Probabilistic and Fuzzy Logic based Event Processing for Effective Business Intelligence

Govindasamy Vaiyapuri[1] and Thambidurai Perumal[2]
[1]Department of Computer Science and Engineering, Pondicherry Engineering College, India
[2]Computer Science and Engineering, Perunthalaivar Kamarajar Institute of Engineering and Technology, India

Abstract: *This paper focuses on Probabilistic Complex Event Processing (PCEP) in the context of real world event sources of data streams. PCEP executes complex event pattern queries on the continuously streaming probabilistic data with uncertainty. The methodology consists of two phases: Efficient Generic Event Filtering (EGEF) and probabilistic event sequence prediction paradigm. In the first phase, a Non-deterministic Finite Automaton (NFA) based event matching allows to filter the relevant events by discovering the occurrences of the user defined event patterns in a large volume of continuously arriving data streams. In order to, express the complex event patterns in a more efficient form, a Complex Event Processing (CEP) language named as Complex Event Pattern Subscription Language (CEPSL) is developed by extending the existing high level event query languages. Furthermore, query plan-based approach is used to compile the specified event patterns into the NFA automaton and to distribute to a cluster of state machines to improve the scalability. In the second phase, an effective Dynamic Fuzzy Probabilistic Relational Model (DFPRM) is proposed to construct the probability space in the form of event hierarchy. The proposed system deploys a Probabilistic Fuzzy Logic (PFL) based inference engine to derive the composite of event sequence approximately with the reduced probability space. To determine the effectiveness of the proposed approach, a detailed performance analysis is performed using a prototype implementation.*

Keywords: *CEP, event filtering, NFA, uncertain events, DFPRM.*

Received June 12, 2013; accepted July 28, 2013; published online April 1, 2015

1. Introduction

In today's world, many distributed software applications generate high quantity of the data stream continuously from geographically distributed multiple sources with unpredictable rate. Business processing systems are interested in acquiring the high level of intelligence from the available data. The standard Data BaseManagement System (DBMS) does not support timeliness of flow processing [9, 19]. This nature of DBMS led to the development of Information Flow Processing (IFP). There are two types of IFP models-Data Stream Management System (DSMS) and Complex Event Processing (CEP). DSMS handles the transient data that flow continuously from the multiple sources [17]. CEP infers complex events. The core of the CEP is the CEP engine (processor) that detects event patterns from the large number of incoming events. An event pattern expresses the rules in a declarative language to describe the complex relationship between the incoming events to obtain the relevant information to trigger the output events [13]. A data source has inherently unreliable data collection method, or the data source generates erroneous data which leads to uncertainty [4]. In this work, a generic framework is presented for modelling and managing the events and rules with uncertainty with the successful implementation of CEP.

This paper, illustrates a probabilistic approach that attempts to provide the solution for the challenges of robust complex event detection. The probabilistic methodology offers declarative language to express the event pattern queries as probabilistic event hierarchies to support probabilistic inference in order to, trigger the relevant sequences under uncertainty.

1.1. Contributions of this Paper

- Complex Query Language: A flexible event language Complex Event Pattern Subscription Language (CEPSL) is developed to enrich the existing restricted query language to specify the complex event pattern queries under uncertainty.
- Event Pattern Matching: An efficient NFA based event matching algorithm filters the relevant events.
- Complex Event Detection: Event sequence prediction phase supports probabilistic inference on complex uncertain events. Probabilistic event hierarchies are constructed in the form of graphical model called as Dynamic Fuzzy Probabilistic Relational Models (DFPRM) that infers the correlations between the sequences of incoming events.
- Computation of Probability: DFPRM model computes the joint probability distribution based on the observation of the correlation between event sequences to enhance the robustness of the event detection process under uncertainty.

2. Related Works

2.1. Review on Event Filtering Schemes

The development of filtering algorithm based on Binary Decision Diagram (BDD) in a large scale publish/subscribe system is discussed in [6]. BDD filters the relevant events according to the matching level between large numbers of incoming publications (events) against the selection criteria of user subscriptions (queries). In order to, reduce the resource utilization for storage and operation overhead, an effective CEP system is proposed that identifies the sequence of relevant complex events with minimal resource consumption [18]. Effective pattern matching is performed in two phases such as threshold phase and detection phase. A scalable and high performance event filtering mechanism for enterprise-wide Distributed Dynamic Multi-Point (DDMP) applications is developed in [1]. DDMP is a data reduction mechanism that minimizes the unnecessary operation overhead and network traffic in order to monitor, detect and deliver the events to interested consumers. Events are modelled using an expressive event language and then, user subscriptions are internally represented in the form of Deterministic Finite state Automata (DFA). The main problem is that it is a domain dependent application and it is focused only on the event filtering in DDMP domain. The system cannot be applicable to all the other domains in a business enterprise system.

