International Journal of Semantic Computing

SENSORS TO EVENTS: SEMANTIC MODELING AND RECOGNITION OF EVENTS FROM DATA STREAMS

OM PRASAD PATRI
Department of Computer Science, University of Southern California
3740 McClintock Ave, Los Angeles, CA 90089
patri@usc.edu

ANAND V. PANANGADAN
Department of Computer Science, California State University Fullerton
800 N. State College Blvd, Fullerton, CA 92381
apanangadan@fullerton.edu

VIKRAMBHAI S. SORATHIA
Kensemble Tech Labs LLP, Gandhinagar, India
vsorathia@gmail.com

VIKTOR K. PRASANNA
Ming-Hsieh Department of Electrical Engineering, University of Southern California
3740 McClintock Ave, Los Angeles, CA 90089
prasanna@usc.edu

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

Detecting and responding to real-world events is an integral part of any enterprise or organization, but Semantic Computing has been largely underutilized for complex event processing (CEP) applications. A primary reason for this gap is the difference in the level of abstraction between the high-level semantic models for events and the low-level raw data values received from sensor data streams. In this work, we investigate the need for Semantic Computing in various aspects of CEP, and intend to bridge this gap by utilizing recent advances in time series analytics and machine learning. We build upon the Process-oriented Event Model, which provides a formal approach to model real-world objects and events, and specifies the process of moving from sensors to events. We extend this model to facilitate Semantic Computing and time series data mining directly over the sensor data, which provides the advantage of automatically learning the required background knowledge without domain expertise. We illustrate the expressive power of our model in case studies from diverse applications, with particular emphasis on non-intrusive load monitoring in smart energy grids. We also demonstrate that this powerful semantic representation is still highly accurate and performs at par with existing approaches for event detection and classification.

Keywords: Event Stream Processing ; Event Ontology ; Information Integration ; Temporal Pattern Mining ; Time Series Shapelets ; Non-intrusive Load Monitoring

1. Introduction

The rise in scale of sensors deployed in the enterprise has led to the need for faster processing of multiple data streams in a variety of real-world applications. Diverse data sources produce streams of operational, maintenance, production and financial data apart from sensor data obtained by historians, monitoring systems and real-time SCADA (supervisory control and data acquisition) systems. From these data streams, "events" of various types are to be detected and acted on within an enterprise's decision making process [35]. It is important to correlate these isolated events across various sectors of the enterprise and obtain a unified view. The increasing complexity of data handling and management, especially for realtime and time-critical data from multiple sources, have led to advanced techniques for enterprise information integration and complex event processing (CEP). CEP techniques [14] are already in use in various realtime enterprise solutions across distributed event-based systems [41, 26].

Semantic data models are useful for knowledge representation and reasoning, contributing towards a unified view of the knowledge base. A comprehensive event model should include event detection, filtering, notification, action determination, context awareness, and escalation mechanisms [12]. To represent, analyze and process events, several *event ontology models* (such as [71, 84] and others shown in Table 1), as well as *semantic complex event processing* (SCEP) approaches (such as [77, 81, 79, 33, 23]), which incorporate Semantic Computing in CEP, have been proposed. An event model refers to a data model for representing events and their relationships to other concepts. A semantic event model is an event model represented in the form of one or more ontologies for reuse and linking to other models.

Most of these event processing models are based on a broad definition for events, classifying *anything which happens* as an "event" [12]. These event models do not work on the raw sensor data by design. This broad scope makes it difficult to apply the approach to a real-world application since only a relatively small number of the possible events are important and need to be processed further. State-of-the-art event models provide a framework to represent and reason about events but the fundamental problem of transforming large-scale sensor measurements to a sequence of only relevant events has not been addressed. Therefore, in practice, using these event processing models requires the detailed definition of each variety of events based on raw data.

Time series analysis methods have traditionally been used to process data streams for classification and identifying anomalies. The areas of event processing and time series mining are intuitively connected, since all form of event processing uses temporal sequences of data elements. However, event models do not work on the raw data directly, they work on events, and they need event definitions to be defined (likely by domain experts). Time series approaches, which work directly on the sensor data have not been incorporated into event models. Incorporating time series analysis for event detection in a comprehensive event processing framework

is challenging since a given semantic event model has specific rules for defining and composing events which do not match the internal representation of a time series classification algorithm. However, recent advances in shape-based approaches to time series analysis [96, 65, 53] provide methods for identifying discriminative subsequences from data which can be composed in a manner similar to how simple events are combined to define complex events. An opportunity therefore exists for using these methods as the basis for identifying relevant events as part of a comprehensive event representation and detection framework. We describe our approach to achieve this combination of developing an event model that is capable of directly processing sensor data.

We extend the Process-oriented Event Model (PoEM) [56] such that a machine learning based time series classification approach can be used as the basis for automatic semantic event detection from data streams. PoEM is a framework for complex event processing that attempts a comprehensive representation of processes, such as those seen in modern industries and organizations. PoEM provides a precise mathematical definition for an "event" which connects events to real-world entities, their properties and timestamps, enabling us to retain only relevant observations. The PoEM model brings together, in a unified framework, the different types of entities that are expected to be present at different stages of an event-processing workflow and a formal specification of the relationships between these entities. PoEM also formally defines the process on what happens in the event processing lifecycle after an event is detected.

We first present a comprehensive overview of Semantic Computing for event processing with an emphasis on complex event modeling approaches. There are various value propositions using Semantic Computing to enhance CEP as exhibited by PoEM, particularly for industrial and enterprise applications. These are delineated below:

- **Integration.** Data from multiple diverse sources can be viewed in a unified manner at the required granularity by using a semantic representation.
- **Interoperability.** Syntactic and semantic heterogeneity between sources can be resolved by using ontological representations events and related concepts. This enables interoperability while providing reasoning and inference capabilities over the data.
- **Dynamism.** Semantics can help bring dynamism to event-based systems. Specific events will trigger specified actions from relevant actors, and semantic approaches can help define correlation between background knowledge, observations and corrective actions.
- **Management by Exception.** Semantic approaches can help us to determine the appropriate priority for a new event, and ensure that critical events are dealt with urgently. This can include dynamic selection of people to be notified of an event, based on its context, and automatic escalation of events in case of non-response.

In this work, we explore and establish a link between event processing and time series analysis. Integrating an automatic time series classification algorithm for event detection into PoEM results in this additional value proposition:

- **Predictive Analytics.** Event-driven systems require queries on data in motion, and Semantic Computing, through a richer representation model for events, can connect event concepts to automatically discovered patterns from sensor data. These patterns can be discovered from the data in a localized manner, and exhibit predictive power for rapid classification of unseen data in the future.

In our approach, we first detect events from sensor data using a shape-based time series classification approach, called time series shapelets [96]. The shapelets method is a machine learning algorithm that automatically identifies critical segments from time series which are discriminative and representative. This method can extract discriminative subsequences from time series without background or domain knowledge, and does not make assumptions on the nature of the input data. We interpret each of these critical segments as simple events for the PoEM model. Thus, this temporal pattern mining approach is a means of learning event definition and detection parameters for the PoEM model from sensor data.

As the shapelet-based approaches do not account for the background, structure, nature or source of data, they are effective for dealing with data from a diverse variety of sources, as usually observed in modern Big Data streams [92, 13]. An approach utilizing shapelet-based time series classification for heterogeneous multidimensional time series sensor data, in the context of Big Data, has been proposed in our previous work in [53]. Further enhancements to the shapelets algorithm, such as Local Shapelets [94], can operate on data streams without needing all of the training time series in memory for shapelet extraction. This enhancement opens up the potential to extend our temporal pattern mining approach for dealing with high velocity Big Data streams. Thus, our approach can be adapted for automatic detection of domain-specific temporal data patterns even for Big Data streams.

We evaluate the combined use of time series classification and event models in different application domains, with particular emphasis on non-intrusive load monitoring (NILM) of household electricity consumption. Smart grids [73, 74, 99] have enabled aggregate electricity consumption to be monitored at very fine time granularity. NILM methods [102] perform energy *disaggregation*, i.e., estimate the electricity consumption of individual appliances from aggregate power and/or voltage measurements. Our experiments are performed using a publicly available dataset, called BLUED [2] containing voltage, current and power measurements for a single family in the United States for one week. We show how the proposed approach can be used to model complex events that go beyond simply switching on/off an appliance (such as blackout events) and represent them in a rich expressive manner in PoEM. Moreover, this expressive power does not come at a cost of lower classification accuracy – our evaluation results for event detection show that the shapelets

approach performs at par with other state-of-the-art classifiers.

The rest of this article is organized as follows. Section 2 describes the motivational use-case of non-intrusive load monitoring. We review related work in Section 3. In Section 4 we investigate how Semantic Computing is needed to enrich various aspects of event processing. In Section 5, we describe the Process-oriented Event Model (PoEM) that will be extended with machine learning based event detection; an earlier version of this model was presented in [56]. Section 6 presents the PoEM semantic web ontology and three case studies where PoEM is applied for practical event processing scenarios. In Section 7, we describe how to incorporate temporal pattern mining using time series shapelets into PoEM. We also quantitatively evaluate the combined semantic event definition and detection approach for the NILM use case. We conclude in Section 8.

2. Motivation: Non-intrusive Load Monitoring

With the rise in the number of sensors and instrumentation, modern electricity grids have become Smart Grids, serving as a source of energy use data, often at a high temporal frequency. However, energy use in households is typically measured in aggregate numbers. It is more useful to report appliance-level information at a finer granularity to consumers, so they can take appropriate steps for energy conservation by managing individual appliances based on their power use patterns. The broad area of nonintrusive load monitoring [98], which includes approaches such as energy disaggregation [20], has drawn increasing attention from researchers in recent times. Energy disaggregation refers to methods that break down aggregate energy consumption into appliance-level itemized measurements without any explicit plug-level sensors.

Existing approaches [70, 3, 20, 22] for energy disaggregation include methods such as estimation of individual appliance usage by differentiating the loads of various appliances and identifying 'signatures' [22] associated with most existing consumer electronic appliances. These signatures can be measured by special sensors, along with analysis of the current-voltage patterns. This commonly used approach has several drawbacks though, as (i) installing specialized hardware is costly and time-consuming, (ii) the number of appliances in use in households is large and diverse along with variations in human usage patterns (iii) the signature of an appliance can vary over time depending on its mode of operation (e.g. washing machine operating as washer or dryer), and most importantly, (iv) if a new appliance is added to the household, this approach won't be able to learn its signature automatically. In this work, we aim to mitigate all of these drawbacks by using a machine learning approach aided by an interpretable semantic model.

The energy disaggregation problem in a typical household poses a challenging framework for Semantic Computing and machine learning. The Semantic Computing challenge is to develop a rich representation for all the objects (i.e. individual appliances or the household as a whole), events (i.e. appliance switching on/off,)

and processes involved (i.e. sending notifications in case of a power blackout). A better semantic representation for all actors and objects involved will undoubtedly aid the machine learning portion, which is to automatically detect appliance-level events from aggregate data.

Even though the event processing and semantic web communities have focused on many smart grid related applications, there has been surprisingly little work to relate them to the energy disaggregation scenario. In this work, we propose a comprehensive model for capturing the semantics of events and event processing, and illustrate how the model can be driven by advanced machine learning approaches to automatically find patterns from temporal power consumption data. But first, we investigate in detail the need for semantics in various common aspects of event processing, and also compare how some of the existing works try to address these issues.

3. Related Work

In this section, we review and compare related work on event models, rule-based systems, semantic event processing and non-intrusive load monitoring.

3.1. *Event Models and Semantics*

A broad survey of existing event processing approaches can be found in Cugola and Margara [14]. Usually, events are modeled as data tuples consisting of attributes, values and timestamps, such as the 3-tuple by Voisard and Ziekow [85], the *(sensor-id; reading; timestamp)* tuple by Zhou et al. [99] or XML schema [8]. The E* event model [21] is useful for modeling multimedia events. A probabilistic event model for processing uncertain events is proposed by Wasserkrug et al. [89]. Hinze [25] proposed identifying event profiles and constructing an event algebra. Event algebras were also proposed by Zimmer and Unland [101], Eckert et al. [16] and Anicic et al. [4].

