Complex Event Processing Over Uncertain Data

Segev Wasserkrug
IBM Research
IBM Haifa Research Lab
Haifa University, Haifa, Israel
972-4-8296337
segevw@il.ibm.com

Avigdor Gal
Technion
Technion City
Haifa, 31200, Israel
972-4-8294425
avigal@ie.technion.ac.il

Opher Etzion
IBM Research
IBM Haifa Research Lab
Haifa University, Haifa, Israel
972-4-8296230
opher@il.ibm.com

Yulia Turchin
Technion
Technion City
Haifa, 31200, Israel
972-54-4514310
yulia.turchin@gmail.com

ABSTRACT
In recent years, there has been a growing need for active systems that can react automatically to events. Some events are generated externally and deliver data across distributed systems, while others are materialized by the active system itself. Event materialization is hampered by uncertainty that may be attributed to unreliable data sources and networks, or the inability to determine with certainty whether an event has actually occurred. Two main obstacles exist when designing a solution to the problem of event materialization with uncertainty. First, event materialization should be performed efficiently, at times under a heavy load of incoming events from various sources. The second challenge involves the generation of a correct probability space, given uncertain events. We present a solution to both problems by introducing an efficient mechanism for event materialization under uncertainty. A model for representing materialized events is presented and two algorithms for correctly specifying the probability space of an event history are given. The first provides an accurate, albeit expensive method based on the construction of a Bayesian network. The second is a Monte Carlo sampling algorithm that heuristically assesses materialized event probabilities. We experimented with both the Bayesian network and the sampling algorithms, showing the latter to be scalable under an increasing rate of explicit event delivery and an increasing number of uncertain rules (while the former is not). Finally, our sampling algorithm accurately and efficiently estimates the probability space.

Keywords
Complex Event Processing; Event Uncertainty Management

1. INTRODUCTION
In recent years, there has been a growing need for active systems that react automatically to events. The earliest active systems were in the database realm [25], impacting both industry (triggers) and academia (view materialization). New applications in areas such as Business Activity Monitoring (BAM), Business Process Management (BPM) [4], sensor networks [7], security applications (e.g., bio hazards, computer security, and prevention of DoS attacks), engineering applications (e.g., forecasting networked resources availability), and scientific applications (e.g., utilization of grid resources) all require sophisticated mechanisms to manage events and materialize new events from existing ones.

An event is a mechanism for delivering data to an active system. Some events are generated externally and deliver data across distributed systems, while other events and their related data are materialized by the active system itself, based on other events and a mechanism for predefined event pattern specifications (e.g., rules in active databases [9]). Event materialization has a clear trade-off between materializing events with certainty, using full and complete information, and the need to provide a quick notification of newly revealed events. Timely event materialization is therefore hampered by the gap between the actual occurrences of events, to which the system must respond, and the ability of active systems to accurately generate events. This gap results in uncertainty and may be attributed to unreliable data sources (e.g., an inaccurate sensor reading), an unreliable network (e.g., packet drop at routers), or the inability to determine with certainty whether a phenomenon has actually occurred (e.g., declaring an epidemic—see detailed example below).

A concrete example of this situation exists in syndromic surveillance systems, which are designed to detect outbreaks of epidemics or bio-terrorist attacks. Such systems use data from external data sources (e.g., transactional databases) and expert knowledge (see Shmueli and Fienberg [22]) to identify outbreaks and to quantify the outbreak attributes, such as the severity of an attack. Responding quickly to such attacks requires recognizing an attack has occurred—a difficult task as no direct indications of it exists. Therefore, hazard events must be materialized based on available data sources, which often provide insufficient information to determine hazard occurrence with certainty.

Shmueli and Fienberg [22] demonstrated that over-the-counter and pharmacy medication sales, calls to nurse hotlines, school absence records, Web hits of medical Web sites, and complaints of individuals entering hospital emergency departments can all serve as indicators of disease outbreak. These measures can be collected by querying local stores, physician databases, or data warehouses. Figure 1 (based on [22]) provides an illustration of the daily sales of over-the-counter cough medication from a large grocery chain in Pittsburgh, Pennsylvania. The data was gathered by intermittent querying of the chain's transactional database. The presentation of the data as a time series illustrates both the problem of uncertainty in event information and our methodology. Figure 1 displays the changes in sales during the months of November and December 1999, where a gradual, yet constant and significant increase of sales indicates the outbreak of the flu. Table 1 provides a simple example of the source of the data in Figure 1. The data points of Table 1 are marked as points A-F in Figure 1.

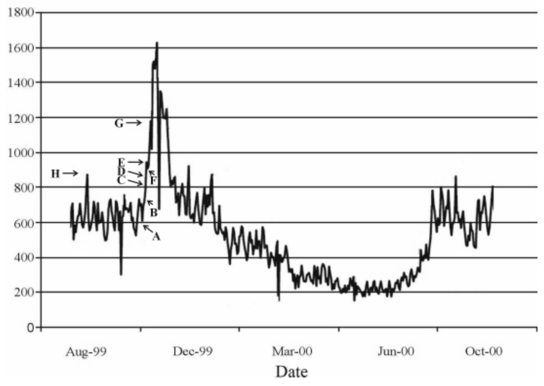

Figure 1: Over-the-counter cough medication sales

The problem can be stated simply: Given a set of events and a set of rules, materialize new events, their associated data, and their level of certainty (even if it is less than 100% certainty) Therefore, events carrying over-the-counter cough medication sales data can serve in processing a rule that *assigns a 90% chance of a flu outbreak whenever there is an increase in over-the-counter cough medication sales for four sequential days to a total increase of 350* units. Together, the events and rules determine whether an event of a flu outbreak should be materialized.[1] From Table 1, however, it is not clear whether an event of flu outbreak should be materialized. First, note that the provided data is rounded to the nearest ten. For example, note the increase from November 30 (with 600 units sold) to December 4 (with 950 units sold). Although there seems to be an increase there of 350 units, rounding may also suggest a smaller increase of 341 units. This is considered *uncertainty at the source*. In addition, an increase of 350 units in sales suggests only a high probability of a flu outbreak; does that mean such an event should be materialized? We call this aspect of uncertainty *materialization uncertainty*. Finally, the impact of uncertainty at the source on the materialization uncertainty should be quantified. In particular, uncertainty at the source reduces the *materialization uncertainty*. We present a model for uncertainty in Section 3 and provide an efficient algorithmic solution for its computation in Section 4.

The rule we presented is a simplified version of rules expected to exist in the real world. For example, instead of just specifying a single certainty level of 90%, one would probably compute the probability of an outbreak as a function of the amount of increase (e.g., computing the probability to be min(90+(X-350)/10,100) where X is the minimum daily sales over a four-day count and should be at least 350). However, even such a simple setup of a single source and a single rule does not seem to have a straightforward solution. Real-life applications, however, need to process multiple rules with multiple data sources. To illustrate, consider a situation in which we are provided with another rule, aimed at identifying an anthrax attack (see Figure 2 for the illustration of this more complex example). This rule combines data from events that come from different data sources and gives a lower probability to an anthrax attack whenever a flu outbreak is recognized (perhaps 30%) and a higher probability (perhaps 80%) otherwise. Therefore, if on December 4 (see Table 1), a flu outbreak event is not materialized, a smaller increase of hospital emergency department visits with respiratory complaints is enough to materialize an anthrax attack event. Other examples of complex rules may involve correlating data among pharmacies or across regions.

Two main obstacles exist when designing a solution to the event materialization problem. First, **event materialization should be performed efficiently**, sometimes under a heavy load of incoming events from various sources. Event materialization should also scale for a large number of (possibly inter-related) rules and complex rules that involve several sources of evidence. In the type of applications we envision, waiting too long to respond (e.g., hours in the case of bio or network attacks and seconds in the case of RFID sensor data) may mean that mitigating actions have little effect once they are applied. In bio-surveillance applications, late response (see point G in Figure 1) may mean it is too late to treat the infected population or to react in other ways, such as stockpiling and dispensing vaccine and medication. This is especially true in the event of bio-terrorist attacks.

