RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

A Database Approach for Modeling and Querying Video Data

Cyril Decleir Mohand-Saïd Hacid Jacques Kouloumdjian

LTCS-Report 99-03

This is an extended version of the article in: 15th International Conference on Data Engineering, Sydney, Australia, 1999.

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

Abstract

Indexing video data is essential for providing content based access. In this paper, we consider how database technology can offer an integrated framework for modeling and querying video data. As many concerns in video (e.g., modeling and querying) are also found in databases, databases provide an interesting angle to attack many of the problems. From a video applications perspective, database systems provide a nice basis for future video systems. More generally, database research will provide solutions to many video issues even if these are partial or fragmented. From a database perspective, video applications provide beautiful challenges. Next generation database systems will need to provide support for multimedia data (e.g., image, video, audio). These data types require new techniques for their management (i.e., storing, modeling, querying, etc.). Hence new solutions are significant.

This paper develops a data model and a rule-based query language for video content based indexing and retrieval. The data model is designed around the object and constraint paradigms. A video sequence is split into a set of fragments. Each fragment can be analyzed to extract the information (symbolic descriptions) of interest that can be put into a database. This database can then be searched to find information of interest. Two types of information are considered: (1) the entities (objects) of interest in the domain of a video sequence, (2) video frames which contain these entities. To represent these information, our data model allows facts as well as objects and constraints. We present a declarative, rule-based, constraint query language that can be used to infer relationships about information represented in the model. The language has a clear declarative and operational semantics. This work is a major revision and a consolidation of [12, 13].

Keywords: Content-Based Access of Video, Rule-Based Query Languages, Object-Oriented Modeling, Constraint Query Languages.

1 Introduction

With recent progress in compression technology, it is possible for computer to store huge amount of pictures, audio and even video. If such media are widely used in today's communication (e.g., in the form of home movies, education and training, scholarly research, and corporate enterprise solutions), efficient computer exploitation is still lacking. Many databases should be created to face the increasing development of advanced applications, such as video on demand, video/visual/multimedia databases, monitoring, virtual reality, internet video, interactive TV, video conferencing and video email, etc. Though only a partial list, these advanced applications need to integrate video data for complex manipulations.

Video analysis and content retrieval based on semantics require multi-disciplinary research effort in areas such as computer vision, image processing, data compression, databases, information systems, etc. (see [38, 15]). Therefore, video data management poses special challenges which call for new techniques allowing an easy development of applications. Facilities should be available for users to view video material in a non-sequential manner, to navigate through sequences, to build new sequences from others, etc. To facilitate retrieval, all useful semantic objects and their features appearing in the video must be appropriately indexed. The use of keywords or free text [20, 40] to describe the necessary semantic objects is not sufficient [11]. Additional techniques are needed. As stated in [11], the issues that need to be addressed are: (1) the *representation of video information in a form that facilitates retrieval and interaction*, (2) the organization of this information for efficient manipulation, and (3) the user-friendly presentation of the retrieved video sequences. Being able to derive an adequate content description from a video, however, does not guarantee a satisfactory retrieval effectiveness, it is only a necessary condition to this end. It is mandatory the video data model be powerful enough to

allow both the expression of sophisticated content representation and their proper usage upon querying a video database. For example, the time-dependent nature of video is of considerable importance in developing adequate data models and query languages.

Many features of database systems seem desirable in a video context: secondary storage management, persistence, transactions, concurrency control, recovery, versions, etc. In addition, a database support for video information will help sharing information among applications and make it available for analysis. The advantages (in general and for video in particular [34, 42, 25]) of database technology, such as object-oriented databases, are (1) the ability to represent complex data, and (2) more openness to the external world (e.g., Web, Java, CORBA, languages bindings) than traditional database systems. However, as existing database technology is not designed to manage digital video as first class media, new techniques are required for organizing, storing, manipulating, retrieving by content, and automatic processing and presentation of visual content. Although some tools for video exploitation are available, their use often amounts to displaying video in sequence, and most modeling methods have been developed for specific needs. In many cases, query languages concentrate on extraction capabilities. Queries over video data are described only by means of a set of pre-defined, ad hoc operators, often incorporated to SQL, and are not investigated in *theoretical framework*. One can argue that logic-based database query languages appropriately designed to support video specific features should form a sound basis for query languages.

From database point of view, video data presents an interesting challenge. Future database systems must cover the range of tasks associated with the management of video content including feature extraction, indexing, querying, and developing representation schemes and operators. For example, the data model should be expressive enough to capture several characteristics inherent to video data, such as movements, shapes, variations, events, etc. The query language should allow some kind of reasoning, to allow, for example, virtual editing [29], and should be able to perform exact as well as partial or fuzzy matching (see [4]).

Despite the consensus of the central role video databases will play in the future, there is little research work on finding *semantic foundations for representing and querying video information*. This paper is a contribution in this direction. The framework presented here integrates formalisms developed in constraint, object and sequence databases. The paper builds on the works of [32, 34, 23, 1, 30, 37, 8, 31, 18] to propose a hybrid data model for video data and a declarative, rule-based, constraint query language, that has a clear declarative and operational semantics. We make the following contributions:

1. We develop a simple video data model on the basis of relation, object and constraint paradigms. Objects of interest and relationships among objects can be attached to a generalized interval[1] either through attribute/value pairs or relations.

2. We propose a declarative, rule-based, constraint query language that can be used to infer relationships from information represented in the model, and to intentionally specify relationships among objects. It allows a high level specification of video data manipulations.

The model and the query language use the point-based approach to represent periods of time associated with generalized intervals. First-order queries can then be conveniently asked in a much more declarative and natural way [39]. There has been some previous research on the power of constraints for the implicit specification of temporal data [10].

