Representation and Processing of Information Related to Real World Events

Aparna Nagargadde, Sridhar V

Applied Research Group, Satyam Computer Services Ltd.

III floor, A Wing, SID BLock, IISc Campus, Bangalore - 560012, INDIA

Krithi Ramamritham

Indian Institute of Technology-Bombay, Powai, Mumbai - 400076, INDIA

Abstract

Event based analysis plays an important role in reducing the latency of information delivery in an event driven world. Also, the perception of an 'event' by a user is at a higher level (a meta event), and would involve the analysis of several less complex or lower order events (basic events) in order to convey meaningful information. This is especially true of real world events and it is often necessary to completely capture the attributes of events and the relationships between them, so that the process of retrieval of event related information is efficient. In this paper, we discuss a formal system for representing and analyzing real world events to address these issues. The event representation discussed in this paper accounts for three important event attributes, namely, time, space, and label. We introduce the notion of *sequence templates* that appears natural for capturing event related semantics. It can also help in semantically analyzing user queries. To harness this potential, we present a formal structure to represent the queries related to real world events as well as an approach to semantically analyze a user query, and collate event related information to be dispatched to the user. Finally, we discuss the design and implementation of the Query-Event Analysis System (QEAS), which is an integrated system to (a) identify a best-matching sequence template(s) given a user query; (b) derive the meta-events based on the selected sequence templates; and (c) and use the meta-event information to answer the user query.

Index Terms

Event Representation, Sequence Templates, Query Processing

I. Introduction

Reducing the latency of information delivery in an event driven world has always been a challenge. Event based analysis plays an important role in the delivery of information to users.

It is often necessary to completely capture the attributes of events and the relationships between them, so that the process of retrieval of event related information is efficient. For example, when a Cricket match is in progress, the various basic events such as ‘ball hits wickets,” “fielder runs,” and “fielder catches ball” are identified. However, the queries from users could span a wide-range such as “What is the current run-rate?,” “How many wickets are down?,” “What was the highest score chased successfully on this ground?,” and so on. Similarly, the various events detected by network monitoring systems include events such as “packet P is lost,” “traffic from A to B is being rerouted” and “server inactivity”. Using these events, a network monitoring system has to determine if more complex events, such as, router congestion and malfunctioning or failure of one or more network components have occurred.

To answer such complex queries based on the observed events, efficient event based analysis is essential. An ability to semantically characterize events enhances the scope and flexibility of the event management system in answering these complex queries. There is a need to formally address the issues related to representation and analysis of real-world events. For example, to answer the query, “How many wickets are down?” we need to first define the event ‘wicket down’ in the context of a cricket match. Also, we need to define the events or a sequence of basic events which results in the desired event ‘wicket down’. An efficient semantic characterization of events include various issues such as characterization of event attributes, identification of event relationships, definition of composite and derived events, and the processing of event related information in order to answer queries.

In this paper, we describe a formal framework to address these issues. In building our formal framework, we draw upon the various advances in the fields of temporal semantics, spatial semantics and event composition in distributed systems. However, though these advances address some of the facets of real world events, there is no comprehensive formalization of representation of real world events. Real world events are characterized by temporal, spatial and label attributes; the lack of even one attribute would lead to an incomplete characterization of the event. The formalism presented in this paper represents a holistic approach to the analysis of the temporal, spatial and label attributes of real world events. For example, the event “Kaif hits half-century at Lords” occuring on 13th July 2002 is characterized by the temporal attribute *13th July 2002*, spatial attribute *Lords* and the label (or event tag) *half-century by Kaif*. It is apparent that the event specification is complete when an event possesses attribute values along these three dimensions.

The main contributions of this work [1] include a) event representation in terms of temporal, spatial and label attributes, b) the use of domain-specific hierarchies along temporal, spatial and label dimensions for enhanced semantic analysis, c) the definition of event closures in conjunction with these domain specific hierarchies in order to recognize the similarities between otherwise unrelated events, d) the use of comprehensive sequence templates for semantic analysis of events, e) a formal structure in the form of event expressions to represent the wide range of queries that can be posed on real world events and f) an approach to semantically analyse user queries and collate the event related information to be dispatched to the user.

This paper presents a technique to abstract event processing by using *meta-events*, and this technique reduces the runtime during query processing. In order to demonstrate the comprehensiveness of the proposed event representation, we have developed a system (QEAS) that aggregates and analyses events with a view to answer diverse user queries using this event representation. We present the architecture of this *Query Event Analysis System (QEAS)* along with a formal analysis of the liveness and the safety properties of the system. We have undertaken an implementation of the QEAS system and initial results from the same have been presented here.

In Section. II, we discuss the related work in this sphere. Section. III discusses in detail event representation along the temporal, spatial and label (TSL) dimensions, event composition and closure operations related to events. Event analysis is described in Section. IV, and the Query Event Analysis System based on the proposed event representation is described in Section. V. An in-depth analysis of the QEAS system is presented in Section. VI, followed by a case study. Several scenarios in this paper are drawn from the realm of international cricket, in particular from the NATWEST series in 2002 [17]. As an aid to those who are unfamiliar with cricket terminology, a table of the various cricketing terms is presented in Appendix. II.

II. Related Work

Event representation and analysis has been an area of active research. In particular, the temporal nature and properties of events have been widely studied. The representation of time and temporal relationships has been addressed in several papers, notably [2]. Though the temporal and spatial

[1]This paper is a major extension of [14]

TABLE I

COMPARISON OF RELATED WORK

Ref	Temporal Dimension	Spatial Dimension	Additional Event Dimensions	Derived Events	Model used to capture Derived Events
1	Interval Based Semantics	--	Event Type used to identify other attributes	Event expressions using temporal relationships	Event Graphs
10	Point Based Semantics				Full power coloured petri nets
5, 8					Coloured petri nets
3		Considered in broader context of event attributes		--	Behavioural Models
7	Interval Based Semantics	Spatial Dimension of multiple objects in event scene are based on bounding box description	Used in description of objects within the video	Spatiotemporal relationships between objects in multiple scenes used to derive events	Specified Using Bilbvideo Query Language
Our Work		Region Semantics based on formalisms such as [9]	Value along label dimension is part of real world events. Operations are defined on label attribute to derive additional events	Events derived using closure operations and relationships along TSL dimensions	Sequence Templates defined for event expressions

attributes of events have been widely studied, there are very few event specification languages that support a unified view of both these attributes in the case of real world events. Composite event detection using event templates has also been proposed in several papers [3], [5], [15]. However, the proposed event templates do not consider the temporal, spatial and label event attributes holistically. Table. I compares of the related work in this sphere. Several of these works use the *type* attribute to characterize the event along additional dimensions in the form of event labels and minimum and maximum values. *Derived events* represent all those events that can be generated using the various event operators and closures. In the real world, event input is received through several loosely coupled event sensors/detectors. We employ the temporal frameworks suggested in [12], [15] to order the event input for processing. We subscribe to the use of interval-based semantics [1] for composite event detection. [6] presents the semantics of several event composition operations using the notion of a global event history. Also an architecture and the implementation of a composite event detector is analyzed in the context of an object-oriented active DBMS. [13] presents formal semantics for capturing the enabling conditions of ECA rules in active real-time databases.

Two aspects are important in the case of real world events: event attributes and event content. Event attributes must be able to provide adequate cues for the automatic generation and dissemination of event contents. As an example, consider an event ‘Six’ by a batsman in a One Day International (ODI) cricket match. In disseminating this event to a subscriber watching the ODI

over a mobile, it is required to automatically generate a video clip depicting the batsman hitting the ball directly outside the boundary line. We believe that the sequence templates, introduced in this paper, can play an important role in this content generation activity. In our related papers [9], [16], we have described issues related to content generation and dissemination.

III. Event Representation

We describe an algebra for real world events, and in this respect, every real world happening is an event. Event related information can be categorized into two kinds: formal attributes and informational attributes. Formal attributes form the basis for formally analyzing events. On the other hand, informational attributes provide more information regarding events (for example, a video clip associated with an event). Informational attributes can also be viewed as a bag of attributes. In this paper, we consider time, space and label as part of the formal attribute set of events. Accordingly, we define an event to be characterized along three event dimensions: *temporal*, *spatial*, and *label*. For example, the event "Kaif hits half-century at Lords" is characterized by the temporal attribute *13th July 2002*, spatial attribute *Lords* and the label (or event tag) *half-century by Kaif* i.e., the event specification is complete when an event possesses attribute values along these three dimensions. Hereafter, we shall refer to the time, space and label dimensions as *TSL* dimensions, and the values along these dimensions for a given event as the *TSL* attributes for that event.