Many event frameworks are tightly coupled with a specific domain (such as CIDOC CRM[a]) and cannot be applied to other domains easily. On the contrary, generic models such as the Event Ontology[b], E* [21] or LODE [71] mainly provide a model for events and not the complete event processing workflow. They are not able to incorporate complex events, entities, actions, roles and inter-relationships between event and non-event concepts. Chandy [11] proposed ideas for an event model based on adapting existing computation approaches for 'data at rest' to 'data in motion'. The Simple Event Model (SEM), proposed by Van Hage et al. [84], can be used to model events in various domains without making assumptions about domain-specific concepts. However, these approaches do not provide a model for describing an end-to-end workflow, moving from data streams to detecting and

[a] `http://cidoc.ics.forth.gr/OWL/cidoc_v4.2.owl`
[b] `http://purl.org/NET/c4dm/event.owl`

Table 1. Existing Semantic Event Models

Domain-dependent Models	**Domain**
ABC Ontology [32]	Digital Libraries
Card Ontology [34]	Smart-card Systems
CIDOC CRM [15]	Museums/Libraries
Event-Model-E [90]	Multimedia
Geospatial Event Model [91]	GeoSpatial
Oil Well Ontology [100]	Oil Fields
Event Ontology [63]	Network Diagnosis
Snap Event Ontology [c]	News Events
Independent and Generic Models	
CEPAT Ontology [77]	
DOLCE + DnS Ultralite (DUL) [d]	
Event Ontology [e]	
IPTC EventsML-G2 [f]	
EVO-Core [33]	
Event-Model-F [67]	
LODE [71]	
OpenCyc [36]	
Simple Event Model (SEM) [84]	
Upper Event Ontology (UEO) [30]	

responding to events. They also do not identify the key real-world concepts (how many and which) to define events and the possible methods of transition from simple to complex events.

We provide an overview of existing semantic event models in Table 1, and highlight whether the model is intended for a specific domain (if so which one), or is an independent generic upper-level model.

3.2. *Semantic Rule-based Systems*

Rule-based systems are closely related to event processing. The area of active databases [46] has explored the use of triggers for rule-based processing of data and events in databases. Snoop [10] is an example of a system defining an event specification language for active databases. A further extension of Snoop, called SnoopIB [1] extends this specification to account for interval-based semantics.

Event-condition-action (ECA) rules [46, 43, 64] are a simple approach to modeling rule-based event processing systems. An ECA rule is of the form: *ON Event IF Condition DO Actions*, and they can be used in conjunction with CEP systems. For instance, these ECA rules can be hardcoded into an event processing system

to detect events from real-time data streams, as well as discover complex events occurring from a combination of simple events according to a rule-based template.

Many production rule systems have a design which performs inference based on the RETE algorithm [19]. This is a pattern matching algorithm which decides which of the system's rules should be triggered based on its data store. An enhancement on the RETE algorithm, known as TREAT [37] was also proposed, and a comparison between the two for testing database rule conditions can be found in [88]. However, the RETE or TREAT algorithms do not provide concepts of time-stamped events and temporal constraints between events. An extension of the RETE algorithm for this purpose was proposed by Berstel [7]. Enhanced approaches integrating RETE-based ECA rules along with a specialized event detection system were proposed by Schmidt et al. [68] and Walzer et al. [86].

The addition of semantic web techniques to rule-based systems makes them reactive, i.e. able to automatically execute certain rules in an application based on event or condition triggers. An ECA rule language for operating on a graph/triple representation of RDF was proposed by Papamarkos et al. [43]. RuleML [9], a family of semantic web rule markup languages, was developed to facilitate the exchange of rules across various systems on the world wide web. A semantic web rule language combining OWL and RuleML, denoted as SWRL [29] was also proposed. Reaction RuleML [44] is a particular branch of the RuleML family, which provides a standardized interchange format for reaction rules and semantic rule-based event processing [80]. ETALIS [4] is a rule-based event stream processing system.

While semantic rule-based event processing approaches provide an efficient approach to manage automatic execution of rules based on event-based triggers, they still require definition of the rules by domain experts. Our work aims to bridge this gap by automatically discovering relevant domain rules directly from sensor data streams using a time series based machine learning approach.

3.3. *Semantic Complex Event Processing*

Semantic complex event processing (SCEP) approaches have been used in diverse applications comprising a variety of complex events including ride sharing events [42], stock market events [83], security and threat detection events [23], user interface integration [60], RFID data integration [87], sensor networks [79], process management systems [31], ubiquitous logistics [66], smartgrids [75], oil well management [100], and E-health and ambient assisted living [95]. Many enterprise information management systems have actively pursued the use of semantic web technologies to integrate information across diverse sources. Semantic Computing approaches have also been used to integrate information from various sources including natural language/text, numeric data, structured data and others for applications such as situation awareness [82], smart grid load demand-response [99] and trip planning [54, 55].

We now briefly describe a few of the existing SCEP approaches, with a de-

tailed comparison between them in Table 2. Stojanovic et al. [77] propose a logic-programming based approach for semantic CEP using the ETALIS event processing language. They describe a rich set of temporal composition operators (such as *sequence, intersection, starts, finishes, equals, meets* etc.) for detecting complex events from atomic events, yet it is not made clear how concepts from the event ontologies can be utilized in this detection process. They also provide rules for detecting complex events through temporal reasoning over atomic events. Teymourian et al [81] focus more on the ontology and its modular structure for enabling CEP. They propose semantic enrichment of event streams, in which derived events are added to the event stream in addition to the observed events; however, such enrichment can also be used for correlating events or adding more context to atomic events from the event stream, without necessarily adding derived events.Taylor et al [79] use event-based ontologies mainly for defining complex events in the domain, and thus, as a guide to complex event processing. Hammar [23] proposes the concept of *observation correlation*, which refers to the CEP systems detecting which observed situations are potentially interesting only when co-occurring. Zhou et al. [99] propose to incorporate semantic knowledge for event processing in smart grids. Liu et al [33] desire to harness the power of Linked Data on the web in a CEP engine, and their model is based on the generic EVO-Core [62] event ontology. EVO-Core, while capturing some characteristics of events, still does not provide a comprehensive model for the various functions of event processing, including detection, filtering, notification, action determination, prediction, and escalation.

3.4. *Non-intrusive Load Monitoring*

NILM approaches for energy disaggregation are surveyed in [98] and [102]. Though several techniques for NILM disaggregation have been proposed, we frame the problem in the context of time series classification. ElectriSense [22] uses electromagnetic interference (EMI) signals during appliance operation to identify and classify individual appliance use. Gemello [5] provides a machine learning approach for generating a more fine-grained electricity bill for a household (from aggregate data) by comparing to similar households. Anderson et. al. [3] survey different approaches for event detection, broadly classified as based on expert heuristics (such as [18]), probabilistic models (such as [6]) or matched filters (such as [72]). However, none of methods take a time series based approach. Shao et al. [69] mine for temporal motifs from energy consumption time series; however, they do not work with labeled data or extract discriminative features. Motifs are frequent patterns but not discriminative like shapelets.

4. Semantic Computing in Event Processing

In this section, we investigate how Semantic Computing is needed in various aspects of event processing tasks. We also survey existing Semantic Computing based event

processing systems on the extent to which they fulfill these aspects. This comparison motivates our proposed semantic event model, described in the next section.

4.1. *Event Detection Semantics*

Detection of complex events involves monitoring multiple heterogeneous data sources, and checking certain conditions to see if an event has occurred. Event profiling is commonly used to capture the relevant details for event detection. Event profiles [27] define a predefined condition, which, if satisfied leads to detection of an event. For a simple event, profiling involves monitoring conditions and observation values to check if they are in a certain critical range (defined as an event). For complex events, profiling involves dealing with correlated events or co-occurrence of events. Semantic Computing can improve event detection and profiling by reducing ambiguity in the meaning of data, and providing detection rules based upon instantaneous values of data. Semantic techniques also improve expressivity and reasoning power of the system.

Even though a large amount of existing work on event processing focuses on event detection, there are only a few approaches which propose semantic event detection, such as Stojanovic et al. [77], Teymourian et al. [81], Hammar et al. [23] and Zhou et al. [99]. Moser et al [38] propose related approaches for semantic event correlation (connecting related events and eliminating duplicates). We propose to enable semantic event detection by incorporating concepts from our proposed event model into an event ontology. Then, instances from the ontology can be used for reasoning whether certain conditions are satisfied, and certain property values fall in the critical range (for an event to occur). By using an integrated ontological repository as our data store, event correlation and elimination of duplicates is possible.

4.2. *Event Filtering Semantics*

Filtering of relevant events is another functional requirement for event processing. Filtering is necessary to eliminate large portions of event related data and only focus on the portions which are relevant to a certain context. Event filters may perform filtering based upon event type, event priority or some other context. This context can be temporal, spatial, segmentation-oriented or state-oriented as enumerated in Section 4.3. For instance, in an event data stream which contains information about all automobiles in the city, a user may be interested only in the portions related to vehicles owned by her, or only those vehicles which are currently located within 10 miles of the vehicle she is located in.

Since an integrated semantic knowledge base includes information from several data sources, it can provide a wider scope for the basis of event filtering. Semantic methods can increase the expressivity and reasoning power of event filtering by using powerful semantic (SPARQL) queries. In Table 2, Complex Event Filtering refers to whether the SCEP system implements some sort of filtering for complex

events. Semantic Event Filtering refers to the criteria that event filtering is driven by semantic concepts/rules. We also report on whether there is some special treatment for critical/high priority events, which can be filtered by assigning priorities to events and filtering those with the highest priority. Such critical event detection is primary to many practical applications such as threat detection [23]. In many industries, this concept of 'management by exception' is a key indicator of usefulness of the technology deployed.

4.3. *Context in Event Semantics*

The role of context in event processing systems has been explored in detail by Etzion et al [17]. Four types of context are identified: temporal context, spatial context, segmentation context and state context. Most current SCEP systems implement temporal parameters and aspects, but spatial context is not found to be present in many works. Segmentation and state contexts are also not found in several state-of-the-art systems.

4.4. *Event Notification Semantics*

Notification about events is a major constituent of CEP systems. Usually such notification is sent in the form of alerts/triggers to human agents, so that they can execute further actions and decide the future course of action. Sometimes, these triggers may lead to automatic execution of certain actions in the case of software agents. The CEP system is responsible for deciding the content of the notification message, method of notification (e.g. SMS/email), to whom the notification should be sent (subscriber list) and how to route the notification message. The message content can contain relevant information regarding the event, such as instantaneous and historical data values, various timestamps (when event occurred, when it was detected, when it was reported), actions related to event (what actions have been taken already, what are best practices associated with such event, what actions still need to be taken), processes associated with event (which processes must be rescheduled in order for action associated with this event to take place), entities related to event (people who are involved, people who can act as experts about this event, resources which are involved, agents which must be instructed to take further actions). Advanced CEP systems should be able to dynamically build the message content as well as subscription lists for events based on the event context, possibly benefiting from background knowledge provided by a semantic model.

Notifications may need to be repeated, or sent iteratively with a certain frequency in many cases. This frequency can be determined from background knowledge as well as the user's preferences. We propose to implement the event escalation pattern for CEP. For instance take the example of a patient whose health and vital stats were being monitored by a CEP system. Assume that the patient needs to be administered a certain medicine by the nurse when her heart rate is found to be abnormally high. At a certain point of time, this event occurred and the nurse in

charge was notified. However, she does not see the message or respond for a certain time. In such a case, the doctor in charge of the patient, or other hospital staff can automatically be notified (probably based on their proximity to the patient). This capability to wait for a response to an action for a suggested time, and if there is still no response, implement a pre-defined action, is known as event escalation. Notification rules can use concepts from the ontology, and the ontology needs to make provisions for efficient notification processes.

Event escalation is another useful feature in a CEP system, and we discuss it in detail in Section 5.5. The role of the CEP system does not end with determination of what action should be taken for which event, the system needs to have a 'feedback' loop to verify that the action was actually performed. Event escalation provides this feedback loop. Existing semantic models do not incorporate event escalation.

4.5. *Action Semantics*

After an event is detected, it is important to take the right action for that event, since the usefulness of event processing is in reacting to events quickly. Based on semantic knowledge and historical data, actions can be suggested and best practices can be listed for a certain event. These actions can be included in the event notification message. Actions to be suggested could be reporting actions (such as logging or monitoring), or correcting actions (such as repair or maintenance). Actions can be driven by semantic concepts and rules. Best practices discovery is very useful in enterprise systems where finding expert advice can be costly. These best practices can be integrated with business rules or company-specific practices.

4.6. *Prediction Semantics*

Predicting future event occurrences accurately based on historical data and recent trends is important for an effective CEP system. Semantics can help in providing relevant background knowledge to serve as training data for a prediction system. Prediction involves predicting simple events, complex events, and situations which are inferred from these events. Discovery of event patterns and adding them to the knowledge base for future reference is also a desired functionality. In this work, we focus our efforts on the detection of representative subsequences (corresponding to events) from time series sensor data. The patterns detected by our approach have predictive power to classify future time series data quickly into one of several previously seen categories.

