VideoGraph: A Graphical Object-based Model for Representing and Querying Video Data

Duc A. Tran, Kien A. Hua, and Khanh Vu

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816, USA.
{dtran, kienhua, khanh}@cs.ucf.edu

Abstract. Modeling video data poses a great challenge since they do not have as clear an underlying structure as traditional databases do. We propose a graphical object-based model, called VideoGraph, in this paper. This scheme has the following advantages: (1) In addition to semantics of video individual events, we capture their temporal relationships as well. (2) The inter-event relationships allow us to deduce implicit video information. (3) Uncertainty can also be handled by associating the video event with a temporal Boolean-like expression. This also allows us to exploit incomplete information. The above features make VideoGraph very flexible in representing various metadata types extracted from diverse information sources. To facilitate video retrieval, we also introduce a formalism for the query language based on path expressions. Query processing involves only simple traversal of the video graphs.

1 Introduction

We deal with the modeling aspect of *video database management systems* - VDBMSs [8] in this paper. Playing an important role in VDBMSs, this is the process of designing the high-level abstraction of raw video to facilitate various information retrieval and manipulation operations. It determines what features are to be used in the retrieval, and therefore in the indexing process. Other components, such as content analysis tools and query processing techniques, are also more or less dependent on it. A good model is essential to enabling a wider range of applications.

Much research has been done in the area of video modeling/retrieval based on audio-visual content, such as audio, color, texture and motion (e.g., [13, 11, 3, 4, 18, 7]). The advantage of this approach is that features can be extracted automatically. The low-level schemes, however, are very limited in expressing queries. For example, it would be difficult to ask for a video clip showing the sinking of the ship in the movie "Titanic" using only color, texture, and audio information. In contrast, video data models based on semantic content ([6, 15, 14, 10, 1, 9, 5, 17, 12]) are capable of supporting more natural queries. They, however, must rely partially on manual annotation. A limitation of this approach is that semantic content can be ambiguous and context dependent. This problem can be

controlled by limiting the context and providing multiple semantic descriptions for different types of applications. We focus on the semantic level in this paper. In particular, we consider two types of video semantics:

- Event Description: This type of description indicates the video segments that show a particular event. Some examples of event description are ["Ship colliding with iceberg", 35^{th} minute - 37^{th} minute] and ["Captain dying", 48^{th} minute - 49^{th} minute]. The first description indicates that the 2-minute long segment, from time 35^{th} minute to time 37^{th} minute, shows the scene of a ship colliding with an iceberg. Similarly, the second description indicates that the scene of the captain dying begins at the 48^{th} minute and ends at the 49^{th} minute of the video.
- Inter-event Description: This type of description describes the temporal relationship between two events. Some examples of interevent description are ["Ship collides with iceberg before it sinks "] and ["Captain dies after the ship sinks"]. This type of semantic information is not associated with any video segment. Instead, it states the temporal relationship between two events. As an example, such information can be obtained from the script. Under this circumstance, we can report on the order of various events, but not the exact locations of their occurrences in the video stream.

We note that each event mentioned in an inter-event description may or may not have an explicit event description. In the first inter-event description given above, only the "colliding" event has an event description (i.e., from the 35^{th} minute to the 37^{th} minute). This situation arises in practice since information extractors are not perfect. As an example, an extractor based on explicit models may recognize a ship and the "colliding" scene, but lacks the knowledge to determine the "sinking" event.

Existing semantic-level video data models support only event, not the inter-event, descriptions. In this paper, we introduce a new model, called VideoGraph, which can accommodate both in one framework. This enables us to deduce implicit event descriptions, and therefore retrieve implicit video information as well. For instance, using the following metadata: ["Ship collides with iceberg before it sinks"], ["Captain dies after the ship sinks"], ["Ship colliding with iceberg", 35^{th} minute - 37^{th} minute], and ["Captain dying", 48^{th} minute - 49^{th} minute], we can imply the implicit temporal description ["Ship sinking", 37^{th} minute - 47^{th} minute] This implicit semantic allows us to answer queries such as "showing the scene of the sinking ship". Existing techniques based on only explicit descriptions would fail to process these queries. Our new capability is reminiscent of implicit information in a deductive database management system. To the best of our knowledge, video semantic implicity has not been exploited in the literature.

Another new feature considered in our model is the flexibility to associate an event with a temporal Boolean-like expression. This enhancement allows us to handle uncertainty. For instance, we can associate the event "Jack went aboard the ship" with the temporal description "[5^{th} minute, 7^{th} minute] or [9^{th} minute, 10^{th} minute]." This expression indicates that the event must be in one of the two video segments. Such conditions occur when we infer implicit video semantics

from incomplete semantics. Thus VideoGraph can also deal with incomplete information, in addition to implicit information. To facilitate video retrieval, we present a formal query language for VideoGraph. The language is an extension to relational calculus with path expressions and has both a clear declarative and operational semantics. It is simple yet powerful enough to allow formulation of complex queries. Query processing involves only simple graph traversal.