[1] A generalized interval is a set of pairwise non overlapping fragments in a video sequence.

The model and the query language will be used as a core of a video document archive prototype by both a television channel and a national audio-visual institute. *To the best of our knowledge, this is the first proposal of a formal rule-based query language for querying video data.*

Paper outline: This paper is organized as follows. Section 2 discusses related work. In Section 3, we informally introduce the indexing of video sequences. Section 4 presents some useful definitions. Section 5 formally introduces the video data model. Section 6 describes the underlying query language. Section 7 draws conclusions.

2 Related Work

With the advent of multimedia computers (PCs and workstations), the world-wide web, and standard and powerful compression techniques[2], it becomes possible to digitize and store common human media, such as pictures, sounds, and video streams worldwide. Nevertheless, storing is the minimal function we are to expect from a computer, its power should also be aimed at *content indexing* and *retrieval*. Two main approaches have been experienced: fully automated content indexing approach, and the approach based on human-machine interaction. Some fully automated research systems have been developed, among others, *VIOLONE* [41] and *JACOB* [28]. However, because of the weakness of content analysis algorithms, they focus on a very specific exploitation. On the other hand, much more aided video content indexing systems have been designed, among others, *OVID* [34], *AVIS* [1], or *VideoStar* [23].

Some fascinating database issues in the context of video data and multimedia in general (see, among others, [14, 21, 6]) are note presented here.

In the context of image and video data, queries can be formulated using several techniques, which fall broadly into two categories: *textual* and *visual*. Several systems have been developed to retrieve visual data based on color, shape, size, texture, image segments, keyword, relational operators, objects, and bibliographic data (see, among others, [7, 9, 22, 26, 33]). In this paper, we focus on textual languages.

The work presented here is closest to and complements the ones in [32, 1, 34, 23].

Meghini [32] proposed a retrieval model for images based on first-order logical language which spans along four main dimensions: visual, spatial, mapping and content. Queries on images can address anyone of these dimensions or any combination of them. In the proposed model, objects cannot be characterized by attributes. Every entity is described by means of relations (predicates). For example, objects' shapes cannot be stated in a declarative and equational manner, as it is the case in our model.

Oomoto and Tanaka [34] proposed a schema-less video-object data model. They focus on the capabilities of object-oriented database features (their extension) for supporting schema evolution and to provide a mechanism for sharing some descriptive data. A video frame sequence is modeled as an object with attributes and attribute values to describe its contents. A semantically meaningful scene is a sequence of (not always continuous) video frames. An interval is described by a pair of a starting frame and an ending frame. It denotes a continuous sequence of video frames. They introduced the notion of inheritance based on the interval inclusion relationship. By means of this notion different video-objects may share descriptional data. Several

[2]Such as *MPEG-I* [16] [17] and its successors.

operations, such as interval projection, merge and overlap are defined to compose new objects from other objects. They provide the user with the SQL-based query language VideoSQL for retrieving video-objects. This model does not allow the description and the definition of relationships among objects within a video-object. The content of a video-object is described in terms of attribute-values of this video-object. Semantic objects are considered as values of attributes of video-objects.

Adali et al. [1] have developed a formal video data model, and they exploit spatial data structures for storing such data. They emphasized some kinds of human-level information in video: objects of interest, activities, events and roles. A given video is divided into a sequence of frames which constitute logical divisions of the video. Associated with each object/event is a set of frame-sequences. Each frame sequence can be viewed as a frame segment. Events are characterized by a set of attributes describing their context. For example, the event give party may be characterized by the multi-valued attribute *host* whose values are *Philip* and *Brandon* and the attribute guest whose value is *Rupert*. In this framework, objects other than events have no complex structure. The only relationships among objects are those given implicitly through the description of events. They also developed a simple SQL-like video query language which can be used to retrieve videos of interest and extracts from them the relevant segments of the video that satisfy the specified query conditions.

These two proposals provide an interval-based approach to represent the periods of time associated with frames of interest in a video sequence.

Hjelsvold and Midtstraum [23] proposed a generic video data model. Their proposal combines ideas from the stratification [2] and the segmentation [11] approaches. The objective is to develop a framework where structuring, annotations, sharing and reuse of video data become possible. Their model is built upon an enhanced-ER model. A simple SQL-like video query language with temporal interval operators (e.g., *equals*, *before*, etc.) is provided. This work concentrates on the structural part of a video in order to support video browsing. Thematic indexing is based on annotations, which give a textual description of the content of frame sequences. In contrast, we allow a more elaborated and structured description of the content of frame sequences.

With regard to the modeling, we extended these works by allowing the description of the contents of video sequences by means of first class citizen objects of the data model and by relating them to each others either through attributes or through explicit relation names, leading to more expressive relationships to link objects. Hence, video frames (which we call generalized intervals) as well as semantic objects (objects of interest in a generalized interval) are modeled and manipulated at the same level. Special queries, like spatial and temporal ones, can be expressed in a much more declarative manner. Generalized intervals, as well as semantic objects and relationships among these elements can be described, making a video sequence appropriate for different applications.

3 Modeling Video Data: An Overview

In this section, we informally introduce our video indexing model. After discussing fundamental aspects regarding video data, we present our time-managing approach through *Generalized Interval* objects and we discuss briefly its relation to conventional approaches.

Basically, video indexing means defining easy-to-retrieve meaningful[3] information which should carry a part of the content of the video sequence. Even if video data consists of sequences of images, and thus they share all the attributes of image data such as color, shape, objects, positions of objects, relative layouts and texture, video have additional temporal and relational attributes and, in most cases, video data are hierarchical in structure. As a consequence, a video indexing model should provide facilities to efficiently capture these additional attributes. Because of temporal nature of video, informations abstracting the video content are only true at a given time. Therefore, a temporal management of video information is required.

