A Concept Lattice-based Event Model for Cyber-Physical Systems

Ying Tan
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588
yingtan@cse.unl.edu

Mehmet C. Vuran
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588
mcvuran@cse.unl.edu

Steve Goddard
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588
goddard@cse.unl.edu

Yue Yu
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
yyu8@iit.edu

Miao Song
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
msong8@iit.edu

Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
ren@iit.edu

ABSTRACT

Cyber-Physical Systems (CPS) involve communication, computation, sensing, and actuating through heterogeneous and widely distributed physical devices and computational components. The close interactions of these systems with the physical world places *events* as the major building blocks for the realization of CPS. More specifically, the system components and design principles should be revisited with a strictly *event-based* approach. In this paper, a concept lattice-based event model for CPS is introduced. Under this model, a CPS event is uniformly represented by three components: *event type*, *its internal attributes*, and *its external attributes*. The internal and external attributes together characterize the type, spatio-temporal properties of the event as well as the components that observe it. A set of event composition rules are defined where the CPS event composition is based on a *CPS concept lattice*. The resulting event model can be used both as an *offline analysis tool* as well as a *run-time implementation model* due to its distributed nature. A real-life smart home example is used to illustrate the proposed event model. To this end, a CPS event simulator is implemented to evaluate the developed event model and compare with the existing Java implementation of the smart home application. The comparison result shows that the event model provides several advantages in terms of flexibility, QoS support, and complexity. The proposed event model lay the foundations of event-based system design in CPS.

Categories and Subject Descriptors

D.4.7 [**Organization and Design**]: Real-time and embedded systems

General Terms

Theory, Design, Languages

Keywords

Cyber-physical systems, event modeling, temporal and spatial event conditions, CPS architecture

1. INTRODUCTION

The Cyber-Physical Systems (CPSs) are envisioned as heterogeneous *systems of systems*, which involve communication, computation, sensing, and actuating through heterogeneous and widely distributed physical devices and computation components [17]. The components of a CPS are connected through wired and wireless networks in a large scale and orchestrated together as a whole. Moreover, CPS introduces several challenges for system design: (1) to support high system flexibility such that the CPS components in the system are free to join or leave dynamically, (2) to support various Quality of Services (QoS) requirements through out *every level* of CPSs. For example, a deadline (i.e., a time-related QoS requirement) on a control-loop in a CPS indicates that when an event of interest occurs in the physical world: firstly, it has to be sensed and detected by certain CPS components in the cyber world; secondly, appropriate actuation decisions should be taken by distributed system components, and lastly, an actuation task needs to be carried out by an actuator in the physical world, all within a limited time frame. The timing constraints for each individual component varies because of the non-deterministic system delay for sensing, computation, communication, and actuation, which becomes a major verification challenge. Due to the close interactions with the physical world, such constraints can be addressed through an *event-based approach*, i.e., using events as the units in CPS for computation, communication, and control [30] [31]. In this work, we refer to the occurrence of interests in a CPS system encompassed by the cyber world and the physical world as a *CPS event*. This paper extends our previous result on formalizing the event model for the CPS [31].

Event-based system design has been studied in various areas. However, existing approaches for event-based design such as data-centric event modeling used in database applications [22], or temporal-order-centric event modeling [18, 3] in distributed applications cannot be directly applied to CPS applications. This is because in traditional system design, the event models generally maintain a consistent view about time and space with respect to a single entity. A CPS, however, is characterized by spatio-temporal information as well as a distributed set of components that operate in different reference frames. Moreover, due to its inherent heterogeneity and distributed nature, a common frame-of-reference does not exist in CPS. To address the distributed and open nature of CPS, in this paper, we define a *CPS event model*, which incorporates the spatio-temporal attributes and observer information into the event definition.

In addition, events in CPS range from lower-level, physical sensing and actuating events to higher-level, human/machine understandable cyber events. To provide seamless interactions between heterogeneous components and devices in cyber and physical domains, a unified representation of *events* is defined. Accordingly, a systematic mechanism is developed to compose CPS events to and from different levels and across different system boundaries. The resulting event model can be used both as an *offline analysis tool* as well as a *run-time implementation model* due to its distributed nature.

The main contributions of the paper are twofold. First, a unified event structure that represents CPS event instance at different layers is defined. Accordingly, a CPS event instance consists of three components: event type, its internal and external attributes. Together, they describe *when* and *where* the event instance is observed to occur and its observer. Furthermore, each observer, such as a sensor, is also defined as a CPS event, which enables observers to dynamically join and leave the CPS at run-time. Second, a formal mechanism is defined for composing CPS events from lower-level events by applying and extending the theory of *concept lattice* [21, 32][1]. To this end, a set of composition functions are introduced to accommodate the temporal and spatial constraints in event composition as well.

The rest of the paper is organized as follows: In Section 2, recent solutions on event modeling in various contexts are reviewed. The unified CPS event structure is introduced and the related concepts are described in Section 3. In Section 4, we discuss the concept-lattice-based CPS event model and event composition. The developed model is evaluated in Section 5 through a case study, where a smart home system is implemented through the event model. We conclude the paper and point out future work in Section 6.

2. RELATED WORK

The concept of events has been investigated in several contexts both within the cyber domain and the physical domain. For instance, the *Event-Condition-Action (ECA)* model is introduced in [22], in which *event* specifies the signal that triggers the evaluation of the *condition* and if true, causes an *action* to be carried out. In the ECA model, actions are triggered by independent events. Extensions to the ECA model [11, 9, 4] introduce a set of event operators to compose events so that composite events can be described. SnoopIB [2] further considers event occurrences in the time domain as intervals (interval events), rather than time points (punctual events). The spatial relationships between different events are studied in [1, 7]. The real-time community aims to add timing constraints to the event-condition-action model. For example, the Real-Time Logic (RTL)-based event model has been proposed with point- and interval-based timing constraints in [23, 24, 34], respectively. Timing constraints in RTL-based event models define the time point-based real-time relationships among the occurrence time of events.

[1]*Concept lattice* (Galois lattice) is a conceptual hierarchical structure based on binary relation proposed by Rudolf Wille [33, 32]. The theory has been widely used in the fields of software engineering [20, 28, 29] and data mining [12, 5, 6].

An event-based approach is adopted to describe interested properties of a running program [18, 3]. The interested program properties, such as safety and liveness, are defined as temporal occurrence patterns of events. For example, Java-MAC [13, 14] uses Linear Temporal Logic (LTL) for Java program run-time monitoring, where events occur instantaneously during system execution and conditions represent information that hold for a duration of time. The event calculus [15, 25, 8] investigates a logic program framework for representing and reasoning about events (or actions) and their effects. Under this framework, time-varying properties (true or false) of the world during certain intervals, called fluents, are initiated by an occurrence of an action continue to (or not to) hold until an occurrence of an action which terminates them.

In most of the solutions mentioned above, there is an implicit assumption that the observer of an event is unique and global, which is the system, or the program. Therefore, to most, the time and spacial information are associated with an event, its observer is nevertheless omitted or is taken as a default 'system'. In distributed computing, event observers are different, however, the observers are interested in the same set of events and the goal is to obtain a consistent view about the ordering of these events.

In [30], the necessity of adopting event-based approach in CPS is discussed, however, a formal CPS event model including the semantics of an event and the event composition rules is not considered in this work. In [31], we introduce the concept of *observers* and a hierarchical spatio-temporal event model for CPS. The event model uses event attributes, occurrence time and space stamps, and event observer together to uniquely identify a CPS event instance. In addition, a set of temporal, spatial and logical operators are defined to support the temporal and spatial event composition. However, in [31], events are differentiated by four categories based on the corresponding four different system layers, namely, physical events, sensor events, cyber-physical events, and cyber events. Furthermore, although event temporal, spatial, and logical compositions are defined in [31], structural representation of observers and formal treatment of event type compositions are not considered.

In summary, CPS as an emerging concept introduces new challenges in system design and an event-based approach is necessary for the realization of CPS [30]. Not only the information carried in CPS events are far richer than the existing systems (e.g., the spatio-temporal and the observer information [31]), but the diversity of CPS events also range from lower-level, physical events to higher-level, human/machine-understandable cyber events. To the best of our knowledge, the work presented in the paper is the first event model that captures the essential information about events in a distributed environment.

3. CPS EVENT STRUCTURE

In this section, we formally define the CPS event model. More specifically, the syntax for the CPS event instance is described and the CPS event extraction functions that extract the internal and external event attributes are introduced. These building blocks for the CPS event model can be used by heterogeneous components for event composition in the CPS.

