledge graph systems do not focus on the dynamic aspects. Event models in previous work [8] [18] only focus on static and presumed events. However, in a real environment, unexpected events may happen at any time and may lead to system malfunction. So it is essential to design a dynamic and adaptive event model to model CPS applications so that they can be easily tailored to a new scenario.

III. Adaptive Discrete Event Model

For CPS applications, when and where an event occurs are crucial information. This is because we must know precedence order of different events to get temporal relationship among them. Therefore, in our early work [8], we introduced temporal and spatial information into event structure to completely capture the characteristics of CPS events. For self-completeness, we restate the definition in the following subsection.

A. CPS Event Instance

Definition 1 (CPS Event Instance): A CPS event instance consists of an event type and a set of attributes. It is structured as:

$$\mathcal{E}_{cps} : \Gamma\, \mu\, @\, (\mathcal{T},\, \mathcal{L},\, \mathcal{O}) \tag{1}$$

where,

- Γ represents the type name of the event instance.
- μ represents the attribute set of the event instances and has the form of $\mu = (\mu_1, \mu_2, \cdots, \mu_i)$ where $\mu_1, \mu_2, \cdots, \mu_i$ indicates the attribute of the event instance.
- $\mathcal{T}$ indicates the time point in reference to the event observer when the event instance happens.
- $\mathcal{L}$ indicates the location in reference to the event observer where the event instance occurs.
- $\mathcal{O}$ is an observer of the event instance. The existence of an observer is also treated as an event.
- $\mathcal{O}_\top$ is the global observer of a CPS system. Its location is the system's origin, and its time interval is defined as the system's life span. There is only one global observer in a given CPS system.

□

In summary, Definition (1) defines an event with time and location attributes detected by the observer.

B. Simplified Discrete Event Calculus

In the definition of Discrete Event Calculus (DEC), there are three types: event, fluent, and time point. The concept of *fluent* used in DEC is the time-varying properties of the world. Actually, as pointed out in [19], event and fluent can be considered to be the same if temporal attribute is added to event. Moreover, as our focus is on reasoning events, we restrict ourselves to the subset of DEC. More precisely, we define a simplified DEC with only the following primitives:

- *HoldsAt(e,t)*: This means that an event happens and holds at time point t.
- *Initiates(e_1,e_2,t)*: Event e_1 triggers another event e_2 to hold at time point $t+1$.
- *Terminates(e_1,e_2,t)*: Event e_1 triggers another event e_2 not to hold at time point $t+1$.
- *ReleasedAt(e,t)*: Event is released from the previous state at time point t. That is to say, the state of event e can be changed at time point t.
- *Releases(e_1,e_2,t)*: Event e_1 releases another event e_2 from the previous state at time point $t+1$. In other words, event e_1 will make the state of event e_2 at time point $t+1$.

Among these predicates, $ReleasedAt(e,t)$ and $Releases(e_1, e_2, t)$ are the crux of the solution to the

problem of inconsistency. The cause of the inconsistency problem is that an event's state may be affected by more than one rule simultaneously. By using these two predicates $ReleasedAt(e,t)$ and $Releases(e_1,e_2,t)$, an event's state can not be changed if it is not released from previous state. In other words, these predicates provide a lock mechanism for the event's state.

By depicting an event's state at a time point, the predicates above form the basis of our reasoning process. Based on these predicates, we define the rules set ℓ to compose events. CPS can make a deduction and control the system by using these rules. The example below illustrates how simplified DEC is used in our model.

