Mining Temporal Constraints among IoT Services in Smart Home Environments

***Abstract*—The smart home is responsible for orchestrating many IoT devices that provide various kinds of services. However, it is challenging for residents, the end-users of smart homes, to develop proper IoT service orchestration rules, because there are a lot of implied temporal constraints among the states and events of IoT devices, which are derived from users' habits, preferences, etc. This paper proposes an approach to mining temporal constraints among IoT services in smart home environments from logged event traces. The approach first models the event trace of a smart home by augmenting them with contexts and IoT device states. It then proposes three types of constraints that specify the temporal relations of (1) a set of events (event-event), (2) a pair of states (state-state), and (3) an event and a state (event-state). After that, the approach designs the constraint mining methods that identify the three types of constraints. Finally, this work performs a preliminary experiment and the experimental results show that the proposed approach is effective in extracting the different kinds of temporal constraints in smart homes. Compared with the existing work that only mining event sequence patterns, this paper systematically defines multiple kinds of constraints and proposes effective mining methods. The obtained constraints are useful for creating IoT service orchestration rules and detecting their behavior deviations.**

***Index Terms*—smart home; temporal constraint; IoT service; service orchestration; trigger action programming**

I. Introduction

The smart home is a prominent application domain of the Internet of Things (IoT) technology. The smart home revenue amounts to $69,551 million and presents an annual growth of 20.3% [1]. A market forecast claims that the global market will grow to over $53 billion by the year 2022 [2].

A smart home refers to a regular home augmented with various types of IoT devices that provide services (i.e, IoT services) for improving comfortableness and convenience of occupants' daily lives, i.e., intelligently controlling lighting, switches, locks, entertainment systems, and other appliances. Furthermore, end-users can make smart homes more efficient by orchestrating multiple IoT services together through TAP (trigger-action programming) rules [3], to customize their own smart homes following their habits, preferences and requirements [4]. In this way, novice users create event-driven rules in the form of "**IF** a *trigger* occurs, **THEN** do an *action*". Such TAP rules are also known as ECA (Event-Condition-Action) rules, which automate smart home IoT devices to provide proper services when some events happen and some specific conditions are satisfied. There are many tools for creating TAP rules, e.g., IFTTT[1], Zapier[2], OpenHAB[3], Home Assistant[4], etc.

However, creating TAP rules is still challenging for novice end-users. A smart home is a complex cyber-physical system (CPS) that involves humans, devices, physical environmental contexts (contexts for short hereafter) and computing systems, making the created TAP rules often inconsistent with user mental models [5]. Although many efforts are devoted to automating TAP rules generation by leveraging program synthesis [3] and machine learning techniques [6], developing correct TAP rules is still thorny.

The other way of creating TAP rules is based on human activity pattern mining. Smart homes are human-centric. Residents' daily lives are often regular following their careers, habits, and preferences. Therefore, the extracted activity patterns [7], especially the usage patterns of devices and appliances in houses [8], are used for designing TAP rules and IoT service orchestrations. Although useful, the existing activity pattern mining based approaches are not efficient enough. Some limitations are listed as follows.

Firstly, most of the approaches only focus on events (or occupants' activities which generate the events), without considering the contexts and IoT device states. The approaches obtain patterns derived from event (or activities) traces, e.g., periodic device usage patterns [8], correlation patterns [7], and sequential temporal patterns [9]. However, the approaches are limited in finding fine-grained invariants and constraints among contexts, events and IoT device states. For example, "*when the inner temperature goes above 30 centigrade and while the air conditioner (AC) is off, turn on the air conditioner*" is a simple rule relating to an event, the environmental temperature and a device state, but most of the existing approaches do not consider such kind of rules.

Secondly, the patterns usually represent temporal sequences [8] that specify the temporal relations among events and the fixed time when the events happen. Nevertheless, although human activities are regular in the long term, uncertainties and randomness still exist. For example, in summer a user always *turns on an AC* and *closes the window* together, but the temporal sequences between the two actions might be random. That is, the user might turn on the AC and then close the window at this time and perform the two actions reversely at the next time. Moreover, the two actions would happen at

[1]https://ifttt.com/
[2]https://www.zapier.com
[3]https://www.openhab.org
[4]home-assistant.io

different times according to the weather and the temperature. In consequence, rather than specify a rule working at the fixed time, it is meaningful to extract two fine-grained invariant constraints, i.e., (1) *the AC is on and the window is open always hold together* and, (2) *the AC always turns on only if the temperature goes above a certain degree*.

This paper proposes an approach to tackle the above limitations, by mining fine-grained temporal constraints among IoT services in smart home environments. The approach first models event traces of a smart home by augmenting them with contexts and IoT device states. It then proposes three types of constraints, which specify the temporal relations of (1) a set of events (*event-event*), (2) a pair of states (*state-state*), and (3) an event and a state (*event-state*). After that, the approach designs constraint mining methods that discover the three types of constraints. The obtained constraints can be further transformed into LTL (linear temporal logic) specifications for automated TAP rules generations [3] and smart home behavior deviation detections.