Table 1: Over-the-counter sales relation

EID	**Date**	**Daily Sales** (Rounded)
113001	Nov 30	600
120101	Dec 1	700
120201	Dec 2	800
120301	Dec 3	900
120401	Dec 4	950
120501	Dec 5	930

The second challenge involves the generation of a correct probability space, given uncertain events. Clearly, correct quantification of the probability of materialized events serves as an important tool for decision making. Consider once more an event of an anthrax attack. To determine whether an anthrax attack has actually occurred, one should determine whether a flu outbreak was detected. However, even if materialized, a flu outbreak event, has an associated probability of 90%. How does that propagate to set the probability of an anthrax attack? Event generation under uncertainty should therefore be accompanied with an appropriate mechanism for probability computation. Wrong probability computation may cause an event to materialize either too early or too late. We have discussed the costs associated with late materialization already, yet we need to keep in mind that early response may also have its consequences. Therefore, wrongly determining that an anthrax attack has occurred (e.g., point H in Figure 1) and carrying out measures such as stockpiling and closure of schools and workplaces could be economically costly.

[1] Rules for determining occurrences in the real world are typically determined by experts. Therefore, we can assume that all probabilities and trends given in this example are provided by bio-surveilence experts, supported by statisticians.

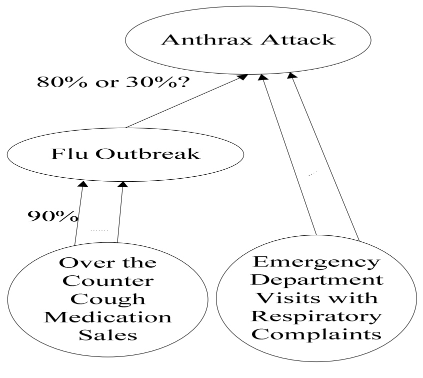
Anthrax Attack
80% or 30%?
Flu Outbreak
90%
Over the Counter Cough Medication Sales
Emergency Department Visits with Respiratory Complaints

Figure 2: Anthrax rule

Clearly, a brute force solution in which the arrival of a new event is evaluated against all possible rules does not scale. It may involve an exponential number of evaluations, depending on the amount of dependency among rules. To illustrate this point, let us consider a natural way of interpreting uncertain events using possible world semantics, where a possible world corresponds to a set of possible events (and the data they carry). A possible world is associated with a probability measure, stating the probability of it being the one true world. Clearly, an explicit representation of each of these possible worlds separately, for the sake of evaluating new event materialization, is practically infeasible. Therefore, our main goal is to provide an efficient (yet generic) mechanism for reasoning with uncertain events.

In this work, we present a generic framework for representing uncertain events and rules with uncertainty. We then present an algorithm to construct the probability space that captures the semantics and defines the probabilities of possible worlds using an abstraction based on a Bayesian network and discuss its complexity. To improve materialization efficiency, we employ a sampling technique over a set of rules, show that the approximations it provides truly represent the probabilities defined by the original probability space, and discuss the algorithm complexity. We validate our solution using a simulation environment, simulating external event generation, and materialize events using our proposed sampling algorithm.

To summarize, the paper makes the following contributions:

- We describe a simple yet powerful generic model for representing the materialization of new events under uncertainty. By doing this, we extend the current complex event processing literature on managing data under uncertainty to allow uncertain materialization of events and data.
- We develop an algorithm, based on Bayesian network representation, to materialize new events given the arrival of an uncertain event and to compute its probability. Bayesian network modeling is not a trivial task and we offer a fully automatic mechanism for its generation.
- We develop a sampling algorithm for efficient approximation of new event materialization. The algorithm enables a quick method to compute the probabilities of a set of events. Its unique contribution is sampling over the set of rules, rather than sampling from the Bayesian network (which is the common solution in the AI literature). In this way, we can produce an efficient solution without constructing a Bayesian network.
- We present experiments that demonstrate the scalability and accuracy of the sampling algorithm.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces our complex event model. Sections 4 and 5 describe our solution. Section 6 presents experimental results and Section 7 concludes.

2. RELATED WORK

Complex event processing is supported by systems from various domains. These include ODE [14] for active databases and the Situation Manager Rule Language [2], a general purpose event language. All but two existing languages, to the best of our knowledge, enable only deterministic materialization of events. Therefore, solutions adopted in the active database literature, such as the Rete network algorithm [13], [25] fail to provide an adequate solution to the problem, since they cannot estimate probabilities. Wasserkrug, Gal, and Etzion proposed an initial approach for managing uncertain rules [23]. Their work was followed by Balazinska, Khoussainova, and Suciu, who proposed a probabilistic event language for supporting RFID readings [3]. We extend the work in [23] by providing algorithms for materializing uncertain events and empirical evidence to the scalability of the approach. Our proposed model shares some commonalities with [3], namely uncertainty specification at both the event occurrence and the uncertainty level (as suggested in [23]), specifying uncertain rules, and providing an algorithm for uncertain event materialization. However, our framework does not limit the type of temporal expressions it can handle nor does it limit attribute derivation types.

Recent works have focused on event stream modeling, proposing formal languages to support situations such as event negations [12]. Various aspects of event-oriented computing were discussed at CIDR 2007 (e.g., [11], [15]). Our focus in this work is on modeling and efficiently managing uncertainty in complex event systems, which was not handled by these works. Modeling probabilistic data and events was suggested in [11], [21], [16], [24]. This work extends those models by proposing a model and a probability space representation of rule uncertainty.

A Bayesian network [20] is a method for graphically representing a probability space, which utilizes probabilistic independencies to enable a relatively sparse representation of the probability space. The network is, in most applications, manually constructed for the problem at hand. We propose an algorithm to automatically construct a Bayesian network from a set of events and rules, following the KBMC paradigm [5].

Existing works [8], [18], [6] for processing complex events in specific security domains tailor probabilistic models or direct statistical models (e.g., regression) to the application. No general framework was defined there to materialize uncertain events, a gap we strive to fill in this work. An additional shortcoming of application-specific tailored models is their inability to combine the results of different models. We propose to overcome this shortcoming using a general framework that supports such a feature.

3. PROBABILISTIC EVENT MODEL

3.1 Event Model

An *event* is an actual occurrence or happening that is **significant** (falls within a domain of interest to the system), **instantaneous** (takes place at a specific point in time), and **atomic** (it either occurs or not). This definition, while limited, suits our specific needs. Examples of events include termination of workflow activity, report on daily sales, and a person entering or leaving a certain geographical area. We differentiate between two types of events. *Explicit events* are signaled by event sources, external to the active system. *Materialized events* are events that are materialized based on other events (see Section 3.2).

Data can be associated with the occurrence of an event. Some data are common to all events (e.g., occurrence time), while others are specific only to some events (e.g., amount of sales). The data items associated with an event are termed *attributes*.

Similar to other works in this area [10], [3], we record an event (either explicit or materialized) using an Event Instance Data (EID) relation, which captures the information of the system. For example, the structure of an EID for daily sales (see Table 1) is *DS(EID,Date,dailySales)*, representing the date and the daily sales of each event. Events that share the same set of attributes are of the same type, e.g., the *DS* event type in the example above.

