This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

Fog-enabled Event Processing Based on IoT Resource Models

Yang Zhang, and Victor S. Sheng, *Senior Member,* IEEE

Abstract—Complex Event Processing (CEP) systems extract interest situations from event flows based on event detection patterns. However, local event processing for distributed Internet of Things (IoT) has not been discussed yet. Besides, it is complex or impossible to discover such patterns in some applications of IoT. In this article, we design a complex event service to process event flows based on IoT resource models, which does not depend on existing patterns, and deals with both discrete events and continuous variables. To improve the CEP performance, local IoT resources are used for local event processing, and a lazy exchange method is designed to realize the collaborated event processing between network edges and a data center. Our evaluation shows that our solution is feasible and effective.

Index Terms—CEP, Satisfiability Modulo Theories, IoT Resource, Fog Computing.

1 Introduction

Events often represent the attribute changes of physical objects. A complex event is the interesting phenomena that are found out from many flows of events, and then published as a new event. Existing Complex Event Processing (CEP) systems try to process event flows based on event detection patterns. Users should define these patterns via specifying relations between interesting phenomena and observed events. It is sometimes complex or impossible to discover such patterns in some applications of Internet of Things (IoT).

For example, *Esper* [40] is a powerful open-source CEP system that helps efficiently compose events from event flows. But it requires users to carefully and accurately define event detection rules for complex events, such as

"***EVENT (****or* ***SELECT * FROM)*** *SHELF-READING* ***WHERE*** *category = 'toy' ∧ manufacturer_id = '2'* ***WITHIN*** *5 minutes*" ,

where the concerned event name is "*SHELF-READING*" in all event flows; events are filtered by two predicates applied to their attributes and one time constraint: the first one requires the attribute category to be 'toy' and the second requires the attribute value of the manufacturer identifier to be '2'; and the interest events should occur in a time window of "*5 minutes*".

In the above example, besides users defining this filter rule, the filter predicates are defined on the occurring events. If the events are normal but indirectly induce other interests from underlying knowledge, *Esper* [40] won't produce a complex event at all. For instance, an event of selling a kind of products may occur without satisfying the above rule. But this sale may induce new market predictions or an insufficient inventory for its warehouse. This could generate a complex event of purchasing raw materials according to the background knowledge of a company rather than the direct predicates over selling events. That is to say, *Esper* does not focus on event processing over underlying knowledge.

In IoT scenarios, events are generated by monitoring physical entities in the physical world. These physical entities are often semantically modeled for interpreting raw data received from industrial buses, such as modeling them by Semantic Sensor Network (SSN) [41]. These semantic models could be explored to find out some complex events hidden behind occurring events.

Teymourian et al. [34] tried to use such semantic models of entities to find complex events in real-world scenarios, where they discussed how to modularly select knowledge as the CEP basis. Our work also follows this line. In our CEP service, however, the monitored physical systems and sensors are modeled as IoT resource models [18], and unknown attribute values of IoT resources are approximated.

The introduction of semantic IoT resource models drastically depresses the event processing performance for additional knowledge checking. **The key issue** in our work is to derive complex events through divide-and-conquer from atomic events and IoT resources. That is to say, the underlying IoT resources and incoming events should be partitioned, multiple event deduction units work concurrently, and the results from different units should be synthesized by a distributed way.

Such distributed concurrent event processing methods are widely needed in IoT scenarios. Physical entities are often deployed at sites far from the data center, such that a real-time processing appeals a fog computing paradigm [57], [58] where some computation and storage are placed at the edges of a network, and the data center collaborates with the fog edges to realize low latency for applications. As the work of [65], which evaluated event severity on fog edges to decide how to collaborate with the data center to process health-related events, our work also adopts such collaboration paradigms for time-sensitive events, *called fog-enabled*, but we focus on semantic knowledge. The work of [68] explored the dependency among events to distribute events between fog edges and the cloud for real-time event processing, the work of [66] addressed the issue of joint scheduling tasks on both fog edges and the data center, the work of [67] focused on jointly executing data analysis tasks, but event processing over semantic resources is not discussed in these works.

For classic concurrent event processing, Hirzel [59] proposed a pattern-partitioned concurrent event processing scheme, where input event streams are partitioned based on CEP pattern partitions, different partitioned streams are isolated each other, and combining partial matches between partitions is avoided. In fact, the partition-independence requirement is too strict for practical event rules. Balkesen *et al.* [60] proposed a state-based concurrent event processing scheme, where complex event models represented by states and state transitions are

- *Y. Zhang is with State Key laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, No. 10, Xitucheng Road, Beijing, China, 100875. E-mail: YangZhang@ bupt.edu.cn.*
- *V. S. Sheng is with Department of Computer Science, University of Central Arkansas, 201 Donaghey Ave., Conway 72035, USA. E-mail: ssheng@uca.edu.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

decomposed by the states, and one processing unit is in charge of some states checking. The results from different processing units should be synthesized. These works did not address **the difficulty** of partitioning semantic knowledge.

In order to deal with the semantic knowledge, we introduce a new concept of interest goals, e.g., the violation of the system safety. Given IoT resource models as the background knowledge, the complex event processing still cannot be carried out because the processing should be directed, meaningful, or related to user concerns, rather than arbitrary. That is, interest goals should be defined.

An interest goal should be specified according to user concerns. Compared with event composition rules, it is not directly defined on the attributes of occurring events. It is often declared as a logical consequence of the knowledge consisting of IoT resources and events. A goal may be derived from different events and event composition rules, and an occurring event and an event relation (a rule) may lead to multiple goals. In the above example, interest goals such as "the inventory of materials being low for future machining" are declared as a deduction consequence without being able to be directly defined on the product selling events.

In this paper, we will design a fog-enabled event processing service to address the above issues. Our contributions are listed as follows.

(1) There exist lots of IoT resources and events in an IoT application, and then iterating through all of them to check the satisfaction of interest goals is inefficient. The divide-and-conquer principle is adopted based on resource locality, *i.e., resource working scopes as demarcation lines*, and a resource selection algorithm is designed to partition a resource set as well as representing IoT resources by a distributed way, *i.e., making our CEP fog-enabled*.

(2) A combination theorem is then proposed to test the satisfiability of interest goals on two resource partitions, which is proved to be complete and sound. An algorithm of concurrent event deductions is designed based on the theorem, and lazy exchange is used to reduce the communication between a data center and the fog edges, where the knowledge of resources and events is partitioned into knowledge sub-sets, with possible search space partitioning. Common information exchange is delayed to a point of a model found or unsatisfiability checked.

(3) The continuous dynamics of an IoT resource is often very complex with no explicit expressions in mutable environments. If a resource attribute about this dynamics has unknown value, *called a continuous variable*, its accurate evaluation and related complex event processing will be hard. We should approximate these variables. However, an approximation that deviates from the true value of a variable always introduces an error rate into CEP. Therefore, we introduce an error rate evaluation scheme into Satisfiability Modulo Theories (SMT) solvers [13], [26], [17], [14] to process the error rates.

The remainder of this paper is structured as follows. Section 2 presents preliminaries. In Section 3, the fog-enabled event processing service is given. In Section 4, we discuss how to select IoT resources from a knowledge set. In Section 5, we discuss the approximation of continuous variables. Section 6 shows how to make concurrent event deduction. Experimental results are presented in Section 7. Section 8 gives the related work. Finally, conclusions are made in Section 9.

2 PRELIMINARIES

Our event processing solution over semantic IoT resources involves not only functions (e.g. power function) but also predicates on the former, and we sketchily describe them for being self-contained. A set of functions forms a term signature Σ, and constants are considered as special functions without arity. Given a set V of variables and the signature Σ, a term is a variable in V, a constant in Σ, or a function in Σ with terms as its input, denoting the term set by $T(\Sigma,V)$. Like [18], we impose predicates on terms, and use first-order logic (FOL) to describe the knowledge about events and IoT resources.

First-order logic is defined on a term set $T(\Sigma,V)$, a predicate set Pre, propositional connectives $\neg,\wedge,\vee,\rightarrow$, quantifiers $\exists,\forall$, the equality symbol $=$, and brackets and other punctuations [13], [14], [27]. The term signature Σ and symbols in Pre form an *FOL* signature Σ'. An atom set consists of logical constants *False* and *True,* predicates with the terms as input, and $s=t$ with s,t being terms. A literal means an atom or its negation. A formula is a literal, or a composition with multiple formulas being composed by the propositional connectives, the quantifiers, and the punctuations. Then, a Σ' sentence is a formula without including free variables (a variable without being quantified by quantifiers in a formula is free). A Σ' interpretation over the variable set is a map over a domain, where each variable and constant is respectively mapped into an element in the domain, functions are mapped into the domain's functions, and predicates are mapped into the domain's relations. A formula is considered to be satisfiable if it is *True* under the Σ' interpretation; otherwise, it is unsatisfiable. A theory is a set of sentences over the *FOL* signature Σ'. Determining whether a formula is satisfiable over some theories is a problem of satisfiability modulo theories (SMT) [13], [15], [16]. We assume that readers are familiar with first-order logic, so we do not describe them in detail.

Non-linear functions and and their constraints over real theories can be evaluated based on *Interval Analysis* [47], [48], which uses a pair of computer numbers to represent an interval of real numbers, and adopts a "branch-and-prune" method to find solutions of real constraints. For these constraints, we use the decision procedure of Branch-and-Prune for interval analysis in [49] to solve them, defined in **Definition 1** (the two functions of branch and prune can be referred to [49] for more details). We sketchily present **Definition 1** to show that the solution to the constraints over real theories can be found out by the divide-and-conquer way.

Definition 1. Branch-and-Prune Procedure [49].

For a quantifier-free formula composed of non-linear real constraint atoms $\{expCon_i(x_1,\cdots,x_n)|i=1,\cdots,m\}$, *and initial interval* $B^0=(I_1^0,\cdots,I_n^0)$, *a stack* $Stack$ *is adopted as a key data structure to store intermediary intervals used in the iteration of solving the constraints. There are two functions [49], [47] :* $Prune(B,expCon_i)$ *and* $Branch(B,x_j)$. *The former is to prune some sub-intervals in* B *for finding solutions in optimistic sub-spaces, and the latter is to split the current interval into two sub-intervals for finding solutions in narrow intervals. The procedure to find out whether there are solutions for these constraints is as follows:*

(1) $Stack.push(B^0)$;
(2) **while** $Stack\neq\emptyset$;
(3) $B\leftarrow Stack.pop()$;
(4) **for** $i=1,\ i\le m,\ i++$ **do**
(5) $Prune(B, expCon_i)$; //delete bad sub-spaces in B
(6) **if** $B\neq\emptyset$ **then**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

(7) **if** $\exists j, 1 \le j \le n, \ |\mathrm{I}_j| \ge threshold$ **then** //interval being large
(8) $\{B_1, B_2\} \leftarrow Branch(B, x_j)$;
$Stack.push(\{B_1, B_2\})$;
(9) **else return** B ;
(10) **return** $\emptyset$.

3 Fog-enabled Event Processing Service

The framework of our fog-enabled event processing service, *called CEP service*, is illustrated in Fig. 1. The CEP service reads event flows in a publish/subscribe middleware, and composes them by the IoT resource knowledge. At the fog edges, some physical systems and devices are deployed, their corresponding IoT resources are locally stored, and the complex function's implicit expressions related to these local resources are downloaded from the data center. When subscribed events relevant to these local IoT resources arrive, the local CEP service starts. If some other resources are involved, the local CEP service interacts with the CEP services at the data center to deduce complex events.