Example 1: Let's assume a scenario in a smart home, if a person comes into the room and the light is dim, we should turn on the light. We define three events, ***PersonIn***, ***RoomDim***, ***TurnOnLight***. The formal definitions are listed below and spatial-temporal information is under the same observer $\mathcal{O}_1$.

$$\begin{aligned} PersonIn(height)@(\mathcal{T}_0,\mathcal{L}_0,\mathcal{O}_1)\\ RoomDim(lightstrength)@(\mathcal{T}_1,\mathcal{L}_1,\mathcal{O}_1)\\ TurnOnLight(lightid)@(\mathcal{T}_2,\mathcal{L}_2,\mathcal{O}_1)\end{aligned}$$

Based on ***PersonIn***, ***RoomDim*** events, we want the smart home system to generate a result event ***TurnOnLight*** and perform corresponding control. The reasoning rules using simplified DEC are defined as:

$$\begin{aligned} HoldsAt(PersonIn,t)\wedge HoldsAt(RoomDim,t)\Rightarrow\\ Initiates(PersonIn,TurnOnLight,t)\end{aligned} \quad (2)$$

$$\begin{aligned} Initiates(PersonIn,TurnOnLight,t)\wedge\\ ReleasedAt(TurnOnLight,t)\wedge HoldsAt(\mathcal{O}_1,t)\Rightarrow\\ HoldAt(TurnOnLight,t+1)\end{aligned} \quad (3)$$

Predicate $HoldsAt(\mathcal{O}_1,t)$ must be true at time point t, because if we want to obtain the precedence order among event occurrences, we need a common reference. Observer is defined as a common reference for the events happened under its observation. In addition, if we want to change the state of event ***TurnOnLight***, we must release it from the previous state first. The state of event $TurnOnLight$ can be changed only after explicitly indicating predicate $ReleasedAt(TurnOnLight,t)$. Now we get the formal reasoning rules as follows:

$$\ell=\{(2),(3)\} \quad (4)$$

Based on the reasoning rule set ℓ and input events ***PersonIn***, ***RoomDim***, we are able to make a deduction and get the event ***TurnOnLight***.

C. Adaptive Discrete Event Model Definition

When the circumstance in which a CPS is designed for is changed, new events which are not considered in the initial design may occur, making some predefined rules in the system invalid. Let's take the reasoning rule (3) as an example. Assume that the left three predicates, i.e., $Initiates(PersonIn,TurnOnLight,t)$, $ReleasedAt(TurnOnLight,t)$ and $HoldsAt(\mathcal{O}_1,t)$, are true. According to the reasoning rule (3), the predicate $HoldsAt(TurnOnLight,t+1)$ should hold at time point $t+1$, which indicates that we must take certain control mechanism to turn the light on at time point $t+1$. However, what if the light is not on? It is intuitive to make an inference that there must exist some unexptected events which lead to the system failure, for instance, the bulb is broken. We call these unexpected events as abnormal events. There might be numerous events belonging to abnormal events because CPS failure can be caused by a wide variety of events. In addition, abnormal events usually happen when a system operates in a new environment. Therefore, rule (3) may be not suitable if abnormal events exist. We must change the rules to accommodate various envrionments and make CPS applications more adaptive. In order to achieve this goal, we introduce an abstract concept event $\mathcal{E}_{ab}$ to generalize all the abnormal events. The concept event $\mathcal{E}_{ab}$ means that the CPS's output event is not consistent with initial expected event.

Initially, the predicate $HoldsAt(\mathcal{E}_{ab},t)$ is false and we assume that our CPS works properly. The occurrence of abnormal events will make predicate $HoldsAt(\mathcal{E}_{ab},t)$ to be true. So we have to take certain mechanism to make $HoldsAt(\mathcal{E}_{ab},t)$ false and prevent CPS from crashing. Hence, we add the predicate $HoldsAt(\mathcal{E}_{ab},t)$ to our previous rule (3) and get the new rule:

$$\begin{aligned} Initiates(PersonIn,TurnOnLight,t)\wedge\\ \neg HoldsAt(\zeta.\mathcal{E}_{ab},t)\wedge\\ ReleasedAt(TurnOnLight,t)\wedge\\ HoldsAt(\mathcal{O}_1,t)\Rightarrow\\ HoldsAt(TurnOnLight,t+1)\end{aligned} \quad (5)$$

In this new rule, $HoldsAt(\mathcal{E}_{ab},t)$ must be guaranteed to be false to get the expected event and make $HoldsAt(TurnOnLight,t+1)$ true at next time point.