Historically, the *segmentation* approach was the first video indexing scheme that has been proposed. With this approach, a sequence is split into independent and contiguous time segments which are annotated individually. Figure 1 shows a basic segmentation of broadcast news. It can be easily seen that the timeline of the document is partitioned into individual segments, each of them being associated with a handwritten description.

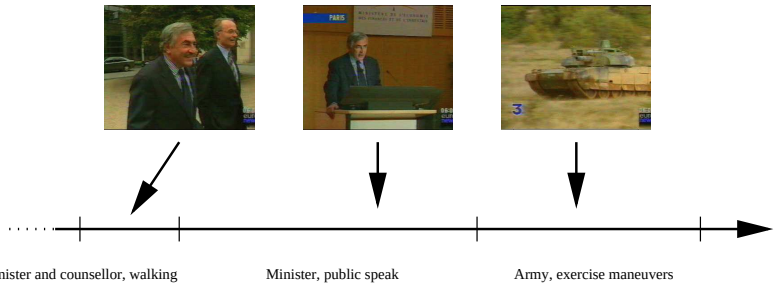

Figure 1: Indexing by Segmentation

Segmentation is still used when light and quick description of video documents are needed, for example in applications like news broadcast archives.

This approach was criticized by Aguierre-Smith and Davenport [2] mainly because its strict temporal partitioning results in rough descriptions of video documents. As a consequence, they introduced a method, called *stratification*, which allows to annotate facts individually. In this approach, each element of interest is being associated with a single temporal descriptor, which is an interval called *strata*, as shown in Figure 2. One can see that stratification allows overlapping of descriptions, therefore allowing the user to specify several levels of descriptions. For example, Figure 2 shows several levels of decomposition of the document "Broadcast News". Basically, the idea in using stratification is to allow any interesting fact to be highlighted, regardless of other descriptions.

We extend the stratification approach by defining what we call *Generalized intervals*. In contrast to the stratification approach, where a time segment is associated with a description, we allow a set of time segments to be associated with a description. Therefore, a generalized interval is defined as a set of non-overlapping intervals providing time boundaries to a description. This allows to handle with a single object all occurrences of an entity in a video document. As an example, suppose we want to index a tv-news broadcast, and annotate each period of time an object of interest appears on the screen. Suppose we have three objects of interest, say *Reporter*, *Reporter #2* and *Minister*. In this case, we simply associate each of these objects with the corresponding generalized interval as shown in Figure 3: three objects of interest are defined,

[3]But application dependent.

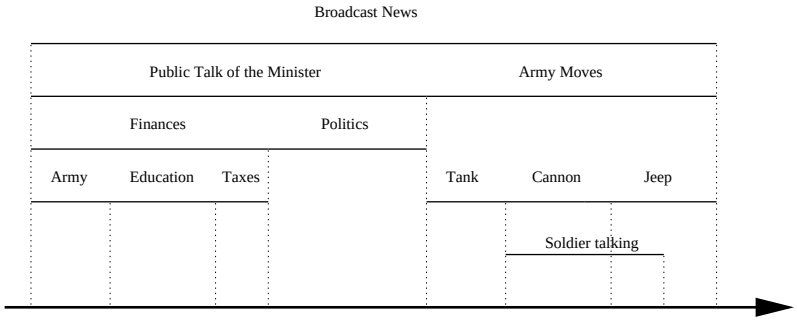

Figure 2: Indexing by Stratification

and each of them has a generalized interval that traces its presence on the screen. Therefore, this allows, with a single identifier, for instance "Reporter", to refer to all occurrences of "Reporter" in the document.

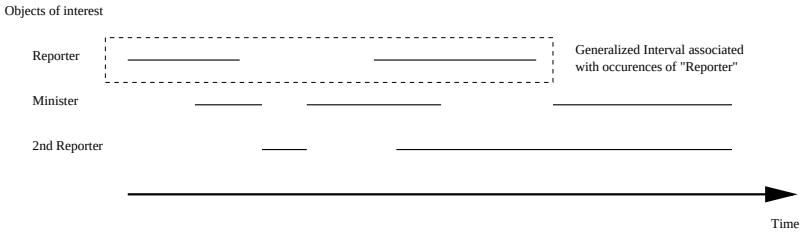

Figure 3: Generalized Interval Indexing

4 Basic Definitions

This section provides the preliminary concepts that will be used to design the video data model and the underlying rule-based, constraint query language.

Definition 1 (Concrete Domains) *A concrete domain* $\mathcal{D} = (dom(\mathcal{D}), pred(\mathcal{D}))$ *consists of:*

- *the domain* $dom(\mathcal{D})$,
- *a set of predicate symbols* $pred(\mathcal{D})$, *where each predicate symbol* $P \in pred(\mathcal{D})$ *is associated with an arity* n *and an* n*-ary relation* $P^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$,

An example of a concrete domain is the set of (nonnegative) integers with comparisons $(=, <, \leq, \geq, >)$.

In the following, we assume *entailment* of conjunctions or disjunctions over $pred(\mathcal{D})$ is decidable.

Definition 2 (Dense Linear Order Inequality Constraints) *Dense order inequality constraints are all formulas of the form* $x\theta y$ *and* $x\theta c$, *where* x, y *are variables,* c *is a constant, and* θ *is one of* $=, <, \leq$ *(or their negation* $\neq, \geq, >$*). We assume that these constants are interpreted over a countably infinite set* $\mathcal{D}$ *with a binary relation which is a dense order. Constants,* $=$,

$<$, and $\leq$ are interpreted respectively as elements, equality, the dense order, and the irreflexive dense order of the concrete domain $\mathcal{D}$.