We compare several aspects of these types of identified semantics in some existing SCEP systems based upon the context of event-based systems in Table 2. The approaches compared are (1) Stojanovic et al. [77], (2) Teymourian et al. [81], (3) Taylor et al. [79], (4) Liu et al. [33], (5) Hammar [23], and (6) Zhou et al. [99].

Table 2. Comparison of event-related semantics enabled in some existing SCEP approaches

Approach	[77]	[81]	[79]	[33]	[23]	[99]
Overview of Capabilities						
Semantic CEP Engine	✓	✓	×	×	–	✓
Semantic Enrichment/Annotation	✓	✓	✓	✓	✓	✓
Generic Event Ontology	✓	✓	×	✓	×	×
Isolated Domain Ontology	×	✓	✓	✓	×	✓
Connection to Linked Data	×	×	×	✓	×	×
Semantic Queries (SPARQL)	✓	✓	✓	✓	✓	✓
Detection Semantics						
Semantic Event Detection	✓	✓	–	–	✓	✓
Complex Event Detection	✓	✓	✓	✓	✓	✓
Event Pattern Detection	✓	✓	–	–	✓	✓
Semantic Event Correlation	×	×	×	×	✓	×
Event-driven Detection Rules	✓	✓	✓	✓	✓	✓
Filtering Semantics						
Semantic Filtering/Inferencing	✓	–	×	×	✓	✓
Complex Event Filtering	✓	✓	–	×	✓	✓
Event Priority Assignment	×	×	×	×	✓	×
Event-driven Filtering Rules	✓	✓	✓	–	✓	✓
Context Semantics						
Temporal Context	✓	✓	–	✓	✓	✓
Spatial Context	×	✓	×	×	✓	✓
Segmentation Context	×	×	×	×	×	×
State Context	×	×	×	×	×	×
Notification Semantics						
Semantic Triggers/Alerts	✓	✓	✓	–	✓	✓
Dynamic Message Content	×	×	×	×	×	×
Dynamic Subscription Lists	×	×	×	×	×	×
Event Escalation	×	×	×	×	×	×
Action Semantics						
Semantic Action Selection	–	✓	×	×	×	×
Event Escalation	×	×	×	×	×	×
Best Practices Discovery	×	×	×	×	×	×
Prediction Semantics						
Pattern Discovery from Sensor Data	×	×	×	×	×	×
Incorporating Time Series Mining	×	×	×	×	×	×

5. Design of the Process-oriented Event Model (PoEM)

Based on our observations about the requirements in a semantic event model and the gaps in existing work, we propose a new event processing model called the Process-oriented Event Model (PoEM), a prior version of which was presented in [56]. In this section, we describe key aspects of the model. PoEM is a generic domain-independent model and can be rapidly adapted for new domain-specific applications easily.

5.1. *Core Concepts*

Several key components of PoEM are based on foundational concepts from the dynamic information management methodology proposed by Sorathia [76]. This modeling approach, originally proposed for situational awareness, provides an effective framework to build a conceptual event model.

Our model aims to incrementally add context to raw data values as the data values move from sensors to events – first from a data stream to a measurement value, then from a measurement value to an observation, and finally from an observation to an event. We begin with raw data values from the data stream. Measurements add the context of unit of measure, measurement type, and bounds to the data values. Observations are abstractions of measurements which deal with instantaneous measurement values, and also have a temporal context associated with them (about when the value ceases to be valid). Events add further context to observations by only including those observations which have a corresponding match in an event profile and are relevant for future processing. Complex events are generalizations of simple events and include multiple events. Each contextual addition or enrichment can be expedited by using a semantic knowledge repository. Figure 1 outlines this process. We now define the related terminology.

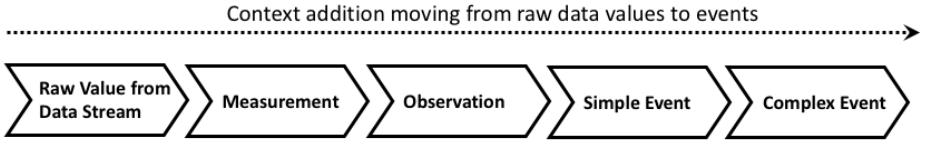

Fig. 1. Moving from raw sensor values to events in the PoEM model, some incremental context being added at each step along the way

5.1.1. *Entity*

Entities are basic elements in the conceptual model, and may be physical or logical. We represent the set of entities in our universe of discourse as K while individual entities are represented as κ. In our NILM motivational scenario described earlier, entities would include various appliances such as refrigerator, heater and lights, as well as human entities residing in the house.

5.1.2. *Observable Property*

Each identified entity may have several *properties*. The next step is to identify and enumerate the set of relevant *observable properties*. The set of relevant observable properties is represented by Π, which includes individual members denoted by π. Observable properties may range from detecting the mere presence of an entity to various physical and chemical properties requiring sensing techniques.

Observable properties can be grouped by the chemical, physical, and other scientific methods used for measurement, and they can be measured in different ways. To accurately capture properties associated with entities, we need a model for *measurement*.

5.1.3. *Measurement*

Observations can be recorded in various ways. For instance, temperature is an observable property measured using digital and analog sensors (thermometers), which may provide readings in degrees Celsius or Fahrenheit. Also, sensitivity of the measuring device may vary significantly affecting precision and accuracy. Some sensors provide a continuous stream of readings, whereas others do so at intervals. As subtle changes in observable properties could be critical in high reliability operations, the modeling procedure must comprehensively handle these aspects for which we propose the concept of *measurement*. A set of measurements is denoted by M while an individual measurement is shown as μ. A measurement record captures the value of a sensor reading, along with its unit of measure (UoM), type, and other details related to measurement. This representation could include additional features related to precision, accuracy or sensor update frequency. However, once such information is identified for a specific sensor, it remains static. Therefore, a measurement model leads to a stream of readings stored as observations.

5.1.4. *Observation*

Measurement concepts enumerate all possible ways the observable properties can be measured. However, in order to interpret the current status, it is useful to retrieve instantaneous values from the sensor. This is achieved by introducing the concept of *observations* (Ω), which is the set of all measurement observations. Individual observations (ω) can be recorded at time t, for a given entity κ as per a

specific measurement model μ. The measurement value associated with an observation at time t_i will frequently be denoted by ω_i^μ in this paper for convenience. From the measurement model (μ_i), property π, UoM and other relevant context can be determined. Sensors typically provide such values in the form of data streams.

5.1.5. *Data Stream*

Sensors reporting values as per specific observations result in continuous *streams* of data. We define Θ as the set of all data streams available for a given entity with individual instances of streams represented as θ. A specific data stream (θ) is linked with observations (ω) for a specific entity (κ) according to a specific measurement model (μ) that determines the type of sensors, UoM, and other relevant details.

5.1.6. *Interpretation*

Raw data values recorded and reported by sensors do not provide insights unless they are evaluated in a specific context. For instance, a thermometer reporting $25°C$ for a tool room might be a standard condition, however the same value for a cold storage facility may be an exception. We introduce the concept of *interpretations* to resolve this issue. An interpretation provides meaning to the recorded values. Users can plug in their own event detection rules and filters here. For instance, for a given observation ω_t, the recorded value may fall into a range that might be *normal* or *critical*. This directly provides contextual information about the entity. Therefore, it is useful to identify such ranges of values that makes them *critical*. Interpretation set (denoted by X or X^μ) is a set of all possible interpretation instances (χ) that can be identified for a specific measurement model (μ) associated with an observable property.

For instance, $\chi_1^{[a,b]}$ is the interpretation provided when the value of ω_{t_1} is in the interval $[a, b]$. Similarly, domain experts and practitioners can provide all the ranges and respective interpretations. An interpretation set for a sensor measuring pH of water may be represented as:

$$X_{\text{pH}} = \begin{cases} \text{Failed} & : \text{Sensor reading N/A} \\ \text{Invalid} & : (\text{pH} < 0) \cup (\text{pH} > 14) \\ \text{Acidic} & : 0 \leq \text{pH} < 7 \\ \text{Neutral} & : \text{pH} = 7 \\ \text{Basic} & : 7 < \text{pH} \geq 14 \end{cases} \tag{1}$$

Each interpretation (χ_i^μ) is associated with an evaluation condition, forming a branch (i) of the set (X). As seen from the above example, there may be a branch dedicated to find out whether the sensor is functioning and providing values in the first place. Placing this branch at the top of the interpretation set may lead to faster detection of failures, in case no data is being read. The evaluation condition is usually a function of the observation (ω), and is denoted as $c(\omega)$. Then, an

interpretation set (X) can be defined as:

$$X^{\mu} = \bigcup_{i=1}^{m} \{\chi_i^{\mu} \mid c_i \implies \chi_i^{\mu}\} \tag{2}$$

which, when expanded, leads to Equation 3.

$$X^{\mu} = \begin{cases} \chi_1^{\mu} & : \text{if } \mathrm{c}_1(\omega) = \text{true} \\ \chi_2^{\mu} & : \text{if } \mathrm{c}_2(\omega) = \text{true} \\ \dots & \\ \dots & \\ \chi_m^{\mu} & : \text{if } \mathrm{c}_\mathrm{m}(\omega) = \text{true} \end{cases} \tag{3}$$

For each measurement model μ, there is at least one interpretation set delineating all possible interpretations which are related to values of the given property. The branches of the interpretation set are non-overlapping and only one of the branches is activated during evaluation. Individual interpretation sets can be defined based on piece-wise functions, boolean functions or other suitable mathematical or logical representations recommended by domain experts or practitioners. With identification of all relevant interpretations that lead to simple or complex events of interest, it is possible to define the *event space* as the set of all possible events that can occur in the given universe of discourse.

So far, we have focused on a single observation value determined at a specific point. However, there can be additional interpretations derived for multiple observations as well. Considering readings at multiple time instances enables representation of complex scenarios such as a sudden increase or decrease in property measurements. For instance, temperature readings of $31°C$ at t_1 and $36°C$ at t_2 may be considered *normal* as they both belong to the *normal* range of $[30-37]°C$. However, two consecutive readings indicating an increase in excess of 10% may be a *critical* change according to a rule in a specific domain.

$$X_t^{\mu} = \chi^{\mu} : \text{if } \Delta(\omega_{t_1}, \omega_{t_2}) \geq 10\% \tag{4}$$

Through a flexible specification for the interpretation of observations (which lead to events), we make our model able to be easily adapted and expanded for new domains, and facilitate using a variety of existing data mining and machine learning models to build the interpretation set. One interesting realization of the model is embodied in Section 7 where we propose a method to directly learn a new representation for the branches and conditions of the interpretation set from raw time series data using shapelet-based methods.

5.2. *Event Concepts*

The core real-world concepts proposed earlier provide a mechanism to identify key entities, attributes, measurements, and interpretations. This lays the foundation for

identification of events. The interpretations link the current measurement to identification of changing situations. Therefore, interpretations directly link to identification of events and related concepts.

5.2.1. *Event*

In conventional event definitions, every change is declared to be an event. Therefore, as per our model, all recorded interpretations should result in events; however, this leads to an exponential number of events. For instance, even if a temperature reading is recorded to be normal, any minor change in temperature will be reported as a new event. To address this issue, we introduced the concept of interpretation for specific ranges. In addition, changes in interpretation may not be relevant to the user's interest. For example, only extreme variations in temperature might be of interest. Taking these factors into account, we propose a new definition for an event that identifies a subset of interpretations to be considered as events. From all interpretations in an interpretation set, any one can be true at given point in time (τ), which leads to identification of a *simple event* (e_s).

Definition 1. *An event is the interpretation of an observation of interest. Conceptually, a simple event is represented as*

$$e_s = \chi_{\omega^{\kappa,\pi,\tau}} \tag{5}$$

where e refers to an event, τ is the time stamp of the event, κ is the entity associated with the event, π is the observable property associated with the entity for which this event was recorded, and χ is the chosen interpretation of the event.

In the NILM use case, a reading of 20 Volts from a refrigerator appliance may cause an "Low Voltage" interpretation and lead to detection of a simple event. The entity of interest here is the refrigerator ($\kappa_{refrigerator}$), the observable property is voltage ($\pi_{voltage}$) and the observation (ω_{t_1}) is the measurement (with scope) recorded from the sensor at a particular timestamp t_1. The observation ω_{t_1} was 20 Volts, which, by referring to the interpretations for an appliance leads to the identification of a simple event (e_1). This example is shown in Figure 2.

