Stream-based Reasoning for IoT Applications – Proposal of Architecture and Analysis of Challenges

Markus P Endler, Jean-Pierre P Briot, Vitor P. de Almeida, Ruhan P dos Reis, Francisco Silva E Silva

Stream-based Reasoning for IoT Applications – Proposal of Architecture and Analysis of Challenges

Markus Endler
Dept of Informatics, PUC-Rio, Rio de Janeiro, Brazil
Email: endler@inf.puc-rio.br

Jean-Pierre Briot
LIP6, UPMC–CNRS, Paris, France
Dept of Informatics, PUC-Rio, Rio de Janeiro, Brazil
Email: Jean-Pierre.Briot@lip6.fr

Vitor P. de Almeida
Dept of Informatics, PUC-Rio, Rio de Janeiro, Brazil
Email: valmeida@inf.puc-rio.br

Ruhan dos Reis
Dept of Informatics, PUC-Rio, Rio de Janeiro, Brazil
Email: rreis@inf.puc-rio.br

Francisco Silva e Silva
LSDi, Univ. Federal do Maranhão, São Luis, Brazil
Email: fssilva@lsdi.ufma.br

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

As distributed IoT applications become larger and more complex, the pure processing of raw sensor and actuation data streams becomes impractical. Instead, data streams must be fused into tangible facts and these pieces of information must be combined with a background knowledge to infer new pieces of knowledge. And since many IoT applications require almost real-time reactivity to stimulus of the environment, such information inference process has to be performed in a continuous, on-line manner. This paper proposes a new semantic model for data stream processing and real-time reasoning based on the concepts of *Semantic Stream* and *Fact Stream*, as a natural extension of Complex Event Processing (CEP) and RDF (graph-based knowledge model). The main advantages of our approach are that: (a) it considers time as a key relation between pieces of information; (b) the processing of streams can be implemented using CEP; (c) it is general enough to be applied to any Data Stream Management System (DSMS). We describe a scenario about patients flux monitoring in a hospital as an example of prospective application. Last, we will present challenges and prospects on using machine learning and induction algorithms to learn abstractions and reasoning rules from

a continuous data stream.

Keywords: Internet of Things (IoT); sensors; data streams; complex event processing (CEP); semantic reasoning; inference; machine learning.

1. Introduction

Several complex IoT applications, such as manufacturing industry, transportation systems and healthcare, put hard real time requirements on the acquisition and processing of sensor data for identifying situations and extracting information from systems' operations and its environment. These typically require on-line processing of continuous streams of sensor data (Data Stream Processing), sensor fusion techniques, pattern recognition and timely and autonomous systems control.

However, so far in current IoT systems, sensing and actuation is mostly done at the *bare bones* data level, whereas many IoT applications demand higher level situation awareness of – and reasoning about – the systems' states and the physical environment where they operate. For this to be possible, it is necessary to have comprehensive semantic models for data stream analysis and actuation. Semantic models are formally defined concepts and relations on which reasoning engines can operate to derive new bits of information and knowledge about a system and its environment. The main problem is that current semantic models (designed for the Semantic Web) are not suitable for efficient and real-time reasoning. Current data analysis for IoT systems is either done off-line or lacks any semantic-based reasoning.

For example, consider a production plant in the near future, where several – mobile or stationary – robots operate in a product assembly and interact with each other to hand over parts and tools of the assembly line. Suddenly, there is a short power outage and the assembly line stops for a few seconds, so that some robots go back to their consistent initial states, while others continue their activity (e.g., on battery power) and only stop when their sensors notice that the production line is not advancing. In this case, the robots have to "understand" what has happened, and have to "know" which of the machinery (and robots) are in which state when activity is resumed, as well as the assembly stage of items being produced. And like magic, only a few seconds after energy is back, the robots synchronize with each other, identify missed steps in the assembly process of each item, and resume cooperating again. Such knowledge and understanding is only possible because all robots have not only a semantic model of their own state, but also situational awareness, i.e. a comprehensive model of the production process as a whole and their role in the entire process. The semantic model furthermore describes possible localized and global problems of the entire production process, as well as individual and specific actuation plans for some situations. As all possible situations cannot be represented in a model, the robots have to classify features, combine situational patterns and combine parts of specific action plans. In the aforementioned IoT scenario, the robots would be capable of such fast recovery of the manufacturing process because their situational understanding (i.e. semantic-centered inference/reasoning process)

is executed very fast, with almost no delay, as soon as each robot's operational capability is back.

With the goal of finding a suitable semantic model for IoT, this paper proposes a novel approach for real-time symbolic reasoning based on the concepts of *Semantic Stream and Fact Stream*, as natural extensions of Complex Event Processing (CEP) [19] and RDF (graph-based knowledge model) [10]. The main advantages of our approach are that: (a) it uses the timestamp and co-location information to correlate actions/events happening at different real-world entities (i.e. objects and subjects); (b) the online processing of semantic streams can be implemented using conventional CEP technology and semantic reasoning approaches; (c) using ontology-based reasoning over a knowledge base, it is possible not only to deduce future or indirect events that would not be detected through CEP, but also to generate new CEP rules for the stream analysis; (d) the approach is generic enough to be applied to many Data Stream Management Systems (DSMS). This research is being carried out in the scope of the ESMOCYP cooperation project between PUC-Rio, Federal University of Maranhão and University of Stuttgart. We are currently developing a prototype of the semantic stream reasoning using ContextNet, our distributed and scalable middleware for the Internet of Mobile Things [22]. It is a mobile-cloud architecture where several interconnected CEP agents can be deployed both in a cloud/cluster [5], as well as on Android mobile devices [21].

The paper is structured as follows. In Section 2, we explain the basic concepts of Complex Event Processing and list some common approaches for modeling knowledge and performing reasoning. Section 3 explains the two steps of semantic stream reasoning. In Section 4, we present a scenario to explain how our reasoning process would be performed using Bluetooth beacon sensors wearable by patients into a hospital emergency environment. Section 5 discusses related work. In Section 6, we discuss the benefits of our approach and prospects. Section 7 presents an initial analysis of how machine learning and induction algorithms could help in automatically or semi-automatically extracting stream analysis patterns and rules. Section 8 then concludes the paper.

2. Fundamentals

2.1. *Complex Event Processing*

Complex Event Processing (CEP) [19] provides a rich set of concepts and operators for processing events, which include the CQL-like (Continuous Query Language) [4] queries, rules, primitive functions (aggregation, filtering, transformation, etc.) and production of derived events. A CEP workflow continuously processes incoming events, analyses and manipulates them, and outputs derived events that are delivered to event consumers. These output usually represent notifications about detected situations of interest to the applications.