Events that have no uncertainty associated with them can be represented by a single tuple of values $\langle val_1,\ldots,val_n\rangle$ (one value for each type of data associated with the event). When an event is uncertain, it is represented by several tuples, where each such tuple has an associated probability. The probability associated with a value of the form $\langle val_1,\ldots,val_n\rangle$ is the probability that this event has occurred, and the value of its attributes is $\langle val_1,\ldots,val_n\rangle$. The complementary probability is the probability that the event has not occurred. In this work, we use the closed world assumption with regards to event occurrence. Therefore, an event has not occurred unless it is explicitly captured as an EID (either explicit or materialized).

An *event history* $eh_{t_1}^{t_2}$ is the set of all events (of interest to the system), as well as their associated data, whose occurrence time falls between t_1 and t_2. Uncertainty regarding individual events leads to uncertainty regarding the event history.

3.2 Materialization Model

Our materialization model is based on rules, which together with information regarding explicit events (Section 3.1) define the probability space of interest (Section 3.3) at each time point.

Our rule definition extends deterministic rule definitions (*e.g.*, [2]) to handle uncertainty. For ease of exposition, we refrain from presenting complete rule language syntax. Rather, we focus on the elements that serve in defining the mechanism for generating events under uncertainty (see Section 4).

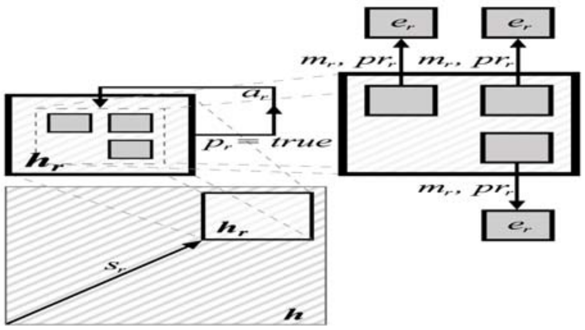

Figure 3: Rule components and roles

A rule $r = \langle s_r, p_r, a_r, m_r, pr_r\rangle$ is a quintuple, defining the necessary conditions for the materialization of new events. It can be implemented as a set of database queries using SQL statements. Figure 3 illustrates the rule components and roles. We detail next each of the rule elements.

s_r is a *selection* function that filters events, relevant to materialization, according to rule *r*. s_r receives an event history *h* as input, and returns h_r such that $h_r \subseteq h$. It is the role of the selection expression to filter out events that are not relevant for inference according to *r*. We say that event $e \in h$ is relevant (or *selectable*) according to rule *r* iff $e \in s_r(h)$.

p_r is a predicate, defined over a filtered event history, determining when events become candidates for materialization. Formally, p_r is a Boolean function $p_r : s_r(h) \to \{true, false\}$.

The third function a_r is an *association function*, whose role is to define how many events should be materialized, as well as which subsets of selectable events are associated with each materialized event. $a_r : s_r(h) \to 2^{s_r(h)}$ is a function from a set of selectable events to its power set. It returns subsets of events, such that each subset is associated with a newly materialized event.

m_r is a *mapping function*, determining the attribute values of the materialized event. Formally, $m_r : A \to e \text{ s.t. } e.c = c_r \text{ and } A \in a_r(s_r(h))$. For each set of events in $a_r(s_r(h))$, m_r determines a set of attribute values of a materialized event of type c_r. For ease of exposition, we assume that a rule materializes a single event type.

The final function pr_r is a *probability function* that determines the probability of the materialized events. pr_r receives as input a set of events in $a_r(s_r(h))$ and the output of m_r on these events, and assigns it a real value that is interpreted as a probability. This function's signature is $p_r : (A, m_r(A)) \to [0,1] \text{ s.t. } A \in a_r(s_r(h))$.

A rule defines how many new events (or EIDs) should be materialized and helps calculate their attributes and probabilities. Intuitively, assume that the set of possible event histories prior to evaluating rule *r* is *H*. Moreover, assume that the actual event history is known to be some $h \in H$. Given that this is the actual event history, four conditions should be satisfied. First, newly materialized events should be considered possible according to rule *r* iff the event pattern of *r* evaluates to *true* on this event history, i.e., iff $p_r(s_r(h)) = true$. Second, if $p_r(s_r(h)) = true$, then the number of materialized events should be equal to $|a_r(s_r(h))|$. Third, for each materialized event, given that it has in fact occurred, the values of the set of events belonging to $a_r(s_r(h))$ determine its values through the mapping expressions, i.e., the values of its attributes should be $m_r(a_r(s_r(h)))$. Fourth, for each materialized event, given that the pattern p_r has evaluated to *true*, its probability should be determined by the probability assignment component pr_{re}.

To conclude this section, note that the components s_r, p_r, a_r and m_r are fully deterministic. Therefore, the quadruple $\langle s_r, p_r, a_r, m_r \rangle$ could be viewed as a general description of **deterministic** composite event rule. In addition, any deterministic rule can be stated in our framework using pr_r functions with a single output of 1.

3.3 Probability Space Definition

The main novelty of our model is in the support of the calculation of probabilities associated with materialized events, at a given time point *t*. At *t,* the set of possible materialized events is determined by the EIDs known at *t* (together with the defined rules). Therefore, since different sets of EIDs are available at different time points, a (possibly) different probability space needs to be defined for each time point separately.

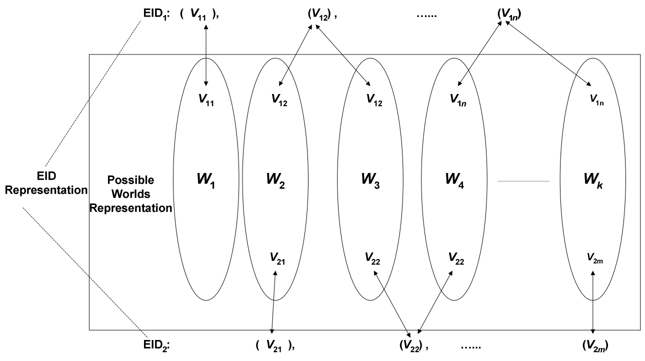

Figure 4: Two probability space representations

Figure 4 provides an illustration of two equivalent representations of the probability space. An intuitively appealing way to define this probability space involves possible world semantics (see [16]). The probability space at time *t* is a triple $P_t = (W_t, F_t, \mu_t)$, where W_t is a set of possible worlds (marked as ovals in Figure 4), with each possible world corresponding to a specific possible event history at *t*, $F_t \subseteq 2^{|W_t|}$ is a σ-algebra over W_t and $\mu_t : F_t \to [0,1]$ is a probability measure over F_t. We assume that each possible world corresponds to an event history that is finite in size and that the real world is one of the possible worlds.

A less intuitive, yet more efficient, representation is based on the observation that each finite event history can be represented by a finite number of EIDs. Therefore, a finite number of possible worlds, where each world is a finite event history, **consist of a finite number of EIDs.** Each possible world W_t, which consists of *m* events (two in Figure 4; top and bottom), is assigned with *m*≤*n* tuples, where a tuple represents the values of an EID in this world. Whenever *m*<*n* in a possible world, the remaining *n-m* events do not occur in W_t. Recall that W_t is finite and therefore each E_i can only have a finite number of associated values (one for each world in W_t in which it appears). Therefore, the probability space P_t can be represented by a set of *n* EIDs - $E_1, \ldots E_n$ by defining P_t as $\left(\Omega_t, F_t, \mu_t'\right)$ where $\Omega_t = \left\{\left\langle Val_1, \ldots, Val_n \right\rangle\right\}$. $\left\langle Val_1, \ldots, Val_n \right\rangle$ is a tuple corresponding to an event history, where this event history is a possible world $w_t \in W_t$ as described above ($|\Omega_t|$ is finite), $F_t = 2^{\Omega_t}$, and $\mu_t'(\left\{\left\langle Val_1, \ldots Val_n \right\rangle\right\}) = \mu_t(w_t)$ such that w_t is the world represented by $\left\{\left\langle Val_1, \ldots, Val_n \right\rangle\right\}$. The non-occurrence of event E_n is (implicitly) represented as $\{notOccurred\}$.