An event that can be detected by an event sensor is called a *basic event*. Basic events could be sensed, i.e. detected automatically, or could be provided as input by an event originator. The TSL attributes for a basic event are attached to the event by the event sensors/detectors and they specify the information regarding the occurrence of the event. A set of occurred basic events forms the *Event History H*, and user queries of interest can be answered based on the analysis of events that are present within this event history *H*.

A *meta-event* is derived from one or more basic events. Meta events can be derived using the composition (conjunction, disjunction and sequence) and/or closure operations on one or more basic events. These event operations are explained in sections III-B and III-C. The label attribute of a meta-event is called *meta-label* and various events which combine to form a meta event are called *component-events*. The TSL attributes of a meta event are derived using TSL attributes of the component events. We now provide an analysis of the event dimensions and

TABLE II

MIN, MAX OPERATIONS ON TEMPORAL ATTRIBUTES

Granularity of t_1 and t_2	Conditions	*min*(t_1, t_2)	*max*(t_1, t_2)
t_1 and t_2 are both points	$t_1 < t_2$	t_1	t_2
t_1 is a point and t_2 is an interval	$t_1 < t_{2s}$	t_1	t_2
$\langle t_{2s}, t_{2t} \rangle$	$t_{2s} < t_1 < t_{2t}$	t_2	$\langle t_1, t_{2t} \rangle$
	$t_1 > t_{2t}$	t_2	t_1
t_1 and t_2 are both intervals	$t_{1t} < t_{2s}$	t_1	t_2
$t_1 = \langle t_{1s}, t_{1t} \rangle$, $t_2 = \langle t_{2s}, t_{2t} \rangle$	$t_{1s} < t_{2s} < t_{1t} < t_{2t}$	$\langle t_{1s}, t_{2t} \rangle$	t_2
	$t_{1s} < t_{2s} < t_{2t} < t_{1t}$	t_1	$\langle t_{2s}, t_{1t} \rangle$

event compositions, and describe the closures of a set of events.

A. Event dimensions

The temporal, spatial and label dimensions are each unique and distinct in their characteristics. Temporal dimension is continuous and dynamic. The temporal attribute of an event can either specify a time point of occurrence or a time-interval over which the event was observed. The granularity of a time point depends on the *event space* within which the event is defined.

Definition 1: An event space ψ is defined as the space encompassing a time interval $\mathrm{T} = < T_s, T_t >$, a region $\Re$ and a set of label hierarchies $\mathcal{L}$.

For example, the time attributes '2004:08:03:05:xx:yy' and '2004:08:03:aa:bb:cc' can both be considered as time points, depending on whether the time granularity is in hours or days. Since events are detected by a distributed network of sensors/detectors; it would be simplistic to presuppose the existence of a global clock. In this paper, we resort to temporal modelling assuming a global reference time as proposed in [12]. We define two operations along the time dimension, *min* and *max*. The *min*, *max* operations determine the earlier and the later time-stamps respectively. These operations are used to determine the time of occurrence of composite events. Table II defines the *min* and *max* of event attributes along the T dimension.

We define the spatial attribute as a region that exhibits physical contiguity and can be well defined using 2D/3D bounds and is used to specify the geographical location of the event's occurrence. Using the lower-level representations of spatial attributes, with bounding boxes as a basis [11], we can define the semantics of region bounding operators to describe 'regions'. However, we attach names to these regions for the sake of simplicity. For example, the region 'Trent-bridge' can be described using a set of 2D/3D points that satisfy its region-attributes.

TABLE III

MIN, MAX OPERATIONS ON SPATIAL ATTRIBUTES

Conditions	*min*(s_1, s_2)	*max*(s_1, s_2)	Remarks
$s_1 = s_2$	s_1	s_1	
$s_1 \subseteq s_2$	s_1	s_2	When $s_1 = \emptyset$ and $s_2 \neq \emptyset$, $s_1 \subseteq s_2$, hence *min*(s_1, s_2) = $\emptyset$.
$s_1 \nsubseteq s_2$	$s_1 \bigcup s_2$	$s_1 \bigcup s_2$	This condition holds only when $s_1, s_2 \neq \emptyset$.

Similar to the temporal dimension, the granularity of space attribute is also determined by the event space. Depending on the granularity, a region could comprise of other smaller regions, with well defined spatial relations defining the orientation of the component regions with each other. A few relational operations (*touch, inside*) are described in [7]. The various spatial relationships can be automatically derived using the procedural semantics associated with these regions [11].

An event location *s* is either a point in the space dimension or a set of points $\{s_1, s_2, s_3, \ldots, s_n\}$ in the space dimension. The comma denotes a logical contiguity between the various location points in a set of points. We define two operations, *min* and *max* along the spatial dimension. The *min*, *max* operations denote the minimum and maximum bounds of the set of points denoted by the two locations s_1, s_2. Note that s_1, s_2 could either be singular locations, or set of points. All set based operations($\bigcap$, $\bigcup$, $'$, $\ldots$) and relations($\subseteq$, $\nsubseteq$, ϵ, $\ldots$) are applicable in the spatial dimension. Table III defines the *min* and *max* of event attributes along the S dimension. When the location attribute of an event is $\emptyset$, it signifies the non occurrence of an event.

Every domain is associated with a set of generic event labels that categorize the various events in that domain. For example, the domain of soccer is associated with event labels such as *Goal*, *Penalty*, *Match*, etc. Event labels can be represented as a hierarchical set, with the root being a generic label, and each child node being a specialization of the corresponding parent. If label l_1 is a specialization of a label l_2, then $l_1 \rightarrow l_2$. $l_1 \leftrightarrow l_2$ indicates the existence of an alias. Table IV defines the *and* and *or* operations along the L dimension. If l_1 and l_2 are labels, then $l_1 \,\&\&\, l_2$ and $l_1 \parallel l_2$ are valid labels.

Label hierarchies form an important tool in determining relationships between events, as well as in analyzing composite events. They can also be used to provide some additional information about events, such as, their frequency, location, sensitivity, and criticality. Similarly, hierarchies

can also be defined along the temporal and spatial dimensions. The temporal, spatial and label hierarchies are collectively referred to as *TSL hierarchies*. A few example temporal and label hierarchies in the domain of cricket are shown in Fig. 1.

B. Event composition and event sequences

A *composite event* is an event that is derived using one or more events. The two fundamental event composition operations are *disjunction* and *conjunction*. Event composition along the temporal dimension is a well researched topic. We follow the temporal semantics as presented in [1]. These semantics follow from the 7 well known relational operators [2] along the temporal dimension, namely, before, during, starts, finishes, overlaps, meets and equals. We define the spatial and label semantics for conjunction and disjunction of events. A detailed description of the semantics of event conjunction and disjunction is provided below and summarized in Table. V.

Event Disjunction: $e^3_{t_3,s_3,l_3} = e^1_{t_1,s_1,l_1} \mid e^2_{t_2,s_2,l_2}$

The event e^3 occurs when at least one of the events e^1, e^2 occurs. Event e^3 becomes is said to occur as soon as one of the events e^1 or e^2 occurs.

