EFFICIENT AND SCALABLE UNCERTAIN EVENT PROCESSING IN BUSINESS INTELLIGENCE

A Thesis

submitted to

Pondicherry University

in partial

fulfilment of the requirements for the award of the Degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

V. GOVINDASAMY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PONDICHERRY ENGINEERING COLLEGE
PUDUCHERRY – 605 014
INDIA

DECEMBER 2013

Dr. P. THAMBIDURAI, M.E., Ph.D., F.I.E.
Professor of CSE and Principal
Perunthalaivar Kamarajar Institute of Engineering and Technology
(Government of Puducherry Institution)
Karaikal – 609 603

CERTIFICATE

Certified that this Thesis entitled "**EFFICIENT AND SCALABLE UNCERTAIN EVENT PROCESSING IN BUSINESS INTELLIGENCE**" submitted for the award of the degree of **DOCTOR OF PHILOSOPHY** in **COMPUTER SCIENCE AND ENGINEERING** of the Pondicherry University, Puducherry is a record of original research work carried out by **Mr. V. GOVINDASAMY** during the period of study under my supervision and that the Thesis has not previously formed the basis for the award to the candidate of any Degree, Diploma, Associateship, Fellowship or other similar titles. This Thesis represents independent work on the part of the candidate.

(Dr. P. THAMBIDURAI)
Supervisor

Date :
Place : Puducherry

ACKNOWLEDGEMENT

My first debt of gratitude goes to my supervisor, **Dr. P. Thambidurai,** Professor of Computer Science and Engineering and Principal, Perunthalaivar Kamarajar Institute of Engineering and Technology, Karaikal. He is not only my guide, but also my mentor. His patience, genuine care, concern and faith in me during the dissertation process enabled me to attend to life and also complete my research. He has been motivating, encouraging and enlightening all these years.

I would like to thank my Doctoral Committee members, **Dr. N. P Gopalan,** Professor, Department of Computer Applications, National Institute of Technology, Tiruchirappali and **Dr. K. Vivekanandan**, Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry for their support, guidance and helpful suggestions. Their guidance has served me well and I owe them my heartfelt gratitude.

I register my gratitude to **Dr. D. Govindarajulu**, Principal, Pondicherry Engineering College, Puducherry and **Dr. V. Prithiviraj**, Former Principal, Pondicherry Engineering College, Puducherry for their support to pursue the research work. My heartfelt thanks are due to **Dr. D. Loganathan,** Professor and Head, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry for the academic, technical and logistical support provided by the Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry. I take this opportunity to thank wholeheartedly **Dr. M. Ezhilarasan**, Professor and Head, Department of Information Technology,

Pondicherry Engineering College, Puducherry for the incredible leadership, timely and proper advice throughout the research.

I would like to thank all of my Fellow Researchers who have extended their continuous encouragement and support so as to improve the quality of the thesis. My special thanks goes to all the faculty members and non-teaching members of Department of Computer Science and Engineering and Department of Information Technology, Pondicherry Engineering College, Puducherry for their support.

Finally, I thank all of my family members for their understanding, help and patience throughout the research.

ABSTRACT

During the last decade, Complex Event Processing (CEP) has been one of the most rapidly emerging fields to identify the composite (high-level) events from the primitive (low-level) events that occur online. Due to the availability of massive amount of business transactions and numerous new technologies for information processing, it has now become a real challenge to provide real-time event driven systems that can process data under high input data rate in an automated and systematic approach. In recent days, many researchers have focused their attention on the challenges of efficient event monitoring of real-time applications like monitoring RSS streams, stock tickers, RFID data streams and click streams that generate a number of events with high uncertainty. The main objective of the research is to develop a probabilistic framework named as Probabilistic Complex Event Processing (PCEP) system in the context of real-world stock ticker streams that execute complex event pattern queries on the continuously streaming data with uncertainty. The PCEP system consists of two phases named as Efficient Generic Event Filtering (EGEF) and Probabilistic Event Sequence Prediction (PESP).

In the EGEF phase, a Non-deterministic Finite Automaton-heap (NFA_h) based event matching allows to filter the relevant events by identifying the occurrences of user defined event patterns in a large volume of continuously arriving data streams. In order to express the complex event patterns in a more efficient form, a CEP language named as Complex Event Pattern Subscription Language (CEPSL) is used. CEPSL is extended from the existing high level event query languages. A Predicate based Subscription Grouping algorithm (PSG) is proposed to group user subscriptions based on access predicates to improve the scalability. Furthermore, the PCEP establishes a query aware partitioning scheme that deploys two efficient techniques such as row/column scaling and pipelining that dispatch the grouped user

subscriptions into separate machines. It proposes an efficient distributed event processing approach that distributes the process based on access predicate of subscription clusters across a multiple number of NFA_h machines. It performs a NFA_h pattern matching mechanism to extract the relevant events from the large number of incoming events based on the user subscriptions.

The second phase of PCEP derives a stateful composite event sequences from the filtered relevant events based on the probabilistic framework. It constructs an event hierarchy in the form of a Dynamic Fuzzy Probabilistic Relational Model (DFPRM) that is used to represent the probability space in terms of the concept of individuals, their properties and relations among them. DFPRM computes the joint probability distribution using the conditional probabilistic dependencies among the event sequences in accordance with the rules. In order to formulate the combination of event sequences with the reduced overhead, Probabilistic Fuzzy Logic (PFL) is used to infer the semantic correlation among the event sequences to estimate the fuzzy linguistic variables from computed Conditional Probability Distributions (CPD) in the large probability space. PFL enhances the robustness of the complex event detection process under uncertainty.

To evaluate the effectiveness of the PCEP system empirically, the system is implemented in the Publisher/Subscriber model using Java Message Service (JMS) subscription API in Java based prototype. The experimental results are presented in terms of scalability and efficiency using the prototype. Results reveal that PCEP system offers high scalability due to an efficient event filtering approach and achieves high efficiency in probabilistic complex event detection through handling of uncertainty using DFPRM model and PFL approximation mechanisms. It is inferred that the PCEP system is more efficient and scalable than the other existing CEP approaches.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.

FIGURE NO.	TITLE	PAGE NO.

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.

LIST OF ABBREVIATIONS

AIG	-	Active Instance Graph
AIH	-	Active Instance Heap
AIR	-	Automaton Intermediate Representation
AMiT	-	Active Middleware Technology
API	-	Application Programming Interface
BAM	-	Business Activity Monitoring
BDD	-	Binary Decision Diagram
BI	-	Business Intelligence
BMS	-	Business Metric Service
BMI	-	Body Mass Index
BN	-	Bayesian Network
BP	-	Blood Pressure
BPM	-	Business Process Management
BPMN	-	Business Process Management Notation
BPEL	-	Business Process Execution Language
BRMSs	-	Business Rule Management Systems
CA	-	Condition Action
CAMEP	-	Complex business Activity Monitoring with Event Processing
CCL	-	Continuous Computation Language
CEL	-	Cayuga Event Language
CEP	-	Complex Event Processing
CEPL	-	Complex Event Processing Language
CEPSL	-	Complex Event Pattern Subscription Language
CLUES	-	Cleansing Language for Unreliable RFID Event Streams
CPD	-	Conditional Probability Distributions
CPT	-	Conditional Probability Table
CQL	-	Continuous Query Language
D/A	-	Debt-to-Asset Ratio

DAG	-	Directed Acyclic Graph
DAHP	-	Database Active Human Passive
DBMS	-	Database Management Systems
DDMP	-	Distributed Dynamic Multi-Point
DFA	-	Deterministic Finite State Automata
DFPRM	-	Dynamic Fuzzy Probabilistic Relational Model
DRL	-	Drools Rule Language
DeSL	-	Default Subscription Language
DSMS	-	Data Stream Management Systems
ECA	-	Event-Condition-Action
EG	-	Earnings Growth
EGEF	-	Efficient Generic Event Filtering
EI	-	Event Instance
EID	-	Event Instance Data
EIS	-	Event Instance Sequence
EPL	-	Event Processing Languages
ET	-	Event Type
FL	-	Fuzzy Logic
GB	-	Gigabytes
GDP	-	Gross Domestic Product
Hb	-	Hemoglobin Content
HMM	-	Hidden Markov Model
IFP	-	Information Flow Processing
IDE	-	Integrated Development Environment
I/O	-	Input/Output
IBM	-	International Business Machine
IR	-	Interest Rate
IT	-	Information Technology
JMS	-	Java Message Service
KPIs	-	Key Performance Indicators
MAV	-	Moving Average Value
MB	-	Megabytes

MI	-	Manufacturing Index
MQO	-	Multi-Query Optimization
NASDAQ	-	National Association of Securities Dealers Automated Quotation
NFA	-	Non-deterministic Finite Automaton
NFA_h	-	Non-deterministic Finite Automaton-heap
OMG	-	Object Management Group
P/E	-	Price-to-Earnings Ratio
PC	-	Personal Computer
PCEP	-	Probabilistic Complex Event Processing
PCEPr	-	Probabilistic Complex Event Processor
PCET	-	Probabilistic Complex Event Triggering
PEs	-	Processing Elements
PEEX	-	Probabilistic Event Extraction System
PESP	-	Probabilistic Event Sequence Prediction
PFL	-	Probabilistic Fuzzy Logic
PGM	-	Probabilistic Graphical Modeling
POJO	-	Plain Old Java Object
PSG	-	Predicate based Subscription Grouping
PSR	-	Price-to-Sales Ratio
PSS	-	Probability Sample Space
QoS	-	Quality-of-Service
rCML	-	RuleCore Markup Language
RAM	-	Random Access Memory
RDBMS	-	Relational Database Management System
RFID	-	Radio Frequency Identification Devices
ROE	-	Return On Equity
RSI	-	Relative Strength Index
S4	-	Simple Scalable Streaming System
SASE	-	Stream-based And Shared Event Processing
SF	-	Sequence Formation
SLA	-	Service Level Agreements
SPC	-	Stream Processing Core

SQuAl	-	Aurora's Stream Query Algebra
SQL	-	Structured Query Language
SiSL	-	Simple Subscription Language
StSL	-	Strict Subscription Language
ST	-	Sequence Traversing
STREAM	-	Standard sTREam datA Manager
TCS	-	Tata Consultancy Services
U-Topk	-	Uncertain Top-k query
U-k Rank	-	Uncertain k-Ranks query
XML	-	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In today's Internet world, the availability of real-time data has increased enormously in size every year due to the increase in number of distributed software applications. These applications generate high volume of data stream continuously from geographically distributed multiple sources at an unpredictable rate. There is a need to derive events from the continuously flowing data in a timely manner [1]. Business processing systems are only interested in acquiring high level of intelligence from the available data with effective reasoning. Complex Event Processing (CEP) is the essential technology in the field of event processing that is useful for business activity management applications [2]. In the emerging market, an active database system with the integration of CEP provides the essential functionality in the business process management to monitor and to optimize the process of business enterprise. In order to react automatically to all events of interest, the business process system must perform effective reasoning in an event composition system to acquire a high level of intelligence from the available data. Therefore, the intelligent system integrates CEP technology that can process the large amount of data flow from multiple sources [3]. The core of the CEP is the CEP engine (processor) that detects event patterns from the large number of incoming events. An event pattern expresses the rules in a declarative language to describe the complex relationship among the incoming events in order to obtain the relevant information to trigger the output events [4].

The accuracy and performance of event derivation largely depend on the reliability of data sources. A data source has inherently unreliable data collection process, or generates incorrect data leading to uncertainty [5]. Further, CEP in the presence of uncertain events is a challenging task and this problem hampers the

accuracy of derived events. Many advanced technologies have been evolved to store and to process the large number of continuously arriving incoming data with uncertainty. There is a possibility for the occurrences of errors and incomplete information in the incoming data due to the unpredictable event sources. In event based systems, uncertainty arises due to the gap among the real occurrences of events, to which the system must react and the capability of event-driven systems to produce exact events. A lot of uncertain data algorithms [6] are developed to process the uncertain data that represent events in terms of probabilistic distributions rather than deterministic values. Thus, a probabilistic interpretation of data has recently attracted the interest of the Active Database community to process the uncertain data. The gap between actual events and event notifications is to be explicitly managed to handle the uncertainty using probabilistic framework. In order to derive the composite events under uncertainty, the uncertainty data can be modeled in the form of probabilistic graphical model to derive the events externally under uncertainty [7]. The main challenge of explicit probabilistic event processing is that the rule-based systems need to process the large number of continuously arriving events.

This research presents a generic system for the modeling, the management of events and the implementation of CEP and rules with uncertainty. This proposal highlights the architecture in the form of Publisher/Subscriber middleware model that conditionally detects the required high-level information from the continuously arriving massive stream of data according to the user defined rules. The event processing is performed in two phases such as Efficient and Generic Event Filtering (EGEF) and Probabilistic Event Sequence Prediction (PESP). In the first phase, the distributed non-deterministic Finite Automata-heap (NFA_h) based pattern matching filters the relevant event sequences from the large number of incoming events based on the domain experts specified rules. In the second phase, the filtered events enter into the PESP phase that derives stateful composite event sequences from the filtered relevant event sequence based on the probabilistic framework. The event hierarchy is constructed in the form of a graphical model called as Dynamic Fuzzy Probabilistic Relational Model that captures the semantics. The probabilities of

possible worlds are represented in terms of the concept of individuals, their properties and relations among them using an abstraction based on a Bayesian Network. To improve the materialization efficiency, this system employs a Probabilistic Fuzzy Logic over a set of rules that approximate the probabilities defined by the possible worlds of probability space. The semantic correlation among the event sequences is inferred using linguistic variables with reduced sample space. PCEP performs efficient complex event detection in a highly dynamic environment that achieves high throughput and efficiency while retaining less detection latency. Further, PCEP enhances the robustness of the complex event detection process under uncertainty by providing high flexibility and scalability.

1.2 PREAMBLE

This section presents the progression from traditional Data Base Management Systems to the present Complex Event Processing systems. The explosion of data events due to the proliferation of internet and the necessity to process this data in near real-time propelled the arrival of Complex Event Processing systems. The integration of CEP with Business Intelligence, uncertainty inherent in event data and a broad classification of uncertainty are discussed.

1.2.1 Evolution of Information Flow Processing

Database Management Systems (DBMS) process the data which are stored on a disk. DBMS defines the structure of the data in the form of the relational model. DBMS can analyze, arrange, collect and manipulate the data according to the user dynamic queries [8]. More commonly, Structured Query Language (SQL) is used to support the query specification in the database systems. The key feature in DBMS is that the data is stored permanently, whereas queries are entered dynamically to process the stored data. In a real-time scenario, the DBMS cannot store and process the continuously arriving data. Furthermore, it does not fulfill the concept of timeliness of flow processing that is mostly desirable in real-time applications that require instantaneous response. However, it is practically infeasible to store a large amount of data in its whole entirety.

The static nature of DBMS led to the development of Information Flow Processing (IFP) [9]. IFP can timely process the large amount of information flows from multiple sources in order to extract new knowledge based on a set of processing rules. IFP is different from DBMS in various aspects such as implementation architecture, processing rule languages, the definition of a data model and the way of processing. IFP is broadly classified into two types such as Data Stream Management Systems (DSMS) [10] and the Complex Event Processing (CEP) systems [11]. Unlike DBMS, DSMS is designed to process the transient data that flow continuously from multiple sources. DSMS provides updated output according to the current state of newly arriving data.

1.2.2 Data Stream Management Systems

Continuous query community has developed Data Stream Management System (DSMS) to improve the scalability of centralized continuous query systems. DSMS is a specialization of DBMS that processes the continuously arriving event streams against standing or continuous queries in a highly dynamic rate [10]. It is not only capable of detecting patterns in event streams but also acquires updated output according to the arrival of incoming event streams. A new interactive model is introduced to execute the queries periodically or continuously according to the arrival of incoming new event streams. DSMS incorporates temporal operators along with the relational algebra operators such as selections, aggregate and joins in the DBMS system. In order to process the incoming transient data in a timely manner, the queries are continuous in nature and stay active for a long time. DSMS provides the declarative subscriptions using more powerful query language. However, DSMS achieves only limited scalability due to the limited number of subscription processing and also the limited Multi-Query Optimization (MQO). DSMS differs from conventional DBMS in several ways:

- Event stream is unbounded and irrespective of the arrival order of the event streams.

- It is difficult to store and process the large number of continuously arriving event streams due to the limited resource and timing constraints.
- Effective active notifications are provided according to the updated information rather than user explicit queries.

1.2.3 Complex Event Processing

Complex Event Processing (CEP) is an emerging technology that extracts the high level information from the distributed message based systems. The term 'CEP' was coined by David Luckham in 1990. CEP is a highly crucial technology that encompasses algorithmic methods, techniques and tools to process the events while they occur in a continuous and timely fashion. This technology derives the valuable high level knowledge (complex events) by making sense of low level primitive events. Here, knowledge takes the form of composite events, which are the combinations of several primitive events. CEP is the most favorable platform to build and to run the various number of real-time applications. CEP performs processing of the large number of real-time events to trigger a suitable action [9]. CEP enables an organization to identify the occurrences of patterns of low level events by filtering, correlating, aggregating and computing multiple streams of high-volume, high speed business events. CEP differs from DSMS in three reasons as follows:

- ♦ DSMS provides general capabilities to process streams, but CEP can be able to detect event patterns.
- ♦ In CEP systems, the event languages express the correlation between the events and ordered constraints, but, in DSMS the long queries are only able to express the relation between the events.
- ♦ CEP systems can handle a large number of concurrent queries with high scalability, which outperforms the DSMS [12] [13].

Gartner defined CEP as follows “CEP is an approach that identifies data and application traffic as events of importance, correlates these events to reveal predefined patterns and reacts to them by generating actions to systems, people and devices” [14]. The CEP engine is responsible for processing the large number of continuously arriving events from distributed message systems and discovering the complex event patterns of interest among the events. Finally, CEP provides notifications to consumers about the detected high level semantic rich events. These applications come up in a various number of domains: trading in the financial markets, potential risk management, RFID-enabled monitoring in logistics management, supply chain management, click-stream analysis, network intrusion detection, business process monitoring, military power grid monitoring and infrastructure monitoring [2]. CEP engine processes tens of thousands of events while concurrently analyzing thousands of event processing strategies. The CEP system supports high performance, scalability, manageability and fault tolerance for mission-critical event-driven systems.

1.2.3.1 Architecture of CEP

CEP is a middleware designed to timely process the large amount of event notifications as they flow from the peripheral to the center of the system in order to identify the composite events relevant for the event based applications. There are event sources and event consumers at the periphery of the system. The general architecture of CEP based applications is shown in Figure 1.1.

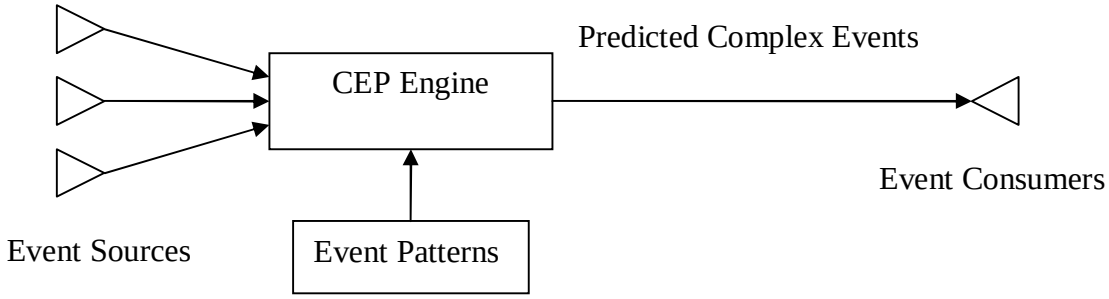

Figure 1.1: Architecture of CEP

The CEP engine is the fundamental component of a CEP middleware that takes the flow of information (the low level of events) arriving from different event

sources or Event Producers as its input, processes them and produces other flows of information (high flow of information) directed toward a set of event consumers or subscribers. The event patterns provide the set of event definition rules as a complex query language that describes how to filter, combine and aggregate incoming information to produce outgoing information. Complex Query language specifies the user-defined patterns that describe how composite events are defined from primitive ones. The event pattern has the ability to derive the complex relationships between the incoming events which are flowing into the CEP engine. After processing the large number of incoming event streams against the user defined query patterns, CEP detects the occurrences of unique patterns of (low-level) events on the higher-level events. Further, the system notifies about the event occurrence of the event consumer or as an input to the other CEP engine in the system [9].

1.2.3.2 Internal Processing of CEP

Figure 1.2 depicts the main functional components involved in the internal event processing of CEP and highlights the precise description of the functionalities offered by the CEP engine. CEP performs the task of identifying composite events from the large number of continuously arriving primitive events. This general behavior of CEP can be decomposed into a set of fundamental operations carried out by the different components as shown in Figure 1.2.

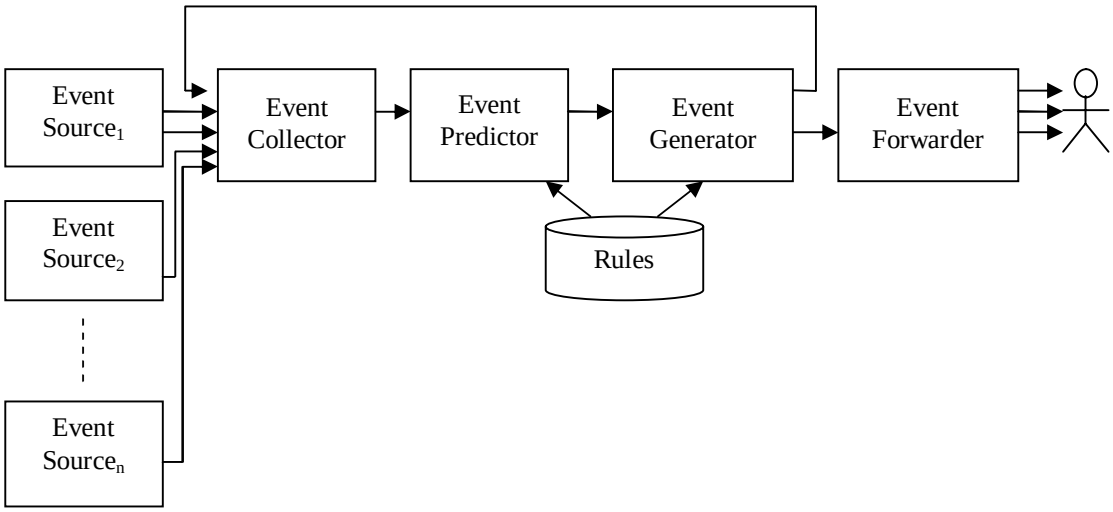

Figure 1.2: Internal Processing of CEP

A large number of incoming events enter into the CEP engine through the Event Collector module that gathers the continuously arriving events from multiple sources. It manages the channels connecting to the source with the CEP engine. The Event Collector module acts as a multiplexer to receive the large number of incoming events. Further, it structures a single stream of events from the multiple streams into the CEP engine. A periodic clock is associated to support timely processing. The CEP engine processes the events according to the rules in two phases such as the Prediction Phase and the Generation Phase. The user defined queries are represented as Condition Action Processing rules. Event Predictor module checks the occurrence of event patterns specified in the condition clause in the large number of incoming events. If the Event Predictor module detects the event pattern of incoming events, the action specified in the action clause is sent to the Event Generator module. Here, the Event Generator module triggers the action to produce results in the form of alarms, emails or messages [9]. The output result is sent to the event consumers through the Event Forwarder module. It can also be sent internally to the input of Event Collector phase to perform the event processing again.

In the CEP systems, all the inbound events cannot be stored for further processing. Any real-time event will have an uncertainty associated with it and the knowledge to be inferred from the events should be just in time. These unique requirements of the CEP systems advocate for the need of efficient filtering techniques that filter the irrelevant events and store only the relevant events for further processing. Thus, there is a need for efficient event processing in the presence of uncertainty.

1.2.3.3 Importance of CEP

The CEP system is relevant in diverse domains due to the following unique aspects:

i) Ability to Give Instantaneous Repose

CEP is suitable for real-time applications due to its inherent nature of instantaneous response. It has processing capability of incoming streams greater than

the event arrival rate. It is especially useful in alarm detection scenarios, stock trading and volcanic monitoring where response time is of the order of milliseconds rather than seconds.

ii) Distributed Processing

CEP provides distributed processing of queries that are particularly useful in the case of sensor network applications with limited resources. It reduces the energy consumption and communication overhead even under high computation capabilities. However, it is harder to implement in the financial applications due to privacy issues and high demand for consistency.

iii) Logging and Analysis of Data Streams

This feature of CEP provides the basis for acquiring improvements in business logic. It is especially beneficial to achieve event detection for financial applications that require logging of data for future analysis (e.g. stock trading).

iv) Heterogeneity of Data Sources

CEP can process the incoming event streams when high heterogeneity exists between the external multiple sources.

v) Learning Methods

CEP can directly integrate the learning methods to process and analyze the incoming data streams with high efficiency.

vi) Handling of High Volume of Data

CEP can handle and process the high volume of incoming data with low latency and high processing speed.

1.2.3.4 Characteristics of CEP Systems

The characteristics of the CEP system are summarized as follows:

- It takes continuous and infinite stream of events as an input from the external sources and not from a central database
- It provides real-time processing and event detection with low latency
- It processes the volatile event streams
- It can cope with the large number of submitted queries as well as a large number of events in real-time.
- It is mainly concerned with the strong temporal relationship that exists between the incoming events and their patterns rather than the individual events.
- It can filter, correlate and aggregate data from multiple external sources to infer the high level information among the events.
- It influences the continuously arriving real-time data directly rather than the historical data stored in the database.
- It follows a DAHP (Database Active, Human Passive) model where the system does continuous processing of incoming event streams and notifies the user based on semantic information [14].

1.2.3.5 CEP General Use Cases

The general belief is that the stock trading is the main use of CEP systems in event monitoring applications [14]. However, in the present days, CEP plays a vital role in many other intriguing applications such as IT, Financial Markets and Manufacturing Organizations. These applications include:

i) Cleansing and Validation of Data

CEP performs event filtering and data validation applications to filter the irrelevant events from the continuously arriving event streams. It processes the incoming events based on the event patterns to determine the irrelevant events which fall outside the pre-defined patterns.

ii) Alerts and Notifications

CEP systems generate event notifications such as alarms, messages and emails in a real-time business system, when problems occur.

iii) Decision Making Systems

CEP systems are used in automated business decision making to take the best decisions using current and past information maintained in knowledge base.

iv) Feed Handling

Most CEP platforms have in-built feed handlers that facilitate to process the common data formats.

v) Data Standardization

CEP engine is capable of standardizing the data of the same entity from different sources within a common reference schema.

1.2.3.6 Applications of CEP

Some applications based on CEP are listed as follows:

i) Stock Market Trading

Financial application needs the continuous analysis of stock data to identity the trends in the stock market. CEP identifies the suitable opportunities for buying or selling securities based on the patterns of price movements in real-time.

ii) Real-time Supply Chain

In retail and logistics industries, CEP performs continuous analysis of RFID readings to automate the object tracking and the supply chain management. It improves the operational performance of the industry in terms of sales and stock control, industrial process automation and human resource management.

iii) Fraud Detection

In the fraud detection application, CEP detects the fraudulent use of credit card by inspecting the continuous streams of credit card transactions. It correlates the fraud-indicator events at all stages of the claim value chain in real-time.

iv) Production Management and Quality Assurance

In manufacturing control systems, certain anomalies are detected and the e-mails or event notifications are generated to provide alerts for the attention of the supervisor.

v) Sensor Network

In environmental monitoring, CEP correlates the large number of sensor data coming from the various sensors. Furthermore, it acquires information about the observed world for predicting disasters as soon as possible.

vi) Intrusion Detection

In intrusion detection systems, CEP processes the information from multiple security devices to detect promptly and to analyze the network traffic in a real-time. It can anticipate attacks in a corporate network to generate alerts when unexpected event happens.

vii) Telecommunication

In Telecommunications systems, CEP captures and analyzes the information emitted by the network elements to signify the proper allocation of service for delivery infrastructure in real-time [2] [14].

1.2.4 Integration of Business Intelligence in CEP

The explosive growth in IT technology leads to the successful implementation of Business Intelligence (BI) techniques in highly sophisticated Business Process Management. BI paves the way for the business enterprise to monitor and to analyze the business process in real-time to trigger necessary actions. It provides the right direction for the growth of an organization in a highly competitive world. BI is extremely helpful to support and improve the overall performance of the enterprise business processes. The business process integrates CEP to monitor, analyze and act on changing business conditions in real-time when the event occurs [15]. CEP [2] [11] is a highly crucial component of any enterprise business solutions that can empower more dynamic, real-time, profitable automated business applications. It gathers data from multiple external sources using different methods over different frequency. Thus, there is a need to derive events from the continuous flow of data in a timely manner. In order to derive the event, business processing systems acquire the high level of intelligence from the available data with effective reasoning. Therefore, an effective business intelligent system must require CEP technology to provide event notification through the filtering, aggregation and correlation of the data [16].

1.2.4.1 Business Intelligence

Business Intelligence transforms raw data into useful information for successful business management. To transform raw data, BI encompasses a set of methodologies, process architectures and the latest technologies to extract meaningful information. It facilitates the businesses to make informed decisions based on real-time data that supports an enterprise ahead of its competitors [17]. Traditionally, BI technology focuses on the core features such as reporting and analytics, but later, a new set of features emerges to make it suitable for various real-time applications in a commercial world. Forrester's BI research explains that the new technology is evolving on the cutting edge of new trends to facilitate the enterprise in order to gain competitive advantage in their industries. BI consists of five principal components that perform multi-dimensional analysis on the business

process. The data warehouse of an enterprise is built in order to assemble the useful information about the organization. The five principal components of Business Intelligence are multidimensional analysis, reporting, data mining, financial consolidation and budgeting and Key Performance Indicators (KPIs).

1.2.4.2 Event-driven Business Process Management

Event-driven Business Process Management is a combination of two key technologies such as Business Process Management (BPM) and CEP. BPM is a software platform that models and optimizes the business process in order to achieve significant gain in a competitive market. On the contrary, CEP is a parallel running platform used to process the business information in the form of events to make a better decision for business improvement [18]. In business processing, a large number of events from multiple event sources are used to trigger the business process that results in another business level event to improve the business. Thus, CEP plays a main role in the processing of a large number of business events to trigger a new complex event in the process of business improvement [11] [15].The event driven BPM acts as an event producer to generate a large number of events. Moreover, the event processing system is used to process the events that result in derived events, which are either returned to the BPM system or provided as an input to any other application. Now, the BPM system can act as an event consumer that receives a resultant event from the event processing system and triggers a specific business event according to the situations detected in the business system.

1.2.4.3 Business Process Management

Business Process Management (BPM) is a unified approach that deals with modeling, managing, orchestrating and executing of an enterprise's business processes. A business process is defined as the structured set of activities to achieve the goal of a business organization. Now-a-days, the systematic BPM software evolves in the platform of service-oriented architecture to model the business process in a workflow system. It promotes the business process in a more effective and efficient manner that also strives for innovation, flexibility and integration. In

order to model the large amount of data, Business Process Management Notation (BPMN) and Business Process Execution Language (BPEL) are the major software standards of BPM. Further, BPEL is a standard language that addresses the execution process of BPM.

1.2.4.4 Business Activity Monitoring

Gartner coined a term called Business Activity Monitoring (BAM) that aims at "providing real-time access to critical business performance indicators to improve the speed and effectiveness of business operations" [19]. A large number of external sources in multiple domains generate a massive amount of events in real-time. Furthermore, the critical business Key Performance Indicators are identified in order to get a better insight into the business activities and thereby to improve the effectiveness of business operations. Most commercial BPM software products such as Oracle BAM, TIBCO Nimbus and IBM Tivoli deploy BAM dash-boards to monitor and to report violations of service level agreements. BAM dash-boards display the performance of the system in the form of graphical meters. However, the main limitation in such products is that the business monitoring is performed within the intra-organizational setting. Nowadays, many companies outsource their business process to other external companies to meet their joint customer needs. This process leads to a cross-organizational process where the organizations delegate their process execution to other companies for supporting an inter-company cooperation [20]. However, it is difficult to monitor the business process across the inter-company cooperation to improve the performance.