3.1 Representation of a CPS Event Instance

As described in Section 2, representation of a CPS event instance is significantly different from traditional event representation. More specifically, the spatio-temporal properties of the CPS event as well as the components that *observe* this event should be an integral component of the event definition. Accordingly, we define a CPS event instance as follows:

$\mathcal{E}_{cps}$	$:= \mathbf{\Gamma}\langle\mu, \mathcal{T}^{\mathbf{g}}, \mathcal{L}^{\mathbf{g}}\rangle @(\mathcal{T}, \mathcal{L}, \mathcal{O})$
$\mathcal{T}^g, \mathcal{T}$	$:= \underline{[}t_1, t_2\underline{]}$
$\mathcal{L}^g, \mathcal{L}$	$:= ((x, y, z), r)$
$\mathcal{O}$	$:= \mathcal{E}_{cps} \vert \top$
$\underline{[}$	$:= (\vert [$
$\underline{]}$	$:=) \vert]$
t_1, t_2, r	$:= \in \Re^+$
x, y, z	$:= \in \Re$

Table 1: CPS Event Syntax

DEFINITION 1 (CPS EVENT INSTANCE). *A CPS event instance is represented by the event type,* internal *event attributes, and* external *event attributes as shown in* (1),

$$\mathcal{E}_{cps} = \Gamma\langle\underbrace{\mu, \mathcal{T}^g, \mathcal{L}^g}_{Internal}\rangle @(\underbrace{\mathcal{T}, \mathcal{L}, \mathcal{O}}_{External}) \tag{1}$$

where

- Γ *represents the type of the event instance.*
- Internal attributes: μ *represents a finite set of attributes of the event instance, while* $\mathcal{T}^g$ *and* $\mathcal{L}^g$ *represent the time and the location at which the event is generated.*
- External attributes: $\mathcal{T}$ *and* $\mathcal{L}$ *represent the time and the location at which the event is observed to occur, which may differ from the time and location from which the event is generated. Finally,* $\mathcal{O}$ *is the observer of the event instance.*

The internal attributes of an event are highly application- dependent. For example, (un)certainty associated with the timestamp of an event can be included as an internal attribute [34]. On the other hand, the external attributes of an event are application-independent and represent fixed properties that all CPS events have. We argue that external attributes are a major difference from traditional event models.

The temporal attributes $\mathcal{T}^g$ and $\mathcal{T}$ in (1) are given in the form of a time interval, i.e., $[a, b]$ (or $(a, b]$, (a, b), $[a, b)$). When $a = b$, the event is an instant event. It is important to note that all timestamps represent "real-clock timestamps" instead of "logical-clock timestamps," such as Lamport's vector clock [16]. This is due to explicit timing constraints, e.g., "A occurs 5 seconds before B", which are very common in applications.

The spatial attributes $\mathcal{L}^g$ and $\mathcal{L}$ in (1) are given in the form of $((x, y, z), r)$, where (x, y, z) is the relative geographical coordinates with respect to the observer $\mathcal{O}$, and r indicates the radius of the event. A point event is will have $r = 0$, and a field event will have $r > 0$. [2]

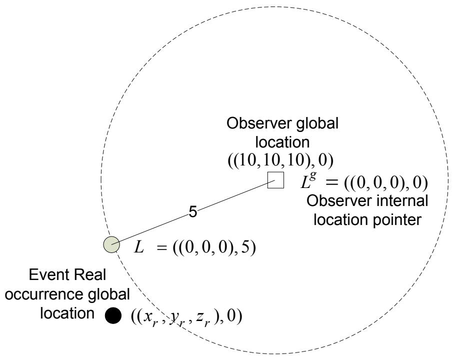
In summary, (1) describes an event instance $\mathcal{E}_{cps}$ of event type Γ with attributes μ observed by $\mathcal{O}$. The event instance is observed to occur at time $\mathcal{T}$ and location $\mathcal{L}$ with respective to the observer location. Then, the event instance is generated at time $\mathcal{T}^g$ and location $\mathcal{L}^g$ with respect to the observer.

In addition to cyber-physical events, the observer $\mathcal{O}$ is also defined as an event instance with an event type Obs. Accordingly, the *observer event instance* is defined as follows:

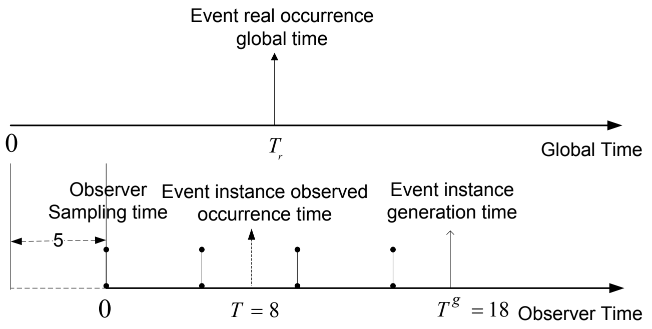
DEFINITION 2 (CPS OBSERVER EVENT INSTANCE).

$$\mathcal{E}_{obs} = Obs\,\langle g, id, \langle\Gamma s\rangle, \mu', \mathcal{T}^g, \mathcal{L}^g\rangle\,@\,(\mathcal{T},\, \mathcal{L},\, \mathcal{O}_\top) \tag{2}$$

[2]Although we use a sphere to represent a 3-dimensional region, it is straightforward to extend the model to use other forms.



(a) Event instance generation location $\mathcal{L}^g$, observed occurrence location $\mathcal{L}$



(b) Event instance generation time $\mathcal{T}^g$, observed occurrence time $\mathcal{T}$

Figure 1: The acoustic sensor in Example 1

where g *is the set of event generation rules associated with the observer,* id *is the observer ID,* $\langle\Gamma s\rangle$ *is the set of event types that this observer can generate, and* μ' *are the observer event attributes related to the specific observer. The spatio-temporal attributes of the observer event instance are the same as in (1), and* $\mathcal{O}_\top$ *represents the* global observer.

The global observer is defined for system analysis purposes, so that a common frame of reference can be provided. For any specific CPS system, there is only one global observer $\mathcal{O}_\top$. Its location is the system origin and the time interval is defined as the system's life span. Accordingly, the global observer is defined as $\mathcal{O}_\top = Obs\langle[0, \infty), ((0, 0, 0), 0)\rangle @([0, \infty), ((0, 0, 0), \infty),\ \top)$, where $\top$ denotes the CPS system itself.

Observers dynamically joining or leaving a CPS are represented as events, which can be reported by any observer, including the entity joining or leaving. Mobile observers are handled similarly, though the application needs to determine the granularity of spatial accuracy required, which will determine the frequency with which location updates must be reported for mobile entities.

Based on the definitions of event instance in (1) and its special case observer event instance in (2) we define $\mathbb{E}$ to be the set of all event instances in a CPS system and $\mathbb{O}$ to be the set of all CPS observers, including the global observer $\mathcal{O}_\top$. Obviously, $\mathbb{O} \subseteq \mathbb{E}$. The syntax for the CPS event instance $\mathcal{E}_{cps}$ is given in Table 1.

Example 1.

To better illustrate the event structure and its components, consider an acoustic sensor that observes a CPS event instance as shown in Fig. 1. More specifically, in Fig. 1(a), an acoustic sensor installed at global point location $((10, 10, 10), 0)$ (the square dot) is shown, where a sound event, e.g., *clapping* occurs at a global point location $((x_r, y_r, z_r), 0)$ (the black round dot). When the acoustic sensor is initialized, its *relative* location is set as $\mathcal{L}^g = ((0,0,0),0)$, and it observes that a $Sound$ event occurs within 5 units of its sensing range (the grey round dot).

The timeline of the sensor initialization and event generation is shown in Fig. 1(b), where the acoustic sensor is initialized at global point time 5s. The acoustic sensor initializes its relative time as $[0,\infty)$ and it starts sampling. Assume that the sensor is programmed to generate a sound event after each 4 samples (the round dot end lines). As shown in Fig. 1(b), the sound event occurs at global time T_r. This event is observed as $Sound$ event instance at relative sensor time $\mathcal{T} = 8$s (global time $5+8=13$s) and the event instance is generated at sensor time $\mathcal{T}^g = 18$s (global time $5+18=23$s). The observed event occurrence time, $\mathcal{T}$, and the event instance generation time, $\mathcal{T}^g$, is $18-8=10$s apart because of the event generation mechanism defined at the acoustic sensor, i.e., it generates a sound event instance after every 4 samples. If the sampled sound values are greater than a certain threshold, a sound event instance is then generated. Accordingly, the event instance is generated at $\mathcal{T}^g = 18$s and the the event instance occurrence time is recorded as $\mathcal{T} = 8$s.

Next, we formally represent the acoustic sensor event and the sound event instance. The acoustic sensor event is represented as an observer event instance as follows:

$$\begin{aligned} S_1 \quad = \quad & Obs\,\langle g_{s_1}, s_1, \langle Sound\rangle, [0,\infty), ((0,0,0),0)\rangle\,@\,(\\ & [5,\infty),\ ((10,10,10),0),\ \mathcal{O}_\top) \end{aligned} \tag{3}$$

where Obs event type indicates that this event instance is an observer event instance; $g_{s_1}, s_1, \langle Sound\rangle$ are the observer event attributes describing the event generation rules of the acoustic sensor (g_{s_1}), the sensor ID (s_1), and the event type the acoustic sensor can generate ($Sound$), respectively; $[0,\infty)$ and $((0,0,0),0)$ represent that initial timer and location for the acoustic sensor, respectively; $([5,\infty), ((10,10,10),0,\mathcal{O}_\top)$ specifies that the sensor starts functioning at global time 5 and installed at global point location $((10,10,10),0)$ with respect to a global observer $\mathcal{O}_\top$.