In summary, the contributions of this paper are as follows.

(1) This paper proposes a model for a smart home, which augments smart home event traces with device states, temporal features and physical environmental contexts.

(2) This paper systematically proposes three types of temporal constraints and realizes the methods for mining them in smart home event traces.

(3) This paper performs a preliminary experiment and the experimental results show that the approach is effective in finding valid constraints.

The rest of this paper is organized as follows. Section II analyzes the problem. Section III, IV and V elaborate the approach, including the model, the temporal constraint patterns definitions and the mining methods. Section VI evaluates the approach with an experiment. Section VII presents the related work, and finally, Section VIII gives a conclusion.

II. Problem Analysis

Residents expect smart home systems to perform as they want, and on the contrary, these novice users are either inability to design proper (or correct) IoT service orchestration rules or specify their expectations precisely and clearly [5]. The temporal constraints are essential to resolve the contradiction because they are effective in helping end-users to design TAP rules [3] and detect deviations from expected system behaviors.

The previous study finds that a DIY (Do-It-Yourself) style smart home usage cycle contains six stages [10], and where end-users perform four stages, i.e., *motivation*, *implementation*, *use-through-routine* and *routinization*). They iteratively design, evaluate and modify IoT service orchestration rules to make the rules follow their daily routines without disturbing their normal lives.

In such processes, end-users operate devices and appliances themselves as if they are in ordinary homes, e.g., lock/unlock doors, turn on/off lights, open/close windows, etc. These human activities result in the corresponding events on specific devices, which reflect end-user intentions, daily routines and their reactions to environmental contexts. For example, a user *turns on an AC* because he *feels hot* and *turns on a light when the room is dim*. Meanwhile, smart home systems monitor and store the events on IoT devices and device states, and various sensors can obtain different kinds of environmental contexts. In consequence, The stored data forms event traces that implying regularities and expected rules of end-users. We are motivated to mine temporal constraints among IoT services form event traces.

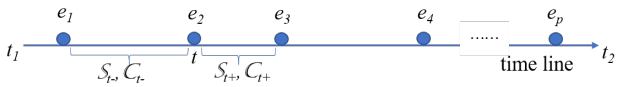

Fig. 1. The Illustration of An Event Trace in A Smart Home.

III. Modeling

A smart home $\mathbb{M}$ comprises a set of IoT devices $\mathbb{D} = \{D_1, D_2, \cdots, D_n\}$, where $\forall D_i \in \mathbb{D}(1 \leqslant i \leqslant n)$ is an IoT device.

A smart home device D_i is defined as a tuple $D_i =< d_i, S_i, F_i, l_i >$, where,

- d_i is a unique device identifer.
- l_i denotes the location of a device, e.g., a living room, a bedroom, etc.
- F_i represents a set of services provided by D_i. For example, an AC can provide *warming*, *cooling* and *ventilation* services.
- $S_i = \{s_{on}, s_{off}, S_i'\}$ is the state set of D_i, where there are two requisite states, i.e., s_{on} and s_{off}, representing the initial state and the final state respectively. Besides, S_i' contains the other additional states if the device has.

The state of a smart home $\mathbb{M}$, represented as $\mathbb{S} = S_1 \times S_2 \times \cdots \times S_n$, is the Cartesian product of every IoT device's state set. At a specific time t, the overall state is a vector, $\mathbb{S}_t =< s_{1,t}, s_{2,t}, \cdots, s_{n,t} >$, and $s_{i,t}(1 \leqslant i \leqslant n)$ is the state of the i-th IoT device at t.

The contexts of a smart home, obtained by a set of IoT sensors, are represented as $\mathbb{C} = \{c_1, c_2, \cdots, c_m\}$, where $\forall c_j \in \mathbb{C}(1 \leqslant j \leqslant m)$ is a context instance sensed by a certain sensor. The typical contexts are *temperature*, *humidity*, *luminance*, etc. At t, the overall context is $\mathbb{C}_t =< c_{1,t}, c_{2,t}, \cdots, c_{m,t} >$.

An event happening at the time t is defined as a tuple $e =< t, l, \mathbb{S}_{t-}, \mathbb{S}_{t+}, \mathbb{C}_{t-}, \mathbb{C}_{t+} >$, where t and l are the timestamp and the location when and where the event e happens. $\mathbb{S}_{t-}, \mathbb{S}_{t+}$ are the states of the smart home before and after the event e happens, and similarly, $\mathbb{C}_{t-}, \mathbb{C}_{t+}$ denote the contexts obtained before and after e.

User's operations on smart home devices result in a set of events that change the states of the devices. Within the smart home, there are unlikely two or more events occurring at the same instant in time [5] and one event usually changes one device's state. In consequence, the events happening in

a period (e.g., one day, one week, one month and one year) form a temporal sequence.

An event trace of a period is a set of events happening in temporal sequence, represented as $E(t_1, t_2) =< e_1, e_2, \cdots, e_p >$, where t_1 and t_2 are the start time and the end time respectively, and $e_1, e_2, \cdots, e_p$ are events happening in sequence. Figure 1 illustrates an event trace augmented with device states and contexts.