We next show how rules, together with EIDs specify the probability space P_t at each time point *t*.

4. PROABILISTIC EVENT MATERIALIZATION

We now describe an algorithm for uncertain materialization of events. In a nutshell, the proposed algorithm works as follows: Given a set of rules and a set of EIDs at time *t*, we **automatically** construct a Bayesian network that correctly represents the probability space at *t*. Probabilities of new events are then computed using standard Bayesian network methods. The Bayesian network construction algorithm iteratively selects a rule and creates a Bayesian network **segment** that corresponds to this rule.

4.1 Rule Triggering Order and Eligibility

Rule triggering order impacts both *determinism* and *termination*. The former requires that, given an event history containing only explicit EIDs at the start of the materialization process, the process always results in the same probability space. The latter requires that the materialization algorithm terminates.

To ensure determinism and termination we enforce rule ordering such that 1) rules are triggered according to the defined order, 2) rule ordering is independent of event histories, and 3) an event may trigger a rule at most once. The last requirement ensures termination. The second requirement ensures determinism.

We define an order using an event triggering graph where nodes represent event **types**. An edge between two nodes *u* and *v* exists iff the event type *u* may serve as input to the materialization of events of the type *v* using rule *r*. In this graph, we say that an event class *et* corresponds to node $v \in V$ iff either *v* is a node corresponding to the explicit event class *et* or *v* corresponds to *r* and *et* is the class of events inferred by r. Definition 1 defines a *triggering graph* as a step towards rule ordering specification.

***Definition* 1** *Given a set of explicit EID*s *ET , materialized EID*s *MT, and a set of rules R, a* Static Triggering Graph *over ET and R,* TG_{ET}^{R} = *(V,E) is a directed graph such that V=ET* ∪ *MT. An* *edge* $(u,v) \in E$ *exists in* TG_{ET}^{R} *iff there exists a rule r such that u is in* s_r *and v is in* a_r .

The entire graph takes into account all rules. Figure 2, discussed in the introduction, is an example of a static triggering graph. For specific event languages, even complex ones, it is possible to construct TG_{EC}^{R} by automatic syntactical analysis of event types and rule definitions [1].

We now define the rule triggering order using a *quasi-topological ordering* [1] on the event triggering graph. For completeness sake, we provide the definition of a quasi-toplogical order next. We say that a vertex *v* is *reachable* from a set of vertices V_S if there is a vertex $u \in V_S$ such that there is a path from *u* to *v*. By definition, we consider a vertex to be reachable from itself.

***Definition* 2** *Let* $G=(V,E)$ *be a directed graph, let* $S \subseteq V$*, and let* V_S *be the subset of V such that* $v \in V_S$ *if v is reachable from S. A* Quasi Topological Order *on G induced by S is a linear ordering of all vertices* V_S *such that if G contains an edge* (u,v)*, then either u appears before v in the ordering* (i.e., $f(u)<f(v)$) *or there exists a path in G,* $v_0, v_1, \ldots, v_k$ *such that* $v_0=w$*,* $v_k=u$*,* $w \in S_V$*, v appears in the path, and for each pair of vertices in the path,* v_i*,* v_j*, if* v_i *appears before* v_j *in the path then* v_i *appears before* v_j *in the ordering* (i.e., $f(v_i) < f(v_j)$).

A quasi-topological order generalizes topological orders for arbitrary directed graph that may include cycles. It is easy to show that a quasi-topological order on a DAG is a topological ordering. In Figure 2, the quasi-topological order is also a topological order, in which the rule for detecting flu epidemic precedes the rule of detecting an anthrax attack. An efficient algorithm for finding a quasi-topological ordering appears in [1].

Given a triggering graph and a quasi-topological order, we define the triggering order by constructing a triggering graph of the rules and event types of the application and finding a quasi-topological order on the rules. Rules would then be triggered according to the quasi-topological order.

Theorem 1: *Triggering each rule at most once in response to an event, in the order defined by the quasi-topological order on the triggering graph ensures determinism and termination.* [2]

4.2 Bayesian Network Construction Algorithm

In this section, we present an algorithm that constructs a Bayesian network that represents the probability space of a set of explicit events and rules. Recall that one of the two main roles of a rule is to define how many new events (or EIDs) should be inferred, and to enable calculating their attributes and probabilities. This role is defined formally as follows: Let $\mathbf{E}_r$ be the set of EIDs inferred by rule *r* on a set of possible event histories *H*. Then for every $h \in H$ such that $p_r(s_r(h)) = true$, if $a_r(s_r(h)) = \{A_1, \ldots, A_m\}$, there must be *m* inferred EIDs $E_1, \ldots, E_m \in \mathbf{E}_r$ such that:

$$\Pr(E_i = \{occurred, m_r(A_i)\} \mid H = h) = pr_r(A_i, m_r(A_i)) \qquad (1)$$

and

$$\Pr(E_i = \{notOccurred\} \mid H = h) = 1 - pr_r(A_i, m_r(A_i)) \qquad (2)$$

where $H = h$ is shorthand for $E_1 = e_1 \wedge E_2 = e_2 \wedge \cdots \wedge E_n = e_n$, where $E_1, \ldots, E_n$ are the EIDs that represent the set of possible histories *H*, and $e_1, \ldots, e_n$ are the events defining the event history *h*. Moreover, if the event history $h \in H$ is such that $p_r(s_r(h)) = false$, then for every $E \in \mathbf{E}_r$ it must hold that

$$\Pr(E = \{notOccurred\} \mid H = h) = 1 \qquad (3)$$

Therefore, the construction algorithm creates a Bayesian network representing a probability space in which Equations (1) - (3) hold.

For the sake of completeness, we provide here a brief description of a Bayesian network. As discussed in Section 2, a Bayesian network is the most widespread paradigm for the representation and efficient inference of a probability space defined over a set of random variables. A Bayesian network has two main components: qualitative and quantitative. The qualitative component of the network is a graph, $G_{BN} = (V_{BN}, E_{BN})$, where V_{BN} is a set of vertices which describe the random variables and E_{BN} is a set of edges that describe the direct probabilistic dependencies between the random variables. Unlike the static triggering graph that represents event **types** and their relationships, this graph represents event **instances**. The quantitative component is a *Conditional Probability Table* (*CPT*), which defines the probabilities of each random variable based on the probabilistic dependencies. For example, Figure 5 contains a depiction of the qualitative (graph) part of a Bayesian network which defines a probability space over five random variables: S_1, S_2, FO, EDV, and AA, depicted as vertices. The edges indicate direct probabilistic dependencies. This is an example of a Bayesian network that may be generated by our algorithm for the Anthrax rule in Figure 2, where S_1 and S_2 correspond to over-the-counter cough medication sales events, FO corresponds to a flu outbreak event, EDV corresponds to an event describing a high volume of emergency department visits with respiratory complaints, and AA corresponds to an anthrax attack event. Table 2 contains the quantitative relationship between AA to FO and EDV. For each possible value of FO and EDV the table provides the conditional probability of each value of AA. For example, $\Pr(AA = true \mid FO = false, EDV = true) = 0.8$. Such a table exists for each vertex in the Bayesian network. For vertices which have no incoming edges, the table contains the prior probabilities.

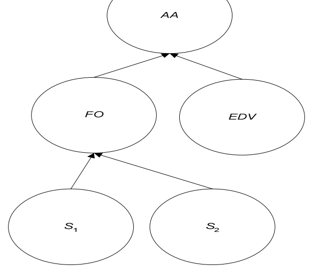

Figure 5: Bayesian network graph example

[2] All proofs are omitted in this paper due to lack of space.