The processing of events is described by CEP rules, which are *Event-Condition-Actions* that combine continuous query primitives with context operators (e.g., tem-

poral, logical, quantifiers) on received events, checking for correlations among these events, and generating complex (or composite) events that summarize the correlation of the input events. For example, a *split rule* takes an input event and creates a set of events, while a *filter rule* only outputs events that satisfy a given criteria. Rules can also operate on a collection of events, for example, an aggregate rule outputs a single event by executing a function on the grouped events, while a join transformation tries to correlate events from various data streams. Another important concept in CEP is that of *sliding time and event windows*. A *time window* is a temporal context that subdivides the stream of events into intervals, where CEP rules and operators are applied only to the events within each window. CEP supports three sorts of windows: *landmark*, *sliding* and *fading*, the latter being a sliding window where a decay factor λ is applied to the events according to their age, i.e. more recent events have higher importance than older events. Most CEP systems have the concept of *Event Processing Agents (EPAs)*, which are software modules that implement one transformation within the event processing workflow. The type of an EPA is defined by the rules it implements, such as filtering, counting or specific event pattern detection. Note that rules are hand written by experts. We will address in Section 7, a preliminary analysis of how machine learning and induction algorithms could help in, automatically or semi-automatically, constructing rules as well as extracting patterns.

2.2. *Knowledge Representation and Reasoning Approaches*

There are plenty of Semantic Models that represent knowledge about a system and its environment, but almost all of them have problems of scale (i.e. the reasoning has high computational complexity), and thus are not suitable for real-time reasoning. The main semantic approaches are (see, e.g., for a survey and comparison in [20]):

- **Frame Based Models:** A frame is an artificial intelligence data structure used to divide knowledge into substructures by representing "stereotyped situations". They are used in artificial intelligence Frame languages.
- **Conceptual Graphs**: are a logical formalism that includes classes, relations, individuals and quantifiers. This formalism is based on semantic networks, but it has direct translation to the language of first order predicate logic, from which it takes its semantics.
- **Description Logic:** are logics serving primarily for formal description of concepts and roles (relations). These logics were created from the attempts to formalize semantic networks and frame based systems. Semantically they are found on predicate logic,
- **Ontologies:** An ontology is a semantic/concept network that contains a body of knowledge describing some domain, typically common sense knowledge relating concepts.
- **Semantic Web: RDF, RDFS and OWL:** RDF (Resource Description Framework) is a framework for representing information about resources in

a graph model, where information is represented by triples *(subject, predicate, object)*. RDFS (RDF Schema) extends RDF vocabulary to allow describing taxonomies of classes and properties. It also extends definitions for some of the elements of RDF, for example it sets the domain and range of properties and relates the RDF classes and properties into taxonomies using the RDFS vocabulary. Web Ontology Language (OWL) brings the expressive and reasoning power of Description Logic (DL) to the semantic web. It is divided into two levels: OWL Lite and OWL DL, which differ in their expressive power and the deduction complexity. The limitation with OWL Lite and OWL DL is that reasoning is hardly implemented in an efficient way, and it also suffers from lack of scalability.

- **Contextualized Ontologies:** Contextualized Ontologies [9] are logical structures of the form $(Entity, Context, Link)$, where Entity and Context are both ontologies and Link is a mapping between the Entity and the Context. A link represents thus an alignment between self-contained ontologies, describing how an entity can be viewed from within a specific context.

3. General Idea

The general idea of our semantic model and reasoning approach is to define two-level CEP transformations, each of which transforms one event flow/stream into a semantically richer one: 1) from annotated preprocessed events to RDF triples; and 2) from RDF triples to a stream of facts. Initially, sensor data received from smart objects are pre-processed so as to identify: a) the entity type and instance from the received UUID; and b) what is happening to the entity, e.g., if it is doing some action, experiencing a state change or any other transformation. This 2nd type of pre-processing may be performed, e.g., through CEP (by matching a sequence of data onto a pre-defined temporal pattern identifying a specific pattern of action). This leads to a stream of semantically annotated data with pairs *(subject, predicate)* or *(object, predicate)*. The entity type/instance and predicate identification is performed by CEP agent close to the sensors, (see Figure 1), that in the specific case of our IoT middleware typically execute on mobile devices. Therefore, we named them *Mobile Event Processing Agents (Mobile EPAs)*.

Then, in the first stream processing stage, our approach transforms the stream of annotated data into a stream of RDF statements, and in the second stage, we transform the stream of RDF-triples into semantically richer facts, i.e combining RDF statements. The details of each of these stages are explained in the following.

3.1. *Mapping Data Events to Semantic Events*

Our reasoning approach dictates that each simple annotated event (actually, a data object with member attributes) represents an action-based *predicate* (i.e. the event is the outcome of an action) and has at least one of the other two remaining RDF elements: the *subject* or the *object*. If the event has the ID of the subject and

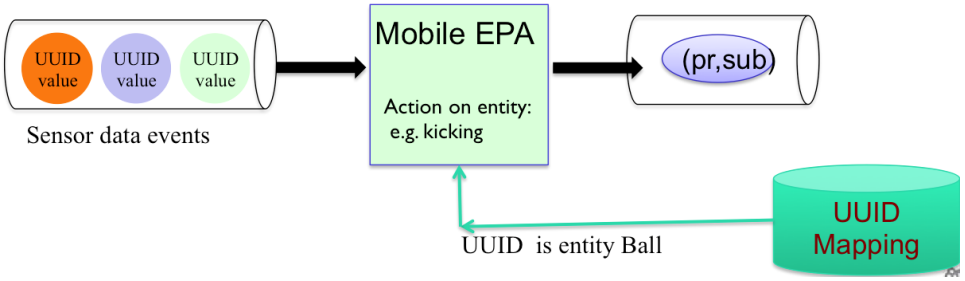

Fig. 1. Semantic annotation from raw sensor data.

the object then we have the complete RDF triple (*subject, predicate, object*), but otherwise, the missing third RDF element of the triple may be inferred from the shared context (i.e. the temporal and spatial correlation) of both elements, the subject and the object when these are received in separate events. For example, if we consider RDF statement *(ball, kicking, in the front-yard)*, then the event instances represent the predicate *kick*. It further carries the ID of either the ball (e.g., when the ball carries an accelerometer sensor), or else the ID of the yard (e.g. the GPS-position or the street number of the yard (e.g., lawn sensors detect some kicking object). And the shared context is defined by the same location (co-location) of the events and the synchronicity of the events that the sensors on the yard ground and the sensor in the ball detect the hitting of the ball with the lawn (the kick). This contextual correlation is performed by CEP rules called *Context mappers*, that analyze the streams of events and match Subjects, Objects and Predicates.