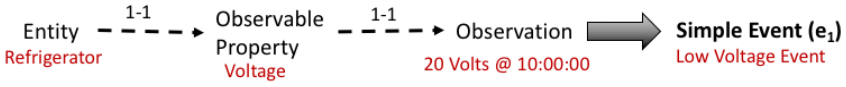

Fig. 2. A 'Low Voltage' simple event observed for a refrigerator. Note the 1-1 correlation between entity, observable property, and observation. A simple or atomic event in PoEM is defined as an event which involves exactly one entity, one observable property and one observation.

5.2.2. *Event Profile*

As defined by Hinze [25], the concept of an *event profile* typically involves a query that is used to determine if a specific event has occurred or not. We can use interpretations and simple events to determine the event profile. Event profiles suggest what needs to be queried to determine the occurrence of an event.

For a simple event, the event profile consists of just an observation that can be performed once or repeated at specific time intervals. Equation 6 indicates a simple event profile that requires a single (one-time) observation of property π for an entity κ at time instance τ.

$$P_s^o = \omega^{\kappa,\pi,\tau} \tag{6}$$

However, in real world scenarios, the requirements can be more complex. An event profile may constantly need to be evaluated at a specific time interval $\Delta\tau$. In such scenarios, a recurring event profile P_s^r can be represented as in Equation 7. An example of an event profile to check the voltage of an appliance every 30 seconds is also shown.

$$\begin{aligned} P_s^r &= \omega^{\kappa,\pi,\Delta\tau} \\ P_s^r &= \omega^{\text{appliance,voltage,30}sec} \end{aligned} \tag{7}$$

5.2.3. *Complex Events*

In real-world applications, it may not be sufficient to determine the occurrence of an event by evaluating a single property at a specific time instance or over multiple time intervals. It may involve multiple entities and their properties evaluated at different times, and thus, require detection of *complex events*, which are abstractions of multiple simple events. The foundational concepts discussed so far provide us a way to represent various types of complexities leading to a complex event.

A simple event involves one entity, one property and one observation. A complex event occurs due to multiplicity in the number of observations, properties, entities or any combination of the above. Thus, a novel contribution of our model is that it quantifies that *there are only four simple ways to combine simple events to form complex events*, in contrast to numerous existing works which fail to provide a finite number of ways of combining simple events to form complex events. These four ways are enumerated below, and an illustrative example with some of these scenarios is shown in Figure 3.

(1) *Multiplicity in Observations*
Multiple observations related to the same entity and observable property taken at different times can lead to a complex event, the profile for which may be represented as follows.

$$P_c^o = \omega^{\kappa_1,\pi_1,\tau_1} \wedge \omega^{\kappa_1,\pi_1,\tau_2} \tag{8}$$

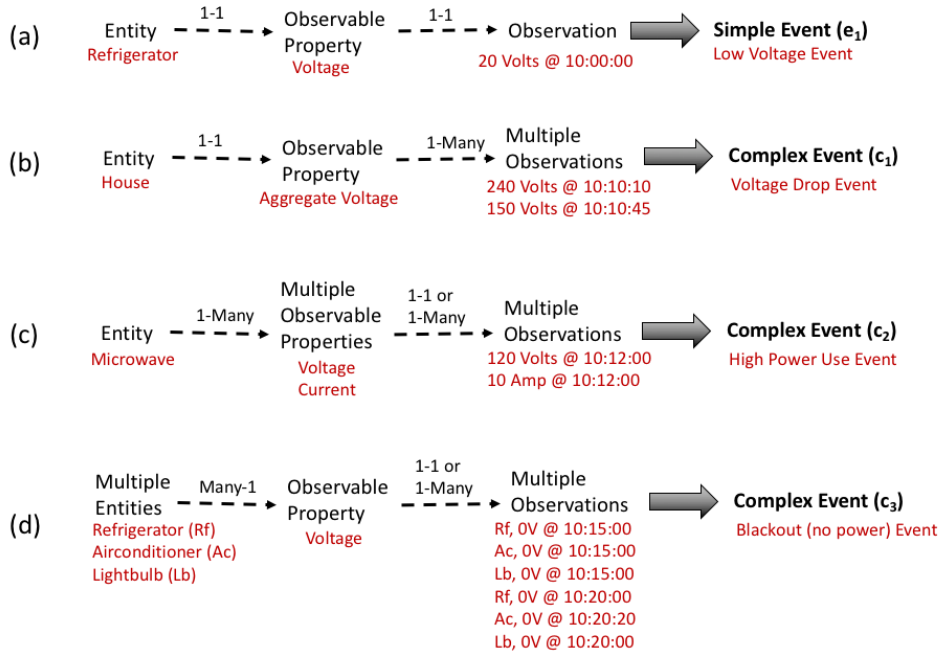

Fig. 3. In PoEM, moving from simple events to complex events can be done by one of these four ways - having a multiplicity in observations, entities, observable properties or a combination of the above. This is in stark contrast to numerous existing works which are unable to define a finite number of ways to move from simple to complex events in a real world system.

For instance, the sequence of two atomic events occurring within a temporal distance is a complex event. Note that events over moving windows are captured in this category.

(2) *Multiplicity in Observable Properties*
The next level of complexity might occur via different observable properties associated with the same entity, as shown by the profile below.

$$P_c^o = \omega^{\kappa_1,\pi_1,T_1} \wedge \omega^{\kappa_1,\pi_2,T_1} \tag{9}$$

(3) *Multiplicity in Entities*
The involvement of multiple entities, which may of the same or different type, lead to complex events. An event profile involving two different entities may simply be shown as the intersection of the two related simple events, as in 10.

$$P_c^o = \omega^{\kappa_1,\pi_1,T_1} \wedge \omega^{\kappa_2,\pi_2,T_1} \tag{10}$$

(4) *Combination of any of the above*
A combination of multiplicities in any of the above three criteria can cause a

Table 3. Common Event Composition Operators

Operator	**PoEM Representation**
Conjunction	$e_i \wedge e_j \wedge \ldots = \bigcap_{i=1}^{n} e_i$
Sequence	$[e_i; e_j]_T$
Disjunction	$e_i \vee e_j \vee \ldots = \bigcup_{i=1}^{n} e_i$
Negation	$\bar{e_i}$

complex event.

Considering the above criteria, a definition for complex events can be specified similar to our definition of simple events with the additional accounting of sets of entities, properties and timestamps instead of a single value.

Definition 2. *A complex event e_c is the interpretation of an observation of interest which depends on multiple simple events. It may be represented as:*

$$e_c = X_{\omega^{K,\Pi,T,}} \tag{11}$$

where e_c refers to the complex event, T is the timestamp of the complex event, K is the set of entities associated with the complex event, Π is the set of observable properties associated with the complex event, and X is the interpretation of the complex event. If the simple events constituting the complex event are $e_1, e_2, \ldots$ as defined in Equation 1, then $K = \bigcup_{i=1}^{n(entities)} \kappa_i$, $\Pi = \bigcup_{i=1}^{n(obsproperties)} \pi_i$ and T is the timestamp of the last observation related to the complex event $T = \max\{t_i\}$ if t_i are timestamps of all related observations within the complex event. X is an interpretation which is based on the individual simple event interpretations through a complex event profile.

In order to obtain complex events, we usually perform a composition of simple events using logical or temporal operators such as conjunction, disjunction, negation, or sequence. The algebraic semantics of complex event operators have been studied in existing approaches for combining events and forming rules [4, 16, 25]. Temporal operators are instances of a broad class of contextual operators, and other contexts, such as a spatial context, could be used instead. Table 3 lists some popular complex event operators and their notations in our model.

We provide a couple of examples of representing complex events based on the definitions and the operators introduced.

- A sequence of events within a temporal gap of t

$$e_{sequence} = e_i \wedge e_j \wedge (\mid \tau_i - \tau_j \mid < t)$$

- Voltage drop of more than 25% recorded for heater within 30 seconds

$$\begin{aligned} e_C = e_i \wedge e_j = \chi_{\omega_i^{heater,voltage,\tau_i,}} \wedge \chi_{\omega_j^{heater,voltage,\tau_j}} \\ \wedge(i \neq j) \wedge (\tau_j - \tau_i < 30 \text{ seconds}) \\ \wedge(\omega_j^{\mu} - \omega_i^{\mu} > 0.25 * \omega_i^{\mu}) \end{aligned}$$

5.3. *Processing Concepts*

The model described so far captures the static aspects of CEP, enabling us to model entities, properties, and their interdependence. However, these concepts are applicable at design time only. In a practical scenario, these rules should be applied over incoming streams of data. We provide a brief overview here of processing requirements from event profiles that leads to detection of events, states, and associated actions, roles and processes in the application domain. For a more detailed examination of the State, Action and Role models, we refer the reader to our previous work [56].

5.3.1. *States, Actions and Roles*

Throughout its life-cycle, an entity goes through several states, e.g., a lightbulb may be switched on or off, and may be in working condition or dead after the end of its expected life. We incorporate the states of all associated entities and relationships of states to other concepts in a state model. An entity belongs to exactly one state at any point in time and has one desirable goal state (obtained from background knowledge). States in our model are denoted by ψ for individual state instances or Ψ for the set of states. The concept of states in our model is similar to "entity status" in the E* event model [21].

A possible representation of the state model is through a deterministic finite automaton (DFA). Note that even though state models, such as nondeterministic finite automatons (NFA) in [78], have been used in prior work, they are usually implemented for event detection. In our case, we achieve event detection through our event profiles and interpretations, so the state model comes into play in the post-detection stage, for determining what actions should be taken when an event occurs.

Actions are simply the transitions between states. An entity can move from one state to another when an action is performed on it. A special type of action is event escalation, which can be triggered by the event processing engine. Based on the identified types of actions, it is possible to determine the roles of actors who are required to perform particular actions. For the identified domain actions, a person who is interested or responsible in specific actions and resulting state transitions can be identified for playing the role of an *actor*. We denote an action by α (set of actions: A) and a role by β (set of roles: B).

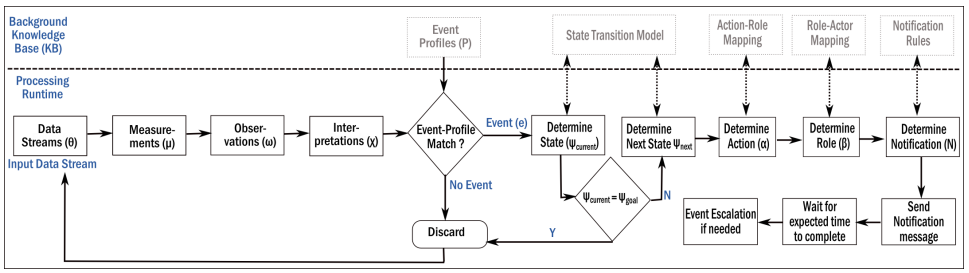

Fig. 4. The PoEM event processing workflow

5.3.2. *Event Notification and Filtering*

When an event (e) is detected, and corrective actions (α) and roles (β) are identified, the next processing step is *event notification*. Here, the CEP system should be able to determine the content of the notification message N shown in Equation 12. Event notification is typically limited to core event detection information but we introduce additional background information to enrich the notification message. For instance, each corrective action (α) can be associated with an expected time for completion τ_{comp} (or other best practices). This not only provides guidance to the actor but also enables a mechanism for validation of the notification. Semantic background knowledge can also be used to dynamically decide who should be subscribed to what type of events.

$$N = \langle \beta, \alpha, \tau_{comp}, \text{event-context} \rangle \tag{12}$$

Filtering mechanisms play a key role in avoiding duplicate or unwanted occurrences of instances to report unique happenings from the deluge of observations. An event profile evaluated at certain time intervals may detect an event that triggers actions, roles and notifications. However, at the next time interval, it might detect the same event again and generate duplicate notifications. Existing patterns for event filtering, as mentioned by Paschke et al. [45] can be used here.

5.4. *Event Processing Workflow*

The complete event processing workflow enabled by the proposed model is depicted in Figure 4. The top half of the figure delineates various elements of the knowledge base (KB) to be looked up by the components involved in event processing at runtime, which are shown in the bottom half. The process begins with reading values from data streams. Simple as well as complex events can be detected by using the respective event profiles and reporting if any interpretations within the event profile are evaluated to be true (event-to-profile match is found, similar to the "event matching" operator in [16]). An event (e) is detected in such a case.