The remainder of this paper is organized as follows. We formally present the details of VideoGraph in Section 2. The query language formalism is described in Section 3. The algorithm for computing the implicit video information is presented in Section 4. Finally, Section 5 summarizes our approach and indicates some directions for future research.

2 Video Data Representation

In this section, we introduce a new model called VideoGraph that has the desirable features discussed earlier. This is a follow-up of our previous work in [16]. Intuitively, a VideoGraph database is a set of edge labeled rooted graphs, each representing the semantics of a single video. Such a graph is a collection of nodes, each in turn representing a single event. Nodes are linked to each other based on their containment and temporal relationships [2]. The relationship between two nodes captures inter-event information involving the two corresponding events. A node may be associated with explicit temporal information or not. If not, its implicit information can still be obtained by reasoning on the graph. We discuss this in section 4.

Let us assume that **ATYPE** denotes the set of all integers, real numbers and strings. Let **TYPE** be a finite set containing special strings classified as data types in the video database, in other words, **TYPE** = {$type_1$, $type_2$, .., $type_p$, **TIME**} where $type_i$ is a string. Each type $type_i$ has a value domain, denoted as dom($type_i$), such that dom($type_i$) $\subseteq$ **ATYPE** if $i \leq p$, and dom(**TIME**) = Ω that is defined later. In what follows, unless we explicitly mention, concepts to be defined are considered within the same context of a single video.

Definition 1. *[Atomic object] Let l_1, l_2, .., l_n ($n \geq 1$) be strings in* **TYPE**, *v_1, v_2, .., v_n values such that $v_i \in dom(l_i)$. Let us consider a graph G containing a node O, called the* root *of G, and n nodes O_1, O_2, .., O_n with the following properties:*

- *Node O stores a unique integer value.*
- *For each $i \in \{1, 2, .., n\}$, node O_i stores value v_i.*
- *For each $i \in \{1, 2, .., n\}$, there is a directed link labeled l_i from O to O_i.*

Then node O is said to be the atomic object *represented by graph G and the value stored in the node is called the* identifier *of the atomic object.*

Definition 2. *[Object] Objects are recursively defined as follows.*

1. *Any atomic object is an object.*

2. *Let G_1, G_2, .., G_n represent n objects, O_1, O_2, .., O_m be m single nodes $(n+m>0)$ and l_1, l_2, .., l_{n+m} be $n+m$ strings in* **TYPE**. *Then the graph G built below represents an object.*
   - *G contains a node O, which is called the* root *of G, nodes O_i's and graphs G_j's for $i \in \{1, 2, .., m\}$ and $j \in \{1, 2, .., n\}$.*
   - *Node O stores a unique identifier for the object.*
   - *For each $i \in \{1, 2, .., n\}$, there is a directed link from O to the root of G_i labeled l_i.*
   - *For each $i \in \{1, 2, .., m\}$, there is a directed link from O to O_i labeled l_{i+n} and node O_i stores a value $v_i \in dom(l_{i+n})$*

We can say that node O is the object *represented by graph G.*

Here after, for flexibility, the terms object and internal node are interchangeably used. That is, we implicitly refer to an object as an internal node and vice versa. Links between any two nodes in the above definitions are called *c-links*. If a node O_1 has a c-link labeled l departing from it and going to another node O_2, then l is a *component* of O_1 and the *type* of node O_2 with respect to O_1. O_2 is the *value* of component l of O_1. Components of an object can be duplicated. Furthermore, if O_2 is an internal node, it is also called a *sub-object* of O_1 which in turn is said to be a *super-object* of O_2.

We divide the set of objects into two classes, key objects and non-key objects. Informally, a key object is an object that has temporal related information telling what parts of the video associate with the object. That can be complete or incomplete (that is, the exact video segment for an event is not known). Before giving the formal definition of a key object, we need to describe a new type for video temporal values.

Definition 3. *[I-expression]* I-expressions *(I stands for "interval") are defined as follows: (1) For any t_1 and t_2 integers $(t_1 \leq t_2)$, $[t_1, t_2]$ is an i-expression. It is also classified as an* interval. *(2) If p and q are two i-expressions, then so are (p&q) and (p | q). The meaning of operations "&" and "|" is that if an object associates with an i-expression p&q (or p | q), then it associates with both (or one) of p and q.*

Let Ω denote the set of all i-expressions. We recall that dom(**TIME**) = Ω. I-expressions tell how to look up the video and can be used to express incomplete information. For instance, [10, 20] corresponds to the video segment from time 10 to time 20; [15, 18] & [25, 30] corresponds to a set of two video segments, one represented by [15, 18] and the other one by [25, 30]; [25, 28] | [15, 20] corresponds to only one video segment, [25, 28] or [15, 20], but it is not known to be which.

Definition 4. *[Key object] If O is an object having* **TIME** *as a component and its corresponding* **TIME** *value is an i-expression, then O is categorized as a* key object.

