Probabilistic Event Processing with Negation Operators in Cyber Physical Systems

Na Mao, Jie Tan
Integrate Information System Research Center
Institute of Automation, Chinese Academy of Sciences
Beijing, China
{*na.mao, jie.tan*}*@ia.ac.cn*

***Abstract*—With the widely used sensor devices in many CPS applications, tremendous amounts of uncertain events are produced, which are too simple to be directly used. So Complex Event Processing (CEP) is introduced to get the complex and meaningful information based on event patterns or rules from the uncertain raw data. Meanwhile, the negation operator for event processing is common and useful in CPS applications, so it is necessary to study the probabilistic event processing method with negation operators. In this paper, we take the mixed-model assembly line as an example for a CPS application. We describe a probabilistic event model and present Conditional Probability Matrix (CPM) to calculate the probability of target complex events. A probabilistic complex event detection algorithm with negation operators (PCED-N) with rNFA structure based on SASE is proposed. The experimental results show that our proposed method is efficient to detect complex events over probabilistic event streams with negation operators and it is sensitive to the pattern lengths.**

***Keywords*-Cyber Physical Systems; Probabilistic Complex Event Processing; Negation Operators; Mixed-model Assembly Lines**

I. Introduction

Cyber Physical Systems (CPS) are controlled complex technical systems like modern automobiles, aircrafts, power grids, production lines, the internet of things, and so on. They exist as complex physical systems in the "real world" with their behavior determined by physical and technical rules and embedded in a physical environment [1]. The cyber systems take the information from the physical systems with the form of events [2]. Because sensor devices are widely used in many CPS applications, tremendous amounts of events are produced. However, the events generated from sensor devices are too simple to be directly used, so we pay more attention to complex information that we can get from the simple events. Complex Event Processing (CEP) [3] is introduced to process huge primitive events based on event patterns or rules to detect the complex and meaningful information from them.

Besides, the raw data from the physical world is of high level of uncertainty because of the inherent inaccuracy of sensor readings. The uncertainty refers to the readings which cannot be determinately eliminated by cleansing rules due to their essence, such as the possibility in producing the same product up to standard in the same operation by different workers. However, some existing complex event processing systems, such as SASE [4] and Cayuga [5], assume that data is precise and certain, or it has been cleansed before processing. They can't detect probabilistic complex events from uncertain event streams well. Lahar [6] system can process RFID data with uncertainty, but it mainly focus on the probabilistic data model and related queries, and it doesn't support negation operators well.

As a result, we define a probabilistic event model and describe a probabilistic event detection model with negation operators. And we use Conditional Probability Matrix (CPM) to calculate the probability of target complex events. A Probabilistic Complex Event Detection Algorithm with Negation operators (PCED-N) based on rNFA structure is proposed in this paper.

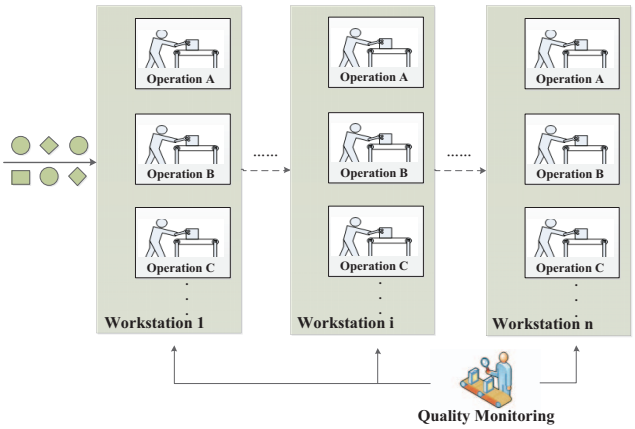


Figure 1. A Simple Scenario in Mixed-model Assembly Lines.