Similarly, the sound event instance that is generated by the acoustic sensor is represented as a CPS event instance as follows:

$$\begin{aligned} e_1 \quad = \quad & Sound\,\langle value_1, [18,18], ((0,0,0),0)\rangle\,@\,(\\ & [8,8],\ ((0,0,0),5),\ S_1) \end{aligned} \tag{4}$$

where $Sound$ is the event type, $value_1$ is the event attribute that characterizes the measured sound strength, $[18,18], ((0,0,0),0)$ describes the event instance is generated at time 18s and location $((0,0,0),0)$ relative to the acoustic sensor since it is generated by the sensor. The sensor also reports that the $Sound$ event is observed to occur at sensor time $[8,8]$ and within $((0,0,0),5)$ units of its location. Finally, S_1 is the observer event instance in (3) and indicates that the event instance e_1 is generated by S_1.

By Definition 1, a CPS event instance can be observed by an observer but the precedence order between the observer and the observed event instance has to be guaranteed. However, to obtain the precedence order among event occurrence times, a common reference is required. Therefore, a group of event extraction functions, including the globalization function, are defined for the purpose next.

3.2 CPS Event Extraction Functions

As defined in Section 3.1, each event instance consists of three types of information, i.e., event type, internal attributes, and external attributes that define when and where an event occurs as well as the observer associated with this event. Given a CPS event $\mathcal{E}_{cps} = \Gamma\langle\mu, \mathcal{T}^g, \mathcal{L}^g\rangle @(\mathcal{T}, \mathcal{L}, \mathcal{O})$, the following extraction functions are defined to extract the corresponding information:

Value function $\mathcal{V}$:$\mathbb{E} \mapsto \mathbb{T} \times \mathbb{I}$ extracts the event type and its event attributes from a CPS event $\mathcal{E}_{cps}$:

$$\mathcal{V}(\mathcal{E}_{cps}) = \Gamma\mu \tag{5}$$

where $\mathbb{T}$ and $\mathbb{I}$ are sets of event types and event attributes, respectively.

Temporal functions T and T^g: $\mathbb{E} \mapsto \Re^+ \times \Re^+$ extract the time during which the event instance occurs and it is generated as:

$$T(\mathcal{E}_{cps}) = \mathcal{T} = \underline{[}t_1, t_2\underline{]} \tag{6}$$

$$T^g(\mathcal{E}_{cps}) = \mathcal{T}^g = \underline{[}t_1^g, t_2^g\underline{]} \tag{7}$$

respectively, where $t_1, t_2, t_1^g, t_2^g \in \Re^+$, $\underline{[} \in \{(, [\}$, and $\underline{]} \in \{),]\}$.

Spatial functions L and L^g: $\mathbb{E} \mapsto \Re \times \Re \times \Re \times \Re^+$ extract the location where the event instance occurs and where it is generated as:

$$L(\mathcal{E}_{cps}) = \mathcal{L} = ((x,y,z),r) \tag{8}$$

$$L^g(\mathcal{E}_{cps}) = \mathcal{L}^g = ((x^g,y^g,z^g),r^g) \tag{9}$$

respectively, where $x, y, z, x^g, y^g, z^g \in \Re$ and $r, r^g \in \Re^+$.

Observer function B: $\mathbb{E} \mapsto \mathbb{O}$ extracts the event observer:

$$B(\mathcal{E}_{cps}) = \mathcal{O} \tag{10}$$

In our model, a CPS event instance is defined based on its observer which itself is also a CPS event instance. There may be many observers in a CPS system, but the global observer $\mathcal{O}_\top$ is unique within a system serving as the system's coordinates and a wall-clock. To compare two CPS event instances in terms of time and location attributes or generate composite events from distinct event instances, the corresponding observers must share the same references. To this end, a *globalization function* is defined to transform a CPS event, which is initially defined with respect to a local observer, to an event with respect to the global observer.

DEFINITION 3 (GLOBALIZATION FUNCTION). *Given an observer*

$$\begin{aligned} s =& Obs\,\langle\mu_s, [t_s^g,\infty), ((x_s^g,y_s^g,z_s^g),0)\rangle\,@\,(\\ & [t_s,\infty),\ ((x_s,y_s,z_s),0),\ \mathcal{O}_\top) \end{aligned} \tag{11}$$

and an event observed by s,

$$\begin{aligned} \varepsilon =& \Gamma\,\langle\mu, [t_1^g,t_2^g], ((x^g,y^g,z^g),r^g)\rangle\,@\,(\\ & [t_1,t_2],\ ((x,y,z),r),\ s) \end{aligned} \tag{12}$$

the globalization function $G : \mathbb{E} \mapsto \mathbb{E}$ *is defined by:*

$$\begin{aligned} G(\varepsilon) =& \Gamma\,\langle\mu, [t_s+t_1^g-t_s^g, t_s+t_2^g-t_s^g], ((x_s+x^g-x_s^g, \\ & y_s+y^g-y_s^g, z_s+z^g-z_s^g), r^g)\rangle\,@\,([t_s+t_1-t_s^g, \\ & t_s+t_2-t_s^g],\ ((x_s+x-x_s^g, y_s+y-y_s^g, \\ & z_s+z-z_s^g), r),\ \mathcal{O}_\top) \end{aligned} \tag{13}$$

The globalization function of the event instance e_1 in (4), of Example 1, changes its observer from S_1 to the global observer $\mathcal{O}_\top$. As a result, the event occurrence time and location and generation time and location are converted to the global observer's perspective. According to (13), the globalized event instance e_1 is defined as follows:

$$\begin{aligned} G(e_1) =& Sound\,\langle value_1, [23,23], ((10,10,10),0)\rangle\,@\,(\\ & [13,13],\ ((10,10,10),5),\ \mathcal{O}_\top) \end{aligned} \tag{14}$$

Recall that the global observer is for system analysis purposes. Moreover, the concept of "global" here is relative: it might be "global" inside one sub-system but becomes "local" for a bigger system, and vice versa. Therefore, in the system implementation stage, as long as the event instances in the "globalization function" share one common reference and are within the allowable system error range, the event instances can be compared and later composed with respect to their occurrence times and locations.

4. CPS EVENT COMPOSITION

Events in a CPS range from low-level physical events such as sensory data to higher-level, human/machine-understandable cyber events. Using only the lower-level physical events in the system is not only inefficient in terms of system bandwidth, but it is also not scalable in distributed systems such as CPS [30]. Therefore, composite events are required to provide an extra means of interaction and keep communication efficient. In this section, the formal approach of event composition using the CPS event model is described.

Given a CPS and an application, the available types of sensors and the associated events that can be generated by these sensors can be determined. For example, a temperature sensor produces an event type $Temperature$ and a humidity sensor produces an event type $Humidity$. These event types produce by the sensors are considered *primitive events* and are used to compose other higher-level event types in CPS. Formally, the following notations are defined:

- $\mathbb{T}$ is the set of all event types in a CPS. For any specific CPS, the $\mathbb{T}$ is a finite set, i.e., $\mathbb{T} = \{\Gamma_1, \Gamma_2, ..., \Gamma_n\}$.
- $\mathbb{B}$ is the set of primitive event types that the available sensors in this CPS can produce, i.e., $\mathbb{B} = \{\Gamma_1', \Gamma_2', ..., \Gamma_i'\}$ where $\mathbb{B} \subseteq \mathbb{T}$. The set $\mathbb{B}$ is also referred to as the *primitive event type set*, which is the foundation to compose other higher-level event types $\mathbb{T} \setminus \mathbb{B}$ in the CPS.
- $\mathbb{I}$ is the set of all event attributes that correspond to the event type set $\mathbb{T}$ in this CPS, $\mathbb{I} = \mu_1 \cup \mu_2 \cup ... \cup \mu_n$.
- $\mathbb{I}'$ is the set of event attributes μ_i' that correspond to the primitive event types Γ_i' in $\mathbb{B}$ in a CPS, i.e., $\mathbb{I}' = \mu_1' \cup \mu_2' \cup ... \cup \mu_i'$ where $\mathbb{I}' \subseteq \mathbb{I}$. The set $\mathbb{I}'$ is also referred as the *primitive event attribute set*.