Figure 2 shows how Context mappers analyze each pair of events in the sliding time window (e.g., 60 s.) of Data Event Stream and try to identify common contexts, based on time proximity or any other data attribute.

3.2. *Mapping Semantic Events to Knowledge Facts*

The mapping from Semantic Events (i.e. RDF triples) to Facts is achieved by *Semantic Event (SEv) rules*. These are CEP rules that look out to find causality and temporal patterns in several Semantic Event sub-Streams, where each stream comprises the Semantic events of a given context. This "context-specific splitting" is possible in most CEP engines by the concept of a stream partition (a.k.a. *context*). Then, depending on the SEv rule, it might consume, filter out, modify or even insert new RDF triples in some SEv streams, a feature that is supported by CEP. This manipulation is achieved by querying the Knowledge base about all the concepts and relations pertaining to the sub-streams analyzed. For example, the inference might deduce that the "kicking ball with a given ID" has "Bob" as its owner, and that the "yard where the ball is kicking" is the one where Bob lives. By this, the new piece of knowledge may be derived such as "someone is kicking Bob's ball on

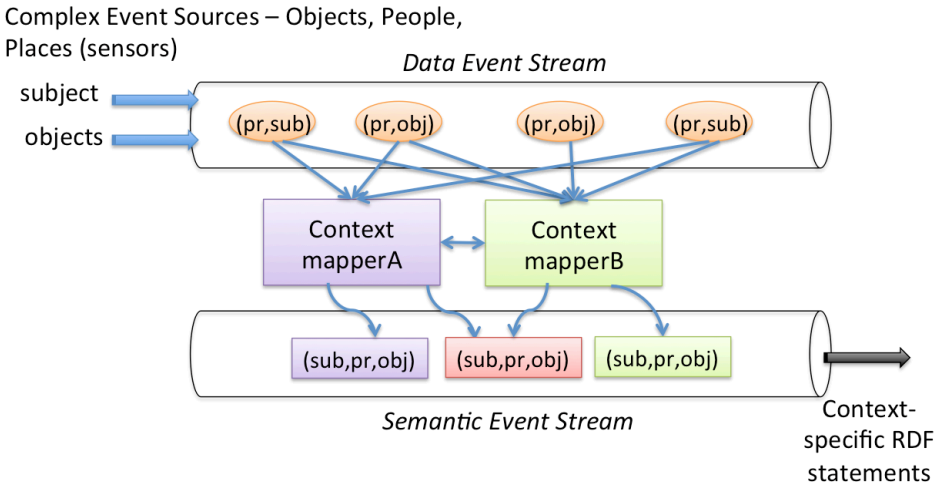

Fig. 2. Mapping data events to semantic events.

his house's yard". And maybe with the context information "Bob has finished his homework", it is possible to deduce – with high probability – that "Bob is kicking his ball in his house's yard".

The Knowledge base is organized as *nested contexts* [9], which allows a much more efficient checking of concepts and relations when compared to single-layer (or flat) ontologies. For example, the ontology of the Knowledge Base may be organized as the following nested contexts: Spatial nested contexts: *"Green Way district"* $\supset$ *"house at 10 Rodeo Dr."* $\supset$ *its yard* $\supset$ *its lawn*; Temporal nested contexts: *"Bob's leisure time"* $\supset$ *"Thursday"* $\supset$ *"afternoon"* $\supset$ *"Bob's homework finished"*; Containment nested context, such as, *"Bob's toys"* $\supset$ *balls* $\supset$ *"Basket ball with ID"*, etc.

Figure 3 shows how Semantic Event rules analyze all RDF triples in the sliding time window (e.g., 180 s.) of sub streams of Semantic Events, trying to find event patterns, filtering, manipulating or adding RDF triples into "their" main context sub-stream or also of sub-streams of semantically related contexts, such as, "the front yard" and the "street in front of the yard".

3.3. *Deriving Situations*

Using the Facts of the stream and checking them against the Semantic Graph (Ontology) of the knowledge base, complex situations may be identified such as "Bob is playing basketball in the front yard, but should be notified that a strong storm is approaching his house's yard". Moreover, some of the complex facts may be used for expanding, reinforcing or removing some the knowledge about a subject, an object or a place. For example, after Bob's pen has finished writing QED on the page with the exercises of his Math's homework notebook, the latter has been

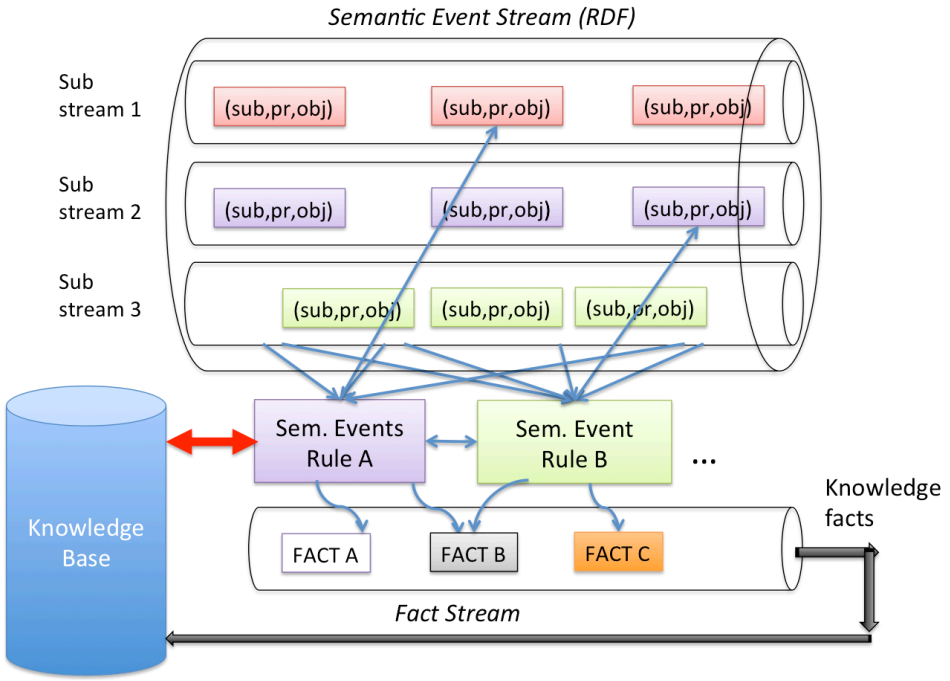

Fig. 3. Mapping semantic events to knowledge facts.

closed, and his Bob house's main door has been opened and closed, sensing that someone left the house, then the Knowledge Base will be expanded with the facts that *(Bob, finished, Math homework)*, *(Bob, left, house)* and *(Bob, stepped into, yard)*.