2.2. Complex Event Processing Systems

This section is to point out the existing CEP systems such as Telegraph Project (TelegraphCQ), SASE, Cayuga and Next CEP. The TelegraphCQ is an adaptive data flow processing scheme that processes the event streams in an adaptive environment [7]. The SASE [16] is another CEP system that deploys a complex event processor for continuously arriving event streams in order to derive the events in a timely manner. SASE is a comprehensive system where filtering, pattern matching, and aggregation mechanism are performed using Non-deterministic Finite Automata (NFA) based query plan approach [28]. It fails to consider the uncertain data which possibly occurs in practical real-time streams. Later, efficient and commonly used online CEP system called as Cayuga that can process the large number of event streams based on complex patterns in user subscriptions is developed [12]. The complex event pattern queries are converted into event algebra using unary operators and binary operators. A formal query evaluation model called as NFA model is derived from the converted event algebra. In order to, achieve distributed event detection, a query plan based approach called as the next CEP system [22] is proposed with query rewriting and distribution. The next CEP system deploys a new expressive automaton based approach to perform distributed event detection. In order to, define the efficient pattern matching approach, a new query evaluation model is proposed to evaluate the pattern queries over a large number of incoming event streams [2]. In addition, a query plan is devised for the NFA model based on the compilation technique. This approach achieves high scalability and throughput. The disadvantage is the high computation required for unnecessary operations for complex event detection.

2.3. Query Processing Schemes in Probabilistic Databases

The aforementioned CEP systems perform the event processing only for a deterministic database. These systems do not address uncertain data. The methodology to manage uncertain data is addressed in [25]. In addition, a working system that incorporates data accuracy and lineage for uncertain database is presented. To handle uncertain data in CEP, an efficient uncertain CEP system called as Lahar is proposed to process the event streams from the uncertain or probabilistic database. Here, uncertain data are modelled as a Hidden Markov Model (HMM) [23] that infers the hidden readings from the RFID data streams. In addition, a query model is designed for complex queries using Cayuga event language with detection operators. An efficient query evaluation framework is proposed to process the queries effectively in a probabilistic or uncertain database [10]. It generates a large number of query results without quality. Therefore, ranking process is needed. The ranking process returns the query results with a high confidence score as the most probable Top-k answers based on the corresponding query, but the approach does not compute the exact probability score to determine the most suitable query results of high quality. In order to, address the drawbacks in the previous approach, an efficient Top-k query processing scheme is presented. Algorithms such as Uncertain Top-k query (U-Top-k) and Uncertain k-Rank query (U-kRank query) that extend the semantics of Top-k queries are developed. Furthermore, determining the number of tuples occurred in a state, a graph is constructed with the set of states according to space and time. It leads to a large search scope with exponentially increasing storage space [26]. In order to, overcome the problems of the existing approaches, novel polynomial algorithm is proposed based on the adopted x-relation model. But the existing approaches do not provide solution for the uncertain data streams continuously arriving from unreliable event sources. Therefore, the main focus of solution the proposed approach is to process the uncertain event streams based on the complex event pattern queries in a probabilistic framework [29].

3. A Probabilistic Model for Reasoning over Uncertainty in Rule based CEP

To provide the event notification in a timely manner under uncertainty, a Probabilistic Complex Event Processing (PCEP) system is implemented in publish/subscribe middleware [8]. In the Figure 1, publishers and subscribers are connected in a distributive manner where the publisher advertises its events using the publish e operation; whereas the subscribers express their interests on an event in the form of subscriptions using subscribe β operation. In this approach, each subscription and each event are associated with the time interval. Therefore, the subscription and the associated events are considered as valid within that the specified time interval.

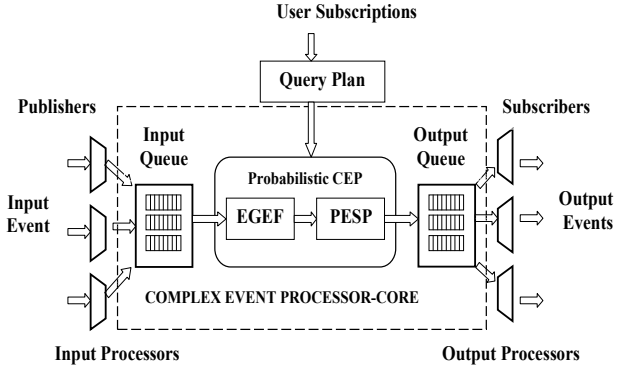

Figure 1. PCEP publish/subscribe middleware system.

The core part of the proposed approach is the PCEP. The incoming events are converted into the tuples suitable for the internal processing of the core. The CEP processor fetches the events from the incoming queue and processes the events based on the user subscriptions. The output events are derived by the processor. The derived output events are placed on the output queues. The proposed approach allows manipulating the queries on the fly. The following sub sections describe the description of the involved components.