Formally, event composition in CPS can be defined as follows: Consider a set of CPS events instances $e_1, e_2, \cdots, e_n$ and their composition as a new CPS event instance e_a, which can be considered as an abstraction from primitive events. More specifically, given $e_i = T_i \langle \mu_i, \mathcal{T}_i^g, \mathcal{L}_i^g \rangle \,@\, (\mathcal{T}_i,\, \mathcal{L}_i,\, \mathcal{O}_i)$, we define an abstraction function A as follows:

$$\Gamma \langle \mu, \mathcal{T}^g, \mathcal{L}^g \rangle \,@\, (\mathcal{T},\, \mathcal{L},\, \mathcal{O}) = A(e_1, e_2, \cdots, e_n) \tag{15}$$

where $A = \{A_\Gamma, A_\mu, A_\mathcal{T}, A_{\mathcal{T}^g}, A_\mathcal{L}, A_{\mathcal{L}^g}, A_\mathcal{O}\}$ and

$$\Gamma = A_\Gamma((\mu_1, \mathcal{T}_1^g, \mathcal{T}_1, \mathcal{L}_1^g, \mathcal{L}_1, \mathcal{O}_1), (\mu_2, \mathcal{T}_2^g, \mathcal{T}_2, \mathcal{L}_2^g, \mathcal{L}_2, \mathcal{O}_2), \cdots, (\mu_n, \mathcal{T}_n^g, \mathcal{T}_n, \mathcal{L}_n^g, \mathcal{L}_n, \mathcal{O}_n)) \tag{16}$$

$$\mu = A_\mu(\mu_1, \mu_2, \cdots, \mu_n) \tag{17}$$

$$\mathcal{T} = A_\mathcal{T}(\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n) \tag{18}$$

$$\mathcal{T}^g = A_{\mathcal{T}^g}(\mathcal{T}_1^g, \mathcal{T}_2^g, \cdots, \mathcal{T}_n^g) \tag{19}$$

$$\mathcal{L} = A_\mathcal{L}(\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_n) \tag{20}$$

$$\mathcal{L}^g = A_{\mathcal{L}^g}(\mathcal{L}_1^g, \mathcal{L}_2^g, \cdots, \mathcal{L}_n^g) \tag{21}$$

$$\mathcal{O} = A_\mathcal{O}(\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_n) \tag{22}$$

In other words, the composite event, e_a, is the union of the event type composition, event attributes composition, spatio-temporal attributes composition, and the observer composition. Next, we first define the CPS concept lattice that is used to formally define the composition functions (16-22) and then, describe each specific composition function.

EC :=	EC ∧ EC | EC ∨ EğC | ¬ EC | (EC) | EC | atom
atom :=	V-exp Op V-exp | T-exp Op T-exp |
	L-exp Op L-exp | O-exp Op$_\mathcal{O}$ O-exp
V-exp :=	algebraic-exp-of(attributes)
T-exp :=	algebraic-exp-of(begin-time, end-time)
L-exp :=	algebraic-exp-of(x-value, y-value, z-value, r-value)
Op :=	==| > | ≤

Table 2: Syntax for CPS Event Constraint Expression

4.1 CPS Concept Lattice

To structurally define the event type composition in the CPS, we adopt the theory of concept lattice [32] in the composition of CPS event types. Concept lattice has been widely used in machine learning, knowledge discovery, and software engineering, however, to the best of our knowledge, has not been applied to event composition and abstraction for CPS applications. The theory of concept lattice is established upon a *formal context*, which is defined as follows:

DEFINITION 4 (FORMAL CONTEXT). *a formal context is a triple* $(\mathbb{I}', \mathbb{T}, M)$, *where* $\mathbb{I}'$ *is the primitive event attribute set,* $\mathbb{T}$ *is the event type set, and* $M \subseteq \mathbb{I}' \times \mathbb{T}$ *defines the bipartite relationships between primitive event attribute set* $\mathbb{I}'$ *and the event type set* $\mathbb{T}$.

A formal context defines the relationship between primitive event attributes in $\mathbb{I}'$ and event types in $\mathbb{T}$. In other words, a formal context defines how the primitive event attributes can be constrained and form event types in $\mathbb{T}$. For example, if the height of an object is classified as short, medium, and tall, and the width as narrow, medium, and wide, the formal context can be defined as $(\langle H(eight), W(idth)\rangle, \{h_s, h_m, h_t, w_n, w_m, w_w\}, M)$, where M is defined as the set

$$\left\{ \begin{array}{c} (\langle [0'0'', 4'0''), W\rangle, h_s), (\langle [4'0'', 8'0''), W\rangle, h_m), \\ (\langle [8'0'', +\infty), W\rangle, h_t), (\langle H, [0'0'', 2'0'')\rangle, w_n), \\ (\langle H, [2'0'', 4'0'')\rangle, w_m), (\langle H, [4'0'', +\infty)\rangle, w_w) \end{array} \right\}$$

However, the formal context supports only the constraints over a single domain (e.g., on the $Height$ attribute) and binary operations to combine constraints from different domains (e.g., $h_s \cup m_w$). Instead, it can not form relationships across multiple domains (e.g., $\mathcal{V}(h_m) == \mathcal{V}(w_w)$). In addition, the spatio-temporal attributes and the observer information can also be used to compose new event types. To accommodate greater flexibility, we extend the formal context to compose event types using constraints across multiple domains.

In CPS, some event type compositions may only be permissible under certain constraints on the event attributes, the spatio-temporal information, and/or the observer information. Such a composition is referred to as *guarded composition*. For a given set of CPS events, $e_i, (i = 1, 2, ..., n)$, and an event constraint expression with respect to the given event set, the guarded composition has the following structure:

$$[EC]A(e_1, e_2, ..., e_n) \tag{23}$$

The syntax for CPS event constraint (EC) expression is given in Table 2.

With guarded composition, event types can be defined across event attributes, spatio-temporal attributes over two or more event instances. Let $Z = \{\mathcal{T}, \mathcal{L}, \mathcal{T}^g, \mathcal{L}^g, \mathcal{O}\}$, then we define an extend event attribute set $\mathbb{I}_{EC}$:

$$\mathbb{I}_{EC} = \langle 2^{\mathbb{I}} \times 2^{Z} \rangle \setminus \emptyset \tag{24}$$

where $\mathbb{I}_{EC}$ is the product of the power set of the event attribute set and the power set of the event instance spatio-temporal and observer information. Accordingly, we define the extended formal context, which not only includes spatio-temporal and observer information, but also has the ability to compose event properties across different domains, as follows:

DEFINITION 5 (EXTENDED FORMAL CONTEXT). *The extended formal context for event composition guard is a triple* $(\mathbb{I}_{EC}, \mathbb{T}, M_{EC})$*, where* $\mathbb{I}_{EC}$ *is defined in* (24), $\mathbb{T}$ *is the event type set, and* $M_{EC} \subseteq \mathbb{I}_{EC} \times \mathbb{T}$ *defines the bipartite relationship between extended event attribute set* $\mathbb{I}_{EC}$ *and the event type set* $\mathbb{T}$.

Finally, the *CPS formal context* is defined as the union of the formal context and the extended formal context.

DEFINITION 6 (CPS FORMAL CONTEXT). *A CPS formal context is a triple* $(\mathbb{C}, \mathbb{T}, \mathbb{M})$*, where* $\mathbb{C} = \mathbb{I}' \cup \mathbb{I}_{EC}$, $\mathbb{T}$ *is the event type set, and* $\mathbb{M} = M \cup M_{EC}$.

The CPS formal context allows complex relationships between event attributes and event types to be defined using either the CPS formal context itself or the event composition guards. These relationships are called *CPS concepts* as defined next:

DEFINITION 7 (CPS CONCEPT). *Let* $(\mathbb{C}, \mathbb{T}, \mathbb{M})$ *be a CPS formal context, then* (X, Y) *is called a* CPS concept *if*

$$X = \{\mu \in \mathbb{C} | \forall \Gamma \in \mathbb{T}, (\mu, \Gamma) \in M\} \quad (25)$$
$$Y = \{\Gamma \in \mathbb{T} | \forall \mu \in \mathbb{C}, (\mu, \Gamma) \in M\} \quad (26)$$
$$\textit{Or}\ \exists g_c, \quad s.t. (X, g_c, Y) \in M_{EC} \quad (27)$$

where $X \in 2^{\mathbb{C}}$, $Y \in 2^{\mathbb{T}}$ *and* g_c *is an event composition guard.*

For example, the detection of a $Person$ can be associated with objects of medium heights and widths, i.e., $\langle [4'0'', 8'0''), W\rangle \cap \langle H, [2'0'', 4'0'')\rangle = \langle [4'0'', 8'0''), [2'0'', 4'0'')\rangle$ of event type $\{h_m, w_m\}$. We associate event constraints $\langle [4'0'', 8'0''), [2'0'', 4'0'')\rangle$ with type $Person$ (an alias for $\{h_m, w_m\}$) as a CPS concept. On the other hand, for an event composition guard $g_c : \mathcal{V}(H) == \mathcal{V}(W)$ defined from event type $\{h_m, w_w\}$ to an event type $Square$, the system has a valid relationship between the event type $Square$ and the event types $\{h_m, w_w\}$ with the help of event composition guard g_c. Clearly, in the particular example, although the $\mathbb{I}_{EC}$ is a very large set, the M_{EC} is fairly simple since have one event composition guard defined:

$$M_{EC} = \{(\langle h_m, w_w\rangle, g_c : \mathcal{V}(H) == \mathcal{V}(W), Square)\}$$