When the system detects that a light is not on when a person is coming in, from the above rule, we can infer that an abnormal event, such as, the bulb is broken, may have occured. In other words, the abnormal event that the bulb is broken leads to system failure:

$$HoldsAt(BulbBroken,t)\Rightarrow HoldsAt(\mathcal{E}_{ab},t) \quad (6)$$

The new rule shows that we must ensure $HoldsAt(BulbBroken,t)$ to be false to make $HoldsAt(\mathcal{E}_{ab},t)$ false. In terms of implementation, we must provide a mechanism, i.e., providing backup bulb, to prevent the event that a bulb is broken from bringing the system down. We can add similar rules which may make

$HoldsAt(\mathcal{E}_{ab}, t)$ true and prepare corresponding mechanisms to handle those abnormal events. By adding more abnormal event rules to the system, the chance that CPS is affected by abnormal events is decreased. Thus, the system adaptability is enhanced by utilizing this methodology.

We call the rules designed for abnormal events as abnormal rules set γ. In this case, γ={(6)}. By combining the abnormal rules set γ and $\mathcal{E}_{ab}$, we get the abnormal reasoning set ζ. We add ζ to the observer definition, so each observer has its own abnormal reasoning set. Now we define the observer's formal mathematical structure as follows:

Definition 2 (CPS Observer Event Instance): A CPS observer event instance is composed of an event type and a set of attributes. It is structured as:

$$\mathcal{E}_{obs} : \Gamma\mu@(\mathcal{T}, \mathcal{L}, \mathcal{O}) \tag{7}$$

where,

- Γ, $\mathcal{T}$, $\mathcal{L}$ have the same meaning with those in Definition 1.
- μ represents the attribute set of the event instances, and we add abnormal reasoning set ζ to this attribute set. $\zeta = \{\gamma, \mathcal{E}_{ab}\}$. ζ is a set containing observer event's abnormal event reasoning rules γ and abnormal event $\mathcal{E}_{ab}$.
- $\mathcal{O}$ is an observer providing observer event, the existence of an observer is also treated as an event. For example, a sensor node can act as an observer under which all the events have the same relative geographical and temporal coordinates. This sensor is also observed by another sensor acting as an observer. Although several observers are allowed in a CPS system, the global observer $\mathcal{O}_\top$ must be unique and serves as the system's coordinate and wall-clock.

□

Based on the discussions above, we provide the definition of Adaptive Discrete Event model for CPS as follows:

Definition 3 (ADE Model): An Adaptive Discrete Event (ADE) Model is a structure:

$$M = \langle X, Y, O, E, \ell \rangle, \tag{8}$$

where,

- X is the set of all the input events.
- Y is the set of all the output events.
- O is observer event sets.
- E is the set of all the events.
- ℓ is the event reasoning rules set.

□

In the next section, we will design an ADE model for the CPS application "iLight". The events and the rules will be explained in detail in the "iLight"system.

IV. An Application for ADE Model

The "iLight" is a passive tracking Cyber-Phsical System. It is used to track a moving target by using multiple groups of sensors, and automatically compute the target's moving patterns such as height, moving speed, etc. Sensors will get raw data which are deemed as input events, and based on these events, the system can make a deduction regarding whether there is a person coming in, whether he/she is an adult or child, and judge whether he/she is walking or running. When "iLight" system is migrated to a new environment, unexpected events may prevent the system from functioning properly. Therefore, for the purpose of adaptability, rules defined in the observer event can add or delete abnormal events dynamically, making "iLight" adaptive to different environments.

A. "iLight" System Background

The "iLight" tracking system uses light sensors and light sources to track moving objects. Light sensors are sensitive to the light change around them. When a moving object comes across the light source and the light sensor, the light level (photo value) will decrease sharply as Figure 1 illustrates. Once a person comes across at 10 second, the photo value sensed by the light sensor decreases from 50 to less than 20. By detecting such event, we can make a simple deduction that a person is coming in.