2.1. Probabilistic Complex Event Processing

The core of the approach is the uncertain or PCEP that consists of two components such as Efficient Generic Event Filtering (EGEF) and Probabilistic Event Sequence Prediction (PESP). After receiving the large number of incoming events, PCEP processor de-serializes and processes the events based on user subscriptions. The PCEP is performed in two phases such as: EGEF and Probabilistic Event Sequence Prediction.

2.1.1. Efficient Generic Event Filtering

The EGEF is performed ahead of the event prediction to filter out the irrelevant events in order to achieve efficiency and scalability. EGEF is implemented in the publisher/subscriber model that performs matching among the large number of incoming events (publications) with the domain expert specified rules (subscriptions). Matching is performed in two steps as follows: First, cluster subscriptions are formed where in the user subscriptions are grouped into clusters and are mapped into their corresponding access predicates by setting the predicate bit vector into one. Second, the NFA matching is performed based on the Non-Deterministic Finite Automaton-Heap (NFA_h) query evaluation model between the large number of incoming events and user subscriptions. EF leads to the filtering out the irrelevant events at the early stage due to the efficient grouping of user subscriptions and the loading of common access predicates in each subscription group.

2.1.2. Complex Event Pattern Subscription Language

To provide a clear description of the user subscriptions in the form of event pattern queries, CEPSL is designed to model the uncertainties of the incoming event streams. It consists of three clauses named as SELECT, FROM and WITHIN. Among the three clauses, SELECT specifies the attributes required to select from the incoming event stream which may have either uncertain or certain attributes. Depending upon the specified attributes of the SELECT clause, two different variants of the FROM clause is declared. The FROM clause is the core part of the query that defines the stream expression to process a large number of incoming event streams. A stream expression is formulated in the form of predicates using the unary construct (filter) and two binary constructs called as (next and fold) [12]. These constructs perform the operation mentioned in the stream expression on the large number of incoming input stream and produces an output stream. A query to invest on a company for the product of a monotonic decrease of stock price less than 1,000 with volume of 5,000 is considered. Due to the decrease in stock price, the stock rebounds abruptly through increasing up to at least 7% in value.

As represented in the Figure 2, the CEPSL queries for the business domain of stock monitoring for a stock ticker stream of schema Stock (name, price and volume) is considered. According to the required pattern for the stock monitoring, CEPSL query is constructed to monitor the incoming event data streams. The innermost expression is a filter that specifies the stream expression to filter out the values of certain attributes such as: Name, quantity and price. The fold construct specifies the two predicates with certain attributes such as Name="MSFT" and Quantity=5,000 to filter out the events of MSFT product with a volume of 5,000. Finally, next construct finds the immediately next stock ticker event for the company. For uncertain attributes, FROM clause specifies the stream expression for the monotonic decrease of stock price [20]. The WITHIN clause outside the fold expression ensures that the monotonic price decrease has to last for at least 15minutes.

```
SELECT Company Name, Low Price, Price from Stock Ticker Stream
RS=FROM FILTER {Category= "SOFTWARE", DUR<15min} (Stock)      from
Stock Ticker Stream
            FOLD {$2Name=$Name, No. of shares >1000, DUR< 15min} (Stock)
            NEXT {$2Name = $1Name} (Stock)
FROM FILTER {Price<1,000} (Stock) from RS
            FOLD {Final Price<1.07*Low Price, DUR<15min} (Stock)
            NEXT {$2Name = $1Name} (Stock)
            WITHIN 15Min;
```

Figure 2. CEPSL query for stock marketing.

2.2. Query Plan based Approach

Query plan based approach is deployed to manage and to extend the expressiveness of the high volume of user subscriptions. Here, user subscriptions are expressed as a CEPSL. Query compilation is used to convert the event pattern queries into the NFA_h model based on a native sequence operators. It takes input as a query defined sequences from the large number of continuously arriving events that improve the flexibility of the query execution.

3.2.1. Predicate based Subscription Grouping

In order to, achieve scalability, complex queries of similar predicates are grouped into a cluster vector using the PBSG scheme. The cluster vector consists of 'n' number of subscriptions that contains a set of similar predicates. Each cluster vector maintains a predicate array. The cluster vector is an n-dimensional array that places the number of predicates present in the set of subscriptions in the cluster. A set of predicates in the sub-queries are assigned as an access predicate of the corresponding cluster [14]. Therefore, each subscription clusters might be associated with more than one access predicates.

3.2.2. Query Aware Partitioning

Query aware partitioning is performed using row/column scaling. In event matching engine, user subscriptions are organized in the form of matrix p×q, where 'p' is the number of access predicates and 'q' is the number of state machines that is equal to the number of subscription groups formed from the user subscriptions. Furthermore, constructed NFA_h automaton from each of the subscription group is loaded directly into the state machine. In order to, manage the large number of user subscriptions, the number of rows is replicated according to the increasing number of subscriptions [5].