As noted in Section 3.1, each EID may be assigned various values, each with a different probability. Therefore, we can view EIDs in our model as random variables. A Bayesian network is well suited to describe probability spaces based on statements such as equations (1)-(3). In addition, Bayesian networks utilize probabilistic independencies to make such a representation more efficient. In our setting, such independencies exist due to the selection function, which determines which EIDs are relevant for the materialization of other events. We use the term *selectable EIDs* to refer to such EIDs. We refrain from presenting selectable EIDs formally due to a lack of space. Also, an exact definition of the probabilistic independencies induced by s_r is beyond the scope of this work. Informally, any EID E_r, materialized by rule *r*, is conditionally probabilistically **independent** of all **non-selectable** EIDs that exist at the point of materialization, given that known values of the EIDs are **selectable** by rule *r*.

Algorithm 1 constructs a Bayesian network from explicit events and rule definitions that describes the desired probability space. The algorithm is triggered with a new event arrival, traverses the rules according to their quasi-topological ordering, and generates conditional probabilities for simple and materialized events. For each rule *r*, each iteration of the while loop (lines 8-39) creates a Bayesian network **segment** (i.e., a segment with relevant nodes, edges, and CPTs) for a specific rule, satisfying equations (1)-(3), and utilizing probabilistic independencies. After carrying out these steps for all rules, the Bayesian network is complete and the P_t at *t* are calculated based on the Bayesian network (line 40, using standard Bayesian network inference algorithms).

Algorithm 1: Bayesian Network Construction Algorithm, triggered by a new event arrival

```
1.  $H \leftarrow getExplicitEvents$
2.  $BN \leftarrow \varnothing$
3.  for all $E \in H$
4.    $V_{BN} \leftarrow V_{BN} \cup E$
5.    $addCPT(E, E.eventProbabilitySpace)$
6.  end for
7.  $order \leftarrow obtainQuasiTopologicalOrder$
8.  while $order \neq \varnothing$
9.    $r \leftarrow deleteNextRule(order)$
10.   $H_r \leftarrow calculateSelectableEIDs(H, r)$
11.   $\mathbf{E}_r \leftarrow \varnothing$
12.   $unAssignedHistories \leftarrow \varnothing$
13.   $assignedHistories \leftarrow \varnothing$
14.   for all $h_r \in H_r$
15.     if $p_r(s_r(h_r)) = true$
16.       // Calculate the associated tuples for each inferred EID
17.       $assignedHistories \leftarrow assignedHistories \cup h_r$
18.       $assocTuples_r \leftarrow a_r(s_r(h_r))$
19.       for all $tuple \in assocTuples_{re}$
20.         $E_r \leftarrow createNewEID()$
21.         $assignEventHistory(E_r, h_r)$
22.         $V_{BN} \leftarrow V_{BN} \cup E_r$
23.         for all $E \in H_r$
24.           $E \leftarrow E \cup (E, E_r)$
25.         end for
26.         $\mathbf{E}_r \leftarrow \mathbf{E}_r \cup \{E_r\}$
27.         $\{s_1, \ldots, s_n\} \leftarrow m_r(tuple)$
28.         $prob \leftarrow pr_r(tuple, \{s_1, \ldots, s_n\})$
29.         $addCPT(E_r, H_r = h_r, \{occurred, s_1, \ldots, s_n\}, prob)$
30.         $addCPT(E_r, h_r, \{notOccurred\}, 1 - prob)$
31.         $H \leftarrow H \cup E_r$
32.       end for
33.     end if
34.     if $h_r \notin assignedHistories$
35.       $unassignedHistories \leftarrow unassignedHistories \cup h_r$
36.     end if
37.   end for
38.   $completeCPTs(\mathbf{E}_r, h_r)$
39. end while
40. $calculateProbabilitySpace(BN)$
```

Table 2: CPT example

	AA=true	*AA=false*
FO=true, *EDV=true*	0.3	0.7
FO=false, *EDV=true*	0.8	0.2
FO=true, *EDV=false*	0.05	0.95
FO=false, *EDV=false*	0.1	0.9

addCPT in line 5 defines the CPTs of explicit EIDs. For each rule, the set of selectable EIDs is returned by the function *calculateSelectableEID,* the specification of which is omitted due to lack of space. From probabilistic independencies, the materialization of EIDs according to rule *r* uses only EIDs that are selectable by *r* and exist prior to its materialization (lines 14-38).

Lines 15-33 ensure that equations (1) and (2) hold. Every event history h_r for which it is possible that a materialized event exists is associated with this EID, and placed in *assignedEventHistories*. After line 37, each materialized EID is associated with the event history that inferred it. However, the conditional probability table for each inferred EID is only *partially* specified, since the conditional probabilities are specified for the materialized EID E_r only for the case in which the values of the EIDs in the selectable set of event history H_r represent the event history assigned to E_r. Therefore, the next step is to complete the CPT entries for each inferred EID for all of the possible event histories in H_r. This is carried out by the function *completeCPTs*. This function ensures that Equation (3) holds, by associating zero probability with each event history that is not associated with this EID.

There are three important comments to make regarding Algorithm 1. First, materialization according to the rules must be carried out continuously in near-real time with every new event. This continuous process requires that the Bayesian network is dynamically updated to ensure that the constructed network reflects, at each time point *t*, the probability space P_t. Moreover, information may be non-monotonic. For example, new information regarding an event occurrence may cause previously selectable EIDs to become non-selectable, or may lower the probability of a materialized event. To support constant, and possibly non-monotonic, updates of the network, the network needs to be constructed from scratch with each new explicit event. Secondly, we assume that the explicit EIDs are probabilistically independent. While this assumption may seem to be restrictive, it can be easily overcome by using a set of rules R_e to define the probabilistic dependencies between the explicit EIDs. These rules result in a set of dependent materialized EIDs. With this new set of rules, other rules can be defined based solely on the dependent materialized EIDs. Finally, the algorithm is equipped to handle rules in which the probability is computed online, when generating the new event. Since the construction process is dynamic, the probability of the new event is injected into the network once the event is created and propagated upwards dynamically.

4.3 Correctness and Complexity

Let *r* be a rule and let *H* be the set of event histories prior to the activation of *r*. We need to show that the probability space defined by the Bayesian network as constructed by Algorithm **1** fulfills the semantics of the rules as defined in Section 3.2: 1) newly materialized events should be considered possible according to rule *r* iff the event pattern of *r* evaluates to *true* on this event history (*i.e.*, $p_r(s_r(h)) = true$); 2) if $p_r(s_r(h)) = true$, the number of materialized events should be $|a_r(s_r(h))|$; 3) for each materialized event, given that it has in fact occurred, the values of the set of events belonging to $a_r(s_r(h))$ determine its values through the mapping expressions, i.e., the values of its attributes should be $m_r(a_r(s_r(h)))$; 4) for each materialized event, given that the pattern p_r has evaluated to *true*, its probability should be determined by the probability assignment component pr_{re}; and 5) the probability space should contain the probabilistic independencies induced by the selection function.

The correctness of Algorithm 1 is summarized in Theorem 2.

***Theorem* 2**: *The probability space represented by the Bayesian network constructed by Algorithm* 1 *fulfills conditions 1-5.*

The complexity of Algorithm **1** is given in terms of the following:

- $n_0 = |H_0|$ and $n = |H|$, the number of explicit events and EIDs in the set of histories *H* when the algorithm terminates.
- $m = \max_{E \in H} |E|$, the largest size of the state space of an event in *H*. Note that the number of possible worlds is $O(m^n)$.
- *m*': the largest number of states of a single EID returned by *calculateSelectableEIDs*.
- *rs*=|*R*| and n': The number of rules in rule set R and the maximum number of EIDs selectable by a rule in *R*.
- *i*: The maximum number of events materialized by any rule for any given event history.
- $|s(n)|$, $|cs(n)|$, $|p(n)|$, $|a(n)|$, $|m(n)|$, $|pr(n)|$: The maximum time complexity of the respective functions s_r, p_r, a_r, m_r and pr_r for any rule *r* when given an event history of size *n* as input. We assume that this complexity can be bounded by the number of events in the event history.
- $|es|$: The maximum complexity of function es_r for any rule *r*.
- *addCPT*: the complexity of adding an entry to the conditional probability table in BN.