1.2.4.5 Key Performance Indicators

Key Performance Indicator (KPI) is the most crucial factor for the success of the business enterprise to detect problems and to trigger business decisions [21]. KPI encompasses a set of attributes that have strong concurrence with the data in the invoking business application or executing code. The domain experts utilize Business Metric Service (BMS) to select the suitable KPIs and to calculate the numerical values for the set of attributes in the KPIs to maintain the business

process**.** BMS is a specialized business service that provides the relevant information by rapidly accessing the wide range of data, not being intertwined with the technical information of the underlying business process. Furthermore, the persistent database stores the derived KPIs, so that the multiple users can access the KPIs without incurring the expense of recalculation. The measurement of the performance of the business is noteworthy to identify the gaps between the current and the desired level of performance [22]. KPIs are to be carefully selected so that the KPI provides information to take action for improving the performance of the business process in the system.

1.2.5 CEP Terminologies

CEP technology consists of a set of basic terms that play a vital role in the event processing functionality. This section highlights a generic idea of the standard terms and definitions involved in the CEP terminology [23].

1.2.5.1 Event

An event is an actual occurrence or a significant happening that falls within a domain of discourse. It is a piece of data representing the fact that something happened in the real world [24][25]. For example,

- Financial market events: Buy 1,000 shares of Microsoft at $25. 45
- Supply chain events: RFID tag 11010 is scanned at 2.45 a.m. at the dock door 5
- Security events: TCP/IP addresses 134. 21. 48. 198 accessed server 7

1.2.5.2 Representation of Events

In the CEP, languages such as XML, Plain Old Java Object (POJO) and Tuples are used to represent the events. Among them, the tuple data structure provides the efficient representation of events that facilitate easier methodology to process the large number of incoming events faster with less computation cost and

overhead [23]. For example, the tuple is a simple data structure consisting of a set of attributes and their corresponding values.

i) Certain Event Representation

The events with certain attributes can be represented by a single tuple with a set of attributes and their corresponding values {$Attribute_1$ = "val_1", $Attribute_2$ = "val_2", $Attribute_3$ = "val_3", ….$Attribute_n$ = "val_n"}.

ii) Uncertain Event Representation

The event represents uncertain attributes as more than one tuple where each tuple is associated with the probability. The associated probability of every event depends on the occurrence of the events with their corresponding set of attributes and their value. For example, the uncertain events in the stock market are represented as a set of tuples where each tuple has a set of attributes, each attribute has a certain value and each value has a corresponding probability.

Stream Id	$Att_1 = val_1$	$Att_2 = val_2$	$Att_3 = val_3$

Figure 1.3: Event Tuple

1.2.5.3 Types of Events

An event is defined to be an occurrence of interest in time. The event is classified into two types such as primitive or simple events and complex or composite events. For example, in the stock exchange scenario, a simple event refers to the buying and selling of stock. The simple event object consists of the stock symbol, the name of the company, the volume of stocks, the price and the identifier of the selling trader. However, the complex event is the aggregation of the number of simple events that are derived from the record of all the buying and the selling of the stocks of a company.

i) Primitive Events

The Primitive events are defined as the occurrence of a single event at a point in time. They are explicitly generated from external event sources.

ii) Composite Events

Composite events are an aggregation of a set of primitive events with the help of various operators of the underlying event algebra. They are mostly derived from the other events. They cannot be directly measured, but their occurrence can be inferred from the logical or temporal relationship between the simple events [23].

1.2.5.4 Event Attributes

Each event associates with itself a set of attributes called as event attributes. These attributes define the information about the action that influences the occurrence of the event [23]. The attributes record the timestamp of the event and also maintain the state of the database.

1.2.5.5 Event Stream

The event processing engine receives a set of events from the multiple sources that are grouped together and are called event streams. Event stream is a linearly ordered sequence of events that is ordered according to the arrival time of the event into the system. The usage of event streams varies according to the requirement of the implementation environment. Most of the applications restrict the implementation for pre-defined event types, but some of the implementations can also support event streams that consist of the user-defined or different types of events.

1.2.5.6 Event History

An event history maintains a history of event occurrences within a given point of time. It stores all the event instances of event ‘E’ as well as their associated

data within a specified time. The event history E (t_2, t_1) is the set of event instances of E within occurrence time between t_1 and t_2 [23] [24].

1.2.5.7 Event Instance Sequence

An Event Instance Sequence (EIS) is a partially ordered set of event instances. EIS reflects the order set of the occurrence times of its event instances [25].

1.2.5.8 Event Type

Event Type (ET) describes the essential features of the events in terms of the parameters in a more abstract way. ET identifies the unambiguous nature of the category of the event.

1.2.5.9 Event Instance

Event Instance (EI) stores and maintains the set of parameters that defines the relevant information about the events. EI is used to represent the concrete occurrences of the event within the system. It helps to identify the impact of the occurrence of the event on the other types of events [25].

1.2.6 Event Processing Languages

CEP aims at deriving a more concise and high meaningful information from the large volume of lower level events based on the correlation acquired, from pre-defined event patterns. In general, the event patterns express the event queries which consist of the events connected by event operators and also specify the constraints on event attributes in the form of predicate expressions [26]. However, these expressions do not correlate the temporal relationship between the events in the dynamic nature. Thus, querying events in CEP is totally different from the traditional querying in the database, since the event patterns are pre-registered queries, which execute against the large number of continuously arriving event streams. In CEP, the event queries are expressed using a high level programming language called Event Processing Languages (EPL). EPL expresses the event patterns in an expressive and declarative way. EPL is classified into two categories

such as stream-oriented or transforming languages and rule-oriented or detecting languages. The stream-oriented languages are preferable in Data Stream Management Systems while rule-oriented languages are used in CEP systems [27].

1.2.6.1 Stream-oriented Languages

Stream-oriented languages are developed based on the context of the DSMS. They express the complex event queries over the DSMS in real-time applications, where continuously a large number of incoming event streams enter into the system [11]. The stream-oriented languages are an optimized language suitable for data streams. These types of languages are mainly extended from the database query language SQL. Furthermore, it consists of three types of operators such as relation-to-relation, stream-to-relation and relation to stream operators. First, the relation-to-relation operator executes complex data queries over relational tables that consist of standard database operators such as select, union, aggregate, intersect, except, duplicate-eliminate and different types of joins. Second, the stream-to-relation operators specify the different sliding window operators that transform the input streams into temporary relational tables. The last, relation-to-stream operators convert the data from a relational table into output streams. On the contrary, the stream-oriented language pays less attention to the queries such as filtering, joins and aggregation to derive the timing and the temporal relation among the events. However, the derivation of temporal relation is extremely useful in the CEP because it helps to achieve the timely processing of incoming events Some examples of these types of languages are Continuous Query Language (CQL) in STREAM systems [28] and Continuous Computation Language (CCL) in coral8 [29]. These query languages are suitable for financial applications, where the aggregation of market data is efficiently correlated to predict the future stock price in market trading.

1.2.6.2 Rule-oriented Languages

The rule-oriented language specifies the rules for processing streams of event processing systems in the area of Active DBMSs. It defines the query

languages in the form of Event-Condition-Action rules [30] that act as a formalism to define the execution of events, when the incoming event satisfies the conditions mentioned in the rule specifications. The rule-oriented language executes the action well integrated with the existing query languages to implement the event queries such as rule languages [31]. In order to achieve high flexibility, Business Rule Management Systems (BRM) provides the standard specifications that implement the event queries as a rule language. BRM hides the complex syntax of production rule languages and also supports the various temporal aspects for modeling the event types and data. Some of these languages include Drools Rule Language and JRules [32]. These languages are used for integrity constraint enforcement, authorization checking and versioning in production systems. Furthermore, these languages provide a more advanced platform for CEP in the large scale and efficient knowledge based expert systems.

1.2.7 Event Filtering

Event filtering is the pre-processing scheme. It is performed ahead of the event prediction to filter out the irrelevant events from the incoming event streams. It is implemented based on the Publisher/Subscriber model to perform the matching between the large numbers of incoming events (publications) and the domain expert specified rules (subscriptions) [33].

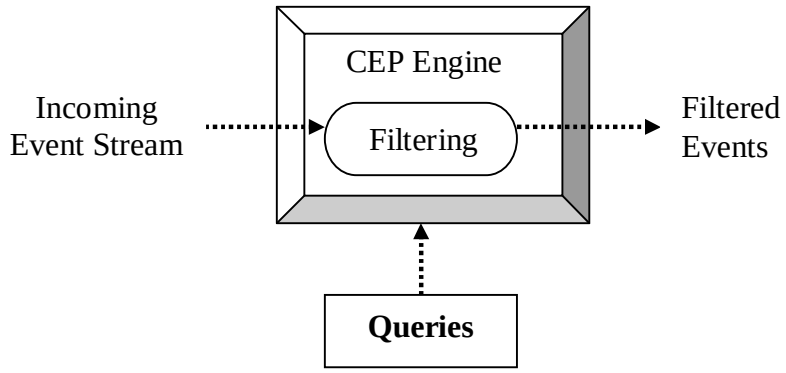

Figure 1.4: Event Filtering

Figure 1.4 shows the exploration of CEP design patterns with the basic filtering. The filtering is easy to implement in the number of non-CEP products or custom-built applications. The CEP engine performs the event filtering on the incoming event streams based on the queries mentioned by the subscribers or users in the form of subscriptions. It evaluates a specified logical condition based on event attributes and if the condition is true, then it publishes the relevant event to the destination stream. For example, the event monitoring in purchasing applications monitors the stream of purchase orders to filter the events that meet out the condition Priority!= 'High' and Amount > 5000. It is a simple filter query that performs the matching sequentially over the number of incoming events and then filters the relevant event, which satisfies the condition of a simple query [34]. In order to compare the incoming events with the other events in the same stream or other stream events, the complex filter is constructed where the event filtering is performed based on the computed metric. It achieves high efficiency and scalability due to the pre-processing of a large number of events based on the minimum specifications in the event query.

1.2.8 Uncertainty

Uncertainty is defined as the lack of certainty and incomplete knowledge of the information which leads to a difficult situation to describe the possible outcome. It generates insufficient, imprecise and vague information that leads to various challenges in the decision making of complex systems [35]. Due to the data conflict, the inaccurate results lead to high uncertainty. In various real-time applications such as object tracking in RFID applications and weather monitoring, it is more difficult to capture the uncertainty in the high volume of raw data streams [36]. In order to analyze the uncertainty issues, various approximation techniques are evolved. Uncertainty is broadly classified into four classes as follows.

1.2.8.1 Epistemic Uncertainty

Epistemic uncertainty arises due to the lack of knowledge. It is caused due to the systematic error, incompleteness, subjective uncertainty and measurement

uncertainty. This uncertainty is handled by various mathematical frameworks such as probabilistic and fuzzy approaches. However, it is a significant challenge to select a suitable mathematical framework for representing the uncertainty.

1.2.8.2 Linguistic Uncertainty

Linguistic uncertainty is the uncertainty produced by the statements in natural language. It is caused due to the vagueness, context dependence, ambiguity, specificity and indeterminacy of the theoretical terms. This uncertainty is caused accidentally and therefore, create problems in risk assessment.

1.2.8.3 Ambiguity Uncertainty

Ambiguity uncertainty arises in the case of ambiguous events where a single event may have more than one relevant meaning. This uncertainty leads to high confusion and vagueness due to the lack of information. The probabilistic approaches handle this type of uncertainty by computing the probabilities of the output events in order to determine the relevant events.

1.2.8.4 Variability Uncertainty

Variability uncertainty is caused due to the variations or differences in a process or quantity by nature. This uncertainty is caused by various environment parameters and is difficult to assign an exact value to the events. It leads to high irregularity because of inherent randomness in repeated processes. This uncertainty can be reduced but cannot be eliminated entirely [35].

1.2.9 Uncertainty Analysis Techniques

In order to reduce the potential risk in estimation, the uncertainty analysis is a statistical or mathematical process used to measure, recognize and identify the uncertainty associated with available raw data streams. In order to estimate the risk, various tasks such as derivation of an uncertainty factor, complex model specification, decision making and monitoring methods are performed [37]. It is a

quantitative approach to reduce the risk and estimation of uncertainty, which depends on the complexity of the variation of uncertainty according to the time.

1.2.9.1 Probabilistic Analysis

Probabilistic analysis analyzes the various kinds of uncertainty associated with the noisy and sensor data from unreliable sources such as sensor networks and RFID applications. In order to model the uncertain data, the machine learning technique constructs a rich and complex Probabilistic Graphical Modeling (PGM) [38] to capture the uncertainty from the true state of the physical world. In addition, a novel inference algorithm called as probabilistic inference is used in the constructed probabilistic database model to speed up query processing significantly. To support probabilistic query processing in the presence of uncertainty, confidence score is assigned to the correlated tuples in the probabilistic databases [39]. Furthermore, the probability computation scheme computes the probability of events based on the assigned confidence score. Thus, the computation of data in the query results is directly decoupled with computed probability (confidence) values. Moreover, probabilistic ranking [40] performs ranking on the set of uncertain data based on the computed probability. Thus, the tuple with a high probability is identified as a relevant event in the presence of uncertainty.

1.2.9.2 Fuzzy Analysis

The systematic behavior of vagueness and imprecision is handled by a mathematical framework called as Fuzzy Logic (FL) or Fuzzy set theory, formulated by Lotfi Zadeh in 1965. FL is a more robust analytic tool that performs better reasoning to elicit and to encode the uncertain knowledge in a domain [41]. A fuzzy set provides knowledge representation in terms of uncertainty in a flexible way. It is a most effective technique that relies on the human knowledge base to deal with the complex concepts associated with the uncertain data. In order to estimate the potential risk, the fuzzy rules express the human knowledge with a set of fuzzy values in terms of linguistic terms such as high, low, very low and very high [42]. It performs effective reasoning over the inaccurate data using knowledge models to

take rational decision in the presence of uncertainty. In risk estimation, the fuzzy logic is an effective multi-criteria analysis system to resolve uncertainty that consists of analytical tools to perform decision making. The characteristics of the fuzzy logic system are:

- It deals with the uncertainty and imprecision of reasoning processes.
- It models the heuristic knowledge in terms of mathematical equations.
- It allows the computation of linguistic information.

1.2.9.3 Bayesian Analysis

Bayesian Network (BN) is one of the most important, efficient and elegant models for representing and reasoning with probabilistic models. In order to represent the knowledge about the uncertain domains, a novel statistical and scientific model is constructed to model the complex systems under high uncertainty [43]. It is an augmented, directed acyclic graph G = (V; E) consisting of a set of nodes where each node in the graphical model represents a set of random variables$\{X_1, X_2 \ldots\ldots X_{|V|}\}$ and each edge represents the conditional dependence relationships between the random variables. BN encompasses two components such as (i) a qualitative model and (ii) a quantitative model. The qualitative model encodes the local correlation among the random variables using a direct acyclic graph. The quantitative model represents the joint probability distribution $P(x_1 \ldots x_n)$ over a finite set $\{x_1, \ldots, x_n\}$ of random variables that possess a set of mutually exclusive states. BN can mathematically express one belief as a conceptual model in a more logical way. Furthermore, the Conditional Probability Table (CPT) models the probability distribution of a set of random variables that specifies how a random variable depends probabilistically on the values of its parent nodes Pa (X_i) [44].

1.3 SCOPE OF THE RESEARCH

In recent years, the explosive growth of IT technology and Business Intelligence (BI) techniques pave way for the business enterprise to monitor and to analyze the business process in real-time in order to trigger necessary actions for

providing the right direction for the organization growth in a highly competitive world. In order to react automatically to all events of interest, the business process system must perform effective reasoning in the event composition system to acquire a high level of intelligence from the available data. Therefore, the intelligent system is to integrate CEP technology that timely processes the large amount of information by filtering, aggregating and correlating the data flows from multiple sources. In event based systems, uncertainty is mainly created due to the gap between the real occurrences of events, to which the system must react and the capability of event-driven systems to produce exact events. The main challenge of explicit probabilistic event processing is that the CEP has to process a large number of rules with multiple event sources. Thus, the probability computation under various types of uncertainty is not trivial. In order to perform effective decision making, the probability of derived events is to be correctly quantified using an appropriate mechanism for probability computation. The CEP engines deployed in real-time mission-critical applications must satisfy the demanding performance requirements. Thus, CEP has to support operational requirements in terms of throughput, response time, event patterns and scalability. An efficient, scalable CEP engine is necessary to fulfill the requirements and to meet the challenges of the real-time mission-critical applications.

1.4 CONTRIBUTIONS OF THE RESEARCH

The literature survey reveals that there is not enough scope for Complex Event Processing to achieve high scalability. The PCEP system takes one step forward in this research through implementing event filtering approach in Publisher/Subscriber system to achieve efficient CEP with high scalability and efficiency. This research presents a generic system for representing events and rules over uncertain data. It designs such a system to manage the uncertainty of the events explicitly under multiple rules with multiple event sources. The main contributions are:

- **Event Pattern Matching:** An efficient NFA_h based event matching algorithm is developed to filter the relevant events which match the

event pattern queries over the large number of incoming events, where Query Aware Partitioning with Predicate based Subscription Grouping algorithm is used to scale up for a large number of queries.

- **Complex Event Detection:** The Probabilistic Event Sequence Prediction phase supports probabilistic inference on complex uncertain events. Probabilistic event hierarchies are constructed in the form of graphical model called as Dynamic Fuzzy Probabilistic Relational models that infer the correlations among the sequences of incoming events.

- **Computation of Probability:** DFPRM model computes the joint probability distribution based on the observation of the correlation between event sequences to enhance the robustness of the event detection process under uncertainty.

- **Probabilistic Fuzzy Logic:** Probabilistic Fuzzy Logic is used to estimate the fuzzy linguistic variables from computed conditional probability distributions in the large probability space. It formulates the combination of the relevant event sequences according to the computed probability of events with the reduced overhead.

- **Heterogeneous domains:** The PCEP system is validated with three diverse domains to ensure that its performance is consistent.

1.5 THESIS ORGANIZATION

The objectives of this research are to study the existing CEP engines and to improve the performance CEP engines with respect to efficiency and scalability. The background information regarding Complex Event Processing is presented in the first chapter. The rest of this thesis is organized as follows:

Chapter 2 presents a critical survey of literature. A detailed description of event filtering schemes and the main features of popular CEP engines, including its time model, syntax, processing model and query languages are also highlighted. It

also explains the complex event languages, complex event detection, continuous queries and production systems. It also presents the probabilistic approaches to process the complex events in active database systems under uncertainty. The limitations of the existing systems are inferred from the literature.

Chapter 3 presents the formal problem statement of research problem with the objectives. The high level conceptual architecture of the PCEP system is presented. The information flow from the primitive events to the complex event through the various modules in the PCEP system is explained.

Chapter 4 proposes an NFA event filtering based on the design of efficient Publisher/Subscriber middleware CEP system. The performance of this filtering approach is evaluated in terms of average processing time and throughput.

Chapter 5 proposes Probability Fuzzy model to derive the most probable events using Dynamic Probabilistic Fuzzy Relational model. The performance evaluation of the PCEP system is highlighted in terms of processing time, scalability and efficiency.

Chapter 6 highlights the implementation of PCEP in three motivating application scenarios such as RFID Monitoring, trading in stock exchange and KPI based business activity monitoring in inter-organizational multiple domains.

Chapter 7 concludes the findings of this research and explores the possible directions for future work to optimize the Probabilistic Complex Event Processing under uncertainty.

CHAPTER 2

CRITICAL SURVEY OF LITERATURE

This chapter describes a detailed survey of the existing research approaches as well as ongoing research in the field of CEP. It discusses about the existing approaches of the event filtering, CEP engines, query languages and probabilistic database systems in detail. It paves the way to fulfill the research gap in uncertain CEP to achieve scalability and efficiency. Finally the limitations in the existing research approaches are obtained.

2.1 REVIEW ON EVENT FILTERING SCHEMES

This section discusses about the existing event filtering approaches to filter the relevant events from the large number of incoming event streams based on the event queries subscribed by users.

2.1.1 Binary Decision Diagram

In this approach, Binary Decision Diagram (BDD) performs an event filtering that is implemented on a large scale content based Publisher/Subscriber middleware architecture. BDD is a data structure that represents the Boolean function for model verification. In order to process the large number of incoming events, an efficient and scalable event filtering engine is performed based on user subscriptions. Here, the subscription query languages such as Simple Subscription Language (SiSL), Strict Subscription Language (StSL) and Default Subscription Language (DeSL) express the user subscriptions more effectively [45]. In order to enhance the performance of BDD in event filtering, this approach carries out three optimizations such as BDD restriction, BDD variable ordering and the BDD evaluation algorithm. It supports a high level of semantics to perform an event matching a query or subscription to handle a half million subscriptions efficiently. The main drawback of this approach is that the filtering operation is not distributed;

thus it leads to high operation overhead. Therefore, the filtering engine is not able to process at the arrival rate of incoming messages that results in high processing time.

2.1.2 Bitmap Indexing

This approach deploys a novel event processing scheme to process the large scale complex events efficiently based on Bitmap Indexing technique. It proposes a new effective technique to detect the complex events that satisfy the minimum conditions in query specifications. In order to reduce the unnecessary resource utilization perfectly for storage and operation overhead, this approach eliminates unnecessary operations over incoming events. Thus, this approach identifies the sequence of relevant complex events with minimal resource consumption. This approach performs an effective pattern matching in two phases such as threshold phase and detection phase using query index and bitmap [46]. Furthermore, a tree structure is exploited to organize query index and manages the primitive events. Thus, the constructed tree structure is used to check whether an incoming event satisfies the minimum conditions required for the complex events in the first phase. If the incoming event satisfies the minimum conditions in the threshold phase, then the second phase will be invoked. In the detection phase, the bitmap structure with query index performs the complex event detection.

2.1.3 High Performance Event Filtering

This approach motivates to develop a scalable and a high performance event filtering mechanism for enterprise-wide Distributed Dynamic Multi-Point (DDMP) applications. It is a data reduction mechanism that minimizes the unnecessary operation overhead and network traffic in order to monitor, detect and deliver the events to interested consumers. In expressive event language models, the events and the user subscriptions are internally represented in the form of Deterministic Finite state Automata (DFA) [47]. Here, the event filtering is performed in an object oriented framework using filter programming interface. Furthermore, an effective monitoring and feedback mechanism are used to span multiple filtering servers between the producers and consumers in the local area or

wide area networks. Two filtering servers such as front-end filtering servers and composite filtering servers are deployed in this approach to perform effective filtering. Here, the front end filtering server is used to classify the primitive events whereas the composite filtering servers detect the composite events. Some novel ideas are integrated with this approach to develop a scalable, configurable and high performance event filtering mechanism in DDMP domain. The main problem is that this approach is a domain dependent application that focuses on the event filtering in DDMP domain. This approach is not suitable for all the other domains in a business enterprise system.

2.2 REVIEW ON CEP ENGINES

This section highlights in detail the discussions about some of the CEP engines that made significant efforts to perform efficient, scalable and fault tolerant event processing. The CEP engines available in the market are analyzed with their characteristics, advantages and disadvantages It describes the significant differences among the CEP engine in terms of rule language, processing model, data structure and their overall system architecture.

2.2.1 Simple Scalable Streaming System

Simple Scalable Streaming System (S4) is a general-purpose, scalable, fault tolerant, free stream processing platform created and released by Yahoo! This is a framework for “processing continuous, unbounded streams of data” [48]. It can perform distributed computation over constantly changing data. It shares many characteristics with IBM’s Stream Processing Core (SPC) middleware, but S4 differs from SPC only in architectural design. The SPC design derives from a subscription model whereas the S4 design derives from the combination of both MapReduce and the Actor model [49]. Stream processing is carried out as in an Actor model where graphs connect the large number of nodes consisting of Processing Elements (PEs). In this mode of interaction, each and every PE has the capability to consume and to emit data events with the other PEs through I/O queues. Thus, the messages are transmitted among between PEs in the form of data

events, which consist of named streams with their corresponding keys and attributes. The PE can process the data events from the streams with the only one specified key. This framework can effectively route the events to the appropriate PEs. This process leads to high level of semantic encapsulation and location transparency. Thus, the application developers can process data streams through a simple programming interface which offers an impressive level of simplicity for developing massively concurrent applications due to its symmetric nature where all nodes in the cluster are identical under decentralized control. It minimizes the access latency and eliminates I/O bottlenecks due to the accumulation of local memory in the PEs. Easily deployable and pluggable architecture provides a generic and customizable design. It provides high flexibility in the design maintenance because of decentralized and symmetric architecture. During handoff, the lossy failover occurs that automatically moves a process into standby mode and leaks a large amount of data events. It has inability to express the queries that span multiple input events thus making it as unsuitable for CEP.

2.2.2 Aurora

The Aurora is a general purpose, Data Stream Management System (DSMS) to provide real-time monitoring applications, developed by the Brandeis University, the Brown University and the Massachusetts Institute of Technology. It provides a new imperative language called as Aurora's Stream Query Algebra (SQuAL) that defines the transforming rules in a graphical representation by adopting a boxes-and-arrows paradigm [50]. This paradigm connects the different operators explicitly and shares the computation of different standing queries. It constructs the network with a set of operators that consist of connection points. The connection points are intermediate tables in the network upon which ad hoc queries can be executed. In the work flow process, the data events are forwarded in the form of tuples along the pathways (arrows). In order to monitor the performance of the system, the Aurora defines the Quality-of-Service (QoS) specifications for individual queries to perform several optimizations to enhance the performance or quality of the results. Among that, the load shedding is the most critical optimization where the QoS specification determines how and when to shed load. In a highly

overloaded state, the optimization enables to drop the number of tuples relating to systems that are more tolerant of missing data. The scheduler allocates resources based on the load of the operator and the optimized user defined plan, which is specified in QoS constraints [51]. It maximizes the overall QoS from the applications. It provides an intermediate storage inside the query plan to recover the operator's failure. Due to the lack of semantics, it is difficult to prove the correctness of the query formulations.

2.2.3 TelegraphCQ

In order to support the highly streaming adaptive flow of data over event streams, the University of California at Berkeley has developed a Telegraph Project (TelegraphCQ). It provides event processing capabilities in the relational database management by implementing the PostgreSQL27 [52]. In this approach, an open source database PostgreSQL27 modifies its existing architecture to process the continuous queries over the streaming data. It is a distributed, continuously adaptive parallel cluster-based processing that can process the continuous queries over the large number of incoming event streams. TelegraphCQ allows the access of the previously-arrived data with high intermittent connectivity [53]. It integrates efficient data management components to manage adaptively the dynamic nature of data availability. This approach supports an efficient event processing rather than the other conventional event processing due to its efficient adaptivity. It provides an efficient resource scheduling for groups of queries and also supports dynamic QoS. The main drawback is the complexity of computation.

2.2.4 STREAM

Standard sTREam datA Manager (STREAM) is the distributed Data Stream Management System built at Stanford University. STREAM can adaptively process the large class of declarative continuous queries over the continuously arriving data streams in real-time [28]. It declaratively expresses user subscriptions in the form of Continuous Query Language (CQL) [54] that is extended from the SQL language. CQL handles continuous queries with sliding window facilities. CQL

consists of two layers such as an abstract semantics layer and an implementation layer. The query language translates the flexible query plans that support effective optimization and fine grained scheduling decisions. It supports effective load shedding to manage the dynamically varying load over time and also manipulates query plans during execution. The main focus of this approach is effectively running the various types of queries by considering the cost of self-maintenance of different materialized views in a bounded amount of memory [55]. Effective memory management and query plan based execution are integrated to achieve stream processing. In order to reduce resource consumption, the operations such as ordering, clustering and referential integrity are implemented on the event streams. Thus, the approximate query answering is deployed to manage the resource consumption in the context of limited resources. Thus, it leads to less performance with inaccurate results due to approximation.

2.2.5 Esper

Esper is an open source and sophisticated CEP suite developed by the Espertech that integrates both Event Streaming and Event Analysis into a single framework. Esper uses an integrated part of software or as a standalone server. It defines a rich declarative language for rule specification called as Event Processing Language that includes all operators in SQL. Further, ad-hoc constructs for sliding window definition and interaction are provided [56]. Thus, Esper provides a powerful mechanism to integrate temporal relations of events using sliding event windows. It allows expressing the complex matching conditions to combine different event streams, filtering and sorting them. It expresses event patterns using nested constructs that include conjunctions, disjunctions, negations, sequences and iterations. Esper supports both centralized and clustered deployments for query processing. Further, it detects sequences and patterns of unrelated events. In order to achieve load balancing, Qos policies are maintained effectively to integrate the processing power of different and well-connected nodes that increase the system's availability [57]. Esper fails to consider the uncertain data which occurs in practical real-time streams.

2.2.6 Stream-based and Shared Event Processing (SASE)

In order to design an efficient and robust RFID stream processing system, UC Berkeley and the University of Massachusetts Amherst developed the SASE research project. SASE is one of the most influential efficient CEP systems modeled as a data flow paradigm that transforms real-time data streams into appropriate actionable information. Furthermore, a new abstraction of the Complex Event Processor is deployed to process the continuously arriving incoming event streams in a timely manner [58]. In order to meet the challenges including the data-information mismatch, incomplete and noisy data in RFID-enabled real-time monitoring applications, SASE defines a new query language by extending the existing event language. In order to model the event language using native sequence operators, Non-deterministic Finite Automata (NFA) is constructed that allows effective pipelining to predict the event sequences based on subsequent operators such as selection, window and negation due to the efficient implementation. SASE is a comprehensive system that performs effective filtering, pattern matching and an aggregation mechanism in order to filter and process the real-time data streams in a timely manner [59]. The query language supports high extensibility to develop a new event language for RFID enabled applications. SASE also possesses high flexibility in query execution that achieves several optimizations. It fails to consider the uncertain data, which occurs in practical real-time streams and also does not address the distributed system issues. It will lead to a poor performance with high access latency because of logic complexity to manage a large volume of data.

2.2.7 Cayuga

Cayuga is a most efficient and commonly used large scale CEP system that supports on-line detection of complex patterns in event streams based on a large number of concurrent subscriptions [60]. The core components of the Cayuga include a query processing engine, an index component, a meta data manager and a memory manager. In order to support stateful subscriptions, Cayuga leverages the traditional Publication/Subscription techniques. The users express their interest as more expressive and structured Cayuga Event Language (CEL). The Cayuga query

translates into Non-deterministic Finite Automata (NFA) and then loads into the query processing engine which processes the incoming streams against queries. This automaton reads a finite sequence of events over a finite relational schema for a specific time interval. In order to process a large number of arbitrary queries, this engine deploys two techniques such as row/column scaling and pipelining to distribute the queries among the large number of machines. The Query Engine manages the state transitions of NFA using predicates. Further, the custom heap memory management is deployed to store the automaton instances that satisfy the predicates and also indexes the operator predicates to improve the performance of the system [61]. This approach deploys a novel Multi-Query Optimization (MQO) technique to achieve high scalability and high-speed event processing. In order to process the large number of concurrent subscriptions, this engine contains efficient in-memory processing that facilitates to achieve high scalability and high speed event processing. This approach does not support automated query rewriting and distributed detection. This leads to difficulty in distributed query processing among the multiple machines in the system.

2.2.8 Coral8 Engine

The Coral8 processes multiple heterogeneous data streams and performs various operations such as filtering, aggregation, correlation, pattern matching over the incoming data streams in real-time. This engine supports effective pattern matching and native XML processing. Therefore, Coral8 is implemented in more powerful CEP applications to execute a real-time enterprise business function [62]. It encompasses the familiar lightweight streaming architecture to implement faster and easier deployment that can process more than millions of events per second for simple queries and thousands of events per second for complex queries with low latency. The Coral8 consists of two main components such as Coral8 Server and the Coral8 Studio [63]. The Coral8 Server is the core of the Coral8 engine that provides clustering support. To monitor the performance and activity of the server, various features such as publication of a status data stream is included with this coral8 engine. Furthermore, the Coral8 Studio provides an IDE-like interface that allows administrators to add or remove queries according to the input and output data

streams. In order to express the event queries, this engine develops a subscription language called Continuous Computational Language (CCL). In order to execute continuous queries in parallel for high-speed data, enterprise-class clustering is configured with this engine. Furthermore, parallel and asynchronous database integration is performed to achieve high performance event storage.

2.3 SURVEY OF QUERY LANGUAGES

This section describes the details of the capabilities and the features of several representative complex event query languages to express the event queries in the CEP systems. The main purpose of this discussion is to determine where extension is required for incorporating the uncertainty in the existing query languages. This survey discusses the event query languages in terms of types of complex events detected, the structure and the semantics of the query languages and their relative advantages and their disadvantages.

