Knowledge-Based Systems 180 (2019) 133–146

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

PERCEPTUS: Predictive complex event processing and reasoning for IoT-enabled supply chain☆

Falak Nawaz [a,*], Naeem Khalid Janjua [b], Omar Khadeer Hussain [a]

[a] School of Business, University of New South Wales (UNSW), Canberra, Australia
[b] School of Science, Edith Cowan University (ECU), Perth, Australia

ARTICLE INFO

Article history:
Received 2 January 2019
Received in revised form 2 May 2019
Accepted 15 May 2019
Available online 17 May 2019

Keywords:
IoT
Supply chain
Complex event processing
Predictive reasoning

ABSTRACT

Internet of Things (IoT) is an emerging paradigm that connects various physical sensor devices spread across different locations. IoT-enabled supply chain provides a natural combination to achieve supply chain visibility (SCV) which refers to ability of supply chain partners to collect and analyse distributed supply chain data for the planning and decision support. This data is normally collected and analysed in real-time by a specialized software known as Complex Event Processing (CEP) engines. However, current CEP engines have two well-known limitations. Firstly, current CEP engines are job specific and fail to combine multiple related sensor data streams coming from distributed sources, thereby, supply chain partners are exposed to manage the underlying information heterogeneity. Secondly, these CEP systems do not provide decision support to the supply chain planners when the information about the potential disruptive event is incomplete and/or uncertain. In this paper, a PERCEPTUS framework is proposed to address the above mentioned issues. It, firstly, utilizes semantic annotation process to integrate and annotate events coming from heterogeneous data streams. Secondly, it performs complex event processing to process and correctly interpret annotated complex events. Thirdly, it provides complex event reasoning (by combining logical and probabilistic reasoning) to predict disruption events (such as process failure) under incomplete and/or uncertain information. Finally, the proposed framework is validated using the dataset of a semi-conductor manufacturing process to demonstrate its superiority in terms of accuracy in predicting disruptive events as compared to the baseline approach.

1. Introduction

Supply chain activities are characterized by increasing dynamicity which arises from the trends of the global economy, political situations, distribution of transport services and individual customer demands. Moreover, the continuous drive towards lean business processes have resulted in supply chain becoming more complex and vulnerable to operational disruptions. This led organizations to be aware of the need to have better supply chain visibility (SCV), which is a commonly used term in supply chain management yet not very well defined in the literature [1]. In literature [2], SCV is defined as "the identity, location and status of entities transiting the supply chain, captured in timely messages about events, along with the planned and actual dates/times for these events". However, from an Information Technology (IT) perspective, SCV refers to the ability of a supply chain partner to collect and analyse distributed data, analyse and generate specific recommendations, and match insights to the strategy [3]. The increasing use of IT in supply chain has enabled suppliers, manufacturers, logistics service providers and retailers to benefit from SCV by making their supply chain data available to their partners [4]. While the availability of supply chain data provides the impression of SCV, it also adds to a company's challenge if it is not used effectively to generate insights [1]. Moreover, there is another limitation in achieving visibility for organizations despite the availability of data. It has been reported that nearly 90% of all companies have inadequate global supply chain technology to provide with timely information required for planning and management [5]. Hence, the partial or complete lack of visibility resulting from the inadequate technology is the main hurdle for realization of the global supply chain.

Internet of Things (IoT) is an emerging paradigm that is focussed on connecting intelligent physical devices (e.g. sensors) spread across different locations. These devices can sense real-time information from the surroundings and can share this information with other devices. It is estimated that, by the end of year

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.05.024.

* Corresponding author.
E-mail addresses: falak.nawaz@student.adfa.edu.au (F. Nawaz), n.janjua@ecu.edu.au (N.K. Janjua), o.hussain@adfa.edu.au (O.K. Hussain).

https://doi.org/10.1016/j.knosys.2019.05.024
0950-7051/

2020, approximately 24 billion devices will be interconnected throughout the world and is expected to produce big amount of data [6]. This technology is giving rise to the concept of IoT-enabled supply chain that takes real-time information from IoT devices installed at different locations within the supply chain, processes this information through the software systems, and shares useful insights with supply chain partners [7]. However, existing software systems are not designed to process big data that is generated with IoT devices. To process this big data, specialized software known as Complex Event Processing (CEP) engines [8–10] are used. The CEP engines infer complex events (e.g. disruptions) and alarms the decision makers about future events for better management of supply chain [11]. However, unlocking the potential of CEP in supply chain is hindered by two main factors. Firstly, current CEP engines are developed for a single job and they fail to combine multiple related data streams coming from distributed sources, thereby, supply chain trading partners are exposed to manage the underlying information heterogeneity [12,13]. Secondly, these CEP systems can only detect existing disruptive events from a data stream. They do not provide predictive and proactive decision support to the supply chain planners when the information about the disruptive event is incomplete and/or uncertain [14–16].

Owing to the above problems, this study aims to address the challenges faced by IoT-enabled supply chains and proposes a PERCEPTUS framework for complex event reasoning. It has the ability to integrate, process and correctly interpret complex events in heterogeneous data streams coming from different supply chain partners using semantic annotation and complex event processing. It then performs complex event reasoning to predict disruptive events using logical and probabilistic reasoning engine. The proposed framework is able to provide decision support to manage the identified disruptive event even when the underlying information about the event is incomplete and/or uncertain. Specifically, the main contributions of our work are as follows:

1. An ontology-based *semantic annotation process* is utilized that integrates and annotates events coming from heterogeneous data streams. This is explained in Section 4.
2. An novel approach for *semantic complex event processing* is proposed to process and correctly interpret semantically annotated complex events. This is explained in Section 5.
3. A *logical and probabilistic reasoning engine* is proposed that performs predictive and proactive reasoning even when the underlying information is incomplete and/or uncertain. This is discussed in Section 6.

The rest of the paper is organized as follows. In Section 2, the existing research in CEP systems and related fields is discussed. The proposed PERCEPTUS framework is presented in Section 3. In Sections 4 and 5, an ontology-based semantic annotation process and semantic complex event processing is presented respectively. In Section 6, the logical and probabilistic predictive reasoning engine for incomplete and/or uncertain information is described. Section 7 presents performance evaluation and the prediction quality of the proposed approach. In this section, the superiority of the proposed approach is validated by comparing to a baseline approach for predicting disruptive events in supply chain using a semi-conductor manufacturing process dataset. Finally, a conclusion is presented in Section 8.

2. Related work

2.1. Complex event processing

In this section, we outline the existing research in CEP systems and related field. In recent years, CEP systems have received a great deal of attention in a variety of domains such as business process monitoring [17], financial services [18], healthcare [19], cyber security [20], and traffic management [21] to name a few. Flouris et al. [11] discussed different CEP techniques covering both deterministic and probabilistic event models and spanning from centralized to distributed network settings. Authors reviewed current status and issues of CEP techniques arising due to use of CEP techniques over big data and cloud platforms. Moreover, a blend of CEP technologies were also presented with emphasis on predictive analytics, scalability and elasticity.

The current state of the art CEP engines [8–10] are designed to deal with the specific problem of defining new (higher level) events starting from primitive events (raw data). However, they do not provide support to integrate events coming from heterogeneous sources. As a result, the complex events coded and detected by the current CEP engines are not expressive enough to capture and reason complex situations. For instance, in situation awareness, time critical actions are not only triggered by an individual event, rather events coming from different sources are integrated through additional contextual knowledge to infer new pieces of knowledge to detect and classify a situation of interest. To overcome the issue of information heterogeneity, CEP researchers need to step back and learn from another stream of research in Artificial Intelligence known as the Semantic Web [22]. The Semantic Web research focuses on how "meaning" can be attached to the data so that data can be understood, shared, reasoned and integrated by machines without human intervention. Use of semantics in CEP means annotating sensor data streams with contextual knowledge, which serves as a metadata to link distributed data streams. Thereby, CEP engine can perform reasoning with various annotated sensor data streams to deduce new or implicit knowledge, discover significant (and erroneous) events/situations and answer complex queries.

Recently, ETALIS [10], which is a CEP engine, has been developed using semantic technologies, however, it suffers from two major limitations: Firstly, it works under the assumption that events knowledge is always consistent. In other words, it is assumed that there will be no conflicting events or situations during the events integration and reasoning process. And if new events information comes into the system, it will be consistent with the already available information. Additionally, new information does not lead to the retraction of previous events/actions. Secondly, it does not support complex actions representation and predictive reasoning. A production rule system for activity recognition and monitoring tasks in smart spaces is presented in [23]. It is based on a hybrid framework consisting of logical and probabilistic reasoning. Our proposed approach differs from [23] in the sense that rather than using the reactive approach, we use the predictive and proactive approach by using the logical reasoning to predict disruptive events and using the probabilistic inference to handle the uncertain/unknown situations.

2.2. Predictive business process monitoring

Predictive business process monitoring under defined performance constraints is a key issue in many organizations. Predictive business process monitoring approaches forecast potential problems during process execution and handle them even before they occur using machine learning and data mining techniques. These approaches are also applicable to supply chain in predicting disruptions and critical situations. These approaches, if combined with CEP technology, can potentially enhance the performance of predicting events of interest in business process or supply chain.

There are different approaches for runtime monitoring of business process management that has been proposed in the literature [17,24–28]. In [28], three different prediction techniques for

business process monitoring are compared and analysed empirically. The compared approaches are based on machine learning, constraint satisfaction and Quality of Service (QoS) aggregation. In another work [29], authors proposed two approaches, a sequential k-nearest neighbour and an extension of Markov model for predictive business process monitoring. These approaches exploit temporal and sequential features of the business process data with an additional sequence alignment component. Cuzzocrea et al. [30] discussed the problem of predictive monitoring when performance constraints are defined in aggregated process instead of single process. This work integrated two predictive models: a clustering-based predictor for estimating the outcome of each ongoing process instance and a time series predictor for estimating the performance outcome of the future process instances.

2.3. IoT-enabled supply chain

With the proliferation of sensor networks and IoT devices, the integration of IoT and supply chain provides a promising way to establish innovative systems for monitoring supply chain activities. Different types of devices are used in various supply chain activities to monitor performance indicators. For example, radio-frequency identification (RFID) devices are used in items tracking and inventory management. In this regard, an approach is proposed in [31] to explore the effectiveness of different types of RFID in reducing the inventory inaccuracies in supply chain. Author's in this work based their approach on the fact that different types of RFID has unequal techniques in tracing items and, therefore, presented a scale factor function to illustrate the relationship between cost of RFID and it abilities in decreasing inventory inaccuracies. This scale factor function is then used to formulate analytical models to investigate the cost-effectiveness of different RFIDs in different scenarios.