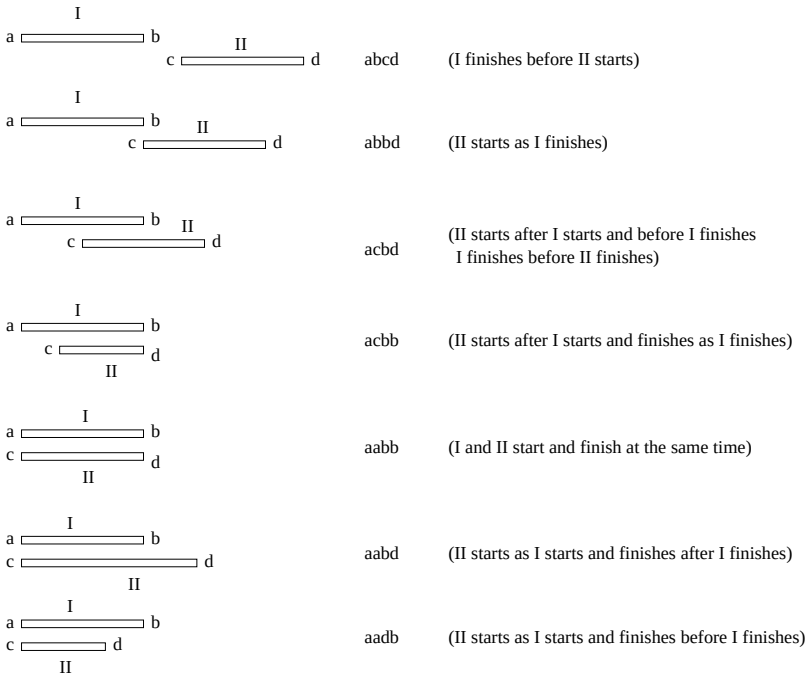

Fig. 1. Temporal relations between two intervals

A video graph is built on objects each corresponding to a single event in reality and their temporal relations reflects inter-event descriptions. An inter-event relation can be described as an element of the set **REL** = {ABBD, ABCD, ACBD, ACDB, AACD, AABB, ACBB} - the set of seven temporal interval relations illustrated in Figure 1 where I and II are two events in the video, and [a, b] and [c, d] are their video temporal segment information respectively.

Definition 5. *[Video graph] Given a video V, let graphs G_1, G_2, .., G_n ($n \geq 1$) represent its objects, that are not sub-objects of any others, the video graph of V is an edge labeled rooted graph G defined as follows: (1) The root A of the graph stores the identifier of V; (2) G contains every graph G_i for $i \in \{1, 2, .., n\}$; (3) For each $i \in \{1, 2, .., n\}$, there is a link from A to the root of G_i labeled* **SEM***; (4) If two internal nodes O_1 and O_2 of G have a temporal relation $r \in$* **REL***, then there is a directed link from O_1 to O_2 labeled r and classified as an* r-link.

A video database *consists of a number of video graphs, each representing knowledge about an individual video in the database.*

The VideoGraph model encompasses both video data and the structure of them. Before going any further, let us give an example of a video database. We are interested in a single movie and the video graph for it is shown in Figure 2 where directed solid lines describe c-links and dotted curves describe r-links. In this video graph, the atomic objects are o_4, o_7, o_9, o_{10}, o_{11} and o_{12}. The key objects

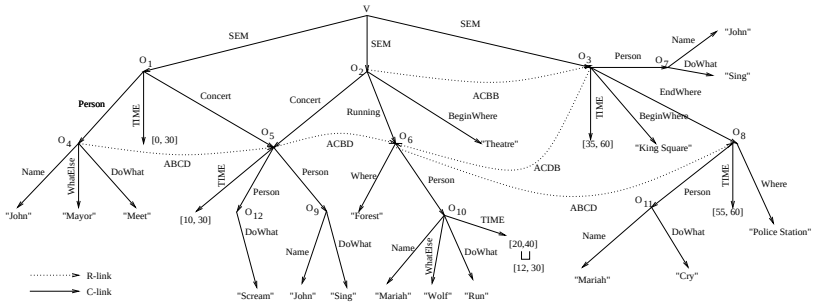

Fig. 2. An example of video graphs

are o_1, o_3, o_5, o_8 and o_{10}. The others are non-key objects. Note that o_{10} has incomplete temporal information ([12, 30] | [20, 40]) because it is not known that node o_{10} associates with which interval, [12, 30] or [20, 40]. VideoGraph allows different kinds of temporal information, which are encapsulated in one type Ω (i-expressions). This distinct property was not possible in previous models. Hence, they have limited capabilities of utilizing semantics extracted from diverse and dynamic knowledge sources because all the events that do not have a temporal description or that have incomplete temporal information are not considered.

3 Query Language

We will now introduce a content-based mechanism to support users' querying the video database. We focus on answers of the following types: (1) Video segments: When the user searches for video segments that satisfy some semantic constraint. E.g., "I would like to watch those scenes where the ship was going to crash into an iceberg". (2) Semantic contents: When the user asks for information about a video clip. E.g., "Tell me what the last scenes of the movie are about?"