Once an event is detected, the current state ($\psi_{current}$) and goal state (ψ_{goal}) of the entity involved are determined. If they are the same, then the entity is in the desired state, and no further processing needs to be done. Thus, the event and its related information can be discarded (after logging, if necessary). If the entity is not in the goal state, a sequence of states which needs to be traversed to reach the goal state is determined by referring to the state transition model. The first state from this sequence is the next state, and an action (α) leading to the next state is also looked up. After obtaining the action, the action-role mapping is looked up in the KB to determine appropriate roles (β) of actors who can perform the specific action. From the role, particular actors, e.g. employees who need to perform the action, are found. Finally, a notification message is sent to the actors (as well as any other event subscribers), the content of the message being enriched with relevant context. However, the event processing workflow does not end here, since it is unknown if the action suggested was actually performed. In such cases, escalation may need to be performed to ensure that appropriate action is taken and the entity reaches its desired goal state.

5.5. *Event Escalation*

Escalation refers to the case when an event is detected, a notification is sent to an actor to take a certain action but no reply from the actor is received (probable reasons for which may be the actor being inactive, the actor being unable to perform the action or the action being performed but the reply message getting lost). Conventional event models would stop processing the event here, or retry transmission of the action message, and wait till a response is received or someone finds out the root cause for the problem. It would be better if a proactive CEP system could monitor the repair action suggested to the actor, and upon exceeding a certain expected time to reply, would automatically *escalate* the event and perform a set of predefined actions (which could include sending a message to the supervisor of the actor or another actor/team to carry out the repair action, or aborting the event altogether). Such predefined actions can be determined from background knowledge and information about the organization and use case domain. A step-by-step algorithm for an event processing scenario which includes event escalation is depicted in Algorithm 1. The process involves several lookup operations to obtain information from the knowledge base (KB). The algorithm shown assumes a simple event. The detected event has an associated entity, timestamp and property, as defined in Definition 1. The process can be extended to a complex event by replacing the individual instances of entities and properties with their corresponding sets.

5.6. *Limitations of the Model*

The event model presented here is limited to identification of key concepts and relationships between them. It assumes that all data is available at a central location holding decision-making capability. The middleware required to implement

Algorithm 1 Event Escalation Algorithm

$e \leftarrow \chi_{\omega^{\tau,\kappa,\pi}}$
Get $\psi_{current}, \psi_{goal}$
while $\psi_{current} \neq \psi_{goal}$ **do**
  $\psi_{next} \leftarrow$ LookupNextState($\psi_{current}, \kappa$)
  $\alpha \leftarrow$ LookupAction($\psi_{current}, \psi_{next}$)
  $\beta \leftarrow$ LookupRole(α)
  $B \leftarrow \{\beta\}$
  repeat
    Get estimated t_α
    SendMsg($\alpha, \beta, t_\alpha, e$)
    Sleep(t_α)
    $B \leftarrow B \cup$ LookupEscalationRole(B, e)
  until $\psi_{current} \neq \psi_{next}$
  $\psi_{current} \leftarrow \psi_{next}$
end while

such an event processing model has not been discussed in detail and needs to be driven semantically as well. There are various components of background knowledge (some of which are obtained from domain experts) which we need to know prior to using the model. In Section 7, we show how we can learn parameters for the event model automatically from raw sensor data by using an advanced machine learning approach for time series mining. This provides us a way to automatically learn event detection rules without explicit and expert domain knowledge about the application.

6. The PoEM Ontology and Case Studies

We developed an ontology for the PoEM model described above. Each concept translates to an OWL class in the ontology and inter-relationships between concepts are represented by properties. Figure 5 shows an overview of the ontology schema. Having an ontology for PoEM enables us to connect to open linked data as well. For instance, time-related concepts can be borrowed from the W3C Time ontology[g], and geospatial concepts can be reused from the Geonames ontology[h]. It also enables us to compare against existing models and provide cross-links to other models.

We now consider case studies in diverse applications and exhibit the utility of our model by showing the ease of mapping it to the desired application scenario.

[g] `http://www.w3.org/TR/owl-time/`
[h] `http://www.geonames.org/ontology/`

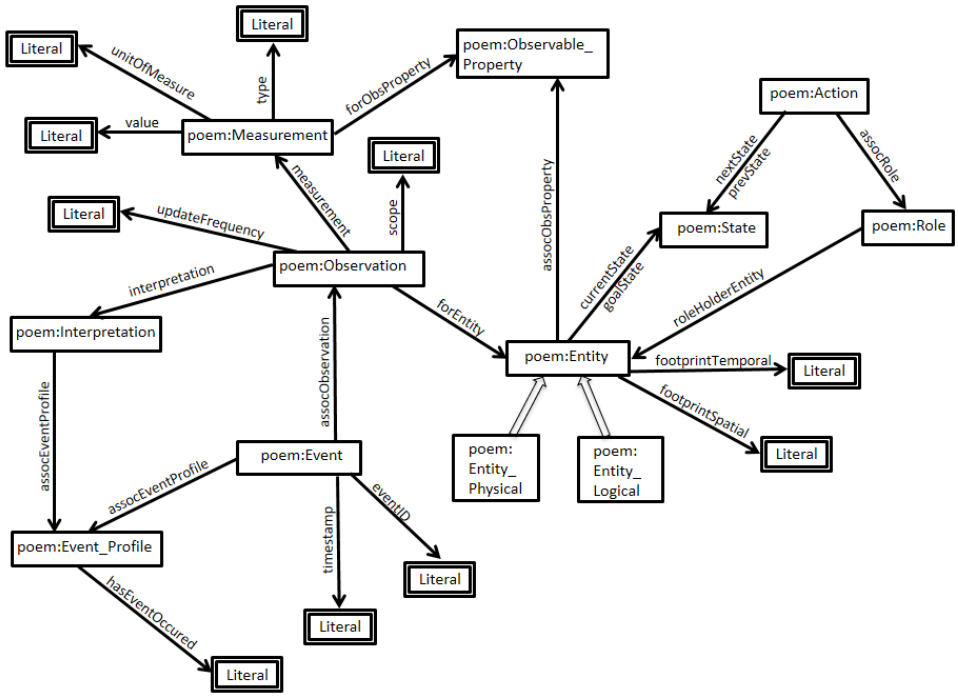

Fig. 5. PoEM Ontology Classes and Properties. Classes are shown as rectangles and literals (resulting from datatype properties) are shown in double rectangles.

6.1. *Case Study: Power Blackout Event in a House*

For our first case study, we focus on a simple power blackout event in a household. During a blackout event, all appliances in a house have lost power, and thus the sensor measuring voltage will either not show a reading or show a flat zero. In a simplistic case, we formulate a heuristic of repeatedly checking the voltage for a few appliances (heater, refrigerator and lightbulb) to see if their voltage is at a value of zero (or not).

It is simple to map this complex event to our PoEM model and ontology as shown in Figure 6. The appliances are entities in PoEM, voltage is an observable property with a measurement value, and the 'constant zero voltage' interpretation leads to a blackout event. If a blackout scenario is detected, the appliance entities will also be moved into an appropriate PoEM state (if not already there). Details of roles and actions are not shown in our illustration. Appropriate notifications to family members can be automatically sent in case a blackout event is detected. Also, if specified, this notification event can be escalated as required. For instance, if it is found that the blackout is not fixed over a period of time, an electrician

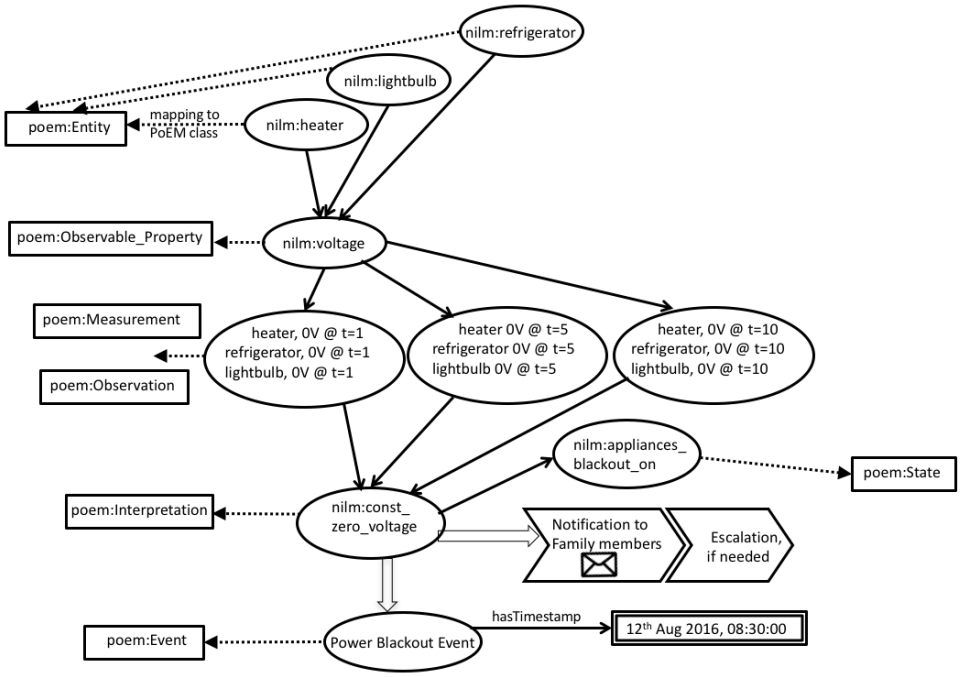

Fig. 6. A power blackout event in a household; `nilm:` denotes a NILM domain ontology.

may be automatically notified. Our illustration links to a domain ontology[i] (`nilm:`) which contains domain concepts such as the details of appliances and schematics for processes which might occur.

6.2. *Case Study: Maritime Piracy Event*

Next, we use the example of modeling maritime piracy events to show the extensible and adaptable capabilities of our model. Van Hage et al. [84] illustrated the Simple Event Model (SEM) with this example. We represent the same domain using the PoEM model by a simple identification and mapping of piracy event related concepts to PoEM ontology classes. This mapping is shown in Figure 7. Note that PoEM enables adding more context than the originally intended purpose of modeling just a piracy event detection scenario based on location of the yacht. Using PoEM, we can also model event notifications, actions, roles, and escalation to complete the event processing cycle. This model can be used to send notifications if a vessel at sea is under possible attack from pirates.

[i]Further discussion and distinction between event ontologies and domain ontologies is deferred to our previous work in [57]

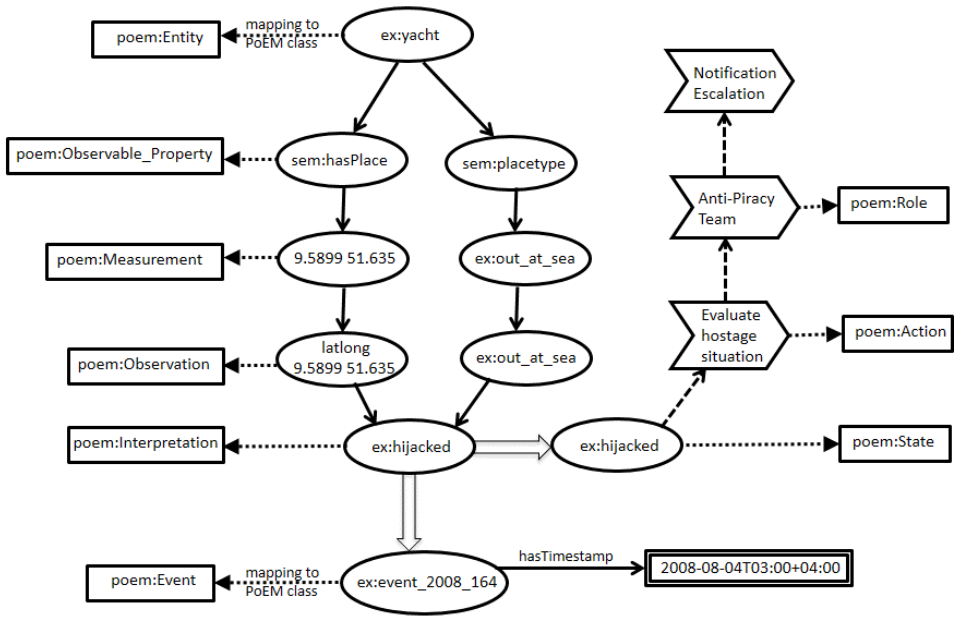

Fig. 7. Maritime piracy events mapped to PoEM; `sem:` denotes the simple event model ontology [84] and `ex:` is a maritime domain ontology.