A CPS formal context and its CPS concepts establish the relationships between event attributes and their corresponding event types. Such relationships can be built into a hierarchy to lay the semantic base for event abstractions. Such a hierarchy is defined as a *CPS concept lattice* as follows:

DEFINITION 8 (CPS CONCEPT LATTICE). *For a formal context* $(\mathbb{C}, \mathbb{T}, \mathbb{M})$*, let* (X_1, Y_1) *and* (X_2, Y_2) *be two CPS concepts. If* $X_1 \supseteq X_2$ *and* $Y_1 \subseteq Y_2$ *, or there is at least one event composition guard placed on concept* (X_1, Y_1) *to* (X_2, Y_2)*, then there is a partial order* $\prec$ *between* (X_1, Y_1) *and* (X_2, Y_2)*, i.e.,*

$$(X_1, Y_1) \prec (X_2, Y_2) \quad (28)$$

Such a partial order relation is of a lattice structure and forms the concept lattice *of the formal context* $(\mathbb{C}, \mathbb{T}, \mathbb{M})$.

The CPS concept lattice will always form a partial order between any two CPS concepts unless the corresponding concepts are the same.

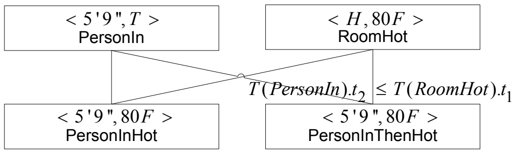


Figure 2: Concept lattice of $PersonInHot$ **and** $PersonInThenHot$

Example 2.
To further illustrate the usage of a CPS concept lattice, and composition guard with a temporal constraint, let us consider an example, where two events are considered. A person with height $5'9''$ stays in a room $\Gamma_1\, \mu_1 = PersonIn\, \langle 5'9'', -\rangle$ and the room is hot $\Gamma_2\, \mu_2 = RoomHot\, \langle -, 80F\rangle$ where the internal attributes of event instances are of the form $\langle$*H(eight)*, *T(emperature)*$\rangle$, "$-$" denotes the don't-care attribute. The composition of the two events is defined as

$$\Gamma\, \mu = PersonInHot\, \langle 5'9'', 80F\rangle,$$

where $PersonInHot$ is a shorthand notation for the set $\{PersonIn, RoomHot\}$. For internal attributes, $\langle 5'9'', -\rangle \supseteq \langle 5'9'', 80F\rangle$ and $\langle -, 80F\rangle \supseteq \langle 5'9'', 80F\rangle$; and for event types, $\{PersonIn\}, \{RoomHot\} \subseteq \{PersonIn, RoomHot\}$. Therefore, according to Definition 8, $\Gamma_1\ \mu_1 \prec \Gamma\ \mu$ and $\Gamma_2\ \mu_2 \prec \Gamma\ \mu$ in the concept lattice. On the other hand, to define the event of type $PersonInThenHot$, we require that the event of type $PersonIn$ to occur earlier than the event of type $RoomHot$, i.e., $T(PersonIn).t_2 \leq T(RoomHot).t_1$, where T is the time extraction function defined in Section 3.2. The resulting CPS concept lattice is shown in Figure 2.

In summary, different CPS applications have different concept lattices which define the composition abstraction relationship among different types of events that are of interest to the applications. Event compositions based on a given concept lattice allow events to be transferred cross different components and used by heterogeneous devices and components distributed in a CPS.

4.2 Event Type and Event Attribute Compositions

Given a concept lattice, a set of event instances can be composed if and only if their corresponding event types are composable in the concept lattice. Furthermore, the composed type (or the composed concept) must be the largest lower bound of the composing concept. Definition 9 gives the formal definition.

DEFINITION 9 (EVENT TYPE COMPOSITION (A_Γ)). *Given a set of CPS events* e_1, e_2, $\cdots$, e_n*, a concept lattice* $\mathbb{C}$*, and their event attributes, spatio-temporal attributes, and observers as* $(\mu_1, \mathcal{T}_1^g, \mathcal{T}_1, \mathcal{L}_1^g, \mathcal{L}_1, \mathcal{O}_1)$, $(\mu_2, \mathcal{T}_2^g, \mathcal{T}_2, \mathcal{L}_2^g, \mathcal{L}_2, \mathcal{O}_2, \cdots,$ $(\mu_n, \mathcal{T}_n^g, \mathcal{T}_n, \mathcal{L}_n^g, \mathcal{L}_n, \mathcal{O}_n)$*, respectively, then,*

$$\Gamma\ = A_\Gamma((\mu_1, \mathcal{T}_1^g, \mathcal{T}_1, \mathcal{L}_1^g, \mathcal{L}_1, \mathcal{O}_1), (\mu_2, \mathcal{T}_2^g, \mathcal{T}_2, \mathcal{L}_2^g, \mathcal{L}_2, \mathcal{O}_2), \cdots, (\mu_n, \mathcal{T}_n^g, \mathcal{T}_n, \mathcal{L}_n^g, \mathcal{L}_n, \mathcal{O}_n))$$

If $(\mu_1, \mathcal{T}_1^g, \mathcal{T}_1, \mathcal{L}_1^g, \mathcal{L}_1, \mathcal{O}_1)$, $(\mu_2, \mathcal{T}_2^g, \mathcal{T}_2, \mathcal{L}_2^g, \mathcal{L}_2, \mathcal{O}_2)$, $\cdots$, $(\mu_n, \mathcal{T}_n^g, \mathcal{T}_n, \mathcal{L}_n^g, \mathcal{L}_n, \mathcal{O}_n)$ *are all immediate predecessor of* Γ *in the given CPS concept lattice* $\mathbb{C}$.

Once the event type composition A_Γ succeeds, then the corresponding event attribute composition $A_\mathcal{V}$ is defined as follows:

$$\mu\ = A_\mathcal{V}(\mu_1, \mu_2, \cdots, \mu_n)$$

where $A_\mathcal{V}$ can be any valid algebraic functions or set functions.

4.3 Temporal and Spatial Compositions

As explained in Section 3, the temporal and spatial external attributes of an event are of the form $\underline{[}t_1, t_2\underline{]}$ and $((x, y, z), r)$, which are essentially 1- and 3-dimensional convex regions, respectively. The compositions of time and locations of events can thus be defined as algebraic operations on these convex regions, i.e., unions, intersections, complements, symmetric differences, averages, etc. The choices of operations are application-dependent. One of the most important issues of time and location compositions, however, is to guarantee their closeness: the time and location attributes of composed events should also be of the form $\underline{[}t_1, t_2\underline{]}$ and $((x, y, z), r)$, respectively. In the following, we illustrate time and location compositions using the *union* operation. Using other operations follows similar principles.

For temporal compositions of the form $A_{\mathcal{T}}(\underline{[}t_{11}, t_{12}\underline{]}, \ldots, \underline{[}t_{n1}, t_{n2}\underline{]})$ as given in (18)[3], the time attribute of the composed event is defined as

$$\mathcal{T} = \underline{[}\min\{t_{11}, \ldots, t_{n1}\}, \max\{t_{12}, \ldots, t_{n2}\}\underline{]}$$

where $\underline{[}$ and $\underline{]}$ comply with the corresponding boundaries chosen in $\min$ and $\max$, respectively. In the 1-dimensional case, $\mathcal{T}$ so defined is the smallest convex region that includes the time attributes of the composing events. This coincides with the intuition that the interval time stamp of the composed event should span those of the earliest and the latest composing events.

For the spatial compositions of the form $A_{\mathcal{L}}(((x_1, y_1, z_1), r_1), \ldots, ((x_n, y_n, z_n), r_n))$ as given in (20)[4], the location attribute of the composed event is defined as the smallest spherical region that contains the locations of the composing events. As defined in Section 3, the global observer's location is $((0, 0, 0), +\infty)$, i.e., a region centered at the origin and with an infinite radius. As a consequence, the location of any composed event will not reach beyond the scope of the global observer, thus guaranteeing the closeness of the composition.

It is worth pointing out that although the temporal and spatial composition functions, T and L, respectively, may have application variations, all choices must ensure that the resulting time and location must be continuous and without "holes". In this paper, for the ease of numerical discussions presented in Section 5, we consider *spherical* locations and *infinite* scope of the global observer.