2.3.1 Continuous Query Language in STREAM

STREAM introduces the Continuous Query Language (CQL). CQL exploits the rich expressive power of SQL [28]. It consists of three public operators such as stream-to-stream, relation-to-stream and relation-to-relation operators. Among that, relation-to-relation operator is a subset of the SQL language and the remaining two operators are newly extended to make suitable for a large number event streams. It provides well-defined semantics that can be implemented based on any language with the help of a composite operator and data type rules. Figure 3.1 represents the CQL as follows:

OUTPUT [var_1,var_2,var_3....var_n];
SELECT [attributes | (sliding-window) aggregates] +
FROM [stream-name [RANGE BY t_1 SLIDE BY t_2] | relation-name] +
WHERE <selection and join predicates>
GROUP BY <grouping attributes>
HAVING <group selection predicates>

Figure 2.1: Continuous Query Language

The CQL language consists of six clauses such as OUTPUT, SELECT, FROM, WHERE, GROUPBY and HAVING. Among that, the OUTPUT clause consists of an attribute name of the number of variables [var_1, var_2, var_3……var_n] in the variable list. SELECT and FROM clauses are mandatory in CQL language and it mainly relies on the input stream. Further, the WHERE clause selects the event from the input stream that satisfies the selection predicates in it. The GROUP BY clause contains one or more attributes used to group the events in the input stream. Finally, the HAVING clause selects the events based on the group selection predicates in it [55]. This language is like a research prototype. So, it is not applicable in the case of a Complex Event System. While it has a manual, it cannot be used for real-world examples because of the lack in documentation.

2.3.2 Continuous Computation Language (CCL) in Coral8

CCL was the first commercial industry standard declarative language. CCL provides a massive head start for creating the CEP applications in the real world. It has the capability to process the continuously arriving dynamic data. Like SQL, it achieves standard event selection using SELECT/ WHERE clauses that are used to filter the events from the incoming event stream [63]. CCL provides additional capabilities such as sliding windows, event matching and output timing controls for manipulating data during real-time continuous processing. It is represented as follows:

```
SELECT <attributes from input stream>
FROM < sliding window expression>
WHERE <predicates>
HAVING <group selection attributes>
GROUP BY <common attributes>
INSERT <output stream>
```

Figure 2.2: Continuous Computation Language

2.3.3 SASE Language

SASE language is a declarative, composition-operator-based language mainly suitable for high-performance querying of event streams. It has a high level structure as SQL query, but the design of the language is focused on the event pattern matching. The structure of the SASE language is represented as follows:

```
FROM <input stream >
EVENT <event pattern>
WHERE <condition >
WITHIN <sliding window>
RETURN <return event pattern>
```

Figure 2.3: SASE Language

In SASE, the FROM clause specifies the name of an input stream. The event matching consists of three mandatory clauses such as EVENT, WHERE and WITHIN to transform an input stream into a stream of composite events [58]. The EVENT Clause specifies the pattern that is to be matched against the input event stream. Further, the WHERE and the WITHIN clauses specify the value based conditional expressions and the occurrence time constraints respectively. Finally, the RETURN clause converts the stream of composite events as a final output.

2.3.4 Event Processing Language in Esper

Esper presented a hybrid Event Processing Language (EPL), which combines the features of continuous query in data stream languages and the pattern matching constructs in a composite operator based language. This language expresses the event patterns using composite operators like Rapide in composite event language. Further, EPL processes the input streams against event patterns using data stream constructs like CQL data stream language as given in Figure 2.4.

```
INSERT <insert_into_def>
SELECT <select_list>
FROM <stream_def[as name] >
WHERE <search_conditions>
GROUP BY <grouping_expression_list>
HAVING <grouping_search_conditions>
OUTPUT <output_specification>
ORDER BY <order_by_expression_list>
LIMIT <num_rows>
```

Figure 2.4: Event Processing Language

In EPL, two new clauses such as ORDER BY and LIMIT are introduced to provide more expressive query specifications [57]. The LIMIT clause limits the lifetime of pattern instance if the incoming input stream consists of multiple events. Furthermore, EPL supports the ORDER BY clause to determine the order of events based on its timestamp. This language mainly concentrates on the partition of the functionality of event detection, but does not provide much for the selection and the collection of events from the incoming event streams.

2.3.5 Cayuga Event Language in Cayuga

Cayuga is the most popular and highly expressive CEP engine running with a large number of queries which are expressed in the form of Cayuga Event Language (CEL) [60]. This language offers pattern queries over event streams based on Cayuga Algebra in the form of regular expression. CEL is represented as follows

```
SELECT <attributes>
FROM <stream_expression>
PUBLISH <output_stream>
```

Figure 2.5: Cayuga Event Language

In CEL, the SELECT clause is optional that specifies the name of the attributes in the output schema. Furthermore, the FROM clause is the core of the query which composes of one unary construct: FILTER and two binary constructs: NEXT and FOLD to specify the stream expression. The PUBLISH clause is also optional that provides the name of the output stream. If the PUBLISH clause is omitted, then the output stream is unnamed.

2.3.6 AMiT in IBM Websphere

IBM Active Middleware Technology (AMiT) proposes a XML language that allows the specification and the detection of the complex events, which are referred as situations [64]. Each situation has an associated lifespan that acts as a context for the event detection. The <event> tags declare the attributes of the events and then <lifespan> tags consist of two events such as an initiator and a terminator event. This language provides various parameters to correlate the termination type and the quantifier in order to provide more fine-grained control over lifespan initiation and termination. The main disadvantage of AMiT is that it does not allow the nested operators to be combined into number of single events to form a complex event. AMiT is represented as follows

```
<event name ="..." >
< eventAttribute name ="..."
        type ="..."  >
< event >
<lifespan name ="..." >
<initiator >... < initiator >
<terminator >... <  terminator >
<  lifespan >
<situation name ="..."
        lifespan ="..." >
<operator >
<  operator >
< situationAttribute name ="..."
        type ="..."  >
<  situation >
```

Figure 2.6: AMiT Language

2.3.7 ruleCore Markup Language

In order to implement Event Condition Action (ECA) rules in active databases, a high expressive XML based extendable language called as ruleCore Markup Language (rCML) is evolved. The ruleCore focuses on the composite event detection. It defines about the event patterns, situations, complex events or derived events. The ruleCore Markup Language is represented as follows:

ON <event_defintions >

CONDITION <event pattern>

ACTION<action_definitions>

Figure 2.7: rCML Language

In rCML, the ON clause specifies the definition of incoming event stream primitive events, the CONDITION clause is a set of predicates to detect the event's patterns. Further, the ACTION clause specifies the action to be invoked after an event pattern is detected [65]. The main advantage is its reusability because a single block definition may be used to specify the multiple rules with slight modifications. However, there is no sufficient documentation to generate the events depending upon the triggered action.

2.3.8 Drools Rule Language

Drools or JBoss rules provide the rule specification for a business rule management system. This language is extended from the domain specific language. The non-technical staff who are not having sound knowledge can also write the queries. There is no standard syntax to write these types of queries. Generally, rule language supports the different form of syntax according to the specific requirement in the production. The production engine can process the Drools language. Therefore, it extends the domain specific language and gives input for the production engine to process the incoming events. The domain specific language is represented as follows:

```
WHEN<event_patterns,conditions>
...
THEN<action_conditions>
...
```

Figure 2.8: Drools Language

In Figure 2.8, the WHEN clause consists of a list of fact patterns represented in the form of conditions [66]. The THEN clause triggers the specific action if only if the condition in the WHEN clause is satisfied.

2.3.9 Comparative Analysis of Event Query Languages

Having described the strengths and weaknesses of the four categories of the language styles, this section summarizes the comparison of event query languages.

- Composition operators based language offers a compact and intuitive way to specify the complex event pattern queries. Thus, it is attractive in business scenarios to define the event pattern queries in a real-time. These types of languages support event instance selection and consumption which are not applicable in other type of languages. More commonly, operator based languages concentrates on the efficient creation of languages, but not on how to process the event queries over the large number of incoming streams. Further, the aggregation of the attributes from the event data is often neglected in these types of languages.

- SQL-based data stream query languages are the most successful approach and efficient and scalable for commercial industries. Stream-oriented languages provide considerable support to aggregate the event data which is particularly necessary for trading in the financial market. On the contrary, deriving the negation and temporal relationship between the events is mostly cumbersome in these types

of languages. Further, streams-to-relation conversation and vice versa is supposed to be unnatural in the case of discrete time axis.

- Production rules are extremely flexible and easy to represent because they are well integrated with the existing domain specific languages of certain domain applications. Rule specifications are represented in the form of Condition-Action (CA) rules where certain actions are to be executed when specific conditions are satisfied by the incoming events. They are particularly useful for business applications such as logistics, RFID tracking and Business Activity Monitoring. Production rule languages are considered to be less efficient than the other data stream query languages and suitable for the low abstraction level in a primary state because it is hard to express the aggregation and the negation between the events.

2.4 PROBABILISTIC DATABASE SYSTEMS

The aforementioned CEP systems perform the event processing based on the assumption that the data is precise. In imprecise data environment, the modeling of the uncertainty in the form of probabilistic events rather than deterministic is essential, especially for RFID based mission-critical deployments and applications. Probabilistic database systems construct probabilistic models to capture and to process the incomplete or imprecise data appearing in real-time applications.

2.4.1 Hidden Markov Model

Laher is an efficient uncertainty CEP system that processes the event streams from the uncertain or probabilistic database. In order to deal with imprecise data, this approach constructs a temporal graphical model called as Hidden Markov Model (HMM) from the uncertain data [67]. Further, a set of optimized algorithms are presented to process the regular, extended, safe and general queries over probabilistic event streams. This approach processes the order of data more efficiently than a naive approach based on sampling. The probabilistic inference is performed on the constructed HMM to infer a hidden state based on a sequence of

observations on RFID data streams. Further, a query model is designed for the complex queries using Cayuga Event Language with the detection operators. This probability computation is accurate and distributed than the naive approach. However, this approach does not support the process to perform pattern matching between the query model and data model.

2.4.2 Top-k Query Processing

A top-k query processing is proposed to process the imprecise data in uncertain databases [68] that facilitates to achieve efficient information retrieval over the imprecise data. This approach extends the traditional Top-k query semantic to manage the uncertain database settings. Due to this query answering semantics capability, this approach provides an efficient query processing in the context of uncertain and probabilistic databases. It presents two algorithms such as Uncertain top-k query (U-top-k) and Uncertain k-Ranks query (U-k Rank query) that extends the semantics of top-k queries. In order to determine the number of tuples in a state, the graphical model is constructed with the set of states according to the probability associated with the set of tuples in the probabilistic database. However, this construction is possible for a small database. Therefore, in the case of large databases, it is difficult to construct the graph for all the tuples in the database. The generated graph exponentially increases in accordance to both space and time. On the other hand, top-k approach performs query searching through all possible states using the arbitrary correlation in the complete model and also leading to a large search scope with exponentially increasing storage space.

2.4.3 Top-k Query Processing in X-Relation Model

The existing top-k query processing approach can process the independent tuples in the set of probabilistic data against the top-k queries. However, this approach only processes the dependent tuples related to X-relation model. In order to overcome the problems of the existing approaches, a novel effective polynomial algorithm is proposed for processing top-k queries in uncertain databases based on the adopted X- relation model [69]. This approach adopts an x-relation model to

process the queries that limit the arbitrary correlation among the tuples in the uncertain database. It consists of 'n' number of x-tuples and then each x-tuple randomly instantiate more than one number of tuple. Therefore, a novel dynamic programming algorithm can process the U-k Rank queries with less runtime and storage overhead in two modes of operation such as a single-alternative case and multi-alternative clause. This approach processes the U-top-k queries and U-k Rank queries that are significantly faster and exploits the minimum memory space under the X-relation model. It provides the solution for the uncertain database but not for the uncertain data streams continuously arriving from unreliable event sources [69]. In this approach, a effective linear query processing is performed with less polynomial time. The proposed U-top-k queries and U-k Rank queries achieve significantly high processing speed and less memory consumption under the x-relational model of tuples in uncertain databases.

2.4.4 Efficient Top-k Query Evaluation

In order to process the queries effectively in a probabilistic or uncertain database, an efficient query evaluation framework is proposed [70]. Due to the imprecise data, top-k query processing generates a large number of query results with low quality. The query result with high probability is considered as a most suitable answer for the queries. However, this leads to imprecise results because of the probability computation based on approximation techniques. Therefore, an efficient query evaluation framework is proposed that provides the most optimal algorithm to determine the most probable top-k answers. This work shifted the focus from the probabilities to the confidence score of each tuple to perform ranking. In order to determine the most suitable query answer, the ranking is performed to rank the query results based on the confidence score of the resulting tuples in the probabilistic database. Hence, the query results with a high confidence score is returned as the most probable top-k answer of the corresponding query [71]. It focuses on the confidence score based ranking, but does not compute the exact probability score to determine the most suitable query results at high quality.

2.4.5 Probabilistic Complex Event Triggering

This approach provides a Probabilistic Complex Event Triggering (PCET) to perform effective probabilistic reasoning on the imprecise data in the sensor environment. PCET provides an event architecture processing system that attempts to resolve the problem of robust event detection. It is triggered under high noisy sensor readings. In order to construct the probabilistic event hierarchies of higher-level events, a complex event language is developed [72]. A Bayesian Network is constructed to support an inference over the underlying sensor readings. This approach deploys a probabilistic inference mechanism that performs a probabilistic reasoning over uncertainty through representing the conditional dependencies between the uncertain events modeled in the form of Bayesian Network. It infers and reasons about the probabilities of triggered events for taking finer-grain decisions according to the event occurrences. Thus, the efficiency of the complex event detection is improved even for an inherent uncertain data stream.

2.4.6 Probabilistic Inference over RFID Streams

Due to the inherent reader mobility, high noise and incomplete data in RFID streams, it is difficult to perform stream processing and monitoring applications. This approach is a cleaning process to translate the incomplete raw data streams from mobile RFID readers into precise event streams with location information. Thus, an effective data cleaning and transformation are performed to obtain the required data for query processing [73]. Furthermore, a novel mechanism is proposed to perform the probabilistic inference over RFID streams for acquiring information from the imprecise data. In order to infer the information from the raw data streams, the probabilistic approach is modeled to capture information from the imprecise readings even under the high dynamic mobility of the RFID reader and the object. In the probabilistic model, partial filtering based on the sampling technique which Event Refinement module, CEP has the capability to infer the precise information about the location of the object. Further, partial filtering mechanism enhances the process of the high volume of streams of a large number of objects. In order to perform enhancement, three advanced techniques such as particle factorization,

belief compression and spatial indexing are used for extension in a partial filtering mechanism. It maintains a high inference accuracy and high scalable cleaning, efficient and transformation of mobile RFID data streams with high precision.

2.4.7 Probabilistic Complex Event Processing

This approach focuses on CEP in the context of the real-world event sources that generate streaming data and fuzzy or probabilistic data. The PCEP [74] is used to process the imprecise data from real world sources. It must reason about the events in scenarios where low level RFID events cannot be monitored in a crisp fashion. A new probabilistic model is constructed to model and to reason about the uncertainty nature of event streams. In order to overcome the limitations in time-based or tuple-based windows, this approach presents the concept of semantic windows to process the simple events. It presents the notion of semantic windows, which goes beyond time-based or tuple-based windows. This approach encompasses four main modules: Event Refinement Module, CEP Engine Module, State Maintenance Module and Application Programming Interface. Event Refinement Module uses the machine learning technique to refine the primitive events based on the past information. CEP Engine Module processes the multiple event streams to filter, correlate and aggregate them into semantically high level composite events. After that, the State Maintenance module maintains the state of the current event which is necessary in the case of fuzzy or probabilistic data. An inference based Application Programming Interface (API) is deployed on the top of the PCEP framework to access the current probability state of events. It focuses on the context-aware ubiquitous application for the smart home and does not consider the instantiation in multiple real-time applications scenarios.

2.4.8 Probabilistic Query Evaluation

This is a novel approach which provides a probabilistic query evaluation over uncertain data. This approach broadly classifies the queries into a set of categories over uncertain data based on a flexible model of uncertainty [75]. The probabilistic queries are classified in the aspect of two dimensions such as

aggregate/non-aggregate queries and entity-based/value-based queries. Furthermore, a new technique is also provided to evaluate the probabilistic queries and also to carry out several optimizations out to enhance the performance of query evaluation over uncertain data. After the classification of queries into a set of classes, the query evaluation has developed new algorithms to determine the typical queries and their corresponding probabilistic answers for each classified query class. This approach quantifies the novel metrics for the query evaluation for computing the quality of the answers. In the resource constraint environment, several data update policies or heuristics are provided to improve the quality of probabilistic queries.

2.4.9 Probabilistic Event Extraction System

In order to overcome the limitations of deterministic event detection, this approach proposes a probabilistic model to enable complex event extraction in the face of uncertainty. This approach implements a Probabilistic Event Extractor, a middleware layer on top of a relational database management system to leverage its feature suitable for RFID stream processing. Probabilistic Event Extraction System **(**PEEX) approach can effectively derive the meaningful and probabilistic high level events from the imprecise and erroneous low-level RFID data [76]. In order to manage the imprecise data, PEEX deploys a probabilistic framework to process the inherent ambiguity in the event extraction. Further, a new expressive and declarative query language called as PEEXL is provided to define the composite high level probabilistic events from the low level primitive events. In order to handle the ambiguity and the reliability issues, the probability of low level RFID events is extracted using confidence tables [77]. Furthermore, the probability of the composite high level event is determined from the confidence score of the underlying lower level RFID data. This type of probability based complex event detection acquires high detection rates compared with the deterministic detection approaches. PEEX approach focuses on the composite event extraction. However, it fails to handle the query execution over the detected events.

2.4.10 Probabilistic Event Stream Processing with Lineage

This approach provides an effective probabilistic framework for query processing over probabilistic event streams. An Active Instance Graph, a data structure that constructs a sequence event processor to detect the probabilistic event streams [78] is proposed. This approach deploys an NFA to maintain a record for the set of active states of an unbounded probabilistic event sequence. A more expressive query language is designed to express Kleene closure patterns that support probabilistic queries for composite event stream matching in the physical world. Further, a new probabilistic data model is constructed to compute the confidence score for each detected sequence pattern based on their lineage in order to trigger the confidence computation using NFA. It is a decoupled framework that performs the query processing in two steps by dividing the pattern matching and the probability computation. The pattern matching performs matching over the probabilistic event streams based on the specified sequence pattern. Thus, a matched pattern sequence is derived as output probabilistic events. The probability of the output probabilistic events is computed based on their lineage for matched patterns. It performs effective complex event detection over the other existing naive approaches.

2.5 LIMITATIONS OF THE EXISTING SYSTEMS

In the case of uncertainty, CEP is an extremely challenging task that transforms the real-time data in the physical environment into useful information suitable for the end user applications. In recent times, many researchers have shifted their interest in CEP framework and proposed various CEP approaches to perform event processing under uncertainty. However, most of the existing approaches provide poor performance with low scalability and less efficiency under uncertainty. Therefore, it is a still ongoing effort to develop a suitable CEP framework to predict the uncertainty associated with the large number of continuously arriving events. The limitations in the existing approaches are:

i) Limited Speed

The existing systems fail to deliver the expected speed to timely process the continuously arriving incoming events to satisfy the mission-critical application. However, timely processing is a crucial characteristic of CEP in real-time applications. Therefore, the inability to process the incoming events in a timely manner makes CEP unsuitable for a real-time business process.

ii) Limited Expressive Power

The existing systems provide query specifications with limited expressive power that only has the capability to process deterministic data from reliable sources. However, real-time applications such as RFID monitoring, click stream analysis may generate inherently unreliable, incomplete or incorrect data, which lead to uncertainty. Uncertainty is caused due to the gap between the actual occurrences of events and the data sources. The data errors and ambiguity are probabilistic rather than deterministic in nature. Therefore, the existing deterministic query languages cannot be able to process the uncertain events effectively from unreliable sources.

iii) Lack of Uncertainty Handling

The existing CEP systems perform event processing only for a deterministic database but not for the uncertain data. There is no graphical or inference modeling approach to capture and to infer the correlation between incomplete and imprecise data appearing in the physical world. It leads to less efficiency in complex event detection with low throughput and high processing time. Thus, uncertainty handling becomes essential.

iii) Low Scalability

In order to improve scalability, the queries must be distributed among the large number of state machines in the system. Some of the existing approaches are capable of performing distributed complex event detection. However, these applications are only in an operator level that can distribute the predicates among the

machines in the system. Nevertheless, the scalability issues are not addressed for the applications that require throughput for the large number of incoming events. In order to achieve high scalability and event detection throughput, it is essential to design an effective CEP system that distributes and executes the large number of event queries simultaneously on separate machines.

2.6 SUMMARY

This chapter discusses the CEP engines that implement different data models, more expressive query languages, complex event detection strategies and several optimizations to perform effective CEP. This discussion considerably facilitates to get a deeper understanding about the implementation of each CEP engine. This is particularly useful to design the best architecture for the proposed probabilistic CEP engine. From the literature survey, it is inferred that the Aurora CEP engine describes a pipelining model that is identified as a suitable model for event passing in order to obtain high event processing. Aurora [50] and stream [28] describes query plan management to improve the efficiency of the system and also to approximate the data in stream management. The most important key feature obtained from the discussion is that the dedicated Cayuga system [60] achieves high throughput, efficiency and scalability through distributed pipelining architecture. However, it does not support uncertainty. In order to process the uncertainty in the data, probabilistic approaches in [72] [74] [76] [78] propose modeling and inference techniques to handle uncertainty. This section discusses the probabilistic approaches in the literature and has obtained the guidelines to model the uncertain data in the form of probabilistic graphical model. The main work of the PCEP system is the extension of CEP engine by incorporating the probabilistic technique to process the event streams under high uncertainty.

CHAPTER 3

PROBLEM STATEMENT AND RESEARCH OBJECTIVES

This chapters presents a high level conceptual view of the PCEP system. A formal problem statement and objectives of the research work is formulated. The chapter explains about the high level architecture of the PCEP system. The architecture of the system depicts the various modules in the proposed PCEP system. The interaction among various modules in the system is given by the information flow diagram.

3.1 PROBLEM STATEMENT

In order to achieve scalability and efficiency, this research proposes a Probabilistic CEP system that meets out the four main challenges in CEP under uncertainty. The PCEP provides a more expressive, rich environment for the progress of event processing applications that may derive stateful composite event sequences by processing thousands of events per second. Accordingly, the focus of this system is to transform the real-time data in the physical environment into useful information suitable for the end user applications. The prominent features of the PCEP system are listed below.

i) Timely Processing

The number of distributed applications increases tremendously. There is a need to process the continuously arriving events from widely distributed sources at unpredictable rate. The prime requirement is to obtain timely responses from complex queries. Hence, the PCEP is a distributed CEP approach that employs an effective pipelining technique to process the large number of continuously arriving events in real-time. This process is to produce the instantaneous response in a timely manner because of effective pipelining in complex event detection to manage the high input rate of complex events.

ii) Handling a High Volume of Events

The existing event processing schemes fail to process the events from the number of multiple heterogeneous sources in real-time applications. In PCEP, the event matching performs event filtering according to the user subscriptions. The PCEP system scales effectively to a large number of events due to an effective event filtering performed on the distributed computing platform using effective query portioning and pipelining technique.

iii) Automated Processing

A large number of research works have been carried out in the field of CEP, where there is no automation to specify the rules correctly as well as the probabilities associated with the results. Therefore, the most promising and vital area of this research is the use of filtering techniques for the automatic generation and the processing of rules. The PCEP provides the solution to this problem through deploying the query clustering to group user subscriptions based on similar predicates in each subscription group. Thus, the PCEP automates the complex event pattern detection according to the domain expert rules.

iv) Handling Uncertainty

The accuracy and performance of event derivation depends on the reliability of data sources. A data source has inherently unreliable data collection method or generates incorrect data leading to uncertainty. CEP in the presence of uncertain events is a challenging task and this problem hampers the accuracy of derived events. The PCEP system efficiently handles the uncertainty associated with the events by modeling the event hierarchy as a probabilistic graphical model. Furthermore, the approximation logic estimates the fuzzy linguistic variables from computed Conditional Probability Distributions in the large probability sample space.

3.2 OBJECTIVES OF THE RESEARCH

To overcome the limitations inferred from the literature survey, the research problem is formulated with the following objectives:

1. To design an effective high performance CEP engine that evaluates uncertain data in order to support effective Business Intelligence.
2. To develop a high speed event processing engine that can scale to handle a large volume of user subscriptions in order to achieve scalability and efficiency.
3. To reduce the complexity of event processing through deploying the query plan based approach that manages and extends the expressiveness of the high volume of user subscriptions.
4. To propose an NFA-heap event matching mechanism to filter the irrelevant events from the large number of incoming events based on the user subscriptions.
5. To construct an event sequence prediction that supports probabilistic inference on complex uncertain events. Probabilistic event hierarchies are constructed in the form of a Dynamic Fuzzy Probabilistic Relational Model that infers the correlations between the sequences of incoming events.

3.3 ARCHITECTURE OF PCEP

In event based enterprise systems, the PCEP extracts higher level knowledge from the large number of incoming complex events over messaging infrastructure from the different external event sources. In order to provide the event notification under uncertainty, the PCEP system is implemented in Publisher/Subscriber middleware that manages a large number stream of incoming complex events (publications) and a more number of queries (subscriptions) [79]. The following Figure 5.1 illustrates the high-level architecture of PCEP in the Publisher/Subscriber middleware system. Here, the publishers and the subscribers connect in a

distributive manner where the publisher advertises input events from the multiple sources using the publish (e) operation, into the PCEP whereas the subscribers express their interests on an event in the form of subscriptions using the subscribe (β) operation.

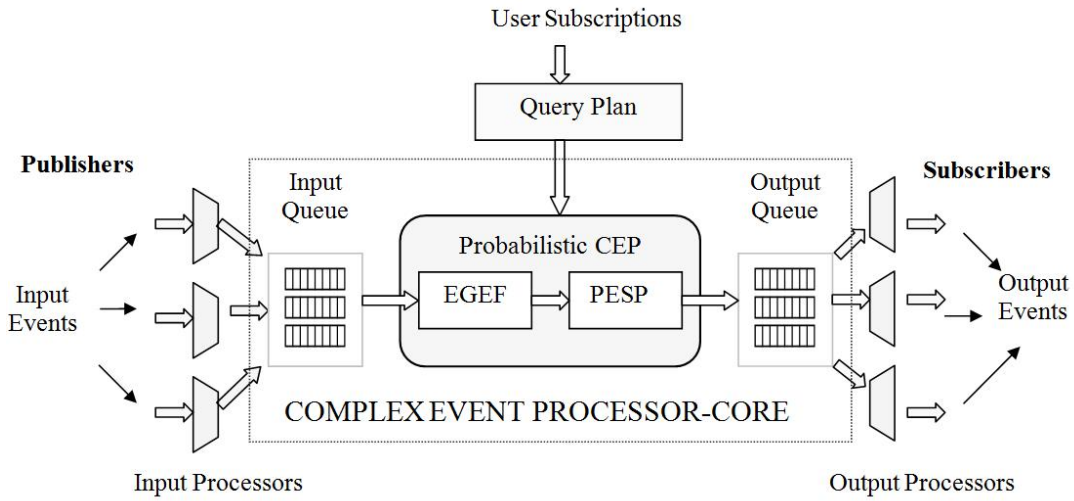

Figure 3.1: PCEP in the Publisher/Subscriber Middleware System

The PCEP architecture composes of three principal components such as i) Input and Output Processors. ii) Query Compilation. iii) Probabilistic Complex Event Processor (PCEPr). The Input processor takes the large number of incoming events from the multiple sources and then converts them into the tuples which are suitable for the internal processing of the core. The Query Compiler converts the access predicate of an each subscription group into a runtime executable automaton and deploys that in the CEP core. The core part of the approach is the Probabilistic CEP that process the events based on the user subscriptions. The CEP processor fetches the events from the incoming queue and processes the events based on the user subscriptions. After deriving the output events by the processor, the derived output events reach the event consumer through the output queues. It allows to manipulate the queries on the fly, such that users can add or remove the subscriptions using the operations such as subscribe (β) and unsubscribe (β) even when the complex event processor is running. The following sub-section highlights the description of the involved components and also explains the internal process and interaction between these components to achieve CEP under uncertainty.

3.4 INFORMATION FLOW IN PCEP

The interaction between various modules in the PCEP system is given in Figure 3.2. The two main modules are Query plan based approach and Probabilistic fuzzy prediction phase.

The publishers execute the publish (e) operation to publish a piece of information in the form of events to the Publisher/Subscriber system. An event 'E' is represented as a tuple (s, t) where 's' is a set of attribute-value pairs as in the relational data model as defined by the schema 'S' and 't' is a sequence of timestamps t = [t_s, t_2, t_3……. t_e] where the first timestamp 't_s' is the start time of an event and the last timestamp 't_e' is the end time. Each valid subscription and the event are associated with a time interval. Therefore, it is considered as valid within that specified time interval.

In the Query plan based approach phase, the event consumers express their interests on an event in the form of subscriptions, which is used to subscribe into a particular category of events within the system. A subscription (β) is expressed in the form of Complex Event Language and composed of set of predicates to filter the relevant events from the large number of incoming events. The subscriptions are grouped using Prediction based subscription grouping. The subscription clusters are deployed using query aware portioning. The input sequence is compiled to a NFA_h automaton using query compilation [80]. A subscriber generates and removes a subscription β from the Publisher/Subscriber system through executing the subscribe (β) and the unsubscribe (β) operations respectively [81]. After receiving the large number of incoming events, CEP processor deserializes and processes the events based on the subscriptions. It takes the responsibility to perform the event matching between the large number of incoming event sequences and to detect event patterns represented in the form of Non deterministic Finite Automata-Heap (NFA_h).

In the Probabilistic fuzzy prediction phase a probabilistic model is built. A probabilistic fuzzy logic inference engine handles the uncertainty in the relevant events and detects the complex events.

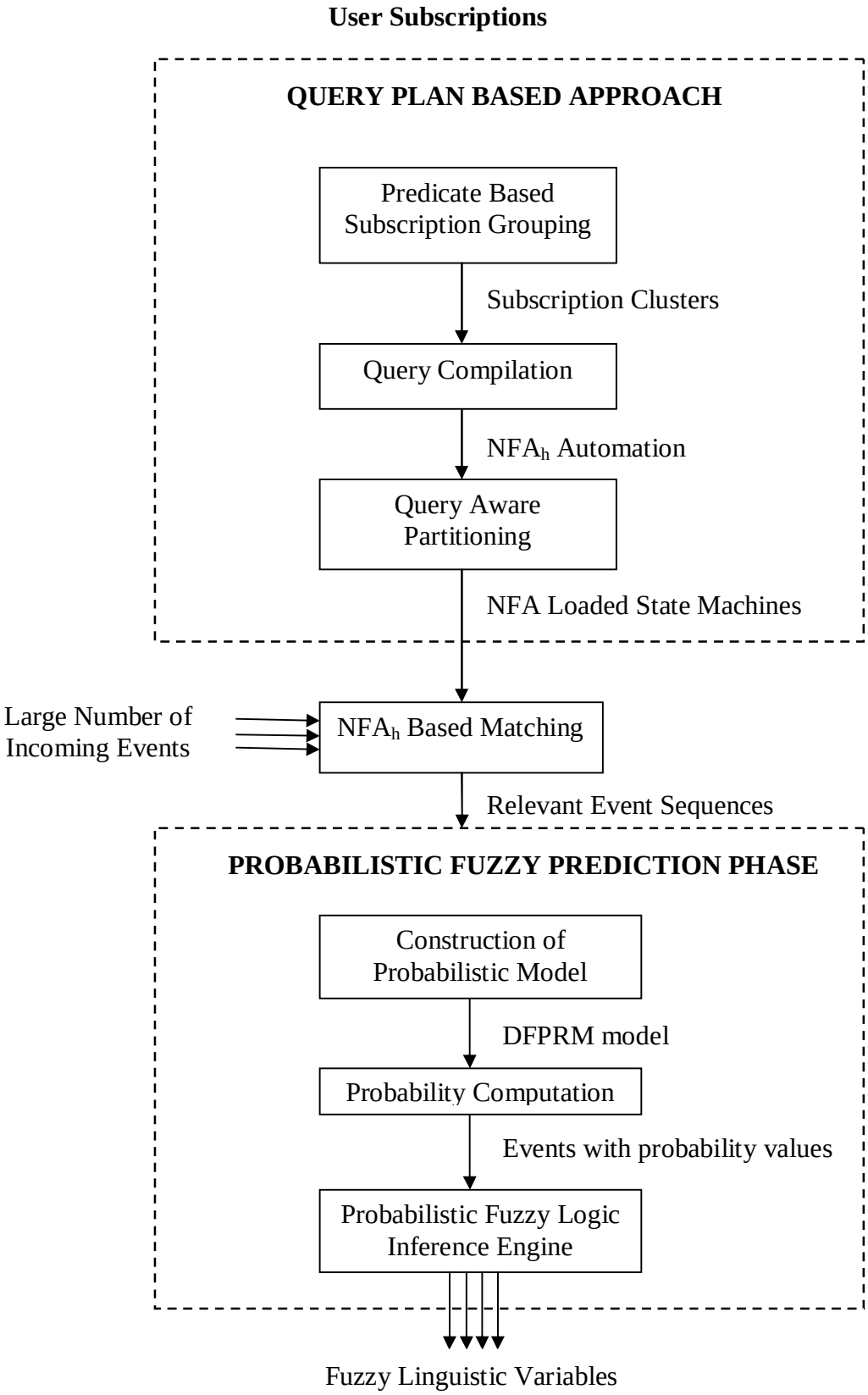

Figure 3.2: Information Flow in PCEP

3.5 SUMMARY

In this chapter, the formal problem statement of the proposed PCEP system is presented. The main research objectives of the PCEP system are formalized. The high level architecture of the proposed system is designed. The modules in the system are identified. Further, the interaction between various modules in the system is established.