In this paper, we take production lines as examples for a CPS application and illustrate our ideas through a simple scenario in the product manufacturing process showed in Figure 1. In mixed-model assembly lines, different types of products are assembled in the same production line at the same time with different parts. After finishing all operations in a certain workstation, the product will transform to another workstation for further operations. But they need different operations at the same workstation. For example, a product needs to take operation A and operation C without

978-1-4673-9200-6/15 $31.00 © 2015 IEEE
DOI 10.1109/CyberC.2015.66

operation B between them, while another product is designed to take three operations A, B and C sequentially in a period of time. It is necessary to monitor the operations sequentially taken for producing the right products. At the same time, in order to ensure the product quality, we monitor the quality of the Working in Process (WIP) when every operation is finished. We use CEP for fast and effective recognition. Because every worker has a certain probability to produce a qualified product, the event generated from this scenario is uncertain. If the probability of a complex event is less than the threshold, an alarm is made by the system.

The outline of the paper is organized as follows. Section II analyzes the related work. In Section III, we present a probabilistic event model. A probabilistic event detection model with negation operator is proposed in Section IV. The performance evaluation of our proposed method is described in Section V. Finally, the conclusion is in Section VI.

II. Related Work

Complex event processing is used widely to find interesting or unusual events to users in CPS applications from different fields. Wang et al. [7] devised RFID application logic into complex events and developed an RFID event detection engine to process complex RFID events with the use of pseudo events. The paper [8] demonstrated how complex event processing and reactive rules allow the description of CPS behavior on the event and action level and integration of control into federated simulations. In the paper of [9] proposed a CEP framework to model surgical events and critical situations in an RFID-enabled hospital and implemented a prototype system for surgical management. The authors in [10] proposed a Cyber-Physical Social System architecture for energy management with Twitter integration based on CEP. Especially, CEP is also applied to monitor the status of production lines for effective production [11] [12]. But recent studies mainly discuss the framework of applying CEP to manufacturing scenarios instead of the specific method of event detection.

Some works have been carried out to study the uncertainty for CEP based on NFA. A high performance complex event processing method is proposed with Nondeterministic Finite Automaton and Active Instance Stacks over distributed probabilistic event streams in [13]. In the paper of [14], the authors proposed an optimized method with DFA automaton and matching tree to detect complex events against uncertain raw input data stream. However, they doesn't suit to detect events with negation operators in patterns. The paper [15] proposed a complex event processing system named SASE+, which support Kleene closure and negation operators over precise streams. But it doesn't support for uncertain event streams. So in this paper, we study the probabilistic event processing method with negation operators in CPS applications on the basis of SASE.

III. Probabilistic Event Model

Our probabilistic event model consists of uncertain event model and rules. An uncertain event is directly detected by sensor devices and can be denoted by a tuple E_P = $\langle$ *id, eventType, timestamp,* $\{$*attributes*$\}$, $\{$*Probability, (0, 1)*$\}$ $\rangle$, where *id* denotes the unique identifier of the event, *eventType* is the type of the event, *timestamp* denotes the occurrence time of the event, $\{$*attributes*$\}$ represents a set of attributes of the event, composing name and value for an attribute. Especially, *Probability* with the value between 0 and 1 means the occurrence probability of the event.

For example, an event in the example scenario mentioned above represented as e = $\langle$ 1 , "Operation A", "2015-06-05 11:03:20", {Workstation, 2}, {WorkerID, 3}, {ProductID , 1010}, {probability, 90%}$\rangle$, which is a probabilistic event that indicates a worker with Number 3 takes Operation A in the second workstation for the product with Number 1010 at the time 2015-06-05 11:03:20, and the probability of qualified operation is 90%.

An uncertain complex event is composed by events from the uncertain event streams, which can be represented as CE_P = $\langle$ *E, R,* T_S*, Pr*$\rangle$, where *E* represents the events that compose the complex event, R represents the rule of the combination, T_S represents the time span of the complex event. We define *Pr* the probability that the complex event occurs. It can be considered as the reliability of the pattern query during the corresponding the time interval T_S and calculated in the complex event matching process.