We limit our queries to retrievals only, however update queries can be easily embedded. Our video query language is based on path expressions formulated according to the graphical structure of the database. The queries are also expressed on the inter-object temporal relationships. Prior to formally presenting what the syntax and semantics of queries are, we give some necessary definitions.

Without loss of generality, we consider only one video graph in the database which we are going to formulate queries on. Let o_1 and o_2 be two nodes and L labeled l be a link from o_1 to o_2. Then we can express $o_1 = \mathbf{from}(L)$ and $o_2 = \mathbf{to}(L)$.

Definition 6. *[Strict Path Sample] A* strict path sample *is of the form* $l_1 \rightarrow l_2 \rightarrow ... \rightarrow l_n$ *where* $l_1, l_2, .., l_n \in$ **TYPE** *and* $\forall\ i \in \{1, 2, .., n-1\}$, *there exist two c-links* L_i *and* L_{i+1} *labeled* l_i *and* l_{i+1} *respectively such that* $\mathbf{to}(L_i) = \mathbf{from}(L_{i+1})$.

Definition 7. *[Path Sample] Let θ be either $\rightarrow$ or $\stackrel{*}{\rightarrow}$ and l_1, l_2, .., $l_n \in$* **TYPE**. *A* path sample *is of the form $l_1\ \theta\ l_2\ \theta\ ..\ \theta\ l_n$ where there exist components l_{i1} , l_{i2} , .., l_{iki} ($i \in \{1, 2, .., n-1\}$) such that $l_1 \rightarrow l_{11} \rightarrow l_{12} \rightarrow ..\rightarrow l_{1k1} \rightarrow l_2 \rightarrow l_{21} \rightarrow l_{22} \rightarrow .. \rightarrow l_{2k2} \rightarrow .. \rightarrow l_n$ is a strict path sample.*

Our query language is based on the expression of paths. We allow the user to formulate any path he or she is interested in. The path may or may not exist in the graph, in other words, it either conforms to a path sample or does not conform to any. In order to facilitate users' querying, we introduce the notion of path expression.

Definition 8. *[Path Expression] Let O be an internal node, l be a label, θ be either $\rightarrow$ or $\stackrel{*}{\rightarrow}$. A* path expression *is recursively defined as follows: (1) $O.$ is a trivial path expression, which contains only a node; (2) If α is a path expression, then so are $\alpha\theta$ and $\alpha\theta(O)$.*

A query will be run successfully if it contains path expressions, each conforming to some path sample of the video graph. Otherwise, the query returns nothing. For the semantics of the query, we define the validity property of a path expression below.

Definition 9. *[Path Validity] Let θ be either $\rightarrow$ or $\stackrel{*}{\rightarrow}$. The validity of a path expression is presented as follows:*

1. *Any trivial path expression is a* valid *path expression*
2. *A non-trivial path expression PE is* valid *if the conditions below hold:*
   - *If $O.\ \theta l$ appears in PE, then there must exist a c-link L labeled l such that $O =$* **from**(L)
   - *If $l_1\ \theta\ l_2$ appears in PE, $l_1\theta l_2$ must be a path sample*
   - *If $l_1(O)\ \theta\ l_2$ appears in PE, then $l_1\theta l_2$ must be a path sample and there must be two c-links, L_1 labeled l_1 and L_2 labeled l_2, such that $O =$* **to**$(L_1) =$ **from**(L_2)
   - *If $l_1\theta l_2(O)$ appears in PE, $l_1\theta l_2$ must be a path sample and there must exist a c-link L labeled l_2 such that $O =$* **to**(L)

Having presented the above concepts, we are ready to give a description of queries and their semantics. Our queries use three salient operators, $\bigtriangledown$, $\oplus$, $\otimes$. Given a node A, $\bigtriangledown(A)$ gives the set of all nodes that are reachable from it by traversing the graph (Node B is said to be reachable from node A if and only if there exists a valid path expression starting with A and ending with B). $\oplus(A, B)$ returns the temporal relation between two internal nodes A and B. $\otimes(A, ie)$ returns the temporal relation between the **TIME** value of internal node A and an i-expression ie. The result of $\oplus$ and $\otimes$ operations must be an element of the set **REL**.

3.1 Syntax of VideoGraph queries

Definition 10. *[Atomic Condition] Let assume that $\odot$ is an operator in the set $\{<, >, =, \geq, \leq\}$, PE and PE' are path expressions, o_1 and o_2 are internal nodes, ie is an i-expression, $r \in$* **REL**, *$v \in$* **ATYPE**. *An atomic condition has one of the forms: (1) PE; (2) PE $\odot$ v; (3) PE $\odot$ PE'; (4) $\oplus$(PE, PE') = r; (5) $\otimes$(PE, ie) = r; (6) $o_1 \in \bigtriangledown(o_2)$.*

Definition 11. *[Condition] Let p and q be themselves conditions, f(O) be a condition in which O appears, a* condition *is recursively defined to be one of the following: (1) any atomic condition; (2) $\neg p$, $p \wedge q$, $p \vee q$, or $p \Rightarrow q$; (3) $\exists O(f(O))$, where O is a variable representing an internal node; (4) $\forall O(f(O))$, where O is a variable representing an internal node.*

In this definition, $\exists$ and $\forall$ are two quantifiers in traditional logic and are said to *bind* to the variable O.