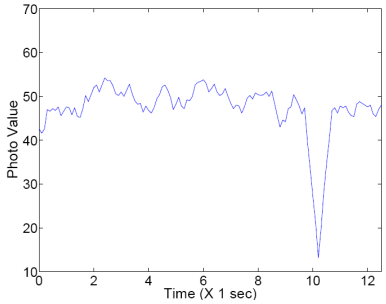

Fig. 1. Photo value results

As shown in Figure 2, two groups of sensors $\{A_1, A_2, A_3, A_4\}$ and $\{B_1, B_2, B_3, B_4\}$ are deployed on either side of the wall. Moreover, a and b are two light resources. Once a moving object with height h goes between two groups of sensors, the photo values of sensors $A_2, A_3, A_4, B_2, B_3, B_4$ will decrease dramatically and the photo value of A_1, B_1 will be almost the same as before. Now segments aB_1, aB_2, bA_1, bA_2 will intersect at point C, D, E, F which form a quadrilateral. As the distance between the highest point of the person and intersection of CF, DE is very small, we can regard these two points as the same. As the coordinates of vertices of a, b, A_1, A_2, B_1, B_2 are known, it is easy to calculate the coordinate of intersection between CF and DE. Hence the moving object's height can be calculated.

The "iLight"Test bed includes 41 wireless sensor nodes and one of them is chosen as sink node and global observer. Ten light sources are installed to provide light, and a base station is used to monitor the data. 40 sensors are divided into 10 groups and each group has 4 sensors respectively. One sensor is randomly chosen from the group acting as local observer. Two groups of sensors are deployed on the opposite side of the wall as shown in Figure 3. Each black node refers to one group of sensor nodes. Red nodes in the graph refer to the light sources. The red dash lines indicate the laser-like narrow

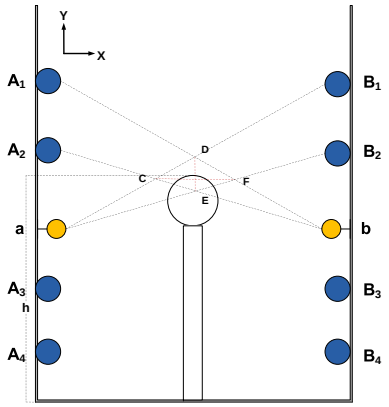

Fig. 2. Blue Block nodes denote two group of sensors where a and b are light sources.

low-divergence light beams. The black curve is the trajectory of the moving object. Little black circle is the location where target is detected by the light sensors.

In addition, under certain situations, the sensor may behave abnormally, and the photo value may decrease even there is no person coming across due to the hardware constraints and background noise ($\mathcal{E}_{ab}$). Light sensor will report such event due to error reading especially when there is too much background noise. So we can use 10 groups of sensors and every two groups deployed on the opposite side of the wall to monitor the events. Only two sensors' photo value decrease within same range, can there be a real event happening, and the alarm event will be triggered to report an intruder.

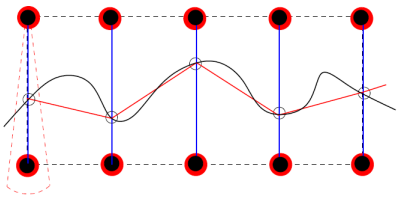

Fig. 3. Vertical views of the monitored area

Table 1 summarizes the main parameters of the "iLight" tracking system. For example, we use five people with different heights as the tracked targets. In this case, we let each target go through the monitored area and the system must provide a concrete event description from the sensed datum.