Based on the above, the complexity of Algorithm 1 is described below by Theorem 3.

***Theorem* 3**: *The overall complexity of Algorithm* 1 *is* $O(rs \cdot m'^{n'}\left[|s(n)'| + |p(n')| + |a(n')| + i \cdot \left[|m(n')| + |pr(n')| + addCPT\right]\right]$ $+ \; mn|es_r| + n'm^{n'}|cs_r(n')|)$.

The algorithm is exponential in the number of selectable events n' and polynomial in the number of rules and possible worlds (which is m^n). It is worth noting that while $n' \le n$, we expect most of the events to be filtered out using s_r. Therefore, we typically have that $n' << n$. In addition, while the maximum number of events materialized for any rule is 2^h, for event history *h,* rules typically aggregate information or correlate events, and therefore it is expected that in many cases, *i*<<|*h*|. Finally, if we denote by i_r the maximum number of events inferred by rule *r*, then we must have that $\sum_{r \in R} i_r \le n$.

In Algorithm 1, the Bayesian network is constructed from scratch with each new event, due to the non-monotonic reasoning of the rules. An alternative approach, and one which may be more efficient, is to dynamically update the Bayesian network. While such an algorithm is a part of our framework, we do not detail it here due to lack of space.

5. SAMPLING ALGORITHM

Algorithm 2: RuleSamp, triggered by a new event arrival

```
1.  $h \leftarrow \varnothing$
2.  for all $E \in H_0$
3.      $e \leftarrow probSampling(E)$
4.      $h \leftarrow h \cup e$
5.  end for
6.  $order \leftarrow obtainQuasiTopologicalOrder$
7.  while $order \ne \varnothing$
8.      $r \leftarrow deleteNextRule(order)$
9.      $h' \leftarrow \varnothing$
10.     $selEvents \leftarrow s_r(eh)$
11.     if $p_r(selEvents) = true$
12.         $assocTuples_{re} \leftarrow a_r(selEvents)$
13.         for all $tuple \in assocTuples$
14.             $\{s_1, \ldots, s_n\} \leftarrow m_r(tuple)$
15.             $prob \leftarrow pr_r(tuple, m_r(tuple))$
```

```
16.         $probSamp \leftarrow sampleBernoulli(prob)$
17.         if $probSamp$=1
18.            $e \leftarrow \{occurred, s_1, \ldots, s_n\}$
19.         else
20.            $e \leftarrow \{notOccurred\}$
21.         end if
22.         $h' \leftarrow h' \cup \{e\}$
23.      end for
24.   end if
25.   $h \leftarrow h \cup h'$
26. end while
27. return $h$
```

Algorithm 1 generates a Bayesian network from which the exact probability of each event can be computed. Given a Bayesian network, it is efficiently possible to approximate the probability of an event occurrence using a sampling algorithm. Given a Bayesian network with nodes E_1, …, E_n, to calculate an approximation for the probability that $E_n = \{occurred\}$ one would proceed by first generating *m* independent samples using a Bayesian network sampling algorithm. Then, $\Pr(E_n = \{occurred\})$ is approximated by $\frac{\#(E_n = \{occurred\})}{m}$, where $\#(E_n = \{occurred\})$ is the number of samples in which E_n has received the value *occurred*. $\frac{\#(E_n = \{occurred\})}{m} \xrightarrow[m\to\infty]{} \Pr(E_n = \{occurred\})$ can be shown to hold and given a sample of size *m*, a confidence interval for the approximation can be compute. Given desired precision in terms of a confidence α and an interval size β, one would need $\left(\frac{2Z_{\alpha/2}}{\beta}\hat{\sigma}\right)^2$ samples, where $Z_{\alpha/2}$ is the $\alpha/2$ quantile of the normal distribution, and $\hat{\sigma}$ is the sample standard deviation.

The challenge in using this method to obtain an efficient sampling algorithm is that the generation of the Bayesian network is exponential in the number of events. Therefore, the goal is **to efficiently obtain a sample from the Bayesian network without actually constructing it**. It turns out that this is possible by sampling on the rules themselves, as presented in Algorithm 2. The main principles underlying the algorithm operation are the following: First, a sample for the explicit events is generated using the mutual independence assumption (lines 2-5). Then, the algorithm traverses the rules in the quasi-topological order and materializes events according to each rule, given the sampled event history. In addition, the materialization according to each rule is based on the probabilistic rule definitions. Note that the Bayesian network does not need to be built as part of Algorithm 2.

Let $E_1, \ldots, E_m$ be the random variables (EIDs) in a Bayesian network *BN*. The correctness of **RuleSamp** is determined below:

***Theorem* 3**: *Let $E_1 = e_1, \ldots, E_m = e_m$ be a set of values with non zero probability. Then the probability that the set of values $E_1 = e_1, \ldots, E_m = e_m$ will be returned by an algorithm which samples on the Bayesian network is identical to the probability that the event history $\{e_1, \ldots, e_m\}$ will be returned by Algorithm 2.*

A detailed proof of Theorem 3 is beyond the scope of this article. However, the principles underlying this proof are the following: Both the sampling algorithm and the Bayesian network construction are derived from the formal probabilistic definition of the rules in equations (1)-(3). Therefore, the sampling algorithm samples in a manner consistent with these equations, and (by Theorem 2) the probability space represented by the Bayesian network constructed by Algorithm 1 is also consistent with those equations. In addition, both the sampling and the Bayesian network construction are carried out according to the quasi–topological order defined on the rules.

6. EMPIRICAL STUDY

We experimented with both the Bayesian network and the sampling algorithms. We conducted an extensive empirical study, showing that **RuleSamp** accurately and efficiently estimates the probability space. It displays nice scalability characteristics as the number of possible worlds increases. The number of samples required for all tested precisions grows **sub-linearly** in the number of possible worlds. In addition, our performance results are of the same order of magnitude as the deterministic event system analyzed in [2]. Although it is difficult to directly compare the results between the two works, this result is encouraging since in our setting, repeated deterministic inference is carried out to enable the sampling. Finally, we observe that the percentage of relevant events and the desired precision are significant performance factors.

6.1 Datasets, Experiment Setup and Evaluation

In our experiments we varied control parameters that impact performance the most. These parameters include the number of relevant events, the number of possible worlds, the approximation precision, and the rule probabilities. A summary of the variable settings for the RuleSamp algorithm appears in Table 3. All of the experiments were based on a rule set with two types of rules: Level 1 rules materialize events based only on explicit events, and level 2 rules materialize events based on events that were materialized by level 1 rules. In all experiments, for **each** explicit event, samples for the materialized events were generated by the RuleSamp algorithm described in Section 5 until the probability of each event was approximated within the desired precision. In all experiments, 20,000 explicit events were generated, and the total time to process these explicit events was measured. The performance measure used in all experiments was the *event processing rate per second,* calculated as $\frac{\text{number of genereted events}}{\text{total processing time}}$. In addition, the results presented were averaged over five runs, to take into account possible variations in individual runs. The number of possible worlds was controlled by changing the number of rules. In our setting, each rule corresponded to a single materialized event. Therefore, the number of possible worlds was always calculated as 2^{rs}, where *rs* is the number of rules in the scenario. In all experiments, the confidence (α) was set to 0.05.