4. An Example Scenario for Hospital 4.0

In this section, we will show how the aforementioned two-phase reasoning could be done with off-the-shelf components and current wireless WPAN technologies, such as Bluetooth Low Energy (BLE).

4.1. *Sensors and Smart Things Protocol Support*

Smart ambient sensors are coming everywhere: houses, offices, hospitals, transportation... and many of these smart devices include a temperature and an accelerometer sensor, have a unique UUID and Bluetooth Low Energy interface. We have designed an Internet of Things middleware named ContextNet [11, 22] which uses a smartphone as a bridge between Bluetooth-enabled smart devices/objects/sensors and IoT application servers executing in a cloud. Our mobile middleware, named Mobile Hub, periodically issues a BLE scan, discovers nearby BLE devices, connects to them, subscribes to the smart device's sensors and writes commands to

the smart objects that have some actuator. Figure 4 shows the Mobile Hub with 4 *SensorTags*, each with 6 sensors (temperature, accelerometer...).

Fig. 4. ContextNet Mobile Hub with four sensor tags.

4.2. *Scenario*

Let us consider the emergency sector of a hospital and a future application named Hospital 4.0 (H4.0). The general organization of H4.0 (various rooms) is presented at Figure 5. A common problem with an emergency sector of a hospital is to detect bottlenecks within the flow of patients. The detection of bottlenecks can help improve the quality of service and satisfaction of the patients (and possibly save lives).

First, the flow of patients starts at the Reception, where each patient registers his/her entrance. Second, at the Initial Diagnosis Room, a nurse makes a first evaluation of the patients and classifies them into different risk classes. Each risk class has a color and a maximum wait time for the patient to wait for a doctor. At the end of the evaluation, the nurse gives the patient a wrist with his/her corresponding risk color. Third, the patients wait in the General Waiting Room for a doctor to call him/her. Finally, after the meeting with the doctor, depending on the diagnosis, he

patient can be sent to the Medication Room, to some of the Exams Rooms (X-Ray or Tomography) for further investigation or even sent back home.

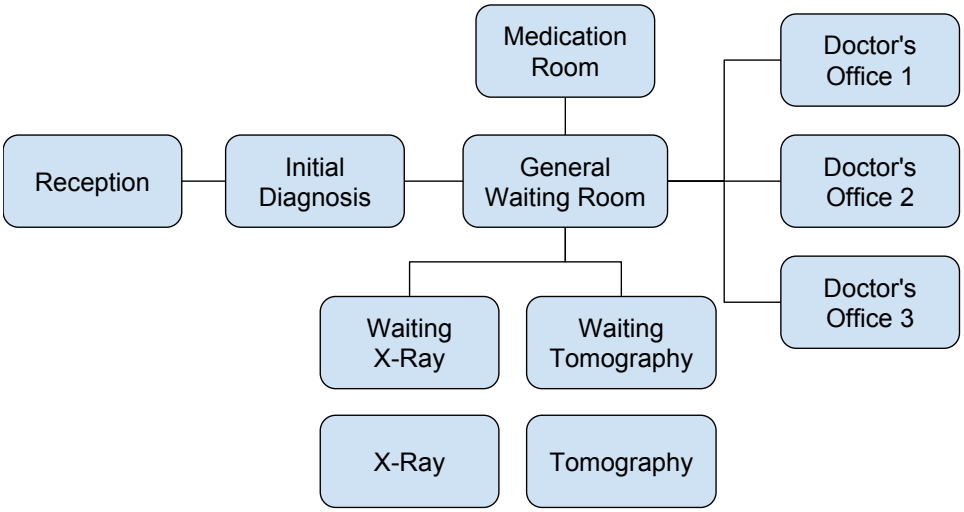

Fig. 5. Organization of the rooms of the hospital.

4.3. *Assumptions*

For the sake of this example and to better understand our proposed two-phase reasoning model, our goal here is to identify bottlenecks with the following characteristics:

A General Waiting Room (GWR) became crowded, in other words, it has more patients than seats.
B Doctor Ferreira, who is responsible for Doctors Office 1, is exceeding the average time to attend her patients.
C Doctor Silva did not came to work, so her office (Doctors Office 2) is empty.

Under these circumstances, to infer this complex event, let us summarize and make some assumptions:

- Each patient receives a BLE device attached to his/her wrist when they finish initial diagnosis and it is associated with the wrist color.
- Each employee has a smartphone with Bluetooth technology and is responsible for only one hospital room.
- Each room has one beaglebone device attached to a wall running our mobile middleware (Mobile Hub).

- Each patients wrist device and employees smartphone can only be connected to only one beaglebone. This is to simplify the problem of determining in which room the patient or staff is.
- Each patient is waiting on the correct corresponding waiting room. For instance, if a patient is waiting for a doctor to see him/her, the patient is waiting at the General Waiting Room.
- Every chair in the waiting room also has a BLE sensor, this will be useful to detect whenever a room is crowded or not.

4.4. *CEP Strategy*

With these assumptions, we can develop a CEP strategy to identify our example bottleneck mentioned above. The CEP strategy (CEP nodes and information flow) is described at Figure 6, each (blue) node representing a CEP operator. Mainly, the CEP strategy is to detect if there is a bottleneck. In particular, to identify the bottlenecks for our example, we need to detect not only when the General Waiting Room is crowded (A), but also, to identify the causes of the bottleneck, calculate the average time that Doctor Ferreira is taking to attend her patients (B) and if Doctor Silva is in the hospital or not (C). First, #1 operator (blue node at Figure 6) receives the patient identification (PID), staff identification (EID) and the room in which they are localized (GWR). As a result, it is possible to infer the RDF triple (EID, treating, PID), which means that the doctor who's smartphone identification is EID is treating the patient that is using the beacon with the identification PID.