Complex constraints are built from primitive (atomic) constraints by using logical connectives. We use the special symbol, $\Rightarrow$, to denote the entailment between constraints, that is, if c_1 and c_2 are two constraints, we write $c_1 \Rightarrow c_2$ for c_1 entails c_2. $c_1 \Rightarrow c_2$ is *satisfiable* if and only if the constraint $c_1 \wedge \neg c_2$ is *unsatisfiable.*

Techniques for checking satisfiability and entailment for order constraints over various domains have been studied. Regarding expressive power and complexity of linear constraint query languages, see [19].

Definition 3 (Set-Order Constraints) *Let $\mathcal{D}$ be a domain. A set-order constraint is one of the following types:*

$$c \in \widetilde{X}, \widetilde{X} \subseteq s, s \subseteq \widetilde{X}, \widetilde{X} \subseteq \widetilde{Y}$$

where c is a constant of type $\mathcal{D}$, s is a set of constants of type $\mathcal{D}$, and $\widetilde{X}$, $\widetilde{Y}$ denote set variables that range over finite sets of elements of type $\mathcal{D}$.

Our set-order constraints are a restricted form of set constraints [5], involving $\in$, $\subseteq$, and $\supseteq$, but no set functions such as $\cup$ and $\cap$.

Note that the constraint $c \in \widetilde{X}$ is a derived form since it can be rewritten as $\{c\} \subseteq \widetilde{X}$.

Satisfaction and entailment of conjunctions of set-order constraints can be solved in polynomial-time using a quantifier elimination algorithm given in [37].

This class of constraints play an important role in declaratively constraining query answers.

Definition 4 (Time Intervals) *An interval i is considered as an ordered pair of real numbers (x_1, x_2), $x_1 \leq x_2$. This definition refers to the predicate $\leq$ of the concrete domain $\mathbb{R}$. If t is a time variable, then an interval (x_1, x_2) can be represented by the conjunction of the two primitive dense linear order inequality constraints $x_1 \leq t$ and $t \leq x_2$.*

Definition 5 (Generalized Time Intervals) *A generalized time interval, or simply a generalized interval, is a set of pairwise non overlapping intervals. Formally, a generalized time interval can be represented as a disjunction of time intervals.*

5 Basic Video Formalism

5.1 Mathematical Structure

The central notion in our video data model is the generalized interval. By deciding to split a video sequence into a set of generalized intervals, each interval can be analyzed to extract the information of interest, and then be indexed on the basis of its content. We assume that the information describing the content of an interval as well as the length of the interval are user specified.

One of the basis for indexing video material is to establish a link between an information and its temporal location in a video sequence. Conventional approaches such as *segmentation* and *stratification* tackled this problem. These approaches assign a continuous temporal interval of a video to an information. Our approach extends this by relating an information to a set of temporal intervals, called generalized interval. This is because a semantically meaningful scene is a sequence of non continuous video intervals.

Because of the complexity of video analysis, we have to handle two main sources of information :

- Machine derived indices: such as shot-change detection or color histograms, basically raw features;
- Application specific desired video indices: higher level semantic information, which is the most needed data. Here, useful semantic objects and their attributes, referred to as semantic units, appearing in the video are indexed. Because of time-dependent nature of video data, these semantic objects are indexed with respect to time.

The video data model allows facts as well as objects. We define a video sequence as a 7-tuple:

$$V = (I, \mathcal{O}, \mho, \mathcal{R}, \Phi, \lambda_1, \lambda_2)$$

- I : a set of generalized intervals. The video sequence is split into a set of arbitrary generalized intervals. Note that generalized intervals overlapping is possible. Each generalized interval is seen as an abstract object with an *identifier*
- $\mathcal{O}$: a set of entities (objects) describing information within a video sequence. According to application needs, only information which is of interest for a given exploitation is considered. These objects are determined by analyzing each generalized interval.
- $\mho$: a set of atomic values. These values are drawn from concrete domains, i.e., integers, strings, etc.
- $\mathcal{R}$: a set of relations on $\mathcal{O}^* \times I$. These relations relate objects within a generalized interval.
- Φ : a set of constraints describing time intervals which define generalized intervals.
- $\lambda_1 : I \longrightarrow 2^{\mathcal{O}}$ maps each generalized interval to a subset of $\mathcal{O}$.
- $\lambda_2 : I \longrightarrow \Phi$ maps each generalized interval to a (complex) dense linear order constraint describing the periods of time associated with the generalized interval.

$\mathcal{O}$, I, and $\mho$ are pairwise disjoint.

5.2 Video Data Model

- *Objects and object identity* Objects are entities of interest in a video sequence. In our model, we refer to objects via their logical object identities, which are nothing but syntactic terms in the query language. Any logical oid uniquely identifies an object. In this paper, we will be using the word "object identity" (or even "object") to refer to ids at logical level. We have essentially two types of objects: (1) generalized interval objects, which are abstract objects resulting from splitting a given video sequence into a set of smaller sequences; (2) semantic objects which are entities of interest in a given video sequence.
- *Attributes* Objects are described via attributes. If an attribute is defined for a given object, then it also has a value for that object.
- *Relations* It has been argued many times that objects do not always model real world in the most natural way, and there are situations when the use of relations combined with objects leads to more natural representation. Although relations can be encoded as objects, this is not the most natural way of handling relations and so we prefer to have relations as first-class language constructs.