CHAPTER 4

GENERIC AND SCALABLE EVENT FILTERING BASED ON NFA_h

This chapter proposes a generic and scalable event filtering in the CEP framework which is designed in the form of Publisher/Subscriber model to achieve scalability. In the Efficient and Generic Event Filtering (EGEF) approach, the complex event processor carries out the event filtering where NFA_h event matching is performed between the stream data that continuously flow from the multiple sources based on the user subscriptions. In order to manage the large number of user subscriptions, Predicate based Subscription Grouping (PSG) algorithm is proposed to group the number of user subscriptions into a set of clusters based on its available predicates. Furthermore, Query Aware Partitioning scheme dispatches the subscription clusters using two techniques - row/column scaling and pipelining to perform fast and efficient event filtering. A NFA_h based effective pattern matching approach is proposed, that filters the required relevant events from the continuously arriving large stream of data according to the user defined rules. It leads to achieve efficient event processing in the presence of voluminous event streams.

4.1 EFFICIENT AND GENERIC EVENT FILTERING

The Efficient and Generic Event Filtering (EGEF) is performed ahead of the event processing to filter out the irrelevant events in order to achieve efficiency and scalability. Event Filtering is implemented in the Publisher/Subscriber model that performs matching among the number of incoming events (publications) and the domain expert specified rules (subscriptions). It is performed in two steps as follows: Cluster subscriptions are formed where the user subscriptions are grouped into clusters and thus corresponding access predicates are mapped by setting the predicate bit vector into one. The NFA matching is performed based on NFA_h query evaluation model between the number of incoming events and user subscriptions.

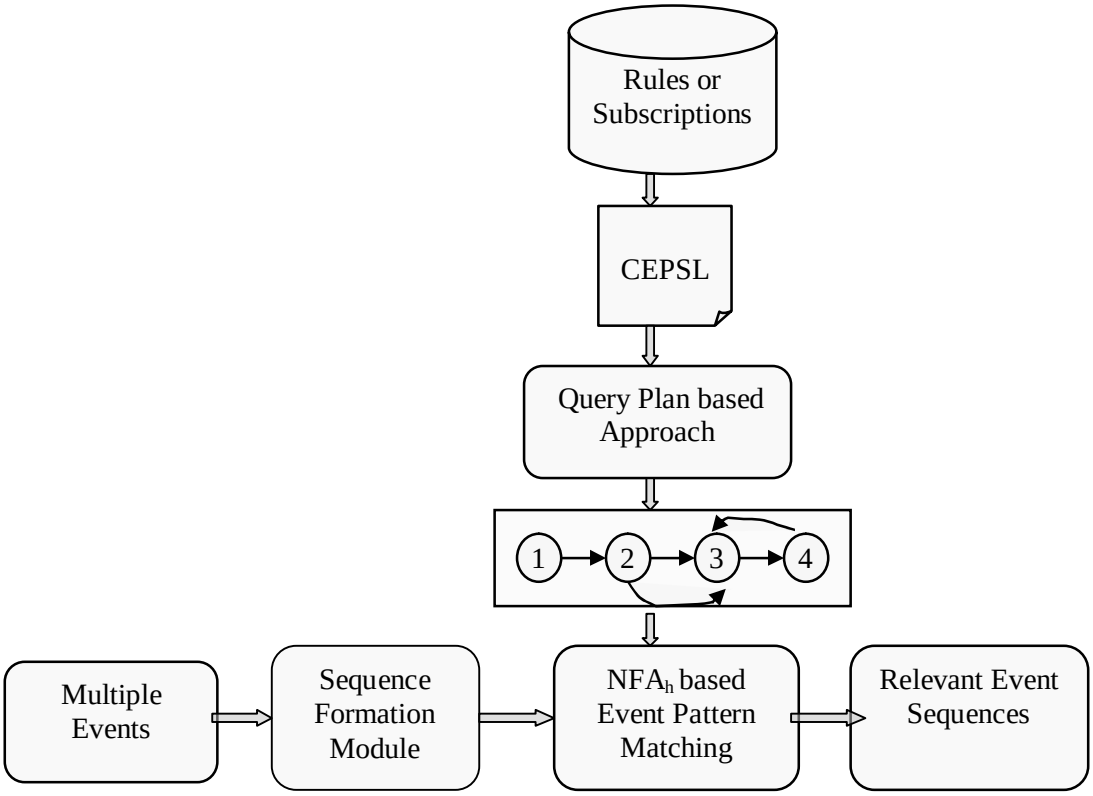

Figure 4.1: Efficient and Generic Event Filtering

Figure 4.1 provides the framework for Efficient and Generic Event Filtering approach. It executes NFA_h based event matching to filter the events which are relevant to the user subscriptions. This approach filters the complex events which satisfy the minimum conditions that have common predicates among the group of similar subscriptions [46]. This process leads to the situation of filtering out the irrelevant events at the early stage due to the efficient grouping of user subscriptions and the loading of common access predicates in each subscription group.

4.2 SEQUENCE FORMATION MODULE

In order to perform NFA_h based event matching, the large number of incoming events are pre-processed into a stream of sequences based on the filter predicate in the Complex Event Pattern Subscription Language (CEPSL) query specification. The sequences of events are constructed through two operators -

sequence traversing or scanning operator and sequence construction operator. The sequence traversing operator (ST←) is used to traverse the sequence of events in order to determine the number of sub-sequence. This is possible in the incoming events and then simultaneously the sequence formation operator (SF←) is used to construct the detected sub-sequence with a set of sequence of events. Therefore, this module takes the set of incoming events as input and constructs the possible set of sub-sequences which are entered as an input to the NFA_h based event pattern matching [59].

4.3 QUERY PLAN BASED APPROACH

The Query Plan based Approach is deployed to manage and to extend the expressiveness of the volume of user subscriptions [59]. The user subscriptions are expressed using CEPSL which defines a set of predicates and the user requirement specification in terms of attributes and their corresponding values. The query compilation is used to convert the event pattern queries into a NFA_h model based on native sequence operators [82]. It takes as input a query defined sequences from the number of continuously arriving events. This process improves the flexibility of the query execution.

4.3.1 Predicate based Subscription Grouping

In order to achieve scalability, the subscriptions clusters are formed. These clusters help to achieve efficient event processing even under the large number of incoming event streams and user subscriptions. The proposed PSG algorithm is performed using a cluster vector. A cluster vector consists of set of predicates, bit vector and reference cluster list. The algorithm is used to group the set of subscriptions based on the subscriptions. In this approach, the subscription cluster consists of ‘n’ number of subscriptions that possess a set of predicates. The subscription cluster is maintained as n-dimensional predicate array. The predicate in the array refers to the position in the predicate bit vector where the predicate associates its binary value by either 0 or 1 [83]. The subscription array is a single dimensional array and consists of subscription identifiers.

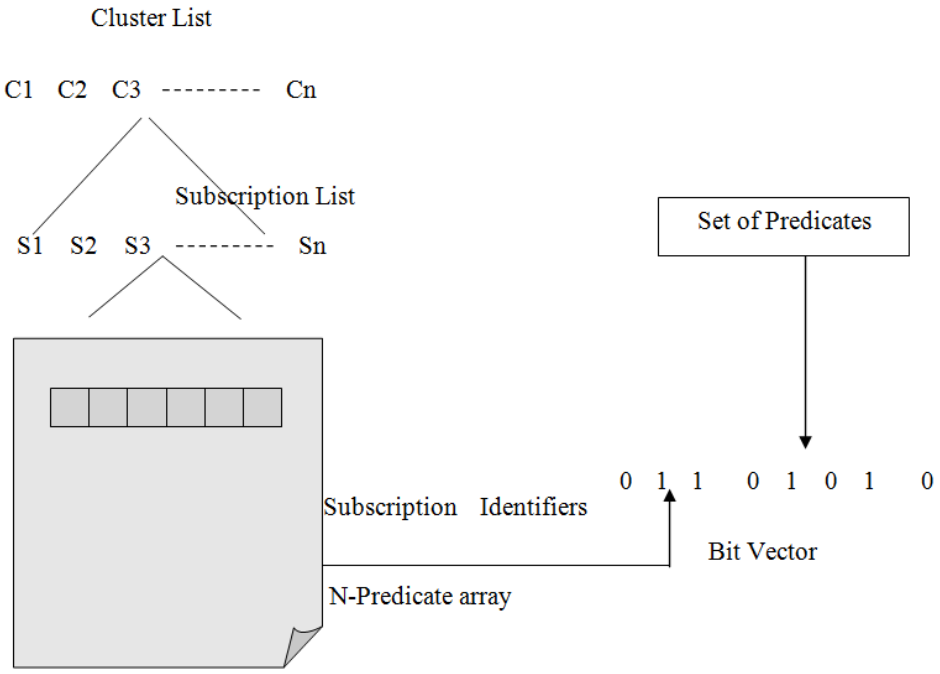

Figure 4.2: Predicate based Subscription Grouping

Here, a single predicate can be present in one or more subscriptions and therefore, it is exceedingly simple to group the relevant subscriptions with same predicates. The predicates present in all of the subscriptions in the cluster have to be assigned as an access predicate for the corresponding cluster. Each subscription cluster is associated with more than one access predicates [83]. If the access predicates of the subscription clusters is satisfied by the incoming event, then subsequently it indicates that the event matches the number of subscriptions in the cluster.

Let there be ‘m’ subscriptions where there are ‘n’ unique predicates then the predicate bit vector B={B[1],B[2],…..,B[n]}. For a given event sequence ‘I’ the predicate bit vector is set according to the presence of a predicate. The subscriptions in the system are grouped according to the similarity in the predicates. A cluster vector is an array which has one entry for each cluster. The entry in the cluster vector contains an access predicate ‘P’ for all the subscriptions in the cluster. A predicate can be an access predicate of a subscription ,if an event fails to satisfy the

access predicate then it does not satisfy the subscription. The algorithm is highlighted as follows:

Input:
Event Sequence E
Output:
Subset of Cluster SC
Variables:
A bit vector B={ B[1],B[2],….,B[n]}
where ‘n’ is the number of predicates
Set of Subscription Cluster C={C[1],C[2],….C[l]}
Where ‘l’ is the number of clusters
A set of subscriptions S
Procedure:
Begin
B=0;
SC={NULL};
S={NULL};
For each predicate ‘P’ in the event sequence ‘E’
with reference i
Set B[i] = 1;
If ‘P’ is the Access predicate for a cluster with reference j
SC= SC U C[j];
Return SC;
End

Figure 4.3: Predicate based Subscription Grouping Algorithm

4.3.2 Query Aware Partitioning

After grouping the user’s subscriptions as clusters using the PSG approach, each subscription cluster must be assigned to a separate machine in order to reduce the heavy load on a single machine and also to scale up the processing capacity even for a number of incoming events and subscriptions. Therefore, the number of machines required to process the queries is determined from the number of clusters formed by the PSG algorithm. Furthermore, the constructed NFA_h automaton from each of the subscription group is loaded directly into the state machine.

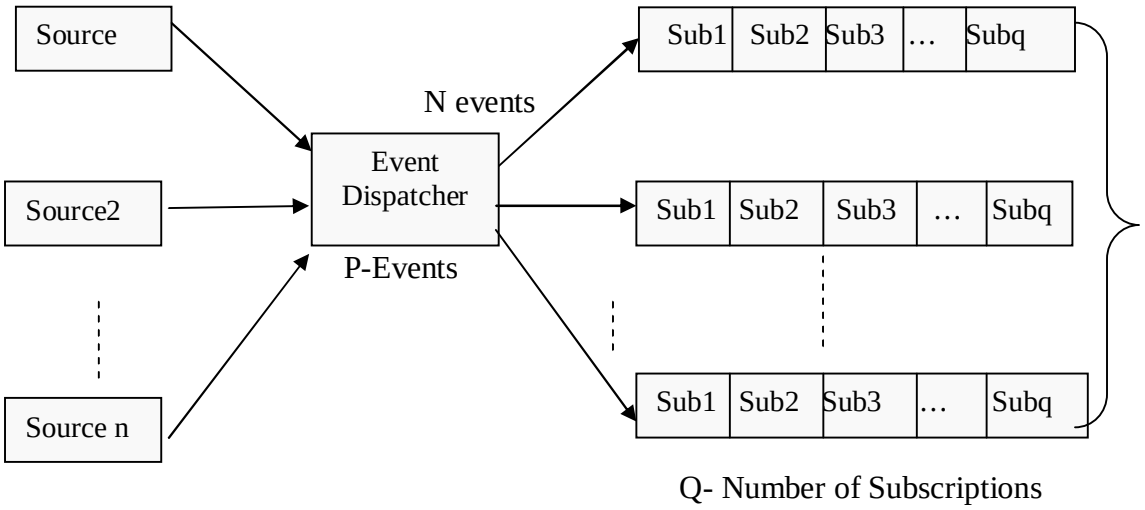

Figure 4.4: Event Dispatching

In order to manage the number of user subscriptions, the number of rows is replicated according to the increasing number of subscriptions [80]. This approach performs the most efficient query partitioning scheme using two techniques - row/column scaling and pipelining. In row/column scaling, the subscriptions are organized in the form of matrix ‘p’ x ‘q’ where ‘p’ number of subscriptions is equally arranged among the ‘q’ number of machines in the system. In addition, the row is replicated ‘q’ times to form a matrix, where ‘p’ is the number of subscription clusters and ‘q’ is the number of incoming events dispatched into the system. Moreover, the event dispatcher is deployed in between the source and query processor, where the dispatcher takes the responsibility to allow a group of ‘q’ number of events into the processing system. An incoming event is dispatched to a row in a round robin manner. The query processing is performed between the incoming event and the subscriptions that are organized in the corresponding row [80]. It guarantees to achieve a fast and efficient processing of query with less complexity and high scalability. In order to process the more number of incoming events, the required number of rows can be added to the matrix that increases the parallelization. The query partitioning scheme can process the large number of incoming events with high scalability and equally improve the throughput rate for the multiple queries by partitioning the queries across a cluster. The pipelining is

used to maintain the processing flow of execution of distributed queries among the large number of separate machines into the system.

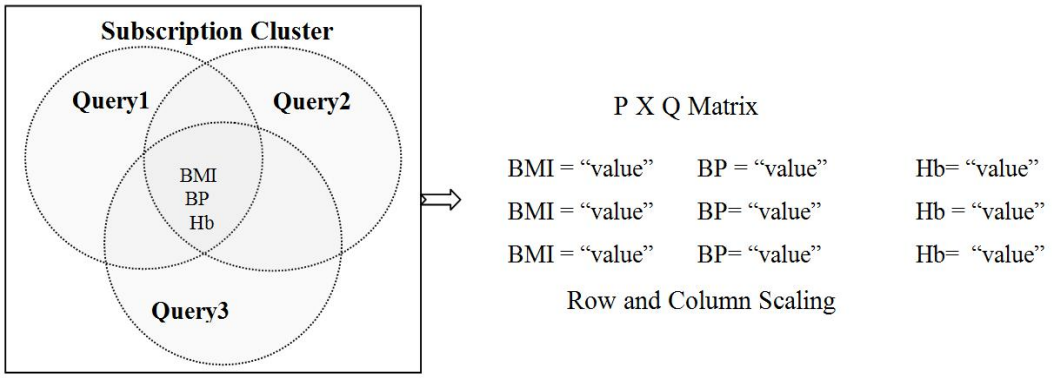

Figure 4.5: Query Aware Partitioning

Each state machine processes the incoming events based on subscriptions using the NFA_h execution model [82] and the output from one state machine is entered into the next subsequent machine using the pipeline [80]. The last machine in the row is referred as an accepting state. Therefore, the event that satisfies the subscription is considered as matched relevant event to the user subscriptions. Figure 4.5 shows the Query Aware Partitioning scheme.

Input:
Subscription Clusters [SC1, SC2, SC3…..SCn]
Output:
NFA Loaded State Machine
Procedure:
Begin
Dispatch (Subscription Clusters);
Row Column Scaling [SC1, SC2, SC3…..SCn]$_{pxq}$;
Pipelining (Query Compilation & NFA_h and Loading);
Construct NFA_h<- Query Compilation (Subscriptions Cluster);
Load State Machine <- Constructed $NFA_{h;}$
End

Figure 4.6: Query Aware Partitioning Algorithm

In query partitioning, the incoming event is dispatched to the ‘q’ number of state machines, but only one state machine can get executed at a time. Therefore, the state machine are arranged in an optimal order with minimum conditions. Pipelining is used to maintain the flow of execution of distributed queries among the large number of separate machines in the system. The algorithm for the Query Aware Partitioning is listed out in Figure 4.6.

4.3.3 Query Compilation

The NFA_h based event matching engine is able to process the large number of incoming event sequences based on the user subscriptions, which may be temporal or conditional queries. Here, the event pattern queries are converted into the suitable query evaluation model called as NFA model. This model is selected for the proposed pattern matching approach because it yields high quality results in terms of efficiency and flexibility for pattern evaluation. The query compilation technique called as Automaton Intermediate Representation (AIR) is used to convert the defined event pattern of CEPSL queries into a new form of automaton called as Non-deterministic Finite Automaton-Heap (NFA_h) [82].The constructed automaton is loaded directly into the state machine to perform the event matching.

```
Input: User Subscriptions
Output: NFAh
        Number of States {st1, st2, st3…};
        Number of Predicates {p1, p2, p3…};
Procedure:
Begin
         NFAh <- Automaton Intermediate Representation (Subscriptions)
        For each subscription
                For (i=1; i < number of Predicates; i++)
                        For (j=1; j < number of States; j++)
                                Create st[j] for NFAh;
         Assign Predicate p[i] to each state st[j];
         Assign Heap memory to NFA;
        Return NFAh
End
```

Figure 4.7: Query Compilation Algorithm

Figure 4.7 highlights the query compilation algorithm. The user subscriptions are given as input to this algorithm and it outputs the NFA_h with number of states and predicates.

4.4 NFA_h BASED EVENT MATCHING ENGINE

In this approach, the NFA model is selected to perform the event matching because it yields high quality results in terms of efficiency and flexibility for pattern evaluation. It provides well-defined semantics for the inclusive set of event pattern queries and also provides a query plan approach to execute the event queries over the large number of incoming event streams.

Figure 4.8 provides the module of NFA_h based Event Pattern Matching that takes two inputs - sub-sequences of events and the set of labeled access predicates of the subscription clusters. The access predicate associated with each subscription group is assigned to the edge of the unique automaton to perform the matching. Therefore, the number of NFA_h automaton required to perform matching is equal to the number of subscription group formed. The access predicate of every group is loaded into a separate state machine with NFA_h Automaton. In NFA_h based Event Pattern Matching, each incoming event sequence traverses through the edge only if it satisfies the assigned corresponding predicate. The events which traverse throughout the edge and reach the final state are considered as relevant events, whereas the events, which do not traverse throughout the edges of the automaton are filtered out as irrelevant events.

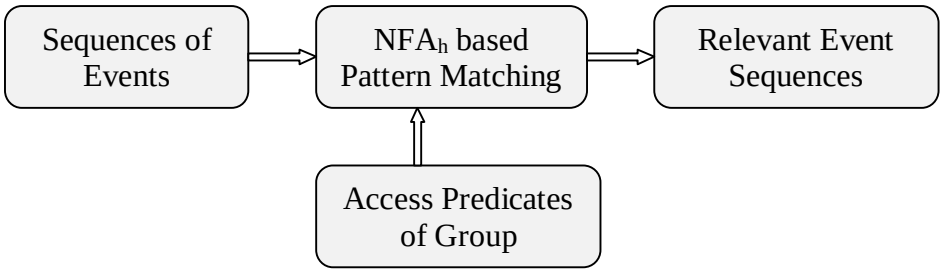

Figure 4.8: NFA_h based Event Pattern Matching

In NFA_h, each state maintains the Active Instance Heap (AIH) to store the active instances of the event that trigger into the other transition state [60]. Therefore, AIH maintains the active instances of each state in a timely manner to arrange the sequence of events in the temporal order. If the incoming event reaches the accepting state of the NFA_h, then it constructs the event sequence using the Directed Acyclic Graph (DAG)[84]. This event sequence is considered as a unique relevant event sequence. The event sequences that reach the final state of the automaton are filtered as relevant sequence of events [85]. The NFA_h pattern matching algorithm is listed in Figure 4.9.

```
Inputs:
        Incoming Events, NFAh
Output:
        Relevant Events
Procedure:
Begin
        For (i=1; i < number of states; i++)
                if (Incoming Event satisfy Conditional Predicates in State)
                        Event Transition to Next State;
                        If (state = final state)
                                Return as relevant event;
                        Else if (not satisfy conditional predicates in state)
                                Eliminate as Irrelevant event;
End
```

Figure 4.9: NFA_h Pattern Matching Algorithm

4.4.1 NFA_h Automaton

The NFA_h model consists of four edges such as Begin Edge, Forward Edge, Filter Edge and Rebind Edge, where each edge deploys a heap memory to store the active instance which is satisfied in its current state. In NFA_h, the Forward Edge is used to find the event using the filtering mechanism performed by the Filter Edge which filters the event using the iteration operator [85]. The access predicate is associated with each subscription group at the edge of the automaton to perform matching. Therefore, the number of NFA_h automaton required to perform matching

is equal to the number of subscription group formed. The access predicate of every group is loaded directly into a separate state machine of NFA_h Automaton.

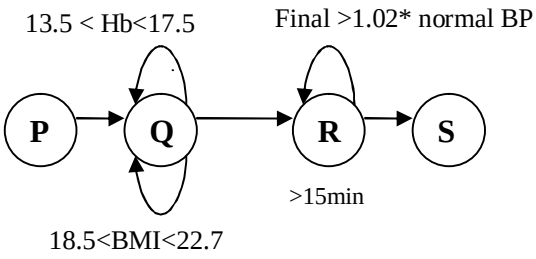

Figure 4.10: NFA_h Automaton for CEPSL Query

The top loop of the automaton has a Filter Edge that takes a conditional argument. A predicate called as name in the first fold construct of the CEPSL query is shown in Figure 4.10.This edge allows automaton instance of the event schema that consists of health deciding factors of the patients coming for the health checkup in the medical domain [82]. The bottom loop of the automaton is called as a Rebind Edge 'Q' that is used to express the FILTER construct of the CEPSL Query. This edge allows the automaton instance of the patient that matches the predicates mentioned in the 'P' and 'Q' edges of the constructed automaton. The Forward Edge QR allows the automaton instance of matched event schema of the patient who has normal Body Mass Index (BMI) and Haemoglobin content (Hb). The Filter Edge in the 'R' state takes the conditional expression in the NEXT Construct that allows to find the immediately next event in the event schemes for the other patients. The last transition state from state 'R' to 'S' allows the automaton instance that matches the associated predicates in the 'R' edge. Thus, the automaton instance reaching the final state has the set of event schemes that matches the predicates associated in each edge of the automaton. The automaton thus predicts the health deficiency of the patient after the rebound.

4.4.2 Pipelining

In existing systems, the event matching is carried out without pipelining where the incoming events are processed sequentially from left to the right where

each machine republishes the matching events to the next machine in the system. However, in the case of a large number of incoming events, it leads to high resource consumption and heavy workload on a single machine.

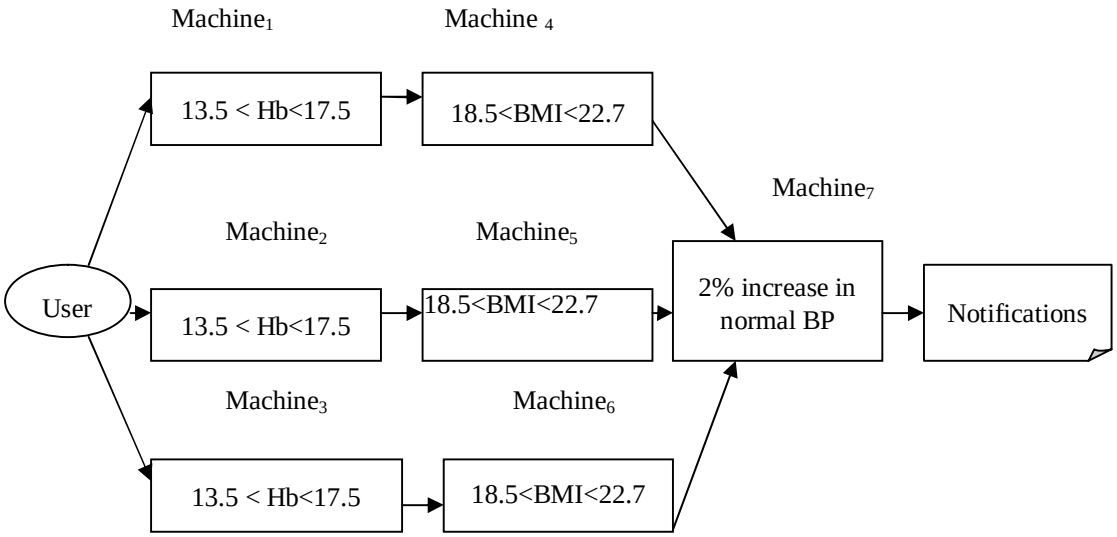

Figure 4.11: NFA_h based Event Matching with Pipelining

In order to scale up the processing capacity, the PCEP system utilizes the event filtering using pipelining to process the large number of incoming events [80]. It breaks the query into a number of sub-queries and distributes them. The sub-queries are loaded onto the large numbers of separate state machines in the different stages of the pipeline in order to achieve scalability. Hence, the heavy load on a single machine is reduced and the number of sub-queries is assigned to more number of state machines. Therefore, the incoming events are processed using the pipeline that helps to achieve faster execution and higher throughput. In the Figure 4.11, the event matching for certain attributes (stateless) such as Haemoglobin content and BMI in the complex query are replicated in order to scale up the system for high throughput. In the last stage of the pipeline, an uncertain attribute (stateful) called as increase in BP is not replicated because it changes over time. Hence, in order to predict the event sequences that match the uncertain attributes, Probabilistic Event Sequence Prediction (PESP) is needed to predict the relevant sequences which are discussed in the next chapter.

4.5 PERFORMANCE EVALUATION

In order to evaluate the NFA_h based event filtering, the PCEP system is implemented in the Publisher/Subscriber model using Java Messaging Service (JMS) based subscription API. The main purpose of this prototype is to validate and to assess the impact of the filtering approach. The performance evaluation is performed in terms of scalability and efficiency.

4.5.1 Experimental Setup

In the experimental setup, the structure of generic application methodology used is a message oriented middleware with the set of software components to perform efficient event processing. These software components communicate through Java Message Service which takes input as event instances from continuously arriving incoming events. The experiment is executed on Windows XP PC with 3.2 GHz processor, 2 GB of RAM and 512 MB cache with the maximum java heap size of 800 Mbytes. It is implemented in open source Java Enterprise Edition/Netbean/ Glassfish /JMS environment.

4.5.2 Datasets

The PCEP system is executed using the dataset of health checkup in a medical domain for event triggering. The performance of the PCEP system is evaluated based on 10000 incoming events. The dataset consists of 15 attributes. The first six attributes are certain attributes such as the name of the patient, age, address, contact number, occupation and blood group. The remaining nine attributes are uncertain attributes such as Haemoglobin content, Body Mass Index, blood pressure, cognitive patterns, communication and hearing patterns, dental status, nutrition status, disease health condition and treatment procedure. This approach provides 500 rules (health condition) in order to trigger relevant events (appropriate treatment) according to the corresponding information in the dataset. The event rate per second is measured from the system.

4.6 EFFICIENCY AND SCALABILITY OF THE EGEF APPROACH

The experiment is carried out to compare the performance of the PCEP system with and without filtering. The PCEP system achieves high efficiency and scalability compared to the CEP without filtering. The PCEP system filters out the irrelevant events before the event detection starts according to the domain expert specified rules. The performance of the PCEP system is evaluated based on the throughput (the number of events executed per second) and the average processing time (time taken to process the number of events per second).

4.6.1 Throughput

In the PCEP system, the user subscriptions are efficiently grouped based on the access predicate into subscription clusters. Nearly, one to ten number of subscription groups is formed and the corresponding access predicates are allotted to separate state machines. Therefore, the throughput is improved. This implies the increased number of events processed per second according to the varying number of the state machine. On the other hand, in the existing approach, the subscriptions are grouped randomly.

Hypothesis with respect to the result of the parameter T: (Throughput)

Null hypothesis H0: T1 = T2 where T1= Throughput obtained in Distributed Cayuga, T2 = Throughput obtained in PCEP Filtering

(There is no significant difference between the two systems that is Distributed Cayuga and PCEP filtering in terms of throughput obtained)

Alternate hypothesis H1: Throughput mean values are not equal for at least one pair of the result mean values of the parameter T.

(There is a significant difference between the two systems in terms of throughput obtained)

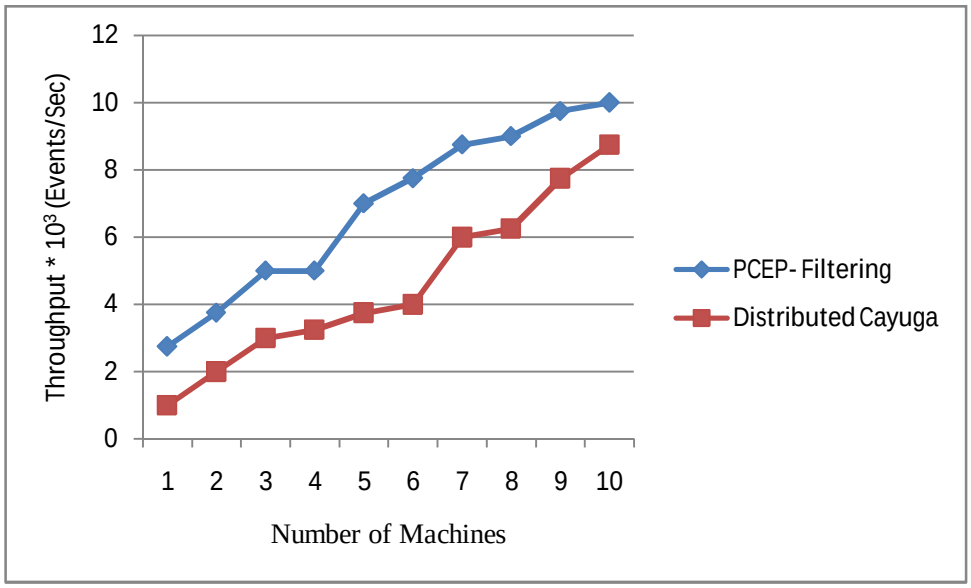

Figure 4.12: Throughput versus Number of Machines

Figure 4.12 shows the average processing time according to the varying number of machines. If the number of machines used to process the events increases, the number of events per second also increase correspondingly. Therefore, the PCEP system achieves scalability in terms of execution time.