Based on the discussion above, we now begin to design the ADE model. Based on this model, CPS must be able to accurately detect an intruder based on the events derived from the light sensors. In order to be more accurate, the decision regarding whether there is an intruder must be made based on the results of two groups G_1 and G_2. One group of sensors are deployed in the way shown in Figure 2. We also define the sensors with the highest position whose light values changed as A_2^1, B_2^1 in group G_1 and A_2^2, B_2^2 in G_2. The defined events are as follows:

Parmeter	Value
Sensors per group	4
Distance between two adjacent sensors in the same group	30 cm
Distance between two groups at the same side	2.5 meters
Distance between two groups on the opposite side of the wall	3 meters
targets(Heights)	A(165 cm),B(170 cm),C(175 cm),D(180 cm),E(185 cm)
Hight of a light source	160 cm
Moving Speed	0.5 and 1 meter/sec

TABLE I
"ILIGHT" SYSTEM PARAMETERS

- $A_2^1 GSenDecrease\ (id, PValDed, coord)\ @\ (\mathcal{T}_0, \mathcal{L}_0, \mathcal{O}b_0)$
- $B_2^1 GSenDecrease\ (id, PValDed, coord)\ @\ (\mathcal{T}_1, \mathcal{L}_1, \mathcal{O}b_1)$
- $A_2^2 GSenDecrease\ (id, PValDed, coord)\ @\ (\mathcal{T}_2, \mathcal{L}_2, \mathcal{O}b_2)$
- $B_2^2 GSenDecrease\ (id, PValDed, coord)\ @\ (\mathcal{T}_3, \mathcal{L}_3, \mathcal{O}b_3)$
- $G_1 PDecrease\ (id, height)\ @\ (\mathcal{T}_4, \mathcal{L}_4, \mathcal{O}b_4)$
- $G_2 PDecrease\ (id, height)\ @\ (\mathcal{T}_5, \mathcal{L}_5, \mathcal{O}b_5)$
- $G_1 PersonIn\ (id, height)\ @\ (\mathcal{T}_6, \mathcal{L}_6, \mathcal{O}b_4)$
- $G_2 PersonIn\ (id, height)\ @\ (\mathcal{T}_7, \mathcal{L}_7, \mathcal{O}b_5)$
- $PeopleWalkIn\ (speed)\ @\ (\mathcal{T}_8, \mathcal{L}_8, \mathcal{O}_\top)$
- $PeopleIsAdult\ (height)\ @\ (\mathcal{T}_9, \mathcal{L}_9, \mathcal{O}_\top)$
- $SinglePersonIn\ (speed, height)\ @\ (\mathcal{T}_{10}, \mathcal{L}_{10}, \mathcal{O}_\top)$
- $Alarm\ (soundstrength)\ @\ (\mathcal{T}_{11}, \mathcal{L}_{11}, \mathcal{O}_\top)$

Property *id* of each event refers to the id number assigned to each group. *PValDed* is the decreased photo value and *coord* is the coordination needed to calculate the top point of a human being who comes across the light sensor's detected area. Property *height*, *speed* refers to the human being's height and speed. The observer is used to synchronize the time among the sensors.

- $\mathcal{O}b_0(\zeta_0, id)@(\mathcal{T}_0, \mathcal{L}_0, \mathcal{O}b_4)$
- $\mathcal{O}b_1(\zeta_1, id)@(\mathcal{T}_1, \mathcal{L}_1, \mathcal{O}b_4)$
- $\mathcal{O}b_2(\zeta_2, id)@(\mathcal{T}_2, \mathcal{L}_2, \mathcal{O}b_5)$
- $\mathcal{O}b_3(\zeta_3, id)@(\mathcal{T}_3, \mathcal{L}_3, \mathcal{O}b_5)$
- $\mathcal{O}b_4(\zeta_4, id)@(\mathcal{T}_4, \mathcal{L}_4, \mathcal{O}_\top)$
- $\mathcal{O}b_5(\zeta_5, id)@(\mathcal{T}_5, \mathcal{L}_5, \mathcal{O}_\top)$
- $\mathcal{O}_\top = obs(\zeta_6)@([0, \infty), ((0,0,0), \infty), \top)$