We assume the existence of the following countably infinite and pairwise disjoint sets of atomic elements:

- relation names $\mathcal{R} = \{R_1, R_2, \ldots\}$;
- attributes $\mathcal{A} = \{A_1, A_2, \ldots\}$;
- (atomic) constants $\mathcal{D} = \{d_1, d_2, \ldots\}$;
- object identities or oid's $\mathcal{ID} = \{id_1, id_2, \ldots\}$. In the following, we distinguish between object identities for entities and object identities for generalized intervals.

Furthermore, in order to be able to associate a time interval to a generalized interval object, we allow a restricted form of dense linear order inequality constraints to be values of attributes. We define the set $\tilde{C}$ whose elements are:

- Primitive (atomic) constraints of the form $t\theta c$ where t is a variable, c is a constant, and θ is one of $<, =, >$;
- conjunctions, and disjunctions of primitive constraints.

Definition 6 (value) *The set of values is the smallest set containing* $\mathcal{D} \cup \mathcal{ID} \cup \tilde{C}$ *and such that, if* $v_1, \ldots, v_n$ $(n \geq 1)$ *are values, then so is* $\{v_1, \ldots, v_n\}$.

Definition 7 (Video Object) *A video object (denoted* v-object*) consists of a pair* (oid, v) *where:*

- *oid is an object identifier which is an element of* $\mathcal{ID}$*;*
- v *is an* m*-tuple* $[A_1 : v_1, \ldots, A_m : v_m]$*, where* A_i $(i \in [1, m])$ *are distinct attribute names in* $\mathcal{A}$ *and* v_i $(i \in [1, m])$ *are values.*

If $o = (oid, v)$ with $v = [A_1 : v_1, \ldots, A_n : v_n]$, then $attr(o)$ denotes the set of all attributes in v (i.e. $\{A_1, \ldots, A_n\}$), and $value(o)$ denotes the value v, that is, $v = value(o)$. The value v_i is denoted by $o.A_i$.

Example Let us see how the example given in [1] can be modeled in our framework. First let us recall the example (extracted from [1]): It concerns the movie "*The Rope*" by Alfred Hitchcock. This movie has a 80 minutes duration. In the movie, two friends, Philip and Brandon decide to commit the perfect crime. They want to prove they are of the privileged group of people who are allowed to kill just for sake of killing and not receiving any punishment for it. Hence, they kill their friend David and hide him inside a chest in the living room. To sign their masterpiece, they give a party where they invite friends of David (David's girlfriend Janet, Janet's old boyfriend Kenneth), and his parents (David's father Mr. Kentley, David's aunt Mrs. Atwater). These individuals would talk about David, not suspecting that David's body is in the same room they are standing. In addition to these people, they invite, as a challenge, their old mentor Rupert Cadell who is known to be very intelligent and suspicious. Rupert will prove worthy of his reputation and he will immediately understand the extraordinary circumstances. As the movie progresses, Rupert will keep asking questions and gather clues to find out what is wrong.

Let us consider, for example, two generalized intervals:

1. The first (gi_1) corresponds to the period of time in the sequence where the crime is committed. This interval contains four objects of interest: Philip, Brandon, David, and the Chest. The three objects Philip, Brandon, and David have an attribute, called role. Role-filler for Philip and Brandon is "murderer", and role-filler for David is "victim".

2. The second (gi_2) corresponds to the period of time in the sequence where the party is given. This interval contains objects Philip, Brandon, David, Janet, Kenneth, Kentley, Atwater, Rupert Cadell, and Chest.

The following is a a simple database extract indexing, in part, by content the two generalized intervals gi_1 and gi_2.

$gi_1 = (id_1, [entities : \{o_1, o_2, o_3, o_4\}, duration : (t > a_1 \wedge t < b_1), subject : "murder", victim : o_1, murderer : \{o_2, o_3\}])$
$gi_2 = (id_2, [entities : \{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8, o_9\}, duration : (t > a_2 \wedge t < b_2), subject :$ "Giving a party", host : $\{o_2, o_3\}$, guest : $\{o_5, o_6, o_7, o_8, o_9\}])$

$o_1 = (id_3, [name : "David", role : "Victim"])$
$o_2 = (id_4, [name : "Philip", realname : "$Farley Granger", role : "Murderer"])
$o_3 = (id_5, [name : "Brandon", realname : "$John Dall", role : "Murderer"])
$o_4 = (id_6, [identification : "Chest"])$
$o_5 = (id_7, [name : "Janet", realname : "$Joan Chandler"])
$o_6 = (id_8, [name : "Kenneth", realname : "$Douglas Dick"])
$o_7 = (id_9, [name : "Mr.Kentley", realname : "$Cedric Hardwicke"])
$o_8 = (id_{10}, [name : "Mrs.Atwater", realname : "$Constance Collier"])
$o_9 = (id_{11}, [name : "$Rupert Cadell", realname : "James Stewart"])

$in(o_1, o_4, gi_1)$
$in(o_1, o_4, gi_2)$
...

□

The first statement says that the generalized interval gi_1 has a duration given by the interval $[a_1, b_1]$. The entities of interest in this fragment of sequence are o_1, o_2, o_3, o_4. It also says that this fragment of a sequence deals with the murder (the value of the attribute subject), where the object o_1 (David) is the victim, the objects o_2 (Philip) and o_3 (Brandon) are the murderers.

The last two statements are facts that define a relationship between the objects o_1 (David) and o_4 (Chest) within the generalized intervals gi_1 and gi_2.

Note that in the first two statements, t is a temporal variable, and a_1, a_2, b_1, and b_2 are integers such that $a_1 < b_1 < a_2 < b_2$. A generalized interval does not necessarily correspond to a single continuous sequence of video frames. This is because a meaningful scene does not always correspond to a single continuous sequence of frames. In this case, the value describing the period of time associated with a generalized interval will be a disjunction of atomic constraints.