In the context of centralized supply chain where orders from online customers are delivered independently, a new delivery strategy based on the synthetically dispatched orders is proposed in [32] to decrease the outbound delivery cost. It utilizes orders and customers matching and customer visiting sequence with beta-heuristic algorithms. Similarly, in another work [33], an RFID-based investment evaluation model is proposed for joint replenishment and delivery (JRD) problem under stochastic demand. In [34], a joint replenishment problem is modelled that considers different types of discounts such as two quantity discounts, all-unit quantity discount, and incremental discount on e-commerce platform. Author's in this work proposed a novel locust swarms algorithm to the joint replenishment problem.

There are a few studies that utilize event-driven monitoring and/or prediction and are also relevant to the supply chain environment [35–38]. For instance, in [38], SLA violations in cloud of things (CoT) environment is addressed by extracting and identifying the impact of the relevant external events on SLA. The external events are usually outside the control of service providers and users but they may impact the quality of CoT services and may result in SLA violation. Authors' presented an approach to extract external events from social media streams using natural language processing and machine learning techniques.

Nevertheless, the problems of the current CEP engines still exist as they fail to combine multiple related sensor data streams coming from distributed sources. They also fail to provide predictive and proactive decision support to the supply chain planners particularly when the information about the disruptive event is incomplete and/or uncertain. In the next section, our proposed framework is presented that aims to address these issues.

3. Proposed framework

In this section, a framework for Predictive and proactive Complex Event Processing and reasoning in IoT-enabled Supply chain (PerCEPTuS) is presented. In the proposed framework, our aim is to shift the notion of reactive computing to proactive computing by predicting future events of interest based on the current event. As a first step, the structure and relationship of different information sources (i.e. sensor data streams) in the supply chain activity are required before the complex events are identified. This is specified using the *annotation ontology* (as depicted in Fig. 2). Then, to determine the complex patterns from the incoming sensor data streams, the *performance constraints* of the process in a supply chain activity are also required (discussed in Section 5). Then the observation and collection of information about the environment using IoT sensor devices begins and the events that match complex patterns are identified. Finally, the new knowledge is inferred from the identified events to discover significant events of interest (e.g. failure of a business process). Fig. 1 depicts the architecture of the novel framework for complex event reasoning engine. As stated before, two prerequisites, namely the *annotation ontology* for sensor data streams and *performance constraints* of business process (i.e. supply chain activity constraints) for predictive reasoning, are needed in the knowledge base before processing of the information begins. Once the prerequisites are captured, the information in different components is processed as follows.

1. *Semantic Annotation.* This component is connected to the physical world of IoT having different sensor data streams. Its responsibility is to model and encode the contextual knowledge of the heterogeneous data streams, which is usually extracted from the meta information of the data stream. This knowledge is encoded in the form of ontologies, which plays an essential role to correctly interpret the distributed events information that could otherwise be interpreted in a number of ways by the CEP engine. Once the contextual knowledge is modelled and encoded, it is then used to process and annotate events information coming from heterogeneous data streams.
2. *Semantic Complex Event Processing.* This is an important component of the proposed system that takes the semantically annotated events as input and after processing identifies the complex event or activity that has just happened or is about to happen in the near future. Event calculus is used to model complex events and actions that may change over a period of time and may introduce incomplete and/or uncertain events information in the knowledge base. There are two types of knowledge bases used in the proposed framework. The main Knowledge base (*KB*) contains rules for defining and capturing complex event patterns, representing states of complex events, and performance constraints of the business process. While the *Actions KB* deals with defining all the alternative actions that can be taken to avoid an undesired event that is predicted. It can recommend suitable action to external applications (e.g. IoT applications) based on the prediction of the proposed system. In this paper, the emphasis has been on utilizing the *KB* for defining rules and facts for capturing complex event patterns, state representation of complex events and defining performance constraints of the business process.
3. *Logical and Probabilistic Reasoning Engine.* As described previously that in many cases not only the future events but also the current events may be detected with uncertainty [14,39]. For example, a sensor device may produce

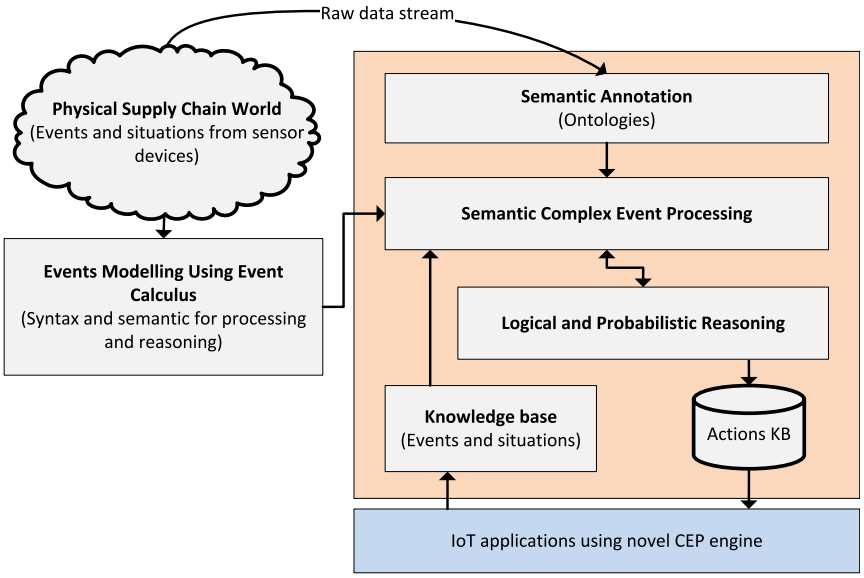

Fig. 1. PERCEPTUS system architecture.

a noisy reading. This means that reasoning engine must be able to handle uncertain input events. Therefore, in this component, a hybrid predictive reasoning is utilized that has the capabilities of both logical and probabilistic reasoning. The logical reasoning is performed on the incoming events by first determining the state of the *EVs* from the incoming data streams and then inferring the failure of the business process. However, when the incoming data stream contains incomplete, inaccurate or missing information about any *EV*, the probabilistic inferencing is used to determine the possible state of that *EV* and then failure of the business process is predicted by the reasoning engine.

In the following sections, we describe important components of the proposed system and their working in detail. The working of the proposed system starts with the semantic annotation of sensor data stream for predictive reasoning. For this purpose, the structure and relationships of different sensor data streams in the supply chain activity or business process are identified. This is explained in the next section.

4. Semantic annotation

The contextual information coming from sensor data streams is not always 100% accurate due to the uncertainty of events and data originating at the source. This contextual information may be of variable quality. For instance, it can be inaccurate, redundant, incomplete, or it can be conflicting with other contextual information [40]. Moreover, various kinds of context have dependencies, causal relationships and temporal characteristics. In this work, we identify two main reasons for different types of uncertainties in the contextual information, which are given as follows:

1. The sensor technology is not capable of producing 100% accurate data due to various technical challenges. For example, hardware malfunction can prevent a sensor to report certain events. A sensor may have the limited accuracy or there may be distortion along a communication channel [41]. Similarly, a hidden object in video monitoring or an unclear voice due to noise signals are examples of imprecise data.
2. The reasoning models are not 100% accurate even when sensors produce near 100% accurate sensor data [42]. They also lack accuracy and do not always provide accurate information about the events and, hence, a probability value is accompanied with such events to address the uncertainty.

Therefore, in this work, ontology-based context model is used that is not just able to represent uncertainty but can also support to model the temporal aspects of the contextual information.

The contextual knowledge is essential to correctly interpret the distributed events. This knowledge could be interpreted in a number of ways if not correctly encoded. This process of contextual modelling and event annotation is known as *semantic annotation*. Currently, most of the CEP systems represent event information as a tuple of values separated by a delimiter. For example, the raw event tuple from an airflow sensor would look like "D105VOL, 2018-05-01T09:30, 510.0, 0.8" (see Fig. 2). In the proposed semantic annotation technique, a codified ontology with UIMA (Unstructured Information Management Architecture) [44] is utilized. The UIMA is a fundamental framework and acts as a backend engine that uses domain ontologies for raw events data annotation.

As a result, semantic relationships are built among events coming from different sources. As shown in Fig. 2, an airflow semantic sensor event is linked with a blood pressure semantic sensor event through a common property i.e. *ee:hasLocation* indicating that both sensors are located in the same office located in building RTH105. Through such semantic relationships between events, reasoning engine can reason on events and can infer new pieces of knowledge and answer complex queries.

5. Semantic complex event processing

In this section, we discuss the basics of semantically annotated events and also describe their representation for the purpose of predictive and proactive complex event reasoning.

5.1. Complex event definition and representation

A semantically annotated event can be classified either as a simple event (*SE*) or complex event (*CE*). Moreover, it may also be a combination of both as shown in a hierarchical structure in

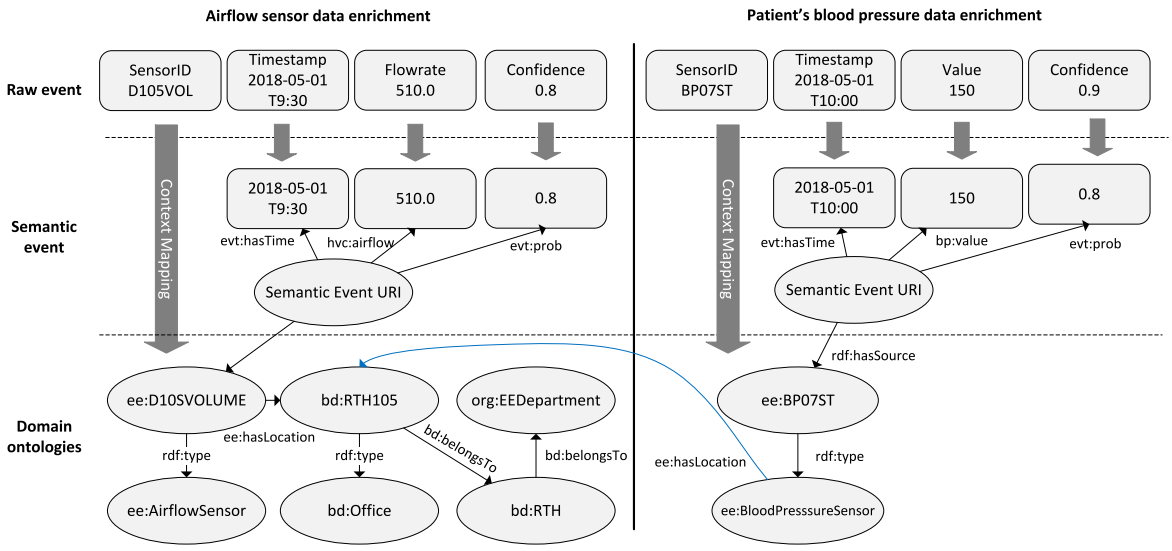

Fig. 2. Semantic Annotation.
Source: Adapted from [43].