Definition 12. *[Free variable] A variable is said to be* free *in a condition or a sub-condition (a condition contained in a larger condition) if the (sub-)condition does not contain an occurrence of a quantifier that binds it.*

Now is time for the formal syntax of a VideoGraph query.

Definition 13. *[VideoGraph Query] A* VideoGraph query *is defined as an expression of the form:*

$$(TIME) \Leftarrow PE(O_1, O_2, .., O_n), SN, SC(O_1, O_2, .., O_n) \quad (1)$$

or

$$(SEM) \Leftarrow PE(O_1, O_2, .., O_n), SN, SC(O_1, O_2, .., O_n) \quad (2)$$

where SN is a predefined node, O_i's ($i = 1..n$) are internal node variables and the only free variables in the formulas SC(O_1, O_2, .., O_n) and PE(O_1, O_2, .., O_n), SC(O_1, O_2, .., O_n) is a condition containing one or more occurrence of each O_i , PE(O_1, O_2, .., O_n) is a path expression containing one or more occurrence of each O_i, **TIME**, **SEM** *are special symbols describing what kinds of output are to be returned, an i-expression (temporal information) or a set of nodes (objects).*

3.2 Semantics of VideoGraph queries

To complete the formalism of the query language, we must state which value assignments to free variables in a condition make the condition true. A query is evaluated in any given instance of the video database. Let each free variable O_i in a condition $SC(O_1, O_2, .., O_n)$ (we call it F for brevity) be bound to a value o_i (a node in the graph). With respect to the video database and for the assignments of values to variables, the condition F must be true if one of the following holds ($\odot \in \{<, >, =, \leq, \geq\}$):

- F is an atomic condition PE, and PE is a valid path expression.
- F is an atomic condition $PE \odot v$, and PE is a valid path expression which by traversing we can obtain a node value v' (an atomic value or an object identifier) that makes the comparison $v' \odot v$ true.
- F is an atomic condition $PE \odot PE'$, and PE, PE' are valid path expressions which by traversing we can obtain two nodes whose values, v_1 and v_2, make $v_1 \odot v_2$ true.
- F is an atomic condition $\oplus(PE, PE') = r$, and PE, PE' are valid path expressions which by traversing we can obtain two internal nodes such that their implicit or explicit **TIME** values are related to each other by relation r.
- F is an atomic condition $\otimes(PE, ie) = r$, and PE is a valid path expression which by traversing we can obtain an internal node whose implicit or explicit temporal relationship with the i-expression ie is equivalent to relation r.
- F is an atomic condition $o_1 \in \bigtriangledown(o_2)$, and o_1 is reachable from o_2.
- F is of the form $\neg p$, and p is not true; or of the form $p \wedge q$, and both p and q are true; or of the form $p \vee q$, and one of them is true; or of the form $p \Rightarrow q$, and q is true whenever p is true.
- F is of the form $\exists O(f(O))$, and there is some assignment of values to the free variables in $f(O)$ and variable O, that makes it true.
- F is of the form $\forall O(f(O))$, and there is some assignment of values to the free variables in $f(O)$ that make it true no matter what value is assigned to variable O.

Now we need to be clear how the answer of a query is returned, that is, we need to formally define what the semantics of $PE(O_1, O_2, .., O_n)$ is, given an assignment of values to variables $O_1, O_2, .., O_n$. The query returns nothing if PE is not a valid path expression. Otherwise, PE represents a set of nodes that are obtained by traversing the video graph based on PE. We call those nodes instances of the path expression PE.

Definition 14. *[Instances] An object O is an* instance *of a path expression PE if one of the following holds:*

- *PE is O.*
- *PE is $PE' \rightarrow l$, and $\exists O'$ an instance of path expression PE' and a c-link L labeled l such that $O' =$ **from**(L) and $O =$ **to**(L).*
- *PE is $PE' \stackrel{*}{\rightarrow} l$, and $\exists O'$ an instance of path expression PE' and a c-link L labeled l such that $O =$ **to**(L) and **from**$(L) \in \bigtriangledown(O')$.*
- *PE is $PE' \rightarrow l(O)$, and $\exists O'$ an instance of path expression PE' and a c-link L labeled l such that $O' =$ **from**(L) and $O =$ **to**(L).*
- *PE is $PE' \stackrel{*}{\rightarrow} l(O)$, and $\exists O'$ an instance of path expression PE' and a c-link L labeled l such that $O =$ **to**(L) and $O =$ **from**$(L) \in \bigtriangledown(O')$.*