To simplify the model, the value of the attribute *duration* in both gi_1 and gi_2 is specified by means of a constraint. Another approach [8] treats constraints as first-class objects, organized in classes. In this case, constraints can have attributes and methods that attach additional information to them. Clearly, we need not such features in our video data model.

6 Rule-Based, Constraint Query Language

In this section, we present the declarative, rule-based query language that can be used to reason with facts and objects in our video data model. The language consists of two constraint languages[4] on top of which relations can be defined by means of definite clauses.

This language has a model-theoretic and fix-point semantics based on the notion of *extended active domain* of a database. The extended domain contains all generalized interval objects and their concatenations. The language has an interpreted function symbol for building new generalized intervals from others (by concatenating them). A constructive term has the form $I_1 \otimes I_2$ and is interpreted as the concatenation of the two generalized intervals I_1 and I_2.

The extended active domain is not fixed during query evaluation. Instead, whenever a new generalized interval object is created (by the concatenation operator, $\otimes$), the new object and the ones resulting from its concatenation with already existing ones are added to the extended active domain.

6.1 Syntax

To manipulate generalized intervals, our language has an interpreted function symbol for constructing[5] complex term. Intuitively, if I_1 and I_2 are generalized intervals, then $I_1 \otimes I_2$ denotes the concatenation of I_1 and I_2.

The language of terms uses three countable, disjoint sets:

1. A set $\mathbb{D}$ of constant symbols. This set is the union of three disjoint sets:
   - $\mathbb{D}_1$: a set of atomic values,
   - $\mathbb{D}_2$: a set of entities, also called object entities,
   - $\mathbb{D}_3$: a set of generalized interval objects.
2. A set $\mathcal{V}$ of variables called object and value variables, and denoted by $X, Y, \ldots$;
3. A set $\tilde{V}$ of variables called generalized interval variables, and denoted by $S, T, \ldots$;

If I_1 and I_2 denote generalized interval objects, generalized interval variables, or constructive interval terms, then $I_1 \otimes I_2$ is a *constructive* interval term.

In the following, the concatenation operator is supposed to be defined on $\mathbb{D}_3$, that is $\forall e_1, e_2 \in \mathbb{D}_3$, $e_1 \otimes e_2 \in \mathbb{D}_3$. The structure of the resulting element $e = e_1 \otimes e_2$ is defined from the structure of e_1 and e_2 as follows:
Let $e_1 = (id_1, v_1)$ and $e_2 = (id_2, v_2)$. Then $e = (id, v)$, is such that:

- $id = f(id_1, id_2)$. Here we follow the idea of [27] that the object id of the object generated from e_1 and e_2 should be a function of id_1 and id_2.
- $attr(e) = attr(e_1) \cup attr(e_2)$.
- $\forall A_i \in attr(e)$, $e.A_i = e_1.A_i \cup e_2.A_i$.

Note that $I_1 \otimes I_1 \equiv I_1$. This means that if I is obtained from the concatenation of I_1 and I_2, then the result of the concatenation of I with I_1 or I_2 is I. This leads to the termination of the execution of constructive rules (see below the definition of constructive rule).

[4] For a formal definition of a constraint language, see [24].
[5] For concatenating generalized intervals.

Definition 8 (Predicate symbol) *We define the following predicate symbols:*

- *each* $\mathsf{P} \in \mathcal{R}$ *with arity* n *is associated with a predicate symbol* P *of arity* n,
- *a special unary predicate symbol* Interval. *It can be seen as the class of all generalized interval objects.*
- *a special unary predicate symbol* Object. *It can be seen as the class of all objects other than generalized interval objects.*

Definition 9 (Atom) *If* P *is an* n*-ary predicate symbol and* $t_1, \ldots, t_n$ *are terms, then* $\mathsf{P}(\mathrm{t}_1, \ldots, \mathrm{t}_n)$ *is an atom. If* O *and* O' *denote objects or object variables,* Att *and* Att' *are attribute names, and* c *is a constant value, then* $O.Att\theta c$ *and* $O.Att\,\theta\,O'.Att'$ *where* θ *is one of* $=, <, \leq$ *(or their negation* $\neq, \geq, >$*) are called inequality atoms.*

Definition 10 (Rule) *A* rule *in our language has the form:*

$$r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$$

where H *is an atom,* $n, m \geq 0$*,* $L_1, \ldots, L_n$ *are (positive) literals, and* $c_1, \ldots, c_m$ *are constraints.*

Optionally, a rule can be named as above, using the prefix "r : ", where r is a constant symbol. We refer to A as the head of the rule and refer to $L_1, \ldots, L_n, c_1, \ldots, c_m$ as the body of the rule.

Note that we impose the restriction that constructive terms appear only in the head of a rule, and not in the body. A rule that contains a constructive term in its head is called a constructive rule.

Recall that we are interested in using order constraints, that is arithmetic constraints involving $<, >$, but no arithmetic functions such as $+, -, *$, and set-order constraints, a restricted form of set constraints involving $\in, \subseteq$, and $\supseteq$, but no set functions such as $\cup$ and $\cap$.

Definition 11 (Range-restricted Rule) *A rule* r *is said to be range-restricted if every variable in the rule occurs in a body literal. Thus, every variable occurring in the head occurs in a body literal.*

Definition 12 (Program) *A program is a collection of range-restricted rules.*

Definition 13 (Query) *A query is of the form:*

$$Q :?q(\bar{s})$$

where q *is referred to as the query predicate, and* $\bar{s}$ *is a tuple of constants and variables.*

Example Let us give some simple examples of queries. In the following, uppercase letters stand for variables and lowercase letters stand for constants.