3.3. Query Compilation

The NFA_h based event matching engine is to process the large number of incoming event sequences based on user subscriptions, which may be temporal or conditional queries. Here, event pattern queries are converted into a suitable query evaluation model called as a NFA model. A query compilation technique called as Automaton Intermediate Representation (AIR) is used to convert the defined event pattern CEPSL queries into a new form of automaton called as NFA_h. The constructed automaton is loaded directly into the state machine to perform event matching.

3.3.1. NFA_h Automaton

The NFA_h model consists of four edges such as begin edge, forward edge, filter edge and a rebind edge where each edge deploys a heap memory to store the active instance which is satisfied in its current state. In NFA_h, the forward edge is used to find the event using the filtering mechanism performed by the filter edge which filters the event using iteration operator. The access predicates are associated with each subscription groups at the edge of the automaton to perform matching.

The top loop of the automaton has a filter edge that takes a conditional argument. A predicate called as a name in the first Fold construct of the CPESL query is shown in Figure 3. This edge allows an automaton instance of the event scheme of company-Microsoft denoted as 'MSFT' and skips over the instance of other companies [2]. The bottom loop of the automaton is called as a rebind edge 'Q' that is used to express the filter constructs of the CPESL Query.

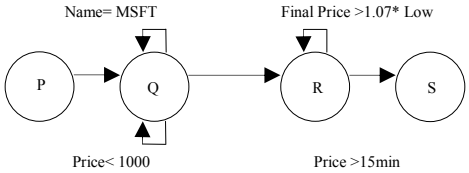

Figure 3. NFA_h automaton for CEPSL query.

3.3.2. NFA_h based Event Matching Engine

In NFA_h pattern matching, the matching engine comprises of a set of state machines deployed with NFA_h, where each edge is associated with a set of predicates the incoming events which traverse throughout the edge and reach the final state are considered as relevant events. The events, which are not traversed through the edges of the automaton, are filtered out as irrelevant events. Each state maintains the Active Instance Heap (AIH) to store the active instances of the event that trigger into the other transition state. If the incoming event reaches the accepting state of the NFA_h, it constructs the event sequence as a Directed Acyclic Graph (DAG) that starts from the state, the most recent event that has been completed of the stack. Event sequences reaching the final state of the automaton are filtered as a relevant sequence of events [11].

3.4. Probabilistic Event Sequence Prediction

PESP as in Figure 4 is used to derive a stateful composite event sequences from the filtered relevant events sequence based on the probabilistic framework.

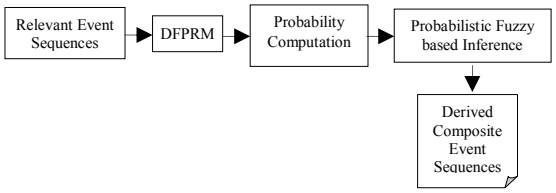

Figure 4. Efficient event sequence prediction paradigm.

Here, event hierarchy is constructed in the form of graphical model called as a DPFRM [24] that computes joint probability distribution using the conditional probabilistic dependencies between the event sequences in accordance of the rules. In order to, reduce the large sample space, fuzzy logic is used to infer the semantic correlation between the event sequences using linguistic variables.

3.4.1. Dynamic Probabilistic Fuzzy Relation Model

A DPFRM is automatically constructed with a set of relevant event sequences and rules to represent the probability space in terms of the concept of individuals, their properties and relations between them [15]. Qualitative knowledge between the set of events and their variable interrelationships is represented graphically while quantitative knowledge of specific probabilities is represented as Conditional Probability Distribution (CPD). Figure 5 shows the DPFRM for the event sequences with a set of events $E=\{e_1, e_2, ..., e_n\}$ and each event e_i is associated with a set of descriptive attributes and reference slots. This model consists of a set of event classes with its associated attributes, and then, the arbitrary correlation between the event sequences is computed by the joint probability distribution [15].

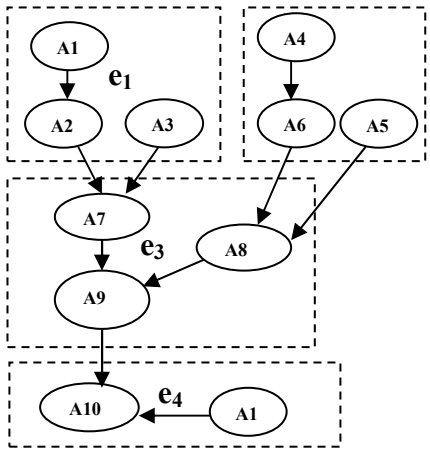

Figure 5. Dynamic fuzzy probabilistic relational model.