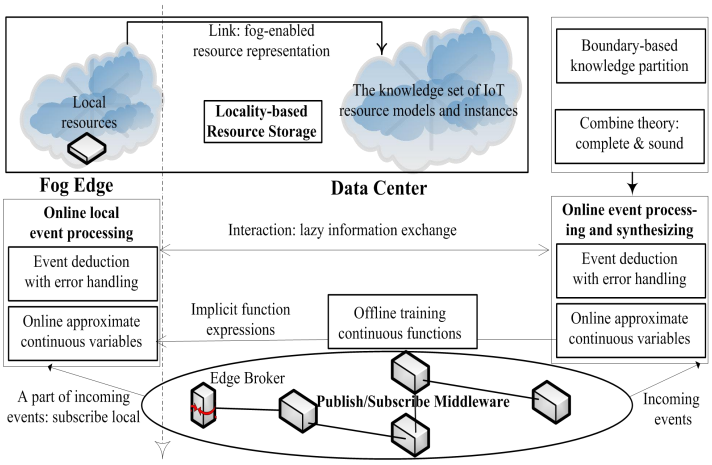


Figure 1. Fog-enabled event processing service

Fig. 2 further illustrates the CEP service itself. It consists of four procedures. The second procedure is about knowledge partition (discussed in Section 4), the concurrent event deduction (the third procedure) is carried out over the partitions (discussed in Section 6), and the error for approximating variables is handled in the last procedure (discussed in Section 5). The first procedure in Fig. 2 is a *CEP Maintaining Procedure*, which receives events from the publish/subscribe middleware.

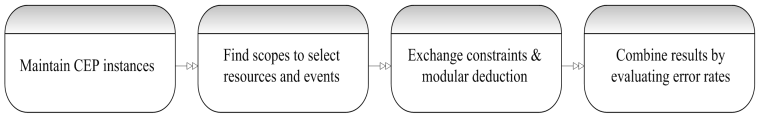


Figure 2. Event processing in CEP services

In this paper, there is an important observation that IoT resources are characterized by the locality. That is, physical systems and physical sensors have their own working scopes, which can be represented by some geographic boundaries, or fog edge sites. For example, we can select all resources in *Sanjingou Coal Mine* for a CEP service in this mine system.

Just this inherent locality of IoT resources and events makes our CEP service fog-enabled at four perspectives. At the first perspective of localized resource storage, a resource in the whole knowledge is represented by crossing edges in distributed RDF (Resource Description Framework) graphs [46, 50] to circle it, and the fog edges can store the local resources with such crossing edges linking to others. At the second perspective of resource selection, the semantic knowledge of resources is used by their working scopes (*e.g. physical boundaries*), and we design a **static *divide-and-conquer*** method (*called Scope Search*) for selecting possible resources from a resource set, which makes one CEP service instance run on a knowledge partition, *i.e., being able to be deployed on fog edges*. At the third perspective of local event delivery, the CEP service on a fog edge only subscribes local events to the publish/subscribe middleware such that the middleware optimizes event flows with low burdens and high efficiencies. At the last perspective of event deduction, a modular proof theorem is provided as the theoretic basis for the fog edges to be able to process events on their local resources, and a **dynamic *divide-and-conquer*** algorithm (*called Concurrent Deduction*) is realized by lazy information exchange between the center and edges.

Compared with classic cloud computing paradigms, our fog-enabled work adopts cloud-to-edges collaboration paradigms [57], and focuses on the locality from the above four perspectives, *i.e., local CEPs over local events and local IoT resources by minimizing communication with clouds*. The data center can collect the event processing results produced by the fog edges for further synthesization, *i.e., realizing complex event processing along the cloud-to-things continuum.*

4 Selecting IoT Resources for CEP

4.1 IoT Resources and Events

We modelled each IoT resource by an object model and a lifecycle model [18], [41]. The object model describes the attributes and the attribute relations of a resource. The lifecycle model describes its progressing ways, state transitions, and continuous resource dynamics.

Established resource object models are stored using a "Resource Description Framework" (*RDF: a collection of triples <subject, property, object>*) triple format. An RDF graph is not only intuitionistic, but also expressive [50] ([50] is a specification mapping OWL2 ontologies to RDF graphs). Within the resources object models, all concepts are connected together without an explicit borderline to delimit a resource from the stored knowledge graph. In principle, each physical entity is a specification unit, *including its attributes and attribute relations.* The graph partitioning method for a distributed RDF graph in [46] is therefore used to circle a resource in the whole knowledge, and to clarify the relations between two resources by crossing edges (see Fig. 6). We extend the RDF graph by attaching a resource lifecycle model to the entity specification unit without changing graph structures, illustrated in **Definition 2**.

Definition 2. IoT Resource. *An IoT resource has an RDF graph* G *and a lifecycle description* T : $IoTR ::= (G, T)$.

The RDF graph G *is specified by* $(V \cup V^e, E \cup E^c, \Sigma)$ *such that*

1. $V \cup V^e$ *is considered as a set of vertexes corresponding to subjects and objects in RDF data;* $E \cup E^c$ *is considered as a multiset of directed edges corresponding to triples in RDF data;* $E \subseteq V \times V$; Σ *is considered as a set of edge labels corresponding to properties in RDF data;*

2. E^c *is considered as a set of crossing edges between* G *and other IoT resources* $G_1, \cdots, G_n$, *i.e.,*

$$E^c = \bigcup_{i=1}^{n} \{\overrightarrow{vv'} \mid v \in V^e \wedge v' \in G_i.V^e \wedge (\overrightarrow{vv'} \in V^e \times G_i.V^e)\} ;$$

3. A vertex $v \in V^e$ *only when a vertex* v' *lies in other resources and* v *is an endpoint of some crossing edges, i.e.,*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

$V^e = \bigcup_{i=1}^{n} \{ v \mid \overrightarrow{vv'} \in E^c \wedge v' \in G_i.V^e \}$;

4. Vertexes in V^e *are denoted as external vertexes and all vertexes in* V *are denoted as internal vertexes;*

A lifecycle $T ::= (State, Event, Const, Fun, Mapping, \delta)$ *is a graph of states and state transitions, and defined as follows:*

1. State is a set of states: $s_0, \cdots, s_m$ *, and* s_0 *is initial;*

2. Event is a set of events $\{e_i\}$ *for state transitions;*

3. Const is a set of constraints over resource attributes $\{form_i\}$ *such as* $temperature > 80^\circ C$ *;*

4. Fun is a set of continuous functions representing running attributes in resource states $\{attr_i = f_i(x_1, \cdots, x_k)\}$ *;*

5. Mapping defines states by attributes and constraints: $Mapping : State \rightarrow Const \times Fun$ *;*

6. δ *is a state transition mapping:* $e(e_i) \wedge s_j \rightarrow s_j', s_j' \rightarrow \neg s_j$.

Compared with the distributed RDF graph [46], the crossing edges in **Definition 2** are unidirectional from one resource itself to others, to specify the resource via other knowledge, which reflects the resource stored at local edges centering over the data center for finding out other relevant IoT resources.

For example, there are two resources in a coal mine system: methane gas and mine ventilation illustrated in Fig. 3. The methane gas in the information world represents poisonous gases in the coal mine system, including its generation, diffusion, and toxin level. The mine ventilation here represents the ventilation system in the coal mine system, involving wind speed, wind quantity, and ventilation distance.

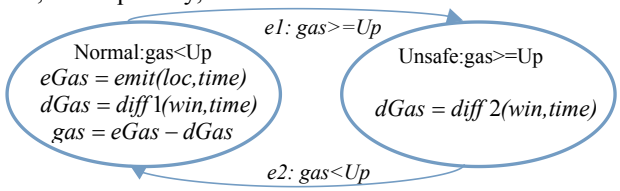


Figure 3a. Methane Gas Resource

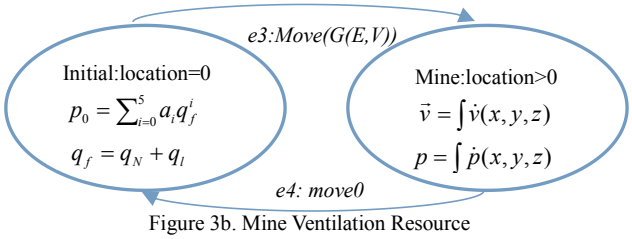


Figure 3b. Mine Ventilation Resource

Object models are specified by RDF graphs as in Definition 2, and its underlying formal language is Description Logic (DL) [20], [24]. Description Logic is a part of first-order logic (FOL). Fig. 4 and Fig. 5 illustrate an object model of two resources, which is described by DL.

$$CoalGas \equiv Gas \cap \exists hasGenerationSpeed.GasGenerationSpeed \cap \exists hasDiffusionSpeed.DiffusionSpeed \cap \exists hasLocation.Location \cap \exists sourceFromTunnel.Tunnel$$

Figure 4. Part Knowledge of the Methane Gas Object

$$CoalVentilation \equiv Ventilation \cap \exists hasWindSpeed.InitialSpeed \cap \exists hasWindQuantity.InitialWindQuantity \cap \exists hasWindSpeed.LoactionSpeed \cap \exists hasWindQuantity.LocationQuantity \cap \exists relatedToTunnel.Tunnel$$

Figure 5. Part Knowledge of the Mine Ventilation Object

From Fig. 4, we know that a methane gas has several attributes, such as methane gas generation speed, methane gas diffusion speed, metered location, and source tunnel. For instance, *Tunnel* in $\exists sourceFromTunnel.Tunnel$ is a concept, the prefix *sourceFromTunnel* before *Tunnel* is a role, and $\exists sourceFromTunnel.Tunnel$ means that *CoalGas* comes from *Tunnel* (*i.e., a coal bed in the tunnel generating gas*). From Fig. 5, we know that the wind speed in a location is related to the model of a coal tunnel. These concepts are connected each other, and form partitioned RDF graphs, illustrated in Fig. 6, where <GoalGas, Tunnel> is a crossing edge.

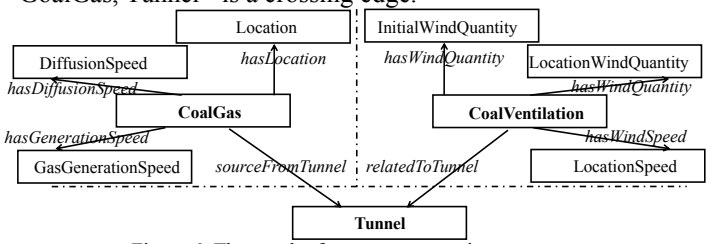


Figure 6. The graph of concept connection

In the lifecycle models of coal resources shown in Fig. 3, each state of a resource has a few continuous mathematical functions to represent the dynamics of the resource in this state. A state transition occurs when these functions exceed corresponding limits, and then the dynamics of this resource can be expressed by the functions in another state. Fig. 3a shows that the normal state of the methane gas resource has two complex continuous functions. One is $eGas = emit(loc, time)$, expressing the generation speed of methane gas produced by the coal bed; and the other is $dGas = diff1(win, time)$, expressing the diffusion speed of methane gas. These functions may not be evaluated directly according to their inputs, because they probably don't have explicit expressions in complex environments. In the unsafe state of the methane gas resource, there is a different diffusion function. Fig. 3b shows that the initial state of the mine ventilation resource has two continuous functions, *i.e., the wind pressure* p_0 *and the wind quantity* q_f , which may have specific expressions as follows:

$$p_0 = \sum_{i=0}^{5} a_i q_f^i \quad \text{and} \quad q_f = q_N + q_l$$

where q_N is an effective wind quantity and q_l is a leaked wind quantity. The second state of the mine ventilation also has two continuous functions about the wind speed v and the wind pressure p , which may be calculated based on the initial state and a tunnel topology $G(E,V)$ of the coal mine system.