The treating relation can be deduced, for instance, because the doctor and the patient are in the same room more than 10 minutes. Furthermore, #2 operator is now able to calculate during how much time EID doctor has treated PID patient and can generate the RDF triple (PID, treated_in, time_value). Finally, #3 operator is able to compute the average time used by EID doctor for treating his/her patients and this information can be used to deduce if the doctor is exceeding the time to attend his/her patients. The time limit to attend a patient depends on the hospital, so it is better to represent this information in the knowledge base.

The detection when the GWR is crowded is managed by #4 and #6 operators. #4 operator is responsible for detecting every patient that enters the GWR and calculate the total of patients. From the information about total number of patients, #6 operator can decide if the room is overcrowded, based on the total number of chairs.

Finally, when the RDF triple (GWR, is_crowded, true) is generated, #7 operator detects if a bottleneck exists and sends all the RDF triples produced to the knowledge base module. The knowledge base module, represented at Figure 6 as KB Module, is a component that is responsible to store the RDF triples and to execute the reasoning process.

Now, to give a more detailed example of how these operators are implemented, Figure 7 is an example implementation of #3 and #5 operators using ESPER

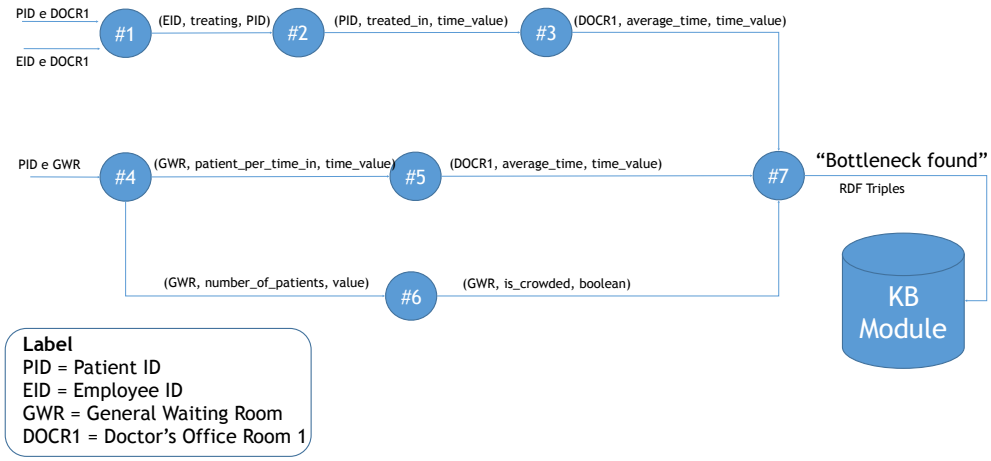

Fig. 6. CEP nodes and information flow.

[12]. ESPER is an engine developed to address the requirements of applications that analyze and react to events, commonly used to implement CEP. The query represented at Figure 7 will retrieve attending times by patients during last one hour and compute the average attending time.

Fig. 7. Implementation of #3 and #5 CEP operators using ESPER.

4.5. *Contextualized Ontologies to use*

The knowledge base (KB) is represented using contextualized ontologies. First we list all the ontologies created for this example:

- People_information: Contains all the names and smartphone numbers of the hospital staff.
- Healthcare_Staff: The complete list of healthcare roles and specializations that the hospital are interested.
- Schedule_Staff: Which room each staff member is responsible for and which day they work.

- Hospital_Rooms: Which rooms the hospital have and how they are interconnected in terms of flow of patients.
- Hospital_Protocols: Maximum time for a doctor to attend a patient.

Second, the contextualized ontologies used in this example are:

(1) People_Information (entity) → Schedule_Staff (context)
(2) People_Information (entity) → Hospital_Rooms (context)
(3) People_Information (entity) → Hospital_Protocols (context)

Finally, to deduce that Doctor Silva did not came to work, we need information from the Schedule_Staff ontology and from the Hospital_Rooms ontology. With the Schedule_Staff ontology it is possible to determine if Doctor Silva is supposed to be at work and with the Hospital_Rooms ontology we can check if the General Waiting Room is the waiting room associated to her office. Therefore, we need to align both Schedule_Staff and Hospital_Rooms ontologies by using the entity People_Information ontology. Figure 8 represents this alignment. Within the contextualized ontologies model, we call this operation context integration.

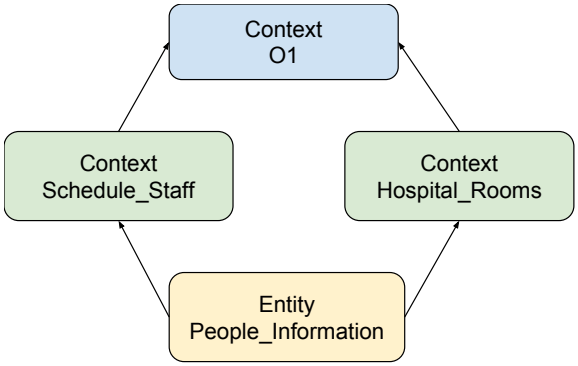

Fig. 8. Context integration between Schedule_Staff and Hospital_Rooms ontologies mediated by entity People_Information.

Figure 9 gives a more detailed information about the context integration between the Schedule_Staff and Hospital_Rooms ontologies. The instances Doctor_Ferreira and Doctor_Silva are linked between the entity and context ontologies, therefore the result ontology (O1) will be the alignment between contexts and the linked instances at both contexts will collapse into a single one. The instance Doctor_Silva at the Shedule_Staff ontology will collapse with the instance Doctor_Silva at the Hospital_Rooms ontology, resulting in a new instance that will have information from both context ontologies.