Complex events are processed according to rules. The rule formulation is defined with event language. In this paper we use a declarative pattern matching language proposed in SASE [16]. A complex event query is defined using the syntax as follows:

[FROM $\langle$ input stream $\rangle$]
PATTERN $\langle$pattern structure$\rangle$
[WHERE $\langle$ pattern matching condition $\rangle$]
[WITHIN $\langle$ sliding window $\rangle$]
[HAVING] $\langle$ pattern filtering condition $\rangle$
[RETURN $\langle$ output specification $\rangle$]

The FROM clause specifies the input stream we used, if this clause is omitted, the default stream is used as input stream. The PATTERN clause declares the structure of a pattern to be matched against the input stream. The WHERE clause contains attribute-based predicates to restrain the relevant events defined in the pattern. The WITHIN clause specifies a sliding window over the pattern, defining the constraints that a complex event should occur. After PATTERN, WHERE and WITHIN generate pattern matches, the HAVING clause further filters each pattern match by applying predicates on the constituent events. A pattern match that satisfies the HAVING predicate is retained for output. Finally, the RETURN clause transforms each pattern match into a result event.

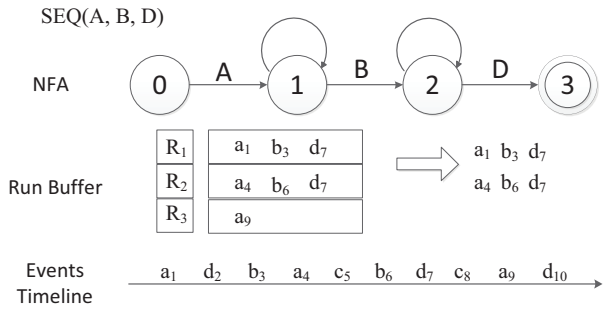


Figure 2. Example of rNFA Structure.

IV. Probabilistic Event Processing with Negation Operators

In this section, we introduce our method for probabilistic event processing with negation operators. We propose an event detection model named rNFA that can support negative operators first. And then we show a probability calculation model and a probabilistic event detection algorithm to match the pattern with negation operators from uncertain event streams.

A. Event Detection Model

SEQ event detection is the main part of the CEP engine. The fixed data structure such as finite automata, tree, graph and Petri net [7] [17] [18] are used to represent the structure of the event pattern for event detection. In this paper, we focus on the SEQ event processing for an event stream based on Nondeterministic Finite Automata (NFA) [19]. Besides, other data structure is needed in order to record the event detection process. So we propose the rNFA structure which is NFA combined with run buffers.

Formally, a rNFA structure is represented as $A = (Q, E, R, \delta, q_1, F)$, where Q denotes a set of states, E is a set of directed edges, R denotes a set of runs used, δ means the value vector, which consists of the inherent attributes of a run, such as probability and timespan. Meanwhile, q_1 is the start state and F is the final state. The runs of rNFA are used to record the events that have partial matching result for the target event pattern. Those runs are stored in the run buffer. Moreover, an accepting run is a run that has reached the final state.

The rNFA for SEQ (A, B, D) is illustrated in Figure 2. We can see that the event detection is started at the "0" state of the NFA and it enters the next "1" state when event A is coming. When a relevant event B is scanned, it goes to "2" state. If the "3" state of the NFA is entered after event D comes, the detection enters the final state which means that the required complex SEQ event is detected. At the same time, we create a set of runs such as R_1, R_2 and R_3 to record the valid events that have been detected for the target event pattern. Specifically, when event a_1 comes and it can match the pattern component (A) or the starting condition of a run, we create a new run R_1 in the run buffer and add event a_1 to R_1. At the same time, we record the current state of R_1 is "1" state according to the NFA structure. And then, only if a new coming event can match the pattern component (A, B) in the existing runs, it is added to the relevant run, like b_3. When event a_4 is coming to satisfy the starting condition of a run, we create a new run R_2 in the run buffer. When event d_7 comes, R_1 and R_2 become accepting runs which means (a_1, b_3, d_7) and (a_4, b_6, d_7) are the matched complex events.