3.4.2. Probability Computation for Event Sequence

The constructed probabilistic model DFPRM is used to compute the joint probability distribution based on the conditional probabilistic independencies between the event sequences. The probability space is represented as triples $\{W_T, \Omega_T, \theta_T\}$ where 'W_T' is a set of possible worlds, 'Ω_T' is an event history associated with each possible world and 'θ_T' is a probability measure of the possible world. The sample space of an event sequence 'E' is represented as the conjunction of a set of possible events with its associated probability measure from the event history. The probability computation of the constructed event sequence is performed by the conditional probability dependence between the set of attributes associated with event e_i. The CPD of the event sequence is computed as follows:

$$P(e_i/e_{i+1})=\Pi\ P(e_i / e_{\Omega i, \Phi i}) \quad (1)$$

Here, 'Ω_i' is the set of parental nodes of' e_i' and then 'Φ_i' is the parameter vector associated with 'e_i' The CPD of event sequence 'E' is determined using the probability distribution over the values of events 'e_i' given each combination of values for its parents $P(e_i)$ [27]. More precisely, the Joint Distribution of the incoming event can be factorized into a product of the CPDs of all the uncertain attributes occurred in the event.

3.4.3. Probabilistic Fuzzy Logic based Inference Engine

In order to, formulate the combination of event sequences according to the computed probability of events with the reduced overhead, Probabilistic Fuzzy Logic (PFL) [21] is used to estimate the fuzzy linguistic variables from computed CPD in the large probability space. In order to, approximate the large sample space, fuzzy partitioning scheme partitions the sample space with a set of possible worlds into a certain number of predetermined membership functions as shown in Figure 6.

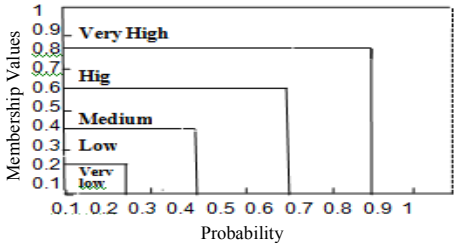

Figure 6. Fuzzy partitioning of large probability space.

Each sample space is composed of 'n' number of membership functions that span the entire input or output sample space of a fuzzy inference system. The probability of each event sequence must be 1 which is the sum of the associated probability with all sets of events in it. A fuzzy set consists of a set of linguistic variables, which is defined by domain experts in the form of characteristic function called as a membership function. It is represented as μ_F:$P(E)\rightarrow[0, 1]$ and is used to define the certainty that element $P(E)$ belong to the fuzzy set F. It is used to associate a degree of membership of each of the possible world in the sample space to the corresponding fuzzy set. A fuzzy set consists of five linguistic variables such as {very low, low, medium, high and very high}.

Figure 7 shows the PFL based inference engine that takes the probability of a combination of events as an input and generates the output of event sequences in order with a fuzzy value. It consists of three main components such as fuzzifier, rule base and inference engine.

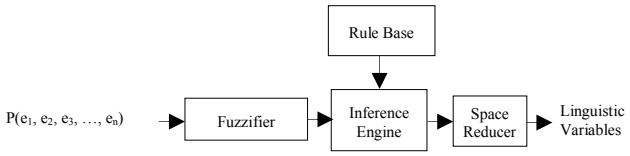

Figure 7. PFL based inference engine.

The fuzzifier is used to convert the input probability of the possible worlds (non fuzzy) of a sample space into a membership degree (fuzzy) with the help of membership function associated with each fuzzy set in the rule input space using normalization of maximum likelihood. The membership degree for the output set is computed from degrees of membership according to the relationships between input fuzzy values. Furthermore, a rule base defines the fuzzy rules provided by the domain experts of the system. The inference engine executes fuzzy logic inference to map the linguistic variable from the fuzzy sets based on the membership function in the rule base. The fuzzy values are assigned according to how well it matches with the membership degree of certainty belonging to the fuzzy set. It is a useful to reduce the sample space to derive the most probable event sequence approximately. Therefore, events with most priority fuzzy values are derived as a composite of event sequence.

4. Performance Evaluation

In this section, the proposed PCEP approach is implemented in the Publisher/subscriber model using Java Message Service (JMS) based subscription API and experiments are carried out based on streams of stock quote of l0,0000 events with one thousand stock symbols each. In this work, the proposed PCEP approach is implemented in stock marketing scenario to detect the relevant events from the large number of incoming events under uncertainty by running preliminary experiments. The two main goals of the performance evaluation are: To analyze the improvement in potential efficiency of query partitioning; and to evaluate the CEP performance when deploying multiple machines for event matching.

4.1. Experimental Setup

The generic application methodology is structured as a message oriented middleware with the set of software components to perform efficient event processing. These software components communicate through a messaging system called as JMS, which takes input as event instances from continuously arriving incoming events. The experiment is executed on Windows XP PC with 3.2GHz processor, 2GB of RAM and 512MB cache with the maximum Java heap size of 800 Mbytes. It is implemented in open source Java EnterpriseEdition/ Netbean/ Glassfish/ JMS environment. The proposed approach is applied to trigger events in a real time application of stock market for a financial domain. The performance of the proposed approach is tested using the event stream of 10,000 incoming events that consist of 11 attributes, where six attributes are certain attributes such as the name of the company, product, category, volume, number of shares, timestamp). The remaining five attributes are uncertain attributes such as stock price, Price-to-Earnings ratio (P/E), Price-to-Sales Ratio (PSR), Return On Equity (ROE), Earnings Growth (EG) and Debt-to-Asset ratio (D/A). In order to, trigger relevant events as per user subscriptions; it takes 500 user subscriptions which are subscribed by users (stock trader and stock investor).