These two resource models have a mathematical link. That is, the methane gas diffusion speed is affected by the ventilation speed and quantity. The diffusion function $dGas = diff1(win, time)$ in the methane gas resource has an input parameter *win (wind)*, which is defined by the mine ventilation resource. When the initial wind quantity drops a little, *represented by the mine ventilation resource*, the level of methane at one location of a coal mine system, *represented by the methane gas resource*, may rise drastically, which means that a dangerous event happens. That is, changing attribute values of one resource leads to the varying of attribute values of other resources. The variation computed by the mathematical link is called **constraints** from one resource to the others.

Combining the object model and the lifecycle model together, an IoT resource can be specified in FOL as a resource knowledge unit, which could be understood as a set of sets as follows.

--A set of attributes defined by the concept and role formulas in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

DL such as $hasDiffusionSpeed(CoalGas, DiffusionSoeed)$*, which is also represented by a RDF graph.*

--A set of states over resource attributes such as $gas < Up$ *in Fig. 3a, where* gas *is a resource attribute variable to describe the dynamic behavior involving continuous functions, and* $gas < Up$ *is a formula to define a state. That is, the lifecycle model of a resource is described in FOL (extending DL with functions). One state is defined by attribute formulas with imposing functions (e.g. <) on resource attributes.*

--A set of state transitions: $s \in States \wedge e(e_i) \rightarrow s' \in States$*, where states* s *and* s' *are defined by above attribute formulas.* $e(e_i)$ *is a formula of event occurrences, representing that this transition produces an event* e_i *or is driven by* e_i*, such as* e_1 *in Fig. 3.*

--Running constraints in each state: in each state, the running behavior of a resource is represented by continuous dynamics. These dynamics functions are included in a state's attribute formula, e.g.,

$$gas = eGas - dGas = emit(loc, time) - diff1(win, time) < Up.$$

When some attributes exceed their thresholds by an actuation event or self-excitation, the state transition occurs and a new state including new continuous functions is running.

Events represent the changes of resource attributes or state transitions. Our event processing services subscribe events to an underlying publish/subscribe middleware, where an event is defined by its name and an XML Schema [45], [18]. When events arrive at the complex event service, we should translate them into knowledge in FOL. The representation of a publish/subscribe event and its occurrence knowledge is denoted as follows.

-Event sort: an event sort $s::=(topic\_name, XML\ Schema) = (topic\_name, Schema = \{element_i = (type_i, attri_i) | i = 1, \cdots, n\})$*, where* $attri_i$ *is restricted in a range such as* $gas \geq 1.0$.

-Individual events: $e_1 : s_1, e_2 : s_2, \cdots, s_1, s_2, \cdots$ *being event sorts.*

-Event predicate: $e(e_i)$ *is introduced to impose on one event* e_i *to mean the occurrence of* e_i *in the system.*

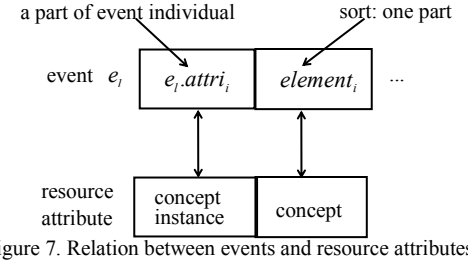


Figure 7. Relation between events and resource attributes

The relation between events and resource attributes is illustrated in Fig. 7. One publish/subscribe event consists of its name and multiple elements, *i.e.*, *multiple resource attributes.* That is, the publish/subscribe event itself can be translated into a composite concept formed by multiple resource attribute concepts (*i.e., a special resource*), and its occurrence is represented by the predicate $e(\cdot)$. It is obvious that complex resource relations, *represented by roles and states,* are not fully included in events. In addition, the resource's state transition labeled by events is not included in events either. Thus, the composite event deduction should be carried out based on resource models and arriving events.

Given a resource knowledge set, we may ask what resource models and resource instances should be used during composing events when an event arrives. An arrival event could be linked to some IoT resources by the information inside its content because events are produced by IoT resources or are used to regulate the IoT resources. But only these resources are not enough for event reasoning. However, how many resources are enough? There is no direct answer.

4.2 Resource Selection Problem Statement

When an event arrives, we should evaluate whether it induces an interest situation. An interest goal can be expressed by events, resource attributes, and continuous functions. Fig. 8 illustrates a goal $SafetyEevent$ derived from several publish/subscribe events $HighLevelofMethane$, $HighTemperature$, and $LowWindQuantity$.

$$e(HighLevelofMethane) \wedge e(HighTempature) \wedge e(LowWindQuantity) \rightarrow e(SafetyEvent)$$

Figure 8. Interest goal

Although interest goals can be used to direct event deduction, *given an interest goal and an arrival event*, whether they are related is not directly known for existing background IoT resources.

The principle is that we can select some resources from a given set of resources to compose events, and the resource selection is carried out based on an observation: ***physical systems and physical devices have their own working scopes***. We therefore define interest goals accompanied by working scopes. An interest goal then consists of an interest and a scope.

In order to specify geographic boundaries, we define non-linear constraints in **Definition 3**, which are suitable for specifying constraints on complex continuous dynamics in IoT resources, and not fully supported by RDF [46]. The work of [51] enhanced the RDF data model using linear constraints to represent spatial and temporal attributes in IoT resources, but it cannot be used to describe complex attribute features.

Definition 3. Non-linear Constraints. *A non-linear expression* $exp(\vec{x})$ *over real numbers* R *is composed of real constants in* R*, real variables* $\vec{x} = < x_1, \cdots, x_n >$*, and real functions (e.g.* $+, -, \times, \div$ *and transcendental functions* $sin(x_1), cos(x_2)$*). A non-linear constraint* $expCon(\vec{x})$ *is defined by* $exp(\vec{x}) \Theta c$, $\Theta \in \{=, \neq, >, \geq, <, \leq\}$, $c \in R$*, called a constraint atom.*

Non-linear constraints can be used to represent not only a spatial boundary but also a temporal boundary (or others) for IoT resources and arrival events. A linear constraint is a special case of non-linear constraints. We then formally introduce interest goals in **Definition 4.** In this paper, we focused on basic goals and their handling as they are foundational, and general goals can be obtained by logically composing the basic ones.

In **Definition 4**, the working scope is defined by non-linear constraints and RDF graph patterns, where the former is used to express geographic boundaries or others, and the latter is used to express some relations on resources' object models such as all selected resources having the same attribute values. An **interest** often reflects users' concerns, and these concerns are conveniently expressed by SWRL (Semantic Web Rule Language) [55] rules (*e.g.* $parent(x, y) \wedge brother(y, z) \rightarrow uncle(x, z)$). The interest is then defined by SWRL-represented event rules and constraints on resource attributes, where the former is expressed on the background knowledge rather than directly on arrival events, and the latter is to define the interest by non-linear constraints, *i.e., extending classic event rules* [40] *into background IoT resource knowledge.*

Definition 4. Interest Goal. *A basic interest goal is specified by* $IQ ::= (Interest, Scope)$ *such that*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

1. A basic interest $Interest$ *consists of an event rule* $evRel$ *and non-linear constraints* $ConSet$.

(1) $evRel$ *is defined by SWRL rules on a set of resources and events: antecedent* $\rightarrow$ *consequent.*

(2) $ConSet$ *is a Boolean combination of constraint atoms, i.e., a FOL formula consisting of constraint atoms.*

2. A basic scope consists of an RDF query Q *and non-linear constraints* $ConScope$.

(1)An RDF query is specified by $Q=(V^Q, E^Q, \Sigma^Q)$ *such that*

a. V^Q *is a considered as set of vertexes in* $V \cup V_{var}$, *where* V *is all vertexes corresponding to subjects or objects, and* V_{var} *is a set of vertex variables;*

b. $E^Q \subseteq V^Q \times V^Q$ *is a multiset of edges in* Q;

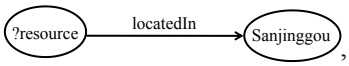
c. Each edge in E^Q *either has an edge label corresponding to properties in* Σ^Q *or the edge is a variable.*

(2) $ConScope$ *is a Boolean combination of constraint atoms.*

For example, the interest goal of *SafetyEvent*, *a complex event*, can be further defined as follows:

1. For the interest, one rule $evRel$ *for the complex event is* $e(HighLevelofMethane) \rightarrow e(SafetyEvent)$, *and one constraint* $ConSet$ *is* $\exists worker, altitude(worker) < 0$.

2. For the basic scope, the RDF query is



and the constraint of geographic boundaries is $ConScope ::= \forall longitude, \forall altitude, (113^\circ 01' \le longitude \le 113^\circ 13') \wedge (36^\circ 30' \le altitude \le 36^\circ 39')$.

The interest constraint $\exists worker, altitude(worker) < 0$ means that the safety event is generated only when there are some workers working in the coal mine. Such interests are more expressive than event detection rules in *Esper [40]* for being able to define composite events on background IoT resources.

Problem Statement: Resource Selection. *Given a set* $\{IoTR_i\}$ *of IoT resources, a set* $\{e_i\}$ *of events, and an interest scope* $IQ.Scope$, *the resource selection problem is to find out a sub-set of resources from* $\{IoTR_i\}$ *and a sub-set of events from* $\{e_i\}$ *to satisfy* $IQ.Scope$.

4.3 Resource Selection Algorithms

For the problem of resource selection, the interest scope satisfaction with an RDF query, $IQ.Scope.Q$, is defined based on graph match [46], [44], *called resource match*.