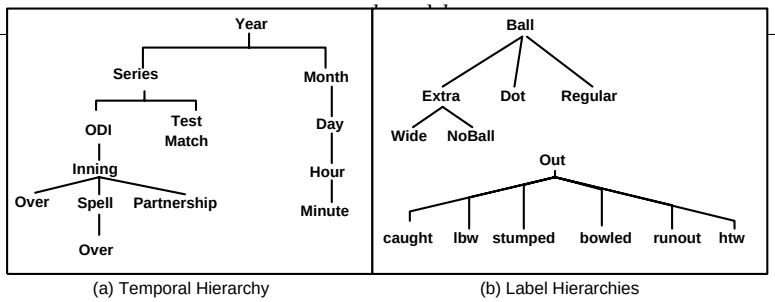
Event Conjunction: $e^3_{t_3,s_3,l_3} = e^1_{t_1,s_1,l_1} \ \& \ e^2_{t_2,s_2,l_2}$

The event e^3 occurs when both events e^1 and e^2 occur. e^3 is detected only when the later of the

TABLE IV

CONJUNCTION, DISJUNCTION OPERATIONS ON LABEL ATTRIBUTES

Conditions	and(l_1, l_2)	or(l_1, l_2)	Remarks
$l_1 = l_2$ or $l_1 \leftrightarrow l_2$	l_1	l_1	
$l_1 \rightarrow l_2$	l_1	l_2	When one of the labels is a specialization of the other, and(l_1, l_2) is a more specialized label; or(l_1, l_2) is a generalized label.
Unrelated	$l_1 \&\& l_2$	$l_1 \parallel l_2$	$l_1 \&\& l_2$ denotes a meta-label based on both l_1 and l_2; $l_1 \parallel l_2$ denotes a meta-label based on either or both of



(a) Temporal Hierarchy (b) Label Hierarchies

Fig. 1. Temporal and label hierarchies in the domain of cricket

TABLE V

ATTRIBUTES OF COMPOSITE EVENT

Composite event $e^3_{t_3,s_3,l_3}$	Condition	$t_3 = < t_{3s}, t_{3t} >$	s_3 = *R*
Conjunction	$t_1 = \infty$	∞	*min*(s_1, s_2)
	$t_1 \neq \infty$ and $t_2 \neq \infty$	$\langle$ *min*(t_1, t_2), *max*(t_1, t_2) $\rangle$	*min*(s_1, s_2)
Disjunction	$t_1 = \infty$ and $t_2 = \infty$	∞	*max*(s_1, s_2)
	$t_1 \neq \infty$ and $t_2 = \infty$	t_1	*max*(s_1, s_2)
	$t_1 \neq \infty$ and $t_2 \neq \infty$	$\langle$ *min*(t_1, t_2), *max*(t_1, t_2) $\rangle$	*max*(s_1, s_2)

two events e^1, e^2 occurs.

Event conjunction and disjunction operators help in combining two or more basic events, thereby resulting in a meta event. The temporal attribute of events can be either a time point or a time interval, and the space attribute can either be a single location or a set of points. The temporal attribute of a meta event is a time interval $\langle t_s, t_t \rangle$, and the spatial attribute is a set of points *R*. Note that meta events can also be combined using conjunction and disjunction operators.

A more generic event composition operation is the *sequence* operation.

Event sequence: $e^3_{t_3,s_3,l_3} = e^1_{t_1,s_1,l_1} \odot_k e^2_{t_2,s_2,l_2}$

An event e^3 composed of two events e^1 and e^2 with a well defined temporal ordering forms an event sequence. By the natural definition of a sequence, $t_1 \leq t_2$ is a constraint that needs to be satisfied. The other constraints to be specified could include constraints such as $t_2 - t_1 \leq \Delta$; locations $s_1, s_2 \in \mathcal{R}$, etc. The sequence operator is $\odot$. Its subscript *k* represents the set of constraints along the TSL dimension that must be satisfied by the two adjacent events in an event sequence. The time of occurrence of the sequence is given by the time interval $t_3 = \langle$*min*(t_1, t_2), *max*$(t_1, t_2)\rangle$. The region of occurrence of the sequence is given by $s_3 =$ *max*(s_1, s_2), where s_3 represents the region encompassed by the individual regions s_1 and s_2. The semantics for *min* and *max* operations along temporal and spatial dimensions are given in section. III-A. An event sequence π can be generalized as an ordering of events $\{e^1_{t_1,s_1,l_1} \odot_{k_1} e^2_{t_2,s_2,l_2} \odot_{k_2} e^3_{t_3,s_3,l_3}, \ldots, e^n_{t_n,s_n,l_n}\}$. The set of events belonging to a sequence π is represented by E^π. $\kappa^\pi = \{k_1, k_2, \ldots k_m\}$ represents the set of constraints to be satisfied by the events in the event sequence π. An example of a generic event-sequence is the 'run-out' event (see Fig. 2 on page 13).

C. Event closure

Various types of *event closures* can be defined on real world events so as to enable a quick retrieval of the related events based on a query. We discuss below two types of event closures, namely, *logical closure* and *sequence closure*. Logical closures help in retrieving the events that are logically related along the *TSL* dimensions and can be used in analyzing various aspects of basic, conjunct, disjunct, and negation events. The sequence closure describes the closure rules for event sequences. Sequence closures help in determining alternative sequences that can be constructed from logical closures of events in a given sequence π. *Event closures are defined only for basic events*. This is because, basic events refer to the actual occurred events, which have been detected. i.e., if the TSL attributes of a basic event define an event space, then the basic event has occurred at every point within that space. The same is not true for meta events which have been derived using two or more basic events.

1) Logical closure: Logical closure falls into two categories: generic closure and semantic closure.

Generic closure, $C_G(e)$, of an event *e* is used to identify those events that are *contained* within an occurred basic event. An event e^a is *contained* in an event e^b, when the TSL attributes of e^a are contained in the event space defined by the TSL attributes of e^b.

The generic closure on an event $e_{<t_s,t_t>,s,l}$ is given by

$C_G(e) = \{e_{t_1,s_1,l_1} \mid e_{t,s,l} \ \wedge \ (t_s \leq t_1 \leq t_t) \wedge \ (\ s_1 \ \subseteq s) \wedge \ (l \ \rightarrow \ l_1)\}$

Consider the event $e^a_{<2004:03:24:16:30:xx,2004:03:24:18:30:yy>,cricket-field,rain}$. Let event e^a represent rains over the region *cricket-field* during the time interval 16:30 to 18:30 on the 24^{th} of March 2004. Since the location *pitch*[2] is contained in the region *cricket-field*, by generic closure, we have: $e^1_{2004:03:24:16:45:xx,pitch,rain} \ \epsilon \ C_G(e^a)$.

Semantic closures, $C_S(e)$, are closures based on logical implication. Semantic closures have been defined in order to allow the closures related to description of time, space attributes such as *today, yesterday, this city,* and *this block*. Semantic closure also addresses the issue of alias along the label dimension. For example, on the 25th of March 2004, the semantic label *yesterday* refers to any point of time on the 24th of March 2004. Therefore, $e_{yesterday,cricket-field,rain} \ \epsilon \ C_S(e^a)$.

[2]A table of cricket related terms is provided in Appendix. II

The semantic closure of an event e is defined as:
$C_S(e) = \{e_{t_1,s_1,l_1} \mid e_{t,s,l} \wedge t \rightarrow t_1 \wedge s \rightarrow s_1$[3] $\wedge l \leftrightarrow l_1\}$.

2) Sequence closure: A sequence closure C_Q is used to determine the closure of a sequence of events in terms of individual events of the sequence. In other words, a sequence closure of an event sequence π is the set of all possible sequences that can be constructed using the events present in the closures of individual events in the sequence such that the sequence constraints are not violated.

Sequence closure of an event sequence π is given by

$C_Q(\pi) = \{\pi^1 \mid \forall\ e^1 \epsilon E^{\pi^1}\ \exists\ e \epsilon E^{\pi}\ \wedge\ (e^1 \epsilon C_G(e)\ \vee\ e^1 \epsilon C_S(e))\ \wedge\ \kappa^{\pi} = \kappa^{\pi^1}\}$

Let e be an event in the sequence. Then, the sequence closure is used to determine whether there exists any other event e′ which can be substituted for e, while still satisfying all the sequence constraints. It is apparent that if such an event e′ exists, it must belong to the closure of event e. π, π^1 are the two event sequences. The events belonging to these sequences are represented by E^{π} and E^{π^1} respectively. Every element in the event sequence π^1, belongs to the generic or semantic closures of the events in the event sequence π. $\kappa^{\pi} = \kappa^{\pi^1}$ indicates that both π and π^1 satisfy the same set of constraints. (Refer to sections III-B and III-C for the notations used.)

Note that the generic and semantic closures are defined only for basic events. As a result, every event that is generated using the closure operation is a valid event and can be derived from the event history H (see IV-A). Sequence closures represent the valid sequences that can be generated by permutations of events generated by logical closures on events of a sequence.