Hypothesis Evaluation with Respect to T: (Throughput)

Table 4.1: T-test for Distributed Cayuga and PCEP Filtering based on Throughput

Events x 10^4	Technique		Throughput	
	I	II	Hypothesis	p value
1-10	Distributed Cayuga	PCEP Filtering	H1	0.039

From Table 4.1, it is concluded that the calculated significance level of the parameter throughput of comparing two systems Distributed Cayuga and PCEP filtering satisfies the condition (p value<0.05) for number of machines. There is a significant difference between the results for different throughput values of Distributed Cayuga and PCEP with filtering. Hence, the null hypothesis for H1 is rejected.

Table 4.2: Descriptive Statistics of Throughput Measures

Technique	Distributed Cayuga	PCEP with Filtering
Max	8750	10000
Min	1000	2750
Mean	4575	6875
Median	3875	7375
Standard Deviation	2517	2596

Further, it is also required to determine the system which has the maximum throughput. This is analyzed using descriptive statistics given in Table 4.2. Table 4.2 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter throughput. There is an increase in Average Throughput of PCEP-Filtering by 33% with respect to Distributed Cayuga.

4.6.2 Average Processing Time

In PCEP system, the Average Processing Time is not increased significantly even when the events per second is increased. Moreover, there is only a gradual increase in the average processing time of the PCEP system with EGEF filtering approach. This is due to the filtering of many irrelevant events using NFA_h based effective pattern matching with minimum requirements. However, the existing CEP approaches acquire a significant increase in Average Processing Time.

Hypothesis with respect to the result of the parameter A: (Average Processing Time)

Null hypothesis H0: A1 = A2, where A1= Average Processing Time in Distributed Cayuga, A2 = Average Processing Time in PCEP filtering

(There is no significant difference between the two systems in terms of Average Processing Time obtained)

Alternate hypothesis H2: Average Processing Time mean values are not equal for at least one pair of the result mean values of the parameter A.

(There is a significant difference between the two systems in terms of Average Processing Time obtained)

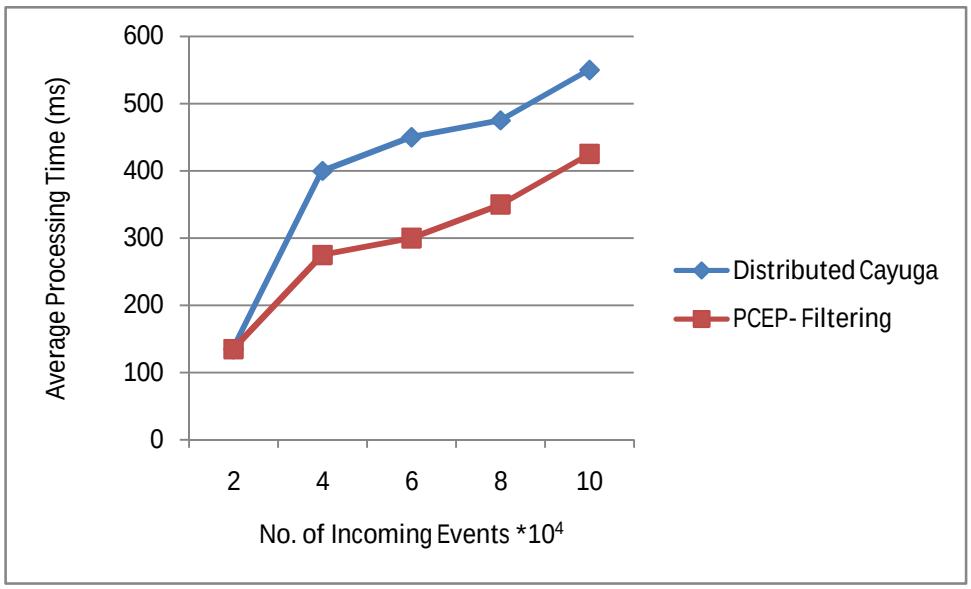

Figure 4.13: Average Processing Time versus Number of Events

Figure 4.13 shows the Average Processing Time taken to process the complex events according to the number of events per second.

Table 4.3: T-test for Distributed Cayuga and PCEP Filtering based on Average Processing Time

No. of events x 10^3	Technique		Average Processing Time	
	I	II	Hypothesis	p value
2-10	Distributed Cayuga	PCEP with filtering	H2	0.041

From Table 4.3, it is concluded that the calculated significance level of the parameter Average Processing Time of comparing the two systems Distributed Cayuga and PCEP with filtering satisfies the condition (p value<0.05) for input

events from 2000 to 10000. There is a significant difference between the results for Average Processing Time values Distributed Cayuga and PCEP with Filtering. Hence, the null hypothesis for H2 is rejected.

Further, it is also required to determine the systems which have the minimum Average Processing Time. Table 4.4 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Average Processing Time.

Table 4.4: Descriptive Statistics of Average Processing Time Measures

Technique	Distributed Cayuga	PCEP 5
Max	135	135
Min	550	425
Mean	402	297
Median	450	300
Standard Deviation	159	107

There is a decrease in Average Processing Time of PCEP-Filtering by 26% with respect to Distributed Cayuga.

4.7 SUMMARY

In this chapter, the PCEP system is implemented based on Publisher/Subscriber model to achieve high scalability and efficiency. The event filtering approach is a pre-processing system, which is performed ahead of event processing in order to manage the arrival of a large number of incoming events. The proposed NFA_h query filtering approach performs event matching between the large number of incoming events (publications) and rules (subscriptions). An expressive language called as Complex Event Pattern Subscription Language is used that provides high expressibility for efficient pattern queries and also supports high extensibility for the formal event language. A Predicate based Subscription Grouping algorithm is proposed to group user subscriptions based access predicate to improve the scalability.

CHAPTER 5

A PROBABILISTIC FUZZY MODEL FOR REASONING OVER UNCERTAINTY

This chapter proposes a Probabilistic Complex Event Processing (PCEP) system that consists of two phases of Efficient and Generic Event Filtering (EGEF) and Probabilistic Event Sequence Prediction (PESP) phase. In the first phase, the relevant events are filtered and the filtered events enter into the prediction phase. The second phase consists of an Efficient Event Sequence Prediction paradigm that triggers complex events usable by the end user application. To determine the effectiveness of the PCEP system, a detailed performance analysis is performed using the prototype implementation. As a result, it is demonstrated that the PCEP system outperforms the existing CEP approach.

5.1 PROBABILISTIC EVENT SEQUENCE PREDICTION

After filtering out the relevant event sequences from the large number of incoming events, Probabilistic Event Sequence Prediction in Figure 5.1 derives a stateful composite event sequences from the filtered relevant events sequences based on the probabilistic framework.

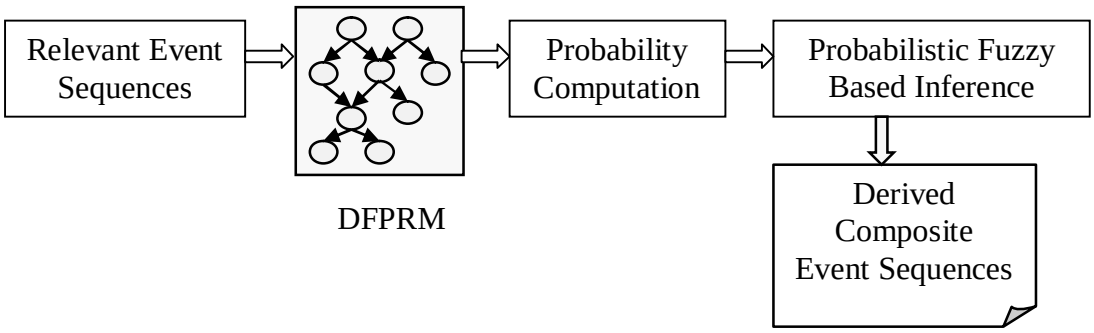

Figure 5.1: Efficient Event Sequence Prediction Paradigm

Event hierarchy is constructed in the form of graphical model called as Dynamic Fuzzy Probabilistic Relational Model (DFPRM) [86] [87] that computes

joint probability distribution using the conditional probabilistic dependencies between the event sequences in accordance to the rules. In order to reduce the large sample space, the Fuzzy Logic is used to infer the correlation between the event sequences using the linguistic variables. It enhances the robustness of the complex event detection process under uncertainty.

5.1.1 Dynamic Fuzzy Probabilistic Relational Model (DFPRM)

Dynamic fuzzy probabilistic relational model is constructed from a set of the relevant event sequences. The rules to represent the probability space are presented in terms of concept of individuals, their properties and relations between them [88]. DFPRM represents the qualitative knowledge between the set of events and their variable interrelationships, whereas the Conditional Probability Distribution [78] [88] represents the quantitative knowledge of probabilities in a large probability space.

Figure 5.2 shows the DFPRM for the event sequences 'E' with a set of events $E = \{e_1, e_2 \ldots .e_n\}$ and each event e_i is associated with a set of descriptive attributes and the reference slots. There is a direct mapping between the event classes and their set of attributes for all the events in the event sequence. This model consists of set of event classes with their associated attributes and the joint probability distribution computed from the arbitrary correlation between the event sequences.

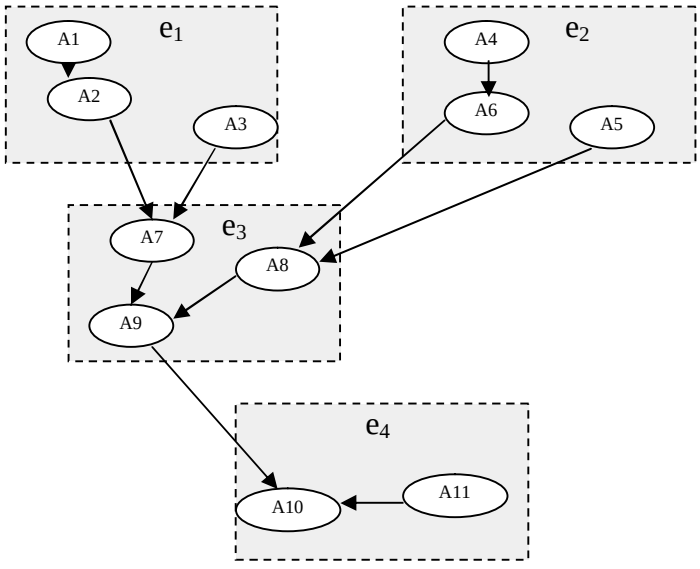

Figure 5.2: Dynamic Fuzzy Probabilistic Relational Model

5.1.2 Probability Computation for Event Sequence

The novelty of this work is the computation of joint probability distribution based on the conditional probabilistic dependencies among the event sequences in the constructed probabilistic model. It also represents the constructed probability space that possesses the formal semantics in terms of probability distributions over a set of relational logic interpretations. It represents the probability space as triples $\{W_T, \Omega_T, \theta_T\}$ where W_T is a set of possible worlds, Ω_T is an event history associated with each possible world and θ_T is a probability measure of the possible world. The sample space of event sequence 'E' consists of the conjunction of set of possible events with their associated probability measure from the event history. The probability of the constructed event sequence is computed from the conditional probability dependence among the set of attributes associated with the events . The overall probability associated with the event sequence is factorized by aggregating the product of local conditional probabilities of the events in the event sequences, according to the conditional dependence of the event given their parent nodes. The computation of Conditional Probability Distribution (CPD) of event sequence is as follows:

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

Here, Ω_i is the set of parental nodes of e_i and then Φ_i is the parameter vector associated with e_i. The CPD of event sequence 'E' computed from the probability distribution over the values of events 'e_i' is given as the combination of values for each of the parents $P(e_i)$ [72] [89]. More precisely the joint distribution of the incoming event can be factorized into a product of the CPDs of all the uncertain attributes occurred in the event.

5.1.3 Probabilistic Fuzzy Logic based Inference Engine

In order to formulate the combination of event sequences according to the computed probability of events with reduced overhead, Probabilistic Fuzzy Logic (PFL) [41][90] is used to estimate the fuzzy linguistic variables from the computed conditional probability distributions in the large probability space. PFL is a

reasoning process to model the heuristic knowledge for estimating the linguistic information from the degree of truth in the imprecise data using membership function. In order to approximate the large sample space, fuzzy partitioning scheme partitions the sample space with a set of possible worlds into a certain number of predetermined membership values as shown in Figure 5.3.

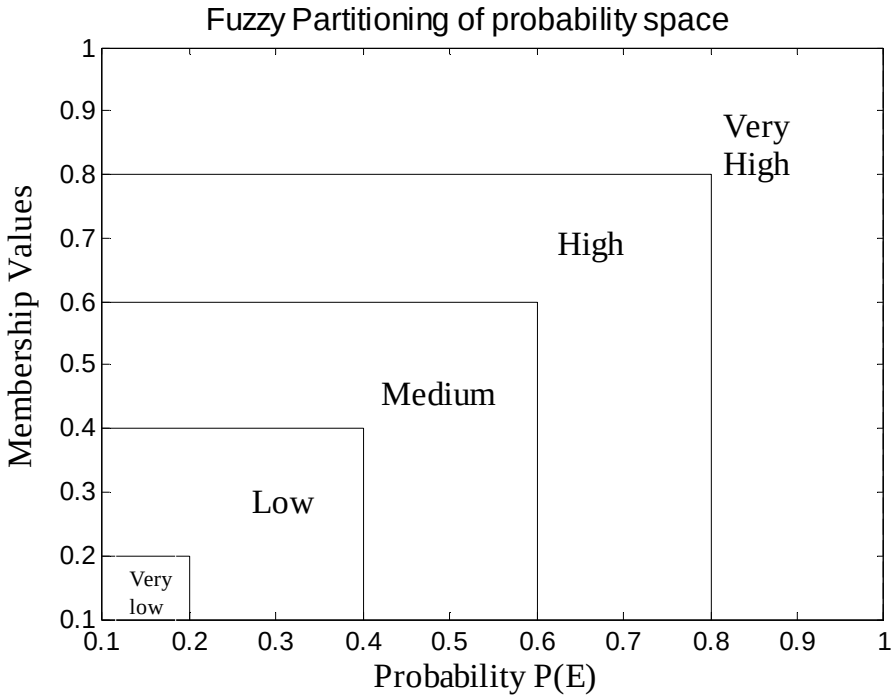

Figure 5.3: Fuzzy Partitioning of Large Probability Space

Each sample space is composed of 'n' number of membership functions that span the entire sample space of a fuzzy inference system. It is ensured that the probability of each event sequence must be 1 which is the sum up of the associated probability with all set of events in it. A fuzzy set consists of a set of linguistic variables, which are defined by the domain experts in the form of the characteristic function called as Membership Function. It is represented as μ_F:P(E)→[0, 1]. The Membership Function is used to define the certainty that element P(E) belong to that fuzzy set F [90]. It is used to associate a degree of membership of each of the possible world in the sample space to the corresponding fuzzy set. In Fuzzy Quantification, the fuzzy set consists of five linguistic variables -very low, low,

medium, high and very high [91] [92]. The linguistic variables represent the words or sentences used to deal with semantic concepts of imprecise nature directly by means of approximate characterization using mathematical formulation. It is assigned to the set of events in the certain probability space based on the membership values. Figure 5.4 shows the Probabilistic Fuzzy Logic based Inference Engine. It takes the probability of combination of events as an input and then generates the output of event sequences in an order with a fuzzy value. It consists of three main components Fuzzifier, Rule base and Inference Engine.

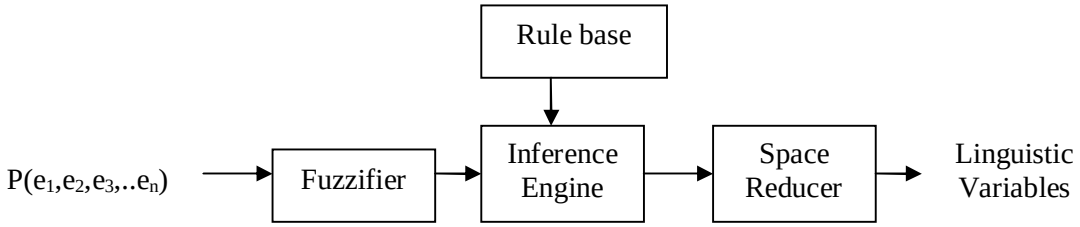

Figure 5.4: Probabilistic Fuzzy Logic based Inference Engine

The Fuzzifier is used to convert the input probability of the possible worlds (non- fuzzy) of a sample space into a membership degree (fuzzy) with the help of membership function associated with each fuzzy set in the rule input space using normalization of maximum likelihood [93]. The membership degree for output set is computed from the degrees of membership according to the relationships among the input fuzzy values. Moreover, the Rule base defines the fuzzy rules provided by the domain experts of the system. The Inference Engine executes the fuzzy logic inference to map the linguistic variable from the fuzzy sets based on the membership function in the rule base. The fuzzy value is assigned according to how well it matches with the membership degree of certainty of the fuzzy set. It is a useful and initiative approach to reduce the sample space to derive the most probable event sequence approximately. Therefore, the events with most priority fuzzy values are derived as a composite of event sequence. The procedure for probabilistic complex event processing is given in Figure 5.5.

Input: Relevant Event Sequences
Output: Derived Composite Events Sequences
Procedure :
Begin
Step 1: DPFRM Model -> Construct Event Hierarchy (Relevant Event Sequences)
Step 2: Compute Joint Probability Distribution -> Conditional Probability between event sequences
Step 3: Probabilistic Fuzzy Logic (Computed Probability)
Step 4: Fuzzy Partitioning (Large Sample Space)
Step 5: Divide into set of possible worlds using membership function
Step 6: Inference engine estimate Linguistic variables using Rule Base
Step 7: Derive Composite Event Sequences
End

Figure 5.5: Procedure for Probabilistic Complex Event Processing

5.2 PERFORMANCE EVALUATION

In this section, the PCEP system is implemented in the Publisher/ Subscriber model using JMS based subscription API and the experiments are carried out based on the streams of stock quote of l0,000. The PCEP approach is implemented in stock marketing scenario. It detects the relevant events from the large number of incoming events under uncertainty. The PCEP system is tested with the IT stocks of TCS, Google, CTS and Microsoft. The data used for evaluating this approach was obtained from the website www.moneycontrol.com that provided the stock prices prevailing at NASDAQ. The system takes incoming events as input of various company stocks. Further, it generates suitable output that ensures that the decision will be suitable for the investor to make a profit in a highly dynamic stock environment. The data are collected for a specific period. From the collected data, it is easier to predict the opening, highest, lowest and closing values of the stock price for each day.

5.2.1 Experimental Setup

The generic application methodology is designed as a message oriented middleware with the set of software components to perform efficient event processing. These software components communicate through Java Message

Service, which takes input as event instances from continuously arriving incoming events. The experiment is executed on Windows XP PC with 3.2 GHz processor, 2 GB of RAM and 512 MB cache with the maximum Java heap size of 800 Mbytes. It is implemented in open source Java Enterprise Edition/Netbean/Glassfish/JMS environment. The PCEP system triggers the new composite events in a real-time application of stock market for a financial domain. The performance of the PCEP system is tested using the event stream to the order of 10,000 incoming events that consist of 11 attributes of which six attributes are certain attributes such as the name of the company, product, category, volume, number of shares, timestamp. The remaining five attributes are uncertain attributes such as stock price, Price-to-Earnings ratio (P/E), Price-to-Sales Ratio (PSR), Return On Equity (ROE), Earnings Growth (EG)and Debt-to-Asset ratio (D/A). In order to trigger relevant events as per user subscriptions, it takes 500 user subscriptions which are subscribed by users (stock trader, stock investor). The time taken to perform matching between the incoming events with all user subscriptions in the system measured as Average Processing Time.

5.2.2 Benchmark Application: Stock Market

In a real-time application scenario, a typical CEP under uncertainty can be found in technical analysis of the stock market. In order to attain a profit margin by investing money of the software market, the stock trader should observe the trading history (Event Instance Data) of the reputed software companies in the stock market. Furthermore, an appropriate decision must be taken by comparing the movement of stock share based on the index and thus indicate the appropriate time to invest the share on the company. The core of the system implements the CEP logic that notifies the change in stock price of incoming event streams of the market stock data according to user subscriptions.

5.2.3 Experimental Results

A preliminary experimental evaluation is conducted to validate the scalability of the PCEP system based on two metrics - Throughput (events/sec) and Average Processing Time (ms).

5.2.3.1 Throughput of PCEP

The throughput of the CEP scheme is defined as the number of events processed in a second by the processor. It relies on the power of the processing engine to process the incoming events based on the user subscriptions. In existing systems, heavy workload occurs because the filtering is not distributed among the multiple machines. Therefore, it cannot process the more number of events, which decreases the throughput. In the PCEP system, a number of state machines are deployed to process the incoming events. Therefore, it can manage a large number of incoming events with the help of the appropriate number of state machines that improves the throughput of the PCEP system.

i) Effect of Varying Number of State Machines in Throughput

Hypothesis with respect to the result of the parameter T: (Throughput) as a function of State Machines

Null hypothesis H0: T1 = T2 = T3, where T1= Throughput obtained in Distributed Cayuga, T2 = Throughput obtained in PCEP 7 and T3 = Throughput obtained in PCEP 10.

(There is no significant difference among the three systems in terms of Throughput obtained)

Alternate hypothesis H3: Throughput mean values are not equal for at least one pair of the result mean values of the parameter T.

(There is a significant difference among the three systems in terms of Throughput obtained)

In Distributed Cayuga approach, the throughput is reduced at the starting stage of processing of queries because the distributed filtering is not performed in the case of a large number of incoming events. Figure 5.6 depicts that the PCEP with 10 state machines achieves high throughput. For example, when the number of incoming events is 70000, the throughput is 4800, 5900 and 7100 for Distributed Cayuga, PCEP with 7 machines and PCEP with 10 machines respectively

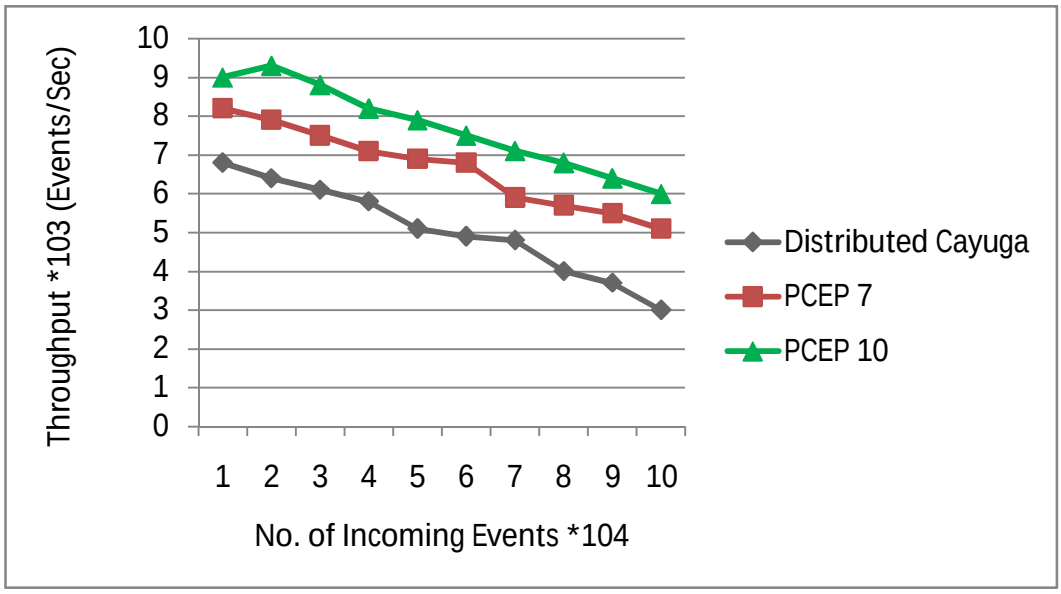

Figure 5.6: Throughput as a Function of Varying Number of State Machines

Figure 5.6 examines how both distributed Cayuga and PCEP systems can handle high event input rates, in the order of tens of thousands of events per second. PCEP outperforms existing distributed in all scenarios. More particularly, PCEP can process the large number of incoming events with high throughput and then it starts declines but less than existing distributed Cayuga scheme. It also shows that the throughput of PCEP is appreciable even for the large number of incoming events.

Hypothesis Evaluation with respect to Throughput as a function of State Machines

Table 5.1: ANOVA for Throughput as a Function of State Machines

Events x 10^4	Technique			Throughput	
	I	II	III	Hypothesis	p value
1-10	Distributed Cayuga	PCEP 7	PCEP 10	H3	0.000

From Table 5.1, it is concluded that the calculated significance level of the parameter throughput of comparing three systems Distributed Cayuga, PCEP 7 and PCEP10 always satisfy the condition (p value<0.05) for input events ranging from 10,000 to 1,00,000. There is significant difference between the results for different throughput values of Distributed Cayuga, PCEP 7 and PCEP10. Hence, the null hypothesis for H3 may be rejected.

Further, it is also required to determine the system which has the maximum Throughput. This is analyzed using descriptive statistics given in Table 5.2. Table 5.2 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Throughput.

Table 5.2: Descriptive Statistics of Throughput Measures

Technique	Distributed Cayuga	PCEP 7	PCEP 10
Max	6.800	8.200	9.300
Min	3.000	5.100	6.000
Mean	5.060	6.660	7.700
Median	5.000	6.850	7.700
Standard Deviation	1.237	1.063	1.132

From Table 5.2, it is evident that there is an increase in mean Throughput values in PCEP 7 and PCEP 10 compared to Distributed Cayuga. It can be concluded that an increase in number of state machines increases the mean Throughput. The increase in the average Throughput of PCEP 7 and PCEP 10 with respect to Distributed Cayuga is 31.62% and 52.17% respectively.

ii) Effect of varying Size of NFA Length in Throughput

Hypothesis with respect to the result of the parameter T: (Throughput) as function of NFA length

Null hypothesis H0 : T1 = T2 = T3, where T1= Throughput obtained in Distributed Cayuga, T2 = Throughput obtained in PCEP 5 and T3 = Throughput obtained in PCEP 10.

(There is no significant difference between the three systems in terms of Throughput obtained)

Alternate hypothesis H4: Throughput mean values are not equal for at least one pair of the result mean values of the parameter T.

(There is a significant difference between the three systems in terms of Throughput obtained)

In NFA based event matching, the size of the NFA automaton gradually grows according to the increasing number of queries that may reflect on the number of state transitions required. The Throughput of the system is affected by parameters such as the number of state transitions required to process each event, number of predicates on the state of the automaton needed to evaluate the queries and the memory requirements to store the NFA automaton.

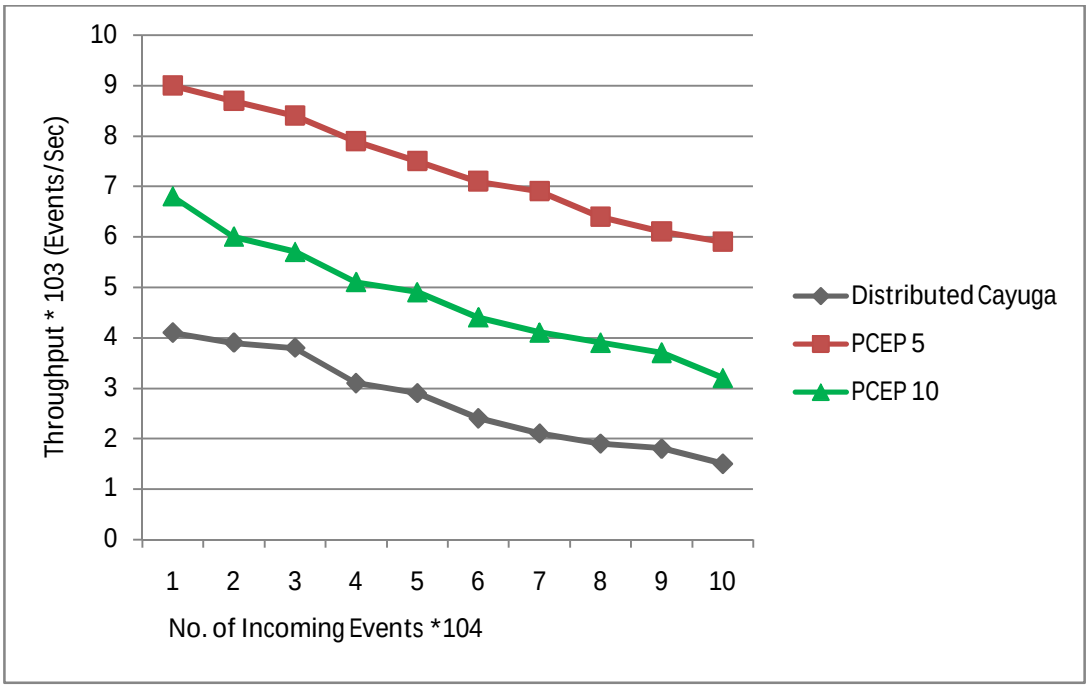

Figure 5.7: Throughput as a Function of NFA Length

In existing systems, the performance of the system will affected by running duplicate or redundant queries at a time. Therefore, the PCEP system deploys a Query Aware Partitioning with the help of a predicate based subscription grouping that manages similar sub-queries among the different queries in the subscription cluster. It eliminates the duplicate and redundant states even under the

large number of queries and also achieves faster execution with high throughput [94]. Figure 5.7 shows how throughput of the PCEP improves as the number of states of the automaton decreases. For 9000 queries, the Distributed Cayuga processes 18000 events per second, whereas PCEP with 10 states processes 37000 events per second and PCEP with 5 NFA length processes 61000 events per second.

Hypothesis Evaluation with respect to parameter T: (Throughput) as function of NFA length

Table 5.3: ANOVA for Throughput as Function of NFA Length

Events x 10^3/sec	Technique			Throughput	
	I	II	III	Hypothesis	p value
1-10	Distributed Cayuga	PCEP 5	PCEP 10	H4	0.000

From Table 5.3, it is concluded that the calculated significance level of the parameter throughput of comparing the three systems: Distributed Cayuga, PCEP 5 and PCEP 10 always satisfy the condition (p value<0.05) for input events ranging from 1000 to 10,000. There is significant difference between the results for different throughput values of Distributed Cayuga, PCEP 7 and PCEP 10. Hence, the null hypothesis for H4 may be rejected.

Further, it is also required to determine the system which has the maximum Throughput. This is analyzed using descriptive statistics given in Table 5.4. Table 5.4 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Throughput.

Table 5.4: Descriptive Statistics of Throughput Measures

Technique	Distributed Cayuga	PCEP 7	PCEP 10
Max	4.100	9.000	6.800
Min	1.500	5.900	3.200
Mean	2.750	7.390	4.780
Median	2.650	7.300	4.650
Standard Deviation	0.950	1.095	1.134

From Table 5.4, it is evident that there is an increase in mean Throughput values in PCEP 5 and PCEP 10 compared to Distributed Cayuga. It can be concluded that there is an increase in the average Throughput as a function of NFA length with respect to Distributed Cayuga by 42.46%.

iii) Effect of Varying Possible World Space in Throughput

In the PCEP system, the Probabilistic Fuzzy Logic based inference is used to derive the most probable event sequence approximately with reduced sample space. Whereas in existing probabilistic based inference, the probability sample space composes of a large number of possible worlds of event sequences is defined by the EID at a time t_1. Therefore, the different set of possible world will be available at different time points: t_2 and t_3. Hence, the probability sample space may vary according to the time. All the same, it is difficult to perform the probabilistic inference on the large number of possible worlds of the sample space.

Figure 5.8 shows how throughput of the PCEP system that deploys a Probabilistic Fuzzy Logic inference achieves high throughput (events/sec) even under the large number of incoming events because of reduced sample space. On the contrary, in the existing probabilistic inference based approach, the throughput declines as the number of possible worlds increases. It reduces the throughput of the system according to the increasing number of events because it yields a large probability space with the number of possible worlds.