Definition 5. Resource Match. *Given an IoT resource* $IoTR$ *and a connected resource query* Q *that has* n *vertexes* $\{v_1, \cdots, v_n\}$, *a graph* M *with* m *vertexes* $\{u_1, \cdots, u_m\}$ $(m \le n)$ *selected from* $IoTR$ *is a match of* $IoTR$ *when a mapping* mpp *from* $\{v_1, \cdots, v_n\}$ *to* $\{u_1, \cdots, u_m\} \cup \{null\}$ *exists with the following conditions:*

1. When v_i *is not a variable,* $mpp(v_i)$ *and* v_i *must have the same URI or literal* $(1 \le i \le n)$, *or* $mpp(v_i) = null$;

2. When v_i *is a variable,* $mpp(v_i) \in \{u_1, \cdots, u_m\} \cup \{null\}$;

3. If an edge $\overrightarrow{v_i v_j} \in Q$, *then (1)* $\overrightarrow{mpp(v_i)mpp(v_j)} \in IoTR$ *with property* p, *and* p *is the same as the property of* $\overrightarrow{v_i v_j}$; *(2)* $\overrightarrow{mpp(v_i)mpp(v_j)} \in IoTR$ *with property* p, *and the property of* $\overrightarrow{v_i v_j}$ *is a variable; (3) there is not an edge* $\overrightarrow{mpp(v_i)mpp(v_j)}$ *in* $IoTR$, *but* $mpp(v_i)$ *and* $mpp(v_j)$ *are both in* $\{IoTR.G.V^e\}$; *(4)* $mpp(v_i) = null$; *(5)* $mpp(v_j) = null$;

4. If $mpp(v_i)$ *is an internal vertex in* $IoTR$ *and there is* $\overrightarrow{v_i v_j}$ *or* $\overrightarrow{v_j v_i}$ *in* Q $(1 \le i \ne j \le n)$, *then* $mpp(v_j) \ne null$ *and* $\overrightarrow{mpp(v_i)mpp(v_j)} \in IoTR$ *or* $\overrightarrow{mpp(v_j)mpp(v_i)} \in IoTR$; *and, if* $\overrightarrow{v_i v_j}$ *or* $\overrightarrow{v_j v_i}$ *has some property* p, *then* $\overrightarrow{mpp(v_i)mpp(v_j)}$ *or* $\overrightarrow{mpp(v_j)mpp(v_i)}$ *has* p;

5. If $mpp(v_i)$ *and* $mpp(v_j)$ *are both internal vertexes in* $IoTR$, *a vertex in the path between* v_i *and* v_j *in* Q *then maps to an internal vertex of* $IoTR$.

For the problem of resource selection, the interest scope satisfaction with non-linear constraints, $IQ.Scope.ConScope$, is defined by checking the constraints in a real number domain. For example, on a resource, there is a constraint that requires the resource location to be $20 \le x \le 80, 30 \le y \le 70$ at time $4 \le t \le 15$. In the resource, the trajectory (its dynamics) is constrained by $40 \le ax^2 + by^2 \le 900$ with the movement speed being $6 \le v \le 60$ and the current location being (cPx, cPy). We then evaluated the satisfiability of constraints:

$\{ 40 \le ax^2 + by^2 \le 900, \quad (x - cPx)^2 + (y - cPy)^2 \le (vt)^2, \quad 4 \le t \le 15, \quad 20 \le x \le 80, \quad 30 \le y \le 70, \quad 6 \le v \le 60 \}$.

For non-linear constraints, we used the decision procedure of Branch-and-Prune for interval analysis in **Definition 1** to solve them. *Scope Search* is designed to solve the resource selection problem in Algorithm 1.

Algorithm 1. Scope Search

Input. *An interest scope* $IQ.Scope$, *and a set of resources* $resSet$
Output. *A set of selected resources* $resScp$
Data. *An intermediary match data structure* M

```
1  foreach resource IoTR ∈ resSet with initial empty M
2     foreach vertex v ∈ IQ.Scope.Q
3        u = mmp(v) ∈ IoTR by condition 1 and 2 in Definition 5
4        Add (v, u) into M
5        if M violates other conditions in Definition 5
6           then delete (v, u) and other violated maps from M
7     if M is not empty then
8        add IoTR into resScp
      // RDF query is ended, then constraints are checked
9  foreach resource IoTR ∈ resScp
10    check IQ.Scope.ConScope on IoTR as in Definition 1
11    if IQ.Scope.ConScope is unsatisfiable then
12       delete IoTR from resScp
13 return resScp
```

The selection of arrival events is similar to the resource selection, because an event can also be viewed as a special resource having a set of attributes from other resources.

Here we have another example (taxi transportation) to show the using of *selecting resources and events.* The position of taxis can be sensed and published as event flows, which are used to derive the level of traffic congestion in a road, *i.e., a composite event.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

When an event of a taxi position arrives, its position attribute is compared with a scope, and the road that the taxi runs along could be a working scope. The taxi resource and the road resource are selected. If the CEP service is started based on an interest "*Traffic Congestion of Jintian Road*", all position events from taxis along *Jintian Road* enter into the same CEP service instance with the scope of *Jintian Road*.

5 Approximating Continuous Dynamics

The functions in IoT resources are often complex without explicit expressions. For example, no accurate equations about air dynamics in a specific coal mine system can be written to compute the ventilation attributes at different locations from its initial parameters and the mine tunnel topology. Thus, some complex events over the *Methane Gas Resource* and *Ventilation Resource* cannot be accurately evaluated.

The structure and parameters of a deep neural network (DNN) [29], [30], [28] can implicitly express higher-order non-linear functions. We do not focus on how to use different structures and parameters of DNNs to optimize the representation of different resource functions, because the optimization is with respect to specific domain problems. Our focus is to keep a unified framework to process discrete events, continuous variables, and errors, when general DNN ideas and algorithms are used.

When we use complex functions learned to approximate some unknown resource attribute values, or to predict some attribute values using known attribute values as inputs, the accuracy of a DNN algorithm is not perfect. As first-order probabilistic logic [31], [32], [33], we attached an error rate to predicates with approximated attributes as input, and then computed the compound error rate for formulas.

The accuracy of computing attributes at_1 and at_2 by approximation is assumed to be ac_1 and ac_2 respectively, and the predicates over at_1 and at_2 are A_1 and A_2 respectively. We respectively attach $1-ac_1$ and $1-ac_2$ to A_1 and A_2 as their error rate. Then, the error rate of a formula about A_1 and A_2, denoting by $ErrP$, is computed below.

(1) $ErrP(A_1) = 1 - ac_1$, $ErrP(A_2) = 1 - ac_2$;

(2) $ErrP(\neg A_1) = 1 - ac_1$, $ErrP(\neg A_2) = 1 - ac_2$;

(3) $$ErrP(A_1 \vee A_2) = \begin{cases} 1-ac_1, if\ e(at_1)=True \wedge e(at_2)=False \\ 1-ac_2, if\ e(at_2)=True \wedge e(at_2)=False \\ min(1-ac_1, 1-ac_2), others \end{cases};$$

(4) $ErrP(A_1 \wedge A_2) = max(1-ac_1, 1-ac_2)$.

Although there are many clauses in the knowledge set of events and IoT resources, only some of them are used to derive a complex event. We can use the trying model [39], [38] in SMT solvers as selected clauses. We can also find out resolution paths in a resolution procedure (or an unsatisfiable core for an unsatisfiable formula from SMT solvers) [26] to compute the error rate. The error rate computation is embedded as a background procedure in SMT solvers.

6 Concurrent Event Deduction

In order to compose events on IoT resource models, the key point is to adopt divide-and-conquer methods to improve the performance of making deduction on the large number of knowledge. Hyvarinen *et al.* [54] proposed a search-space partitioning method to use SMT solvers on many problem instances concurrently. Although their method is very efficient, it cannot be used to partition a knowledge set into small sub-sets. In addition, there is a limitation that, in unsatisfiability cases, the performance is determined by the "worst" instances, *i.e., waiting for all instances to be checked*. In the work of [13], [15], [16], a knowledge set can also be partitioned. But formulas in these methods should be decomposed and purified according to predefined signatures and theories. In IoT applications, however, lots of IoT resource instances share the same signatures and theories such that these methods cannot be used to partition these instances. In addition, the knowledge set of IoT resources and events are decomposed according to interest goals rather than the signatures and theories in our work. The method of partition-based proof in [19] did not require to purify the formulas in advance, while it did not consider underlying theories, *i.e., no processing non-linear constraints*.

In this section, we firstly informally describe our problem and ideas, and then provide a theoretic basis for divide-and-conquer event deduction over two IoT resource partitions. Finally, a fog-enabled concurrent algorithm is actually presented.

As we discussed in Section 4, multiple interest goals will be divided and conquered by partitioning IoT resources and arrival events. Multiple event deduction units for different interest goals can concurrently work on these partitioned knowledge sub-sets. For example, in one deduction unit, an interest goal $e(HighLevelofMethane) \rightarrow e(SafetyEvent), DiffusionSpeed < 2.0$ is defined, and the selected resource of methane gas is in the normal state s_0 with $DiffusionSpeed = 1.8$ and the event of $HighLevelofMethane$ being defined by $gas \geq 1.0$. When a publish/subscribe event e_i arrives with carrying $gas = 1.5$, e_i is recognized as $HighLevelofMethane$, and the methane gas resource has its attribute $DiffusionSpeed < 2.0$, such that the complex event $SafetyEvent$ is detected on the methane gas resource and event e_i.

Furthermore, the scope of one interest goal could be decomposed such as splitting a geographic boundary into multiple sub-boundaries, and then multiple event deduction units work for one goal with different sub-scopes. For one interest goal being detected on multiple deduction units with multiple sub-partitions of IoT resources and events, some information exchanges are needed at two layers: upper SAT (Boolean satisfiability problem) solvers and underlying theory solvers, as follows:

(1) If a proposition is shared among these resource sub-partitions (a predicate is also viewed as a proposition in SAT solvers), its assignment is exchanged. That is, the proposition assignment in one deduction unit should be propagated into other units.

(2) Solutions to non-linear constraints should be exchanged. For example, a constraint $diff1(win,time) < 1.0$ on the function $diff1$ in the methane gas involves the wind speed of the mine ventilation resource. This exchange may happen between an event deduction unit on the methane gas and the unit on the mine ventilation resource, *i.e., the value of wind speed (an interval of real numbers) being exchanged*. We use real theory solvers based on Interval Analysis [47], [48] to solve the constraint and then to exchange the results.

For an interest goal, the resources and events are selected as a knowledge set Λ. We assume Λ is partitioned into two

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

knowledge sub-sets Λ_1, Λ_2 with $\Lambda = \Lambda_1 \cup \Lambda_2$. Each $\Lambda_i\ (i = 1, 2)$ denotes a partition, and $\Gamma(\Lambda_i)$ denotes a language with symbols from Λ_i. Our modular proof is presented below as the theoretic basis for concurrent event deduction on two knowledge partitions.

Definition 6. Modular Proof *ModularP(q, Λ_1, Λ_2)*. *For a knowledge set $\Lambda = \Lambda_1 \cup \Lambda_2$, and an interest goal q, it partitions q into q_1 and q_2 with $q = q_1 \wedge q_2$, $q_1 \in \Gamma(\Lambda_1)$ and $q_2 \in \Gamma(\Lambda_2)$. Modular proof works as follows:*

(1) Do satisfiability testing:

*(**a**) Do satisfiability testing for one part of query q_1 in Λ_1. If tested clauses include common symbols q' (e.g. variables, constants, functions, predicates) that also belong to $\Gamma(\Lambda_2)$, then exchange the interpretation of q' to Λ_2.*

*(**b**) Do satisfiability testing for one part of query q_2 in Λ_2. If some symbols q' included in tested clauses also belongs to $\Gamma(\Lambda_1)$, then exchange the interpretation of q' to Λ_1.*

*(2) If (a) and (b) are both satisfiable, **True** is returned; or **False** is returned.*

The decision procedure in **Definition 1** acts as a theory solver in SMT solvers, which works as a background procedure for **Modular Proof** in **Definition 6.**

Theorem 1. Modular Proof is complete and sound.

Proof.

Completeness.