Since the query outputs can be of two types, an i-expression or a set of node identifiers from which semantic contents are withdrawn, we consider the following cases: (1) **SEM** filter: The answer to a query is a set of objects (node identifiers),

each being an instance of the path expression $PE(O_1, O_2, .., O_n)$ where O_i's $\in \bigtriangledown(SN)$ and O_i 's make the condition $SC(O_1, O_2, .., O_n)$ true. (2) **TIME** filter: The answer to a query is an i-expression which is computed by applying "&" operation on the **TIME** values of all the instances of the path expression $PE(O_1, O_2, .., O_n)$ where O_i's $\in \bigtriangledown(SN)$ and O_i's make the condition $SC(O_1, O_2, .., O_n)$ true.

3.3 Examples

Now we present some examples of queries. The video database is the video graph illustrated in Figure 2 where there is only one video. Suppose the identifier of that video is V.

Example 1. The scence going on in a police station where John does not appear.

$$\begin{aligned}(TIME) \Leftarrow o., V, o. \rightarrow Where = \text{“}PoliceStation\text{”} \wedge (\forall o'(o' \in \bigtriangledown(o) \\ \wedge o'. \rightarrow Person \rightarrow Name = \text{“}John\text{”} \\ \Rightarrow \oplus(o', o) = ABCD \vee \oplus(o, o') = ABCD) \end{aligned} \quad (3)$$

Example 2. The information about the scene from time 15 to time 20 and from time 30 to time 32.

$$(SEM) \Leftarrow o., V, \otimes(o, [15, 20]\&[30, 32]) = AABB \quad (4)$$

Example 3. The information about the scene in which Mariah was trying to run away from a wolf.

$$\begin{aligned}(SEM) \Leftarrow o., V, \exists o'(o'. \rightarrow Person \rightarrow Name = \text{“}Mariah\text{”} \\ \wedge o'. \rightarrow Person \rightarrow DoWhat = \text{“}Run\text{”} \\ \wedge o'. \rightarrow Person \rightarrow WhatElse = \text{“}Wolf\text{”} \wedge \oplus(o', o) = ACDB) \end{aligned} \quad (5)$$

Example 4. How the scene containing John's singing and Mariah's running begins.

$$\begin{aligned}(SEM) \Leftarrow o. \rightarrow BeginWhere, V, \exists o_1(\exists o_2(o_1 \in \bigtriangledown(o) \wedge o_2 \in \bigtriangledown(o) \\ \wedge o_1. \rightarrow Person \rightarrow Name = \text{“}John\text{”} \wedge o_1. \rightarrow Person \rightarrow \\ DoWhat = \text{“}Sing\text{”} \wedge o_2. \rightarrow Person \rightarrow Name = \text{“}Mariah\text{”} \\ \wedge o_2. \rightarrow Person \rightarrow DoWhat = \text{“}Run\text{”})) \end{aligned} \quad (6)$$

The path expression $PE(O_1, O_2, .., O_n)$ in the query can contain more than one node variable. This capability makes our language more expressive and more powerful. The SN (start node) component is not necessarily the root, which helps limit the graph traversal scope for the results. An example of such queries is below.

Example 5. The following query returns the video segments that are included in a "running" scene and where Mariah appears.

$$(TIME) \Leftarrow o. \stackrel{*}{\rightarrow} o', o_2, \exists o''(o''. \rightarrow Running = o) \\ \wedge o'. \rightarrow Person \rightarrow Name = \text{“}Mariah\text{”} \quad (7)$$

In this query, the PE expression contains two node variables o and o'. Note that, the start node is not the root, but node o_2 of the graph.

4 Implicit Information Inference

Having presented the video model and its related formal query language, one issue left is how to compute the implicit information from a VideoGraph database. This can be considered a preprocessing refinement phase. In our VideoGraph model, an internal node may or may not have a **TIME** link from it. If not, its temporal information can still be obtained by traversing the graph taking into account r-links to other nodes that have a temporal value. In other words, we are able to obtain implicit complete semantics from event descriptions, inter-event descriptions and incomplete information. In what follows, we introduce a simple way of how to do it. The algorithm converts an instance of the video data model to another called refined graph. The refined graph has direct temporal features associated with the internal nodes (that is, each node has a **TIME** link and a **TIME** value). Now we define what a refined graph is and shortly introduce a simple algorithm to compute the refined graph of a given video graph.

Given a single video, its video graph can be represented as a tuple $G = (V, e, f, g, h)$ where V is the set of nodes; e: $V \rightarrow$ {INTERNAL, LEAF}, a unary function returning the type of each node; f: $V \rightarrow$ **ATYPE** $\cup\ \Omega$ a unary function returning the value stored in each node; g: $V \times V \rightarrow$ **TYPE** $\cup$ VOID, returning the label of the link from one node to another, VOID if there is no c-link between them; h: $V \times V \rightarrow$ **REL** $\cup$ VOID, returning the temporal relationship between one node to another, VOID if there is no r-link between them.