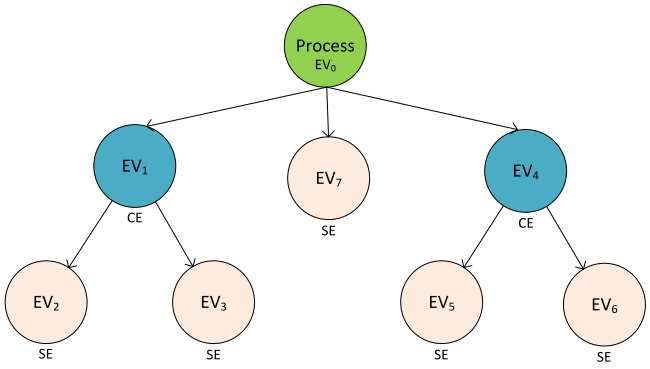

Fig. 3. Different levels of nested relationships between *EVs*.

Fig. 3. It has complete information about its state including its type, time of occurrence and its relationship with other events. In our proposed approach, a semantically annotated event is represented using a vector as (E, R, T, S_t) and is defined as follows:

- E represents the unique event along with all of its attributes such as *event id, type, relationships* etc.
- R represents the pattern (if any) and the information of other events the pattern is applied to.
- T represents the timestamp when the event is first sensed or detected.
- S_t represents the state (of the constraint) of the event. This state shows the compliance or violation of the complex event in business process. The state S_t can be either assigned or derived from the other complex events during the course of execution of business process.

To differentiate between different types of events and the purpose of having state S_t for every event, consider that Fig. 3 represents an activity or a stage in the business process that has a value (or *performance constraint*) which measures its compliance. During the process execution, the chance of the process failure is determined by comparing its current value with the constraints. This run-time comparison is measured for each event value to its defined constraints. In our proposed approach, this run-time comparison is specified using a multi-valued logic which is defined below [14]:

- *Normal (N)* If the value of the event is within its defined constraints, its state is represented as *normal*.
- *Failure (F)* If the value of the event is not within its defined constraints, its state is represented as *failure*.
- *Uncertain/Unknown (U):* If the value of the event can neither be evaluated as *normal* nor as *failure* then it is represented as *uncertain*. This situation can happen when either (a) the event value cannot be measured due to some technical failure of the sensor [45] or (b) the event value is at the boundary of the defined constraints but due to uncertainty in sensor measurement it cannot be accurately defined as *normal* or *failure* [14].

For each event (*SE* or *CE*), the above-mentioned states are determined and then combined (using complex patterns and operators) to determine the failure of a business process or supply chain activity (SCA) as shown in Fig. 3. Different types of events are treated differently in the proposed system, therefore, it is important to classify them based on their properties. These are explained as follows:

- *Simple Event (SE):* An *SE* is a primitive event which is captured by a sensor device in the business process. This event is not dependent on any other event in the system. For example, it can be seen from Fig. 3 that events such as EV_2, EV_3, EV_5, EV_6 and EV_7 are all *SEs*.
- *Complex Event (CE):* A *CE* is an event that results from the combination of two or more *SEs* (or *CEs* or a combination of both) connected through a complex pattern *R*. For example, it can be seen from Fig. 3 that events such as EV_1, EV_4 and EV_0 are *CEs*. Unlike *SEs, CEs* are defined using logical rules and complex patterns. These complex event definitions are described in the next section.
- *Event with a Value/Constraint (EV):* An *EV* is a specialized complex event that marks an important step in a business process. It acts as key performance indicator (KPI) and therefore its value is monitored to identify the compliance or violation of a business process. An *EV* can either be a *CE* or *SE* or any combination of these. For example, it can be seen from Fig. 3 that EV_1, EV_7 and EV_4 combine to ascertain the state of the business process EV_0. In this case, EV_7 is *SE* while EV_1 and EV_4 are *CEs*. Moreover, in this process, any *SE* or *CE* can be marked as *EV* for monitoring its state

Table 1
Complex event pattern detection operators.

Operator	Description
SEQ	The *SEQ* operator identifies a given sequence of events from the input events. For example, "*Peter goes to bed after dinner*" can be represented as *SEQ(dinner, bed)*.
PAR	The *PAR* operator identifies a pattern of two or more events occurring at the same time either overlapping completely or partially. For example, *"Peter eats dinner and reads the newspaper simultaneously"* can be represented as *PAR(dinner, newspaper)* or *PAR(newspaper, dinner)*.
DURING	This complex event is detected when an events occurs during an interval when another event is happening. For example, "During dinner, Peter reads the newspaper" can be represented as *DURING(dinner, newspaper)*.
STARTS	When two events start at the same time. For example, *"Peters starts reading the newspaper as soon as he starts eating the dinner"*. It is represented as *STARTS(dinner, newspaper)* or *STARTS(newspaper, dinner)*.
FINISHES	When two events finish at the same time. For example, *"Peters stops reading the newspaper as soon as he finishes eating the dinner"*. It is represented as *FINISHES(dinner, newspaper)* or *FINISHES(newspaper, dinner)*.
COUNT	Then number of occurrences of an event during a time period. For example, *"The number of calls Peters receives in a day"* can be represented as *COUNT(Peter, received_calls)*.
OR	This complex event is detected when one event occur out of two or more events during a time interval. For example, "*Peter watches either a movie or plays video game at evening*".
AND	The *AND* operator identifies the occurrence of two or more events in any order during a time interval. For example, "*Peter watches television and talks to his friend on the phone in the evening*".
NOT	Any event does not happen or exist at time t. For example, *"an event e does not happen at time t"* can be represented as *NOT(e)*.

during the process execution. Therefore, in order to avoid confusion, hereon in this paper, all of the *SEs* or *CEs* are represented simply as *EVs* unless otherwise stated, as shown in Fig. 3.

5.1.1. Complex event pattern detection operators

In this section, the semantics to capture complex events are explained. This is important when a particular event of interest is anticipated from a set of events which are originated from sensor devices. For example, a complex event pattern can identify a sequence of any two activities in a business process. The complex event operators are generally applied to events originating from sensor devices. In some cases, these operators are also applied to other complex events. The name and the brief description of each of the operators are described in Table 1.

Note that the time duration is not mentioned in all of the event definitions above. The time window is controlled through event calculus ontology (Table 3). Moreover, the detailed description including event calculus rules of all of the above operators is also presented in the next section.

5.1.2. Complex event state composition operators

As described previously that a complex event can be a combination of simple events, complex events, or any combination of both. Moreover, in our model, every complex event (e.g. an *EV*) is assigned a state representing its compliance or violation of the business process. For instance, consider a complex event $COUNT(SEQ(x_1,x_2), \Delta t))$ is an EV_X and its *normal* state is represented by *n* occurrences of $SEQ(x_1,x_2)$ pattern during a time period Δt. Now, if during time period Δt, the number of occurrences of $SEQ(x_1,x_2)$ pattern are found to be either more or less than *n*, the current state of EV_X will be set as *failure*. A complex business process can combine multiple complex events in a nested or hierarchical form. The evaluation of the whole process may require

Table 2
Complex event state composition operators.

EV_X	EV_y	$EV_{CON(x,y)}$	$EV_{DIS(x,y)}$	$EV_{IMP(x,y)}$	$EV_{BIC(x,y)}$	$EV_{NEG(x)}$
N	*N*	*N*	*N*	*N*	*N*	*F*
N	*F*	*F*	*N*	*F*	*F*	*F*
N	*U*	*U*	*N*	*U*	*U*	*F*
F	*N*	*F*	*N*	*N*	*F*	*N*
F	*F*	*F*	*F*	*N*	*N*	*N*
F	*U*	*F*	*U*	*N*	*U*	*N*
U	*N*	*U*	*N*	*N*	*U*	*U*
U	*F*	*F*	*U*	*U*	*U*	*U*
U	*U*	*U*	*U*	*U*	*U*	*U*

Table 3
Event calculus ontology.

Syntax	Description
initially(f)	Fluent f holds from the initial time
initiates(e, f, t)	Event e initiates fluent f at time t
terminates(e, f, t)	Event e terminates fluent f at time t
holds_at(f, t)	Fluent f holds at time t
happens(e, t)	Event e happens at time t
clipped(t1,f,t2)	Fluent f is terminated between time t1 and t2
mvi(f, t1, t2)	Fluent f holds in the interval (t1, t2]

evaluation of each complex event and its relationship with other events. Here, the logic for evaluation of the state of complex events is described when two or more complex events combine using *conjunction (CON), disjunction (DIS), implication (IMP), biconditional (BIC)* and *negation (NEG)* operators. This logic is based on Kleene and Priest logic [46]. In Table 2, $EV_{CON(x,y)}$, $EV_{DIS(x,y)}$, $EV_{IMP(x,y)}$, $EV_{BIC(x,y)}$, and $EV_{NEG(x)}$, representing *conjunction, disjunction, implication, biconditional* and *negation* respectively, of the states of the complex events EV_X and EV_y are shown.

Any of the event definition above can be used to evaluate the state of the complex event. In the next section, complex events semantics are defined using event calculus.

5.2. Complex event semantics in event calculus

The syntax and semantics for the complex event reasoning engine are defined in this section. It has been observed that logic programming has advantages over other formalisms [12]. For example, in the implementation of the CEP concepts, logic programming is more reusable and extensible than procedural programming. Secondly, the rule-based formalism presented in this paper, is expressive and powerful enough to represent complex event patterns. In particular, we use event calculus [47] to model complex events. It is a general framework for representing and reasoning about events and their effects. The general theory of event calculus, which defines the meaning of the predicates, is called event calculus ontology. It contains a set of predicates as shown in Table 3.

To define an event, event calculus uses a set of rules to describe its different aspects. For instances, the event instances are described with the use of the *happens* predicate, the effects of events are described with the use of the *initiates* and *terminates* predicates, and the values of the fluents (variables) are described with the use of the *holds_at*, *mvi* (maximal validity interval) and other predicates. Each rule has a body that represents the preconditions and a rule head representing the effect. The event calculus fluents are defined as facts in the logic program. Once the event calculus models are captured in logic language, forward chain reasoning is started by the introduction of annotated events information as facts into the Rete network [16]. This results in the activation of new events. The derived events flow back into the Rete network which, in turn, results in the activation of new complex events and situations.