O1 context (see Figure 10) will have both information about whether Doctor Silva should be working and if her office is associated with the General Waiting

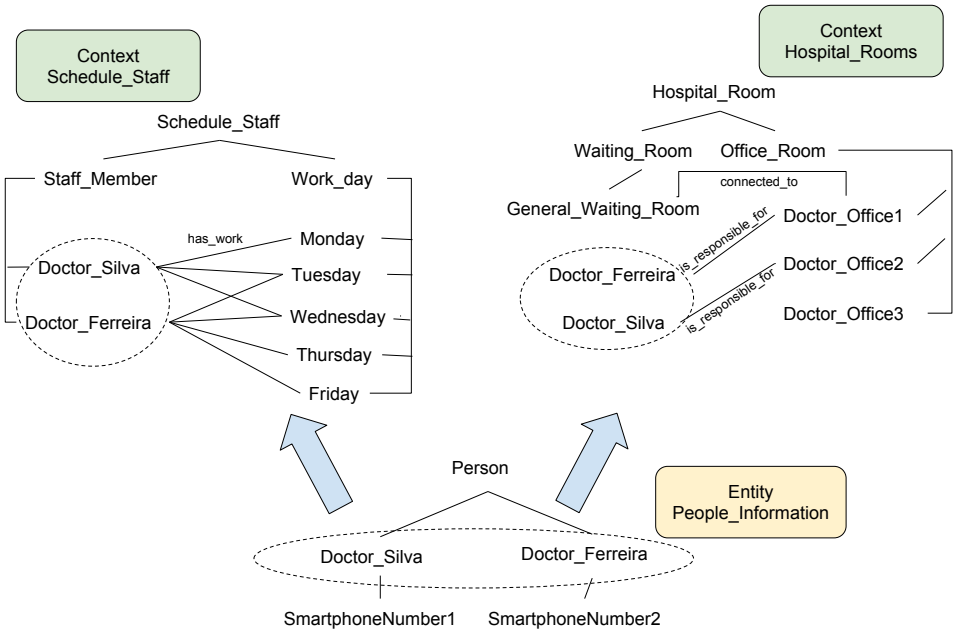

Fig. 9. Detailed information about the alignment between Schedule_Staff and Hospital_Rooms ontologies.

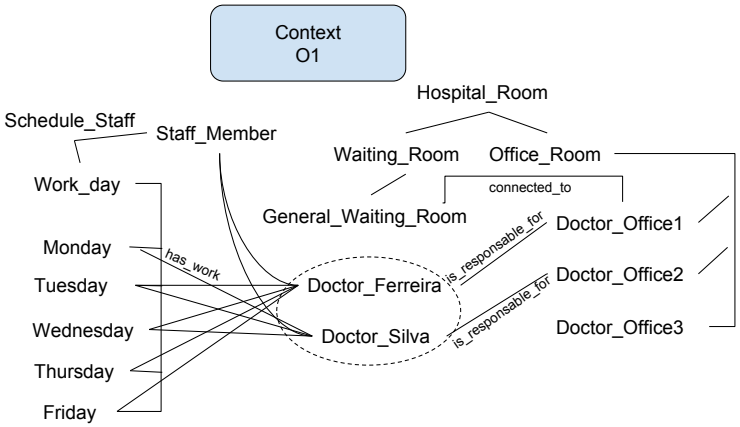

Fig. 10. Resulting ontology from the context integration between Schedule_Staff and Hospital_Rooms ontologies.

Room. So, if we add the RDF triple that came from the CEP layer, that says if Doctor Silva is at her room, it is possible for the reasoner to deduce that (C) "Doctor Silva did not came to work, so her office (Doctor's Office 2) is empty" is one of the

possible causes of the bottleneck.

Furthermore, O1 context also has information about who is responsible for the Doctor's Office 1. If we make the context integration between O1 and the Hospital_Protocol ontology, that contains the maximum time that the hospital allows the doctor to attend a patient, we will generate ontology O2 (see Figure 11). As a result, with O2 the reasoner can deduce (B) "Doctor Ferreira, who is responsible for Doctor's Office 1, is exceeding the average time to attend her patients" as another possible cause of the bottleneck.

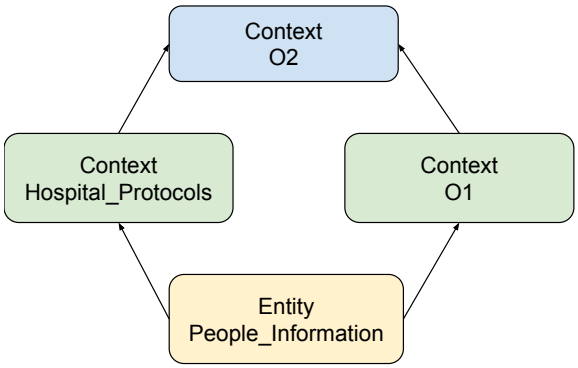

Fig. 11. Resulting ontology from the context integration between O1 and Hospital_Protocols ontologies.

To summarize, by using contextualized ontologies we can reason with a partition of the knowledge base. It is important to highlight that all the alignments used by the context integration operations are executed only once, so the application did not need to execute alignments every time a new fact have to be deduced.

5. Related Work

In an early work, Adi *et al.* [1] presents abstractions that describe semantic relationships between events, object and tasks. These are defined as generalizations and associations and through attributes that may reference events. Their abstractions are suitable for specification but cannot be computed efficiently. On the other hand, the work [3] describes a system (ETALIS) that can perform reasoning over streaming events with respect to background knowledge, similar to our Knowledge Base. It implements two languages for specification of event patterns: the rule based ETALIS Language for Events (ELA), and Event Processing SPARQL. ETALIS can evaluate domain knowledge on-the-fly, thereby proving semantic relations among events and reasoning about them. Their semantic relations among events are time-based, but don't have the synchronicity requirement. Another difference is that they do not generate a RDF Stream which they check against a knowledge base. Thus, their

inference is much simpler than the one proposed in our project.

Tachmazidis *et al.* [24] propose a reasoning method over RDF triples based on defeasible logic (i.e. a non-monotonic logic) which can be implemented in a massively parallel way. They used Hadoop, an open-source implementation of the MapReduce paradigm, and a stratified rule set for a more efficient processing of the knowledge base. Unlike our proposal, they do not handle Stream Processing and do not apply their method to reasoning for time-critical systems, such as CPS. Moreover, their choice for defeasible logic limits the sorts of knowledge that can be inferred by their system, as opposed to temporal logic, which shall have highly parallelizable implementations.

The following projects CityPulse [16] Star-City [18] and FIESTA-IoT [2] also present research toward the use of Semantic Stream reasoning. All of these projects use the knowledge base in order to deduce new context/facts. Also, they use a single-layer (or flat) ontology model, which differs from our ontology model that is organized as nested contexts. Moreover, none of these projects focus on the problem of delivering real-time reasoning.

The FIESTA-IoT project [2] integrates several other projects and one of them is the CityPulse project [16]. The main goal of these projects is to achieve semantic interoperability at different levels (hardware, data, model, query, reasoning and application levels). The StarCity project has a similar idea, but it is aimed at using semantics to provide interoperability at the data level.