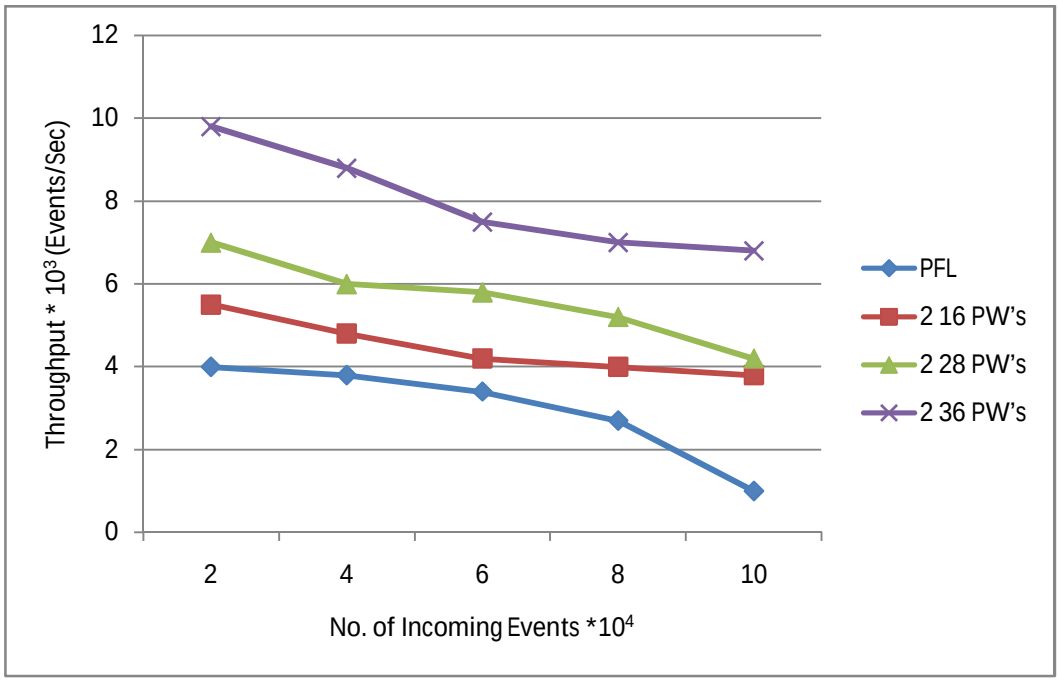

Figure 5.8: Throughput as a Function of Probability Space

5.2.3.2 Average Processing Time of PCEP

Hypothesis with respect to the result of the parameter A: (Average Processing Time)

Null hypothesis H0: A1 = A2 = A3, where A1= Average Processing Time in Distributed Cayuga, A2 = Average Processing Time in PCEP 5 and A3 = Average Processing Time in PCEP 10.

(There is no significant difference among the three systems in terms of Average Processing Time obtained).

Alternate hypothesis H5: Average Processing Time mean values are not equal for at least one pair of the result mean values of the parameter A.

(There is a significant difference between the three systems in terms of Average Processing Time obtained)

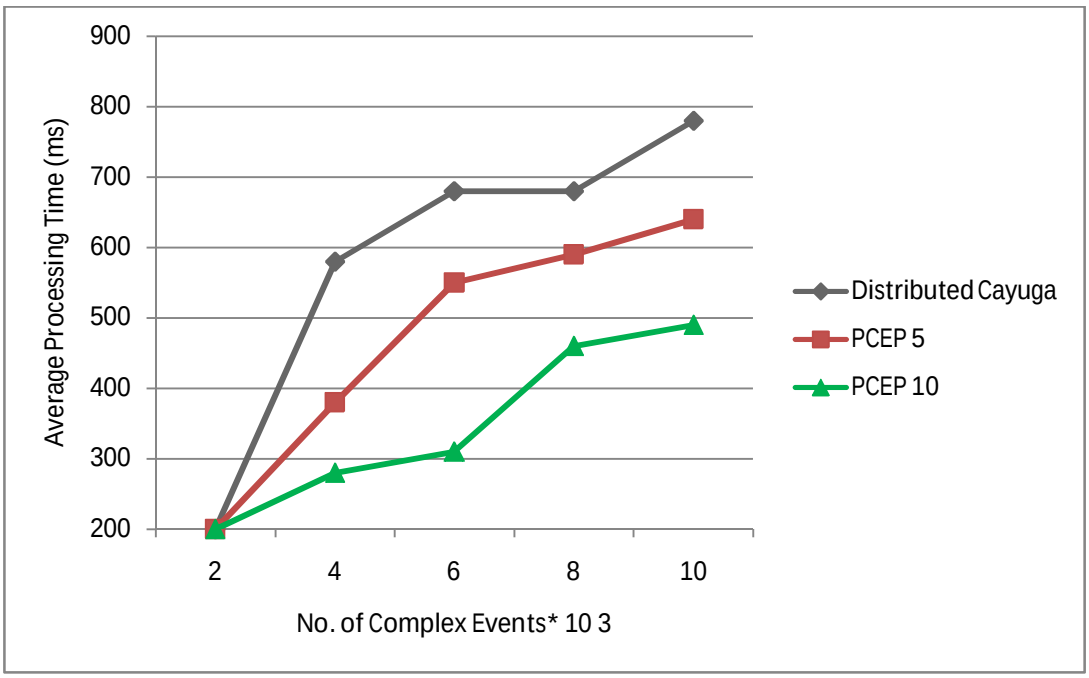

Figure 5.9: Average Processing Time for varying NFA Sequence Length

In the result, the PCEP system significantly decreases the Average Processing Time over the other existing CEP schemes such as distributed Cayuga. Furthermore, the number of state machines available in the system decides the elapsed time to process the incoming events. In case of more number of state machines, it can manage a large number of incoming events with reduced complexity that may reduce the Average Processing Time required to process the incoming events. Figure 5.9 depicts the Average Processing Time of the CEP. The PCEP system achieved less processing time if more state machines are deployed to process the incoming events. The figure also illustrates that the Average Processing Time of the distributed Cayuga scheme starts getting higher after 60,000 incoming events, indicating the bottleneck in performance due to the large number of incoming events. It can be eliminated by deploying an appropriate number of state machines according to the number of incoming events.

Table 5.5: Average Processing Time versus State Machines

Scheme	Distributed Cayuga	PCEP 5 Machines	PCEP 10 Machines
Processing Time (ms) Ω	240- 798	200-640	200-490

Hypothesis Evaluation with respect to parameter A: (Average Processing Time) as function of NFA length

Table 5.6: ANOVA for Average Processing Time as Function of NFA Length

Events x 10^3/sec	Technique			Average Processing Time	
	I	II	III	Hypothesis	p value
1-10	Distributed Cayuga	PCEP 5	PCEP 10	H5	0.016

From Table 5.6, it is concluded that the calculated significance level of the parameter Average Processing Time of comparing three systems of Distributed Cayuga, PCEP 5 and PCEP10 does satisfies the condition (p value<0.05) for input queries. There is a significant difference between the results for different Average Processing Time values of Distributed Cayuga, PCEP5 and PCEP10. Hence, the null hypothesis for H5 may be rejected.

Further, it is also required to determine the system which has the minimum Average Processing Time. This is analyzed using descriptive statistics given in Table 5.7. Table 5.7 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Average Processing Time.

Table 5.7: Descriptive Statistics of Average Processing Time as Function of NFA Length

Technique	Distributed Cayuga	PCEP 5	PCEP 10
Max	780	640.0	490.0
Min	200	200	200
Mean	584	472.0	348.0
Median	680	550.0	310.0
Standard Deviation	226	180.7	123.2

From Table 5.7, it is evident that there is an decrease in mean Average Processing Time values in PCEP 5 and PCEP 10 compared to Distributed Cayuga. It can be concluded that an increase in NFA length affects the mean processing time. There is a decrease in Average Processing Time for PCEP 5 and PCEP 10 with respect to Distributed Cayuga by 19.17% and 40.41% respectively.

5.2.3.3 Scalability of PCEP

Hypothesis Evaluation with respect to the result of the parameter S: (Scalability)

Null hypothesis H0: S1 = S2, where S1= Scalability obtained in Distributed Cayuga, S2 = Scalability obtained in PCEP with Filtering

(There is no significant difference between the two systems in terms of Scalability obtained)

Alternate hypothesis H6: Scalability mean values are not equal for at least one pair of the result mean values of the parameter S.

(There is a significant difference between the two systems in terms of Scalability obtained)

Scalability is the unified performance metric. It depends on average processing time, throughput and workload. The existing probabilistic systems materialize and sort each event sequences in the possible world's space individually. Therefore, they are suitable only for a small dataset of size less than 25 tuples with limited event pattern queries.

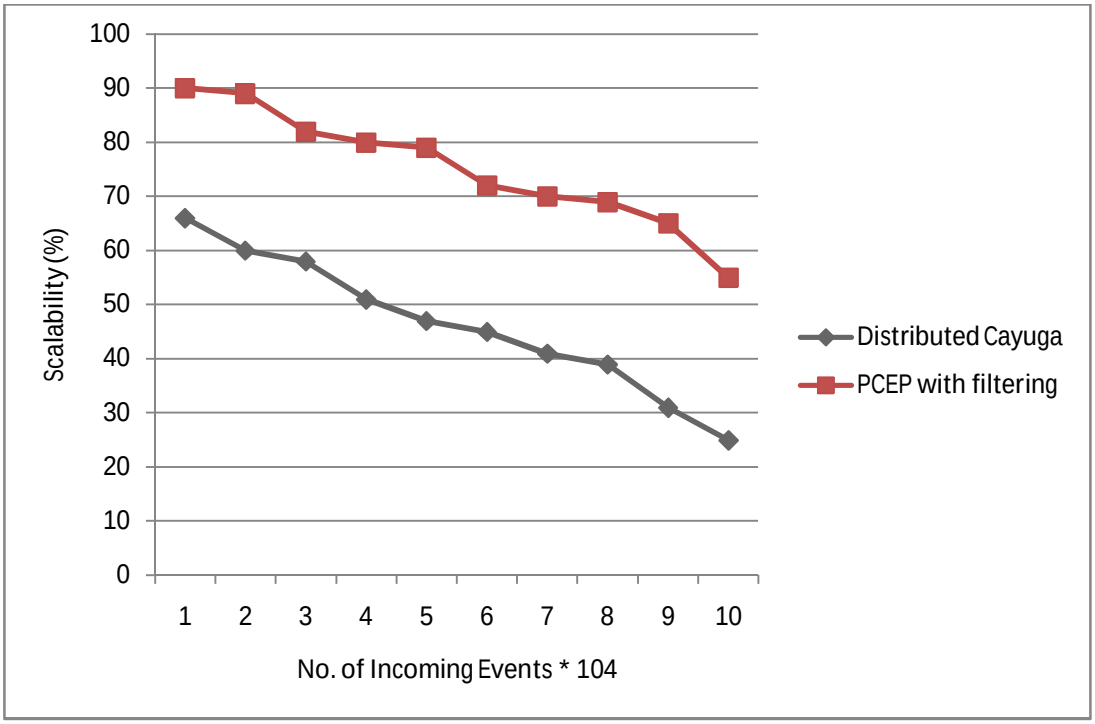

Figure 5.10: Scalability of PCEP with and without Filtering

The PCEP system deploys an efficient event filtering approach to filter and to correlate events by performing the matching between the large number of incoming events (publications) and rules. Therefore, the event sequence prediction scheme derives the stateful event sequence based on the probabilistic ranking. It achieves high scalability due to the filtering of irrelevant results. It also reduces the complexity in predicting the event sequence for the large number of incoming event streams. Figure 5.10 depicts that the scalability of the PCEP with filtering is 1.62 times greater than the Distributed Cayuga scheme. The scalability of the distributed Cayuga scheme is maintained around 60-70 % because it incurs less throughput and high average processing time for the large number of incoming events.

Hypothesis Evaluation with respect to parameter S: (Scalability)

Table 5.8: T-test for Scalability

Events x 10^4	Technique		Throughput	
	I	II	Hypothesis	p value
1-10	Distributed Cayuga	PCEP	H6	0.000

From Table 5.8, it is concluded that the calculated significance level of the parameter Scalability of comparing two systems, Distributed Cayuga and PCEP with filtering always satisfy the condition (p value<0.05) for input events ranging from 10,000 to 1,00,000. There is a significant difference between the results for different Scalability values of Distributed Cayuga and PCEP with filtering. Hence, the null hypothesis for H6 may be rejected.

Further, it is also required to determine the system which has the maximum Scalability. Table 5.9 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter scalability. It shows a increased Scalability in PCEP with filtering.

Table 5.9: Descriptive Statistics of Scalability Measures

Technique	Distributed Cayuga	PCEP with Filtering
Max	66.00	90.00
Min	25.00	55.00
Mean	46.30	75.10
Median	46.00	75.50
Standard Deviation	12.94	10.96

From Table 5.9, it is evident that there is an increase in mean Scalability values in PCEP with filtering compared to Distributed Cayuga by 62.20%.

5.3 SUMMARY

This chapter has proposed a probabilistic framework named as PCEP that performs complex event detection even in the presence of uncertain data. In the PCEP, the filtered events enter into a probabilistic event sequence prediction engine to derive the correlation between the relevant events. Then, the system transforms the relevant events into new composite events as output. Here, the probability space is constructed as DFPRM model in the form of event hierarchy. To reduce the large sample space, a novel probability Fuzzy Logic is used to derive the most probable events using approximation consistent mathematical formulation. The performance evaluation shows that the PCEP system is the most effective and efficient that achieves high scalability even under the large number of incoming events. The performance of PCEP against the existing Distributed Cayuga is statistically analyzed using ANOVA and T-Tests.

CHAPTER 6

PCEP IN SMART REAL-TIME USE CASES

The developed PCEP system is verified for consistency in performance. The system is tested in processing events from RFID management systems, stock market trading and business activity monitoring. The probabilistic framework deals with uncertainty and results in efficient CEP. Further, event filtering is performed to filter the relevant events for achieving high scalability.

6.1 RFID PCEP IN A REAL-TIME PRODUCT MANUFACTURING

The PCEP system performs complex event detection in RFID monitoring for product manufacturing. It integrates CEP to detect the in-sequence error, incorrect assembly and delay identification in the large number of incoming parts (events) from the various sections. It is used to detect the misplaced or duplicate events effectively through real-time analysis and also notifies about the exceptional events to the responsible technical supervisors to trigger a needy action in a timely manner. Figure 6.1 illustrates the CEP based RFID model in production manufacturing system.

The products used to manufacture the car are entered into the PCEP system through production path phase. This phase consists of two main components i) RFID readers ii) RFID processor that will be used for the RFID based product manufacturing [95]. The parts from the production phase to be assembled enter into the assembly point through the CEP core module for the car manufacturing. This module filters the relevant parts [96] for the car manufacturing and discards the duplicate parts based on the Complex Event Processing Language (CEPL) queries devised by the technical experts in the factory.

Further, PCEP is carried out to predict the uncertainty in the complex events. Finally, it will automatically notify about the inferred events as alerts to the technical supervisors to initiate corrective measures in order to improve the overall performance.

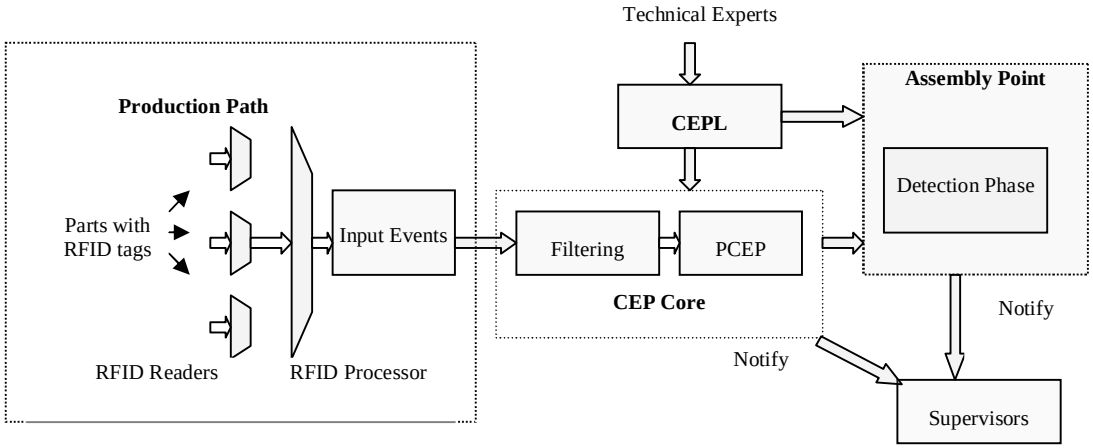

Figure 6.1: CEP based RFID Model in Product Manufacturing System

6.1.1 Production Path

In a real-time smart factory, the products moving through production lines meet at certain assembly points where assembling is carried out. The production path is the starting point for the architecture. The architecture consists of two main components such as RFID readers and RFID processor. The RFID readers read the product information from RFID tags pasted in the product. The PCEP system does not directly take the raw incoming data streams as input. The data streams are not adaptable for the internal processing. The RFID Input Processor collects the large number of incoming events read by RFID readers. The RFID Input Processor converts the events into the form of event tuples as shown in Figure 6.2, which are suitable for the CEP [97]. For instance, the RFID reader reads raw data streams in the form of <r, i, t>, where 'r' represents the RFID reader that collects information 'i' from the RFID tag at a time 't' [98]. The RFID Processor converts the raw data streams into meaningful and actionable RFID event formats which are represented as follows:

```
{Type= “Tyre-TT” , Name= “Transtone” , Model = “12R22.5” , Design= “RADIAL” };
{Type=  “Brake” , Name= “Detroit”, Model= “SER 12.7” , Spacers ID= “#790”};
{Type= “Engine” , CC= “1196” , Model= “4A-GEC14” , Fuel= “Petrol” , Milage= “116kmph”};
```

Figure 6.2: RFID Event Stream

6.1.2 Complex Event Processing Language (CEPL)

In order to process the incoming event streams in the production path ,the technical experts express their required event patterns. They are represented using the CEPL query language. CEPL language is a query language used to represent the event patterns under uncertainty as illustrated in Figure 6.3. CEPL provides declarative, expressive and sequence-based language that can express the attributes and values of the event in the form of algebraic expressions. In addition, CEPL provides a powerful support for RFID data processing that includes data filtering, data transformation, aggregation and real-time monitoring. It consists of three clauses named as SELECT, FROM and WITHIN [99]. Among the three clauses, the SELECT clause specifies the attributes required to select from the incoming event stream which may be either uncertain or certain . The CEPL to detect the part engine with relevant specifications suitable to manufacture the car Ford-Figo is as follows:

```
SELECT Part Type, CC, Mileage, Part Model, Fuel Type
FROM FILTER ({Type = “Engine”} (RFID Event Stream))
PATTERN ($ Model = “4A-GEC14”, CC = “1196 kmph”, Fuel = “Petrol”)
IF true
        DO allow →Assembly point
ELSE
        Set notify →Technical supervisor
WITHIN 15Min;
```

Figure 6.3: Complex Event Processing Language

Figure 6.3 shows the CEPL with event specification to trigger the action with IF and ELSE condition [98]. When the condition specified in IF condition is satisfied then, the system allows the part into assembly point. Otherwise, the action specified in ELSE condition is triggered for notification. Depending upon the specified attributes of the SELECT clause, two different variants of the FROM clause are declared. The FROM clause is the core part of the query that defines the stream expression to process a large number of incoming event streams. A stream expression is formulated in the form of predicates using the unary constructs -FILTER and PATTERN [100] [101]. The FILTER is represented as {FILTER (θ)} which takes input as events from the incoming event streams and then only selects the events which satisfy the predicate θ, whereas PATTERN is represented as {PATTREN (θ_1, θ_2……θ_n} which is same as FILTER construct, but takes more than one inputs.

6.1.3 Probabilistic Complex Event Processing

The core of the approach is the complex event processor that performs event processing to detect the complex events with the help of two phases - Event Filtering and Probabilistic CEP.

6.1.3.1 Event Filtering

The Event Filtering filters the events which satisfy the FILTER construct of the CEPL query and then discards the remaining events as duplicate, redundant and abnormal RFID events from the incoming events. Event Filtering is done based on the query formulated by the technical experts. In the production path, many product parts move towards the assembly point. From these parts, the irrelevant parts are filtered based on the query formulated. The relevant product parts are forwarded to the assembly point. Due to the filtering of irrelevant parts, the system can manage the large number of incoming events. This process improves scalability [102]. The Probabilistic Event Sequence Prediction (PESP) thus predicts the relevant parts.

6.1.3.2 Probabilistic Complex Event Processing

Due to the uncertainty caused by data errors, the events are probabilistic rather than deterministic in nature. Therefore, a probabilistic model is proposed to enable complex event detection in the presence of uncertainty. The PCEP system detects the duplicate or exceptional events on the production path and then notifies the technical experts about the most important issues. The probability of the events is computed based on the correlation among the different attributes of the different events from the event history. The event history consist of 'n' number of events $\{E_1, E_2 \ldots\ldots E_n\}$. Then, the overall joint probability distribution is computed based on aggregating the product of local conditional probabilities of the events using the conditional probabilistic dependencies. The overall joint probability distribution represented as follows:

$$P(E / E_n) = \sum_{1}^{n} P(E \cap E_n) / P(E_n)$$

After computing the conditional probability of the events**,** the probabilistic ranking is performed to rank the event based on the conditional probability . Furthermore, an event with a high state probability is triggered and detected as a suitable event. This phase generates events according to the detected complex event based on probabilistic phase. If the detected event matches the pattern construct in the CEPL, then the system allows the detected event into the assembly point for further processing. Otherwise, the system generates a notification as an alarm, SMS or email to the technical supervisor. The technical supervisor takes the corrective measure to achieve fault recovery and to prevent this error in the future to improve the overall performance.

6.1.3.3 Event Detection Phase

There is a possibility for unfortunate situations to occur, even after entering the relevant parts to assemble the car in the assembly point. The unfortunate situations are i) there may not be synchronization among the detected parts while entering into the assembly point ii) The product parts moving on the assembly line

may arrive late at the assembly point and may result in an incorrect assembly of products or an overall delay for the entire production. In order to overcome these difficulties, three types of complex event detections such as i) synchronization error detection, ii) delay detection and iii) incorrect assembly detection will be performed in the assembly point section[103].

i) Synchronization Error Detection

In order to assemble the car effectively, the detected different product parts should arrive at the assembly points within the defined time. The parts of a product should reach the point of the assembly in synchronization with each other. A delay or break up of this clockwork precision may lead to incorrect assembly of the product. The RFID reader sends read event to the virtual reader (R_i) responsible for monitoring the corresponding product part. If R_i is not able to detect the corresponding part within a defined time span, it generates a synchronization error and sends the notification information to the supervisor [95].

ii) Delay Detection

Delays can be caused by mechanical or human failures. In this framework, a delay is detected using a timer triggered an event that is fired once in 't' seconds. The production is performed according to the plan of technical experts where each timer detects the new different product parts for assembling the car in the assembly line. If a new part is not detected on the assembly line, a new timer event is fired and a delay event is generated. It generates the notifications.

iii) Incorrect Assembly Detection

When two parts that do not belong to the same product are assembled together, an incorrect assembly occurs. When a product part is assembled with another part, both of them will be detected at the same time at the next physical reader. If a physical reader detects two parts, its corresponding virtual reader compares these two parts with the product to which they belong. If both of these

parts belong to the same product, then it triggers an event of correct assembly detection. However, if both the parts do not belong to the same product, the virtual reader generates an incorrect assembly error event and notifies the supervisor. An incorrect assembly error may be arising due to the delays on assembly lines or synchronization errors [95].

6.1.4 Experimental Setup

The system is implemented in Siddhi CEP engine with the help of WSO2 Complex Event Processor. The experiment is run on Windows XP PC with 3.2 GHz processor, 2 GB of RAM with the minimum JAVA heap size of 250 Mbytes. It can be run in Windows XP and requires a Java Virtual Machine version 5.0 runtime or above. It can process in the order of 10,000 (events/sec) on a dual CPU 2GHz Intel based hardware with minimum latency.

The performance of the PCEP system is evaluated using two common evaluation metrics such as processing time and throughput. The experiment was carried out to compare the performance of the proposed RFID-PCEP with existing RFID-CEP approach.

i) Throughput

The Throughput of the CEP scheme is defined as the number of events processed per second by the processor. It is demonstrated in Figure 6.4. The incoming events are processed based on the rules specified by the technical experts. In RFID-CEP, filtering is not performed. As the number of events increases, it cannot process the increased number of events, which decreases the Throughput. RFID-PCEP provides filtering approach which filters out the irrelevant events and processes only the most relevant events. As the number of events increases, the Throughput also increases.

Hypothesis Evaluation with respect to the result of the parameter T:(Throughput)

Null hypothesis H0: T1 = T2, where T1= Throughput obtained in RFID-CEP, T2 = Throughput obtained in RFID-PCEP

(There is no significant difference between the two systems in terms of Throughput obtained)

Alternate hypothesis H7: Throughput mean values are not equal for at least one pair of the result mean values of the parameter T.

(There is a significant difference between the two systems in terms of Throughput obtained)

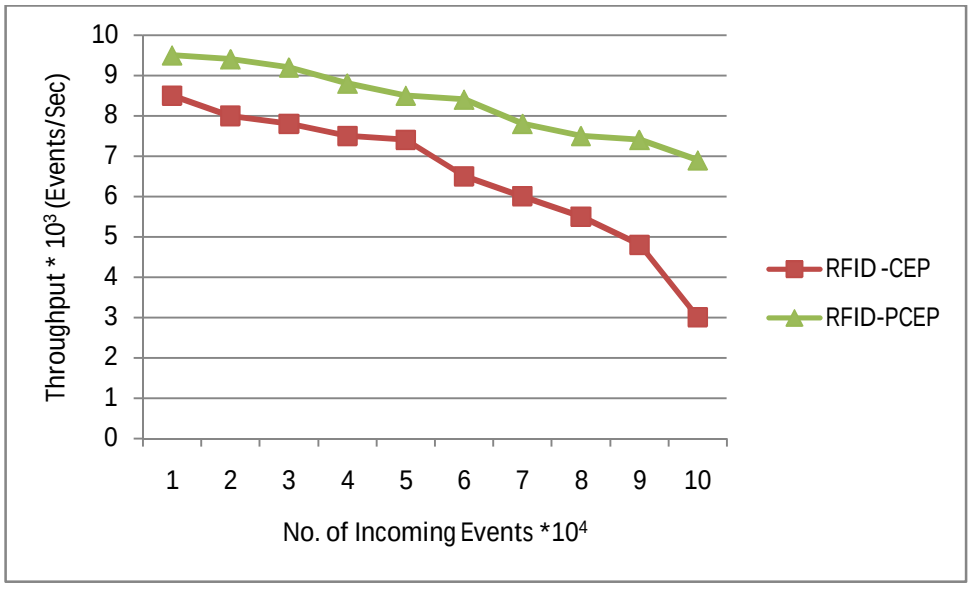

Figure 6.4: Throughput of RFID-PCEP System

Table 6.1: T-test for Throughput of RFID-PCEP System

Events x 10^4/sec	Technique		Throughput	
	I	II	Hypothesis	p value
1-10	RFID-CEP	RFID-PCEP	H7	0.007

From Table 6.1, it is concluded that the calculated significance level of the parameter throughput of comparing two systems RFID-CEP and RFID-PCEP always satisfy the condition (p value<0.05) for input events ranging from 10000 to 100,000. There is significant difference between the results for different throughput values of RFID-CEP and RFID-PCEP. Hence, the null hypothesis for H7 may be rejected.

Table 6.2: Descriptive Statistics of Throughput Measures of RFID-PCEP System

Technique	RFID-CEP (Existing System)	RFID-PCEP (Proposed System)
Max	8.500	9.500
Min	3.000	6.900
Mean	6.500	8.340
Median	6.950	8.450
Standard Deviation	1.704	0.907

Further, it is also required to determine the system which has the maximum Throughput. This is analyzed using descriptive statistics given in Table 6.2. Table 6.2 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Throughput. It is evident that there is an increase in the mean Throughput of RFID-PCEP with respect to RFID-CEP by 28%.

ii) Average Processing Time

The Average Processing Time of the proposed PCEP approach is calculated based on the time taken to process the complex events. Figure 6.5 demonstrates that the proposed approach achieves better Average Processing Time than the existing RFID-CEP without filtering.

Hypothesis Evaluation with respect to the result of the parameter A: (Average Processing Time)

Null hypothesis H0: A1 = A2, where A1= Average Processing Time in RFID-CEP, A2 = Average Processing Time in RFID-PCEP

(There is no significant difference between the two systems in terms of Average Processing Time obtained)

Alternate hypothesis H8: Average Processing Time mean values are not equal for at least one pair of the result mean values of the parameter A.

(There is a significant difference between the two systems in terms of Average Processing Time obtained)

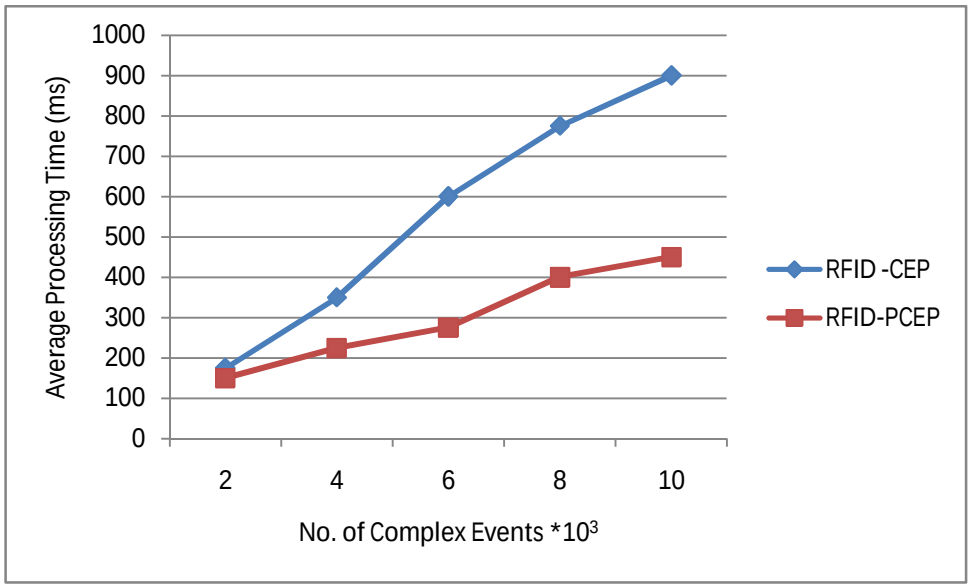

Figure 6.5: Average Processing Time of RFID-PCEP System

Table 6.3: T-test for Average Processing Time of RFID-PCEP System

Events x 10^4/sec	Technique		Average Processing Time	
	I	II	Hypothesis	p value
1-10	RFID-CEP	RFID-PCEP	H8	0.109

From Table 6.3, it is concluded that the calculated significance level of the parameter average processing time of comparing two systems RFID-CEP and RFID-PCEP does not satisfy the condition (p value<0.05) for input events. There is no significant difference between the results for different average processing time values of RFID-CEP and RFID-PCEP. Hence, the null hypothesis for H8 may be accepted.

Table 6.4: Descriptive Statistics of Average Processing Time of RFID-PCEP System

Technique	RFID-CEP (Existing System)	RFID-PCEP (Proposed System)
Max	900	450
Min	175	150
Mean	560	300
Median	600	275
Standard Deviation	298	123.7

Further, it is also required to determine the system which has the minimum Average Processing Time. This is analyzed using descriptive statistics given in Table 6.4. Table 6.4 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Average Processing Time. It is evident that there is a decrease in the Average Processing Time of RFID-PCEP with respect to RFID-CEP by 46%.

6.2 UNCERTAIN STOCK PRICE PREDICTION USING PFL

The stock prediction system is modeled as a Publisher/Subscriber middleware [82]. It is represented in the Figure 6.6. It can predict the stock value for stocks of companies across various sectors of the economy like Automobiles, IT and Oil & Gas.