An event trace can be simply represented as a string comprising a set of symbols, where each symbol represents an event. It is worth noting that the same events, represented with the same symbols, might recur multiple times, according to the regularity of a resident's daily life. For example, if x represents an event "*turn on the light in the living room*", then a logged event trace represented as a string might contain multiple x since the event instances recur several times.

Based on the modeling, the problem of mining temporal constraints is abstracted into extracting frequent sequence patterns from the augmented event traces.

IV. Temporal Constraints

Events and states are the two main kinds of elements in developing smart home TAP rules. There are three temporal paradigms [5] among them, i.e., **event-event**, **state-state**, as well as **event-state**. This paper proposes three types of temporal constraints based on the existing studies and our observations.

A. Event-event constraints

The constraints of this type specify the temporal relations among events. The basic constraint pattern is represented as "**event** e_1 **occurs [within time interval** T**] after event** e_2 **occurs**". For instance, when using a kettle to boil water, two events always happen in sequence, i.e., *turn on the kettle* and *turn off it after a while*. The former event is triggered by a user's operation and the latter happens automatically when the inner water is boiled. Furthermore, the time interval between these two events might vary slightly each time, according to the water volume. Multiple event pairs form a temporal sequence and each event in it would happen at a certain time.

The event-event constraint is defined as $cons(E'(t_1, t_2))$, where $E'(t_1, t_2)$ is a sub-trace of $E(t_1, t_2)$ (see Fig. 1). That is, E' contains a sub set of events in E, and $cons(E'(t_1, t_2))$ specifies the events in E' must happen in the fixed order within the fixed time period. It is worth noting that there might be multiple instances of E' in E.

B. State-state constraints

The constraints of this type specify the temporal invariants among the expected states of different devices, without regarding what events happen and what the event sequences are. The constraint pattern is represented as "**the state of device** d_1 **is** s_1 **when the state of the other device** d_2 **is** s_2**.**" For example, "*an AC is off while the window is open*" specifies the expected states of the two devices. The states changes are caused by two events (i.e., *turn on the AC* and *close the window*) whose sequences are random because the user might not operate the devices in the same orders every time. That is, unlike the previous constraint type, the event pairs implying state-state constraints might happen randomly only if they happen in a very short period, e.g., opening windows and turning off an AC would happen in the morning, at noon or at night. Whenever the events happen, the expected states must hold.

The state-state constraint is defined as $cons(s_1, s_2)$, which specifies the states s_1 and s_2 must hold together. s_1 and s_2 are states of two different IoT devices (e.g., D_1, D_2), and two events (e_1 and e_2 happen in a very short time interval, denoting the two states after them are expected simultaneously) change the device states respectively.

C. Event-state constraints

In a smart home some events would happen when the system is in the corresponding states with suitable contexts. This constraint pattern is represented as "**an event** e **occurs [at time** t**] [while state is** S**][and context is** C**]**" For example, a user would *close the window when it is raining and the window is opened now*. In contrast, the event *close the window* would not happen if it is neither raining nor the window is closed. In essence, constraints of this type specify the conditions that trigger the specific events. The conditions are grouped into three types, e.g., *time*, *device states* and *physical environmental contexts*. Note that all the conditions are optional for a constraint.

The event-state constraint is defined as $cons(t, s, c, e)$, where t, s, c represent the specific time, the device states and the environmental contexts that trigger the event e, and t, s, c can be empty. $cons(t, s, c, e)$ specifies an invariable rule where an event would happen when the conditions are satisfied.

V. Mining the Temporal Constraints

This section elaborates the methods for mining different types of constraints from the augmented event traces defined in Section III.

A. Mining event sequence patterns

Without loss of generality, it is assumed that there are two kinds of events happening on a device, i.e., *turn on* and *turn off*. An event type is represented as an identity comprised of a letter and a number and the identity of each event type is unique. An event instance is represented as the joint of the type identity and its occurring time. For instance, given a smart home device D_i, its *turn-on* and *turn-off* event are represented as $A0$ and $a0$. Moreover, an instance of $A0$, represented as $A01220$, denoting it happens at 12:20 in a day.

$cons(E'(t_1, t_2))$, in the form of an event sequecne, is the generalized event-event constraint, and PrefixSpan [11] is a high-performance algorithm for mining sequential patterns. Its general idea is to examine only the prefix subsequences and project only their corresponding postfix subsequences into projected databases. In each projected database, sequential patterns are grown by exploring only local frequent patter.

By using PrefixSpan algorithm, the event trace of one day is a sequence α and all the items in it are event instances. All the sequences in a period (e.g., several days, weeks and months) form a sequence database DB. In consequence, the problem of extracting event-event constraints is abastracted into mining event sequence patterns from a sequence database containing a set of event sequences. The mining process is peformed in several steps as follows.