Table 3: Experiments and control parameter values

Experiment	Events %	Precision (β)	Possible worlds	Rule Prob.
Sensitivity to Relevant Events	0, 1, 80	0.05, 0.1, 0.15, 0.2	64	Function of event attributes
Sensitivity to Possible Worlds	20, 80	0.05, 0.1, 0.15, 0.2	16 – 2^36	Function of event attributes
Sensitivity to precision	20, 80	0.05, 0.1, 0.15, 0.2	2^18 2^28, 2^36	Function of event attributes
Sensitivity to event prob.	20,80	0.05, 0.1, 0.15, 0.2	64	0.1, 0.5, 0.9

The functions s_r , p_r , a_r , m_r and pr_r were implemented using custom code. The simulation environment and the RuleSamp Algorithm were implemented in Java version 1.6.03, using JDK version 5.0, on the Windows XP operating system, and were carried out on a single computer with 1GB of memory and an AMD Athlon 64 3200 CPU.

Explicit events were generated using a Bernoulli distribution. We varied the number of level 1 rules from 2 to 36. Each rule uses between 2 to 15 explicit events. Level 2 rules materialize events based on events that were materialized by level 1 rules. Each level 2 rule uses between 2 to 34 level 1 events and no explicit events.

6.2 Scalability: Event Relevance Analysis

In this experiment, we tested the sensitivity of our framework to the percentage of explicit events relevant for materialization. This experiment was modeled after the performance experiments carried out in [2], where the performance of a deterministic event materialization system was tested. We wish to stress here that an exact comparison between our results and the results obtained in [2] cannot be made. The main reason for this is that in [2], the deterministic event materialization rules were implemented in a general purpose language intended for such materialization, while in our experiments, the deterministic parts of the rule were implemented using custom Java code. In addition, the experiments in [2] were carried out three years ago on a weaker hardware platform. Therefore, we use this set of experiments as a feasibility study of our approach.

In order to carry out these tests, four scenarios were defined. The first is *Standby* scenario, in which all generated events were irrelevant to event materialization. This gives an upper bound on the event rate that can be handled by the system. Note that the processing time of the system is due to the need to both read in the event, and discard it based on filtering criteria.

The second scenario is the *Noisy* scenario. Here, only 1% of the explicit events were relevant to event materialization. In the third scenario (called the *Filtered* Scenario), 20% of the explicit events were relevant. Finally, the fourth scenario is the *Complex* scenario, in which 80% of the explicit events were relevant for materialization. These four scenarios correspond to the *Standby World*, *Noisy World*, *Filtered World*, and *Complex World* scenarios in [2]. In all of our scenarios, six rules were used (again, similar to [2]), which corresponds to 64 possible worlds.

Table 4: Relevant explicit event results

	Standby	Noisy	Filtered	Complex
Explicit Events/sec (D)	72887	57406	6060	804
Explicit Events/sec (P) $\beta = 0.05$	305810	290697	458	94
Explicit Events/sec (P) $\beta = 0.1$	303030	318725	1414	221
Explicit Events/sec (P) $\beta = 0.15$	290697	319488	2534	282
Explicit Events/sec (P) $\beta = 0.2$	291545	279329	3377	336

The results of these experiments appear in Table 4. In this table, the results from our experiments are in the rows denoted by (P). The results from the work in [2] appear in the row marked (D).

For three out of the four precision values, our framework achieves performance on the same order of magnitude as the deterministic system in [2]. Only for $\beta = 0.05$ in the *Filtered* and *Complex* scenarios the performance impact is such that the performance of our framework is approximately 8-11% of the deterministic system. It is not surprising that for $\beta = 0.05$ the performance is significantly impacted, because when $\beta \rightarrow 0$, it is expected that the required number of samples will approach infinity. Despite the caveats described above, this result is encouraging. It shows that despite the significantly larger effort required by our algorithm to obtain the samples, the same order of performance can be achieved. Moreover, our algorithm lends itself to massive parallelism, and can be optimized so that samples can be generated independently on different machines or CPUs.

We observe that the percentage of relevant events has significant performance implications. When the percentage of relevant events is non-trivial (e.g., 20% as in the *Filtered* scenario), the performance is several orders of magnitude less than those cases in which only a small percentage of events are relevant. Indeed, it can be seen that for both the *Standby* and *Noisy* scenarios, the same level of performance is achieved, outperforming the results obtained in [2]. This difference is attributed due to the differences outlined above between our setting and the setting in [2].

6.3 Scalability: Possible Worlds Analysis

In our second set of experiments, we tested the scalability of our algorithm as the number of possible worlds increases. We carried out this type of experiment for both the *Filtered* Scenario and *Complex* scenario. Figure 6 illustrates the results for the *Complex* scenario. Similar results were observed for the *Filtered* scenario. In both figures, axes are given in a **logarithmic** scale, where value on the x-axis represents the exponent in the number of possible worlds. Therefore, a value of 4 represents 2^4=16 possible worlds.

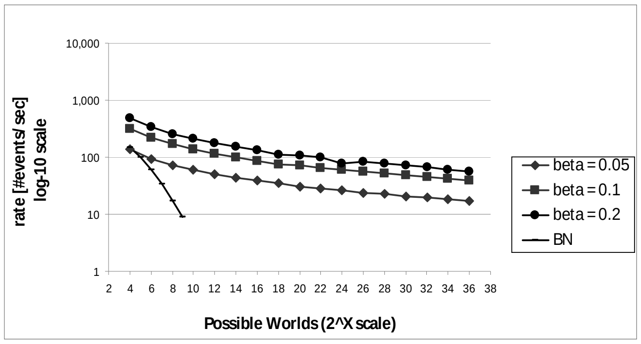

Figure 6: Event rate as a function of possible worlds (*Complex* scenario)

For the RuleSamp algorithm, the event rate decays **sub-linearly** relative to the growth in the number of possible worlds. This demonstrates that the RuleSamp algorithm is scalable in connection with the number of possible worlds. The rate is affected by the required accuracy of sampling. This impact remains more or less constant, independent of the number of possible worlds. The Bayesian network algorithm demonstrates an exponential decline in performance with the increase in the number of worlds. Beyond 2^{29} possible worlds, the algorithm processing rate was less than one event per second.

The reason for this phenomenon is that in many cases, as the number of possible worlds grows, many worlds have a negligible (or even zero) probability of occurring. In such cases, our sampling algorithm (almost) never samples such worlds, concentrating instead on the possible worlds that have significant probability. For Bayesian networks, even worlds with a very small probability must be represented.

6.4 Accuracy: Precision Analysis

In this experiment, we tested several settings of the confidence width β (0.05. 0.1, 0.15, and 0.2). We tested the effect of the different settings for both the *Complex* and the *Filtered* scenarios. In addition, for each scenario, several settings for the possible worlds were tested. The results for the *Filtered* scenario appeared in Figure 7. Similar trends were observed in the *Complex* scenario.

As observed in Figure 6, the required precision obviously has an impact on the performance. However, this impact depends on other factors as well, especially the number of possible worlds. In Figure 7, we observe that the decay in event rate is uniform and depends on the number of possible worlds. For example, for the case of 2^{36} possible worlds, the rate of decay is linear.

Finally, we tested the accuracy of the sampling algorithm. To do so, we ran the Bayesian network algorithm to gather the exact probabilities and measured the distance between the exact probabilities and the results of the RuleSamp algorithm. Figure 6 showed the poor performance of Algorithm 1. As a result, we needed to devise a method in which we could increase the number of possible worlds without deteriorating the performance beyond the feasible limits of our system. The number of possible worlds increases in one of three ways. First, new events increase the number of possible worlds. New rules also do that same thing. Finally, if a rule is updated with additional conditions and probabilities, the number of possible worlds increases. The last method does not result in any change to the Bayesian network; only the marginal distribution tables need to be updated. Therefore, the computational impact on the system becomes reasonable, allowing us to perform the experiment with up to 2^{20} possible worlds.