In this case, $\top$ denotes the system itself and *obs* denotes this observer's name. Property *id* is the id number assigned to each sensor which acts as an observer. First, we assign a value for the decreased photo value sensed by the sensors, say between 30 to 50. Now, based on these events, we define the rules as follows:

$$30 \leq value \leq 50 \wedge HoldsAt(\mathcal{O}b_0, t) \Rightarrow HoldsAt(A_2^1 GSenDecrease, t) \quad (9)$$

$$30 \leq value \leq 50 \wedge HoldsAt(\mathcal{O}b_1, t) \Rightarrow HoldsAt(B_2^1 GSenDecrease, t) \tag{10}$$

$$30 \leq value \leq 50 \wedge HoldsAt(\mathcal{O}b_2, t) \Rightarrow HoldsAt(A_2^2 GSenDecrease, t) \tag{11}$$

$$30 \leq value \leq 50 \wedge HoldsAt(\mathcal{O}b_3, t) \Rightarrow HoldsAt(B_2^2 GSenDecrease, t) \tag{12}$$

$$\begin{aligned} &HoldsAt(A_2^1 GSenDecrease, t) \wedge \\ &HoldsAt(B_2^1 GSenDecrease, t) \wedge \\ &HoldsAt(\mathcal{O}b_4, t) \wedge \neg HoldsAt(\zeta_4.\mathcal{E}_{ab}, t) \Rightarrow \\ &HoldsAt(G_1 PDecrease, t) \end{aligned} \tag{13}$$

$$\begin{aligned} &HoldsAt(A_2^2 GSenDecrease, t) \wedge \\ &HoldsAt(B_2^2 GSenDecrease, t) \wedge \\ &HoldsAt(\mathcal{O}b_4, t) \wedge \neg HoldsAt(\zeta_5.\mathcal{E}_{ab}, t) \Rightarrow \\ &HoldsAt(G_2 PDecrease, t) \end{aligned} \tag{14}$$

$$\begin{aligned} &HoldsAt(G_1 PDecrease, t) \wedge \\ &Initiate(G_1 PDecrease, G_1 PersonIn, t) \wedge \\ &ReleaseAt(G_1 PersonIn, t) \Rightarrow \\ &HoldsAt(\mathcal{O}b_4, t) \Rightarrow \\ &HoldsAt(G_1 PersonIn, t+1) \end{aligned} \tag{15}$$

$$\begin{aligned} &HoldsAt(G_2 PDecrease, t) \wedge \\ &Initiate(G_2 PDecrease, G_2 PersonIn, t) \wedge \\ &ReleaseAt(G_2 PersonIn, t) \wedge \\ &HoldsAt(\mathcal{O}b_4, t) \Rightarrow \\ &HoldsAt(G_2 PersonIn, t+1) \end{aligned} \tag{16}$$

$$\begin{aligned} &HoldsAt(G_1 PersonIn, t) \wedge \\ &HoldsAt(G_2 PersonIn, t) \wedge \\ &ReleaseAt(SinglePersonIn, t) \wedge \\ &(G_1 PersonIn.height = G_2 PersonIn.height) \wedge \\ &HoldsAt(\mathcal{O}_\top, t) \Rightarrow \\ &HoldsAt(SinglePersonIn, t) \end{aligned} \tag{17}$$

$$\begin{aligned} &(HoldsAt(G_1 PersonIn, t) \vee \\ &HoldsAt(G_2 PersonIn, t)) \wedge \\ &ReleaseAt(PeopleWalkIn, t) \wedge \\ &((G_1 PersonIn.speed \leq 1) \vee \\ &(G_2 PersonIn.speed \leq 1)) \wedge \\ &HoldsAt(\mathcal{O}_\top, t) \Rightarrow \\ &HoldsAt(PeopleWalkIn, t) \end{aligned} \tag{18}$$

$$\begin{aligned} &(HoldsAt(G_1 PersonIn, t) \vee \\ &HoldsAt(G_2 PersonIn, t)) \wedge \\ &ReleaseAt(PeopleIsAdult, t) \wedge \\ &((G_1 PersonIn.height \geq 165) \vee \\ &(G_2 PersonIn.height \geq 165)) \wedge \\ &HoldsAt(\mathcal{O}_\top, t) \Rightarrow \\ &HoldsAt(PeopleIsAdult, t) \end{aligned} \tag{19}$$