With this, we conclude the section which details the various composition (conjunction, disjunction, sequence) and closure (generic, semantic, sequence) operations on events. Events and event operations are used to form Event Expressions. An event expression consists of one or more events combined using the composition and closure operations along with the related constraints(if any). Observe that using event expressions is an efficient way to express meta events.

[3]Let ς denote a verbose translation of a region with respect to an observer; examples include *here, this city* etc. We have $s \rightarrow \varsigma$ if s belongs to the region $\mathcal{R}$ that is referred to by ς.

IV. Event Analysis

While dealing with real world events and trying to answer queries based on such real world events, there is a need for detailed analysis of the observed events. For example, consider the event set: $\{e^1_{2004:07:07:12:46:30,Manchester,Bowl:Vaughan}, e^2_{2004:07:07:12:46:55,Manchester,\ Miss:Sangakara}, e^3_{2004:07:07:12:47:15,Manchester,BallHitsPad}, e^4_{2004:07:07:12:47:50,\ Manchester,OutCalled:Umpire}\}$. In order to deduce that the above set of events depicts an *lbw* event, a proper semantic analysis of the observed events needs to be carried out. The event analysis is done based on an event history *H* that is compiled from events received from one or more event detectors in different locations. Event history *H* is a set of occurred, basic events (Section. III) and typically, an event needs to be analyzed in the context of those events that occurred prior to the event under consideration and those that could occur after the event. Such an analysis is required as observed or basic events are quite primitive and are not sufficient to answer complex user queries. The objective of event analysis is to analyze the events contained in *H* to derive meta-events that are of interest in answering user queries.

A. Derived events

In order to be able to process queries, it is necessary to augment the event history H and in this subsection, we briefly discuss this augmentation process. In the previous subsections, we discussed several operations (such as conjunction, disjunction, sequence and closure) related to a set of events and repeated application of one or more of these operations on H is one of the ways to augment H. The derivation rules for deriving events from H are given below:

1) $e \ \epsilon \ H \rightarrow H \vdash e$
2) $e \ \epsilon \ H \ \wedge \ e^1 \ \epsilon \ C_G(e) \rightarrow H \vdash e_1$
3) $e \ \epsilon \ H \ \wedge \ e^1 \ \epsilon \ C_S(e) \rightarrow H \vdash e_1$
4) $H \vdash e_1 \ \wedge \ H \vdash e_2 \rightarrow H \vdash e_1$ *op* e_2, where *op* represents the conjunction, disjunction or sequence operator.

Where, '$a \vdash b$' is used to denote that *b* can be derived by using *a* using a set of inference rules (here, a is a set of events and b is the event to be derived).

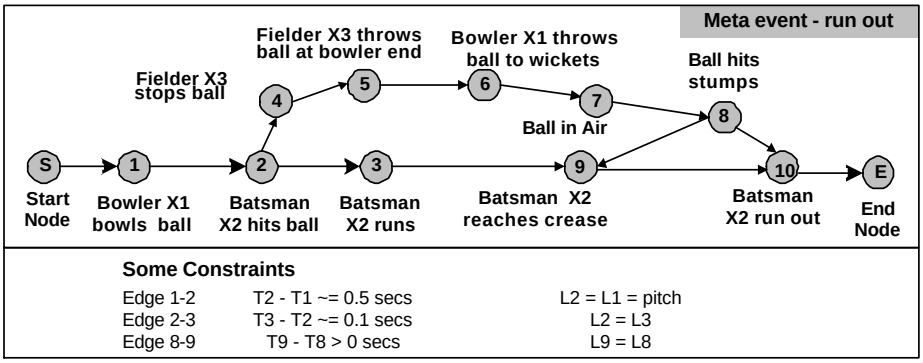


Fig. 2. Example Sequence Template

B. Identification of meta-events in H

In order to semantically characterize *H*, we need some additional information about the event space ψ. In this section, we propose the notion of capturing event semantics in the form of sequence templates. A *sequence template* is semantic characterization of a meta event that addresses the temporal and spatial relationship of a set of events from a semantic point of view. An illustrative sequence template is depicted in Fig. 2. Note that, as Fig. 2 depicts a sequence template, the actual event attributes are left unspecified. Furthermore, a sequence template defines certain important constraints on the event attributes such as temporal and spatial constraints.

Based on domain and related queries of interest, multiple sequence templates are defined and are made part of the *Sequence Template Database* (ST Database). The objective is to analyze *H* with respect to *ST Database* to generate the G_S, which is a semantic characterization of *H*. A meta event in G_S is depicted by using only the initial event(e_i) and final event(e_f) of the sequence template and a directed edge from e_i to e_f. The label of this directed edge holds the information of the instantiated sequence template corresponding to the actual meta-event that has transpired, and label hierarchies play an important role in event analysis (see section III-A). The event history H, the ST Database and the corresponding semantic representation G_S are used to develop a Query-Event Analysis System, a system that aggregates and analyzes events with a view to answer diverse user queries using the proposed event representation.

A user query can be represented as an *ordered pair (E, ψ)*, where E is the event expression and ψ is the event space that corresponds to the user query. An event expression corresponding to a user query contains the following information:

- Reference to the various events of interest in the form of partial or complete labels;
- Reference to the time and location of interest for the referenced events;

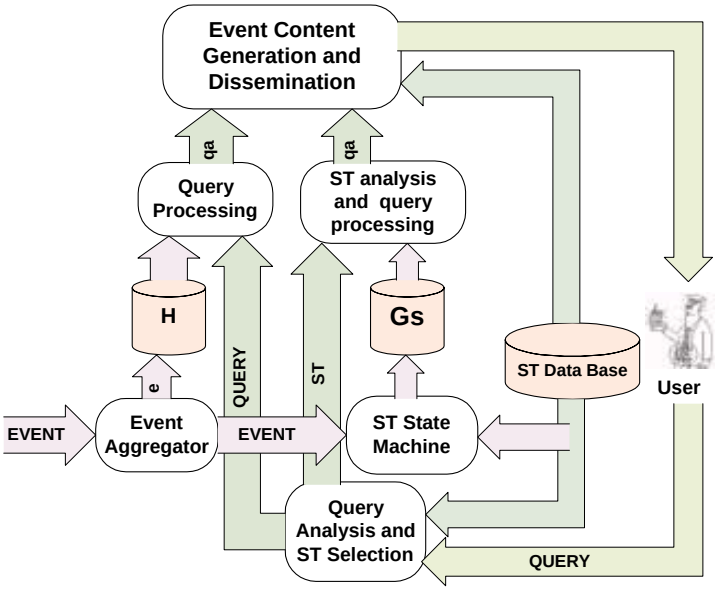


Fig. 3. Functional description of QEAS

- Relationship specifiers that specify the relationship between the various events of interest; and
- Type of event content (audio, video, text) that the user requires as output.

The Query Event Analysis System discussed here is a formal system that generates responses to user queries using the event history H and the meta events in G_S. An example of a user query in the cricket domain is "Generate video clips of all boundaries hit by each Sri-Lankan player today." In order to answer such queries, the QEAS system has to be capable of processing basic events to generate the related meta events based on various sequence templates defined in the cricket domain. Also, the QEAS system should be capable of mapping the event expression of a user query to suitable meta events and basic events, which can be used to answer the query.

V. Query-Event Analysis System (QEAS)

In this section we discuss a query and event processing system that is based on events contained in H. Query Event Analysis System is a formal system that generates responses to user queries using either H or G_S as input. Fig. 3 depicts the functional description of QEAS. The QEAS has two basic functions namely a) Analysis of input events and b) Analysis of user queries. Every new observed event must be made a part of the event history H. These events are received by the Event Aggregator, which adds the new basic event e to H. It also dispatches e to the ST State Machine, in order to verify whether the event e is a part of a meta-event.