Event calculus axioms can be formulated in many different ways depending upon the required situation. For example, two basic axioms are given as follows:

$$holds\_at(p, t_2) \leftarrow holds\_at(p, t_1) \wedge \neg terminate(e, p, t_1) \quad (A1)$$

$$\neg hold\_at(p, t_2) \leftarrow terminates(e, p, t_1) \quad (A2)$$

In axiom A1, fluent p holds at time t_2 if it holds at time t_1 and is not terminated at time t_1 due to event e. In axiom A2, fluent p does not hold at time t_2 if an event e terminates p before time t_2 i.e. at t_1. Similarly, many different formulations can be constructed to model any environment. In the next section, we define basic rules in event calculus which are then used as basic building blocks to represent complex event patterns.

5.2.1. Modelling an EV in event calculus

As described previously that an event can be simple as well as complex, its state and the transition between the states need to be modelled based on its current and the desired state. In the following, the basic rules to represent the state of a complex event (i.e. *EV*) are described. For this purpose, three predicates *init_state*, *curr_state* and *trans_state* are defined, which perform various operations on *EVs*. These predicates are defined as follows:

- *Initial state (init_state)* This predicate models the initial state of an *EV* at the beginning of the business process management. It is represented as follows:

  $$initially(ev(x, s_t)) \leftarrow init\_state(x, s_t)$$

  The initial state of the *EV* is modelled by the above rule such that variable x represents two things: (i) constraint of the *EV* and (ii) unique ID of the *EV*. Moreover, fluent $ev(x, s_t)$ represents the state of the *EV* identified by x and s_t represents its current state.
- *Transition state (tran_state)* This predicate represents the change in the state of an *EV* from one state to another (e.g. s_1 to state s_2) after the occurrence of an event e. It is represented by the following two rules:

  $$terminates(e, ev(x, s_1), t) \leftarrow happens(e, t) \wedge trans\_state(e, t, x, s_1, s_2) \wedge curr\_state(x, s_1, t)$$

  $$initiates(e, ev(x, s_2), t) \leftarrow happens(e, t) \wedge trans\_state(e, t, x, s_1, s_2) \wedge curr\_state(x, s_2, t)$$

  With the execution of these two rules, the fluent $ev(x, s_1)$ terminates and the fluent $ev(x, s_2)$ initiates, which represents the change of state of the *EV* from s_1 to s_2.
- *Current state (curr_state)*: This predicate returns the current state s_t of an *EV* at any time instant t. It is represented by the following rule:

  $$holds\_at(ev(x, s_t), t) \leftarrow curr\_state(t, x, s_t)$$

These predicates can describe the process of identifying and translating business constraints into logical rules. Moreover, these predicates can also represent the transition of constraints from one state to another during execution. Fig. 4 depicts the flow among three predicates to represent a change in the state of an *EV*. First of all, *EVs* and their constraints are identified from the business process. Then, fluents corresponding to each of the *EVs* are initiated by using *init_state* with their initial states. When new runtime observation for an *EV* is received, it may lead to two different scenarios. It may, either, lead to a transition to a different state or it may stay in the current state. These two scenarios are represented by *tran_state* and *curr_state*, respectively. The final state of an *EV* is represented by the *curr_state* predicate.

5.2.2. Final state of an EV in event calculus

In this section, rules are defined to get the state of an *EV* at any time t during execution of a business process. Let X be an *EV* identified in a business process. Its possible final state can be defined as:

- *Normal (X)* An *EV* can be in a normal state in two possible ways. (i) If the initial state of an *EV* after the identification in business process is in *normal* state or (ii) due to transition from *failure* (or *uncertain*) state to a *normal* state.

In the first case, an *EV* is in a *normal* state and does not violate the defined constraints. This can be represented using the following syntax.

$$init\_state(X, N)$$

The current state of the X can be inquired by the predicate *curr_state* which is given by the following rule:

$$holds\_at(ev(X, N), t) \leftarrow curr\_state(t, X, N)$$

In the second case, the state of the X transitions from *failure* (or *uncertain*) state to a *normal* state at time t. The rule for the transition from an *uncertain* to a *normal* state is as follows:

$$trans\_state(e, t, X, U, N)$$

This predicate terminates the *uncertain* state of the X and initiates its *normal* state, which is represented as follows:

$$terminates(e, ev(X, U), t) \leftarrow happens(e, t) \wedge trans\_state(e, t, X, U, N) \wedge curr\_state(X, U, t)$$

$$initiates(e, ev(X, N), t) \leftarrow happens(e, t) \wedge trans\_state(e, t, X, U, N) \wedge curr\_state(X, N, t)$$

The current state of the X after the transition can be inquired by the predicate *curr_state* which given by the following rule:

$$holds\_at(ev(X, N), t) \leftarrow curr\_state(t, X, N)$$

Similarly, when the state of the X transitions from a *failure* to a *normal* state, the transition rule is represented as:

$$trans\_state(e, t, X, F, N)$$

This terminates the *failure* state of the X and initiates its *normal* state.

- *Uncertain (X):* At time t when it is not possible to decide whether X is in *normal* state or *failure* state (e.g. due to missing sensor value or constraint boundary value), it moves to an *uncertain* state.

  $$trans\_state(e, t, X, N, U)$$

  $$trans\_state(e, t, X, F, U)$$

- *Failure (X):* At time t when X moves from either a *normal* or *uncertain* state to a *failure* state, it is represented by the following rules, respectively:

  $$trans\_state(e, t, X, N, F)$$

  $$trans\_state(e, t, X, U, F)$$

5.2.3. Complex event pattern detection and state composition using event calculus

In Section 5.1, the complex event operators and their corresponding syntax to represent them were defined. However, that syntax does not show how different events can be combined in a complex pattern in a time interval and how the states of *EVs* are changed or transitioned when an event occurs. Our complex event definitions rely on event calculus for time interval using

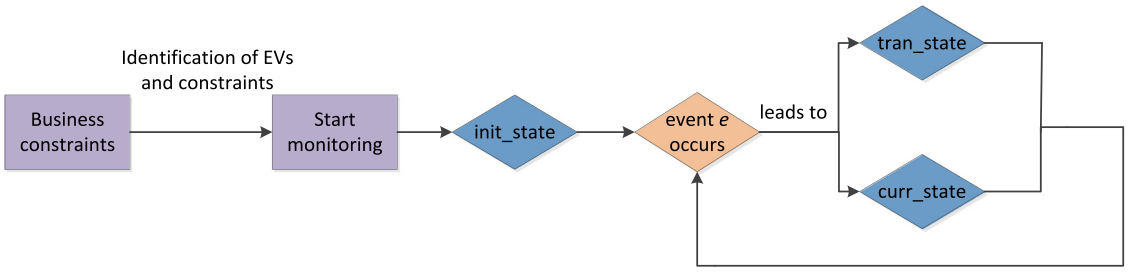

Fig. 4. Flow among three predicates to represent a change in the state of an *EV*.

interval-based semantics, where each complex event has a *start* time which represents when it is detected in a time window. The main motivation of using event calculus for complex event detection is that it not just enables to model the events, it can also capture the effects of those events that in turn can help to recognize and learn the event pattern. For example, *SEQ(dinner, bed)* pattern can be identified using the following rule:

$$happens(SEQ(dinner,\ bed), t_4) \leftarrow mvi(dinner, t_1, t_2) \wedge mvi(bed, t_3, t_4) \wedge t_2 < t_3$$

where t_1, t_2, t_3 and t_4 represents time such that $t_1 < t_2 < t_3 < t_4$. This rule detects the sequence of two activities *dinner* and *bed* in which the *dinner* activity must happen before the *bed* activity. The time when the sequence is detected is attached to the event instance. Similarly, *PAR(dinner, newspaper)* pattern can be identified using the following rule:

$$happens(PAR(dinner,\ newspaper), max(t_2, t_4)) \leftarrow mvi(dinner, t_1, t_2) \wedge mvi(newspaper, t_3, t_4) \wedge max(t_1, t_3) < \min(t_2, t_4)$$

where $max(t_2, t_4)$ returns either t_2 or t_4 whichever is maximum and $min(t_1, t_3)$ returns either t_1 or t_3 whichever is minimum.

The rules for complex event definitions such as *DURING(dinner, newspaper), STARTS(dinner, newspaper), FINISHES(dinner, newspaper), OR(movie, game), AND(movie, game)* and *NOT(e)* can be identified using the following rule respectively:

$$happens(DURING(dinner, newspaper), t_4) \leftarrow mvi(dinner, t_1, t_4) \wedge mvi(newspaper, t_2, t_3) \wedge t_1 < t_2 < t_3 < t_4$$

$$happens(STARTS(dinner, newspaper), t_1) \leftarrow mvi(dinner, t_1, t_2) \wedge mvi(newspaper, t_1, t_3)$$

$$happens(FINISHES(dinner, newspaper), t_3) \leftarrow mvi(dinner, t_1, t_3) \wedge mvi(newspaper, t_2, t_3)$$

$$happens(OR(movie, game), t) \leftarrow (happens(movie, t) \vee happens(game, t)) \wedge t_1 < t < t_2$$

$$happens(AND(movie, game), t) \leftarrow (happens(movie, t) \wedge happens(game, t)) \wedge t_1 < t < t_2$$

$$happens(NOT(e), t) \leftarrow \neg happens(e, t)$$

Each of the event detected using the rules above is assigned a unique id. Moreover, as described earlier, every complex event (represented as *EV*) is assigned a state representing its compliance or violation of the business process. Furthermore, the evaluation of the state of a business process or activity also requires the evaluation of all the complex events detected in the process, which may be connected with each other in sequential, nested, or hierarchical form. In our model, the states of the *EV*, which is composed of other *EVs*, is defined using event calculus rules.

In the rest of this section, the *conjunction* and *disjunction* rules are explained to determine the state of an *EV* from the states of other *EVs* it is composed of. Similar rules can be defined for *implication*, *biconditional* and *negation* operators to achieve the desired state as shown in Table 2. Consider the example of two complex events *SEQ(x_1,x_2)* and *PAR(y_1,y_2)*, which are defined as *EVs* namely, EV_1 and EV_2 respectively. Both are monitored for some predefined constraints (such as number of occurrences etc.) in a time duration. Now assume, a business process requires the conjunction (say $EV_{CON(1,2)}$) of both EV_1 and EV_2 to be in *normal* state in order to be functional. For this purpose, during execution, appropriate states (*normal*, *failure*, or *uncertain*) are assigned to each of them. The conjunction operator then can determine the state of $EV_{CON(1,2)}$ depending upon the current states of EV_1 and EV_2 based on the multi-valued logic presented in Table 2.