6.3. *Case Study: Collision Avoidance in an Automobile*

We consider the process of automatically monitoring sensor measurements in an automobile for collision avoidance. We show how this process can be mapped on to the PoEM model through an example. We consider an automobile fitted with two types of sensors that are relevant for prediction of side collisions during merging. These are side-facing Light Detection and Ranging, or LIDAR, to measure distance to nearest object at the side and Inertial Measurement Unit, or IMU, to measure the vehicle's velocity and orientation. Measurements from these sensors are continuously matched against a complex event profile to detect side collision warning events. Note that this is a complex event with multiplicity in observable properties since the collision warning event depends on both the distance to the side obstacle and the turn angle of the vehicle. Detection of the warning event can change the state of the side collision avoidance state model which in turn initiates a corrective action based on returning to the non-warning goal state. Figure 8 illustrates the event detection concepts using the PoEM ontology (the states, actions, and roles of this process are not shown).

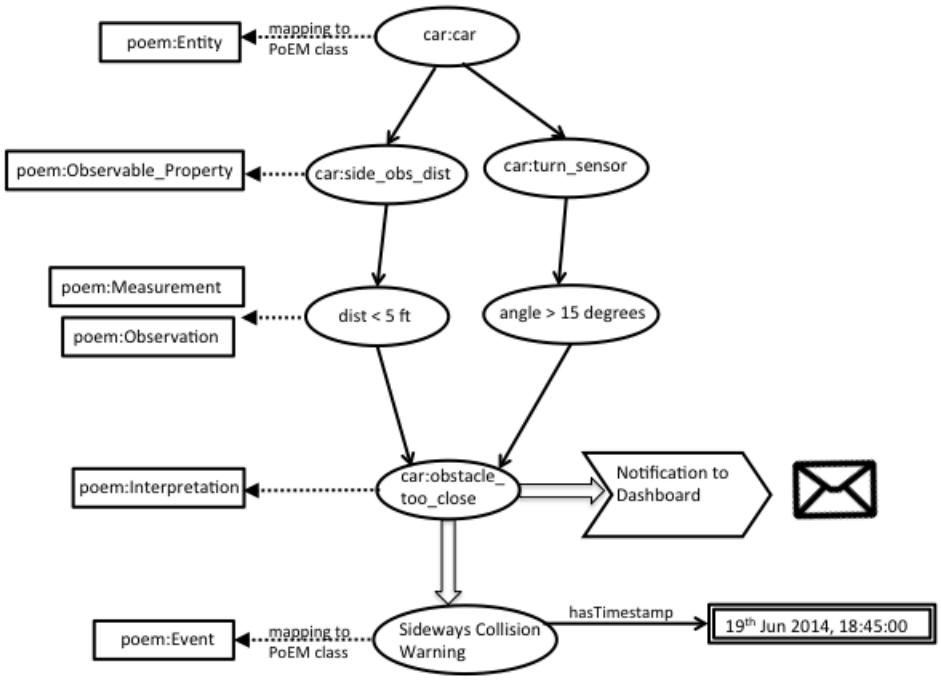

Fig. 8. Side collision warning event in a car mapped to PoEM

7. Incorporating Temporal Pattern Mining in the PoEM Model

In this section, we propose an enhancement to our event model to incorporate the mining of temporal pattern directly from raw sensor data. Integrating temporal pattern mining helps us automatically learn parameters for the PoEM model, and also facilitates the use of advanced machine learning approaches with predictive capabilities. We focus on a particular time series classification approach called time series shapelets [96], which are well-suited to our motivational use case in non-intrusive load monitoring. We also exhibit the utility of our shapelet-based approach on a real, large-scale NILM dataset.

7.1. *Time Series Shapelets for Classification*

Shapelets have been applied for a variety of time series data mining applications in diverse fields [49, 50, 53, 48, 52, 58, 47, 59, 24, 40, 96, 94]. A shapelet is a subsequence in a time series that is discriminative and has predictive power based on the distance of a new time series to the shapelet. Mathematically, they are identified using an information gain criteria (trying to find the subsequence in the training time series data which can discriminate best between the classes in the data).

In this work, we consider shapelets for time series classification though they can also be used for clustering [97]. Useful extensions of shapelets are Logical Shapelets [40], which consider logical combinations of multiple shapelets, and Local Shapelets [94], which consider approximate shapelet mining for early classification of time series and can be used in a realtime streaming scenario. In our evaluation, we use a faster, enhanced, state-of-the-art version of the original supervised shapelet-based classification algorithm, called Fast Shapelets [65].

Though several other approaches exist for time series classification, the popularity of shapelets is due to these factors (1) shapelet methods impose no assumptions or restrictions on nature of data unlike autoregressive or moving average time series models, crucial for processing events from real sensor data, (2) they are easy to interpret for domain experts and shapelets can be visualized easily, (3) they provide a method for localized pattern mining from time series as opposed to global methods which may not capture the local characteristics, and (4) once shapelets have been extracted from training data, we can classify a new time series very quickly since we discard the training data after shapelet extraction.

The basic shapelet extraction algorithm, first proposed by Ye and Keogh [96], is detailed in Algorithm 2. The input to the algorithm are time series instances along with their class labels, and the output is a shapelet subsequence. This algorithm performs a brute force search for the best shapelet candidate over the range of all possible generated candidate subsequences between the minimum (minL) and maximum (maxL) parameters. The information gain of each candidate subsequence is evaluated by creating a 'split' of the dataset into predicted classes as per the candidate and a distance threshold, and then computing the total information gain across the dataset (by comparing the entropy of the data before and after the split). The subsequence with the highest information gain is designated as a shapelet, and an appropriate distance threshold is also learned. The distance of a new time series from a shapelet is the minimum Euclidean distance obtained by sliding the (smaller length) shapelet subsequence across the (longer) new time series. The information gain criteria is a direct measure of the discriminative power of a subsequence to separate between classes in the input data. Note that this example shows a simple use case with just one shapelet, but in a real-world example, we typically observe multiple shapelets from the data arranged in the form of a decision tree classifier.

A decision tree classifier is built based on shapelets learned from the data. This tree has shapelets as internal nodes (with corresponding distance thresholds) and predicted class labels as leaf nodes. Classification can then be performed (for a test time series) by traversing the decision tree starting from the root till a leaf node is reached, taking the left branch at each node if the distance of the test time series from the shapelet is less than the distance threshold of corresponding shapelet in that node, and vice-versa.

An illustration of a shapelet is shown in Figure 9. This shapelet was automatically extracted from our evaluation dataset for the task of event detection, or in this evaluation context, the task of determining whether or not an appliance was

Algorithm 2 Shapelet Discovery Algorithm

```
Given 𝓘 = [TS, Label]
max_gain ← 0
for len = minL to maxL do
    Candidates ← GenAllCandidates(TS, len)
    for each cand in Candidates do
        create a split sp from cand
        gain ← ComputeInfoGain(𝓘, sp)
        if gain > max_gain then
            max_gain ← gain
            shapelet ← s
        end if
    end for
end for
```

switched on within this evaluation time window given the aggregate power use data (details of our evaluation are provided in Section 7.3). This shapelet has a length of 156 and a distance threshold of 11.0253, and was extracted from a time series in our training set which belonged to the 'event' class (appliance was switched on).

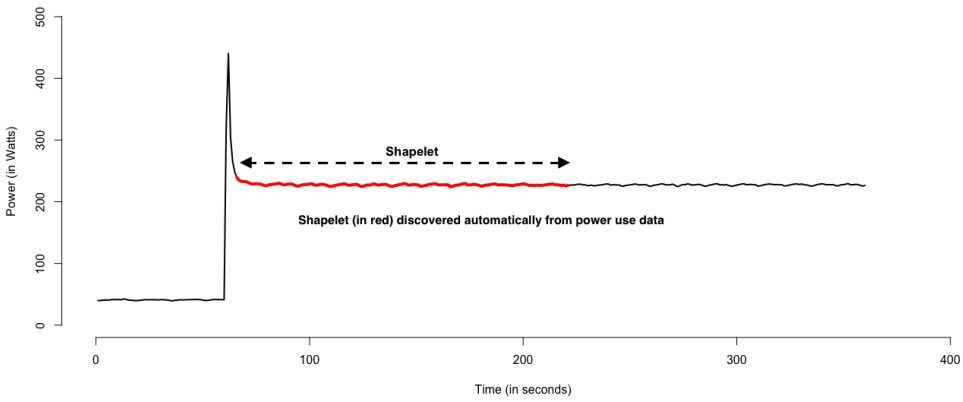

Fig. 9. An illustration of a shapelet extracted from our evaluation dataset. This shapelet (denoted by S1), extracted automatically from power use data in a household recorded by sensors, is able to semantically capture the event of an appliance switching on without any explicit background knowledge about the appliance or the non-intrusive load monitoring domain.

Visually, we can see that the shapelet can be interpreted as referring to the shape of change in aggregate power use following what looks like an event of an appliance being switched on. The power consumption suddenly increases and then decreases

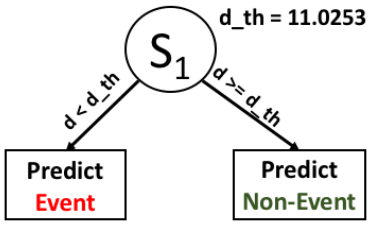

Fig. 10. Shapelet-based decision tree for classification of a new test instance, the shapelet S_1 is the one extracted in the previous Figure.

and remains stable, which is typical following the switching on of an appliance. In this experiment, this was the only shapelet extracted from our training dataset (of 922 times series, each being 360 data points long), and thus, after shapelet extraction we discard the entire training dataset of 922*360 = 331,920 data points and store just the 156 data points in the shapelet subsequence and the distance threshold value. Figure 10 shows how classification of a new instance happens with the shapelet-based decision tree formed from this shapelet. If the distance of the new test time series is less than the threshold, the test instance is predicted to be an event (appliance switched on within the time window of the test instance), otherwise, it is predicted to be a non-event. This shapelet, by itself, is able to achieve a 98.6% classification accuracy on the test dataset as shown in Table 5. Semantically, this shapelet captures the event of an appliance switching on automatically without any background knowledge. Shapelets are a simple but powerful tool to capture the local discriminative characteristics of events within sensor data.

7.2. *Shapelet-based Event Detection*

Time series shapelets provide an intuitive way to realize the design of our proposed event model in practice. They enable the model to operate directly on raw sensor data (typically time series), and automatically learn the required domain-specific parameters by the use of machine learning. As shapelets relate to the detection of critical happenings in time series, they are closely related to the idea of detecting 'events' from time series. Initial efforts towards finding a similar automatic link between time series shapelet mining and event processing have recently been proposed in autoCEP [39], but no event processing rules or interpretations, or event model is explicitly provided and their approach is not driven by the rich expressive and reasoning power of Semantic Computing.

To explore the link between these two areas further, we start by revisiting how a shapelet-based classifier classifies a new test time series instance. For simplicity, consider the case of a dataset with two classes, and just one shapelet, sh of length m, with a distance threshold d_{th}, in the decision tree. The predicted class of a test time series t_{test}, which has a distance d_{test} from sh (found by sliding the smaller

length shapelet across the longer test time series, computing the Euclidean distance at each point, and reporting the minimum), can be represented as:

$$prediction(t_{test}) = \begin{cases} \text{Same class as shapelet} & : d_{test} < d_{th} \\ \text{Opposite class from shapelet} & : d_{test} \geq d_{th} \end{cases} \tag{13}$$

Notice the similarity of the structure of this decision rule to the structure of our concept of interpretations and interpretation sets, proposed in Section 5.1.6. The example shown here is a binary classification instance with just one shapelet, but it is straightforward to extend this to a multi-class scenario with multiple shapelets, by just adding branches to this decision rule. This decision rule is equivalent in structure to our interpretation set, and the decision rules used to make predictions for a shapelet-based classifier essentially become the new event processing and event detection rules. All parameters in this rule, including the distance threshold and class label identification, are found automatically from the data by the powerful shapelet extraction method.

An automatic representation for an event learned from this interpretation can be the interpretation itself. Based on the output of a shapelet-based classifier (denoted by SBC), we can build an interpretation set for events automatically. Here is an example with multiple classes. The task here is to identify which appliance caused a detected event (which appliance was switched on).

$$X_{\text{switch-on-event}} = \begin{cases} \text{Refrigerator} & : \text{SBC predicts Refrigerator class} \\ \text{Lights} & : \text{SBC predicts Lights class} \\ \text{Fan} & : \text{SBC predicts Fan class} \\ \text{Other} & : \text{SBC predicts other appliance class} \end{cases} \tag{14}$$

Timestamped data (time series) can be easily collected from several different applications, possibly at very high frequencies making this connection between shapelets and our proposed event model applicable to a large number of domains. As the shapelet mining component fits in within the PoEM model by providing a new way to approach interpretations, the rest of the model does not need to be modified to accommodate this component, and the event parameters can now be learned automatically. Background knowledge from the domain can also be added on top of the shapelet mining framework to further enrich the interpretations of extracted events. An interesting future direction of research is to also incorporate semantics into the event processing middleware, for instance, to automatically deploy CEP as well as time series mining components based on the situation and maintain their lifecycle. An effective approach for managing these middleware components are enterprise integration patterns [28]. These enterprise integration patterns propose middleware components including message construction patterns, message routing patterns and message transformation patterns. An initial approach towards semantically managing these patterns in a smart grids application is proposed in our previous work in [51].