However, if the pattern structure for event detection contains the negation operator, which is labelled as "!", we should model the subsequence without negation first. For example, if the required pattern is SEQ (A, B, !C, D), we create the rNFA structure for subsequence type (A, B, D), detect the matched complex event and store the intermediate results in a run of rNFA. When the subsequence has matched, we consider the negative part ignored before. That is, we check whether a C event has occurred in the timespan of the complex event in the related run of rNFA.

It is worth noting that there are three scenarios about the timespan of complex events [16]: (1) for a sequence of SEQ (A, !B, C), the timespan is defined as (A.timestamp, C.timestamp); (2) for SEQ (!A, B, C), the window size T and the timestamp of a C event are used to set the time interval to be (C.timestamp - T, C.timestamp); and (3) SEQ (A, B, !C) represents that a C event is not allowed to follow event A and B with the window size T, so the timespan is defined to be (A.timestamp, A.timestamp + T).

B. Probability Calculation Model

The complex event detected from uncertain input stream is also uncertain. When negation operators exist in a pattern for uncertain CEP, a new probability calculation model is needed to express the probability of the complex event.

Based on rNFA structure, we ignore the negation part and find that some of the primitive events have Markov property when calculating SEQ events. That is to say, the probability of the next event in the sequence is only related to the current event but has nothing to do with the previous events. The probability of the sequence with Markovian events can be calculated by the multiplication theorem of probability [20]. In the example mentioned above, some operations are not designed for a product at a workstation, and they are negation parts in event detection. For the product quality, the final quality is related with all the finished operations. So the probability with negation operators in mixed model assembly lines doesn't consider the negation part. Therefore, we can calculate the probability of an SEQ event with negation operators as follows.

For an SEQ event E = SEQ (e_1, e_2, ..., e_n), the input stream is partitioned into set S and set N, where set N contains the primitive events with negation operators and

set S contains one or more Markov chains for positive subsequence. The probability of event E is represented as follows:

$$Pr(E) = \prod_{S_i \in S} (Pr(e_{i1}) \prod_{m=1}^{|S_i|-1} Pr(e_{m+1}|e_m)) \quad (1)$$

where e_{i1} means the first event in the Markov chain S_i that is in set S, and $Pr(e_{m+1}|e_m)$ denotes the conditional probability of the events in the Markov chain.

Because the qualified rate of products differs by workers and operations in manufacturing lines, we introduce Conditional Probability Matrix (CPM) to describe the dependence of probability between the current event and the previous. Assuming that there are *m* workers assigned to take Operation k and *n* workers for Operation k+1. The matrix $M = (p_{ij})_{m \times n}$ ($i \in [0, m)$, $j \in [0, n)$) saves the conditional probability from Operation k to the next operation and p_{ij} means the conditional probability when the (*j*+1)th worker finished Operation k before the (*i*+1)th worker finished the next operation.

For example, the matrix M below saves the conditional probability matrix from Operation A to Operation B. So if Operation A is taken by worker 2 and the Operation B is finished by worker 1, the conditional probability chose for probability calculation is 0.85.

$$M = \begin{bmatrix} 0.90 & 0.87 \\ 0.85 & 0.88 \\ 0.78 & 0.95 \end{bmatrix}$$

However, the probability distribution and the CPM are obtained through the statistics or machine learning from historical event streams. Thus, it should be updated in a period of time.

C. Probabilistic Complex Event Detection Algorithm with Negation Operators

On the basis of the existing stream processing engine SASE, we propose a novel probabilistic complex event processing method based on rNFA to detect the pattern with negation operators from a probabilistic event stream. The basic idea of our method is that we ignore the negation part and deal with the positive subsequence with rNFA model. When any event is selected to add a related run, we calculate the probability of the current run. If the run is an accepting run, we check whether the negation part is appeared in the timespan of the run. If the negation part does not occur, a complex event is detected with the calculated probability. This algorithm is shown in Algorithm 1.