In the following, the *conjunction* rules to determine the *normal* (N), *failure* (F), or *uncertain* (U) states of the *EV* consisting of *m* different *EVs* detected at time *t* are presented. Following rule detects the *normal* (*N*) state of the *EV* identified as $EV_{CON(1,2,\ldots,m)}$ when conjunction operator is applied to *m* different *EVs*.

$$holds\_at\ (ev(x, N), t) \leftarrow (\forall i \in \{1{:}\ m\}\ (\ holds\_at(ev(i, N), t))) \vee (\exists i \in \{1{:}\ m\}\ (holds\_at(ev(i, N), t)) \wedge \forall j \in \{1{:}\ m\}, j \neq i\ (holds\_at(ev(j, N), t) \vee holds\_at(ev(j, U), t)))$$

In the above rule, *ev(x, N)* is a fluent representing an $EV_{CON(1,2,\ldots,m)}$ identified as *x* with a *normal* state (represented as *N*) such that its state represents the conjunction of *m* different *EVs* at time instant *t*. It is evaluated to a *normal* state when either all of the *EVs* it is composed of are evaluated to *normal* state or at least one of the *EVs* it is composed of is in a *normal* state at any time instant *t* and the remaining *EVs* are evaluated to either *normal* or *uncertain* state. The following rule detects the *failure* (*F*) state of the *EV* when conjunction operator is applied to it.

$$holds\_at(ev(x, F), t) \leftarrow \exists i \in \{1{:}\ m\}\ (holds\_at(ev(i, F), t))$$

In the above rule, fluent *ev(x, F)* evaluates to a failure state when at least one of the *EVs* it is composed of is evaluated to a *failure* state at any time instant *t*. Similarly, the following rule detects the *uncertain* (U) state of the *EV* when conjunction operator is applied to it.

$$holds\_at(ev(x, U), t) \leftarrow \forall i \in \{1{:}\ m\}\ (holds\_at(ev(i, U), t))$$

In the above rule, fluent *ev(x, U)* is evaluated to an *uncertain* state when all of the *EVs* it is composed of are evaluated to an *uncertain* state at any time instant *t*.

Similar to the *conjunction* rules, the *disjunction* rules to determine the *normal* (N), *failure* (F), or *uncertain* (U) states of the *EV* consisting of *m* different *EVs* at time *t* are defined below.

$$holds\_at(ev(x, N), t) \leftarrow \exists i \in \{1{:}\ m\}\ (holds\_at(ev(i, N), t))$$

The fluent *ev(x, N)* is evaluated to a *normal* state when at least one of the *EVs* it is composed of is evaluated to a *normal* state at any time instant *t*.

$$holds\_at(ev(x, F), t) \leftarrow (\exists i \in \{1{:}\ m\}\ (holds\_at(ev(i, F), t))$$

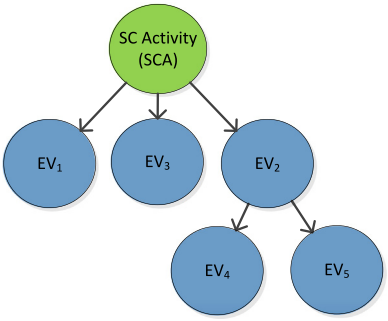

Fig. 5. Relationships between the *EVs* in a supply chain activity.

$$\wedge\, \nexists j \in 1{:}\, m(holds\_at(ev(j, N), t)))$$

The fluent *ev(x, F)* represents the disjunctive *EV* with a *failure* state. It is evaluated to a *failure* state when at least one of the *EVs* it is composed of is evaluated to a *failure* state at any time instant *t* and none of the *EVs* it is composed of is evaluated to a *normal* state at time instant *t*.

$$holds\_at(ev(x, U), t) \leftarrow\ \forall i \in \{1{:}\, m\}\ (holds\_at(ev(i, U), t))$$

The fluent *ev(x, U)* is evaluated to an *uncertain* state when all of the *EVs* it is composed of are evaluated to an *uncertain* state at any time instant *t*.

The rules for *implication*, *disjunction* and *negation* can also be defined in similar way. All of the above mentioned rules are essential for predictive complex event reasoning. In the next section, the reasoning process to predict the failure of a business process, which relies on the above mentioned rules, is described.

6. Logical and probabilistic reasoning engine

In this section, we describe the reasoning process of the proposed framework which consists of two steps. In step 1, the logical reasoning is performed on the incoming events from data streams to identify the situations of interest (e.g. failure of a business process). This is done by first determining the state of the *EVs* from the incoming data streams and then inferring the failure of the business process. However, in case, when the incoming data stream contains incomplete, inaccurate or missing information about any *EV*, the probabilistic inferencing in step 2 is performed. In the probabilistic reasoning, a variant of Bayesian Network (called Adaptive Dynamic Bayesian Network) is constructed and then used to determine the possible state of the *EVs* and the failure of the business process is determined.

6.1. Logical reasoning

In the logical reasoning process, the sequence of events from data streams are given as input to the reasoning engine. The logical reasoning process is described in Algorithm 1, which is based on the work presented in [14] but has been adapted for application in IoT-enabled supply chain for predictive monitoring. The algorithm takes sensor observations as input at any time t and returns the updated knowledge base, represented as $KB_{(t+1)}$, with the possible future states of the *EVs* for time $t+1$. To start the reasoning process, the event calculus theory and its basic components such as domain-independent axioms (Σ), domain specific axioms ($\Delta_{(0)}$), and initial knowledge base ($KB_{(0)}$) with initial facts, are initialized (line 1–3). The complex event rules, which model *EVs* and their constraints, are represented using the state constraints (Ψ) and the effect constraints (ε) (line 4–5). When an event occurs, these rules trigger and change the states of the *EVs* represented using *fluents* in the system. The *EVs* whose states are affected due to an event, are represented by the symbol $\bar{O}_{(t)}$ (line 7). This symbol represents the collection of states of the *EVs* (8–10).

Algorithm 1: ComplexEventReasoning

Data: Sensor observations at timeslot t
Result: updated $KB_{(t+1)}$
1. Initialization;
2. Initialize $KB_{(0)}$ with domain axioms $\Delta_{(0)}$ and initial facts;
3. Initialize *Event Calculus* domain-independent axioms Σ
4. Activate *Event Calculus* state constraints Ψ
5. Activate *Event Calculus* effect constraints $\mathcal{E}$
6. **loop**
7. input sensor observations at timeslot t and update observation axioms $\bar{O}_{(t)}$
8. **for all** $KB_{(t)}$ at timeslot t **do**
9. $\acute{K}\beta_{(t+1)} \leftarrow KB_{(t)} \cup \bar{O}(t) \cup \Sigma$
10. $\acute{K}\beta_{(t+1)} \leftarrow \acute{K}\beta_{(t+1)} \cup \mathcal{E}$
11. **if** $\acute{K}\beta_{(t+1)}$ is consistent with Ψ then
12. $KB_{(t+1)} \leftarrow \acute{K}\beta_{(t+1)} \cup \Psi$
13. store $KB_{(t+1)}$
14. **else**
15. $KB_{(t+1)}$ = PredictUnknownStates($KB_{(t+1)}$)
16. **end if**
17. **end for**
18. **end loop**

The symbol $\acute{K}\beta_{(t+1)}$ represents the change in the knowledge base after occurrence of an event, which is validated against the state constraints to determine states of the *EV* fluents. At this stage, algorithm classifies *EVs* into one of two types, namely *known EVs* and *unknown EVs*. (i) *Known EVs* ($\bar{O}(t)$). On occurrence of an event, if state of an *EV* is known either as a direct result of the event or due to its dependence on other *EVs* is called as *known EV*. The *known EVs* are represented by the vector $O = \{EV_1, EV_2, \ldots, EV_n\}$ and each of them is either in a *normal*, *failure* or *uncertain* state (line 11–12). The symbol $KB_{(t+1)}$ represents updated knowledge base at time $t+1$. At this time, the future state of business process can be inferred from the states of all the known *EVs* (line 13). (ii) *Unknown EVs* ($\bar{Q}_{(t)}$). An *EV*, which is not impacted by an event and its future state cannot be determined from the current set of events, is termed as *unknown EV*. For example, in Fig. 5, if the current state of any *EV* (e.g. EV_1 or EV_2) is unknown, the potential state of the process (*SCA*) cannot be determined. So, the algorithm determines the probability of the unknown *EVs* being in each of the three states, namely *normal*, *failure* or *uncertain* by using the *EVs*' previous states which gives a measure of the effect of past events on the *EV* (line 15).

Algorithm 2: PredictUnknownStates

Data: $KB_{(t+1)}$
Result: updated $KB_{(t+1)}$
1. $\bar{Q}_{(t)} \leftarrow$ produce all combinations of truth values for the *EV* fluents which are not affected at timeslot t
2. $\bar{Q}Set \leftarrow$ Perform Probabilistic Inference($\bar{Q}_{(t)}$)
3. **foreach** $q \in \bar{Q}Set$ **do**
4. **if** $\acute{K}\beta_{(t+1)} \cup q$ is consistent with Ψ then
5. $KB_{(t+1)} \leftarrow \acute{K}\beta_{(t+1)} \cup \Psi \cup q$
6. store $KB_{(t+1)}$
7. **end if**
8. **end for**

First of all, in Algorithm 2, the probability distribution tables for the possible states of unknown *EVs*, which are represented by $\bar{Q}_{(t)}$, are generated (line 1–2). The knowledge base is updated with the new information coming from probabilistic inference to determine the future state of the business process (line 3–8). The process of probabilistic inference using Bayesian network is described in next section.

6.2. Probabilistic inferencing

To determine the state of unknown *EVs*, a variant of the Bayesian Network (BN) is used, which is known as *Adaptive Dynamic Bayesian Network* (ADBN). A *BN* is a probabilistic model that is represented via a directed acyclic graph where nodes represent variables and arcs represent their dependencies as shown in Fig. 5. It provides full representations of probability distributions and can support any direction of reasoning [48]. For example, the direction of reasoning can be between nodes to ascertain the causes of an effect. An *ADBN* enhances the *BN* with dynamic learning capability to build structure of the *BN* from the training data and then adapting the learned structure by continuously updating the conditional probability tables (CPTs) to improve prediction accuracy.