Step 1: Mine the sequence database DB by using PrefixSpan and output a set of sequence patterns. The occurrence of each obtained pattern must be equal to or larger than the pre-defined support h_s. In particular, the algorithm is optimized by regarding the feature of smart homes, i.e., *turn-on* and *turn-off* should occur in pairs, and the former event should occur before the latter. In consequence, the algorithm optimizations are: (1) when constructing a new prefix, a *turn-off* event should be ignored if there is not the corresponding *turn-on* event in the current prefix p, even though its occurrence exceeds h_s; (2) given a prefix p and its projection set $S\mid_p$, the operation of counting the occurrence of each kind of events, when the first event in $S\mid_p$ is found that can form an event pair with the other event in p.

Step 2: Make the filtering of the obtained event patterns in step 1 and get the final pattern set P. A pattern is thought incompleted if it contains some standalone *turn-on* (or *turn-off*) events that cannot form event pairs with other events. The reason is that *turn-on* and *turn-off* events of a device should happen in pair in a pattern.

Step 3: Recommend happening time for each event instance in P, by making statistics of the event occurrence time. This paper proposes two recommendation strategies. (1) The average time, for each type of event, take the average occurrence time of its instances happening in everyday sequence as the recommendation. (2) The time range, generate a time range for each event, according to the earlist and the latest time when the event instances happen in the sequence pattern set P.

B. Mining state-state constraints

Events usually change device states, e.g., *turning on an AC* changes the AC state from *off* to *on*. We notice that (1) one event usually changes one state and (2) if two events happen in a very short time interval, the states they lead to might be expected to hold together with high probability. For example, a resident always *turns on* (or *turns off*) his television and cable STB (set-top box) together. The two events always happen in a very short time interval (e.g., several seconds) without in the fixed orders. In such cases, it is inferred that the television and the cable STB are expected either *on* or *off* together.

Our method mines state-state constraints by analyzing state changes caused by the events in the given trace $E(t_1, t_2)$. The method works as the follows.

Step 1: The method traverses the event traces E and identifies event pairs. The two events in a pair are adjacent and happen in a very short time interval (evaluated by a pre-defined threshold t_h). For example, given two adjacent events e_k and e_{k+1} in E, if the time interval $(t_{k+1} - t_k) \leq t_h$, e_k and e_{k+1} are identified as a candidate event pair. This step finally gets a set of event pairs E_p.

Step 2: For each event pair in E_p, this step analyzes the states before and after each event and takes the changed states after the two events as a pair, represented as $\mathbb{S}_{t_k+}$ and $\mathbb{S}_{t_{k+1}+}$. It is assumed that one event only changes one device's state, in other words, there only one element in $\mathbb{S}_{t_k+}$ and $\mathbb{S}_{t_{k+1}+}$ changes respectively before and after e_k and e_{k+1}. As such, for simplicity, we use the changed elements to represent the whole states. For example, e_k changes the state of device D_i from $s_{i,off}$ to $s_{i,on}$, and similarly, e_{k+1} changes the state of device D_j from $s_{j,off}$ to $s_{j,on}$. In consequence, a state pair $< s_{i,on}, s_{j,on} >$ is obtained, which represents the overall changed states. This step finally gets a set of state pairs SP.

Step 3: This step identifies the state pairs occurring frequently (occurring not less than a pre-defined support h_f)and take them as the final state-state constraints. For example, if the occurrence of $< S_{i,on}, S_{j,on} >$ in SP, represented as $o(S_{i,on}, S_{j,on})$, is larger than h_f, then "*devices D_i and D_j are always on together*" is identified as a state-state constraint.

C. Mining event-state constraints

Event-state constraints specifies the invariant conditions that trigger the specific events, e.g., physical environmental contexts, device states and time. The mining method is as follows.

Step 1: Identifies the events occurring frequently (evaluated by a pre-defined support h_o) in a trace E and groups them into a set $ES = \{E_1, E_2, \cdots, E_o\}$, where $\forall E_l \in ES (1 \leq l \leq o)$ is the event instance set of a same type. For E_l, whose occurrence is represented as O_{E_l}, and $O_{E_l} \geq h_o$.

Step 2: For each element in ES, collect the conditions triggering its instances. For a certain type of events $E_l = \{e_{l,1}, e_{l,2}, \cdots, e_{l,q}\}$, (1) get all the state spaces before the event instances, represented as $\{\mathbb{S}_{l,1}, \mathbb{S}_{l,2}, \cdots, \mathbb{S}_{l,q}\}$; (2) get all the timestamps of the event instances, represented as $\{t_{l,1}, t_{l,2}, \cdots, t_{l,q}\}$; (3) get all the corresponding contexts of the event instances, in general, a kind of device relates to a specific kind of physical environmental context (e.g., an AC relates to the temperature, a light relates to the luminance, etc.), therefore the instances of the corresponding type of context C_{E_l} are $\{C_{l,1}, C_{l,2}, \cdots, C_{l,q}\}$.