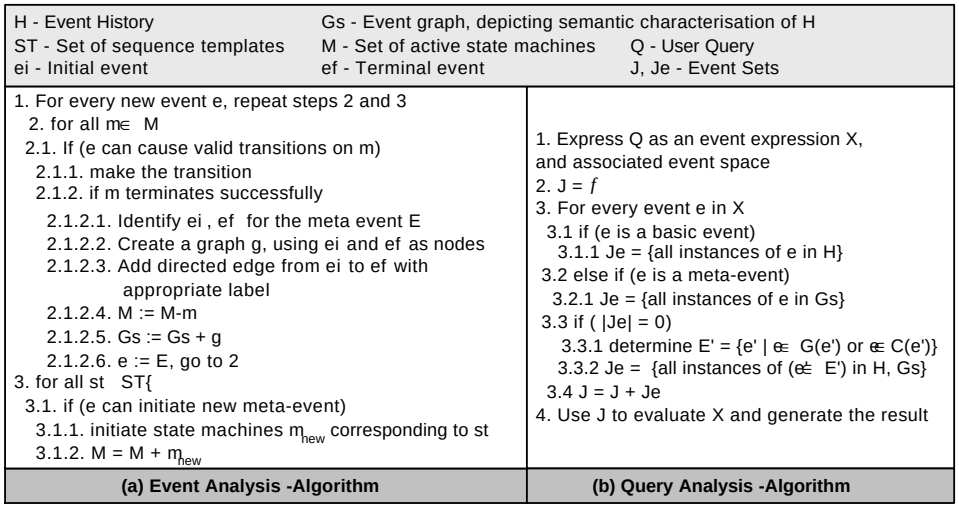


Fig. 4. Algorithms for event and query analysis

In order to identify meta events, we employ state machines, which correspond to the sequence templates of the meta event. The state machine is initiated when the first event in the sequence template of the meta event occurs. Later, as each event in the sequence template occurs, the state machine makes a transition to the next state. A transition to the final state is made, when the terminal event of the sequence template occurs. At this stage, the meta event is detected. The ST State Machine matches an event *e* to the available sequence templates in ST, and if it detects the occurrence of a meta event, it updates G_S with the new meta event. The mechanism of deriving meta events based on incoming basic events is described later in this section.

The basic events (from H) and meta events (from G_S) are used to answer user queries. A user query Q is first analyzed with respect to the available sequence templates. If a matching template is found, the query is analyzed using G_S as input. Else, the query is analyzed by the Query Processing System using H as input. The result of a query is the set of events that match the user query. The appropriate content associated with these events is sent to the user.

A. Event analysis

Events are analyzed by using state machines associated with the sequence templates, in order to identify the instantiated meta-events. A new state machine is instantiated when the start of a new meta event is detected. The occurrence of an event could (a) Cause one or more state machines to terminate successfully (b) Cause one or more state machines to make a legal transition, (c) Invalidate one or more state machines, and (d) Instantiate a new state machine *m*, which corresponds to a sequence template of a meta-event. Every time a state machine terminates, G_S

is updated to reflect the meta events that have been identified. Lapse of time/space constraints could also cause state machine invalidation. The algorithm to generate G_S is shown in Fig 4

B. Query analysis

As depicted in Fig. 3, Query analysis begins by comparing the input query with sequence templates contained in ST database. As sequence templates capture semantics, using them in query processing enables semantic analysis of a query. The objective of comparison is to select a sequence template *st* that best matches with the input query. This *st* is used in generating the query answer (q_a) using H and G_S. Finally, the content database is analyzed to extract the relevant content using *st* and q_a to generate the most appropriate event related content that is delivered to a user. The algorithm for query analysis is shown in Fig. 4(b). The QEAS analyzes every user query as being equivalent to an event expression.

In general, query processing involves three distinct steps namely (a) query pre processing (b) retrieval of event information using SQL queries and (c) post processing. The pre processing step involves identification of (a) meta event labels (b) temporal characteristics and (c) spatial characteristics of the input query. The TSL hierarchies associated with each domain play a significant role in this process. In the pre processing step, the QEAS system converts an *English query* to an event expression. The event expression is simplified, so that it consists of a sequence of basic and meta events which can be directly retrieved from the event database. Observe that the input events are analysed in real time to update H and G_S (refer Fig. 3, Fig. 4) which are stored in the database as the basic and meta event tables respectively. The retrieval of event information using SQL queries mainly uses these two tables. Post processing involves filtering and rearrangement of events to suit the user requirements. Such a three step query processing system would help in answering complex queries. Examples of such queries are:

1) Generate the list of possible candidates for the man of the match award
2) Analyze the consistency of a player's performance during the course of a match
3) Analyze the consistency of a batsman's performance against a particular bowler

As an example, consider the following query that was posed after the first innings of the match between England and SriLanka on the 27^{th} June, 2002: "Generate all boundaries hit by each SriLankan player today?" The processing of the query is depicted below:

Query Preprocessing

The query can be mapped to the event-expression:

E = ($e_{today,\ trent-bridge,\ boundary\ by\ SriLankan\ player\ X}$)$^+$ and the corresponding event space ψ = {today, trent-bridge, {Set of labels in the domain 'cricket'} }

Event closures see Section. III-C are used to simplify the event expression, generated from the query. The simplification of the Event expression, which gets evaluated for every SriLankan player X is shown below:

E = $e_{(today,trent-bridge,boundary:X)}$

= $e_{(today,\ trent-bridge,four:X)}$ | $e_{(today,trent-bridge,six:X)}$ (by generic closure)

= $e_{(<2002:06:27:10:30:xx,2002:06:27:13:30:yy>,\ trent-bridge,four:X)}$ |

$e_{(<2002:06:27:10:30:aa,2002:06:27:13:30:bb>,trent-bridge,six:X)}$ (by semantic closure)

Retrieval of Event Information

The events four:X and six:X are meta-events and are a part of G_S. The QEAS looks for the events in G_S in order to evaluate the event expression. The following SQL query is used to retrieve the information related to the event (such as link to a video clip for the event)

"`Select eventInformation from` H_S`, playerName from SriLankanPlayer where` H_S`.eventLabel = 'four' OR` H_S`.eventLabel = 'six' AND` H_S`.time LIKE '2002:06:27:%' AND` H_S`.space = 'Trent-bridge' AND` H_S`.playerName = SriLankanPlayer.playerName` "

Query Post-processing

Post processing involves generation of the content to be sent to the user. Here, post processing could involve creating a set of video clips, one per boundary to be sent to the user.

The proof for the safety and liveness properties of the QEAS system is presented in Appendix I

VI. Query Processing capabilities of the QEAS

The QEAS incorporates the three step query processing system described above and can be used to effectively answer various queries that require both extensive computation as well as the detection of complex patterns of events within the data. We have undertaken an implementation of the QEAS system in order to demonstrate the following

- The feasibility of implementing the QEAS system as proposed in the paper,
- The use of sequence templates and meta events in capturing qualitative aspects of events that occur, thereby helping in the derivation of higher order events, and
- The performance gain in query processing obtained by capturing meta events beforehand.

One of the important characteristics of meta events and sequence templates is their ability to incorporate semantic information into a sequence of events. Identification of the meta events of interest helps to reduce the time taken to process user queries. A user query would typically require the QEAS system to process a subset of both basic events and meta events in order to generate the results for the query.

Consider a typical user query *Q*, which requires *N* events to be processed, of which n_{meta} events are meta events. Let the average number of basic events per meta event be n_{avg}. The total time T_{query} taken to process the user query is directly proportional to the time taken to retrieve information from the events corresponding to the query. Let the average time taken to retrieve the information corresponding to a basic event be t_{basic} and for a meta event be t_{meta}.

When the meta events have been preprocessed,

$$T_{query} \propto t_{basic} * (N - n_{meta}) + t_{meta} * (n_{meta})$$

In the absence of any preprocessing of meta events,

$$T_{query} \propto t_{basic} * (N - n_{meta}) + t_{basic} * n_{avg} * n_{meta}$$

The time gained T_{gain} in query processing, due to the presence of pre processed meta events is given by:

$$T_{gain} \propto \frac{t_{basic}*(N-n_{meta}) \ + \ t_{basic}*n_{avg}*n_{meta}}{t_{basic}*(N-n_{meta}) \ + \ t_{meta}*n_{meta}}$$

Processing an event involves retrieving event related information such as video clip(s) associated with the event. When the time taken to process both basic and meta events can be approximated to a unit event processing time,

$$T_{gain} \propto \frac{N-n_{meta}+n_{avg}*n_{meta}}{N}$$

We observe that the time gained T_{gain} in query processing, due to the presence of pre processed meta events is considerable whenever n_{avg} is greater than 1, which is always the case.