Assume M be a model (an interpretation) of $(q_1 \wedge \Lambda_1) \cup (q_2 \wedge \Lambda_2)$. We project M to the two knowledge partitions: $M_1 = M^{\Lambda_1 \wedge q_1}$ and $M_2 = M^{\Lambda_2 \wedge q_2}$, where M_1 is the interpretation of $q_1 \wedge \Lambda_1$ in M, and M_2 is the interpretation of $q_2 \wedge \Lambda_2$ in M. It is obvious that:

(1) M_1 satisfies $q_1 \wedge \Lambda_1$;

(2) M_2 satisfies $q_2 \wedge \Lambda_2$;

(3) $M_1^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)} = M_2^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)}$ which means that the common parts of two knowledge partitions will have the same interpretation.

That is, if $(q_1 \wedge \Lambda_1) \cup (q_2 \wedge \Lambda_2)$ is satisfiable, then $q_1 \wedge \Lambda_1$ and $q_2 \wedge \Lambda_2$ are both satisfiable and have the same interpretation for their common parts.

Soundness.

Assume there is a model M_1 for $q_1 \wedge \Lambda_1$, a model M_2 for $q_2 \wedge \Lambda_2$, and $M_1^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)} = M_2^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)}$ which means that the common parts of two knowledge partitions have the same interpretation by exchanging their assignments during satisfiability testing (i.e., Modular Proof).

We then construct a model M for $(q_1 \wedge \Lambda_1) \cup (q_2 \wedge \Lambda_2)$:

(1)Let $M = M_1$, and then extend M;

(2)For variables and constants in $(q_1 \wedge \Lambda_1) \cup (q_2 \wedge \Lambda_2)$,

$$M(x) = \begin{cases} M_1(x), \text{if } x \in \Gamma(\Lambda_1 \wedge q_1) \\ M_2(x), \text{if } x \in \Gamma(\Lambda_2 \wedge q_2) \setminus \Gamma(\Lambda_1 \wedge q_1) \end{cases};$$

(3)For a function $f(a_1, \cdots, a_n)$ with arity n, $a_1, \cdots, a_n$ are interpreted into $b_1, \cdots, b_n$ according to step (2) or according to the iteration of steps (2) and (3); and

$$M(f(a_1, \cdots, a_n)) = \begin{cases} M_1(f(b_1, \cdots, b_n)), \text{ if } f \in \Gamma(\Lambda_1 \wedge q_1) \\ M_2(f(b_1, \cdots, b_n)), \text{ if } f \in \Gamma(\Lambda_2 \wedge q_2) \setminus \Gamma(\Lambda_1 \wedge q_1) \end{cases};$$

(4)For a predicate $p(a_1, \cdots, a_n)$ with arity n, $a_1, \cdots, a_n$ are interpreted into $b_1, \cdots, b_n$ according to step (2) or according to the iteration of steps (2) and (3); and

$$(a_1, \cdots, a_n) \in M(p) \Leftrightarrow \begin{cases} (b_1, \cdots, b_n) \in M_1(p), \text{if } p \in \Gamma(\Lambda_1 \wedge q_1) \\ (b_1, \cdots, b_n) \in M_2(p), \text{if } p \in \Gamma(\Lambda_2 \wedge q_2) \setminus \Gamma(\Lambda_1 \wedge q_1) \end{cases}.$$

The condition of $M_1^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)} = M_2^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)}$ makes the input of functions or predicates meaningful in the construction of step (3) and (4), because, when it belongs to the other partition, the input (constant, variable, or function) is also a shared part with non-conflict interpretations.

We thus construct a model M for $(q_1 \wedge \Lambda_1) \cup (q_2 \wedge \Lambda_2)$ according to the models M_1 for $q_1 \wedge \Lambda_1$ and M_2 for $q_2 \wedge \Lambda_2$.

In summary, if $q_1 \wedge \Lambda_1$ and $q_2 \wedge \Lambda_2$ are both satisfiable, then $(q_1 \wedge \Lambda_1) \cup (q_2 \wedge \Lambda_2)$ is satisfiable, when the common parts of $q_1 \wedge \Lambda_1$ and $q_2 \wedge \Lambda_2$ have their interpretations exchanged to keep $M_1^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)} = M_2^{(\Lambda_1 \wedge q_1) \cap (\Lambda_2 \wedge q_2)}$.

Although **Modular Proof** provides a basis to carry out divide-and-conquer event deductions, doing exchange for each common symbol between two deduction units is inefficient. We then optimize **Modular Proof** to design Algorithm 2. Knowledge partitioning, space splitting, and lazy exchanging are the basic components in our optimization.

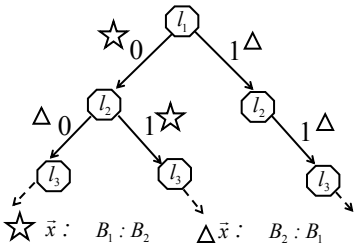


Figure 9. Split search space

The knowledge partitioning method is discussed in Section 4. The lazy exchanging way, *i.e., exchanging the assignments to all common symbols rather than one common symbol*, will be shown in Algorithm 2, and the space splitting makes different deduction units work on different spaces for concurrently checking common symbols. The way to split the solution space is illustrated in Fig. 9 [63], [62], [54], where the common propositions between two event deduction units are assumed to be $l_1, l_2, l_3, \cdots$, and common continuous variables are $\vec{x}$. One unit labeled by star has a discrete search space $\Omega_1 : l_1 = 0, l_2 = 1, \cdots$, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

the other unit labeled by triangle has a discrete search space Ω_2 :

$$l_1 = 1, l_2 = 1, \cdots,$$
$$l_1 = 1, l_2 = 0, \cdots,$$
$$l_1 = 0, l_2 = 0, \cdots.$$

The solution space for continuous variables $\vec{x}$ in the non-linear constraints is assumed to be split into intervals B_1 and B_2 according to Definition 1 [49]. The first unit (labeled by star) firstly searches solutions on B_1 and Ω_1 and then others. If intervals of $\vec{x}$ are found as well as assignments of $l_1, l_2, l_3, \cdots$, the first unit exchanges them into the second unit. The second unit (labeled by triangle) firstly searches solutions on B_2 and Ω_2, and then does the rest tasks as the first unit does. The split search spaces including discrete search spaces and continuous intervals are denoted as F_1, F_2 for Λ_1, Λ_2 respectively.

Algorithm 2. Concurrent Deduction on Unit 1 and Unit 2

Input. *A knowledge set* $\Lambda = \Lambda_1 \wedge \Lambda_2$ *of resources and events, an interest* $q = q_1 \wedge q_2$ *for* $L(\Lambda_1), L(\Lambda_2)$, *and search space* F_1, F_2 *for units 1, 2*

Output. ***False***, *or* ***True*** *and model*

1. $\Lambda_1' = \Lambda_1 \cup q_1$, $\Lambda_2' = \Lambda_2 \cup q_2$
2. **find** *models* m_i *for* $\Lambda_i', i = 1,2$ concurrently on F_1, F_2
3. **if** $m_1 = \emptyset \vee m_2 = \emptyset$ **then return** ***False*** *//has no model*
4. **extract** *common Boolean sub-models* $m_{1,2}, m_{2,1}$ *from* m_1, m_2
5. **extract** *common sub-intervals* $B_{1,2}, B_{2,1}$ *from* $m_{1,2}, m_{2,1}$
6. **exchange** $B_{1,2}, m_{1,2}$ and $B_{2,1}, m_{2,1}$ *between the two units*
7. **if** $m_{2,1} \wedge \Lambda_1' \models_{B_{2,1}} False$ *//check satisfiability under theory* **then send** ***False***
   to unit 2 for **discarding** $B_{2,1}, m_{2,1}$ *and* **goto 2.** *for new models,*
   else get *a model* m_1' *for* $m_{2,1} \wedge \Lambda_1'$ and **send** ***True*** *to unit 2*
8. *unit 2 does as in 7.*
9. **get** *a complete model* $m = m_1' \wedge m_2$ *or* $m = m_1 \wedge m_2'$
10. **return** ***True*** *and m*

In each event deduction unit, we can further partition the knowledge set if there are more computation capabilities such as on cloud computing platforms in a data center. Compared with the work of [64], [63], [62], [54], the **interpretations** of common symbols should be **exchanged** between the two deduction units (***Line 7*** *in Algorithm 2, it firstly checks whether the common Boolean sub-model* $m_{2,1}$ *results in satisfiability for the other knowledge set* Λ_1' *under the condition of the solution* $B_{2,1}$ *to common non-linear constraints, or not:*

$$m_{2,1} \wedge \Lambda_1' \models_{B_{2,1}} False \quad meaning\ unsatisfiability)$$

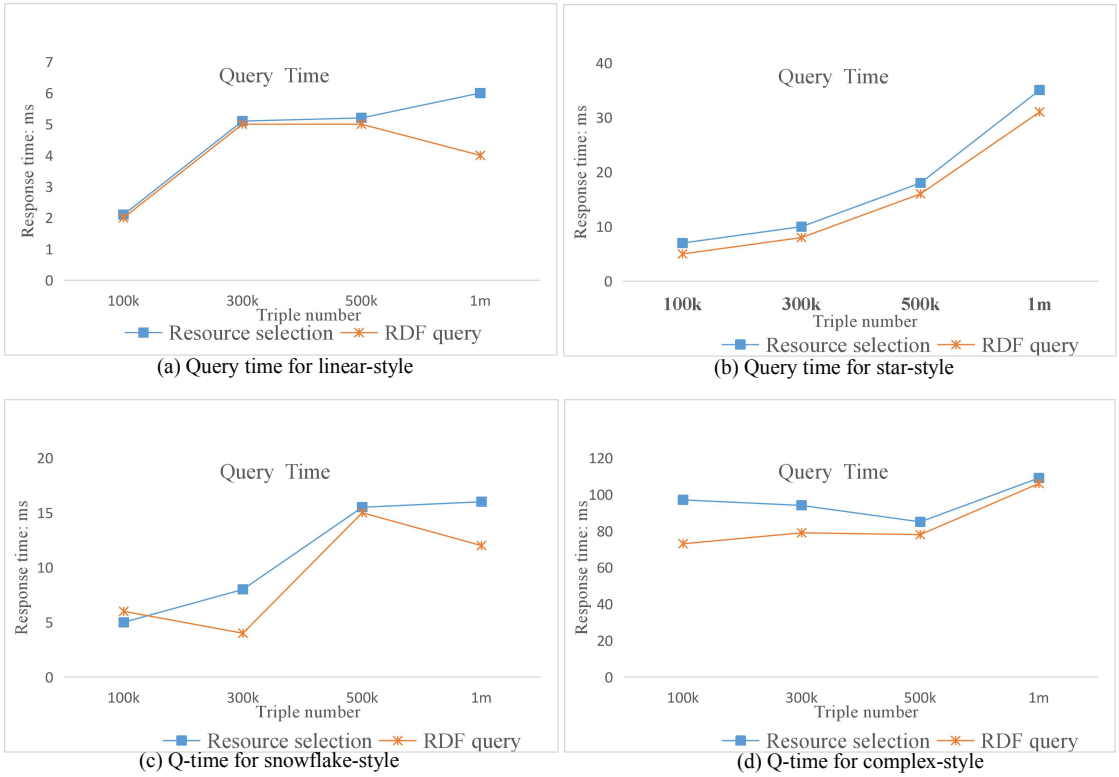
in our work, such that **Modular Proof** and **Theorem 1** are provided as the basis.