4.2. Experimental Results

4.2.1. Throughput of Proposed CEP Scheme

Throughput of the CEP scheme is defined as the number of events processed for a second by the processor. In existing systems, heavy workload occurs because the only one processing engine is available to manage the incoming events.

- *Effect of Varying Number of State Machines in Throughput*: The performance of the proposed scheme is smooth even for the large number of queries, whereas in distributed Cayuga approach, the throughput is reduced at the starting stage of queries because the filtering is not performed in the case of a large number of incoming events. Figure 8 depicts that the proposed uncertain CEP with 10 state machines achieves high throughput. It also, shows that an increasing number of queries affect the number of state machines.

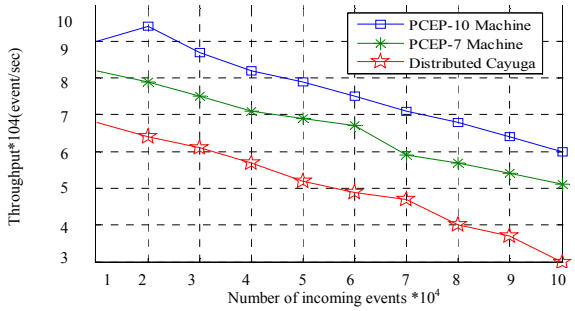

Figure 8. Throughput as a function of varying number of state machines.

From the graph, it can be found that both distributed Cayuga and PCEP systems can handle high event input rates, in the order of tens of thousands of events per second. Existing distributed Cayuga starts to drop input events at a rate of 60,000 events per second with maximum throughput. The proposed PCEP can handle

up to 90,000 events per second. In this perspective, the proposed PCEP outperforms existing distributed Cayuga in all scenarios. It can be noticed that the throughput of the proposed system is appreciable even for the large number of incoming events. In the case of distributed Cayuga, the throughput starts decreasing after the 30,000 (events/sec).

- *Effect of Varying Size of NFA_h Length in Throughput*: In NFA_h based event matching, the size of the NFA_h automaton gradually grows according to the increasing number of queries that may reflect on the number of state transitions required. The throughput of the system is affected by three parameters such as: The number of state transitions required to process each event, the number of predicates on the state of the automaton needed to evaluate the queries and the memory requirements to store the NFA automaton. The proposed approach deploys a query aware partitioning with the help of a predicate based subscription grouping that manages similar sub-queries among the different queries in the subscription cluster. It clearly eliminates the redundant states even under the large number of queries and also achieves faster execution with high throughput.

Figure 9 shows how throughput of the proposed PCEP declines as the number of states of the automaton increase. The proposed approach manages the throughput approximately 90,000 (events/sec) with the sequence length of 3 states, but it gradually decreases into 40,000 (events/sec), when the sequence length of the automaton increases from 3 to 10. Thus, throughput declines approximately 40% each time when the NFA size doubles, it assures that the number of states in the automaton is appropriately constructed by query aware partitioning and PBSG approach.

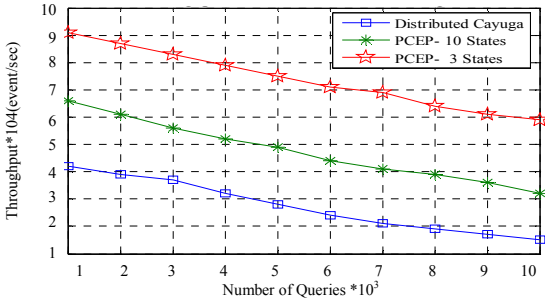

Figure 9. Throughput as a function of NFA length.

- *Effect of Varying Possible World Space in Throughput*: In the proposed approach, PFL based inference is used to derive the most probable event sequence approximately with reduced sample space. In existing probabilistic based inference, probability sample space consisting of a large number of possible worlds of event sequences is defined by the EID at a time 't1'. Therefore, the difference set of possible world will be available at a different time points: 't2' and 't3'. So, the probability sample space may vary according to the time. All the same, it is difficult to perform the probabilistic inference on the large number of possible worlds of the sample space.

Figure 11 shows how throughput of the proposed PCEP approach that deploys a PFL inference that achieves high throughput (events/sec) even under the large number of incoming events because of reduced sample space. On the contrary, existing probabilistic inference based approach, throughput declines as the number of possible world increase.