We undertook the implementation of the QEAS system to answer user queries in the cricket domain. The implementation has four phases.

1) Building the domain database for ODI matches

Match Events:	Stroke Play Events:
a) Begin Match	a) Bat hits ball
b) Toss	b) Ball to fielder
c) Lunch Break	c) Ball in air
d) Drinks	d) Pitch (How many times?)
e) End Match	e) Ball rolls on ground
f) New over	f) Fielder runs (Who all?)
g) End of over	g) Ball crosses ropes
Run-up between wickets:	h) Ball to fielder
a) Batsman1 leaves crease	i) Ball hits Leg
b) Batsman2 leaves crease	j) Ball hits Batsman
c) Batsman1 runs	k) Fielder Dives
d) Batsman2 runs	l) Fielder Jumps
e) Batsman1 reaches crease	**Bowling Action Events:**
f) Batsman2 reaches crease	a) Run up
g) Ball hits wickets	b) Bowl
j) Ball to fielder	c) No Ball
On field events:	d) Ball in Air
a) Drop Ball	e) Pitch
b) Catch	f) Ball to Batsman
c) Stump	g) Wide

TABLE VI

LIST OF BASIC EVENTS

2) Generation of basic events from trusted sources
3) Identification of meta events, and the corresponding sequence templates.
4) Implementation of the QEAS system capable of both event analysis and query analysis

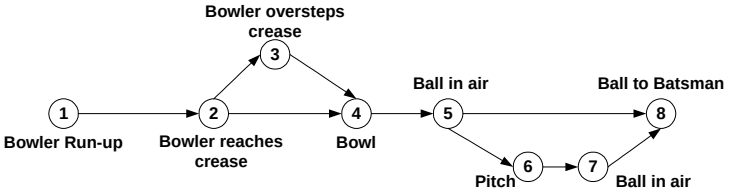
A. Generation of basic events

The basic events were generated by using the ball by ball commentaries of the cricket matches [17], and the associated videos. In order to be able to generate the large number of basic events, we developed an interface to allow a user to input the various basic events in the match using high level descriptions from the ball by ball commentaries. Table. VI below shows some of the more common basic events that were identified during the course of a cricket match.

B. Meta Events and Sequence Templates

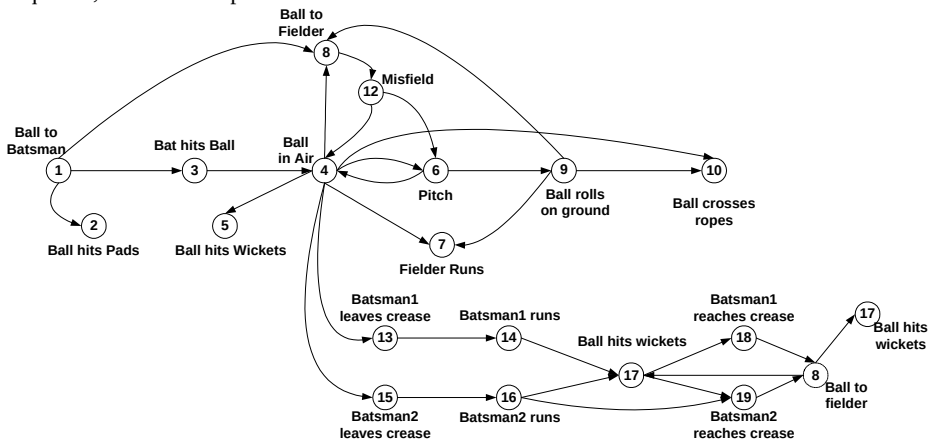
Meta events and their corresponding sequence templates are used to assess the qualitative and quantitative aspects of batting for rank ordering of batsmen. These meta-events can be combined to form higher order meta events such as bowl-sequence and strokeplay-sequence, with each higher order meta event consisting of several lower order events. Sequence templates of two

meta events bowl-sequence and strokplay-sequence are shown in Fig. 5 and Fig. 6 respectively. Some of the sub level meta events and the event sequences corresponding to these events are also shown.



Meta Event	Basic Event Sequence	TSL Attributes considered
Bad Delivery: No Ball	1->2->3	...
Bad Delivery: Full Toss	1-...->4->5->8	...
Bad Delivery: Wide	1-...->4->5->6->7->8	L8
Bad Delivery: Short	1-...->4->5->6->7->8	L4
Good Delivery: FullToss	1-...->4->5->8	...
Good Delivery	1-...->4->5->6->7->8	L6, L7, L8
Remarks: The qualification of a delivery as short, good length, etc is based on the location attribute of the pitch event.		

Fig. 5. Bowl-Sequence, with its component meta events



Meta Event	Basic Event Sequence	TSL Attributes considered
Bad Play: Miss	1->2	...
Bad Play: Miss	1->8	L1
Bad Play: Edge	1->3	L3
Good Play	1->3	L3
Catch	1->3->4->8	
Bowled	1->3->4->5	
Runs Scored	1->3->4-...>11 1->3->4->6...->8	
Four Runs	1->3->4-..6..->10	
Six Runs	1->3->4->10 1->3->4->8->12->4->10	
Gap	1->3->4-..->8 1->3->4-...->7	L4, L9, L7, L8
Stump	1->3->4...->13->14->17	L13, L14, L17 (L13 = L17)
Runout	1->3->4...->13->14->17->18	L13, L14, L17, L18 (L17=L18) T17, T18
	1->3->4...->15->16->17->19	L13, L14, L17, L19 (L17=L19) T17, T19
Remarks: The location attribute for the events GoodPlay and BadPlayindicate the location of the point of contact between the bat and the ball (whether it was an edge or middle of the bat). Locations of where the ball hit the wickets and the positions of the batsman involved are essential to differenciate between stump amd run out events.		

Fig. 6. Strokeplay-Sequence, with its component meta events

The various terms presented in this figure represent the events in a cricket match. List of MetaEvents - Run, 1Run, 2Run, 3Run, 4Run, 6Run, Miss, DeliveriesFaced, Good Delivery, GP(Good Play) The sequence tem[lates for these meta events is shown in Fig. 6 and Fig. 7 Events are represented as $Type_{Information}$, where *Information* refers to the informational attributes of an event with label *type* • Represents the sequence operator	
Quantitative Analysis: Total Quantitative. Score = Contribution * Runs' Distribution (QT=c*d)	
Contribution (c)	$\frac{\sum Run_{BatsmanX}}{\sum Run_{(Team)}}$
Runs' Distribution (d)	$\frac{\sum 1Run_{BatsmanX}}{\sum 1Run_{Team}} + \frac{\sum 2Run_{BatsmanX}}{\sum 2Run_{Team}} * 2 + \frac{\sum 3Run_{BatsmanX}}{\sum 3Run_{Team}} * 3 + \frac{\sum 4Run_{BatsmanX}}{\sum 4Run_{Team}} * 4 + \frac{\sum 6Run_{BatsmanX}}{\sum 6Run_{Team}} * 6$
Qualitative Analysis: Total Qualitative. Score = Good Play - Miss (QL=p-m)	
Miss (m)	$\frac{\sum GoodDelivery.Miss_{Batsman:X} * 0.7 + \sum BadDelivery.Miss_{Batsman:X} * 0.3}{\sum DeliveriesFaced_{Batsman:X}}$
Good Play (p)	$\frac{\sum GP.1Run_{Batsman:X} + \sum GP.2Run_{Batsman:X} * 2 + \sum GP.3Run_{Batsman:X} * 3 + \sum GP.4Run_{Batsman:X} * 4 + \sum GP.6Run_{Batsman:X} * 6}{\sum DeliveriesFaced_{Batsman:X}}$
Total Score = QT + QS	

TABLE VII

FORMULAE FOR QUANTITATIVE AND QUALITATIVE ANALYSIS FOR RANK ORDERING OF BATSMEN

C. Case study

As a case study we present the analysis of the user query for rank ordering of batsmen for the man of the match award. The rank ordering of batsmen uses both basic and meta events and is based on both qualitative and quantitative aspects of the batsmen's performance during the match. The quantitative and qualitative scores for each batsman are determined as shown in the Table. VII. The SQL query, used to extract meta-events for generating the qualitative and quantitative scores for a player is given below:

```
" Select label, time, space, eventInformation from H_S,
event from ST_Database, where H_S.batsman='<player-name>' AND
H_S.label = ST_Database.event
AND ST_Database.metaEvent = 'rankOrderingBatsman'
ORDER BY time "
```

The variable 'player-name' is replaced as appropriate.