Step 3: For each type of event in ES, extract its invariant conditions if exist. Given the type of event E_l, (1) get the states of devices (denoted as $\{s_1, s_2, \cdots, s_p\}$) that occurring in $\mathbb{S}_{l,1}, \mathbb{S}_{l,2}, \cdots, \mathbb{S}_{l,q}$; (2) construct a time range ($[t_s, t_e]$) which covers the majority happening time of the event instances in E_l; (3) get the physical environmental context relating to the events, take the boundary value (c_l) of the context as the contextual condition triggering this type of events.

VI. Evaluation

We evaluate the approach proposed in this paper by performing a preliminary experiment.

TABLE I
THE DEPLOYMENT OF IOT DEVICES IN THE HOUSE.

	Room	Device (ID)
0	living room	light (A0), window (B0), curtain (C0), AC (F0), TV (G0), robot vacuum (H0), temperature sensor (Y0), humidity sensor (Z0)
1	bedroom-1	light (A1), window (B1), curtain (C1), door (D1), table lamp (E1), motion sensor (X1), temperature sensor (Y1), humidity sensor (Z1)
2	bedroom-2	light (A2), window (B2), curtain (C2), door (D2), table lamp (E2), motion sensor (X2), temperature sensor (Y2), humidity sensor (Z2)
3	kitchen	light (A3), coffee machine (I3), kettle (J3)
4	bathroom	light (A4), door (D4), water heater (K4), motion sensor (X4)

TABLE II
THE MAIN DAILY ACTIVITIES OF THE RESIDENTS.

Time	People	Activities
07:00	father	Washes in the bathroom and then cooks breakfast in the kitchen.
07:30	child	Washes in the bathroom and have breakfast with father in the kitchen.
08:30	father, child	Go to work and school respectively.
17:30	child	Returns home and does his home work in bedroom-2.
18:00	father	Returns home and prepares for dinner in the kitchen.
18:30	child	Watches TV in the livingroom.
19:00	father, child	Have their dinner in the kitchen.
19:30	father, child	Watch TV or play games together in the livingroom.
22:30	father, child	Shower in the bathroom.
23:00	father, child	Go to sleep respectively.

A. Experimental setup

1) Experimental environment and scenarios: We constructed an experimental environment by deploying multiple kinds of IoT devices in a house. The house comprises five rooms, i.e., one living room, two bedrooms (bedroom-1 and bedroom-2), one kitchen and one bathroom. Two residents live in the house, i.e., the father (whose bedroom is bedroom-1) and the child (whose bedroom is bedroom-2). Note that one author of this paper is the house owner. There are various smart home IoT devices (including sensors and smart devices) in the house. The deployment details are listed in Table I, where each device is represented by its turn-on event. We enhanced an open-source smart home system, i.e., Home Assistant, for logging events and device states, by improving its event scheduling and management mechanism. It is worth noting that all the IoT devices used in the experiment are controlled by connecting them to the system.

The main daily activities of the two residents are listed in Table II. Note that Table II just lists the approximate time of the residents' activities and the exact time of each activity might be slightly different every day.

2) The dataset: We collected 15 days' data of resident activities and device usage records in the experimental environment and took it as the experimental dataset.

B. Experiment and evaluation

The experiment aims to evaluate the effectiveness of the proposed approach and analyze the influences of parameter settings (e.g., support and time interval threshold.) on the mining performance. We use *recall* as the evaluation metrics.

Overall, the obtained constraints are shown in Table III, IV and V, respectively. The constraints are extracted by the mining methods with different algorithm parameter settings. The constraints in the tables are true ones after manual checking.

Note that, for simplicity, we only present the context conditions of event-state constraints, without listing other conditions that specify the conditions of device states and the happening time.

1) Evaluation of mining event-event constraints: The key points in constraint mining are, (1) making the trade-off between "*finding true constraints and filtering out false ones as many as possible*", and meanwhile, (2) limiting the total number of valid constraints because too many constraints make the system too complicated to control. As such, setting the algorithm parameter (i.e. support) properly is crucial to the final mining result.

In this experiment, the support (h_s, see Section V-A) is set with 11 at the beginning and increased to 15 finally. The results are shown in Table VI. Four constraints are identified when setting h_s with 11. The recall decreases step by step along with the increase of h_s, and it finally becomes 0% when the support is 15. Note that we tried to set h_s smaller than 11, but we found that the identified patterns increase multiple times, which would complicate the system significantly.

In our experiment, the event patterns occurring in about 70% (11/15) sequences are thought of as true ones. In consequence, it is reasonable for setting the support with the number of 70% total sequences.

2) Evaluation of Mining State-State Constraints: Mining state-state constraints are affected by two factors, i.e., the support (h_f) and the time interval (t_h). We observed that the parameters are correlated, i.e., the shorter the time interval, the less frequently the constraints occur and vice versa. In consequence, we set the time interval with 60, 120 and 180 (denoting 1, 2 and 3 minutes) respectively, and meanwhile, we set the support with different values according to the corresponding time interval.

The overall evaluation results are shown in Table VII (TP denotes true positive and FN denotes false negative). On the one hand, when the time interval is fixed, the recall decreases along with the increase of support. On the other hand, to obtain the same performance (i.e., recall), the support should be smaller if the time interval is shorter.