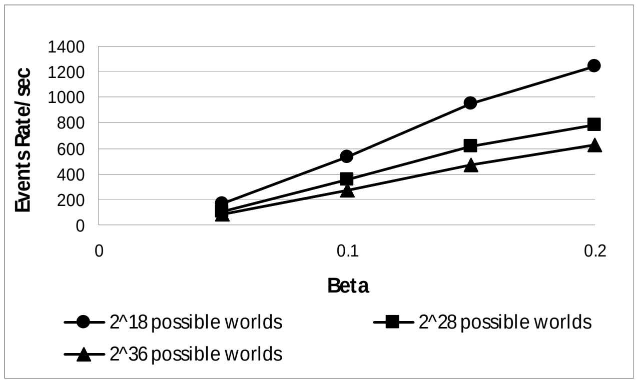

Figure 7: Event rate as a function of precision (*Filtered* scenario)

For each probability estimation, we ran RuleSamp 80 times on average. Table 5 summarizes the results of this set of experiment runs. We observed that for all experimented numbers of possible worlds, the actual accuracy exceeded the expected one.

To better understand the algorithm performance we tracked the *true interval* parameter. We define the *true interval* to be the size of an interval such that only 5% of samples are outside the interval. The results show the true interval to be significantly smaller than the expected interval. Also, all samples were within the 0.05 expected interval.

Table 5: Comparison of true and expected intervals

#worlds	Exp. interval	True Interval			
		True Interval	Avg	Min	Max
2^6	0.05	0.023	0.00840	0.00001	0.02936
2^8	0.05	0.018	0.00761	0.00008	0.02253
2^10	0.05	0.021	0.00932	0.00058	0.03311
2^12	0.05	0.021	0.00827	0.00001	0.02578
2^14	0.05	0.016	0.00777	0.00001	0.01921
2^16	0.05	0.019	0.00751	0.00001	0.02146
2^18	0.05	0.018	0.00862	0.00058	0.02758
2^20	0.05	0.022	0.00912	0.00097	0.02675

To summarize, we have shown that the RuleSamp algorithm performs well in a benchmark of four scenarios. Its performance deteriorates gracefully with the increase of load (represented by the increase in the number of possible worlds) while Algorithm 1 is shown not to scale well. Sampling precision impacts performance, yet the algorithm's ability to exceed the minimum theoretical confidence indicates that choosing a higher β value is likely to provide accurate probability estimation while keeping performance up.

7. CONCLUSIONS

In this work, we presented an efficient mechanism for event materialization under uncertainty. A model for representing materialized events was presented and two algorithms for correctly specifying the probability space of an event history were given. The first provides an accurate, albeit expensive, method based on the construction of a Bayesian network. The second is a Monte Carlo sampling algorithm that approximates the materialized event probabilities. We experimented with the sampling algorithm, showing it to be comparable to the performance of a deterministic event composition system. It is scalable under an increasing number of possible worlds (and uncertain rules), while the Bayesian network algorithm does not scale well. Finally, the sampling algorithm provides an accurate estimation of probabilities.

Our work provides a generic mechanism for managing events and the data they deliver under uncertainty conditions. Additional experiments on our sampling algorithm may enable improving its performance. Another promising area of research is the use of machine learning techniques for the automatic generation and maintenance of rules.

8. REFERENCES

[1] Adi, A. A Language and an Execution Model for the Detection of Reactive Situations. Ph.D. Thesis, Technion – Israel Institute of Technology, 2003.

[2] Adi, A. and Etzion, O. Amit - the situation manager. *The VLDB journal 13*, 5 (May 2004), 177-203.

[3] Balazinska, M., Khoussainova, N., and Suciu, D. PEEX: Extracting probabilistic events from rd data. *Proceedings of ICDE*, (2008).

[4] Beeri, C., Eyal, A., Milo, T., and Pilberg A. Query-Based Monitoring of BPEL Business Processes. In *Proceedings of the ACM-SIGMOD*, (2007), 1122-1124.

[5] Breese, J. S., Goldman, R. P., and Wellman, M. P. Introduction to the Special Section on Knowledge-Based Construction of Probabilistic and Decision Models*. IEEE Transactions on Systems, Man and Cybernatics,* 24, 11, (1994), 1577.

[6] Campbell, M., Li, C.-S., Aggarwal, C., Naphade, M., Wu, K.-L., and Zhang, T. An evaluation of over-the-counter medication sales for syndromic surveillance. *IEEE International Conference on Data Mining - Life Sciences Data Mining Workshop*, (2004).

[7] Chu, D., Deshpande, A., Hellerstein, J., and Hong, W. Approximate data collection in sensor networks using probabilistic models. In *Proceedings ICDE*, (2007), page 48.

[8] Cowie, J., Ogielski, A.T., Premore, B., and Yuanb, Y. *Internet worms and global routing instabilities*. SPIE, 4868, (2002).

[9] Dayal U. et al. The HiPAC project: Combining active databases and timing constraints. *ACM SIGMOD Record*, 17, 1, (March 1988), 51-70.

[10] Demers, A., Gehrke, J., Hong, M., Riedewald, M., and White, W. Towards expressive publish/subscribe systems. *Advances in Database Technology - EDBT 2006,* (2006), 627-644.

[11] Demers, A., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., and White, W. Cayuga: A General Purpose Event Monitoring System. *In IDR 2007, Third Biennial Conference on Innovative Data Systems Research*, (2007) 412-422.

[12] Diaz, O., Jaime, A., and Paton, N. Dear: A Debugger for Active Rules in an Object-Oriented Context. In Proceedings of the 1st International Workshop on Rules in Database Systems, (1993) 180-193.

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

[14] Gehani, N. H., Jagadish, N. H., and Shmueli, O. Composite event specification in active databases: Model and implementation, *Proceedings VLDB*, (1992), 23-27.

[15] Girod, L., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H., and Madden, S. The Case for a Signal-Oriented Data Stream Management System. *In CIDR 2007,* (2007) 397-406.

[16] Green, T. and Tannen, V. Models for incomplete and probabilistic information. *IEEE Data Eng. Bull* 29, 1 (2006) 17-24.

[17] Halpern, J. Y. *Reasoning about Uncertainty*. MIT Press, 2003.

[18] Li, C.-S., Aggarwal, C., Campbell, M., Chang, Y.-C., Glass, G., Iyengar, V., Joshi, M., Lin, C.-Y., Naphade, M., and Smith, J. R. Epi-spire: A system for environmental and public health activity monitoring*. IEEE International Conference on Multimedia and Expo*, (2003).

[19] Paton, N. W. *Active Rules in Database Systems*. Springer, 1999.

[20] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[21] Re, C., Dalvi, N., and Suciu, D. Query evaluation on probabilistic databases*. IEEE Data Eng. Bull.* 29, 1, (2006) 25-31.

[22] Shmueli, G. and Fienberg. S. Current and potential statistical methods for monitoring multiple data streams for biosurveillance. In *Statistical Methods in Counterterrorism*, pages 109-140. Springer Verlag, 2006.

[23] Wasserkrug, S., Gal, A., and Etzion, O. A model for reasoning with uncertain rules in event composition systems. *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence* (UAI-05), (2005), 599-606.

[24] Widom, J. Trio: A system for integrated management of data, accuracy, and lineage. *CIDR*, (2005), 262-276.

[25] Widom, J. and Ceri, S., editors. *Triggers and Rules for Advanced Database Processing.* Morgan-Kaufmann, San Francisco, CA, 1996.

[26] Zimmer, D. and Unland, R. On the Semantics of Complex Events in Active Database Management Systems. *Proceedings ICDE* (1999), 392-399