4.4 Observer Composition

As mentioned in Section 3, observers are treated as events with respect to a global observer $\mathcal{O}_\top$. Thus, the composition rules described for other event types apply to observer events as well. According to Section 4.2, the type of composed observers/events are sets of basic types. At the same time, as discussed in Section 4.3, the spatio-temporal external attributes of observers also expand as more observers/events are composed. Eventually, an observer with a type as the whole event type set and time and location external attributes as the life span and range of the entire system, respectively, will become the bottom of the concept lattice for compositions. This observer is the *global observer*, $\mathcal{O}_\top$, as defined in Section 3. The CPS event composition theory is hence, complete with the addition of the global observer at the bottom of the concept lattice.

5. CASE STUDY: SMART SPACE

In this section, the *Smart Space* [27], an ongoing CPS smart home prototype project at the University of Nebraska-Lincoln, is introduced as a case study to illustrate the usage of the proposed event model. More specifically, the following simple scenario (denoted as $Target$) in the *Smart Space* is considered: *If a person stays in the living room for more than 2 seconds and the living room is dim, turn on the lights in 5 seconds.* The process of how the CPS concept lattice is formed and events are composed from the lower to higher levels for this example are shown in a step-by-step manner. In addition, a CPS Event Simulator that adopts the proposed event model is implemented. The implementation of the CPS event model is then compared with the two other traditional approaches to event detection (one of the approach is used in the original *Smart Space* Java implementation) in terms of QoS support, localization error and time duration error.

The goal of the *Smart Space* is to help people with disabilities to live a better life through a wireless sensor-actor network equipped smart home environment. In this example, only two types of sensors in the Smart Space are used: (1) the Cricket Localization Sensors (CLS) [26] that measure the range for the target person and the fixed CLS nodes installed on the ceiling of *Smart Space*. (2) a lighting sensor that measures the strength of light in the living room. The CLS system keeps track of the real-time location of the target person, and together with real-time lighting strength in the living room, the *Smart Space* decides whether the target scenario is detected. Accordingly, the lights are turned on in the living room.

5.1 Development of CPS Concept Lattice

Step 1: First, the primitive event type set $\mathbb{B}$, primitive event attribute set $\mathbb{I}'$ and the event type set $\mathbb{T}$ for the given CPS application are determined. Accordingly, the primitive event type and attribute sets are defined as $\mathbb{B} = \{Range, LtStr\}$ and $\mathbb{I}' = \langle Range, LtStr, SeqNum\rangle$, respectively, where the event type $Range$ is for the Cricket Localization Sensor (CLS), $LtStr$ is for the light sensor, $SeqNum$ is the sequence number of event instance. To generate Loc event type from the $Range$ event type, trilateration of at least 3 independent *synchronized* range measurements are required [5]. Therefore, the event instance sequence number $SeqNum$ is used transform $Range$ event instances generated by different CLS nodes to a Loc event instance.

The event type set, $\mathbb{T}$, is application-dependent , where in our case, the target scenario is formed by 3 sub-events: "*a person stays in the living room for more than 2 seconds*", "*the living room is dim*" and "*turn on the light*". Accordingly, 4 event types are created: $Target$, $InRoom2s$, $RoomDim$ and $LightOn$. For simplicity, assume time difference between generation time of the event $LightOn$ and the time at which "turn on the light" event is generated is negligible. Therefore, event type $Target$ is generated when the 5 seconds constraint between the generation time of $LightOn$ and the occurrence of $InRoom2s$ and $RoomDim$ events is met. In addition, $InRoom2s$ is composed of $InRoom$ and Loc event types, where $InRoom$ describes whether the person is the living room and Loc describes the location of the person. The Loc and $RoomDim$ event types can be generated from the basic event types $Range$ and $LtStr$, respectively. The $LightOn$ event can be generated from the event types $InRoom2s$ and $RoomDim$. Finally, Obs event type exists in every CPS system to register and update the status for the observers in the system. Therefore, a total 9 event types are defined for this example:

$$\mathbb{T} = \{Range, LtStr, Loc, InRoom, InRoom2s, \\ RoomDim, LightOn, Target, Obs\}$$

Step 2: Next, we develop a formal context $(\mathbb{I}', \mathbb{T}, M)$ for the

[3]the other time composition $A_{\mathcal{T}^g}$ follows the same discussion below

[4]similarly, the other spatial composition $A_{\mathcal{L}^g}$ follows the same discussion below

[5]due to page limits, for the technical details of CLS, please refer to [26]

given CPS application. The relation M is given as:

$$M = \{(\langle Range, [0, 300)\rangle, RoomDim)\}$$

where a new event type $RoomDim$ is generated when the corresponding attribute of basic event type $LtStr$ is in the range $[0, 300)$.

Step 3: Calculate the extended event attribute set $\mathbb{I}_{EC}$. We have $Z = \langle \mathcal{T}^g, \mathcal{L}^g, \mathcal{T}, \mathcal{L}, \mathcal{O}\rangle$ and

$$\mathbb{I}_{EC} = \langle Range, LtStr, SeqNum, \mathcal{T}^g, \mathcal{L}^g, \mathcal{T}, \mathcal{L}, \mathcal{O}\rangle$$

Step 4: Develop an extended formal context $(\mathbb{I}_{EC}, \mathbb{T}, M_{EC})$ and the corresponding event composition guards. Using the $\mathbb{I}_{EC}$ set, we can define the M_{EC} and the corresponding event composition guards as follows:

$$M_{EC} = \left\{ \begin{array}{l} (\langle Range\rangle, cg_1, Loc), \quad (\langle Loc\rangle, cg_2, InRoom), \\ (\langle InRoom\rangle, cg_3, InRoom2s), \\ (\langle InRoom2s, RoomDim\rangle, cg_4, LightOn), \\ (\langle InRoom2s, RoomDim, LightOn\rangle, cg_5, Target) \end{array} \right\}$$

where

$$\begin{aligned} cg_1 &= Independent(\mathcal{V}(Range).SeqNum) \geq 3 \\ cg_2 &= Within(L(Loc), LR) == true \\ cg_3 &= Continue(T(InRoom)) \geq 2s \\ cg_4 &= T(InRoom2s).t_1 > 0 \wedge \mathcal{V}(RoomDim) == true \\ cg_5 &= T^g(LightOn).t_1 - Max(T(InRoom2s).t_1, \\ &\quad\quad T(RoomDim).t_1) < 5s \end{aligned}$$

where the $Independent$ function returns the number of range measurements with the same event instance sequence number, $Within$ returns true if the input location is within the range of the living room, LR.

Step 5: Finally, we compute the CPS formal context $(\mathbb{C}, \mathbb{T}, \mathbb{M})$ by combining the formal context $(\mathbb{I}', \mathbb{T}, M)$ and extended formal context $(\mathbb{I}_{EC}, \mathbb{T}, M_{EC})$, where

$$\mathbb{C} = \langle Range, LtStr, SeqNum, \mathcal{T}^g, \mathcal{L}^g, \mathcal{T}, \mathcal{L}, \mathcal{O}\rangle$$

and $\mathbb{M} = M \cup M_{EC}$.

The CPS concept lattice for this example is shown in Figure 3(a), where in the notation $\langle -, -, -, \mathcal{T}^g, \mathcal{L}^g, \mathcal{T}, \mathcal{L}\rangle$ the "$-$" refers to the don't care attributes that would not interfere with the event composition. The geometric shapes on the left side of the event types are the sensors / observers which generate these events, please refer to Section 5.2 and Figure 3(b) for more explanations.

5.2 CPS Event Simulator

To validate the proposed CPS event model, a Tossim [19] based CPS event simulator (CPSim) is implemented and the Smart Home example has been simulated on this simulator. The CPSim consists of two parts as shown in Fig. 3(c): physical event simulation (PES) and cyber event simulations (CES). The PES is written in Python and simulates the physical phenomenon of interest. The CES consists of mote code written in nesC [10] and simulates the behaviors of sensors and observers in the CPS. Inside CES, the simulated sensors and observers communicate through the Tossim simulated wireless network. The PES and CES communicate with each other through the Tossim packet injection interface and the get variable interface. Based on the mote variables retrieved from Tossim, the PES will adjust the packet injection content to CES to simulate the actuation process. As a result, the PES and CES form a complete control-loop in a CPS. A global time reference and 3-D coordinate system is used to control the simulation at a macro level, but each individual component generates CPS event instances using their local time and coordinate system.

The example in Section 5.1 has been simulated. The simulation runs from time 0 to 30 seconds. The network topology is a $4m \times 4m$ square grid with a total of 9 nodes installed as shown in Fig. 3(b). More specifically, 3 CLS sensors, 2 light sensors, and 4 observers with observer ID Ob_{Loc}, Ob_{InRoom}, $Ob_{InRoom2s}$ and Ob_{Target} are installed. The sampling interval for the three CLS sensors is $0.3s$, and the sampling interval for the 2 light sensors is $1s$. A 2% random variance has been added to all $Range$ event attributes to simulate the randomness of the ultrasonic range measurements. As both Figures 3(a) and 3(b) show: the CLS sensors generate $Range$ events, light sensors generate $LtStr$ and $RoomDim$ events, Ob_{loc} generates Loc events, Ob_{InRoom} generates $InRoom$ events, $Ob_{InRoom2s}$ generates $InRoom2s$ events, and finally Ob_{Target} generates $LightOn$ and $Target$ events. The observer event definitions are omitted for space considerations but follow the convention in (2). The event composition guards are programmed according to Section 5.1.