Table. VIII which shows the statistics of the various basic and meta events that were used to generate the rank ordering of three batsmen Saeed Anwar, Sachin Tendulkar and Yuvraj Singh during the match between India and Pakistan on 1st March 2003. The table also shows the

MetaEvents	Saeed Anwar				Sachin Tendulkar				Yuvraj Singh			
	ME	BE	Qt. Score	Ql. Score	ME	BE	Qt. Score	Ql. Score	ME	BE	Qt. Score	Ql. Score
BadDelivery:Full Toss . BadPlay:Miss	0	0	0	0	0	0	0	0	0	0	0	0
BadDelivery:Noball . BadPlay:Miss	3	25	0	-0.00714	0	0	0	0	0	0	0	0
BadDelivery:Short . BadPlay:Miss	2	16	0	-0.00476	1	0	0	-0.004	0	0	0	0
GoodDelivery . Miss	10	3	0	-0.05556	6	0	0	-0.056	10	0	0	-0.13208
BadPlay:Edge . 1Run	38	380	0.132628	0	4	40	0.017534	0	0	0	0	0
BadPlay:Edge . 2Runs	1	17	0.036996	0	0	0	0	0	1	17	0.030193	0
BadPlay:Edge . 3Runs	0	0	0	0	0	0	0	0	0	0	0	0
BadPlay:Edge . 4Runs	0	0	0	0	1	4	0.048976	0	1	4	0.024988	0
GoodPlay . NoRun	42	84	0	0.166667	21	42	0	0.14	14	28	0	0.132075
GoodPlay . 1Run	3	30	0.010471	0.301587	21	210	0.092056	0.28	17	170	0.038021	0.320755
GoodPlay . 2Runs	9	153	0.332967	0.142857	3	51	0.177536	0.08	2	34	0.060386	0.075472
GoodPlay . 3Runs	4	96	1.10989	0.095238	1	24	0.355072	0.04	1	24	0.181159	0.056604
GoodPlay . 4Runs	7	35	0.470862	0.222222	12	60	0.587706	0.64	5	25	0.124938	0.377358
GoodPlay . 6Runs	0	0	0	0	2	6	1.42029	0.16	0	0	0	0
Total	119	839	2.093815	0.861111	72	437	2.69917	1.28	51	302	0.459685	0.830189
$T_{gain} \propto BE_{avg} / ME$	7.050420168				6.069444444				5.921568627			

ME - No of Meta Events	BE - No of Basic Events
Qt. Score - Quantitative Score	Ql. Score - Quanlitative Score

TABLE VIII

QUANTITATIVE AND QUALITATIVE ANALYSIS FOR RANK ORDERING OF BATSMEN

qualitative and quantitative scores which were generated during the detection of these basic and meta events. Observe that the number of basic events far exceeds the number of meta events which were used to answer the user query. Also, various qualitative aspects such as *edge* and *bad-play* that would otherwise not form a part of the rating of players have also been considered due to the presence of well-defined sequence templates. In general, the processing of basic events as and when they occur using sequence templates helps in processing of posed queries in an efficient manner. Note that a typical query in the context of cricket would involve processing of events during some N number of balls.

Let B be the number of basic events per ball. For the scenario of the Fig. 5, B is about 7. Let M be the number of meta events that are of significant interest for the query under consideration.

Each such meta event has an associated sequence template and gets defined using a sequence of basic events. Let M_B be the number of basic events used to describe a meta event M. For the meta event 'rank ordering of batsmen' described in Table. VIII, M_B is around 6.

D. Summary

The implementation of the QEAS system as proposed in this paper demonstrates the feasibility of our approach for any given domain of interest, as well as demonstrates the ease of query processing, due to pre processing of meta events. The case study presented here, demonstrates the time gained during processing the query 'Rank ordering of batsmen.' The average number of basic events per meta event (n_{avg}) is around 6.5 (from Table. VIII) , and this translates into an average performance gain of 6.33 in the case study.

VII. Conclusions and Future Work

In this paper, we have described an approach for representing and analyzing real world events. Events are characterized along three dimensions, namely, time, space, and label dimensions. Such a multidimensional characterization helps in better syntactic and semantic analysis of events. We have defined the notion of event closures and have also introduced the notion of sequence templates that are useful in (a) providing a semantic structure (G_S) to the event history H; and (b) characterizing content related to meta events. We have described the main steps involved in the generation of G_S given H and have illustrated the use of G_S in answering queries related to real world events. This paper presents a technique to abstract event processing by using meta-events, and this technique reduces the runtime during query processing.

The Query-Event Analysis System describes an important step of event related content identification and dissemination. We have illustrated the usefulness of the QEAS in answering complex queries, using real world examples. The proof of concept implementation of the QEAS was undertaken in order to validate the feasibility of putting the proposed system to use.

The representation of user queries in the form of event expressions, and the subsequent mapping of meta events to user queries as proposed in this paper, help in generating optimal information content that can be disseminated to the user. From the point of view of event related content dissemination, it is desirable to derive the meta events that are of interest to users as soon a basic event occurs. We are focusing our efforts on how to (a) identify all such meta

events; (b) identify related content; (c) identify users who are interested in the occurred and meta events; and (d) efficiently and effectively deliver content. Observe that some of the users could be mobile, demanding effective caching and transcoding techniques. Our objective in event representation and analysis is to ultimately deliver content to mobile users with minimum delay. The described query processing system (QEAS) can be enhanced by exploiting context related information [16] and delivering content in an enhanced way by exploiting network and server characteristics [9].

ACKNOWLEDGMENT

The authors would like to thank Dr. Alex Buchmann for his insightful comments on an earlier version of this paper.

REFERENCES

[1] R. Adaikkalavan, *Snoop Event Specification: Formalization Algorithms, And Implementation Using Interval-Based Semantics*. MS thesis, The University of Texas At Arlington, 2002.

[2] J. F. Allen, "Time and time again: the many ways to represent time," *International Journal of Intelligent Systems*, vol. 6, pp. 341–355, 1991.

[3] P. Bates., "Debugging heterogeneous distributed systems using event-based models of behavior," *ACM Transactions on Computer Systems*, pp. 1–31, 1995

[4] L. N. de Barros et al., "Model Based Diagnosis for Network Communication Faults," *procs of International Workshop on Artificial Intelligence for Distributed Information Networking (AIDIN'99)*, Orlando, EUA, Jul. 1999.

[5] H. Branding, A. P. Buchmann, T. Kudrass, J. Zimmermann., "Rules in an Open System: The REACH Rule System," in *Procs of 1st Intl. Workshop on Rules in Databases*, (Edinburgh), Sept. 1993.

[6] S. Chakravarthy, V. Krishnaprasad, E. Anwar, S. K. Kim., "Composite Events for Active Databases: Semantics, Contexts and Detection," in *Procs of 20th International Conference on Very Large Data Bases (VLDB 1998)*, (Santiago de Chile, Chile), 1994.

[7] M. E. Dönderler, Ö. Ulusoy., U.Güdükbay., "Rule based spatiotemporal query processing for video databases," *The VLDB Journal*, vol. 12, 2004, 86–103.

[8] S. Gatziu, K. R. Dittrich, "SAMOS: an Active Object-Oriented Database System," *IEEE Quarterly Bulletein*, Jan 1993.

[9] D. Gujjar, A. Thawani, S. Gopalan, S. Varadarajan., "An Efficient Event Management System for Distributed Multimedia Services,"*IASTED conference on "Internet and Multimedia Systems and Applications"(EuroIMSA 2005)*, Beijing, China, 2005.

[10] M. Z. Hasan, *The Management of Data, Events, and Information Presentation for Network Management*. PhD thesis, University of Waterloo, 1996

[11] B. Kuijpers, J. Paredaens, L. Vandeurzen., "Semantics in Spatial Databases," *Semantics in Databases, LNCS 1358*, pp. 114–135, 1998.

[12] C. Liebig, M. Cilia, A. Buchmann., "Event Composition in Time-dependent Distributed Systems," in *Procs of the 4th IFCIS (CoopIS 99)*, 1999.