As described in Algorithm 1, the function *indexNegationForRun(e)* is used to store the negative event e for further judgement. We use *createNewRunForNegation(e)* to create a new run r which is in activeRuns and add event e to the run r. At the same time, we initialize the probability of r with the probability of e. The function *evaluateEventToRunForNegation(e,r)* is used to evaluate whether event e belongs to the run r. If event e is selected for run r, we add event e to run r and calculate the probability of the current run r. Finally, the function *outputMatchForNegation(r)* outputs the matched complex event with the probability of the current run r.

Algorithm 1 Probabilistic Complex Event Detection Algorithm with Negation Operators (PCED-N)

Input: Uncertain Event Stream s, Complex Event Query q.
Output: Complex Event Set CE.

```
1: initialization and build NFA by subsequence of Query q
   without negation operators; label the negation state NS;
2: initialize the set of runs: activeRuns and negationRuns;
3: for all instance e in uncertain event stream s do
4:     if e belongs to the negation part then
5:         indexNegationForRun(e);
6:     else if e triggers the start state of NFA then
7:         createNewRunForNegation(e);
8:     else if e triggers the state transition of NFA then
9:         for all run r in the run set activeRuns do
10:            evaluateEventToRunForNegation(e,r);
11:            if run r is an accepting run then
12:                if NS.timestamp is not in r.timespan then
13:                    outputMatchForNegation(r);
14:            delete run r from the run set activeRuns
15:    else discard event e
16: end
```

To explain our method well, Figure 3 is used to show our algorithm for SEQ (A, B, !C, D) in the example. That is to say, we want to retrieve the complex event that means Operation A, B and D in the same workstation for the same product should be operated sequentially without Operation C between Operation B and D within 10 units. When the event stream comes, event a_1 triggers the start state of the NFA structure described in Figure 3. So we add a_1 to the new run R_1 and set the probability of R_1 to be 0.95. Because event d_2 doesn't satisfy the condition of state transition of the NFA, it will be discarded. And event b_3 triggers the NFA to transfer from "1" state to "2" state. Then we check the predicate, that is, compare b_3 and a_1 with the value of *workstation* and *product*, and we find they are the same. So b_3 is added to R_1 and the probability of R_1 is updated by 0.95×0.78. Similarly, a_4 is added to a new run R_2 and b_6 is added to R_2. Because event c_5 is the negation part, so add c_5 to a list for negation. When event d_7 arrives, it triggers the NFA to transfer from "2" state to "3" state. But it just satisfies the predicate for R_1. We add d_7 to R_1 rather than R_2. The probability of R_1 is updated to $0.95 \times 0.78 \times 0.85$. R_1 is an accepting run and we check whether there is a negation event C appeared in the timespan of [3, 7]. We can