TABLE III
EXPERIMENTAL RESULT OF MINING EVENT-EVENT CONSTRAINTS.

No.	Constraint	Occurrence	Description
1	D4(open) - A4(turn on) - K4(turn on) - k4(turn on) - a4(turn off) - d4(close)	14	take a shower
2	C1(open) - D1(open) - A1(trun on) - c1(close) - d1(close) - a1(turn off)	12	father's activities in his living room
3	C2(open) - D2(open) - A2(trun on) - c2(close) - d2(close) - a2(turn off)	11	child's activities in his living room
4	A2(turn on) - a2 (turn off) - A3(turn on) - G0(turn on) - g0(turn off) - a3(turn off)	13	child's acitvities from return home to finish the dinner

TABLE IV
EXPERIMENTAL RESULT OF MINING STATE-STATE CONSTRAINTS.

No.	Constraint	Occurrence	Description
1	A0 and G0 are both *on*.	13	A0 changes to on firstly 9 times and the other 4 times are the reverse orders. The two state changes happen within 1 minute (5 times), 2 minutes (6 times) and 3 minutes (2 times).
2	I3 and J3 are both *on*.	15	I3 changes to on firstly 8 times and the other 7 times are the reverse orders. The two state changes happen within 1 minute (3 times), 2 minutes (7 times) and 3 minutes (5 times).
3	I3 and J3 are both *off*.	15	I3 changes to off firstly 8 times and the other 7 times are the reverse orders. The two state changes happen within 1 minute (3 times), 2 minutes (8 times) and 3 minutes (4 times).
4	C1 and D1 are both *closed*.	14	C1 changes to closed firstly 12 times and the other 2 times are the reverse orders. The two state changes happen within 1 minute (7 times), 2 minutes (2 times) and 3 minutes (4 times).
5	C2 and D2 are both *closed*.	13	C1 changes to closed firstly 2 times and the other 11 times are the reverse orders. The two state changes happen within 1 minute (8 times), 2 minutes (1 times) and 3 minutes (4 times).

TABLE V
EXPERIMENTAL RESULT OF MINING EVENT-STATE CONSTRAINTS.

No.	Constraint	Occurrence	Description
1	(Y0 $\geq$ 25) $->$ (open B0)	13	When the temperature in the living room exceeds 25 degree, open the window.
2	(Y0 $<$ 25) $->$ (close B0)	13	When the temperature in the living room is less than 25 degree, close the window.
3	(Y1 $\geq$ 20) $->$ (open B1)	9	When the temperature in bedroom 1 exceeds 20 degree, open the window.
4	(Y1 $<$ 20) $->$ (close B1)	9	When the temperature in bedroom 1 is less than 20 degree, close the window.
5	(Y2 $\geq$ 20) $->$ (open B2)	3	When the temperature in bedroom 2 exceeds 20 degree, open the window.
6	(Y2 $<$ 20) $->$ (close B2)	3	When the temperature in bedroom 2 is less than 20 degree, close the window.
7	(Z0 $\leq$ 10%$\vee$ Z0 $\geq$ 25) $->$ (turn on F0)	7	When the humidity in the living room is not more than 10% or not less than 30%, open the AC.
8	(10%$<$Y0$<$30%) $->$ (turn off F0)	7	When the humidity in the living room is between 10% and 30%, turn off the AC.
9	(X1 is true) $->$ (turn on E1)	2	When there is people in bedroom 1, turn on the lamp.
10	(X1 is false) $->$ (turn off E1)	2	When there is not people in bedroom 1, turn off the lamp.
11	(X2 is true) $->$ (turn on E2)	14	When there is people in bedroom 2, turn on the lamp.
12	(X2 is false) $->$ (turn off E2)	14	When there is not people in bedroom 2, turn off the lamp.
13	(X4 is true) $->$ (turn on K4)	14	When there is people in the bathroom, turn on the water heater.
14	(X4 is false) $->$ (turn off K4)	14	When there is not people in the bathroom, turn off the water heater.

TABLE VI
EVALUATION OF MINING EVENT-EVENT CONSTRAINTS.

Support	True Positive	False Negative	Recall
11	4	0	100%
12	3	1	75%
13	3	1	75%
14	2	2	50%
15	0	4	0%

3) Evaluation of Mining Event-State Constraints: This experiment first finds those single events happening frequently and then extracts the invariable conditions triggering them. Unlike mining event-event constraints, the support (i.e., h_o) in this experiment varies in a larger range, i.e., from 2 to 12. We found that although some events occurring not very frequently, their happening conditions are the same, as such they are still taken as constraints.

The evaluation results are shown in Table VIII. Overall, the number of obtained constraints decrease along with the increase of support. In particular, the same performance (i.e., recall) is achieved by setting h_o with some different values, e.g., 4, 5 and 6.

TABLE VII
EVALUATION OF MINING STATE-STATE CONSTRAINTS.