To elaborate the objective of using probabilistic inferencing, consider that we want to find the conditional probability (represented as *Pr(SCA|O)*) for the state of complex event (represented as *SCA*) for a given set of *EVs* (represented by the vector $O = \{EV_1, EV_2, \ldots, EV_n\}$). This represents the probability of the state of a supply chain activity. However, in order to determine *Pr(SCA|O)*, all of the *EVs'* states are required. Consequently, for those $ev_j \notin O$ for which there is no information about their states at a particular time *t*, we consider their probability of being either in a *failure*, *normal* or *uncertain* state. Let *Q* be the vector of *EVs* represented as $Q = \{E\tilde{V}_1, E\tilde{V}_2, \ldots E\tilde{V}_m\}$ whose states are unknown at a particular time *t*. Then the probability of *SCA* when *O* terms are known can be calculated as follows, according to the Bayes rule:

$$Pr\,(SCA|O) = \frac{\sum Pr(\langle SCA, EV_1, \ldots, EV_n, E\tilde{V}_1, \ldots, E\tilde{V}_m\rangle)}{\sum Pr(\langle EV_1, \ldots, EV_n, E\tilde{V}_1, \ldots, E\tilde{V}_m\rangle)} \tag{1}$$

where *EV* represent *EVs* with known states and $E\tilde{V}$ represent *EVs* with unknown states. The probability distributions of all possible combinations of the *EVs* is represented using the summation sign in the numerator and the denominator of the above equation. To explain this equation with an example, consider one *SCA* that has three *EVs* represented as EV_1, EV_2 and EV_3 as shown in Fig. 5 (ignoring EV_4 and EV_5). Assume that at a specific time *t*, the states of EV_1 and EV_3 (also represented by their names) are known only and the state of EV_2 (represented as $E\tilde{V}_2$) is unknown. Then, the state of *SCA* when the states of EV_1 and EV_3 are given is determined as:

$$Pr\,(SCA|EV_1 \wedge EV_3) = \frac{\sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr\,(SCA)\, Pr(EV_1, EV_3, E\tilde{V}_2|SCA)}{\sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr(EV_1, EV_3, E\tilde{V}_2)} \tag{2}$$

Each *EV* can have state either *n*, *f* or *u*, we evaluate the probabilities of the unknown *EVs* being in these three states. The expression $Pr(EV_1, EV_3, E\tilde{V}_2|SCA)$, in the above equation, is difficult to estimate when the number of terms increases. Therefore, it can be substituted with $Pr\,(EV_1|SCA)\, Pr\,(EV_3|SCA)\, Pr(E\tilde{V}_2|SCA)$ by the assumption that all the *EVs* are independent [49]. Hence, Eq. (2) takes the following form:

$$Pr\,(SCA|EV_1 \wedge EV_3) = \frac{\sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr\,(SCA)\, Pr\,(EV_1|SCA)\, Pr\,(EV_3|SCA)\, Pr(E\tilde{V}_2|SCA)}{\sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr(EV_1, EV_3, E\tilde{V}_2)} \tag{3}$$

From the above equation, the term $\sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr(EV_1, EV_3, E\tilde{V}_2)$ can be substituted by a normalization constant $\alpha = 1/\sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr(EV_1, EV_3, E\tilde{V}_2)$ [49]. Hence, Eq. (3) can be written as:

$$Pr\,(SCA|EV_1 \wedge EV_3) = \alpha \sum_{E\tilde{V}_2 \in \{n,f,u\}} Pr\,(SCA)\, Pr\,(EV_1|SCA) \times Pr\,(EV_3|SCA)\, Pr(E\tilde{V}_2|SCA) \tag{4}$$

The arc from SCA to EV_1, as shown in Fig. 5, represents the conditional probability $Pr\,(EV_1|SCA)$. A conditional probability table (CPT) is created, which contains probabilities for all the possible combinations of states of EV_1 for each the state of the *SCA*. Consider, three states of EV_1 are denoted by n_1, f_1 and u_1 and the three states of *SCA* are denoted by *normal*, *failure* and *uncertain*. Then CPT for $Pr\,(EV_1|SCA)$ is given as:

$$Pr(EV_1|SCA) = \begin{bmatrix} Pr(n_1|normal) & Pr(n_1|failure) & Pr(n_1|uncertain) \\ Pr(f_1|normal) & Pr(f_1|failure) & Pr(f_1|uncertain) \\ Pr(u_1|normal) & Pr(u_1|failure) & Pr(u_1|uncertain) \end{bmatrix} \tag{5}$$

where each column in the above matrix sums to 1 and shows the probability of EV_1 being in n_1, f_1 and u_1 for every state of the *SCA*. Similarly, matrices for $Pr\,(EV_2|SCA)$, $Pr\,(EV_3|SCA)$, $Pr\,(EV_4|EV_2)$ and $Pr\,(EV_5|EV_2)$ can be found from the past data of *EVs* being in a state for each state of the *SCA*. Referring back to the example of three *EVs* and an *SCA* described above, if the states for EV_1 and EV_3 are known as n_1 and n_3 respectively, then the probability of the *SCA* being in a *failure* state can be found by determining the state of the unknown EV_2 (whose *normal*, *failure* and *unknown* states are represented as n_2, f_2, and u_2 respectively) as:

$$\begin{aligned} &Pr\,(SCA(failure)|n_1 \wedge n_3) \\ &= \alpha(Pr\,(failure)\, Pr\,(n_1|failure)\, Pr\,(n_3|failure)\, Pr\,(n_2|failure) \\ &+ Pr\,(failure)\, Pr\,(n_1|failure)\, Pr\,(n_3|failure)\, Pr\,(f_2|failure) \\ &+ Pr\,(failure)\, Pr\,(n_1|failure)\, Pr\,(n_3|failure)\, Pr(u_2|failure)) \end{aligned} \tag{6}$$

By replacing the corresponding probabilities from the CPTs of EV_1, EV_3 and EV_2 in the above equation will give the probability of a *failure* state of *SCA*. Similarly, the probabilities for the *normal* and *uncertain* states of *SCA* can be found. The state with the highest probability is used as the possible state of the *SCA* in this case.

7. Validation of complex event reasoning engine

The experimental setup for the validation of the complex event reasoning engine in terms of execution performance is presented. For this purpose, a prototype is developed in SWI-Prolog. The prototype system allows to configure business constraints by defining simple and complex events and their relationships. For system evaluation, the experiment were conducted on a computer equipped with an Intel i7-4790 3.60 GHz CPU with 16 Gb RAM. The performance as well as accuracy of the reasoner in executing event calculus and the proactive determination of business process compliance (violation) management was determined. The experiments were conducted for two key reasons. Firstly, to determine the performance of the system in terms of the execution time of the reasoning process. Secondly, to validate ability of the system to predict the possibility of business process failure before it actually happens. These are explained in the following sections.

7.1. Performance evaluation

The performance evaluation of the system is carried out by measuring the execution time by scaling the size of the number of events, fluents and axioms. The execution time is measured in a setting where two or more sensor objects continuously send the response time measurements at several timeslots.

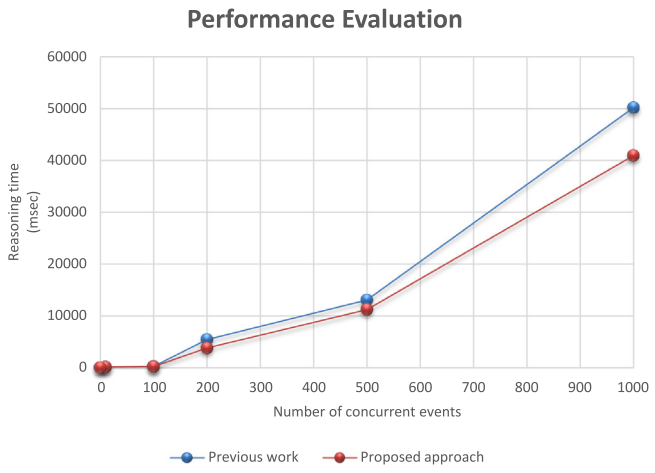

Fig. 6. Proactive reasoning performance.

The dataset for performance evaluation is based on information obtained by a number of binary sensors providing data about the state of machines, valves, devices etc. in an organization at different timestamps [50]. Moreover, the data also contains some missing values when sensor was unable to generate a reading. These measurements are used to evaluate the business process represented by Fig. 5. Each of these measurement affects one or more fluents and subsequently different event types. These changes then can trigger a chain of other fluents leading to multiple effect axioms such as initiates() and terminates().

The performance of the current system is compared with a previous work [14] under similar settings. Table 4 shows the number of events that are used to evaluate the systems. These events range from 100 to 1000 with different number of facts and rules. The fluents are affected when an event triggers change in the knowledge base and consequently other fluents are initiated or terminated in the system. The system is based on event calculus modelling and few properties are incorporated from our previous work [14]. However, some characteristics are unique and enhance the system performance. For example, if an *EV* move to *failure* state from a *normal* state, the fluents corresponding to that *EV* initiate or terminate their respective effects. There are at least two fluents that change for every state change in an *EV*, one fluent for the *normal* (represented by true) and *failure* (represented by false) state while the second fluent handles *uncertain* state (when set to true). The second fluent, when set to true, takes the priority over the first fluent. When it is false, the determination of the state depends on the first fluent. In previous work, three fluents, one each for *normal*, *failure*, and *uncertain* state, were used. This means for each *EV*, an additional fluent (or fact) was used and hence more fluents were to be managed for each rule execution. Moreover, in some cases, these changes consequently affect other event types which, in turn, trigger other fluents. From this experiment, it can be concluded that reasoning time directly depends on the number of changes needed to model the effect of an event. This is also evident from Fig. 6 where the reasoning time is increasing gradually as the number of events processed are increasing.

7.2. Evaluation of the prediction quality

In this section, the quality of the inferred conclusions are subject to investigation. The objective is to show the applicability of our approach to a supply chain activity of an organization.

7.2.1. Dataset description

For validation of the proposed framework, a dataset of semi-conductor manufacturing process is used, which is available at https://archive.ics.uci.edu/ml/datasets/SECOM. This is a real-world dataset collected from complex modern semi-conductor manufacturing process that is normally under consistent surveillance via monitoring of signals. The signals are collected from both sensors devices (e.g. vacuum chamber sensor) and process measurement points. In IoT-enabled supply chain, the signals from sensor devices (e.g. vacuum chamber sensor data) as well as process measurement points are remotely monitored and then valuable data and alerts are delivered to the supply chain partners.

The dataset consists of 1567 process instances each with 591 different features. The features indicate monitoring signals such as temperature, infrared radiation etc., which are collected from sensors and process measurement points. The measured signals contain useful, irrelevant, noise as well as missing information. A process instance can result in either *failure* or success (i.e. *normal*). Although, the target of the process instances consist of two values (i.e. *failure* and *normal*), the features of the process instances are annotated with three values (i.e. *failure, normal, uncertain*). The *uncertain* value is assigned to a feature if it has a missing information or its value lies at or around the boundary of its constraint. To make evaluation of the proposed framework manageable, a subset of important features are selected from the dataset using Correlation-based Feature Selection (CFS) method. This results in 34 important features by considering the individual predictability of each feature along with the degree of redundancy between them. Moreover, the causal relationship among features is also considered for selection of key features. This subset of the data is used to evaluate the proposed framework for predicting the chances of failure or success of the supply chain activity.