7 Experiments

We conducted experiments on computers with Intel(R) Core(TM) i5-3479 CPU: 3.20GHz, and 8.0 GB RAM, running Windows 7. The *libdeep* free software [25] is used as the underlying DNN learning component, Z3 software [26] and dReal tool [52] are adopted as the SMT solver in our solution with embedding the error rate evaluation, and *Apache Jena* [43] is used for SPARQL resource queries.

7.1 Resource Selection

For evaluating our resource selection algorithm, a benchmark of WatDiv [53] is used to enable diversified stress querying RDF data. Four data sets are generated with sizes from 100 thousand to 1 million triples. Twenty query templates are provided by WatDiv including linear, star, snowflake, and complex categories. The data sets are annotated to represent IoT resources according to Definition 2. An additional program for



(a) Query time for linear-style

(b) Query time for star-style

(c) Q-time for snowflake-style

(d) Q-time for complex-style

Figure 10. Resource selection performance

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

Apache Jena [43] is developed to further process the response results of RDF queries for getting resources by space boundaries and resource graphs.

Fig. 10 illustrates the performance comparison between the RDF query and the resource selection, where the time spent for resource selections is comparable to the one for RDF queries. Although the query time is at the level of milliseconds in Fig. 10, the resource selection can be carried out before the complex event processing actually starts. A publish/subscribe event can be viewed as a special resource (*often having less than tens of triples*), and its selection performance is represented by the response time of RDF queries in a dataset consisting of such a resource, *also illustrated in Fig. 10*, which is very efficient (*nanoseconds*).

7.2 Approximating Continuous Variables

To approximate the continuous variables in IoT resources, our experimental data are collected from monitoring *Methane Gas Resource* and *Mine Ventilation Resource*, where a coal mine data simulator generates data sets with different record numbers (from one thousand to one million records). The resource attributes in our data set include the metered speeds/quantities of wind produced by the mine ventilation system, the variation of wind speeds metered at the concerned position, the metered gas generation speed at the coal bed, the geometric parameters of tunnel at the concerned position, and gas density. The accurate coal tunnel's geometric model is kept unknown, *i.e., no explicit equations about air dynamics in the coal mine*. We try to predict the values of gas density when the initial wind generation varies. Some data records are used to train a DNN model, and the other records are used to test the trained DNN. Note that the training records and the testing records are disjoint.

Table 1 illustrates the error rate of computing the gas density. From Table 1, we can conclude that the continuous variable computation method by DNN may be usable with a low error rate.

Table 1. Error Rates for Computing Continuous Variables

DNN Parameters		Training Error Rate	Testing Error Rate
Hidden Layers	Hidden Units		
6	53	0.449%	6.326%

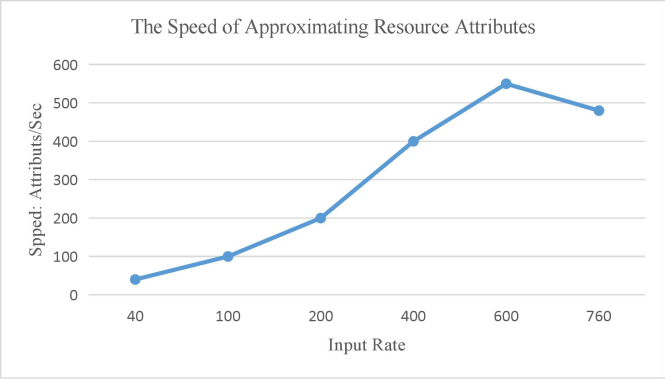


Figure 11. Performance of approximation

After having learned the parameters of DNN for resource attributes, we can use them to rapidly approximate these attribute values. Fig. 11 illustrates the approximation speed of attributes on a single DNN instance. When the input number of approximation-needed attributes increases per second, the throughput of approximation increases with its top boundary. If a high throughput is needed, we can deploy multiple DNN instances to assume this task. The speed of approximation is fast, although it is slow to train the DNN parameters.

7.3 Event Deduction

As the work of [42], we set an event Schema, randomly generate events according to the Schema, and adopt the throughput metric (the number of events processed per second) to measure the performance of a complex event processing system. We make five kinds of experiments. The first one is to test the performance of CEP service without introducing IoT resources, which is used to compare it with classic CEP systems. The others are based on the knowledge of IoT resource models, where the resource example in Fig. 3 is used as a resource template to randomly generate IoT resources, only a small part of them are deliberately generated with some features for experiments, and the two kinds of resources together act as a selected resource set. The second one is to derive interest goals indirectly from event flows, where only one deliberate resource model exists without random resource models (Fig. 13). We then add random resources to show how the number of resources affects the performance of CEP, which is tested in the next two experiments (Fig. 14 and Fig. 15). In the last kind of experiments, we split the resource set, and compute interest goals by concurrent deductions.

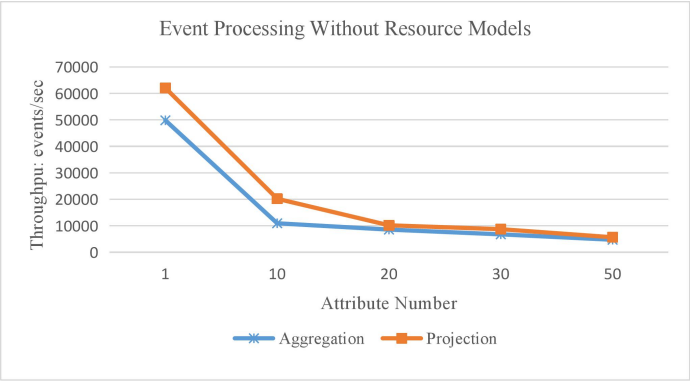


Figure 12. Performance without resource models

Fig. 12 illustrates the event processing without resource models, where "*Filter*" means selecting events by executing filters of ">, < and =" on event attributes; and "*Aggregation*" means imposing the functions of summation and median over event attributes. As we mentioned before, this experiment is to compare our CEP service system with classic event processing systems. Fig. 12 shows that when the number of event attributes becomes big, the performance of our CEP service is considerable and comparable to the ones in the work of [42].

Figure 13. Performance with resource models

Fig. 13 shows our experimental results with IoT resource models, where only *Mine Ventilation Resource* and *Methane Gas Resource* are used without random resources, and the selectivity (the horizontal axis) indicates how many interest goal events are indirectly derived based on IoT resource models in all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering

input events, *i.e., the ratio of derivable events to all the events*. Fig. 13 shows that when the ratio increases, the throughput decreases. Although the throughput of event processing is down to 25,000 events/second, it is also close to the half of the highest one in Fig. 12.

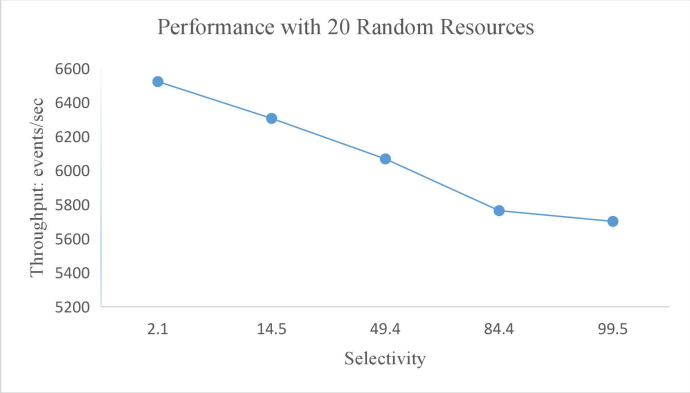


Figure 14. 20 random resources

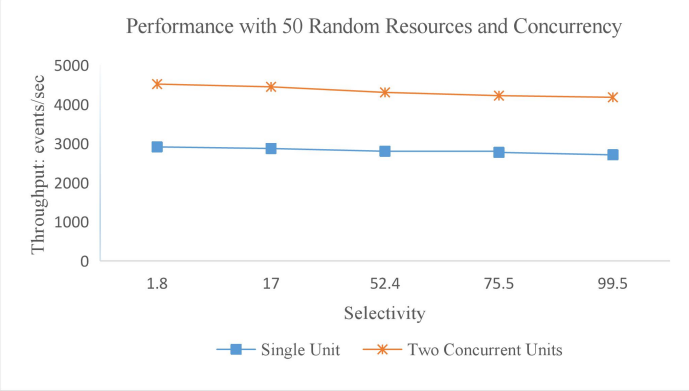


Figure 15. 50 random resources

In Fig. 14, only one unit works on a resource set. From Fig. 14, we know that, when the number of resources increases, the throughput of our CEP service rapidly decreases, and concurrent event deduction is needed. Fig. 15 shows that the throughput of each deduction unit in the two concurrent deduction units is not two times as much as the one of single deduction unit, because interactions between two units and synchronization among threads also take certain cost. But the sum of two deduction units' throughput is more than twice a single unit's one, because each unit in the concurrent way handles with half of the resources in the single one.

7.4 Performance Comparison

We compared our work with the work of [61] , where the latter [61] is the most recent related work, as far as we know, and it parallelized the Z3 SMT solvers [26], *called PD*. We also adopted *PD* to carry out the event deduction task, *i.e., as an underlying parallel SMT solver for comparison*. Although *PD* [61] can decompose a SMT problem into sub-problems with iterating search for interpolants between sub-problems, splitting solution spaces was not further explored in [61], as well as partitioning knowledge including non-linear constraints. In order to complete the comparison, the benchmark data from the website of [69] is used as the basic knowledge set, where the benchmarks were submitted by different participants, and they often represented hard problems that need more time to be solved than general ones. The main track of [69] is used in our work (randomly selecting about 10 benchmarks).

For comparing the event deduction performance on different benchmarks, the deduction time cost is normalized by dividing by one of the standard Z3 solver [26]. For comparing with *PD* [61] on multiple processor cores, the knowledge is partitioned into 4 or 8 partitions. In Fig. 16, the first two cases showed that ours is comparable to *PD* [61] when a knowledge set is partitioned into small sets and the lazy exchanging strategy is adopted. Our method and the *PD* method respectively outperform each other at different points. For example, ours wins with 0.7 at 8 partition point in the second case. When the search-space splitting method is used together with the knowledge partition method, the last case in Fig.16 showed that the deduction performance in our work is improved over the one of *PD*, and the deduction performance for 8 partitions is close to 0.14, *i.e., about tens times faster than the one in the second case*. The combination of space splitting and knowledge partitioning works better.

For comparing the fog-enabled event processing with the event processing only on the data center, we conduct experiments from the four fog-enabled perspectives in our CEP service. From the event deduction perspective, the event deduction algorithm on multiple knowledge partitions is constrained by adding communication delay into the lazy exchange between one main partition and the others, and the low computation capability on edges can also be represented by this delay. From the event delivery perspective, we tested the delay and throughput of event delivery between the edges and the center to investigate *whether such communication supports the fog-enabled event processing*, which is carried out on our publish/subscribe middleware for IoT applications [18]. From the resource storage and resource selection perspectives, the local resource storage reduces the cost of dynamic resource selection from the whole knowledge set on the data center, and Fig. 10 indirectly showed the gain in terms of saving the resource selection cost.