The target person walks along a fixed path (the dot-dashed bold line in Fig. 3(b)) with a constant speed S for each individual simulation. Thus, the target person is physically in the following rooms based on his walking speed S: during period $[0, 2/S)$, he / she is in the foyer; during period $[2/S, 5.5/S)$, in the living room; during period $[5.5/S, 9/S)$, in the bedroom; during period $[9/S, 11/S)$, in the kitchen; and finally, during period $[11/S, 30]$, in the foyer again. After 30 seconds, regardless of where he/she is, the simulation is terminated.

In addition, two light switches exist that can turn on/off the lights in the living room and bedroom. The light strength $PLtStr$ in the living room and when the light is turned on or off is given by the following equation:

$$PLtStr = \begin{cases} rand()/RAND\_MAX \times 50 + 50 & \text{light off} \\ rand()/RAND\_MAX \times 50 + 350 & \text{light on} \end{cases}$$

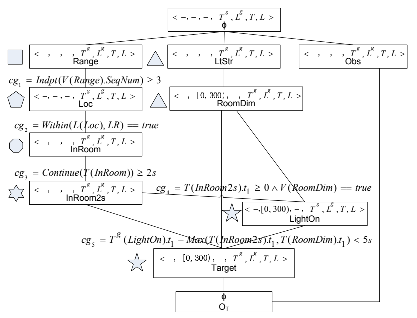
where 300 is the threshold below which the room is determined to be dim.

Finally, two additional "traditional" approaches are simulated for event detection in Tossim using the same experimental configurations as the CPS event model implementation. The first approach is referred as "Event Detection (ED) by counting", where instead of using the timestamps of lower level events and generating events based on the explicit timing constraints, the events are defined by counting the number of received events from the lower level. For example, the "ED by counting" implementation counts the number of consecutively received $InRoom$ events and when the number exceeds a predefined threshold, say 3, a new event $InRoom2s$ is generated. This approach is commonly used in non-real-time systems where timelines is not a critical issue.

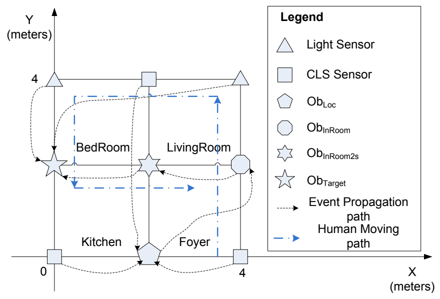
The second approach is referred as "timestamp at App.", where observers (applications) do not differentiate between event- occurrence time and event-generation time, and directly assign the event-generation time as the corresponding event-occurrence time. For example, in the "timestamp at App." implementation, the occurrence timestamp of an $InRoom$ event is assigned based on its generation timestamp simply because both the lower level events $Range$ and Loc do not contain occurrence timestamps in their event bodies for this implementation. This approach is commonly used in small-scale systems where the event propagation time and event processing time is negligible compared to the timing constraints defined at the higher level.

5.3 Simulation Results

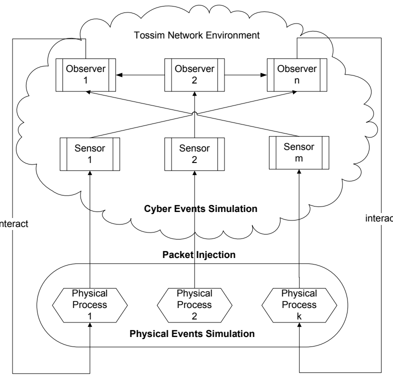
The following three QoS metrics are defined and used for each



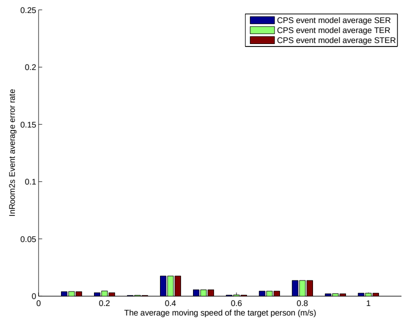
(a) CPS concept lattice of the Smart Space example



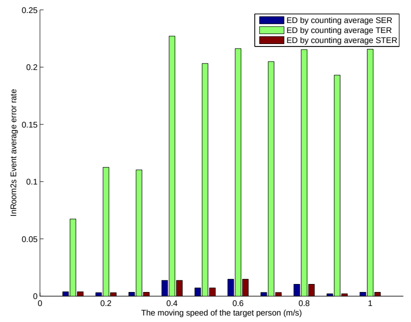
(b) The Smart Home network topology



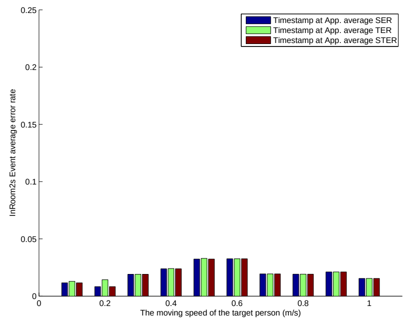
(c) Overview of the CPSim



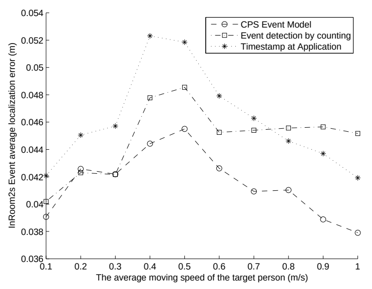
(d) Average SER, TER and STER for CPS event model



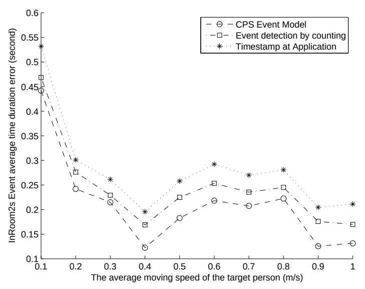
(e) Average SER, TER and STER for Event Detection by counting



(f) Average SER, TER and STER for timestamp at Application



(g) Average localization error



(h) Average time duration error

Figure 3: Smart Space application simulation setup and results

implementation type: event spatial error rate (SER), event temporal error rate (TER) and event spatio-temporal error rate ($STER$):

$$SER(\mathcal{E}_{cps}, LC) = \frac{\text{\# of } \mathcal{E}_{cps} \text{ s.t. } L(\mathcal{E}_{cps}) \text{ does not satisfy } LC}{\text{total \# of } \mathcal{E}_{cps}}$$

where $L(\mathcal{E}_{cps})$ is the occurrence location function defined in equation (8) and LC is the spatial constraint specified by the user. Similarly,

$$TER(\mathcal{E}_{cps}, TC) = \frac{\text{\# of } \mathcal{E}_{cps} \text{ s.t. } T(\mathcal{E}_{cps}) \text{ does not satisfy } TC}{\text{total \# of } \mathcal{E}_{cps}}$$

where $T(\mathcal{E}_{cps})$ is the occurrence time extraction function defined in equation (6) and TC is the temporal constraint specified by the user. Finally,

$$STER(\mathcal{E}_{cps}, LC, TC) =$$
$$\frac{\text{\# of } \mathcal{E}_{cps} \text{ s.t. } L(\mathcal{E}_{cps}),\ T(\mathcal{E}_{cps}) \text{ do not satisfy } LC \text{ and } TC}{\text{total \# of } \mathcal{E}_{cps}}$$

where $STER$ counts the percentage of time that an event satisfies neither TC nor LC at the same time. Clearly, the SER, TER and $STER$ are metrics that examine how well the target event $\mathcal{E}_{cps}$ satisfies the user specified spatial, temporal or spatio-temporal QoS constraints. In the simulations we focus on examining event $InRoom2s$ (which stands for "staying in a room for at least 2 seconds") since this event has both spatial and temporal constraints. The LC is defined as "$L(InRoom2s)$ occurs in the same room that the $InRoom2s$ event actually occurs" and TC is defined as "the duration of $T(InRoom2s)$ is greater or equal to 2 seconds".