$$\begin{aligned} &(HoldsAt(G_1 PersonIn, t) \vee \\ &HoldsAt(G_2 PersonIn, t)) \wedge \\ &ReleaseAt(Alarm, t) \wedge \\ &HoldsAt(\mathcal{O}_\top, t) \Rightarrow \\ &(Initiate(G_1 PersonIn, Alarm, t) \vee \\ &Initiate(G_2 PersonIn, Alarm, t)) \end{aligned} \tag{20}$$

Based on the rules above, we get the composition rule set:

$$\ell = \{(9), (10), (11), (12), (13), (14), (15), (16), (17), (18), (19), (20), \}$$

Consider the case that we need to migrate "iLight" to outdoor environment where CPS may malfunction due to severe environment. We may find that the CPS gives an alarm when nobody is there. To find what event makes the system malfunction, CPS begins to diagnose itself. In our example, the CPS may compare the data obtained from different sensors and find some sensor's data is different from the majority. Hence it may make a conclusion that this sensor is fake. Then new abnormal event *SensorFake* is added to CPS. Because the *SensorFake* event happens, the $HoldsAt(\mathcal{E}_{ab}, t)$ will be true indicating that this sensor is broken and the alarm will not sound anymore.

$$HoldsAt(SensorFake) \Rightarrow HoldsAt(\zeta_4.\mathcal{E}_{ab}, t) \tag{21}$$

$$HoldsAt(SensorFake) \Rightarrow HoldsAt(\zeta_5.\mathcal{E}_{ab}, t) \tag{22}$$

Therefore, we get abnormal reasoning set $\zeta = \{(21), (22), \mathcal{E}_{ab}\}$.

Now we define the event set used in our ADE CPS model. The event set E is:

$$\begin{aligned} E = (&A_2^1 GSenDecrease, B_2^1 GSenDecrease, G_1 PersonIn, \\ &A_2^2 GSenDecrease, B_2^2 GSenDecrease, G_2 PersonIn, \\ &G_1 PDecrease, G_2 PDecrease, PeopleWalkIn, Alarm \\ &PeopleIsAdult, SinglePersonIn, \mathcal{O}b_0, \mathcal{O}b_1, \\ &\mathcal{O}b_2, \mathcal{O}b_3, \mathcal{O}b_4, \mathcal{O}b_5, \mathcal{O}_\top) \end{aligned}$$

The input events set X is:

$$\begin{aligned} X = (&A_2^1 GSenDecrease, B_2^1 GSenDecrease, \\ &A_2^2 GSenDecrease, B_2^2 GSenDecrease) \end{aligned}$$

The output events set Y is:

$$Y = (PeopleWalkIn, PeopleIsAdult, \\ SinglePersonIn, Alarm)$$

The observer events set O is

$$O = (\mathcal{O}b_0, \mathcal{O}b_1, \mathcal{O}b_2, \mathcal{O}b_3, \mathcal{O}b_4, \mathcal{O}b_5, \mathcal{O}_\top)$$

Therefore, the ADE model is $M = \langle X, Y, O, E, \ell \rangle$, and the model is illustrated in Figure 4.

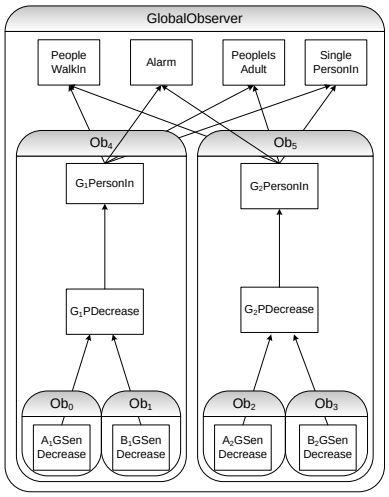

Fig. 4. ADE model

V. Conclusion

In this paper, we present an Adaptive Discrete Event (ADE) model for Cyber-Physical System (CPS) to deal with the rule inconsistency problem and unexpected events issue. In particular, a simplified DEC to compose different levels of events is used, and we further introduce abstract concept event to capture unexpected events caused in heterogeneous environments in which a CPS application operates. Moreover, abnormal reasoning set is added into the common sense reasoning rules to makes the system adaptive to different environments. In our current work, the abductive process for system to perform the self diagnosis is, nevertheless, not formulated in the rules, and it is our future research focus.