On the other hand, the work by Teymourian *et al.* [25] has the same focus as our work. They use a similar idea and combine the use of SCEP rules (semantic web plus CEP) with a semantic knowledge base to deliver real-time reasoning. The difference is that our work uses an ontology model organized as nested context to represent context information, rather then a flat ontology model. As a result, it is more efficient on query processing, because when we execute a query, the query will be processed only using a sub-set of the knowledge base (a partition of the knowledge base). Furthermore, another difference is that we plan to insert new SCEP rules on-the-fly, based on new facts generated by the reasoning over the knowledge base. Consequently, it will give the application a more efficient approach to adapt to different situations. For example, in a monitoring application, we only need a CEP rule that triggers an action based on an altitude situation only if the monitored person is in a high altitude, until then this rule does not need to be there.

6. Discussion

Combining symbolic reasoning based on ontologies with Complex Event Processing has several advantages. Firstly, it allows to leverage CEP's efficient processing of dense flows of simple events, not just over raw sensor events but also over RDF triples. Secondly, CEP's ability to produce complex events is also necessary for the iterative generation of higher level information from lower level bits of information.

On the other hand, while CEP is appropriate for processing data that is carried

by the incoming events, it is incapable of detecting domain-specific relationships between events that are produced by distinct entities/objects that apparently have no relation with each other, or when this relationship cannot be directly encoded by the (meta-)information carried by the events. Symbolic reasoning using ontologies, on the other hand, can very well model these "indirect" relationships among the monitored entities and/or their corresponding events. And hence, by using the results of a query over a domain-specific ontology during a CEP-based continuous processing, it becomes possible to generate new sorts of events (i.e., fact events), which are produced independently by the Semantic Event reasoners in response to the consumption of some RDF triples. These Fact events, which in some sense embody some semantic knowledge that was forked off the knowledge base, can in turn be further processed by other CEP engines, and may be used to predict events that actually did not yet happen, but which are a natural consequence of initial events that have been detected by CEP.

This makes us consider the Semantic Web reasoners as a special kind of CEP engines, which have access to the knowledge base, consume RDF events and eventually produce fact events that are passed on to other CEP engines in the Event Processing Network. (See Figure 2).

7. Towards Automated Rules and Patterns Induction

In this section, we briefly discuss prospects for using machine learning and induction techniques to extract useful information from the data stream. At first, let us mention that although there are known and proven techniques for extracting information from data, from raw data to structured data and knowledge, most of them have been designed as off-line techniques and with the assumption of all data present in the working memory. Therefore, there is a great challenge in adapting when possible current techniques to on-line stream data processing with huge volumes of data or designing new techniques. A good review of the issues (continuous data streams flow, unbounded memory requirements, mining changes, avoiding overfitting...) can be found in [13]. Another good analysis could be found is [23].

Let us start with the raw data produced by the sensors. Unsupervised algorithms, such as *sparse autoencoders*, may be used to automatically extract higher level features [17]. The basic idea is to use an *autoencoder*, a neural network with a hidden encoding layer and a decoding output layer identical to the input layer. We add an additional *sparsity* activation constraint, in order to enforce specialization of each neuron as a specific feature detector. Training an autoencoder, sometimes called *self-supervised*, relies on traditional supervised learning on learning the identity, as the autoencoder learns to reconstruct its input data on its output. Once trained, to extract features from an input, one just needs to feed forward the input data and gather the activations. One may use successive levels (stacks) of autoencoders in order to extract more abstract features. Although standard training is off-line, one may make use it incrementally, with successive rounds of batch train-

ing.

An interesting end to end approach has been proposed by Ganz *et al.* in [14]. The first step, named SensorSAX (as for Sensor Symbolic Aggregate Aproximation), is the discretization of data into qualitative attributes, encoded in some alphabet words. Then a *clusterization* step is applied, using a *k*-means non supervised clusterization algorithm, by considering time as one of the criteria, to form patterns, which are *proto-concepts* (not yet named concepts). *Temporal relations* between these proto-concepts are extracted by constructing a *Markov model*, a statistical predictive model of temporal occurrences of proto-concepts. Three kinds of temporal relations are considered: *occursAfter*, *occursBefore* and *occursSame*. The last step consists in *manual* labeling, i.e. naming proto-concepts into symbolic *concepts* (e.g., "coldTemperature"). The authors are also considering the possibility of *automatic* labeling, derived from the labels of the sensors and a common sense ontology.

When starting from RDF triples, one may consider various knowledge extraction methods based on ontologies (mostly based on OWL), designed for the Semantic Web. One objective is induction, to be able to construct more abstract knowledge (concepts/hypotheses) from the facts. Various algorithms exist and aim at both generalizing examples into concepts, while specializing them in order to uncover counter examples. *Inductive Logic Programming (ILP)* is a seminal formalism but there are many variant (see for instance the DL-Learner framework [8] which includes various ones), as well as related techniques like *decision trees* construction and also exploratory approaches based on *genetic algorithms*.

An interesting proposal in [7] offers *inductive reasoning* as well as *deductive reasoning* on RDF data streams. Deductive reasoning is performed on queries constrained by concepts expressed in OWL. C-SPARQL [6] is the query language used. It is an extension for continuous queries on RDF streams of the SPARQL RDF query language. Inductive reasoning is performed on a subset of data in order to be practically computable. The user defines *statistical units* (entities, e.g., persons) as well as a *population* of these entities (e.g., at a specific institution or location) on which he wishes to make inductive queries. The inductive engine periodically updates *data matrices* representing the *features* of the population of the statistical units considered (actually, there are two kinds of matrices, one *long term* stable and one *short term* representing the trends) and conducts a *multivariate analysis* of theses matrices. The trained model could then be used to predict *relationships* between entities at query time.

Last, it is also important to be able to extract *temporal relations*. A promising proposal is by Georgala *et al.* [15] to efficiently extract all possible temporal relations (along seminal Allen's taxonomy and algebra of temporal intervals) from time stamped RDF streams.

In summary, we could see that there are various interesting directions for introducing automated machine learning and knowledge extraction techniques into our framework. One important issue is the *dynamicity* of the data produced. That is, because of the continuous stream of data, we need to find good *trade-offs* between:

the demand for higher level knowledge, the cost for extracting it (processing cost as well as memory cost/limitation) and the risk of it being obsolete, depending on: the usage, the nature of the data and the computing & communicating resources available. For instance, in the scope of our two stage process, we believe that machine learning could be effective as a way to consolidate into the knowledge base the facts which occur very frequently (see Figure 3). An example of such learning (in that case, inductive) is the identification of two temperature settings, inside a air-conditioned bus and outside, that will be extracted from repeated facts of passengers entering and exiting air-conditioned buses. Therefore, one needs to carefully examine what exact machine learning techniques we will insert into our framework, and at which stage.