7.2.2. Model creation

For evaluation and comparison of the proposed framework, two models are created. The first model is called the *proposed* model, which utilizes the predictive CEP engine with logical and probabilistic reasoning while the second model is called the *baseline* model, which utilizes probabilistic inferencing but does not include the logical reasoning capability of the complex event component. Both models are trained on the same training dataset. In the following, we describe the characteristics of the two models and then compare and discuss their results.

As described previously, the baseline model employs probabilistic inferencing using BN. The structure of the BN is learned from training dataset that captures the dependency among the features of the dataset. This model simulates process of an organization that employs various monitoring means using IoT sensor devices to detect deterioration, compliance or non-compliance in the process of its production line; for example: monitoring temperature indicators, monitoring electric indicators and performing radiation analysis etc. [51].

On the other hand, the proposed model employs CEP engine along with probabilistic inferencing using *ADBN* to identify several different patterns that imply various failure distributions. For example, the occurrence of an event in a feature (or a set of features) will have an impact on a target feature. This dependency relationship between the features is defined through a semantic annotation ontology. Using this model, the IoT-enabled supply chain environment is simulated where the data is coming from different sensors and process measurement points. We focus on indicators (i.e. temperature and radiation) in semi-conductor manufacturing process and use one example pattern ($PAR(EV_1, EV_2)$ which is represented as EV_3 in Fig. 7), which occur if a combination of two conditions occur in parallel: (i) steady increase

Table 4
Performance comparison of the proposed approach with previous work.

Events	Previous work [14]			Proposed approach		
	# of facts in kB	# of rules in kB	Reasoning time (ms)	# of facts in kB	# of rules in kB	Reasoning time (ms)
100	460	178	249	420	178	194
200	660	278	5 406	620	278	3 790
500	1160	378	13 065	1120	378	11 185
1000	2160	478	50 176	2120	478	40 918

Table 5
Confusion matrix.

Target	Total instances (actuals)	Baseline model (predicted)		Proposed model (predicted)	
		Normal	*Failure*	*Normal*	*Failure*
Normal	104	6	98	35	69
Failure	1463	21	1442	10	1453

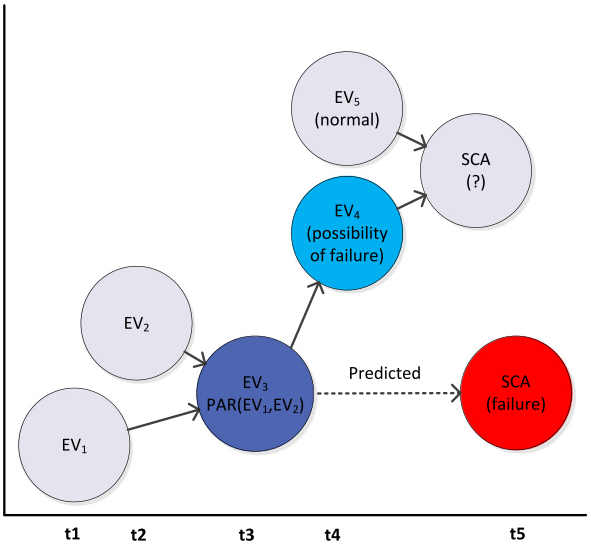

Fig. 7. Event timeline of the example.

Table 6
Overall accuracy.

	Baseline model			Proposed model		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Normal	0.222	0.058	0.092	0.778	0.337	0.470
Failure	0.936	0.986	0.960	0.955	0.993	0.974
Avg./Total	0.889	0.924	0.903	0.943	0.950	0.940
Overall F1-score:	90%			94%		

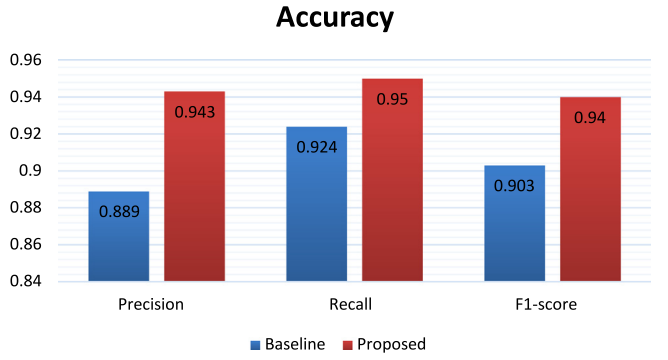

Fig. 8. Accuracy comparison.

in infrared radiation (say EV_1) and (ii) abnormal temperature on a specific location of a component (say EV_2). Occurrence of this pattern with *failure* state implies a higher probability of failure of business process as EV_4 is directly dependent on EV_3 as shown in Fig. 7. However, if the reasoning algorithm receives insufficient information about EV_2 at a specific time (say t_3 as shown in Fig. 7) while the state of EV_1 is known, then the decision for the possible state of complex event EV_3 (as well as EV_4) depends on EV_2 which is unknown. In this scenario, the possible state of EV_2 can be determined using a probabilistic inferencing by the proposed model.

7.2.3. Evaluation results

The detailed quality of the prediction results of the two models are evaluated and compared in Tables 5 and 6. The labels *Normal* and *Failure* represent the states of *process instances* in the dataset. A comparison of confusion matrix is presented in Table 5. In the actual dataset, out of 1567 process instances, 104 process instances have *Normal* state while process instances having *Failure* state are 1463. The baseline model predicted merely 6 instances correctly out of total 104 *Normal* instances while 98 instances were falsely predicted as *Failure*. On the other hand, the proposed model correctly predicted 35 out of 104 *Normal* instances while 63 were falsely predicted as *Failure*. This shows that baseline model achieved a precision of 22.2% and a recall of 5.8% while the proposed model achieved a precision of 77.8% and a recall of 33.7% for *Normal* instances (shown in Table 6). Similarly, significant accuracy is achieved by the proposed model in predicting the *Failure* instances. The baseline model predicted 1442 correctly out of total 1463 *Failure* instances while 21 were falsely predicted as *Normal*. Whereas the proposed model predicted 1453 correctly out of 1463 *Failure* instances and only 10 were falsely predicted as *Normal*.

Table 6 shows a comparison of the prediction accuracy indicators such as precision, recall and F1-scores of the two models. It also shows the accuracy in predicting *Normal* and *Failure* instances as well as overall accuracy. The overall precision, recall, and F1-score of the two models is also illustrated in Fig. 8. It is clear from this figure that the proposed model performed better in all accuracy indicators as compared to the baseline model.

8. Conclusion

The emergence of IoT is giving rise to the concept of IoT-enabled supply chain. It provides a natural combination to achieve SCV which refers to ability of supply chain partner to collect and analyse distributed data for decision support. This data is collected and analysed in real-time by a specialized software known as CEP engines. However, CEP engines have two limitations which include their inability to combine multiple related

sensor data streams from distributed sources and their inability of performing predictive reasoning when the information about the event of interest is incomplete and/or uncertain. In this paper, we proposed a framework for predictive and proactive complex event reasoning which processes, integrates, and provides reasoning over complex events to predict disruptive events under incomplete and/or uncertain information using the logical and probabilistic reasoning approaches. The main contributions of this work are: (1) semantic annotation of heterogeneous data stream to integrate and annotate the distributed events information, (2) a novel approach for semantic complex event processing to process and correctly interpret semantically annotated complex events, and (3) development of a methodology for predictive and proactive complex event reasoning when underlying information is incomplete and/or uncertain. The validation of the proposed approach for prediction of failures in IoT-enabled supply chain activity is performed on real-world semi-conductor manufacturing dataset. The results of the validation of the proposed model were compared with a baseline model, which indicated that the proposed approach is much efficient and correctly predicts failures even under incomplete and/or uncertain information. In future work, CEP definitions in event calculus will be extended to explore its applicability in other domains. In particular, we will explore the application of the proposed CEP in transportation and logistics industry to capture and predict future events of interest.

Acknowledgements

This research was partially supported by Edith Cowan University (ECU), Australia Early Career Researcher Grant — 2018.

The authors would also like to take this opportunity to express their deepest appreciations for the time and effort devoted by the anonymous reviewers.

References

[1] A.N. Zhang, M. Goh, F. Meng, Conceptual modelling for supply chain inventory visibility, Int. J. Prod. Econ. 133 (2011) 578–585, http://dx.doi.org/10.1016/j.ijpe.2011.03.003.

[2] V. Francis, Supply chain visibility: Lost in translation? Supply Chain Manag. 13 (2008) 180–184, http://dx.doi.org/10.1108/13598540810871226.

[3] N. Tohamy, L.M. Orlov, L. Herbert, Supply Chain Visibility Defined, Forrester Research, Cambridge, MA, 2003.

[4] N.K. Janjua, O.K. Hussain, E. Chang, S. Mohammed, S. Islam, Conjoint utilization of structured and unstructured information for planning interleaving deliberation in supply chains, IEEE/ACM Conf. Web Intell. (2017) http://dx.doi.org/10.475/123.

[5] P. Myerson, Lean and Technology: Working Hand in Hand to Enable and Energize your Global Supply Chain, Pearson Education, Inc., 2017.

[6] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of Cloud computing and Internet of Things: A survey, Future Gener. Comput. Syst. 56 (2016) 684–700, http://dx.doi.org/10.1016/j.future.2015.09.021.

[7] K. Yang, D. Forte, M. Tehranipoor, ReSC: An RFID-enabled solution for defending IoT supply chain, ACM Trans. Autom. Electron. Syst. 23 (2018) http://dx.doi.org/10.1007/s00706-014-1316-4.

[8] B.F. David, C. Luckham, Complex Event Processing in Distributed Systems, Hewlett-Packard, 1998, pp. 98–754, doi:10.1.1.56.876.

[9] D. Gyllstrom, E. Wu, H.-J.H. Chae, Y. Diao, P. Stahlberg, G. Anderson, SASE: Complex event processing over streams, Gen. Syst. abs/cs/061 (2006) 363–374, http://dx.doi.org/10.1016/j.pmcj.2009.06.002.

[10] D. Anicic, P. Fodor, S. Rudolph, R. St, ETALIS : Rule-based reasoning in event processing, Reason. Event-Based Distrib. Syst. (2011) 99–124, http://dx.doi.org/10.1007/978-3-642-19724-6_5.

[11] 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, J. Syst. Softw. 127 (2017) 217–236, http://dx.doi.org/10.1016/j.jss.2016.06.011.