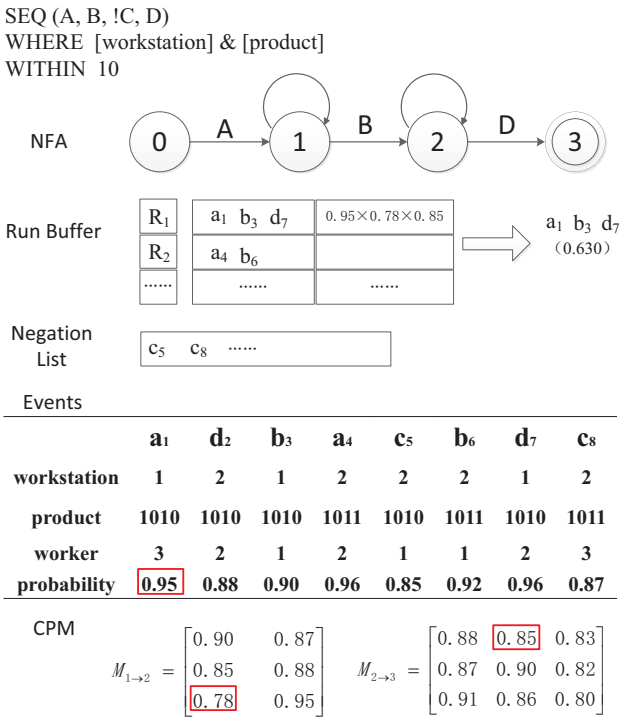


Figure 3. Example of Probabilistic Event Detection Algorithm with Negation Operators.

see that c_5 is occurred at timestamp of 5, but the value of *workstation* is not the same as that of the run R_1. Thus, (a_1, b_3, d_7) is a match that satisfies the pattern, so we output R_1 for a matched complex event (a_1, b_3, d_7) with the probability of 0.630.

V. Experimental Results

In this section, we perform a series of experiments based on the application scenario mentioned above in our prototype system and present a detailed performance analysis of our method. We use throughput to evaluate the performance of the proposed approach. The throughput defines the number of events processed for a second by the system. In order to get a constant performance, we repeat 20 times for the average value.

A. Experimental Setup

The experiment is run on Microsoft Windows 7 operating systems, Intel i7-4510U 2.0GHz and 2.6GHz CPU processor, 8GB memory, and the prototype system is implemented with Java. The JVM maximum allocation pool size is 1GB. To test the proposed method, we implement an event generator to create a synthetic stream of probabilistic events through the parameters shown in Table I.

In real applications, data is collected from different sensor devices and then fused to a single stream. In our experiment, we assume that there are 100000 events in the event stream

Table I
Parameters for Event Generation

Parameter	Description	Value
S	Number of the uncertain events	100000
WS	Number of workstations	2
P	Number of products in processing	10
W	Number of workers per operation	5
T	Number of event types (operations)	6
Pro	Range of probability	[80%,99%]

Query 1:
```
PATTERN   SEQ (A, !B, C)
WHERE  {
        [Workstation]
 AND  ProductID = 1010
}
WITHIN  TIME-WINDOW
RETURN   *
```

Query 2:
```
PATTERN   SEQ (!A, B, C)
WHERE  {
        [Workstation]
 AND  ProductID = 1010
}
WITHIN  TIME-WINDOW
RETURN   *
```

Query 3:
```
PATTERN   SEQ (A, B, !C)
WHERE  {
        [Workstation]
 AND  ProductID = 1010
}
WITHIN  TIME-WINDOW
RETURN   *
```

Figure 4. Queries for Experiments.

with 6 event types(operations) and ten kinds of products are produced in the same manufacturing line, which has two workstations with five workers on each operation. Noting that the range of probability in producing qualified product is set to be [80%, 99%] for the quality monitoring in the real application.

We check three kinds of patterns in our experiment. Figure 4 gives the three queries. They describe the different positions for the negation operator. The queries aim to find the target patterns and output all the possible sequences with corresponding probability to each complex event. TIME-WINDOW can be varied to evaluate different sizes of the time window.

B. Experimental Results

In this subsection, we evaluate performance of the proposed approach with the throughput of the system and analyze the sensitivity of necessary parameters such as the position of the negation operator and the length of the pattern.

1) Evaluation by Different Negation Operator Positions: In this experiment, we investigate the sensitivity of positions of the negation operator to our algorithm. We deploy three queries in Figure 4 to evaluate the throughput of PCED-N algorithm. By varying the window size, the experimental result is shown in Figure 5.