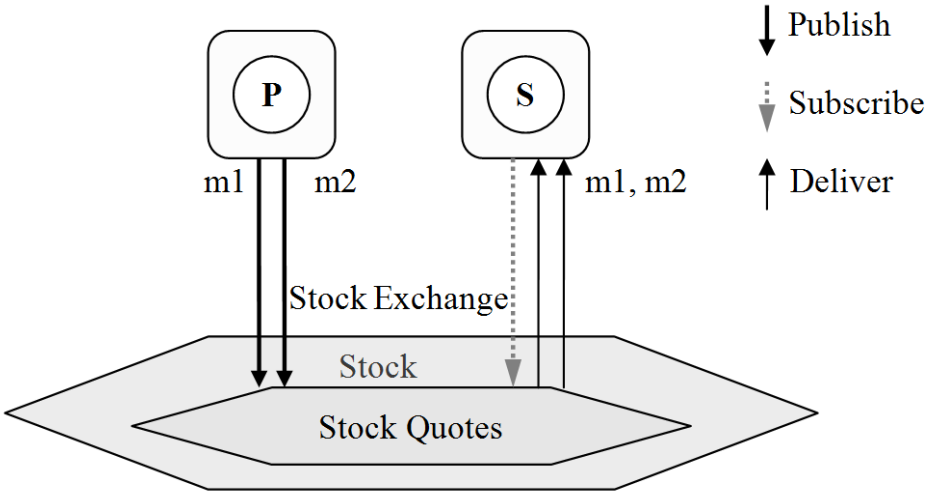

Figure 6.6: Stock Exchange Publisher/Subscriber Middleware System

In Figure 6.6, the publisher (stock exchange) publishes stock quotes to a large number of subscribers (stock broker, traders and investors). The stock quotes consist of events related to the various sectors. They have five attributes - a global identifier, the name of the company, the volume of stocks, the price and the identifier of the selling trader. The subscribers those who are interested in stock trading can express their interest on the incoming events as a stock request to buy or sell a share in accordance to the event patterns available in the stock market. The Probabilistic Fuzzy Logic methodology predicts the future price of stock value by extracting the higher level knowledge from the large number of incoming complex events from the stock quotes.

6.2.1 Stock Exchange Scenario

In the stock market, all companies in the different domains have some business dependencies to each other. For example, a manufacturing company is strongly inter-correlated with a company for production of raw material and a finance company. This causes business dependencies among multiple companies. When the stock broker needs to invest he should comprehend this dependency chain. So he should monitor for events occurring along multiple dependency chains.

The stock trading consists of two types of stock events - stock quote and stock request. The stock exchange disseminates the number of continuously arriving stock data as a stock quote as depicted in Figure 6.7. The stock brokers use stock request to express their interest in buying/selling stock [104]. The publishers publish the continuously arriving stock exchange stream as follows:

{…….
{(Name, “TATAMOTORS”) (price, 174.88) (volume, 2,735)},
{(Name, “FORD”) (price, 69.31) (volume, 8,991)},
{(Name, “FIAT INDIA LTD”) (price, 189.31) (volume, 1,481)},
{(Name, “HONDA MOTORS”) (price, 87.84) (volume, 6,165)},
{(Name, “TOYOTA MOTORS”) (price, 91.36) (volume, 5,742)},
{(Name, “MARUTI SUZUKI”) (price, 150.36) (volume, 3,354)},
{……..

Figure 6.7: Stock Exchange Event Stream

The stock broker or trader will like to start a query on the event stream as a stock request similar to the query as represented in Figure 6.8. The stock brokers express their interest as event queries, represented in the form of CEPSL [60]. The broker wants to invest share on car company for the product of monotonic decrease of stock price less than 90 with volume of 5,000. Due to the decrease in stock price, the stock abruptly rebounds through increasing of up to at least 5% in value in last 15 minutes.

```
SELECT Company name, Low Price from Stock quote
        RS = FROM FILTER {DUR <15min} (Stock quote))
        FOLD {$2 Name = $ Name, 4,500≤ volume ≤ 5, 000} (Stock)
        NEXT {$2Name = $1Name} (Stock)
FROM FILTER {85 ≤ Stock Price ≤ 90} (Stock)) from RS
        FOLD {Final Price <1.05* Low Price, DUR<15min} (Stock)
        NEXT {$2Name = $1Name} (Stock)
        WITHIN 15Min;
```

Figure 6.8: Stock Request

6.2.2 Stock Prediction using DFPRM

From the web, the historical stock (open, close, low, high) prices of different companies hare collected. The event hierarchy called as Dynamic Fuzzy Probabilistic Relational Model (DFPRM) is constructed [86] as portrayed in the Figure 6.7. The DFPRM consists of set of event classes with their associated attributes. The joint probability distribution is computed using the conditional probabilistic dependencies among the event sequences in accordance to the stock request. The constructed graphical DFPRM model is used to learn the non-linear and dynamic functional relationships among the incoming events. DFPRM predicts the future trends of the stock market.

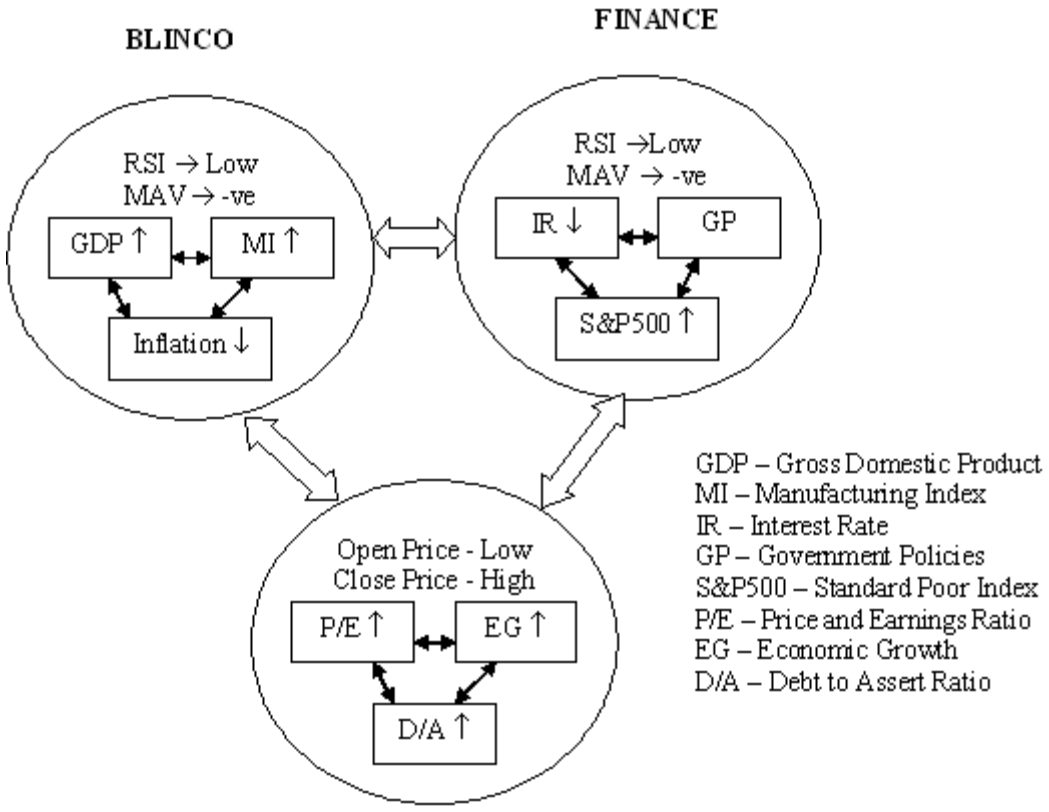

Figure 6.9: Dynamic Fuzzy Probabilistic Relational Model

This model has capability to perform the technical analysis on the incoming stock quotes based on different parameters to predict the future. This model is used by the investor for evaluating the opening price, closing price, highest

price and lowest price of a company stock based on the technical indices- Relative Strength Index (RSI), Gross Domestic Product (GDP), Manufacturing Index (MI) and Interest Rate (IR) and the fundamental indices namely Price-to-Earnings Ratio (P/E), Price-to-Sales Ratio (PSR), Return On Equity (ROE), Earnings Growth (EG)and Debt-to-Asset ratio (D/A). Figure 6.9 shows the DFPRM for the event sequences 'E' with a set of events 'E' = $\{e_1, e_2 \ldots .e_n\}$ and each event 'e_i' is associated with a set of descriptive attributes and reference slots. The sample space of event sequence 'E' is represented as the conjunction of set of possible stock events with its associated probability measure from the stock event history.

6.2.3 Conditional Probability Computation of Stock Events

The novelty of work is that the constructed probabilistic model (DFPRM) is used to compute the probability of future stock price based on the conditional probabilistic independencies among the events in stock history. The probability space is represented as triple $\{W_T, \Omega_T, \theta_T\}$ where W_T is a set of possible stock values, Ω_T is a history associated with each possible stock and θ_T is a probability measure of the stock value. The conditional probability distribution of event e_i is determined using the probability distribution over the values of events given each combination of stock values in its stock history $P(e_i)$. The conditional probability distribution is computed as follows:

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

Here, Ω_i is the set of stock values in stock history of e_i and Φ_i is the parameter vector associated with e_i. The overall probability associated with the stock events is factorized by aggregating the product of local conditional dependence of the event given their stock history events [72].

6.2.4 Probabilistic Fuzzy Logic based Inference Engine

In order to make a better decision whether to buy or sell the share for making a profit in the stock market, Probabilistic Fuzzy Logic [90] estimates the fuzzy linguistic variables from the computed conditional probability distributions in

the large probability space. The fuzzy partitioning scheme reduces the large sample space by partitioning the more number of possible worlds into pre-determined classes using membership function. A fuzzy system consists of set of rules defined by the characteristic function called as membership function, which is represented as μ_F: P (E) → [0, 1] [91] [92]. It is used to define the certainty that element P (E) belong in that fuzzy set 'F'. The performance of fuzzy logic is influenced by the selection of membership functions and the fuzzy logic rules.

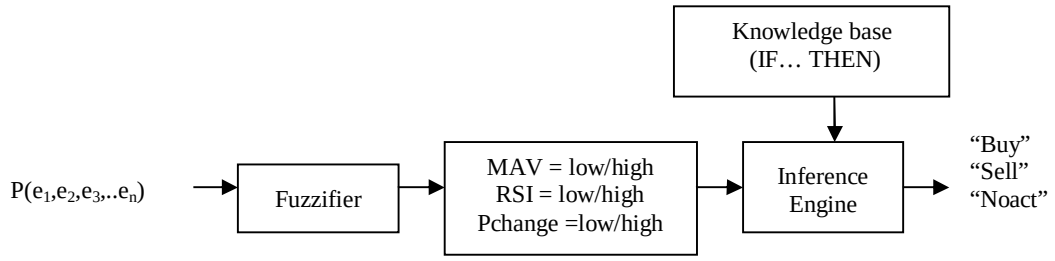

Figure 6.10: Probabilistic Fuzzy Logic for Stock Prediction

Figure 6.10 shows the Probabilistic Fuzzy Logic used in DFPRM model in which the Fuzzifier converts the input probability of the possible worlds (non fuzzy) of a sample space into fuzzy values [93] to train the DFPRM model. Further, the knowledge base consists of fuzzy if-then rules. The set of statements listed below give an idea about the rules that are implemented:

→ If (MAV is negative) and (%price change is Positive) and (Low RSI) THEN "BUY" (Tomorrow close price value > than today's price)

→ If (MAV is negative) and (%price change is Positive) and (High RSI) THEN "SELL" (tomorrow close price < than today's price)

→ If (MAV is null) and (%price change is null) and (Stable RSI) THEN "NO ACTION" (tomorrow close price value remains same as today's price).

The inference engine executes the fuzzy logic to map any one of the linguistic variable from the fuzzy sets. The fuzzy sets consist of three linguistic variables {"Buy", "Sell", "No Action"} according to the rule base. Finally, the

satisfied condition throws the corresponding linguistic variable as the output whether to buy or sell the share to gain profit. In addition to daily stock prediction, the system is also capable of predicting the open, high, low and close prices of desired stock on a weekly and monthly basis. It is proposed for predicting the future direction of stock prices using past historical datasets. The PFL approach enhances the decision making for investors in the stock market by offering more accurate stock prediction compared to existing approaches. The experimental evaluation is carried out to test the methodology under real-time financial data.

6.2.5 Experimental Setup

The PCEP system for stock price prediction was tested with the automobile stocks of TATA motors, Honda motors, Toyota motors, Fiat India Ltd and Ford. The data used for evaluating this approach was obtained from the website www.moneycontrol.com that provided the stock prices prevailing at NASDAQ stock quotes. The data was collected for the period from July 2012 to April 2013. From the collected data, the opening, highest, lowest and closing values of the stock price for each day within this period was collected. A publisher/subscriber system using Java Message Service based subscription API in the environment of Java Enterprise Edition/NetBeans/Apache/JMS was implemented. It takes incoming events as inputs of various company stocks. The system generates suitable output to indicate the type of decision suitable for the investor.

The performance of the system is evaluated using two common evaluation metrics accuracy in predicted stock price versus actual price and Mean Absolute Percentage Error (MAPE).

i) Accuracy of Stock Prediction

The stock price prediction system performs effective prediction in the stock price. The Figure 6.11 shows that the accuracy of stock prediction over a period from July 2012 to April 2013. The performance of PCEP system in stock for Throughput and Average Processing Time is already discussed in chapter 5.

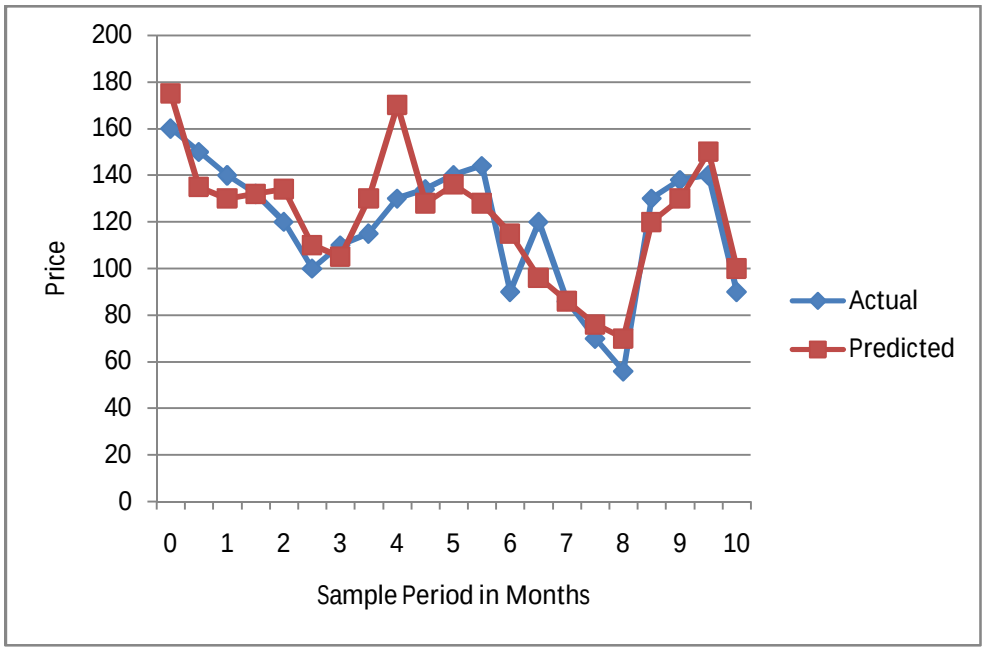

Figure 6.11: Accuracy of Stock Prediction

ii) Mean Average Error Percentage in Stock Prediction

The error rate of PFL approach of NASDAQ stock is calculated in terms of mean square error which is represented in Figure 6.12.

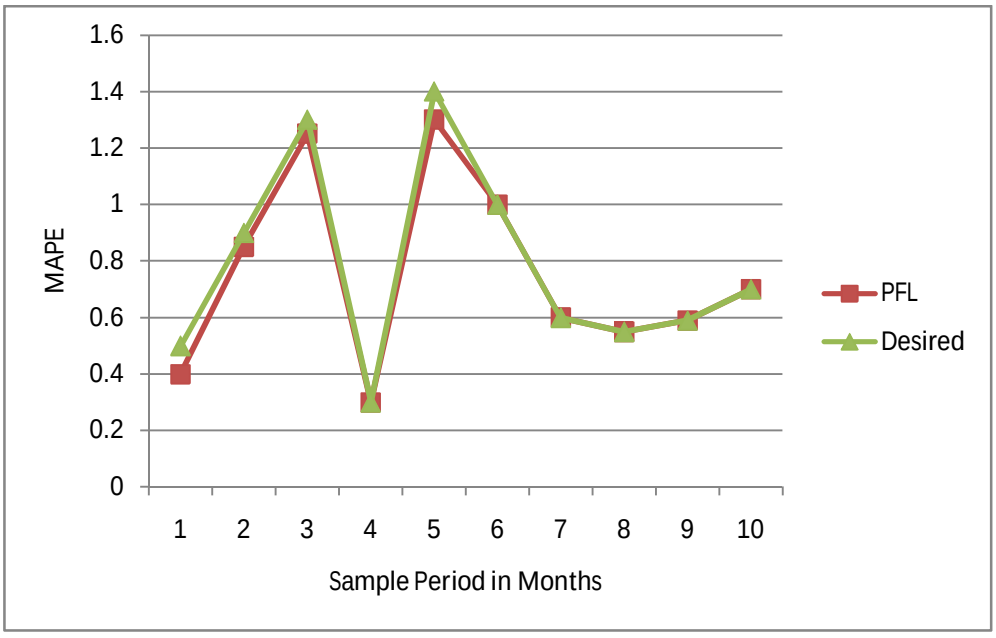

Figure 6.12: Mean Average Error Percentage in Stock Prediction

6.3 KPI BASED CEP AND CBAM

The PCEP system is Complex business Activity Monitoring with Event Processing (CAMEP). CAMEP combines two main platforms such as CEP and BAM into a single framework. It is an integrated end-to-end developer environment that performs real-time analysis on the input events under uncertainty. Figure 6.13 shows the four-tier architecture of CAMEP which consists of four layers: Event Collection layer, Event Filtering Layer, Complex Event Processing Layer and Event Display or Delivery Layer.

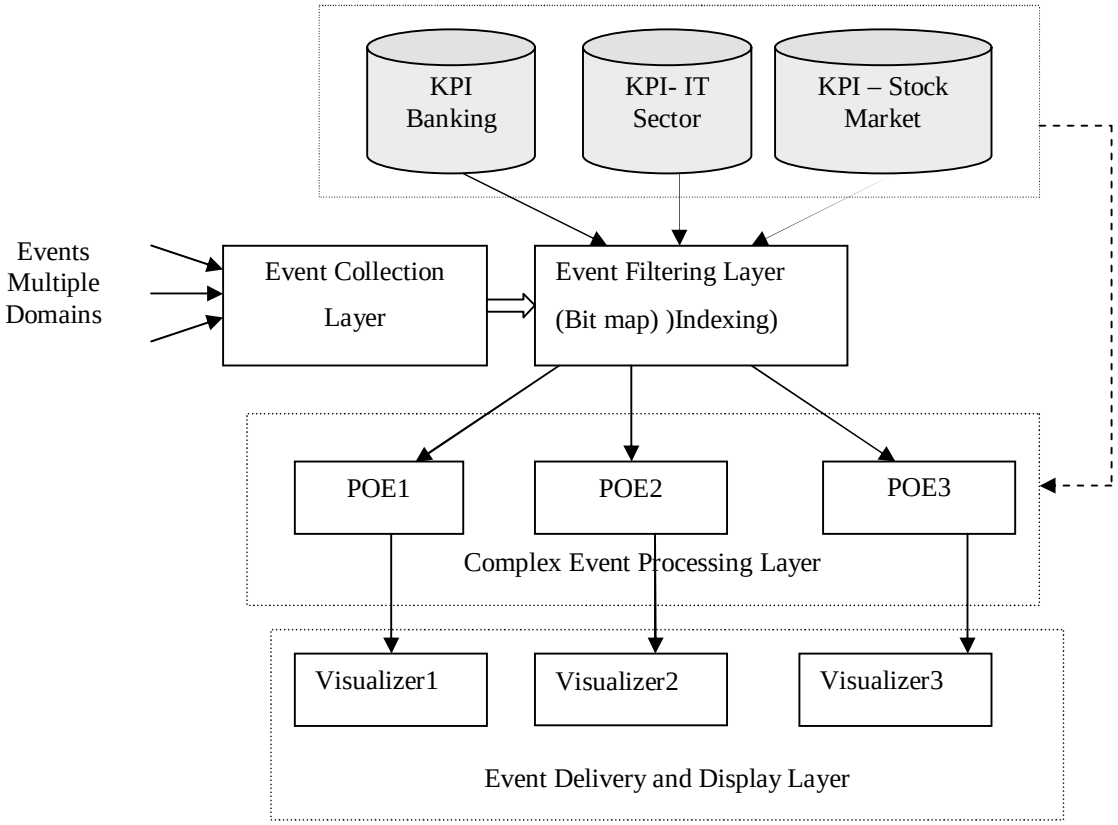

Figure 6.13: CAMEP in Real -Time Enterprise of Multiple Domains

In this CAMEP architecture, the first layer is the Event Collection Layer that collects a large number of heterogeneous events from multiple domains such as banking, stock market and IT sector. It relies on object oriented middleware layer to collect the messages from heterogeneous transactional systems. The second layer is the Event Filtering Layer employs a bit map indexing scheme to filter the incoming

events based on the Key Performance Indicators. This layer filters the relevant events which are related KPIs mentioned for the specified domain and then discards the exceptional events which are unrelated to the domain. The next layer is the Complex Event Processing Layer; where the relevant events are entered into this processing layer that deploys a Persistent Object Engine (POE) to perform CEP under uncertainty. It identifies the divergence in the business performance according to the concurrent values in KPIs. The Event Delivery and Display layer is the final layer which activates automated action to help Business Process Management(BMP) tools. Further, it generates and communicates alerts via various communication systems to ensure the reaction to performing changes to improve the business performance.

6.3.1 Event Collection Layer

In this CAMEP model, the first layer is the Event Collection Layer. Owing to the heterogeneity of the incoming events coming from multiple domains, the CAMEP approach does not directly take the raw incoming data streams as an input. It collects messages from heterogeneous transactional systems. The incoming messages in the middleware are aggregated into events. Therefore, an object oriented middleware layer is deployed to manage all the messages that enter into the system and to convert the events as objects suitable for continuous real-time analysis. The incoming event messages are pre-processed into event instances, which are suitable to process the events in upcoming layers.

6.3.2 Selection of KPIs

The event-driven CAMEP approach supports a universal system architecture that provides the high abstraction and projection capabilities for the selection of KPIs, which is used to monitor the business process continuously at run time. Domain experts utilize Business Metric Service (BMS) to select suitable KPIs and to calculate the numerical values for the set of attributes in the KPIs to maintain the business process**.** BMS is a specialized business service that provides relevant information based on rapidly accessing the wide range of data not being intertwined

with the technical information of the underlying business process. Furthermore, the derived KPIs are stored in a persistent database so that multiple users can access the KPIs without incurring the expense of recalculation.

6.3.3 Event Filtering Layer

Events collected and pre-processed by the Event Collection Layer are entered into the Event Filtering Layer. It filters the incoming events/messages which are related to the KPIs specified by domain experts. The PCEP system deploys an inbuilt analytical model called as a transactional bitmap indexing. It performs extremely fast filtering of the large number of continuously arriving events based on the KPIs [21] [22]. Here, the formulated KPIs are organized in the form of matrix 'u' x 'v' where 'u' is the set of identifiers (object) and then 'v' is the corresponding values (property) that are maintained as an object. When a bit of the corresponding row is set to 1, it means that the row/object has that value for the column/property. The Bitmap Indexing performs Boolean operations (AND, OR) on the indexes to determine exactly which event instances match the attribute in the KPIs without searching throughout the entire database.

6.3.4 Complex Event Processing Layer

The Complex Event Processing Layer accepts the filtered events from the filtering layer. This layer considers the uncertainty associated with the set of attributes of the incoming filtered events. This layer consists of a Persistent Object Engine (POE) which has two highly integrated components – a virtual machine and an object store. The object store maintains all the incoming messages entering the system persistently which also increases system reliability [105] The virtual machine deploys event-driven logic so that processes can react rapidly to the number of incoming events from the multiple heterogeneous domains. The monitoring of the business process is carried out based on the KPIs to detect exceptional events. If the numerical values of the KPIs in the incoming events exceed the specified range, then corrective reasonable action must be a triggered to take over the enterprise to a good position.

6.3.5 Event Delivery and Display Layer

After analyzing and processing the business events based on KPIs, this layer is responsible for notifying the impact and severity of the events in the business performance of the recipients. Here, the notification may be in the form of graphical display or alerts to the user about the deviation of the concurrent values in the KPIs of the incoming events. The KPIs values can be displayed on dashboards via one or more meters and then corresponding actions are triggered to improve business operations and processes. Further, it provides the notification as feedback to the relevant business process to react dynamically according to changes in the incoming events [105].

6.3.6 Experimental Setup

The CAMEP approach is implemented in open source Siddhi CEP engine that runs on a WSO2 Complex Event Processor. The hardware configuration consists of the Intel Core 2 Duo of 2.10 GHz with Memory 1.9 GB with the maximum JAVA heap size of 800 Mbytes. It is implemented in open source Java Enterprise Edition/Eclipse/WSO2. The system requires Java Virtual Machine version 5.0 or above. The CAMEP approach is implemented to monitor the large number of incoming events from the multiple domains. It can process more than 500 events/sec on a dual CPU 2GHz Intel based hardware. The performance of the proposed approach is evaluated using a stock brokerage system.

The heterogeneous sources of events in this case study are Customer, Bank, Stock Brokerage and Stock Market. One sample KPI to be calculated for this case study is Customer Order Fulfillment Time (COFT). The COFT is computed from three metrics. They are Customer Order Placement Time, Money Transfer Time and Stock Transfer Time. Of this, Customer Order Placement Time metric is sourced from Customer domain, Money Transfer Time metric is sourced from Bank domain and Stock Transfer Time from Stock Market domain. All the metrics have to be processed using Complex Event Processing to calculate the COFT. The event streams is in the order of 10,000 incoming events .The overall performance of the

business process is evaluated by monitoring the incoming streams in terms of KPI, which is derived by the technical experts in the corresponding domain.

i) Throughput

Hypothesis Evaluation with respect to the result of the parameter T:(Throughput)

Null hypothesis H0: T1 = T2, where T1= Throughput obtained in BAM, T2 = Throughput obtained in CAMEP 25 KPI

(There is no significant difference between the two systems in terms of throughput obtained)

Alternate hypothesis H9: Throughput mean values are not equal for at least one pair of the result mean values of the parameter T.

(There is a significant difference between the two systems in terms of throughput obtained)

The throughput of CAMEP 25 KPI is compared with that of BAM is illustrated in Figure 6.14.

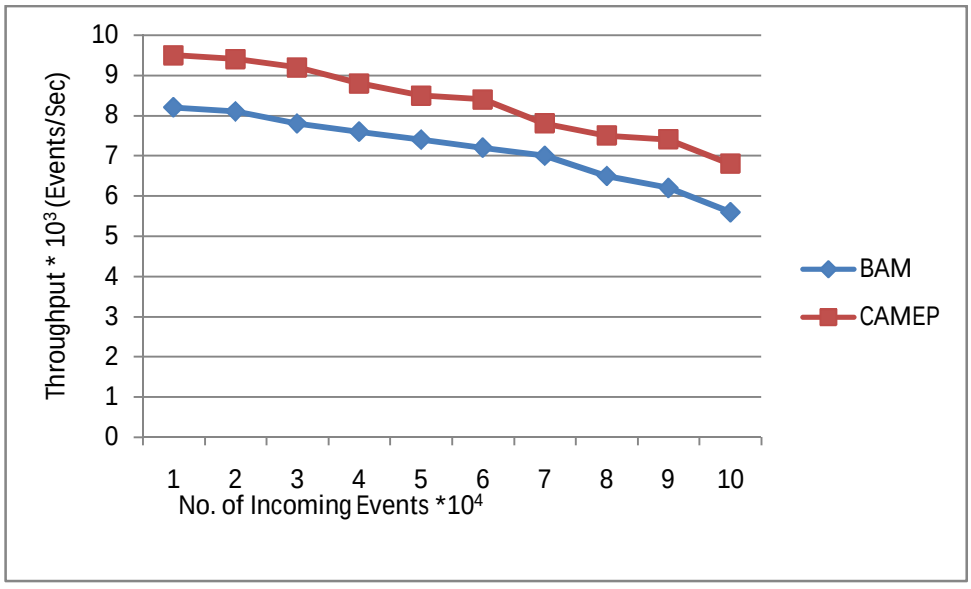

Figure 6.14: Throughput of CAMEP System

Table 6.5: T-test for Throughput of CAMEP 25 KPI System

Events x 10^4/sec	Technique		Throughput	
	I	II	Hypothesis	p value
1-10	BAM	CAMEP 25 KPI	H9	0.009

From Table 6.5, it is concluded that the calculated significance level of the parameter Throughput of comparing two systems ,BAM and CAMEP 25 KPI always satisfy the condition (p value<0.05) for input events ranging from 10000 to 100,000. There is significant difference between the results for different Throughput values of BAM and CAMEP 25 KPI. Hence, the null hypothesis for H9 may be rejected.

Table 6.6: Descriptive Statistics of Throughput Measures of CAMEP 25 KPI System

Technique	BAM	CAMEP 25 KPI
Max	8.200	9.500
Min	5.600	6.800
Mean	7.160	8.330
Median	7.300	8.450
Standard Deviation	0.846	0.925

Further, it is also required to determine the system which has the maximum Throughput. This is analyzed using descriptive statistics given in Table 6.6. Table 6.6 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Throughput. It is evident that there is an increase in the average Throughput of CAMEP 25 KPI with respect to BAM by 16.34%.

ii) Average Processing Time

The Average Processing Time taken to monitor the incoming large number of heterogeneous events from the multiple domains is evaluated. Figure 6.15 illustrates that the proposed CAMEP approach outperforms the existing BAM

approaches due to the integration of CEP in the BAM. In addition, event filtering is performed to filter out the irrelevant events based on the KPIs. Furthermore, the number of KPIs available for monitoring decides the elapsed time to monitor the incoming events. In case of the more number of KPIs, the average processing time required to process the incoming events gets gradually increased.

Hypothesis Evaluation with respect to the treatment of the factor A: (Average Processing Time)

Null hypothesis H0: A1=A2=A3, where A1= BAM, A2=CAMEP 50 KPI and A3=CAMEP 25 KPI

(There is no significant difference among the three systems in terms of Average Processing Time obtained.)

Alternate hypothesis H10: Treatment means are not equal for at least one pair of the treatment means of the factor A

(There is a significant difference among the three systems in terms of Average Processing Time)

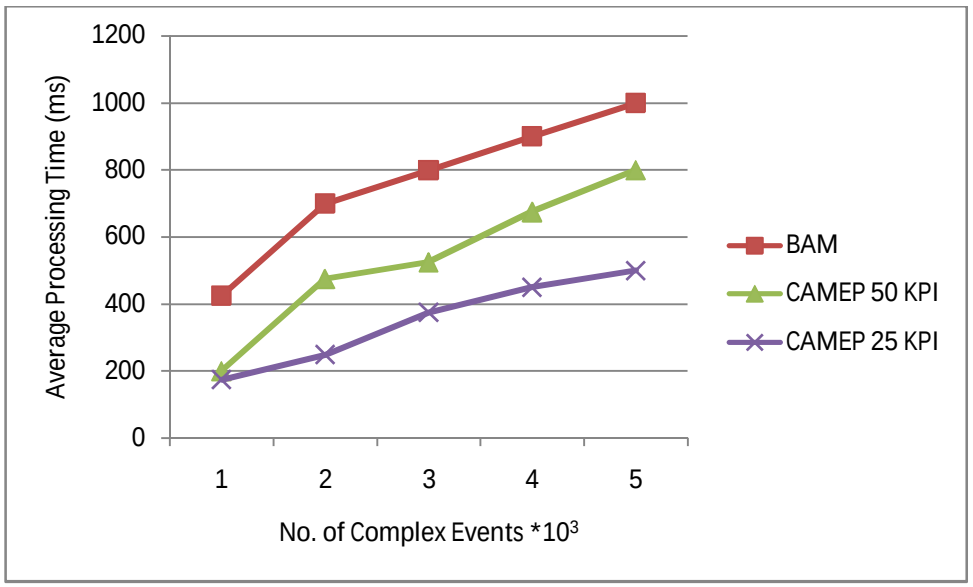

Figure 6.15: Average Processing Time of CAMEP System

Table 6.7: ANOVA Results of Average Processing Time of CAMEP System

Technique			Preprocessing time	
I	**II**	**III**	**Hypothesis**	**p value**
BAM	CAMEP 50 KPI	CAMEP 25 KPI	H10	0.001

From Table 6.7, it is concluded that the calculated significance level of the parameter detection time of comparing three systems BAM, CAMEP 50 KPI and CAMEP 25 KPI always satisfy the condition (p value<0.05). There is significant difference between the results for different Average Processing Times of BAM, CAMEP 50 KPI and CAMEP 25 KPI. Hence, the null hypothesis for H10 can be rejected.