Definition 15. *[Refined Graph] Given a video graph $G = (V, e, f, g, h)$ of a video. Its* refined graph *is also a VideoGraph $G_1 = (V_1, e_1, f_1, g_1, h_1)$ with the following properties.*

- $V \subseteq V_1$
- *card*(V_1) = *card*(V) + *card*(V_2) *where* $V_2 = \{v \in V \mid e(v) = INTERNAL \wedge \forall\ v_1 \in V: g(v, v_1) \neq$ **TIME**$\}$
- *For each $v \in V_2$, there exists only a node $v_1 \in V_1 - V$ such that $g_1(v, v_1) = TIME$. Conversely, for each $v_1 \in V_1 - V$, there is only a node v of V_2 such that the above condition holds.*
- $e_1(v) = e(v)$ *if* $v \in V$, *LEAF otherwise*
- *If* $v, v_1 \in V$, *then* $f_1(v) = f(v)$, $g_1(v, v_1) = g(v, v_1)$
- *For* $v, v_1 \in V_1$ *such that* g_1 *is not yet defined for,* $g_1(v, v_1) = VOID$

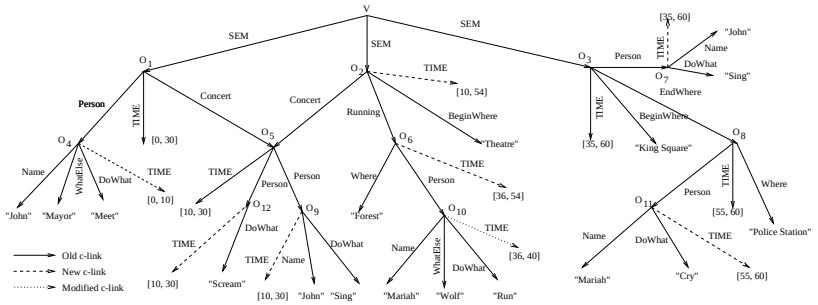

Fig. 3. A refined graph example

- $h_1\ (v,\ v_1\ ) = VOID\ \forall\ v,\ v_1 \in V_1$
- *If* $v_1,\ v_2 \in V$ *such that* $e(v_1) = e(v_2) = INTERNAL$, *and* $v_1',\ v_2' \in V_1$ *such that* $g_1(v_1,\ v_1') = g_1(v_2,\ v_2') =$ **TIME**, *then the relationship between* $f_1(v_1)$ *and* $f_1(v_2)$ *must not conflict with* $h(v_1,\ v_2)$.

The corresponding refined graph of the video graph in Figure 2 is in Figure 3. We note that node o_{10} now has a more meaningful **TIME** value, which tells that it associates with i-expression [36, 40], not like in the source video graph where o_{10}'s certain temporal information was unknown. Now comes an algorithm to determine the refined graph of a video graph.

Algorithm 1 *[Refinement Algorithm] Given a video graph* $G = (V,\ e,\ f,\ g,\ h)$ *for a video. The corresponding refined graph is built as the following steps.*

1. V_2 *is initialized to the set of all non-key objects in the source video graph.*
2. *For each node* $v \in V_2$, *add a new node storing* $[0,\infty]$ *and add a link labeled* **TIME** *from* v *to the new node.*
3. *Initialize* $UpdateCounter$ *and* $UpdateCounter_1$ *to* 0.
4. *For any two internal nodes* v *and* v' *that are connected by an r-link, adjust the value of their* **TIME** *node so as to satisfy* $h(v,\ v')$. *If a node is a key object with a complete* **TIME** *value, do not change the value. If one of the values is changed, increase* $UpdateCounter_1$ *and* $UpdateCounter$ *both by* 1.
5. *If* $UpdateCounter_1 \neq 0$, *go back to step 3.*
6. *For any two internal nodes* v *and* v' *such that the former is a sub-object of the latter, adjust the value of their* **TIME** *node so that the* **TIME** *values of* v *and* v' *are related by the relation ACDB (containment relationship). If a node is a key object with a complete* **TIME** *value, do not change the value. If one of the values is changed, increase* $UpdateCounter_1$ *and* $UpdateCounter$ *by* 1.
7. *If* $UpdateCounter_1 \neq 0$, *assign it to* 0 *and go back to step 6.*
8. *If* $UpdateCounter \neq 0$, *go back to step 3.*
9. *Remove all the r-links resulting in a new graph, which is the refined graph of* G.

In a refined graph, all the r-links have been removed. It has another property that every object (internal node) has a temporal descriptor which is captured by applying the refinement algorithm above on the source video graph. Thus whenever a node is visited for its temporal information, no further graph traversal is needed. The merit of the algorithm is that it helps reduce the overhead of query processing. For example, if we only have the source video graph (i.e., without applying the refinement algorithm) as in Figure 3 and if the user very often wants to watch the scene associated with object o_{10}, then the query system will have to compute the implicit temporal description of o_{10} many times back and forth.