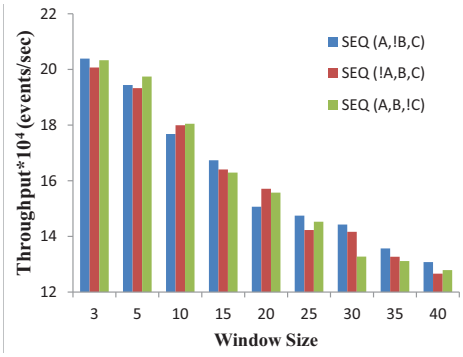


Figure 5. Throughput over Different Negation Operator Positions.

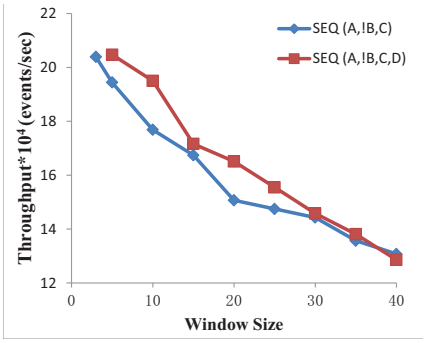
From the figure, we can see that as the growing of the time window, the throughput decreases for the three patterns. That is to say, longer window size will reduce the ability of processing. However, the throughput of the three patterns differs not much for the same window size. So we can conclude that the position of the negation operator has less effect on the PCED-N algorithm.

2) Evaluation by Different Pattern Lengths: We take the experiment to study the sensitivity of pattern length to our algorithm. Figure 6 describe the experimental results about the throughput of different negation position with different pattern lengths as window size changing. We observe that the throughput of the pattern with four operands differs little when the negation operator in different positions. This validates the analysis of the previous experiment. So we consider Figure 6(a) as a representative.

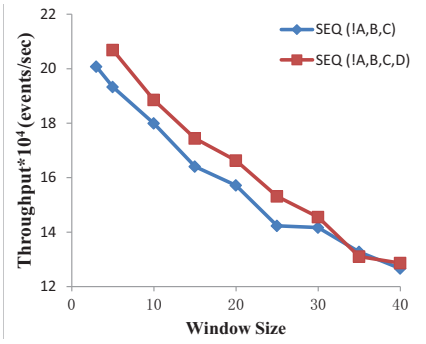
Figure 6(a) shows the throughput over different pattern lengths for the middle negation operator. We can obtain that the trend of throughput for the two pattern lengths has dropped for the three negation operator positions when the window size increases. But pattern (A, !B, C, D) with four operands has higher performance than pattern (A, !B, C) with three operands when the window size is less than 30. While the window size is increased over 30, there is little difference between the two pattern lengths.

VI. Conclusion

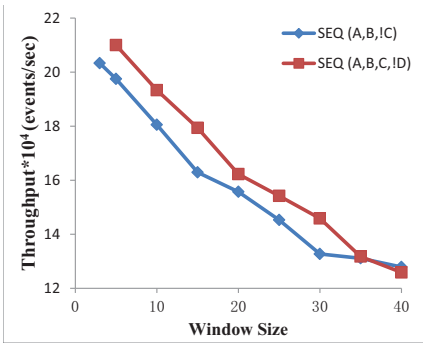
It is necessary to study the probabilistic event processing method with negation operators since which is common and useful in CPS applications. In this paper, based on the research of SASE, we extend its language to process uncertain event streams and describe a probabilistic event model. And we present Conditional Probability Matrix to calculate the probability of target complex events. Meanwhile, a probabilistic complex event detection algorithm with negation operators (PCED-N) based on rNFA structure is proposed. The experimental results show that our proposed approach is efficient to detect complex events over probabilistic event streams with negation operators and it is sensitive to the pattern lengths.



(a) Throughput for the Middle Negation Operator.



(b) Throughput for the Left Negation Operator.



(c) Throughput for the Right Negation Operator.

Figure 6. Throughput over Different Pattern Lengths.

Acknowledgment

The research is supported by the National Natural Science Foundation of China under Grant U1201251 and National Science & Technology Support Plan of China under Grant 2015BAF09B01.