Fig. 3(d), 3(e), 3(f) show the average SER, TER and STER results for event $InRoom2s$ when the target person walks at speeds from $0.1m/s$ to $1m/s$ for the three implementations. Each simulation configuration runs for 100 iterations, and the average value for each metric is plotted. Fig. 3(d) shows that the CPS event model keeps a constant low SER, TER and STER of less than 1.8% for all cases. Fig. 3(e) shows that the "ED by counting" approach has TER above 6.7% for all cases. This is expected because this approach does not use explicit timing constraints to generate new events. Fig. 3(f) shows that the "timestamp at App." approach has constant larger SER, TER, and STER than the CPS event model approach. Another interesting result is that for all three approaches, the SER and STER are almost always the same for all cases. This is because the spatial error is usually caused by the temporal error in this application. Fig. 3(g) shows the average localization error for event $InRoom2s$. The localization error is the average distance differences between the reported event occurrence location and the actual event occurrence location. Localization for the three implementations is relatively accurate, with the error always less than $6cm$, which is to be expected given the low range error noise injected into the system. Fig. 3(h) shows the average time duration of the $InRoom2s$ event. The time duration error is the average time difference between the reported event duration and the actual event duration. We can see that as the person's walking speed varies, the event time duration error varies from around 500ms to 50ms. Again, the CPS event model implementation always has the lowest average localization error and average time duration error for all cases in both figures.

In summary, compared to the CPS event model implementation, the traditional approaches (i.e., the "ED by counting," which is also the original approach taken by our *Smart Space* Java program, and the "timestamp at App.") have the following disadvantages: 1) non-uniform event representation in different system layers, e.g., the lower level CLS nodes only provide their sensor readings but lack timestamps, making it impossible for higher level applications to specify explicit spatio-temporal constraints. 2) No differentiation between event occurrence time/ location and event generation time/ location. Thus, high spatio-temporal error rates are possible for events, even in small-scale systems. 3) Since observer information is also omitted in *every level* of the events, traditional approaches have to hard-code all of the necessary observer information, and generally assume no change in observer status, which precludes observers dynamically joining and leaving the system. Moreover, any change in the definition of events or configuration of the system requires a complete re-compilation of the program. 4) An "event" in traditional approaches, e.g., the original Java program, is usually implemented as a nested "*if-else*" statements with several redundant code blocks. Consequently, the overall code space for the Java implementation is large (about 2400 lines).

In contrast, CPSim, which implements the CPS event model is 1200 lines and has the following advantages: 1) Unified representation of event instances with both temporal and spatial information preserved, so that it is possible to specify timing and location constraints for events. 2) Explicitly differentiates event occurrence time/ location and event generation time/ location, providing low spatial-temporal error rates. 3) In additional to the spatio-temporal information, the event also includes observer information. As a result, the system is highly flexible, which allows the observers to dynamically join and leave the system without affecting independent components of the CPS. 4) Finally, a decrease of 50% of the code space compared to the Java implementation because of support for event composition and the unified event representation.

6. CONCLUSION

In this paper, we present a concept lattice-based event model for Cyber-physical systems (CPS). The developed model not only captures the essential information about events in a distributed and heterogeneous environment, but it also allows events to be composed across different boundaries of different components and devices within and among both cyber and physical domains. The treatment of observers as events results in a complete concept lattice. A smart home example is used to illustrate the application of the model, and a CPS event simulator is implemented and tested.

The composition rules given in Section 4 are essentially *horizontal*, meaning that all the composing events directly link to a composed event as in the lattice. However, *vertical* compositions of the form $\Gamma_1\,\mu_1\,@\,(\mathcal{T}_1,\,\mathcal{L}_1,\,\Gamma_2\,\mu_2\,@\,(\mathcal{T}_2,\,\mathcal{L}_2,\,\mathcal{O}_2))$ are not systematically covered in this paper except for the globalization function which can be seen as a special case of vertical composition. The treatment of vertical compositions in the developed event model is left as a future work.

7. ACKNOWLEDGEMENTS

This material is based in part upon work supported by the U.S. Department of Agriculture under cooperative agreement USDA FCIC/RMA 2IE08310228 and NSF CAREER Award CNS0746643.

8. REFERENCES

[1] T. Abraham and J. F. Roddick. Survey of Spatio-Temporal Databases. *Geoinformatica*, 3(1):61–99, 1999.

[2] R. Adaikkalavan and S. Chakravarthy. SnoopIB: interval-based event specification and detection for active databases. *Data Knowl. Eng.*, 59(1):139–165, 2006.

[3] H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Rule-Based Runtime Verification. *5th International Conference on Verification, Model Checking and Abstract Interpretation*, 2937:277–306, 2004.

[4] S. Chakravarthy. Sentinel: an object-oriented DBMS with event-based rules. *SIGMOD Rec.*, 26(2):572–575, 1997.

[5] R. Cole and P. Eklund. Scalability In Formal Concept Analysis. *Computational Intelligence*, 15:11–27, 1999.

[6] R. Cole, P. Eklund, and G. Stumme. {CEM} – A Program for Visualization and Discovery in Email. In *Proc. Principles of Data Mining and Knowledge Discovery.*, volume 1910, pages 367–374, 2000.

[7] M. J. Egenhofer and K. K. Al-Taha. Reasoning about Gradual Changes of Topological Relationships. *Proc. of the International Conference GIS*, pages 196–219, 1992.

[8] V. Ermolayev, N. Keberle, and W.-E. Matzke. An Ontology of Environments, Events, and Happenings. In *Proceedings of the 32nd Annual IEEE International Computer Software and Applications, COMPSA'08*, 2008.

[9] S. Gatziu and K. R. Dittrich. Detecting Composite Events in Active Database Systems Using Petri Nets. *RIDS'94: Proceedings 4th International Workshop on Active Database Systems*, pages 2–9, 1994.

[10] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1–11, New York, NY, USA, 2003. ACM.

[11] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Composite Event Specification in Active Databases: Model & Implementation. In *Proc. of the 18th Intl. Conference on Very Large Data Bases*, pages 327–338, 1992.

[12] B. Groh and P. W. Eklund. Algorithms for Creating Relational Power Context Families from Conceptual Graphs. In *Proceedings of the 7th International Conference on Conceptual Structures*, pages 389–400, 1999.

[13] M. Kim and M. V. Etc. Formally specified monitoring of temporal properties. *Proc. of the 11th Euromicro Conference on Real-Time Systems*, pages 114–122, 1999.

[14] M. Kim and S. K. Etc. Java-MaC: a Rigorous Run-time Assurance Tool for Java Programs. *Formal Methods in System Design*, 24(2):129–155, 2004.

[15] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, 1986.

[16] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[17] E. A. Lee. Cyber Physical Systems: Design Challenges. *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369, May 2008.

[18] I. Lee and O. Sokolsky. Semantics of high-level event definition. In *First Workshop on Event-based Semantics*, 2007.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM.

[20] C. Lindig and A. Softwaretechnologie. Concept-Based Component Retrieval. In *Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs*, pages 21–25, 1995.

[21] M. Liquiere and J. Sallantin. Structural machine learning with Galois lattice and Graphs. In *Proc. of the 1998 Int. Conf. on Machine Learning*, pages 305–313, 1998.

[22] D. McCarthy and U. Dayal. The architecture of an active database management system. *SIGMOD Rec.*, 18(2):215–224, 1989.

[23] A. K. Mok, P. Konana, and G. L. Etc. Specifying timing constraints and composite events: an application in the design of electronic brokerages. *IEEE TSE*, 30(12):841–858, 2004.

[24] A. K. Mok, C.-G. Lee, H. Woo, and P. Konana. The monitoring of timing constraints on time intervals. *RTSS '02: Proceedings of the 23rd IEEE International Real-Time Systems Symposium*, pages 191–200, 2002.

[25] E. T. Mueller. Automating commonsense reasoning using the event calculus. *Commun. ACM*, 52(1):113–117, 2009.

[26] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43, New York, NY, USA, 2000. ACM.

[27] S. Shen, C. Xia, R. Sprick, L. C. Perez, and S. Goddard. Comparison of three Kalman filters for an indoor passive tracking system. In *Electro/Information Technology, 2007 IEEE International Conference on*, pages 284–289, May 2007.

[28] G. Snelting. Reengineering of configurations based on mathematical concept analysis. *ACM Trans. Softw. Eng. Methodol.*, 5(2):146–189, 1996.

[29] G. Snelting and F. Tip. Reengineering Class Hierarchies Using Concept Analysis. In *In ACM Trans. Programming Languages and Systems*, pages 99–110, 1998.

[30] C. Talcott. Cyber-Physical Systems and Events. In *Software-Intensive Systems and New Computing Paradigms*, pages 101 – 115, Berlin, Heidelberg, 2008. Springer-Verlag.

[31] Y. Tan, M. C. Vuran, and S. Goddard. Spatio-Temporal Event Model for Cyber-Physical Systems. *International Conference on Distributed Computing Systems (ICDCS) Workshops on Cyber-Physical Systems*, 0:44–50, 2009.

[32] R. Wille. Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies. *Lecture Notes in Computer Science*, 3626:1–33, 2005.

[33] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In *ICFCA '09: Proceedings of the 7th International Conference on Formal Concept Analysis*, pages 314–339, Berlin, Heidelberg, 2009. Springer-Verlag.

[34] Y. Yu, S. Ren, and O. Frieder. Interval-Based Timing Constraints Their Satisfactions and Applications. *IEEE Transactions on Computers*, 57(3):418–432, 2008.