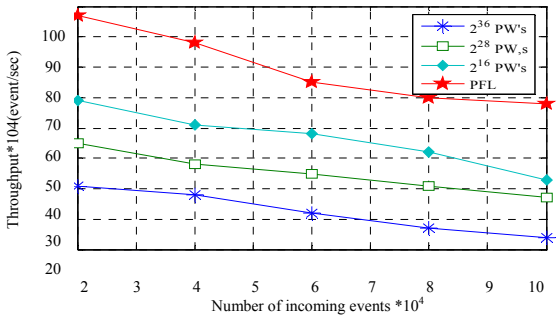

Figure 11. Throughput as a function of probability space.

4.3. Scalability of Proposed PCEP Scheme

Scalability is the unified performance metric that is effectively magnified with respect to average processing time, throughput and workload. The proposed approach deploys an efficient event filtering approach. It achieves high scalability due to the filtering of irrelevant results. Figure 12 depicts that the scalability of the proposed scheme with filtering is significantly enhanced better than distributed Cayuga scheme. Scalability of the distributed Cayuga scheme is maintained around 60-70% because it incurs less throughput and high average processing time for the large number of incoming events.

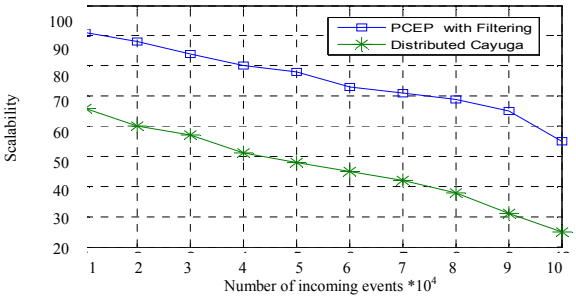

Figure 12. Scalability of PCEP with and without filtering.

5. Conclusions

In this paper, a probabilistic framework named as PCEP is proposed. PCEP performs complex event detection even in the presence of uncertain data, while at the same time offers a declarative language CPESL to express the event pattern queries over uncertainty.

This PCEP approach is implemented in two phases such as EGEF and probabilistic event sequence prediction paradigm. The EGEF is an event filtering approach and is a pre-processing system that filters out the relevant events based on user subscriptions in order to manage the arrival of a large number of incoming events. A subscription language CEPSL is presented. CEPSL combines elements from Cayuga and CLUES to define the uncertainty. Furthermore, the NFA_h based query evaluation approach performs event matching that allows subscriptions to filter the relevant events from the large number of incoming events.The performance of the system for heterogeneous types of events containing textual, visual as well as aural contents can also be considered as future enhancement [3].

References

[1] Al-Shaer E., "High-Performance Event Filtering for Distributed Dynamic Multi-point Applications: Survey and Evaluation," *Technical Report*, Old Dominion University, VA, USA, 1997.

[2] Agrawal J., Diao Y., Gyllstrom D., and Immerman N., "Efficient Pattern Matching over Event Streams," *in Proceedings of the Conference on Management of Data*, USA, pp. 147-160, 2008.

[3] Alfian A., Mandava R., and Mohammad A., "Soccer Event Detection via Collaborative Multimodal Feature Analysis and Candidate Ranking," *the International Arab Journal of Information Technology*, vol. 10, no. 5, pp. 493-502, 2013.

[4] Bakshi A., "Introducing and Reasoning about Uncertainty in MIndiGolog," available at: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.121.3020, last visited 2008.

[5] Brenna L., Gehrke J., and Johansen D., "Distributed Event Stream Processing with Non-Deterministic Finite Automata," *in Proceedings of the 3rd ACM International Conference on Distributed Event-Based Systems*, USA, pp.1- 12, 2009.

[6] Campailla A., Chaki S., Clarke E., Jha S., and Veith H., "Efficient Filtering in Publish-Subscribe Systems using Binary Decision Diagrams," *in Proceedings of the 23rd International Conference on Software Engineering*, USA, pp. 443-452, 2001.

[7] Chandrasekaran S., Cooper O., Deshpande A., Michael F., and Joseph H., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," *in Proceedings of the International Conference On Management of Data*, New York, USA, pp. 668-668, 2003.

[8] Corsaro A., Querzoni L., Scipioni S., Piergiovanni T., and Virgillitoa A., "Quality of Service in Publish/Subscribe Middleware," available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.1956&rep=rep1&type=pdf, last visited 2006.

[9] Cugola G. and Margara A., "Processing Flows of Information: From Data Stream to Complex Event Processing," *in Proceedings of the 5th International Conference on Distributed Event-based System*, USA, pp. 359-360, 2011.

[10] Dalvi N. and Suciu D., "Efficient Query Evaluation on Probabilistic Databases," *International Journal on Very Large Data Bases*, vol. 16, no. 4, pp. 523-544, 2004.