[13] G. Liu, A. Mok, P. Konana., "A Unified Approach for Specifying Timing Constraints and Composite Events in Active Real-Time Database Systems," in *Procs of the Fourth IEEE Real-Time Technology and Applications Symposium (RTAS 98)*, 1998.

[14] A. Nagargadde,S. Varadarajan, K. Ramamritham., "Semantic Characterization of Real World Events," *procs of The 10th International Conference on Database Systems for Advanced Applications(DASFAA 2005)*, Beijing, China, 2005.

[15] P. R. Pietzuch, B. Shand, J. Bacon, "Composite Event Detection as a Generic Middleware Extension," *IEEE Network Magazine, Special Issue on Middleware Technologies for Future Communication Networks*, pp. 44–55, 2004.

[16] A. Thawani, S. Gopalan, S. Varadarajan., "Event driven semantics based ad selection," in *Procs of IEEE International Conference on Multimedia and Expo (ICME'2004)*, Taipei, Taiwan, June 2004.

[17] http://plus.cricinfo.com/link_to_database/Archive/2002/OD_Tourneys /NWS/Scorecards/ENG_SL_NWS_ODI7_07JUL2002_BBB-COMMS.html.

APPENDIX I
SAFETY AND LIVENESS PROPERTIES OF QEAS

The event history, along with the derivations, is used to develop the QEAS. The QEAS system itself works on a set of events belonging to a particular domain. The set of axioms on which the QEAS system is based are given below:

Axiom 1: There exist no cyclic dependencies between the meta-events within a domain.

There exists a dependency from meta-event E^1 to E^2 if the meta-event E^2 is a *component-event* of E^1.

Axiom 2: The event input is temporally ordered

At any point in time, if the QEAS receives an event with the time-stamp t; it shall not receive any other event with time-stamp $t' < t$. This rule ensures that the events received are temporally ordered and that the inputs from various sources are temporally synchronized.

Axiom 3: Every state machine associated with a meta event is temporally consistent.

A state machine M is associated with a meta event, that can be represented as an event sequence. From the definition of event sequence, if event e^b follows event e^a then $t_b > t_a$ (also supported by Assumption 2). If a state machine M contains two successive transitions f_a and f_b (f_b follows f_a), which are caused by events e^a and e^b respectively; then for a valid transition, e^a must occur before e^b (i.e. $t_a < t_b$). A state-machine that conforms to this rule is temporally consistent. Temporal consistency of state machines is important in order to avoid repeated backtracking while detecting events.

The following theorems prove the liveness and safety property of QEAS.

Theorem 1: With the assumption that no two events occur simultaneously, an occurred event alone is adequate to identify all the consequential meta-events in G_S

Proof: by contradiction

Initial Assumption: $\exists$ a meta-event E, such that H $\vdash$ E $\wedge$ E $\notin G_S$

H $\vdash$ E $\Rightarrow \exists$ H_1 $\subseteq$ H $\wedge$ H_1 $\vdash$ E, where H_1 = $\{e_s, e_1, e_2, \ldots e_n, e_t\}$; where e_s and e_t are the initial and final events within the meta-event E.

$\Rightarrow \{e_1, e_2, \ldots e_n\}$ is temporally ordered (from Assumption 3)

$\Rightarrow$ The state machine M corresponding to E is traversed when event e_s occurs (from Step 3.1, Fig. 4a); The events $e_1, \ldots, e_n$ cause valid transitions on the M (from Step 2.1, Fig. 4a); M terminates successfully on input of e_t.

$\Rightarrow$ E $\in G_S$ (from Step 2.1.2, Fig. 4a)

This contradicts our initial assumption E $\notin G_S$ and hence the theorem. ■

Corollary 1: If a sequence of events E that causes valid transitions on state machine M appears in the event history H, then the events e $\in$ E are used to traverse M to recognize the corresponding meta-event.

We use the results of Theorem 1 to prove the safety and liveness properties of QEAS.

Theorem 2: **The Safety and Liveness Theorem:**

The QEAS is capable of answering a user query Q if and only if the event history H derives the corresponding event expression E_Q.

We shall represent the same as: QEAS $\dashrightarrow$ Q $\Leftrightarrow$ H $\vdash E_Q$

Proof: The user query Q represents an event-expression that can consist of (a) basic events, (b) meta-events and (c) events derived by closure. We show that the theorem is true for all these three cases.

Case 1:

QEAS $\dashrightarrow$ Q $\wedge$ Q $\rightarrow$ e, e is a basic event

$\Leftrightarrow$ (e $\in$ H) $\vee$ (e $\in G_S$) $\wedge$ e is a basic event

$\Leftrightarrow$ (e $\in$ H)

$\Leftrightarrow$ H $\vdash$ e $\equiv E_Q$ (and hence Q)

Case 2:

QEAS $\dashrightarrow$ Q $\wedge$ Q $\rightarrow$ e, e is a meta-event

$\Leftrightarrow$ (e $\in$ H) $\vee$ (e $\in G_S$) $\wedge$ e is a basic event

$\Leftrightarrow$ (e $\in G_S$)

$\Leftrightarrow$ $\exists$ H_1 $\subseteq$ H $\wedge$ H_1 $\vdash$ E (from Theorem 1)

$\Leftrightarrow$ H $\vdash$ e $\equiv$ E_Q (and hence Q)

Case 3:

QEAS $\dashrightarrow$ Q $\wedge$ Q $\rightarrow$ e, e is derived from closure operation.

$\Leftrightarrow$ $\exists$ e$'$ such that (e$'$ $\in$ H $\vee$ e$'$ $\in G_S$) $\wedge$ e$'$ $\rightarrow$ e

$\Leftrightarrow$ ((e$'$ $\in$ H) $\vee$ ($\exists$ H_1 $\subseteq$ H $\wedge$ H_1 $\vdash$ e$'$)) $\wedge$ e$'$ $\rightarrow$ e

$\Leftrightarrow$ H $\vdash$ e $\equiv$ E_Q (and hence Q)

Therefore: QEAS $\dashrightarrow$ Q $\Leftrightarrow$ H $\vdash$ E_Q (and hence Q) ■

APPENDIX II

TABLE OF A FEW COMMON TERMS IN CRICKET

Pitch	The pitch is a rectangular area of the ground between two bowling creases and is at the center of the cricket-field. The batsmen hit the balls bowled to them and run between the wickets on the pitch to score the runs.
ODI	A form of cricket that is completed in a single day
Bye	A run generally scored through any means other than being struck by the bat
Catch	A method of dismissal wherein the ball after being hit by the batsmen, is caught by a fielder, without touching the ground even once
Dot ball	A delivery (ball bowled) in which no runs are scored
Extra	The batting team is awarded an extra run/ball or both
Four	The ball after being hit by the batsmen, crosses the boundary ropes, after pitching once or more on the ground
Hat-Trick	When a bowler is able to manage 3 dismissals from 3 consecutive deliveries in the same match.
Inning	The period of time spent batting by a team or individual
LBW	Better known as Leg Before Wicket. This is a method of dismissal where the ball when having been bowled would normally strike the stumps if not for the fact it strikes the batsmans leg first

Leg Bye	A run scored after the ball hits the batsman's leg, and is not struck by the bat
Maiden	When applied to a bowler, describes an over where no runs have been scored by the batsman from any delivery. For batsman, it refers to a maiden innings or maiden century, both being the very first occasion of each.
No Ball	A delivery in which the bowler's front foot crosses the crease
Over	The set number of balls bowled by a bowler. An over consists of 6 balls. The term Over is also called by the umpire when the bowler has completed his 6 balls
Partnership	Refers to the batting performance by two particular batsmen whilst batting together during any particular innings. Or to the cumulative score made by two partnering batsmen.
Run Out	A method of dismissing the batsman by disturbing the stumps before the batsman has made his ground and is within the batting crease.
Six	The ball after being hit by the batsmen, crosses the boundary ropes, without touching the ground even once
Stump	A method of dismissing the batsman by disturbing the stumps, before the batsman begins the run, if he's outside the batting crease
Test Cricket	Longest Form of cricket, played over five full days
Wicket	a dismissal by a bowler
Wide	A delivery such that the ball is out of reach of the batsman.