7.3. *Evaluating Shapelets for NILM*

We demonstrate the utility of shapelets for energy disaggregation on the publicly available Building-Level fUlly-labeled dataset for Electricity Disaggregation (BLUED) dataset [2], which contains voltage, current and power measurements for a single family in the United States for one week. Every appliance transition (switching on/off) for each appliance in the house is labeled and time-stamped, thus providing the ground truth for the energy disaggregation task. The data available is downsampled to 60 Hz (from the collection sampling rate of 12 kHz). Real and reactive power, for both phase A and phase B are available in the dataset. We use the real power values from both phase A and B appliances in our experiments. The complete BLUED dataset (one week long) contains nearly 37 million data points. We used the first 50% of the data for training, and the next 50% of the data for testing. A typical advantage of shapelet-based algorithms is that they find localized shapelet patterns, and once they find these patterns the rest of the data is discarded, so they often perform well with a smaller fraction of the data used for training than other methods.

To evaluate the utility of shapelets on the BLUED dataset, we first need to preprocess the data into the appropriate (labeled time series) format for training and testing. Ground truth events, in the form of timestamps when a certain appliance was switched on/off, are available to us. However the data about power is just one long time series, and ideally, we want multiple training and testing time series instances. To resolve this, we consider a window around each event. As BLUED suggests that each event lasts for at least 5 seconds, we define this window to contain all data points within 1 second before the event and up to 5 seconds after the event. Since each second of data contains 60 data points, the length of our training and testing time series is set at 360 (but can be changed by varying the parameters above).

We evaluate the performance of shapelets on detecting events and classifying appliance activity from aggregate electrical power data. The first task is to *detect* events, i.e. differentiate event data segments from the non-event segments. To achieve this, we extract all event instances, according to the pre-processing steps described earlier. Then, we extract several non-event instances of the same length from the power consumption data (ensuring that no events overlap at all with any of the non-event segments). We perform this sampling of non-event instances in a balanced manner (1:1 ratio of events:non-events) as well as an imbalanced manner (1:4 ratio of events:non-events). This experiment is performed for the power data in both phase A and phase B. The second task is to perform actual disaggregation, i.e. to *classify* which appliance is actually responsible for which event. This is achieved by a multi-class shapelet-based classifier. We divide the appliances into groups, based on the nature of the appliances and their average power consumption. This gives us multiple classes of appliances for training and classification. The classes we use, along with instances of appliances of those classes are mentioned in Table 4.

Table 4. Putting BLUED dataset appliances into classes

Phase A appliance classes	
Class	Examples
1. Refrigerator	refrigerator
2. Lights	backyard lights, washroom light, bedroom light
3. High-Power ($>$ 150W)	hair dryer, air compressor, kitchen chopper
Phase B appliance classes	
Class	Examples
1. Lights	desktop lamp, basement light, closet lights
2. High-Power ($>$ 150W)	printer, iron, garage door
3. Low-Power ($<$ 150W)	computer, LCD monitor, DVR/blu-ray player

In our previous work in [50], we showed preliminary analysis of the utility of time series shapelets for these energy data mining tasks on BLUED. However, the events there were just the single switching on/off of an appliance. The approach proposed here can now be used to model many more complex events and represent them in a richer, expressive manner (such as the blackout event case study in Section 6.1). This can be done without much additional effort since the shapelet mining method is the same, but can now be connected to a comprehensive event model in PoEM.

To evaluate the efficacy of our shapelet-based approach, we compare the classification accuracy (on test data) to ten other popular classifiers. We formed a baseline classifier which always assigns the class label of the majority class in the training data to any test data instance. The other nine classifiers used are implemented using scikit-learn [61] in Python with their default parameters. These classifiers are Naive Bayes, Linear Discriminant Analysis (LDA) based classifier, 1-Nearest Neighbor (1-NN) classifier, Logistic Regression based classifier (Logistic), Support Vector Machines (SVM), Multilayer Perceptrons (MLP), Decision Tree classifier, Random Forests classifier and AdaBoost. For the event detection experiment, we evaluate our methods on both the balanced (1:1 ratio of events:non-events) as well as imbalanced (1:4 ratio of events:non-events) datasets. For every experiment, evaluation is done on both Phase A and Phase B. The event detection experiment is a binary classification task while the event classification experiment is a 3-class classification task.

We present our evaluation results (classification accuracy on test data) for event detection in Table 5 and event classification in Table 6. We note that our shapelets approach performs at par with other state-of-the-art classifiers. It has been previously reported [93] that the 1-nearest neighbor classifier is hard to beat for time series classification tasks on many datasets, and we agree. In all the experimental cases considered, our shapelet-based classifier is either close in performance to the best performing classifier, or performs the best out of all classifiers. Particularly, we observe that shapelets perform better when the data is more complex or challeng-

Table 5. Classification accuracy of different classifiers for event detection (detecting whether an appliance was switched on/off)

Phase	Phase A	Phase A	Phase B	Phase B
Dataset Balance	1:1	1:4	1:1	1:4
Shapelets	98.6	99.0	98.3	97.9
Baseline	50.0	75.0	50.0	75.0
Naive Bayes	68.1	84.3	65.8	82.4
LDA	68.4	87.4	66.9	83.2
1-NN	99.4	99.5	95.6	97.8
Logistic	94.9	96.8	63.0	83.2
SVM	95.4	95.4	50.0	80.0
MLP	99.1	95.3	89.5	83.0
Decision Tree	94.3	98.1	85.6	93.6
Random Forests	96.6	99.0	91.9	96.0
AdaBoost	91.2	92.1	66.6	83.4

Table 6. Classification accuracy of different classifiers for event classification (identifying which appliance(s) was switched on/off), which is the energy disaggregation step

Phase	Phase A	Phase B
Shapelets	83.8	77.9
Baseline	77.3	41.3
Naive Bayes	70.5	52.4
LDA	56.0	48.6
1-NN	90.2	63.7
Logistic	78.8	53.0
SVM	78.0	34.0
MLP	77.3	42.3
Decision Tree	84.5	58.0
Random Forests	85.8	63.4
AdaBoost	80.0	50.8

ing - they perform the best for the imbalanced data (Phase B) of all classifiers, and again, they perform significantly better than any other classifier for the multi-class event classification task (Phase B). Additionally, our shapelet-based approach has several other advantages - (i) the classification step is extremely fast as the training data is discarded once shapelets are found, (ii) it does not make any assumptions about the structure of the data, which does not need to be smooth or differentiable or come from any specific distribution, and (iii) our approach provides visual intuition and interpretability in the form of shapelets - these can be used for feedback

from users and domain experts, unlike several machine learning classifiers which behave like a blackbox.

8. Conclusion

In this work, we addressed the critical problem of resolving the difference in representation between high-level semantic event models and low-level sensor data streams, and bringing together the communities of Semantic Computing based event processing and time series data mining. We explored the need for Semantic Computing in several aspects and functioning components of complex event processing systems. For efficiently detecting and processing events from Big Data streams, we proposed an approach to incorporate an advanced machine learning approach based on time series shapelets into the Process-oriented Event model. Our combined event representation and detection framework is able to rapidly represent events in new application domains, and we also demonstrated its high accuracy in classification and prediction of new events from data streams.

References

[1] R. Adaikkalavan and S. Chakravarthy. Snoopib: Interval-based event specification and detection for active databases. *Data & Knowledge Engineering*, 59(1):139–165, 2006.

[2] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges. Blued: A fully labeled public dataset for event-based non-intrusive load monitoring research. In *Proceedings of the 2nd KDD workshop on data mining applications in sustainability (SustKDD)*, pages 1–5, 2012.

[3] K. D. Anderson, M. E. Bergés, A. Ocneanu, D. Benitez, and J. M. Moura. Event detection for non intrusive load monitoring. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 3312–3317. IEEE, 2012.

[4] D. Anicic, P. Fodor, S. Rudolph, R. Stühmer, N. Stojanovic, and R. Studer. Etalis: Rule-based reasoning in event processing. *Reasoning in EventBased Distributed Systems*, 347:99–124, 2011.

[5] N. Batra, A. Singh, and K. Whitehouse. Gemello: Creating a detailed energy breakdown from just the monthly electricity bill. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016.

[6] M. Berges, E. Goldman, H. S. Matthews, L. Soibelman, and K. Anderson. User-centered nonintrusive electricity load monitoring for residential buildings. *Journal of Computing in Civil Engineering*, 25(6):471–480, 2011.

[7] B. Berstel. Extending the rete algorithm for event management. In *Temporal Representation and Reasoning, 2002. TIME 2002. Proceedings. Ninth International Symposium on*, pages 49–51. IEEE, 2002.

[8] R. Blanco, J. Wang, and P. Alencar. A metamodel for distributed event based systems. In *Proceedings of the second international conference on Distributed event-based systems*, pages 221–232. ACM, 2008.

[9] H. Boley, A. Paschke, and O. Shafiq. Ruleml 1.0: the overarching specification of web rules. *Lecture Notes in Computer Science*, 6403(4):162–178, 2010.

[10] S. Chakravarthy and D. Mishra. Snoop: An expressive event specification language for active databases. *Data & Knowledge Engineering*, 14(1):1–26, 1994.

[11] K. M. Chandy. Theory and implementation of a distributed event based platform. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pages 205–213. ACM, 2016.

[12] K. M. Chandy, O. Etzion, and R. von Ammon. The event processing manifesto. In *2010 Dagstuhl Seminar on Event Processing*, 2010.

[13] S. Chaudhuri. What next?: a half-dozen data management research goals for big data and the cloud. In *Principles of Database Systems*, pages 1–4, 2012.

[14] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, June 2012.

[15] M. Doerr. The cidoc conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI magazine*, 24(3):75, 2003.

[16] M. Eckert, F. Bry, S. Brodt, O. Poppe, and S. Hausmann. Two semantics for CEP, no double talk: Complex event relational algebra (CERA) and its application to XChangeEQ. *Reasoning in Event-Based Distributed Systems*, pages 71–97, 2011.

[17] O. Etzion, Y. Magid, E. Rabinovich, I. Skarbovsky, and N. Zolotorevsky. Context-based event processing systems. In S. Helmer, A. Poulovassilis, and F. Xhafa, editors, *Reasoning in Event-Based Distributed Systems*, volume 347 of *Studies in Computational Intelligence*, pages 257–278. Springer Berlin / Heidelberg, 2011.

[18] L. Farinaccio and R. Zmeureanu. Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses. *Energy and Buildings*, 30(3):245–259, 1999.

[19] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1):17–37, 1982.

[20] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel. Disaggregated end-use energy sensing for the smart grid. *IEEE Pervasive Computing*, 10(1):28–39, Jan 2011.

[21] A. Gupta and R. Jain. Managing event information: Modeling, retrieval, and applications. *Synthesis Lectures on Data Management*, 3(4):1–141, 2011.

[22] S. Gupta, M. S. Reynolds, and S. N. Patel. Electrisense: single-point sensing using emi for electrical event detection and classification in the home. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 139–148. ACM, 2010.

[23] K. Hammar. Modular semantic cep for threat detection. In *Operations Research and Data Mining ORADM 2012 workshop proceedings*. National Polytechnic Institute, 2012.

[24] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[25] A. Hinze. *A-MEDIAS: Concept and Design of an Adaptive Integrating Event Notification Service*. PhD thesis, Freie Universitaet Berlin, 2003.

[26] A. Hinze, K. Sachs, and A. Buchmann. Event-based applications and enabling technologies. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, pages 1–15, New York, NY, USA, 2009. ACM.

[27] A. Hinze and A. Voisard. A parameterized algebra for event notification services. In *Temporal Representation and Reasoning, 2002. TIME 2002. Proceedings. Ninth International Symposium on*, pages 61–63. IEEE, 2002.

[28] G. Hohpe and B. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[29] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member sub-*

mission, 21:79, 2004.

[30] K. Kaneiwa, M. Iwazume, and K. Fukuda. An upper ontology for event classifications and relations. *AI 2007: Advances in Artificial Intelligence*, pages 394–403, 2007.

[31] A. Kumar, W. Yao, C.-H. Chu, and Z. Li. Ensuring compliance with semantic constraints in process adaptation with rule-based event processing. In *Proceedings of the 2010 international conference on Semantic web rules*, RuleML'10, pages 50–65, Berlin, Heidelberg, 2010. Springer-Verlag.