[11] Demers A., Gehrke J., Hong M., Riedewald M., and White W., "Towards Expressive Publish/Subcribe Systems," *in Proceedings of the 10th International Conference on Advances in Database Technology*, Springer-Verlag, Heidelberg, pp. 627-644, 2006.

[12] Demers A., Gehrke J., and Panda B., "Cayuga: A General Purpose Event Monitoring System," available at: http://www.cs.cornell.edu/johannes/papers/2007/2007-cidr-cayuga.pdf, last visited pp. 412-422, 2007.

[13] Eckert M. and Bry F., "Complex Event Processing," *Informatik-Spektrum*, vol. 32, no. 2, pp. 163-167, 2009.

[14] Franchoise F., Jacobsen A., Llirbat F., Pereira J., Ross A., and Shasha D., "Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems," available at: http://msrg.org/publications/pdf_files/2001/FaJL01-Filtering_Algorithms_and_Implementatio.pdf, last visited 2001.

[15] Gatoor L., Taskar B., and Kollar D., "Selectivity Estimation using Probabilistic Models," available at: http://www.seas.upenn.edu/~taskar/pubs/sigmod01.pdf, last visited 2001.

[16] Gyllstrom D., Wu E., Chae H., Diao Y., Stahlberg P., and Anderson G., "SASE: Complex Event Processing over Stream," available at: http://arxiv.org/ftp/cs/papers/0612/0612128.pdf, last visited 2007.

[17] Hebrail G., "Data Stream Management and Mining," available at: http://bilab.enst.fr/fichiers/NICSP19-0089.pdf, last visited 2008.

[18] Kyoung B., Myung L., and Jae Y., "Efficient Complex Event Processing over RFID Streams," *the International Journal of Distributed Sensor Networks*, vol. 2012, pp. 9, 2012.

[19] McCarthy D. and Dayal U., "The Architecture of an Active Data Base Management System," available at: http://citeseerx.ist.psu.edu/viewdoc/download?do

i=10.1.1.465.9959&rep=rep1&type=pdf, last visited 1989.

[20] Nie Y., Zhanhuai L., and Chen Q., "Complex Event Processing over Unreliable RFID Data Streams," *in Proceeding of the 13th Asia-Pacific Web Conference on Web Technologies and Applications*, China, pp. 278-289, 2011.

[21] Nunes I., "Handling Human-Centered Systems Uncertainty using Fuzzy Logics-A Review," *the Ergonomics Open Journal*, vol. 3, pp. 38-48, 2010.

[22 Poul N., Migliavacca M., and Pietzuch P., "Distributed Complex Event Processing with Query Rewriting," *in Proceedings of the 3rd International Conference on Distributed Event-based Systems*, USA, pp. 1-4, 2009.

[23] Re C., Letchner J., Balazinska M., and Suciu D., "Event Queries on Correlated Probabilistic Streams," *in Proceedings of the 4th International Conference on SIGMOD'08*, Canada, pp.715-728, 2008.

[24] Sanghai S., Domingos P., and Weld D., "Dynamic Probabilistic Relational Models," *in Proceedings of the 18th International Joint Conference on Artificial Intelligence*, USA, pp. 992-997, 2003.

[25] Sarma A., Benjelloun O., Halevy A., and Widom J.,"Working Models for Uncertain Data," *in Proceedings of the 22nd International Conference on Data Engineering*, USA, pp. 1-21, 2006.

[26] Soliman A., Ilyas F., and Chang C., "Top-k Query Processing in Uncertain Database," *in Proceedings of the 23rd International conference on Data Engineering*, Istanbul, pp. 896-905, 2007.

[27] Wang Z., Michelakis E., and Tancau L., "Probabilistic Complex Event Triggering," *Technical Report-EECS-2009-114*, EECS Department, University of California, Berkeley, 2009.

[28] Wu E., Diao Y., and Rizvi S., "High-Performance Complex Event Processing over Streams," *in Proceedings of the International conference on Management of data*, USA, pp. 407-418, 2006.

[29] Yi K., Li Feifei, Kollios G., and Srivastava D., "Efficient Processing of Top-k Queries in Uncertain Databases with x-Relations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 12, pp. 1669-1682, USA, 2008.

Govindasamy Vaiyapuri is a research scholar in Department of CSE, Pondicherry Engineering College, Puducherry, India. He is currently working as Assistant Professor in Department of IT, Pondicherry Engineering College, Puducherry, India.

Thambidurai Perumal is at present working as Principal and Professor of Computer Science and Engineering, Perunthalaivar Kamarajar Institute of Engineering and Technology (PKIET), Karaikal, Union Territory of Puducherry, India. He has completed his M.E.(CSE) from Anna University, Chennai in 1984 and PhD (CSE) from Alagappa University, Karaikudi in 1995. He is Fellow Institute of Engineers, Life Member in CSI and ISTE. He has published more than hundred research papers in National and International Journals and Conferences. His area of interest includes networking, image processing, natural language processing, and information security.