Time interval	Support	TP	FN	Recall
180	12	5	0	100%
180	13	5	0	100%
180	14	3	2	60%
180	15	2	3	40%
120	9	5	0	100%
120	10	3	2	60%
120	11	2	3	40%
120	12	0	5	0%
60	3	5	0	100%
60	4	5	0	100%
60	5	3	2	60%
60	6	3	2	60%
60	7	2	3	40%
60	8	1	4	20%
60	9	0	5	0%

TABLE VIII
EVALUATION OF MINING EVENT-STATE CONSTRAINTS.

Support	True Positive	False Negative	Recall
2	14	0	100%
3	12	2	85.7%
4	10	4	71.4%
5	10	4	71.4%
6	10	4	71.4%
7	6	8	42.9%
8	6	8	42.9%
9	4	10	28.6%
10	4	10	28.6%
11	2	12	14.3%
12	0	14	0%

C. Discussion

(1) The mining result heavily depends on proper settings of parameters, which requires interaction with end-users. Nevertheless, comparing with declaring constraints themselves, adjusting the thresholds is easier. Furthermore, the more effective method for automatically tuning parameters will be explored.

(2) In our experiment, the activities of the two people are not distinguished. In consequence, there might be contradictory constraints in the real world. The existing researches on identifying activities of multiple residents and resolving rule contradictions can be integrated into future work.

(3) It seems that the different kinds of constraints are overlapping. For example, two events occur (e.g., *close a curtain* and *close a door*) in an event sequence and the states led by them are identified as a state-state pattern. This situation is thought of reasonable because the two types of constraints focus on different aspects of IoT event traces. The former specifies the temporal relations among events while the latter specifies the expected states after events happen.

(4) The proposed approach cannot find the constraints specifying *what should not do (or happen)*, because event traces do not imply such kind of constraints.

VII. RELATED WORK

This work mostly relates to researches on TAP rules, activity/event pattern mining and IoT service orchestration.

TAP is the main way of designing orchestration rules for IoT devices. AutoTap [3] translates properties specified by novice users to LTL specifications and automatically synthesizes TAP rules that satisfy the specifications. AutoTap formally models a property and the corresponding devices as a Büchi Automaton and synthesizes property-compliant TAP rules by removing edges in the Automaton leading to invalid states. Multiple classes of TAP bugs [5] and security issues [12] are identified, and a lot of work is devoted to detecting and fixing TAP bugs and vulnerabilities by leveraging various technical methods, e.g., formal verification [13], [14], information-flow control [15], [16], privilege isolation and priority ranking [17], and dynamic instrumentation [18]. Specifically, MenShen [13] automatically checks smart home systems based on their Linear Hybrid Automata. It presents a quantifier elimination-based method to analyze the counterexample and synthesize fix suggestion. BuildingRules [17] automatically detects conflicts that occur among specified policies and resolves them by using Z3 SMT solver [19]. Besides, empirical studies are performed [20], [21] by analyzing and characterizing a large number of TAP rules. The existing studies assume that those expected properties, rules and constraints are pre-defined. However, designing the rules is not trivial, and our work is complementary to these existing studies.

Researches on IoT service orchestrations are mainly based on activity (or event) pattern mining. PCMiner [8] discovers periodic composite IoT services from event sequences, by employing significance and proximity strategies to make filtering. Besides, it estimates time intervals and locations for the obtained composite IoT services. CoPMiner [22] mines the temporal relations among appliances in smart homes by transforming interval-based event sequences into endpoint based sequences and discovering frequent patterns from the endpoint sequences. Urbieta et al [23] presented an adaptive service composition framework based on an abstract service model that represents IoT services and user tasks in terms of their signatures, specifications and conversations. The existing work focuses on constructing a completed workflow constituted of IoT services, by mining device usage patterns [8], [22] or reasoning conversations among IoT services [23]. In contrast, our work focuses on mining fine-grained temporal constraints among events, contexts and device states, which specifies expected properties rather than working processes in which each event happens at a fixed time point (or time interval).

Researches on event pattern mining usually leverage different algorithms and heuristic rules. PBuilder [24] mines sequential patterns regarding the electrical usage of smart home appliances. Zamil et al [9] proposed a model for temporal events based on an ordered decision tree and spatio-temporal characters. They leveraged LTL as the query language for users to obtain temporal patterns, by assuming events happen in a specific time at specific locations. There is some work devoted

to mining general LTL specifications [25], [26] from system traces, and the LTL specifications are further used for system model checking and runtime verifications. Texada [25] takes a pre-defined LTL property type template and a log of traces as input and outputs a set of property instantiations compliant to the system executions, based on two mining algorithms, i.e., a linear miner and a map miner. Neider and Gavran [26] proposed two novel algorithms for learning LTL formulas from event traces based on SAT solving and learning decision trees, respectively. Overall, the LTL formulas generation approaches can be leveraged to transforming the temporal constraints obtained by our approach into formal specifications, for model checking and runtime verifications.