In environments where most objects in the video database are accessed with high frequency, it is a good idea to build the refined graph just once, ahead of time, and store it in the database. Whenever the video graph is updated, its corresponding refined graph is recalculated. Subsequent query processing steps will be taken on the refined graph, resulting in more processing overhead being reduced. However, it is not always necessary to save it permanently, especially if we take into account the cost of storing an additional graph. Depending upon the context where the video database is, we have to consider the tradeoff between the efficiency of using the refined graph and the storage cost charged. From that standpoint, we can decide to compute it, whether or not on the fly, as the user questions the video.

5 Conclusions

We have introduced in this paper a new video data model called VideoGraph. This model can capture not only descriptions of individual events, but also their interevent relationships. To fully benefit from these two types of semantics, we have provided an algorithm to compute the implicit information from the initial metadata. Another contribution is the support for uncertainty. This arises because information extraction tools usually fail to produce the complete metadata set and reasoning on incomplete information often results in uncertain information. To address this, VideoGraph associates each event with an i-expression. For instance, a disjunctive ("|") form can be used to indicate that the video event must occur in one of the listed video segments. Our scheme hence is more intelligent and expressible than existing techniques.

To facilitate video retrieval, we have also presented in this paper a declarative query language based on path expressions. To the best of our knowledge, it is the first query language of this kind. Path expressions are easy to use yet powerful enough to allow the expression of fairly complex video queries.

This paper focuses on the formalism of the VideoGraph approach. For the time being, we are working toward an implementation of a networked video database system based on the model. In this effort, we are investigating various query processing techniques and different storage organizations suitable for storing video data. Another important direction is to extend VideoGraph framework to incorporate abstraction (such as classification, generalization and aggregation)

and presentation mechanisms in which we believe our graphical approach will offer better flexibility and possibilities.

References

1. S. Adali, K. S. Candan, S.-S. Chen, K. Erol, and V. S. Subrahmanian. Advanced video information system: Data structures and query processing. *ACM-Springer Multimedia Systems Journal*, 1996.
2. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM, Springer-Verlag*, 26(11), 1983.
3. E. Ardizzone and M. L. Cascia. Automatic video database indexing and retrieval. *Multimedia Tools and Applications*, 4(1), 1997.
4. S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong. Videoq: An automated content based video search system using visual cues. In *ACM Multimedia*, November 1997.
5. T.-S. Chua and L.-Q. Ruan. A video retrieval and sequencing system. *ACM Transactions on Information Systems*, 13(4), 1995.
6. C. Decleir and M.-S. Hacid. A database approach for modeling and querying video data. In *IEEE Data Engineering*, Australia, 1999.
7. N. Dimitrova, T. McGee, and H. Elenbaas. Video keyframe extraction and filtering: A keyframe is not a keyframe to everyone. In *Int'l Conf. on Information and Knowledge Management*, 1997.
8. A. K. Elmagarmid, H. Jiang, A. A. Helal, A. Joshi, and M. Ahmed. *Video Database Systems: Issues, Products, and Applications.* Kluwer Academic Publishers, 1997.
9. A. Hampapur, R. Jain, and T. Weymouth. Production model based digital video segmentation. *Journal of Multimedia Tools and Applications*, 1(1), 1995.
10. H. Jiang, D. Montesi, and K. Elmagarmid. Videotext database systems. In *IEEE Int'l Conf. on Multimedia Computing and Systems*, Ontario, Canada, June 1997.
11. J. Oh and K. A. Hua. An efficient and cost-effective technique for browsing, querying and indexing large video databases. In *ACM SIGMOD*, Dallas, TX, May 2000.
12. E. Oomoto and K. Tanaka. Ovid: Design and implementation of a video-object database system. *IEEE Trans. on Knowledge and Data Engineering*, 5, August 1993.
13. Y. Rui, S. Huang, and S. Mehrotra. Constructing table-of-content for videos. *ACM Springer-Verlag Multimedia Systems*, 7(5), 1999.
14. T. G. A. Smith and G. Davenport. The stratification system: A design environment for random access video. In *Proceedings of the 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, 1992.
15. D. Swanberg, C.-F. Shu, and R. Jain. Knowledge guided parsing in video databases. In *SPIE Conf. on Image and Video Processing*, volume 1908, San Jose, CA, February 1993.
16. D. A. Tran, K. A. Hua, and K. Vu. Semantic reasoning based video database systems. In *Proc. of 11th International Conference on Databases and Expert Systems Applications*, London, U.K, September 2000.
17. R. Weiss, A. Duda, and D. Gifford. Content-based access to algebraic video. In *IEEE Int'l Conf. on Multimedia Computing and Systems*, Boston, USA, 1994.
18. A. Yoshitaka, Y. Hosoda, M. Yoshimisu, M. Hirakawa, and T. Ichikawa. Violone: Video retrieval by motion example. *Visual Languages and Computing*, 7, 1996.