References

[1] R. Klein, J. Xie, and A. Usov, *Complex events and actions to control cyber-physical systems.* The 5th ACM International Conference on Distributed Event-Based Systems (DEBS'11), pp. 29-37, July 2011.

[2] C. Talcott, *Cyber-Physical Systems and Events.* In: Software-Intensive Systems and New Computing Paradigms. Springer-Verlag, Berlin, Heidelberg, pp.101-105, 2008.

[3] Luckham D.C., *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston: USA. 2001.

[4] Gyllstrom D, Wu E, Chae H J. *SASE: Complex Event Proeessing over Streams*. Rro of CIDR, pp.407-411, Dec 2007.

[5] Brerma L, Demers A, Gehrke J.*Cayuga: a high-performance event Processing engine*. Proeeedings of the 2007 ACM SIGMOD international conference on Management of data. pp.1100-1102, June 2007.

[6] R-e C, Letehner J, Balazinska M. *Event Queries on Correlated Probabilistic Streams*. Proc of the SIGMOD Conf. pp.715-72, June 2008.

[7] F. Wang, S. Liu, P. Liu and Y. Bai, *Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams*. Lecture Notes in Computer Science, pp.588-607, 2006.

[8] R. Klein, S. Rilling, A. Usov and J. Xie, *Using Complex Event Processing for Modelling and Simulation of cyber-physical systems*. Int. J. Critical Infrastructures, Vol. 9, Nos. 1/2, 2013.

[9] W. Yao, C.H. Chu and Z. Li, *Leveraging Complex Event Processing for Smart Hospitals Using RFID*. Journal of Network and Computer Applicaions 34(2011), pp.799-810.

[10] D.N Crowley, E. Curry and J.G. Breslin, *Closing the Loop - from Citizen Sensing to Citizen Actuation*. IEEE International Conference on Digital Ecosystems & Technologies, pp.108-113, 2013.

[11] C. Zang and Y. Fan, *Complex event processing in enterprise information systems based on RFID*. In: Enterprise Information System, Vol. 1, No. 1 (2007), pp.3-23.

[12] W. Ahmad, A. Lobov, J. Lastra, *Formal Modelling of Complex Event Processing: A Generic Algorithm and its Application to a Manufacturing Line*. In: 10th IEEE International Conference on Industrial Informatics (INDIN) (2012), pp. 380-385.

[13] Yongheng Wang, Xiaoming Zhang, *Complex Event Processing over Distributed Probabilistic Event Streams*. 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, pp.1489-1493, May 2012.

[14] H. Kawashima, H. Kitagawa and X. Li, *Complex Event Processing over Uncertain Data Streams*. Proceedings of the fifth international conference on P2P, parallel, grid, cloud and internet computing. pp.521-526, Nov 2010.

[15] Yanlei Diao, Neil Immerman, Daniel Gyllstorm, *SASE+: An Agile Language for Kleene Closure over Event Streams*. UMass Technical Report, 2007.

[16] Wu E., Diao Y., Rizvi S. *High-performance complex event processing over streams*. Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp.407-418. New York: USA. 2006.

[17] W. Hu, W. Ye, Y. Huang and S. Zhang, *Complex event processing in rfid middleware: a three layer perspective*. Proceedings of the 3rd international conference on in convergence and hybrid information technology, vol 1, pp.1121-1125, 2008.

[18] H. Liu, S. Goto and J. Li, *The study and application of tree-based rfid complex event detection algorithm*. Proceedings of the 2nd international symposium on electronic commerce and security, pp.520-524, 2009.

[19] Diao Y., Altinel M., Zhang H., Franklin M.J., and Fischer P.M. *Path sharing and predicate evaluation for high-performance XML filtering*. TODS, 28(4), pp.467-516, Dec 2003.

[20] Y.H. Wang, K. Cao, X.M. Zhang. *Complex event processing over distributed probabilistic event streams*. Computer and Mathematics with Applications 66(2013), pp 1808-1821.