[32] C. Lagoze and J. Hunter. The abc ontology and model. In *DC-2001: International Conference on Dublin Core and Metadata Applications 2001*, volume 2, pages 1–18. British Computer Society and Oxford University Press, 2001.

[33] D. Liu, C. Pedrinaci, and J. Domingue. A framework for feeding linked data to complex event processing engines. In *1st International Workshop on Consuming Linked Data (COLD 2010) at The 9th International Semantic Web Conference (ISWC 2010)*, 2010.

[34] J. Liu and X. Guan. Complex event processing for sequence data and domain knowledge. In *Mechanic Automation and Control Engineering (MACE), 2010 International Conference on*, pages 2899–2902. IEEE, 2010.

[35] D. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, 2002.

[36] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, volume 3864, pages 44–49. AAAI Press, 2006.

[37] D. Miranker. *TREAT: A new and efficient match algorithm*. PhD thesis, Ph. D. dissertation, Columbia University, 1987.

[38] T. Moser, H. Roth, S. Rozsnyai, R. Mordinyi, and S. Biffl. Semantic event correlation using ontologies. In *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part II*, OTM '09, pages 1087–1094, Berlin, Heidelberg, 2009. Springer-Verlag.

[39] R. Mousheimish, Y. Taher, and K. Zeitouni. Automatic learning of predictive rules for complex event processing: doctoral symposium. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pages 414–417. ACM, 2016.

[40] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162. ACM, 2011.

[41] G. Mühl, L. Fiege, and P. R. Pietzuch. *Distributed event-based systems*. Springer, 2006.

[42] D. Mukherjee, S. Banerjee, and P. Misra. Ad-hoc ride sharing application using continuous sparql queries. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 579–580. ACM, 2012.

[43] G. Papamarkos, A. Poulovassilis, and P. Wood. Event-condition-action rule languages for the semantic web. In *Workshop on Semantic Web and Databases*, 2003.

[44] A. Paschke, H. Boley, Z. Zhao, K. Teymourian, and T. Athan. Reaction ruleml 1.0: standardized semantic reaction rules. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 100–119. Springer, 2012.

[45] A. Paschke, P. Vincent, A. Alves, and C. Moxey. Tutorial on advanced design patterns in event processing. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, pages 324–334, New York, NY,

USA, 2012. ACM.

[46] N. W. Paton and O. Díaz. Active database systems. *ACM Computing Surveys (CSUR)*, 31(1):63–103, 1999.

[47] O. P. Patri. Modeling and recognition of events from multidimensional data: doctoral symposium. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pages 430–433. ACM, 2016.

[48] O. P. Patri, R. Kannan, A. Panangadan, and V. K. Prasanna. Multivariate time series classification using inter-leaved shapelets. In *Time Series Workshop in Neural Information Processing Systems (NIPS)*, 2015.

[49] O. P. Patri, A. V. Panangadan, C. Chelmis, R. G. McKee, and V. Prasanna. Predicting failures from oilfield sensor data using time series shapelets. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2014.

[50] O. P. Patri, A. V. Panangadan, C. Chelmis, and V. K. Prasanna. Extracting discriminative features for event-based electricity disaggregation. In *Technologies for Sustainability (SusTech), 2014 IEEE Conference on*, pages 232–238. IEEE, 2014.

[51] O. P. Patri, A. V. Panangadan, V. S. Sorathia, and V. K. Prasanna. Semantic management of enterprise integration patterns: A use case in smart grids. In *10th International Workshop on Information Integration on the Web (IIWeb), IEEE 30th International Conference on Data Engineering (ICDE)*, 2014.

[52] O. P. Patri, N. Reyna, A. Panangadan, V. Prasanna, et al. Predicting compressor valve failures from multi-sensor data. In *SPE Western Regional Meeting*. Society of Petroleum Engineers, 2015.

[53] O. P. Patri, A. B. Sharma, H. Chen, G. Jiang, A. V. Panangadan, and V. K. Prasanna. Extracting discriminative shapelets from heterogeneous sensor data. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 1095–1104. IEEE, 2014.

[54] O. P. Patri, K. Singh, and P. Szekely. Photo odyssey: Create personalized photography itineraries in realtime. In *AI Mashup Challenge Demo at the 10th Extended Semantic Web Conference (ESWC)*, 2013.

[55] O. P. Patri, K. Singh, P. Szekely, A. V. Panangadan, and V. K. Prasanna. Personalized trip planning by integrating multimodal user-generated content. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 381–388. IEEE, 2015.

[56] O. P. Patri, V. S. Sorathia, A. V. Panangadan, and V. K. Prasanna. The process-oriented event model (PoEM): a conceptual model for industrial events. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pages 154–165. ACM, 2014.

[57] O. P. Patri, V. S. Sorathia, and V. Prasanna. Event-driven information integration for the digital oilfield. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2012.

[58] O. P. Patri, A. S. Tehrani, V. K. Prasanna, R. Kannan, A. Panangadan, N. Reyna, et al. Data mining with shapelets for predicting valve failures in gas compressors. In *SPE Western Regional Meeting*. Society of Petroleum Engineers, 2016.

[59] O. P. Patri, M. Wojnowicz, and M. Wolff. Discovering malware with time series shapelets. In *2017 Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS-50)*, 2017.

[60] H. Paulheim. Efficient semantic event processing: Lessons learned in user interface integration. *The Semantic Web: Research and Applications*, pages 60–74, 2010.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Courna-

peau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] C. Pedrinaci, J. Domingue, and A. K. A. de Medeiros. A core ontology for business process analysis. In *European Semantic Web Conference*, pages 49–64. Springer, 2008.

[63] M. Popescu, P. Lorenz, and J. Nicod. A semantic-oriented framework for system diagnosis. *International Journal On Advances in Telecommunications*, 3(3 and 4):173–193, 2011.

[64] A. Poulovassilis, G. Papamarkos, and P. T. Wood. Event-condition-action rule languages for the semantic web. In *International Conference on Extending Database Technology*, pages 855–864. Springer, 2006.

[65] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 13th SIAM international conference on data mining*, pages 668–676. SIAM, 2013.

[66] P. Rosales and J. Jung. Semantic annotation on event pattern languages for complex event processing in ubiquitous logistics. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 160–163. IEEE, 2011.

[67] A. Scherp, T. Franz, C. Saathoff, and S. Staab. F–a model of events based on the foundational ontology dolce + dns ultralight. In *Proceedings of the fifth international conference on Knowledge capture*, pages 137–144. ACM, 2009.

[68] K.-U. Schmidt, R. Stühmer, and L. Stojanovic. Blending complex event processing with the rete algorithm. In *Proceedings of iCEP2008: 1st International Workshop on Complex Event Processing for the Future Internet*, volume 412. Citeseer, 2008.

[69] H. Shao, M. Marwah, and N. Ramakrishnan. A temporal motif mining approach to unsupervised energy disaggregation. In *Proceedings of the 1st International Workshop on Non-Intrusive Load Monitoring, Pittsburgh, PA, USA*, volume 7, 2012.

[70] H. Shao, M. Marwah, and N. Ramakrishnan. A temporal motif mining approach to unsupervised energy disaggregation: Applications to residential and commercial buildings. In *AAAI*, 2013.

[71] R. Shaw, R. Troncy, and L. Hardman. Lode: Linking open descriptions of events. *The Semantic Web*, pages 153–167, 2009.

[72] S. R. Shaw, S. B. Leeb, L. K. Norford, and R. W. Cox. Nonintrusive load monitoring and diagnostics in power systems. *Instrumentation and Measurement, IEEE Transactions on*, 57(7):1445–1454, 2008.

[73] P. Siano. Demand response and smart gridsa survey. *Renewable and Sustainable Energy Reviews*, 30:461–478, 2014.

[74] Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, Q. Zhou, and V. Prasanna. Cloud-based software platform for big data analytics in smart grids. *Computing in Science & Engineering*, 15(4):38–47, 2013.

[75] Y. Simmhan, Q. Zhou, and V. Prasanna. Semantic information integration for smart grid applications. *Green IT: Technologies and Applications*, page 361, 2011.

[76] V. Sorathia. *Dynamic Information Management Methodology with Situation Awareness Capability*. PhD thesis, Dhirubhai Ambani Institute of Information and Communication Technology, 2008.

[77] N. Stojanovic, L. Stojanovic, D. Anicic, J. Ma, S. Sen, and R. Sthmer. Semantic complex event reasoning - beyond complex event processing. In *Foundations for the Web of Information and Services'11*, pages 253–279, 2011.

[78] S. Suhothayan, K. Gajasinghe, I. Loku Narangoda, S. Chaturanga, S. Perera, and V. Nanayakkara. Siddhi: a second look at complex event processing architectures. In *Proceedings of the 2011 ACM workshop on Gateway computing environments*, pages

43–50. ACM, 2011.

[79] K. Taylor and L. Leidinger. Ontology-driven complex event processing in heterogeneous sensor networks. *The Semanic Web: Research and Applications*, pages 285–299, 2011.

[80] K. Teymourian and A. Paschke. Semantic rule-based complex event processing. *Rule Interchange and Applications*, pages 82–92, 2009.

[81] K. Teymourian and A. Paschke. Enabling knowledge-based complex event processing. In *Proceedings of the 2010 EDBT/ICDT Workshops*, page 37. ACM, 2010.

[82] K. Teymourian and A. Paschke. Semantic enrichment of event stream for semantic situation awareness. In *Semantic Web*, pages 185–212. Springer, 2016.

[83] K. Teymourian, M. Rohde, and A. Paschke. Knowledge-based processing of complex stock market events. In *Proceedings of EDBT 2012*, 2012.

[84] W. Van Hage, V. Malaisé, R. Segers, L. Hollink, and G. Schreiber. Design and use of the simple event model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, 2011.

[85] A. Voisard and H. Ziekow. Architect: A layered framework for classifying technologies of event-based systems. *Information Systems*, 36(6):937–957, 2011.

[86] K. Walzer, T. Breddin, and M. Groch. Relative temporal constraints in the rete algorithm for complex event detection. In *Proceedings of the second international conference on Distributed event-based systems*, pages 147–155. ACM, 2008.

[87] F. Wang, S. Liu, P. Liu, and Y. Bai. Bridging physical and virtual worlds: complex event processing for rfid data streams. *Advances in Database Technology-EDBT 2006*, pages 588–607, 2006.

[88] Y.-W. Wang and E. N. Hanson. A performance comparison of the rete and treat algorithms for testing database rule conditions. In *Data Engineering, 1992. Proceedings. Eighth International Conference on*, pages 88–97. IEEE, 1992.

[89] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. Efficient processing of uncertain events in rule-based systems. *Knowledge and Data Engineering, IEEE Transactions on*, 24(1):45–58, 2012.

[90] U. Westermann and R. Jain. E - a generic event model for event-centric multimedia data management in echronicle applications. In *Proceedings of the 22nd International Conference on Data Engineering Workshops*, ICDEW '06, pages 106–, Washington, DC, USA, 2006. IEEE Computer Society.

[91] M. Worboys and K. Hornsby. From objects to events: Gem, the geospatial event model. *Geographic Information Science*, pages 327–343, 2004.

[92] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[93] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040. ACM, 2006.

[94] Z. Xing, J. Pei, S. Y. Philip, and K. Wang. Extracting interpretable features for early classification on time series. In *SDM*, volume 11, pages 247–258. SIAM, 2011.

[95] Y. Xu, P. Wolf, N. Stojanovic, and H.-J. Happel. Semantic-based complex event processing in the aal domain. In *9th International Semantic Web Conference (ISWC2010)*, November 2010.

[96] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956. ACM, 2009.

[97] J. Zakaria, A. Mueen, and E. Keogh. Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–

794. IEEE, 2012.

[98] M. Zeifman, C. Akers, and K. Roth. Nonintrusive appliance load monitoring (nialm) for energy control in residential buildings: Review and outlook. In *IEEE transactions on Consumer Electronics*. Citeseer, 2011.

[99] Q. Zhou, Y. Simmhan, and V. Prasanna. Incorporating semantic knowledge into dynamic data processing for smart power grids. *The Semantic Web–ISWC 2012*, pages 257–273, 2012.

[100] T. Zhu, A. Bakshi, V. Prasanna, and K. Gomadam. Applying semantic web techniques to reservoir engineering: Challenges and experiences from event modeling. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 586–591. IEEE, 2010.

[101] D. Zimmer and R. Unland. On the semantics of complex events in active database management systems. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 392–399. IEEE, 1999.

[102] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012.