Further, it is also required to determine the system which has the maximum Throughput. This is analyzed using descriptive statistics given in Table 6.8. Table 6.8 shows the descriptive statistics (the maximum, minimum, median, mean values and standard deviation) of each of the technique for the parameter Throughput.

Table 6.8: Descriptive Statistics of Average Processing Time of CAMEP System

Technique	BAM	CAMEP 50 KPI	CAMEP 25 KPI
Max	1000.0	800.0	500.0
Min	300.0	150.0	100.0
Mean	715.0	497.5	335.0
Median	762.5	512.5	362.5
Standard Deviation	223.7	225.0	140.5

From Table 6.8, it is evident that there is decrease in Average Processing Time values in CAMEP 25 KPI and CAMEP 50 KPI compared BAM by 53.14% and 30.4%respectively.

6.4 PERFORMANCE EVALUATION OF SMART REAL-TIME USE CASES

The performance of PCEP system with respect to Throughput and Processing time is tested in multiple domains such as stock, RFID and KPI to verify the consistency and generalness of PCEP system.

6.4.1 Throughput Comparison

The Figure 6.16 illustrates the performance of Distributed Cayuga with PCEP implemented in Stock domain, RFID domain and KPI domain. From the chart, it is evident that the Throughput of PCEP is consistently increasing with respect to the throughput of Distributed Cayuga. Thus, it can be concluded that the performance of PCEP in Throughput does not depend on the application domain.

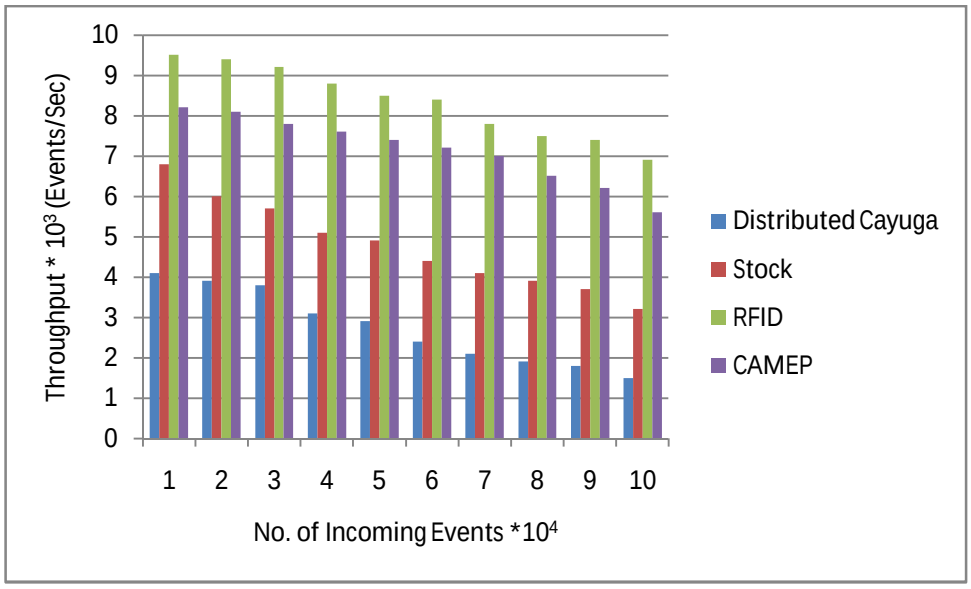

Figure 6.16: Throughput Comparison for Smart Real-Time Use Cases

6.4.2 Average Processing Time Comparison

The Figure 6.17 illustrates the performance of Distributed Cayuga in comparison with PCEP implemented in Stock domain, RFID domain and KPI domain. From the chart, it is evident that the Average Processing Time of PCEP is consistently decreasing with respect to the Average Processing Time of Distributed

Cayuga. Thus, it can be concluded that the performance of PCEP in Average Processing Time does not depend on the application domain.

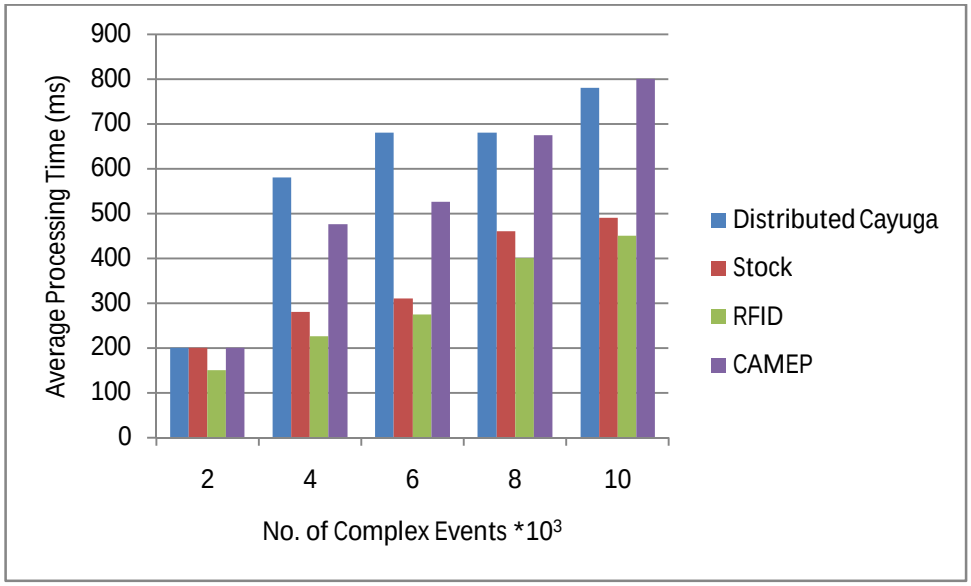

Figure 6.17: Average Processing Time Comparison for Smart Real-Time Use Cases

6.5 SUMMARY

This chapter provided the three use case applications of PCEP system in the Business Process Management to improve the business process using Business Intelligence. First, the PCEP system is implemented in the RFID based CEP Framework that is capable of detecting complex manufacturing events. The Event Filtering reduces the overhead of the processing engine by filtering out the irrelevant events from the large number of incoming events. And then, the probabilistic CEP model is presented, which computes the probability of event sequence patterns and derives the most probable events with high probability based on the rules designed using CEPL. Second, the PCEP is integrated with the Probabilistic Fuzzy Logic approach for implementing decision making tasks in the stock market prediction. It is proposed for predicting the future direction of stock prices using empirical datasets. Therefore, the PFL approach enhances the effective decision making for investors in the stock market by offering more accurate stock prediction when compared to existing approaches. Third, the PCEP is integrated with BAM to

perform an efficient monitoring of events coming from the multiple domains based on Key Performance Indicators. It triggers the necessary action for maintaining the overall business performance that displays the performance of the business process in terms of KPIs . The performance of PCEP against the existing Distributed Cayuga is statistically analyzed using ANOVA and T-Tests.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The last chapter of the Thesis provides a brief summary of the various research contributions and highlights the advantages of the PCEP system. Further it list a few problems for future research.

7.1 CONCLUSION

The main focus of the research is to develop a scalable and efficient Probabilistic Complex Event Processing (PCEP) system that can handle uncertain event stream. The research objectives has been formulated from the literature survey. In order to achieve the research objectives, experimental setup has been designed. The PCEP system is evaluated using the designed experimental setup. It is inferred that all the objectives have been achieved effectively. The performance of the system was compared with the existing Distributed Cayuga System. The statistical analysis of the results obtained has been performed using T-test and ANOVA.

The PCEP system is an integrated approach that comprises of Efficient Generic Event Filtering (EGEF) and Probabilistic Event Sequence Prediction (PSEP). The EGEF module incorporates Predicate based Subscription Grouping Algorithm, NFA_h based filtering, row/column scaling and pipelining techniques. The PSEP module incorporates the Dynamic Fuzzy Probabilistic Relational Model (DFPRM) and Probabilistic Fuzzy Logic.

The EGEP module was validated for the events from the health domain against the parameters, Throughput and Average Processing Time. The results indicate that the EGEF module has an increased Throughput of 33% and decreased Average Processing Time of 26%, compared to the existing Distributed Cayuga system.

The PCEP system was also validated for the events from the stock market domain with the parameters, Throughput and Average Processing Time. The Throughput is considered along the following two dimensions: Throughput as a function of State Machines and Throughput as a function of NFA Length. The results point out that PCEP system has an increase of 31.62% with respect to Throughput as a function of State Machines compared to Distributed Cayuga. It is inferred from the results that PCEP system has an increase of 42.46% with respect to Throughput as a function of NFA Length compared to Distributed Cayuga. The performance of the PCEP system with respect to Average Processing Time decreases by 19.17% compared to Distributed Cayuga. The scalability inferred from the observed Throughput and Average Processing Time signifies that the PCEP system is better than Distributed Cayuga by 62%.

To validate the performance of the PCEP across multiple heterogeneous domains in addition to the stock market domain, the PCEP system was tested with events from RFID event data of product manufacturing domain and Key Performance Indicators. The results show that the PCEP system processes the RFID events with increased Throughput of 28% and decreased Average Processing Time of 46%, compared to the existing Distributed Cayuga system. It is inferred that the PCEP system processes the KPI as events with increased Throughput of 16.34% and decreased Average Processing Time of 30.4%, compared to the existing Distributed Cayuga system.

The PCEP system has an efficient pre-event filtering module, which filters and allows the relevant events to the next prediction phase. The unnecessary computation for the irrelevant events is reduced and this improves the throughput of the event detection. The significant contribution of the PCEP system is the fabrication of the probabilistic framework for the CEP.

This research has made a noteworthy contribution in the field of Complex Event Processing in two aspects: Improving the efficiency of the filtering and handling the uncertain data using Probabilistic Relational Model. As a successful attempt in the field of Probabilistic Event Processing, various interesting and challenging directions are introduced for future exploration.

7.2 FUTURE WORK

- The PCEP system is limited to perform effective business monitoring within the intra-organizational setting and explicit dependency is available between the interacting parties to achieve high scalability. However, in the case of an inter-company cooperation, CEP is a challenging task to perform a robust pattern matching over uncertain events due to the lack of cooperation and heterogeneity available between the interacting parties in multiple domains. Thus, there is a need to perform semantic decoupling between the publishers and subscribers in terms of three temporal parameters such as time, space and synchronization. The system needs to acquire an explicit independency between the interacting participants to perform semantic matching and to achieve high scalability even under heterogeneous environments.

- The other possible direction is the extension of query languages to support the various monitoring applications. This will achieve effective query processing to execute multiple event queries simultaneously over inferred data to detect business opportunities and risks.

- The efficiency of DFPRM model can be further improved by self-optimizing its scaling inference for handling continuous variables that provide timely processing of a large number of uncertain events with high scalability.

- The notion of Concept Drift can also be included when generating complex events.

REFERENCES

[1] Ellie Fields and Brett Sheppard, "A New Approach to Business Intelligence: Rapid-fire BI," White Paper, Tableau Software, http://www.tableausoftware.com, 2013.

[2] Michael Eckert and Francois Bry, "Complex Event Processing," Proceedings of the Conference on Informatik-Spektrum vol. 32, no. 2, pp. 163-167, 2009.

[3] Analyze and act on Fast Moving Data: An overview of Complex Event Processing http://www.sybase.in/detail?id=1069696, 2013.

[4] Segev Wasserkrug et al., "A Model for Reasoning with Uncertain Rules in Event Composition Systems," Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence, pp. 599-608, 2005.

[5] Anish Das Sarma et al., "Working Models for Uncertain Data," Proceedings of the 22nd International Conference on Data Engineering, IEEE Computer Society Washington, pp. 7-10, 2006.

[6] C.C.Agarwal and P.S.Yu, "A Survey of Uncertain Data Algorithms and Applications," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 2, pp. 609-623, 2009.

[7] Wenjie Zhang et al., "Managing Uncertain Data: Probabilistic Approaches," Proceedings of the Ninth International Conference on Web-Age Information Management, pp. 405 – 412, Zhangjiajie Hunan, 2008.

[8] R G Healy, "Database Management Systems," http://anetlibrary.com/wp-content/uploads/2012/07/BB1v1_ch18.pdf, 2013.

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

[10] Georges Hebrail, "Data Stream Management and Mining," Mining Massive Data use for Security, IOS Press, pp. 89-102, 2008.

[11] Alajendro Buchmann and Boris Koldehofe, "Complex Event Processing," International Journal on Information Technology, vol. 51, no. 5, pp. 241-242, 2009.

[12] Lukasz Golab and M. Tamer Ozsu, “Issues in Data Stream Management,” ACM SIGMOD Record, vol. 32, no. 2, pp. 5-14, 2003.

[13] Brian Babcock et al., “Models and Issues in Data Stream Systems,” Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.1-16, New York, USA, 2002 .

[14] Opher Etzion and Peter Niblett, “Event Processing in Action,” Manning Publications Co. Greenwich, USA, 2010.

[15] Michael Daum et al., “Integrating CEP and BPM - How CEP Realizes Functional Requirements of BPM Applications,” Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 157-166, USA, 2012.

[16] Christian Janiesch et al., “Beyond Process Monitoring: a Proof-of-Concept of Event-Driven Business Activity Management,” Business Process Management Journal, vol. 18, no. 4, pp. 625-643, 2012.

[17] Solomon Negash, “Business Intelligence,” Communications of the Association for Information Systems, vol. 13, pp. 177-195, 2004.

[18] Rainer Ammon et al., “Event-Driven Business Process Management,” Proceedings of the Second International Conference of Distributed Event based Systems, 2008.

[19] Joseph M. DeFee and Paul Harmon, “Business Activity Monitoring and Simulation,” Business Process Trends Whitepaper, http:// www. bptrends.com , 2004.

[20] David W. McCoy, “Business Activity Monitoring: Calm Before the Storm,” http://www.gartner.com/id=354283, 2013.

[21] Ron Thomas, “Key Performance Indicators Measuring and Managing the Maintenance Function,” http://www.plant-maintenance.com/ articles/ KPIs.pdf, 2005.

[22] Branimir Wetzstein et al., “Preventing KPI Violations in Business Processes based on Decision Tree Learning and Proactive Runtime Adaptation,” Journal of Systems Integration, vol. 3, no. 1, pp. 3-18, 2012.

[23] N. H. Gehani et al., “Composite Event Specification in Active Databases: Model & Implementation,” Proceedings of the 18th International Conference on Very Large Data Bases, pp. 327-338, USA, 1992.

[24] S. Chakravarthy et al., “Composite Events for Active Databases: Semantics, Contexts and Detection,” Proceedings of the 20th International Conference on Very Large Data Bases, pp. 606-617, USA, 1994.

[25] David Luckham and Roy Schulte, “Event Processing Glossary - Version 1.1,” http://www.complexevents.com/2008/08/31/event-processing-glossary-version-11/, 2008.

[26] Overview of the Event Processing Language (EPL) http://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html, 2013.

[27] Michael Eckert et al., “A CEP Babel Fish: Languages for Complex Event Processing and Querying Surveyed,” Reasoning in Event-Based Distributed Systems, Studies in Computational Intelligence, vol. 347, pp. 47-70, 2011.

[28] Arvind Arasu et al., “STREAM: The Stanford Stream Data Manager,” IEEE Data Engineering Bulletin, vol. 26, no. 1, pp 19–26, 2003.

[29] “Coral8 Continuous Computation Language (CCL),” SAP Sybase Event Stream Processor, http://www.sybase.in, 2009.

[30] Joachim Reinert and Norbert Ritter, “Applying ECA Rules in DB-based Design Environments,” Proceedings of the International Conference on Distributed Systems and Information Systems, pp.188-201, Tagungsband, 1998.

[31] Eric N. Hanson and Jennifer Widom, “An Overview of Production Rules in Database Systems,” The Knowledge Engineering Review, vol. 8, no. 2, pp. 121-143, 1993.

[32] Darko Anicic et al., “Rule-Based Language for Complex Event Processing and Reasoning,” Proceedings of the Fourth International Conference on Web Reasoning and Rule Systems, pp. 42-57, 2010.

[33] Olle Wedin et al., “Data Filtering Methods,” http://www.transportresearch.info/Upload/Documents/201204/20120411_170257_85602_ROADIDEA%20D3.1%20Data%20filtering%20methods.pdf, 2008.

[34] Francis Wolinski et al., “Using Learning-based Filters to Detect Rule-based Filtering Obsolescence,” Proceedings of the 6th International Conference on Computer-Assisted Information Retrieval, France, April,pp.12-14, 2000.

[35] Lakshmish. Dutt and Mathew Kurian, “Handling of Uncertainty - A Survey,” International Journal of Scientific and Research Publications, vol. 3, no.1,pp. 1-4, 2013.

[36] Yanlei Diao et al., "Capturing Data Uncertainty in High Volume Stream Processing,"https://www.unido.org/foresight/rwp/dokums_pres/ruff_exercise_50.pdf, 2009.

[37] Alexander Artikis et al., "Event Processing Under Uncertainty," ACM Conference on Distributed Event-Based Systems , pp. 32-43, 2012.

[38] Prithviraj Sen et al., "PrDB: Managing and Exploiting Rich Correlations in Probabilistic Databases," The International Journal on Very Large Data Bases , vol.18, no.5, pp. 1065-1090 , October 2009.

[39] Omar Benjelloun et al., "ULDBs: Databases with Uncertainty and Lineage," Proceedings of the 32nd International Conference on Very Large Databases, pp.953-964, 2006.

[40] Ming Hua et al., "Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach," Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 673-686, USA, 2008.

[41] Amir H. Meghdadi and Mohammad-R Akbarzadeh, "Probabilistic Fuzzy Logic and Probabilistic Fuzzy Systems," Proceedings of the 10th IEEE International Conference on Fuzzy Systems, vol. 3, pp. 1127 - 1130, Melbourne, 2001.

[42] Hani Hagras, "Fuzzy Logic Systems to Enable Better Uncertainty Handling for Real World Application," IEEE Computational Intelligence Magazine, vol. 7, no. 3, pp. 14-24, 2012.

[43] Luis M. de Campos et al., "The BNR model: Foundations and Performance of a Bayesian Network-based Retrieval Model," International Journal of Approximate Reasoning, vol 34, no 2-3, pp 265-285, 2003.

[44] Kristian Kersting and Luc De Raedt, "Bayesian Logic Programming: Theory and Tool," In: Getoor, L., Taskar, B. (eds.) An Introduction to Statistical Relational Learning, pp.291-321, MIT Press, Cambridge, 2007.

[45] Alexis Campailla et al., "Efficient Filtering in Publish-Subscribe Systems using Binary Decision Diagrams," Proceedings of the 23rd International Conference on Software Engineering, pp. 443-452, Canada, 2001.

[46] Kyoung Soo Bok et al., "Efficient Complex Event Processing over RFID Streams," International Journal of Distributed Sensor Networks, pp.1-9, 2012.

[47] Ehab S. Al-Shaern, "High-performance Event Filtering for Distributed Dynamic Multi-point Applications: Survey and Evaluation," Technical Report, Old Dominion University Norfolk, VA, USA, 1997.

[48] Leonardo Neumeyer et al., “S4: Distributed Stream Computing Platform,” Proceedings of the 2010 IEEE International Conference on Data Mining Workshops, pp.170-177, USA, 2010.

[49] Carl Hewitt and Henry Baker. JR, “Actors and Continuous Functionals,” Cambridge : Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1977 .

[50] Daniel J. Abadi et al., “Aurora: a New Model and Architecture for Data Stream Management,” The VLDB Journal, pp. 120-139, 2003.

[51] Don Carney et al., “Monitoring Streams - A New Class of Data Management Applications,” Proceedings of the 28th International Conference on Very Large Data Bases, pp. 215–226, 2002.

[52] Sirish Chandrasekaran et al., “TelegraphCQ: Continuous Dataflow Processing for an Uncertain World,” Proceedings of the 2003 Conference on Innovative Data Systems Research, New York, USA, 2003.

[53] Sailesh Krishnamurthy et al., “TelegraphCQ: An Architectural Status Report,” IEEE Data Engineering Bulletin,no.1, pp. 11-18, 2003.

[54] Rajeev Motwani et al., “Query Processing, Resource Management, and Approximation in a Data Stream Management System,” CIDR,Technical Report, Stanford InfoLab, 2003.

[55] Arvind Arasu et al., “The CQL Continuous Query Language: Semantic Foundations and Query Execution,” The VLDB Journal, vol. 15, no. 2, pp. 121–142, 2006.

[56] EsperTech: Event Stream Intelligence, http://www.espertech.com/ 2013.

[57] Espertech, Esper Reference Manual, http://esper.codehaus. org/esper/documentation/documentation.html, 2013.

[58] D. Gyllstrom et al., “SASE: Complex Event Processing Over Stream,” Proceedings of the Biennial Conference on Innovative Data Systems Research, pp. 407–411, 2007.

[59] Eugene Wu et al., “High-Performance Complex Event Processing over Streams,” Proceedings of the ACM SIGMOD International conference on Management of data, pp. 407-418, USA, 2006.

[60] Alan Demers et al., “Cayuga: A General Purpose Event Monitoring System,” Proceedings of the Conference on Innovative Data Systems Research, pp. 412-422, 2007.

[61] Lars Brenna et al., “Cayuga: A High-Performance Event Processing Engine,” Proceedings of ACM SIGMOD International Conference on Management of Data, pp 1100-1102, 2007.

[62] John Morrell and Stevan D. Vidich, “Complex Event Processing with Coral8,” 2008.

[63] Coral8 Complex Event Processing Engine, White Paper, http://www.coral8.com/system/files/assets/pdf/Complex_EventProcessing_with_Coral8.pdf, 2013.

[64] Asaf Adi and Opher Etzion, “AMiT-the Situation Manager,” The VLDB Journal, vol 13, no 2, pp 177–203, 2004.

[65] MS Analog Software, “ruleCore(R) Complex Event Processing (CEP),” http://www.rulecore.com, 2013.

[66] “JBoss Rules User Guide,” http://docs.jboss.org/drools/ release/ 5.3.0. Final/drools-expert-docs/html/ch01.html.

[67] Christopher Ré et al., “Event Queries on Correlated Probabilistic Streams,” Proceedings of the Fourth International Conference on SIGMOD’08, pp. 715-728,Canada, 2008.

[68] Mohamed A. Soliman et al., “Top-k Query Processing in Uncertain Database,” Proceedings of the 23rd International conference on Data Engineering, pp. 806-905,2007.

[69] Ke Yi et al., “Efficient Processing of Top-k Queries in Uncertain Databases with X-Relations,” IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 12, pp. 1669-1682, USA, 2008.

[70] Christopher Re Nilesh Dalvi and Dan Suciu, “Efficient Top-k Query Evaluation on Probabilistic Data,” Proceedings of the 23rd International Conference on Data Engineering, pp. 886-895, Turkey, 2007.

[71] Nilesh Dalvi and Dan Suciu, “Efficient Query Evaluation on Probabilistic Databases,” International Journal on Very Large Data Bases, vol. 16,no. 4, pp. 523-544, New York, USA,2007.

[72] Daisy Zhe Wang et al., “Probabilistic Complex Event Triggering,” Technical Report-EECS-2009-114, EECS Department, University of California, Berkeley, 2009.

[73] Thanh Tran et al., “Probabilistic Inference over RFID Streams in Mobile Environments,” IEEE 25th International Conference on Date of Conference, pp. 1096 – 1107, 2009.

[74] Shariq Rizvi, "Complex Event Processing Beyond Active Databases: Streams and Uncertainties," Berkeley Technical Report No. UCB/EECS-2005-26, 2005,http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-26.pdf,2005.

[75] Reynold Cheng et al., "Evaluating Probabilistic Queries over Imprecise Data," Proceedings of the 2003 ACM SIGMOD International Conference on Management of data, pp. 551-562, USA,2003.

[76] Nodira Khoussainova and Magdalena Balazinska Dan Suciu, "PEEX: Extracting Probabilistic Events from RFID Data," Technical Report, 2007-11-2, Department of Computer Science and Engineering, University of Washington, 2007 .

[77] Nodira Khoussainova et al., "Probabilistic Event Extraction from RFID Data," IEEE 24th International Conference on Data Engineering, pp. 1480-1482, 2008.

[78] Zhitao Shen et al., "Probabilistic Event Stream Processing with Lineage," In Proceedings of Data Engineering Workshop, 2008.

[79] Patrick TH. Eugster et al., "The Many Faces of Publisher/Subscriber," ACM Computing Surveys, vol. 35, no. 2, pp. 114–131, 2003.

[80] Lars Brenna et al., "Distributed Event Stream Processing with Non-deterministic Finite Automata," Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, no.3, pp. 1-12,USA, 2009.

[81] Angelo Corsaro B et al., "Quality of Service in Publisher/Subscriber Middleware," Global Data Management, pp. 1–19. IOS Press, Amsterdam, 2006.

[82] Jagrati Agrawal et al., "Efficient Pattern Matching over Event Streams," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 147-160, USA, 2008.

[83] Franchoise Fabret et al., "Filtering Algorithms and Implementation for Very Fast Publisher/Subscriber Systems," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 115-126, USA, 2001.

[84] Yanming Nie et al., "Complex Event Processing over Unreliable RFID Data Streams," Proceeding of the Thirteenth Asia-Pacific Web Conference on Web Technologies and Applications, pp. 278-289, China, 2011.

[85] Alan Demers et al., "Towards Expressive Publish/Subcribe Systems," Proceedings of the 10th International Conference on Advances in Database Technology, pp. 627-644, Heidelberg, 2006.

[86] Sumit Sanghai Pedro Domingos and DanielWeld, "Dynamic Probabilistic Relational Models," Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 992-997, USA, 2003.

[87] Lise Gatoor et al., "Learning Probabilistic Relational Model," International Joint Conferences on Artificial Intelligence, pp. 1300-1309, 1999.

[88] Lise Gatoor et al., "Selectivity Estimation using Probabilistic Models," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 461-472, USA, 2001.

[89] Fabian Kaelin and Doina Precup, "An Approach to Inference in Probabilistic Relational Models using Block Sampling," Workshop and Conference Proceedings of 2nd Asian Conference on Machine Learning, pp. 325-340,Tokyo, Japan, 2010.

[90] Lotfi A.Zadeh, "Probability Theory and Fuzzy Logic are complementary rather than Competitive," Technometrics, vol. 37, no. 3, pp. 271-276, 1995.

[91] E Diaz-Hermida et al., "A General Framework for Probabilistic Approaches to Fuzzy Quantification," EUSFLAT Conference, pp. 34-37, 2001.

[92] Yihua li and Wenjing Huang, "A Probabilistic Fuzzy Set for Uncertainties based Modeling in Logistics Manipulator System," Journal of Theoretical & Applied Information Technology, vol. 46, no. 2, pp. 977-982, 2012.

[93] Isabel L. Nunes, "Handling Human-Centered Systems Uncertainty Using Fuzzy Logics – A Review," The Ergonomics Open Journal, vol. 3, pp. 38-48, 2010.

[94] Gianpaolo Cugolaa and Alessandro Margaraa, "Complex Event Processing with T-REX," The Journal of Systems and Software, vol. 85,no. 8, pp. 1709-1728, 2012.

[95] Bilal Hameed et al., "An RFID Based Consistency Management Framework for Production Monitoring In a Smart Real-Time Factory," 2nd International Conference on Internet of Things, pp. 1-8,Tokyo, Japan, 2010.

[96] Oleksandr Mylyy, "RFID Data Management,Aggregation and Filtering," Proceedings of the 31st International Conference on Very Large Data Bases Seminar on RFID Technology, pp. 6-15, USA, 2007.

[97] Tao Ku et al., "Novel Complex Event Mining Network for Monitoring RFID-Enable Application," IEEE Computer Society, pp. 925-929, 2008.

[98] Guangqian Zhang et al., "Study on CEP-Based BCEPS Model of RFID based RLTLUM System," Journal of Software, vol. 4, no. 6, pp. 605-613, 2009.

[99] Fusheng Wang et al., “Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams,” Lecture Notes in Computer Science, vol.3896, pp. 588-607, 2006.

[100] Xiaoyong Su and Rajit Gadh, “A Rule Language and Framework for RFID Data Capture and Processing in Manufacturing Enterprise System,” International Journal of Internet Manufacturing and Services, vol. 2, no. 2, pp. 111 - 127, 2010.

[101] Yijian Bai et al., “RFID Data Processing with a Data Stream Query Language,” Proceedings of the IEEE 23rd International Conference on Data Engineering, pp. 1184-1193, Istanbul, 2007.

[102] Mark Palmer, “Seven Principles of Effective RFID Data Management,” Progress Softwares Realtime Division, http://www.psdn.progress.com/ realtime/ docs/articles/7principles_rfid_ mgmnt.pdf, 2004.

[103] Nova Ahmed et al., “RF2ID: A Reliable Middleware Framework for RFID Deployment,” IEEE International Parallel and Distributed Processing Symposium, pp. 1-10, USA, 2007.

[104] Kia Teymourian et al., “Knowledge-Based Processing of Complex Stock Market Events,” Proceedings of the fifteenth International Conference on Extending Database Technology, pp. 594-597, USA, 2012.

[105] “Enabling the Real-time Enterprise Business Activity Monitoring with Ensemble,” White Paper of Intersystem Corporation, www.Inter Systems. com/Ensemble, 2013.

LIST OF PUBLICATIONS

International Journals

1. Govindasamy, V. and Thambidurai, P., "Probabilistic and Fuzzy Logic based Event Processing For Effective Business Intelligence," International *Arab Journal of Information Technology*, (Accepted for Publication), **SCIE, Impact Factor : 0.390**

2. Govindasamy, V. and Thambidurai, P., "CAMEP – Complex Business Activity Monitoring System with Event Processing," Journal Tehnički Vjesnik/Technical Gazette, Vol. 20, No. 5, October 2013, pp. 823 -829, **SCIE, Impact Factor : 0.601.**

3. Govindasamy, V. and Thambidurai, P., "Uncertain Event Processing Using Prediction Correction Paradigm," *International Journal of Engineering Research and Technology*, 2, 4, April 2013, pp. 973-979.

4. Govindasamy, V. and Thambidurai, P., "Heuristic Event Filtering Methodology for Interval based Temporal Semantics," *International Journal of Computer Applications*, 70, 7, May 2013, pp. 16-20.

5. Govindasamy, V. and Thambidurai, P., "RFID Probabilistic Complex Event Processing in a Real-Time Product Manufacturing," *International Journal of Engineering and Innovative Technology*, 2, 10, April 2013, pp. 139-144.

6. Govindasamy, V. and Thambidurai, P., "Complex Event Processing - A Survey," *Journal of Computing*, 5, 4, April 2013, pp: 1-7.

7. Govindasamy, V. and Thambidurai, P., "An Efficient Methodology for Uncertain Complex Event Processing," *European Journal of Scientific Research*, 101, 1, May 2013, pp.125-137.

8. Govindasamy, V. and Thambidurai, P., "Probabilistic Fuzzy Logic based Stock Price Prediction," *International Journal of Computer Application*, 71, 5, June 2013, pp. 28-32.

International Conferences

9. Govindasamy, V. and Thambidurai, P., "An Efficient and Generic Filtering Approach for Uncertain Complex Event Processing," in *Proceedings of International Conference on Data Mining and Computer Engineering (ICDMCE'2012)*, Bangkok, Thailand, 22 December, 2012, pp. 211-216.

10. Govindasamy, V. and Thambidurai, P., "Study of Event Query Languages in Complex Event Processing," in *Proceedings* of *IEEE International Conference on Computing, Cybernetics and Intelligent Information System (CCISS '2013)*, Vellore, India, 21-23 November, 2013.

VITAE

V.GOVINDASAMY, the author of this thesis is currently working as Assistant Professor in the Department of Information Technology at Pondicherry Engineering College, Puducherry, India. He was born in October 1974 at Puducherry. He received his B.Tech. degree in Computer Science and Engineering from Pondicherry University, Puducherry, India in the year 1996 and M.E degree in Computer Science and Engineering from Vellore Engineering College, Vellore in 2000. He joined as Lecturer in Department of Information Technology, Pondicherry Engineering College, Puducherry in the year 2002. His areas of interest include Business Intelligence, Business Process Management and Event Processing. He has published several research papers in International Journals and Conferences.