[12] D. Anicic, S. Rudolph, P. Fodor, N. Stojanovic, Stream reasoning and complex event processing in ETALIS, Semant. Web. 3 (2012) 397–407, http://dx.doi.org/10.3233/SW-2011-0053.

[13] Y.P. Tsang, K.L. Choy, C.H. Wu, G.T.S. Ho, C.H.Y. Lam, P.S. Koo, An Internet of Things (IoT)-based risk monitoring system for managing cold supply chain risks, Ind. Manag. Data Syst. 118 (2018) 1432–1462, http://dx.doi.org/10.1108/IMDS-09-2017-0384.

[14] F. Nawaz, N.K. Janjua, O.K. Hussain, F.K. Hussain, E. Chang, M. Saberi, Event-driven approach for predictive and proactive management of SLA violations in the Cloud of Things, Future Gener. Comput. Syst. 84 (2018) http://dx.doi.org/10.1016/j.future.2018.02.025.

[15] F. Nawaz, O.K. Hussain, N. Janjua, E. Chang, A proactive event-driven approach for dynamic QoS compliance in cloud of things, in: Proc. Int. Conf. Web Intell. - WI '17, 2017, pp. 971–975, http://dx.doi.org/10.1145/3106426.3109431.

[16] N.K. Janjua, F.K. Hussain, Web@IDSS - Argumentation-enabled Web-based IDSS for reasoning over incomplete and conflicting information, Knowl.-Based Syst. 32 (2012) 9–27, http://dx.doi.org/10.1016/j.knosys.2011.09.009.

[17] M. Montali, F.M. Maggi, F. Chesani, P. Mello, W.M.P. van der Aalst, Monitoring business constraints with the event calculus, ACM Trans. Intell. Syst. Technol. 5 (2013) 1–30, http://dx.doi.org/10.1145/2542182.2542199.

[18] A. Adi, D. Botzer, G. Nechushtai, G. Sharon, Complex event processing for financial services, in: Proc. IEEE Serv. Comput. Work., IEEE Computer Society, Washington, DC, USA, 2006, pp. 7–12, http://dx.doi.org/10.1109/SCW.2006.7.

[19] S. Meister, Telemedical events: Intelligent delivery of telemedical values using CEP and hl7, in: L. Niedrite, R. Strazdina, B. Wangler (Eds.), Work. Bus. Informatics Res. BIR 2011 Int. Work. Dr. Consortium, Riga, Latv. Oct. 6, 2011 Revis. Sel. Pap., Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 1–13, http://dx.doi.org/10.1007/978-3-642-29231-6_1.

[20] L. Aniello, G.A. Di Luna, G. Lodi, R. Baldoni, A collaborative event processing system for protection of critical infrastructures from cyber attacks, in: F. Flammini, S. Bologna, V. Vittorini (Eds.), Comput. Safety, Reliab. Secur. 30th Int. Conf. 2011, Naples, Italy, Sept. (2011) 19-22. Proc., Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 310–323, http://dx.doi.org/10.1007/978-3-642-24270-0_23.

[21] A. Kibangou, A. Artikis, E. Michelioudakis, G. Paliouras, M. Schmitt, J. Lygeros, C. Baber, N. Morar, F. Fournier, I. Skarbovsky, An Integrated and Scalable Platform for Proactive Event-Driven Traffic Management, (2017) 1–21. http://arxiv.org/abs/1703.02810.

[22] A. Margara, J. Urbani, F. Van Harmelen, H. Bal, Streaming the Web: Reasoning over dynamic data, J. Web Semant. 25 (2014) 24–44, http://dx.doi.org/10.1016/j.websem.2014.02.001.

[23] T. Patkos, D. Plexousakis, A. Chibani, Y. Amirat, An event calculus production rule system for reasoning in dynamic and uncertain domains, Theory Pract. Log. Program. 16 (2016) 325–352, http://dx.doi.org/10.1017/S1471068416000065.

[24] S. Bragaglia, F. Chesani, P. Mello, M. Montali, P. Torroni, Reactive event calculus for monitoring global computing applications, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), in: LNCS, vol. 7360, 2012, pp. 123–146, http://dx.doi.org/10.1007/978-3-642-29414-3_8.

[25] I. Teinemaa, M. Dumas, F.M. Maggi, C. Di Francescomarino, Predictive Business Process Monitoring with Structured and Unstructured Data, 2678 (2016) 1019-1019–1019. doi:http://dx.oi.org/10.1007/3-540-44895-0.

[26] F.M. Maggi, C. Di Francescomarino, M. Dumas, C. Ghidini, Predictive monitoring of business processes, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), in: LNCS, vol. 8484, 2014, pp. 457–472, http://dx.doi.org/10.1007/978-3-319-07881-6_31.

[27] C. Di Francescomarino, M. Dumas, F.M. Maggi, I. Teinemaa, Clustering-based predictive process monitoring, IEEE Trans. Serv. Comput. 14 (2016) http://dx.doi.org/10.1109/TSC.2016.2645153.

[28] A. Metzger, P. Leitner, D. Ivanovi, E. Schmieders, R. Franklin, M. Carro, S. Dustdar, K. Pohl, Comparing and combining predictive business process monitoring techniques andreas, IEEE Trans. Syst. Man Cybern. Syst. 45 (2015) 276–290, http://dx.doi.org/10.1109/TSMC.2014.2347265.

[29] M. Le, B. Gabrys, D. Nauck, A hybrid model for business process event and outcome prediction, Expert Syst. 34 (2017) 1–11, http://dx.doi.org/10.1111/exsy.12079.

[30] A. Cuzzocrea, F. Folino, M. Guarascio, L. Pontieri, Predictive monitoring of temporally-aggregated performance indicators of business processes against low-level streaming events, Inf. Syst. (2018) 1–31, http://dx.doi.org/10.1016/j.is.2018.02.001.

[31] L. Cui, J. Deng, F. Liu, Y. Zhang, M. Xu, Investigation of RFID investment in a single retailer two-supplier supply chain with random demand to decrease inventory inaccuracy, J. Clean. Prod. 142 (2017) 2028–2044, http://dx.doi.org/10.1016/j.jclepro.2016.11.081.

[32] L. Cui, L. Wang, J. Deng, J. Zhang, Intelligent algorithms for a new joint replenishment and synthetical delivery problem in a warehouse centralized supply chain, Knowl.-Based Syst. 90 (2015) 185–198, http://dx.doi.org/10.1016/j.knosys.2015.09.019.

[33] L. Cui, L. Wang, J. Deng, RFID technology investment evaluation model for the stochastic joint replenishment and delivery problem, Expert Syst. Appl. 41 (2014) 1792–1805, http://dx.doi.org/10.1016/j.eswa.2013.08.078.

[34] L. Cui, J. Deng, L. Wang, M. Xu, Y. Zhang, A novel locust swarm algorithm for the joint replenishment problem considering multiple discounts simultaneously, Knowl.-Based Syst. 111 (2016) 51–62, http://dx.doi.org/10.1016/j.knosys.2016.08.007.

[35] R.A. Hemmat, A. Hafid, SLA Violation Prediction In Cloud Computing: A Machine Learning Perspective, (2016). http://arxiv.org/abs/1611.10338.

[36] D. Ivanović, M. Carro, M. Hermenegildo, Constraint-based runtime prediction of SLA violations in service orchestrations, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), in: LNCS, vol. 7084, 2011, pp. 62–76, http://dx.doi.org/10.1007/978-3-642-25535-9_5.

[37] F. Nawaz, M.R. Asadabadi, N.K. Janjua, O.K. Hussain, E. Chang, M. Saberi, An MCDM method for cloud service selection using a Markov chain and the best-worst method, Knowl.-Based Syst. 159 (2018) 120–131, http://dx.doi.org/10.1016/j.knosys.2018.06.010.

[38] F. Nawaz, O. Hussain, F.K. Hussain, N.K. Janjua, M. Saberi, E. Chang, Proactive management of SLA violations by capturing relevant external events in a Cloud of Things environment, Future Gener. Comput. Syst. 2019 (2019).

[39] Y. Engel, O. Etzion, Z. Feldman, A basic model for proactive event-driven computing, in: Proc. 6th ACM Int. Conf. Distrib. Event-Based Syst. - DEBS '12, 2012, pp. 107–118, http://dx.doi.org/10.1145/2335484.2335496.

[40] Q. Wei, Z. Jin, Service discovery for internet of things: a context-awareness perspective, in: Proc. Fourth Asia-Pacific Symp. …, 2012, pp. 2–7, http://dx.doi.org/10.1145/2430475.2430500.

[41] E. Alevizos, A. Skarlatidis, A. Artikis, G. Paliouras, Probabilistic Complex Event Recognition: A Survey, 2017, pp. 1–30, http://dx.doi.org/10.1145/3117809.

[42] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Context Aware Computing for the Internet of Things: A Survey, X, 2013, pp. 1–41, http://dx.doi.org/10.1109/SURV.2013.042313.00197.

[43] Q. Zhou, Y. Simmhan, V. Prasanna, Knowledge-infused and consistent Complex Event Processing over real-time and persistent streams, Future Gener. Comput. Syst. 76 (2017) 391–406, http://dx.doi.org/10.1016/j.future.2016.10.030.

[44] D. Ferrucci, A. Lally, K. Verspoor, E. Nyberg, OASIS Standard - Unstructured Information Management Architecture (UIMA) Version 1.0, Architecture, 2009.

[45] Context-aware QoS prediction for web service recommendation and selection, Expert Syst. Appl. 53 (2016) 75–86, http://dx.doi.org/10.1016/j.eswa.2016.01.010.

[46] D. Kozen, A completeness theorem for kleene algebras and the algebra of regular events, Inform. and Comput. 110 (1994) 366–390, http://dx.doi.org/10.1006/inco.1994.1037.

[47] M. Shanahan, The event calculus explained, Artif. Intell. Today Recent Trends Dev. (1999) 409–430, http://dx.doi.org/10.1007/3-540-48317-9.

[48] K.B. Korb, A.E. Nicholson, Bayesian Artificial Intelligence, second ed., CRC Press, Inc., Boca Raton, FL, USA, 2010, 2nd ed..

[49] D. Heckerman, D.M. Chickering, D. Geiger, D.M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, Mach. Learn. 20 (1995) 197–243, http://dx.doi.org/10.1007/BF00994016.

[50] C. Carlsson, M. Heikkilä, J. Mezei, Fuzzy entropy used for predictive analytics, IEEE Int. Conf. Fuzzy Syst. 341 (2015) 187–209, http://dx.doi.org/10.1007/978-3-319-31093-0_9.

[51] S. Sethiya, Condition Based Maintenance (CBM), Secy. to C. (2006). http://irsme.nic.in/files/cbm-sethiya.pdf.