In the first case in Fig. 17, the communication delay from the edges to the center is set from 10 milliseconds to 500 milliseconds, and the performance is measured in terms of the ratio of the average event deduction time of this delayed method to that of the non-delayed method only on the center. It shows

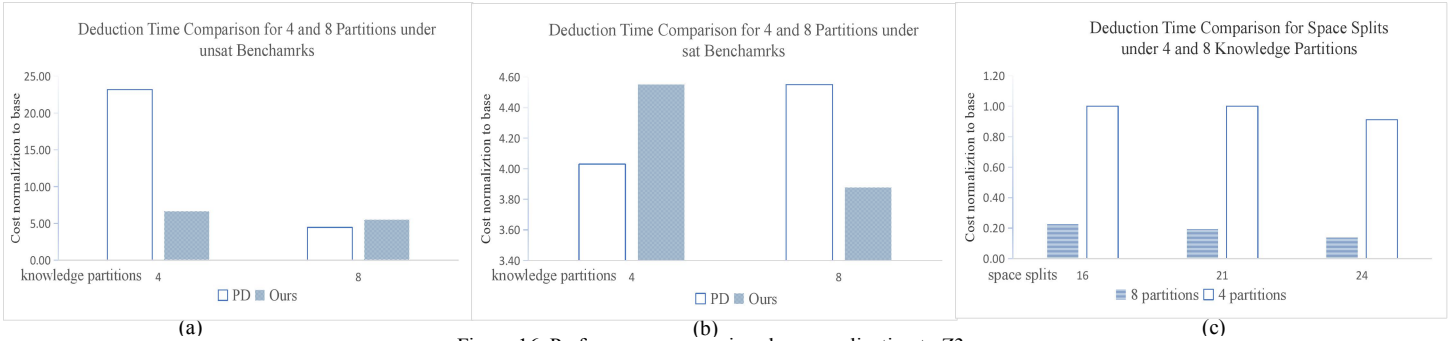
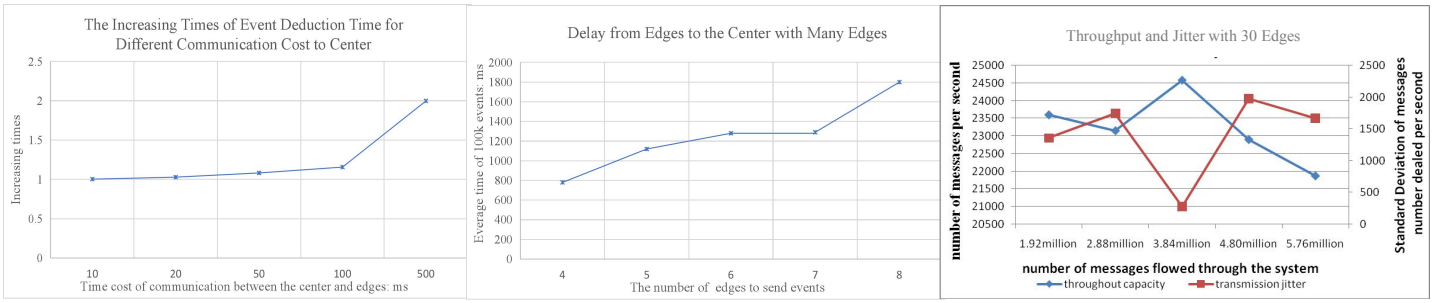


Figure 16. Performance comparison by normalization to Z3

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2867504, IEEE Transactions on Knowledge and Data Engineering



(a) Cost times for communication delay (b) Event delay from edges to the center (c) Event delivery throughput and jitter

Figure 17. Fog-enabled performance and comparison

that, when the communication delay is less than 100 milliseconds, the fog-enabled deduction time is less than 1.157 times of the one only on the data center. When the communication delay is close to 500 milliseconds, the computation cost increases rapidly. The second case in Fig. 17 showed that the actual event delivery delay can be optimized to a very small one with supporting fog-enabled event processing, where the number of edges represents the edge concurrency degree. From the third case in Fig. 17, we can see that the throughput can reach the maximal 24500 events per second when there are 30 edges working concurrently; and the jitter ranges in 450~2000 events/second, *about 8%*. These cases showed that our fog-enabled event processing has a low response delay (shown in the first case of Fig. 17, Fig. 16, and Fig. 10), as well as a low event delivery delay and a high throughput, *i.e., concurrent edges and real-time local responses*.

8 Related Work

Existing event processing methods can be classified into five categories [12]: non-deterministic finite automata [1], [2], [3], finite state machine methods [4], [5], tree methods [6], graph methods [7], [8], [9], and network methods [10], [11]. In the first category of event processing methods, a complex event is represented by a non-deterministic finite automaton, where a state transition is labeled by an atomic event, and the related event found in the event stream triggers the transition. Agrawal *et al.* [1] and Zhang *et al.* [2] adopted a match buffer to realize non-deterministic finite automata [3]. Non-determinism means that two edges at some states are not mutually exclusive. The method based on finite state machine is a simple version of the first category, where the input for each state is treated as incoming edges. The method based on finite state machine was discussed by [4], [5]. Tree structures are also used to detect complex events, where leaves are atomic events, and internal nodes are composite operators to compose atomic ones [6]. Event composition graphs were used by [7], [8], [9] to represent all composition rules in a single structure, and an event used by multiple rules had one occurrence in the graph. In an event processing network, one event processing agent translates incoming events to derived events, and the derived events are sent to the next agents. This category of event processing methods was presented by [10], [11]. How to detect complex events over IoT resources is not discussed in these exiting works, which complicates the event processing by introducing semantic knowledge, and requires divide-and-conquer methods to improve the processing performance.

Based on cloud computing platforms, some divide-and-conquer schemes have been proposed to detect complex events in existing work [12]. Hirzel [59] proposed a pattern-partitioned concurrent event processing scheme, where input event streams were partitioned and isolated by CEP pattern partitions, and combining partial matches between partitions was not needed. In fact, the partition-independence requirement is too strict for practical event rules. Balkesen *et al.* [60] proposed a state-based concurrent event processing scheme, where complex event models represented by states and state transitions were decomposed according to the states, and one processing unit was in charge of some states checking. The results from different processing units should be synthesized. Schultz-Moller *et al.* [4] proposed an operator-based concurrent event processing scheme, where each event processing unit was in charge of a part of event composition operators, and different units exchanged their partial results to obtain complete event composition. All these related works did not address the difficulty of partitioning a semantic knowledge set.

Teymourian et al. [34] also used modular knowledge to process events in complex real-world scenarios. However, like other previous works, this work does not focus on IoT applications to address the issues of event processing. Our work focuses on IoT applications, and addresses the issue of attribute approximation and concurrent event processing.

The work by Cugola et al. [37] is similar to ours. They used formal event knowledge in first-order logic to analyze and correct CEP rules. In order to achieve the processing efficiency for practical usage, property checking tasks are translated into constraints solving problems. In our work, we use divide-and-conquer methods to handle deduction tasks. Margara et al. [36] explored how to learn CEP rules from historical traces, avoiding using the error-prone manual labor. They focused on learning event causality from the past, and advocated generating CEP rules automatically. Lee *et al.* [56] proposed a mining method to automatically generate event processing rules. Event sequence similarity was firstly estimated based on reference point designation, relative distance, and peripheral similarity; and then a Markov probability transition model was used as the basis to obtain complex event patterns. In our work, declarative event processing goals are defined, and complex relations among events and resources (e.g. non-linear constraints) are allowable if they derive the goal.

To determine whether a formula is satisfiable over some theories [13], [14], [27] is a problem of satisfiability modulo theories (SMT). There are lots of significant developments of SMT in the past decade, such as modular combination theorems [15], [16], [17]. As knowledge representation, DL [20], [22], [23] is mainly used in many previous works [21], [24], [35].

9 Conclusions

In IoT applications, complex events should be processed on IoT resource models. The key point of completing this task is to partition the knowledge set of IoT resources, and to make

fog-enabled concurrent event deductions over multiple resource partitions. In this paper, we define the resource selection problem using non-linear constraints and RDF graph queries, and then solve the problem using the *Scope Search* algorithm that is designed based on the resource match concept and the locality observation of IoT resources and events. A method of modular proof is proposed in this paper to test satisfiability of the knowledge partitions of events, resources, and non-linear constraints, which provides a theoretic basis for concurrent event deductions. A concurrent event deduction algorithm is proposed to optimize the modular proof method by knowledge partitioning, search-space splitting, and lazy exchanging, which realizes the collaboration between fog edges and the data center. Our experimental evaluation shows that our solution is feasible and effective.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (no. 61372115), and the National High-Tech R&D Program of China (863 Program) under Grant no. 2013AA102301.

References

[1] J. Agrawal, Y. Diao, D. Gyllstrom, N. Immerman, "Efficient Pattern Matching over Event Streams," *Proceedings of SIGMOD 2008*, pp. 147-160, 2008.

[2] H. Zhang, Y. Diao, N. Immerman, "Optimizing expensive queries in complex event processing," *Proceedings of SIGMOD 2014*, pp. 217–228, 2014.

[3] A. Demeers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, W. White, "Cayuga: A General Purpose Event Monitoring System," *Proceedings of 3rd Biennial Conference on Innovative Data Systems Research (CIDR)*, pp. 412-422, 2007.

[4] N. P. Schultz-Moller, M. Migliavacca, P. Pietzuch, "Distributed Complex Event Processing with Query Rewriting," *Proceedings of DEBS 2009*, pp. 1-12, 2009.

[5] M. Akdere, U. Cetintemel, N. Tatbul, "Plan-based Complex Event Detection Across Distributed Sources," *Proceedings of VLDB Endowment 1(1)*, pp. 66-77, 2008.

[6] Y. Mei, S. Madden, "Zstream: A Cost-based Query Processor for Adaptively Detecting Composite Events," *Proceedings of SIGMOD Conference on Management of Data*, pp. 193-206, 2009.

[7] S. Wasserkrug, A. Gal, O. Etzion, "A Model for Reasoning with Uncertain Rules in Event Composition Systems," *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, UAI-P-2005-PG-599-608, 2012.

[8] S. Wasserkrug, A. Gal, O. Etzion, Y. Turchin, "Complex Event Processing over Uncertain Data," *Proceedings of the Second International Conference on Distributed Event-based Systems,* ACM, New York, USA, pp. 253–264, 2008.

[9] S. Wasserkrug, A. Gal, O. Etzion, Y. Turchin, "Efficient Processing of Uncertain Events in Rule-based Systems," *IEEE Transactions on Knowledge & Data Engineering*, 24(1), pp. 45-58, 2012.

[10] Y. Wang, K. Cao, X. Zhang, "Complex Event Processing over Distributed Probabilistic Event Streams," *Comput. Math. Appl. 66 (10)*, pp. 1808-821, 2013.

[11] O. Etzion, P. Niblet, "Event Processing in Action," Manning Publications Co., 2010.

[12] I. Flouris, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, M. Mock, "Issues in Complex Event Processing: Status and Prospects in the Big Data Era," *The Journal of Systems and Software*, 127, pp. 217-236, 2017.

[13] C. Barret, R. Sebastiani, S. A. Seshia, C. Tinelli, "Satisfiability Modulo Theories," *Chapter 26, Handbook of Satisfiability*, pp. 825-885, 2009.