VIII. Conclusion

Mining temporal constraints of smart home IoT devices is helpful for end-users to develop proper TAP rules and locate service behavior deviations. This paper systematically proposes an approach to mining multiple kinds of constraints from event traces. The approach constructs a model for smart homes that augments the event trace with devices states and physical environmental contexts. After that, the approach proposes three types of constraints among events and states and designs the corresponding mining methods for them. Finally, a set of preliminary experiments are performed for evaluating the approach.

Besides the work mentioned in Section VI-C, the future work also includes (1) generating TAP rules based on obtained constraints; (2) integrating this approach into the smart home systems, i.e., Home Assistant; (3) constructing more complex scenarios with more IoT devices, to evaluate the approach thoroughly.

References

[1] P. Smiari, S. Bibi, and D. Feitosa, "Examining the reusability of smart home applications: A case study on eclipse smart home," in *International Conference on Software and Systems Reuse*. Springer, 2019, pp. 232–247.

[2] S. Manandhar, K. Moran, K. Kafle, R. Tang, D. Poshyvanyk, and A. Nadkarni, "Helion: Enabling a natural perspective of home automation," *arXiv preprint arXiv:1907.00124*, 2019.

[3] L. Zhang, W. He, J. Martinez, N. Brackenbury, S. Lu, and B. Ur, "Autotap: synthesizing and repairing trigger-action programs using ltl properties," in *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 2019, pp. 281–291.

[4] L. Smirek, G. Zimmermann, and M. Beigl, "Just a smart home or your smart home–a framework for personalized user interfaces based on eclipse smart home and universal remote console," *Procedia Computer Science*, vol. 98, pp. 107–116, 2016.

[5] W. Brackenbury, A. Deora, J. Ritchey, J. Vallee, W. He, G. Wang, M. L. Littman, and B. Ur, "How users interpret bugs in trigger-action programming," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 552.

[6] C. Liu, X. Chen, R. Shin, M. Chen, and D. Song, "Latent attention for if-then program synthesis," in *Advances in Neural Information Processing Systems*, 2016, pp. 4574–4582.

[7] Y.-C. Chen, W.-C. Peng, and S.-Y. Lee, "Mining temporal patterns in time interval-based data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3318–3331, 2015.

[8] B. Huang, A. Bouguettaya, and A. G. Neiat, "Convenience-based periodic composition of iot services," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 660–678.

[9] M. G. Al Zamil, S. M. Samarah, M. Rawashdeh, and M. A. Hossain, "An odt-based abstraction for mining closed sequential temporal patterns in iot-cloud smart homes," *Cluster Computing*, vol. 20, no. 2, pp. 1815–1829, 2017.

[10] J.-b. Woo and Y.-k. Lim, "User experience in do-it-yourself-style smart homes," in *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. ACM, 2015, pp. 779–790.

[11] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *proceedings of the 17th international conference on data engineering*. Citeseer, 2001, pp. 215–224.

[12] Z. B. Celik, E. Fernandes, E. Pauley, G. Tan, and P. McDaniel, "Program analysis of commodity iot applications for security and privacy: Challenges and opportunities," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–30, 2019.

[13] L. Bu, W. Xiong, C.-J. M. Liang, S. Han, D. Zhang, S. Lin, and X. Li, "Systematically ensuring the confidence of real-time home automation iot systems," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 3, p. 22, 2018.

[14] C.-J. M. Liang, L. Bu, Z. Li, J. Zhang, S. Han, B. F. Karlsson, D. Zhang, and F. Zhao, "Systematically debugging iot control system correctness for building automation," in *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM, 2016, pp. 133–142.

[15] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia, "Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1501–1510.

[16] I. Bastys, M. Balliu, and A. Sabelfeld, "If this then what?: Controlling flows in iot apps," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1102–1119.

[17] A. A. Nacci, V. Rana, B. Balaji, P. Spoletini, R. Gupta, D. Sciuto, and Y. Agarwal, "Buildingrules: A trigger-action–based system to manage complex commercial buildings," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 2, p. 13, 2018.

[18] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, "Fear and logging in the internet of things," in *Network and Distributed Systems Symposium*, 2018.

[19] L. de Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340.

[20] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman, "Practical trigger-action programming in the smart home," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2014, pp. 803–812.

[21] B. Ur, M. Pak Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze, and M. L. Littman, "Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 3227–3231.

[22] Y.-C. Chen, C.-C. Chen, W.-C. Peng, and W.-C. Lee, "Mining correlation patterns among appliances in smart home environment," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014, pp. 222–233.

[23] A. Urbieta, A. González-Beltrán, S. B. Mokhtar, M. A. Hossain, and L. Capra, "Adaptive and context-aware service composition for iot-based smart cities," *Future Generation Computer Systems*, vol. 76, pp. 262–274, 2017.

[24] M. Hassani, C. Beecks, D. Töws, and T. Seidl, "Mining sequential patterns of event streams in a smart home application." in *LWA*, 2015, pp. 159–170.

[25] C. Lemieux, D. Park, and I. Beschastnikh, "General ltl specification mining (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 81–92.

[26] D. Neider and I. Gavran, "Learning linear temporal properties," in *2018 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2018, pp. 1–10.