8. Conclusion and Future Work

This paper presented a real-time reasoning approach based on semantic events and fact streams for IoT systems. The reasoning approach is based on the assumptions that all objects, people, buildings, places, vehicles, environments, etc. will have many embedded tiny sensors that will emit simple events whenever some action is performed with/to it by an actor, and that each event will carry the items' unique UUID and an accurate time-stamp. By enforcing the restriction that predicates in a RDF triple must be action-based, such as "kick", "put", "grab", etc., rather than state-based, such as "has", "is", "belongs to", etc., we are of course limiting the amount of information that the data/event streams are capable to express. However, we believe that the action-based predicates are the really important ones for reasoning in IoT applications. All the state predicates, on the other hand, should instead be represented by the nested context-based ontology in the Knowledge Base. We are aware that this is only a first and initial step towards adding semantics to real-time reasoning over data streams, and that much more theoretical and practical research is required to validate our approach, evaluate it under a broader perspective and show its feasibility for large-scale and distributed IoT applications. However, we are confident that it is a promising first step. As next steps, we will finish the development of the Context Mappers and Semantic Event Rules using ESPER EPL (Event Processing Language) and deploy them on our mobile IoT middleware. In parallel, we will finalize the implementation of the Hospital 4.0 scenario (described in Section 4), and start to test it.

Acknowledgements Our ESMOCYP (Efficient Semantic MOdels and Fault-tolerant Middleware for CYber-Physical Systems) Project is supported by PROBRAL CAPES–DAAD Brazil-Germany cooperation program (Process No 8148/2015-05) and by a CAPES PVE fellowship to J.-P. Briot.

References

[1] A. Adi, D. Botzera, and O. Etzion. Semantic Event Model and its Implication on Situation Detection. In *European Conference on Information Systems (ECIS 2000)*, 2000.

[2] Y. Al-Hazmi and T. Magedanz. Towards semantic monitoring data collection and representation in federated infrastructures. In *IEEE International Conference on Future Internet of Things and Cloud (FICloud)*, pages 17–24, 2015.

[3] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream Reasoning and Complex Event Processing in ETALIS. *Semantic Web*, (1):1–5, 2009.

[4] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: Semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, June 2006.

[5] G. Baptista, F. Carvalho, S. Colcher, and M. Endler. A middleware for data-centric and dynamic distributed complex event processing for iot real-time analytics in the cloud. In *34th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC'2016)*, Salvador, Brazil, June 2016.

[6] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-SPARQL: SPARQL for continuous querying. In *18th International World Wide Web Conference (WWW'09)*, pages 1061–1062, 2009.

[7] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, Y. Huang, V. Tresp, A. Rettinger, and H. Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 25:32–41, November–December 2010.

[8] L. Bühmann, J. Lehmann, and P. Westphal. DL-Learner – A framework for inductive learning on the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 39(C):15–24, August 2016.

[9] I. Cafezeiro, J. Viterbo, A. Rademaker, E. Haeusler, and M. Endler. Specifying ubiquitous systems through the algebra of contextualized ontologies. *The Knowledge Engineering Review*, 29(02):171–185, 2014.

[10] K. S. Candan, H. Liu, and R. Suvarna. Resource description framework: Metadata and its applications. *SIGKDD Explor. Newsl.*, 3(1):6–19, July 2001.

[11] M. Endler, G. Baptista, L. D. Silva, R. Vasconcelos, M. Malcher, V. Pantoja, V. Pinheiro, and J. Viterbo. Contextnet: Context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. In *ACM/IFIP/USENIX 12th International Middleware Conference Workshop on Posters and Demos Track*, pages 2:1–2:2. ACM, 2011.

[12] EsperTech. Esper, Accessed on 20/04/2017. http://www.espertech.com/esper/.

[13] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *SIGMOD Record*, 34(2):18–26, June 2005.

[14] F. Ganz, P. Barnaghi, and F. Carrez. Automated semantic knowledge acquisition from sensor data. *IEEE Systems Journal*, 10(3):1214–1225, September 2016.

[15] K. Georgala, M. Sherif, and A.-C. Ngonga Ngomo. An efficient approach for the generation of Allen relations. In *22nd European Conference on Artificial Intelligence (ECAI'2016)*, Den Haag, The Netherlands, August–September 2016.

[16] M. Giatsoglou, D. Chatzakou, V. Gkatziaki, A. Vakali, and L. Anthopoulos. CityPulse: A platform prototype for smart city social data mining. *Journal of the Knowledge Economy*, 7(2):344–372, 2016.

[17] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *29th International Conference on Machine Learning*, Edinburgh, U.K., 2012.

[18] F. Lécué, S. Tallevi-Diotallevi, J. Hayes, R. Tucker, V. Bicer, M. L. Sbodio, and P. Tommasi. Star-City: semantic traffic analytics and reasoning for city. In *19th in-*

ternational conference on Intelligent User Interfaces, pages 179–188. ACM, 2014.

[19] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[20] N. Fridman Noy and C. D. Hafner. The state of the art in ontology design – A survey and comparative review. *AI Magazine*, 18(3), 1997.

[21] L. Talavera Rios, M. Endler, and S. Colcher. An energy-aware IoT gateway with continuous processing of sensor data. In *34th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC'2016)*, Salvador, Brazil, June 2016.

[22] L. Talavera Rios, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. Silva e Silva. The mobile hub concept: Enabling applications for the internet of mobile things. In *12th IEEE Workshop on Managing Ubiquitous Communications and Services (MUCS 2015)*, pages 123–128, St. Louis, MI, USA, March 2015.

[23] X. Su, E. Gilman, P. Wetz, J. Riekki, Y. Zuo, and T. Leppänen. Stream reasoning for the Internet of Things: Challenges and gap analysis. In *6th International Conference on Web Intelligence, Mining and Semantics (WIMS'16)*, Nîmes, France, June 2016.

[24] I. Tachmazidis, G. Antoniou, G. Flouris, S. Kotoulas, and L. McCluskey. Large-scale parallel stratified defeasible reasoning. In *ECAI-12*, 2012.

[25] K. Teymourian, M. Rohde, and A. Paschke. Fusion of background knowledge and streams of events. In *6th ACM International Conference on Distributed Event-Based Systems*, DEBS'12, pages 302–313. ACM, 2012.