References

[1] E. Mueller, *Commonsense reasoning*. Morgan Kaufmann, 2006.

[2] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 1998, pp. 37–45.

[3] V. Ermolayev, N. Keberle, and W.-E. Matzke, "An ontology of environments, events, and happenings," in *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, jul. 2008, pp. 539 –546.

[4] W. Mepham and S. Gardner, "Implementing discrete event calculus with semantic web technologies," in *Next Generation Web Services Practices, 2009. NWESP '09. Fifth International Conference on*, sep. 2009, pp. 90 –93.

[5] Y. Tan, M. C. Vuran, S. Goddard, Y. Yu, M. Song, and S. Ren, "A concept lattice-based event model for cyber-physical systems," in *ICCPS '10: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*. New York, NY, USA: ACM, 2010, pp. 50–60.

[6] M. Shanahan, "The event calculus explained," *Artificial intelligence today*, pp. 409–430, 1999.

[7] ——, "An abductive event calculus planner," *The Journal of Logic Programming*, vol. 44, no. 1-3, pp. 207–240, 2000.

[8] Y. Yu, M. Song, S. Ren, C. Hood, J. Zhu, and G. Quan, "Real-Time Process Control in Producing Clean Air and Bio-Energy from Animal Waste," *Work-In-Progress Proceedings*, p. 93, 2009.

[9] M. Auguston, "Software architecture built from behavior models," *ACM SIGSOFT Software Engineering Notes*, vol. 34, no. 5, pp. 1–15, 2009.

[10] D. Luckham and M. Augustin, "Specification and analysis of system architecture using Rapide," *IEEE Transactions on Software Engineering*, vol. 21, no. 4, 1995.

[11] M. Auguston, "Program behavior model based on event grammar and its application for debugging automation," in *AADEBUG, 2nd International Workshop on Automated and Algorithmic Debugging*. Citeseer, 1995, pp. 277–291.

[12] M. Bujorianu, M. Bujorianu, and H. Barringer, "A unifying specification logic for cyber-physical systems," in *Control and Automation, 2009. MED '09. 17th Mediterranean Conference on*, jun. 2009, pp. 1166 – 1171.

[13] M. Bujorianu and H. Barringer, "An integrated specification logic for cyber-physical systems," in *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on*, jun. 2009, pp. 291 –300.

[14] E. T. Mueller, "Automating commonsense reasoning using the event calculus," *Commun. ACM*, vol. 52, no. 1, pp. 113–117, 2009.

[15] V. Singh and R. Jain, "Situation based control for cyber-physical environments," in *Military Communications Conference, 2009. MILCOM 2009. IEEE*, oct. 2009, pp. 1 –7.

[16] R. Kowalski and M. Sergot, "A logic-based calculus of events," *New generation computing*, vol. 4, no. 1, pp. 67–95, 1986.

[17] G. Sullivan, "A knowledge-based control architecture with interactive reasoning functions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 8, no. 1, pp. 179 –183, feb. 1996.

[18] M. Rohl and A. Uhrmacher, "Definition and analysis of composition structures for discrete-event models," in *Simulation Conference, 2008. WSC 2008. Winter*, dec. 2008, pp. 942 –950.

[19] R. Kowalski and F. Sadri, "The situation calculus and event calculus compared," in *ILPS '94: Proceedings of the 1994 International Symposium on Logic programming*. Cambridge, MA, USA: MIT Press, 1994, pp. 539–553.