[14] R. Sebastiani, "Lazy Satisfiability Modulo Theories," J*ournal on Satisfiability, Boolean Modeling and Computation,* JSAT 3, 3-4, pp. 141–224, 2007.

[15] G. Nelson and D. C. Oppen, "Simplication by Cooperating Decision Procedures," *ACM Transactions on Programming Languages and Systems*, 2(1), pp. 245-257, 1979.

[16] R. E. Shostak, "Deciding Combination of Theories," *Journal of the Association for Computing Machinery*, 1(31), pp. 1-12, 1984.

[17] [Online]. Available: http://www.spass-prover.org, Accessed on: 2014.

[18] Y. Zhang, J. L. Chen, "Declarative Construction of Distributed Event-driven IoT Services Based on IoT Resource Models," *IEEE Transactions on Services Computing*, DOI: 10.1109/TSC.2017.2782794, 2017.

[19] Eyal Amir, Sheila McIlraith, "Partition-based Logical Reasoning for First-order and Propositions Theories," *Artificial Intelligence*, Volume 162, 2000, pp. 49-88, 2000.

[20] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, "The Description Logic Handbook," Cambridge University Press, 2003.

[21] R. Eshuis, F. Lecue, N. Mehandjiev, "Flexible Construction of Executable Service Compositions from Reusable Semantic Knowledge," *ACM Transactions on the Web*, Vol. 10, No. 1, Article 5, pp. 5:1-5:27, Publication date: February 2016.

[22] S. Staab and R. Studer, editors, "Handbook on Ontologies," International Handbooks on Information Systems, Springer, 2nd edition, 2009.

[23] H. Stuckenschmidt, C. Parent, S. Spaccapietra, editors, "Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization," *Volume 5445 of Lecture Notes in Computer Science*,Springer, 2009.

[24] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, "Pellet: A Practical OWL-DL Reasoner," *Journal of Web Semantics*, 5(2), pp. 51-53, 2007.

[25] [Online]. Available: https://github.com/bashrc/libdeep, Accessed on: 2015.

[26] [Online]. L. de Moura, N. Bjørner, "An Efficient Theorem Prover," Available: http:// research. microsoft.com/en-us/um/redmond/projects/z3/ and https://github.com/ z3prover/z3/wiki, Accessed on: 2015.

[27] G. Robinson, L. Wos, "Paramodulation and Theorem-proving in First-order Theories with Equality," *Automation of Reasoning Symbolic Computation 1983*, pp 298-313, 1983.

[28] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and Bottleneck Features for LVCSR of Meetings," *Proc. ICASSP 2007*, pp. 757–760, 2007

[29] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504-507, 2006.

[30] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why Does Unsupervised Pretraining Help Deep Learning?" *Proc. of AISTATS 2010*, vol. 9, May 2010, pp. 201-208, 2010.

[31] J. E. Fenstad, "The Structure of Probabilities Defined on First-order Languages," *Studies in Inductive Logic and Probabilities*, pp. 251-262, California Press, 1980.

[32] B. Wuthrich, "Probabilistic Knowledge Bases," *IEEE Trans. Knowl. Data Eng.* 7(5), pp. 691-698, 1995.

[33] R. Ng, V. S. Subrahmanian, "Probabilistic Logic Programming," *Information and Computation* 101(2), pp. 150-201, 1992.

[34] K. Teymourian, G. Coskun, A. Paschke, "Modular Upper-Level Ontologies for Semantic Complex Event Processing," *Proceedings of the 2010 Conference on Modular Ontologies: Proceedings of the Fourth International Workshop (WoMO 2010),* pp. 81-93, 2010.

[35] H. Stuckenschmidt, A. Schlicht, "Structure-based Partitioning of Large ontologies," *Modular Ontologies*,Volume 5445 of the series Lecture Notes in Computer Science, pp. 187-210, 2009.

[36] A. Margara, G. Cugola, G. Tamburrelli, "Learning from the Past: Automated Rule Generation for Complex Event Processing," *ACM International Conference on Distributed Event-based Systems*, pp. 47-58, 2014.

[37] G. Cugola, A. Margara, M.Pezze, M. Pradella, "Efficient Analysis of

Event Processing Applications," *ACM International Conference on Distributed Event-based Systems*, pp. 10-21, 2015.

[38] M. Davis, G. Logemann, D. Loveland, "A Machine Program for Theorem-proving," *Comm. of the ACM* 5, 7, pp. 394–397, 1962.

[39] M. Davis, H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of the ACM* 7, pp. 201–215, 1960.

[40] [Online]. Available: http://esper.codehaus.org/, Accessed on: 2014.

[41] [Online]. Available: http://www.w3.org/2005/Incubator/ssn/ssnx/ssn, Accessed on: 2015.

[42] M. R.N. Mendes, P. Bizarro, P. Marques. "A Performance Study of Event Processing Systems," *Performance Evaluation and Benchmarking*, Volume 5895 of the series Lecture Notes in Computer Science, pp. 221-236, 2009.

[43] [Online]. Available: http://jena.apache.org/, Accessed on: 2015.

[44] [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/, Accessed on: 2015.

[45] [Online]. Available: http://www.w3.org/TR/xmlschema-1/, Accessed on: 2015.

[46] P. Peng, L. Zou, M.T. Özsu, L. Chen, D. Zhao, "Processing SPARQL Queries over Distributed RDF Graphs," *The VLDB Journal*, 25 (2), pp. 243-268, 2016.

[47] R. E. Moore, R. B. Kearfott, M. J. Cloud, "Introduction to Interval Analysis," *Society for Industrial and Applied Mathematics*, 2009.

[48] J. Ninin, F. Messine, P. Hansen, "A Reliable Affine Relaxation Method for Global Optimization," *4OR*, 13 (3), pp. 247-277, 2015.

[49] S. Gao, J. Avigad, E.M. Clarke, "δ-Complete Decision Procedures for Satisfiability over the Reals," *Computer Science*, 7364, pp. 286-300, 2012.

[50] [Online]. Available: https://www.w3.org/TR/owl2-mapping-to-rdf/, Accessed on: 2013.

[51] M. Koubarakis, K. Kyzirakos, "Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL," *International Conference on the Semantic Web: Research & Applications*, 2 (1), pp. 425-439, 2010.

[52] S. Gao, S. Kong, E.M. Clarke, "dReal: An SMT Solver for Nonlinear Theories over the Reals," *International Conference on Automated Deduction*, 7898, pp. 208-214, 2013.

[53] G. Aluç, O. Hartig, M. T. Özsu, K. Daudjee, "Diversified Stress Testing of RDF Data Management Systems," *Proc. 13th Int. Semantic Web Conf.*, pp. 197–212, 2014.

[54] A. E. J. Hyvarinen, M. Marescotti, N. Sharygina, "Search-space Partitioning for Parallelizing SMT Solvers," *Springer International Publishing Switzerland 2015, M. Heule and S. Weaver (Eds.): SAT 2015*, LNCS 9340, pp. 369–386, 2015.

[55] [Online]. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, "SWRL: a Semantic Web Rule Language combining OWL and RuleML," W3C Member submission May 21st 2004, Available: http://www.w3.org/ Submission/SWRL/, Accessed on: 2015.

[56] O. J. Lee, J. E. Jung, "Sequence Clustering-based Automated Rule Generation for Adaptive Complex Event Processing," *Future Generation Computer Systems*, 66, pp. 100-109, 2017.

[57] M. Chiang, T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, Vol. 3, No. 6, pp. 854-864, 2016.

[58] A. Sill, "Standards at the Edge of the Cloud," *IEEE Cloud Computing*,vol. 4, no. 2, pp. 63–67, 2017.

[59] M. Hirzel, "Partition and Compose: Parallel Complex Event Processing," *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pp. 191–200, 2012.

[60] C. Balkesen, N. Dindar, M. Wette, N. Tatbul, "Rip: Run-based Intra-query Parallelism for Scalable Complex Event processing," *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, pp. 3-14, 2013 .

[61] X. Cheng, M. Zhou, X.Y. Song, M. Gu, J.G. Sun, "Parallelizing SMT Solving: Lazy Decomposition and Conciliation," *Artificial Intelligence,* 257, pp. 127-157, 2018.

[62] H. Zhang, M.P. Bonacina, J. Hsiang, "PSATO: A Distributed Propositional Prover and Its Application to Quasigroup Problems," *Journal of Symbolic Computation* 21, pp. 543–560, 1996.

[63] G. Audemard, L. Simon, "Lazy Clause Exchange Policy for Parallel SAT Solvers," *International Conference on Theory and Applications of Satisfiability Testing*, pp. 197-205, 2014.

[64] M. Bohm, E. Speckenmeyer, I.F. Informatik, "A Fast Parallel SAT solver - Efficient Workload Balancing," *Annals of Mathematics & Artificial Intelligence*, 17 (2) , pp. 381-400, 1996.

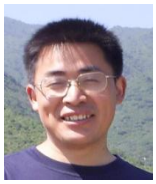
[65] P. Verma, S. K. Sood, "Fog Assisted-IoT Enabled Framework for Patient Health Monitoring in Smart Homes," *IEEE Internet of Things*, February 2018, DOI 10.1109/JIOT.2018.2803201.

[66] D. Zeng, L. Gu, S. Guo, S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-defined Embedded System," *IEEE Transactions on Computers*, Volume: 65, Issue: 12, pp. 3702-3712, December 2016.

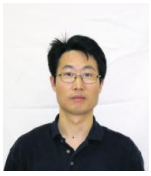
[67] I. Azimi, A. Anzanpour, A.M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, N. Dutt, "HiCH: Hierarchical Fog-assisted Computing Architecture for Healthcare IoT," *ACM Transactions on Embedded Computing Systems*, 16 (5s), October, 2017, DOI 10.1145/3126501.

[68] M.B.A. Prashan Madumal, D.A.S. Atukorale, T.M.H.A. Usoof, "Adaptive Event Tree-based Hbrid CEP Computational Model for Fog Computing Architecture," *International Conference on Advances in ICT for Emerging Regions (ICTer)*, pp. 5-12, 2016.

[69] [Online]. Available: https://baldur.iti.kit.edu/sat-competition-2017/index.php?cat=benchmarks, Accessed on: 2018.



Yang Zhang received the degree of Ph.D. in computer applied technology from Institute of Software, Chinese Academy of Sciences in 2007. His research interests include *Service-oriented Computing*, *Internet of Things, and Service Security and Privacy*. He currently works in State Key laboratory of Networking and Switching Technology, Beijing University of Posts & Telecommunications, Beijing, China. He leads a team making scientific research on *The Theoretic Foundation of EDSOA for IoT Services* (National Natural Science Foundation of China under Grant No. 61372115).



Victor S. Sheng is an Associate Professor of Computer Science at the University of Central Arkansas, and the founding Director of Data Analytics Lab (DAL). He received his Master's degree in Computer Science from the University of New Brunswick, Canada, in 2003, and his Ph.D. degree in Computer Science from Western University, Ontario, Canada, in 2007. His research interests include data mining, machine learning, and related applications. He was an Associate Research Scientist and NSERC Postdoctoral Follow in Information Systems at Stern Business School, New York University after he obtained his Ph.D. Dr. Sheng is a senior member of IEEE and a lifetime member of ACM. He received the best paper award runner-up from KDD '08, and the best paper award from ICDM'11. He is a PC member for many international conferences and a reviewer for many international journals.