The query "list the objects appearing in the domain of a given sequence g" can be expressed by the following rule:

$$q(O) \leftarrow Interval(g), Object(O), O \in g.entities$$

In this example, g is a constant and O is the output variable. Here, we suppose that for a given generalized interval, the set-valued attribute "entities" gives the set of semantic objects of interest in that generalized interval. This query involves an atomic (primitive) constraint. To compute the answer set to the query, we need to check the satisfiability of the constraint $O \in g.entities$ after O being instantiated.

The query "list all generalized Intervals where the object o appears" can be expressed as:

$$q(G) \leftarrow Interval(G), Object(o), o \in G.entities$$

The query "does the object o appear in the domain of a given temporal frame $[a, b]$" can be expressed as:

$$q(o) \leftarrow Interval(G), Object(o), o \in G.entities, G.duration \Rightarrow (t > a \wedge t < b)$$

Where t is a temporal variable. This query involves one primitive constraint $o \in G.entities$, and a complex arithmetic constraint $G.duration \Rightarrow (t > a \wedge t < b)$. To compute the answer set to the query, we need to check satisfiability of these two constraints.

The query "list all generalized intervals where the objects o_1 and o_2 appear together" can be expressed as:

$$q(G) \leftarrow Interval(G), Object(o_1), Object(o_2), o_1 \in G.entities, o_2 \in G.entities$$

or equivalently by:

$$q(G) \leftarrow Interval(G), Object(o_1), Object(o_2), \{o_1, o_2\} \subseteq G.entities$$

The query "list all pairs of objects, together with their corresponding generalized interval, such that the two objects are in the relation "*Rel*" within the generalized interval", can be expressed as:

$$q(O_1, O_2, G) \leftarrow Interval(G), Object(O_1), Object(O_2), O_1 \in G.entities, O_2 \in G.entities, Rel(O_1, O_2, G)$$

The query "find the generalized intervals containing an object O whose value for the attribute A is *val*" can be expressed as:

$$q(G) \leftarrow Interval(G), Object(O), O \in G.entities, O.A = val$$

□

6.2 Inferring new relationships

Rules can be used to infer (specify) new relationships, as facts, between existing objects.

Example Suppose we want to define the relation *contains*, which holds for two generalized interval objects G_1 and G_2 if the time interval associated with G_1 overlaps the time interval associated with G_2. This can be expressed as follows:

$$contains(G_1, G_2) \leftarrow$$
$$Interval(G_1), Interval(G_2), G_2.duration \Rightarrow G_1.duration$$

G_1 and G_2 are in the relation *contains* if the constraint ($duration\text{-}filler$) associated with G_2 entails the one associated with G_1.

If we want to define the relation *same-object-in* of all pairs of generalized intervals with their common objects, we write the following rule:

$$same - object - in(G_1, G_2, O) \leftarrow$$
$$Interval(G_1), Interval(G_2), Object(O), O \in G_1.entities, O \in G_2.entities$$

The following rule constructs concatenations of generalized intervals that have some objects, say o_1 and o_2 in common.

$$concatenate - Gintervals(G_1 \otimes G_2) \leftarrow$$
$$Interval(G_1), Interval(G_2), Object(o_1), Anyobject(o_2),$$
$$\{o_1, o_2\} \subseteq G_1.entities, \{o_1, o_2\} \subseteq G_2.entities$$

□

6.3 Semantics

Our language has a declarative model-theoretic and a fix-point semantics.

6.3.1 Model-theoretic Semantics

Recall that $\mathcal{V}$ denotes a set of variables called object and value variables, and $\tilde{V}$ denotes a set of variables called generalized interval variables. Let $\mathbb{V} = \mathcal{V} \cup \tilde{V}$.

Let *var* be a countable function that assigns to each syntactical expression a subset of $\mathbb{V}$ corresponding to the set of variables occurring in the expression. If $E_1, \ldots, E_n$ are syntactical expressions, then $var(E_1, \ldots, E_n)$ is an abbreviation for $var(E_1) \cup \ldots \cup var(E_n)$.

A ground atom A is an atom for which $var(A) = \emptyset$. A ground rule is a rule r for which $var(r) = \emptyset$.

Definition 14 (interpretation) *Given a program P, an interpretation $\mathcal{I}$ of P consists of:*

- *A domain $\mathbb{D}$;*
- *A mapping from each constant symbol in P to an element of domain $\mathbb{D}$;*
- *A mapping from each n-ary predicate symbol in P to a relation in $\mathbb{D}^n$.*

Definition 15 (Valuation) *A valuation ν is a total function from $\mathbb{V}$ to the set of elements $\mathbb{D}$. This is extended to be identity on $\mathbb{D}$ and then extended to map free tuples to tuples in the natural fashion. ν is extended to constraints in a straightforward way. In additions, if I_1 and I_2 are generalized interval terms, then $\nu(I_1 \otimes I_2) = \nu(I_1) \otimes \nu(I_2)$.*

Definition 16 (Rule Satisfaction) *Let r be a rule of the form:*

$$r : A \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$$

$L_1, \ldots, L_n$ are (positive) atoms, and $c_1, \ldots, c_m$ are constraints. Let $\mathcal{I}$ be an interpretation, and ν be a valuation that maps all variables of r to elements of $\mathbb{D}$. The rule r is said to be true (or satisfied) in interpretation $\mathcal{I}$ for valuation ν if $\nu[A]$ is present in $\mathcal{I}$ whenever:

- *Each $\nu(c_i)$, $i \in [1, m]$ is satisfiable, and*
- *each $\nu[L_i]$, $i \in [1, n]$ is present in $\mathcal{I}$.*

Definition 17 (Model of a Program) *Consider a program P. An interpretation $\mathcal{I}$ is said to be a model of P if each of the rules of P is satisfied, for every valuation ν that maps variables of the rule to elements of $\mathbb{D}$.*

Definition 18 (Meaning of a Program) *The meaning of a program is given by its unique minimal model.*

Theorem 1 *Let P be a program and $\mathcal{I}$ be an interpretation. If P admits a model including $\mathcal{I}$, then P admits a minimal model containing $\mathcal{I}$.*

Proof Let P be a program and $\mathcal{I}$ an interpretation such that P admits a model containing $\mathcal{I}$. Let χ be the set of models of P containing $\mathcal{I}$. We must show that $\cap\chi$ satisfies all the rules in P. Let $r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$ be a rule in P and ν a valuation based on $\mathbb{D}^{ext}_{\cap\chi}$ such that $\forall i \in [1, n]$, $\nu(L_i) \in \cap\chi$, and $\forall i \in [1, m]$, $\nu(c_i)$ satisfiable. $\forall \chi_k \in \chi$, since $\cap\chi \subseteq \chi_k$, we have $\forall i \in [1, n] \nu(L_i) \in \chi_k$, and $\forall i \in [1, m] \nu(c_i)$ satisfiable. Thus $\nu(H) \in \chi_k$, since χ_k is a model of P. Hence, $\nu(H) \in \cap\chi$. So, $\cap\chi$ satisfies any rule in P. Because each instance χ_k in χ contains $\mathcal{I}$, $\cap\chi$ contains $\mathcal{I}$. Hence, $\cap\chi$ is a model of P containing $\mathcal{I}$. By construction, χ is the minimal model of P containing $\mathcal{I}$.

□

6.3.2 Fix-point Semantics

The fix-point semantics is defined in terms of an immediate consequence operator, T_P, that maps interpretations to interpretations. An interpretation of a program is any subset of all ground atomic formulas built from predicate symbols in the language and elements in $\mathbb{D}$.

Each application of the operator T_P may create new atoms which may contain new objects (because of the constructive rule). We show below that T_P is monotonic and continuous. Hence, it has a least fix-point that can be computed in a bottom-up iterative fashion.

Recall that the language of terms has three countable disjoint sets: a set of atomic values ($\mathbb{D}_1$), a set of entities ($\mathbb{D}_2$), and a set of generalized intervals ($\mathbb{D}_3$). A constant generalized interval is an element of $\mathbb{D}_3$. We define $\mathbb{D} = \mathbb{D}_1 \cup \mathbb{D}_2 \cup \mathbb{D}_3$.

Definition 19 (Extensions) *Given a set $\mathbb{D}_3$ of generalized interval objects, the extension of $\mathbb{D}_3$, written $\mathbb{D}^{ext}_3$, is the set of objects containing the following elements:*

- *each element in $\mathbb{D}_3$;*
- *for each pair of elements in $\mathbb{D}_3$, the element resulting from their concatenation.*

Definition 20 (Extended Active Domain) *The active domain of an interpretation $\mathcal{I}$, noted $\mathbb{D}_\mathcal{I}$ is the set of elements appearing in $\mathcal{I}$, that is, a subset of $\mathbb{D}_1 \cup \mathbb{D}_2 \cup \mathbb{D}_3$. The extended active domain of $\mathcal{I}$, denoted $\mathbb{D}^{ext}_\mathcal{I}$, is the extension of $\mathbb{D}_\mathcal{I}$, that is, a subset of $\mathbb{D}_1 \cup \mathbb{D}_2 \cup \mathbb{D}^{ext}_3$.*

Lemma 1 *If $\mathcal{I}_1$ and $\mathcal{I}_2$ are two interpretations such that $\mathcal{I}_1 \subseteq \mathcal{I}_2$, then $\mathbb{D}^{ext}_{\mathcal{I}_1} \subseteq \mathbb{D}^{ext}_{\mathcal{I}_2}$.*

Definition 21 (Immediate consequence Operator) *Let P be a program and $\mathcal{I}$ an interpretation. A ground atom A is an immediate consequence for $\mathcal{I}$ and P if either $A \in \mathcal{I}$, or there exists a rule $r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$ in P, and there exists a valuation ν, based on $\mathbb{D}^{ext}_{\mathcal{I}}$, such that:*

- *$A = \nu(H)$, and*
- *$\forall i \in [1, n]$, $\nu(L_i) \in \mathcal{I}$, and*
- *$\forall i \in [1, m]$, $\nu(c_i)$ is satisfiable.*

Definition 22 (T-Operator) *The operator T_P associated with program P maps interpretations to interpretations. If $\mathcal{I}$ is an interpretation, then $T_P(\mathcal{I})$ is the following interpretation:*

$$T_P(\mathcal{I}) = \{A \mid A \text{ is an immediate consequence for } \mathcal{I} \text{ and P}\}$$

Lemma 2 (Monotonicity) *The operator T_P is monotonic; i.e., If $\mathcal{I}_1$ and $\mathcal{I}_2$ are two interpretations such that $\mathcal{I}_1 \subseteq \mathcal{I}_2$, then $T_P(\mathcal{I}_1) \subseteq T_P(\mathcal{I}_2)$*

Proof Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two interpretations such that $\mathcal{I}_1 \subseteq \mathcal{I}_2$. We must show that if an atom A is an immediate consequence for $\mathcal{I}_1$ and P, then $A \in T_P(\mathcal{I}_2)$.
Since A is an immediate consequence for $\mathcal{I}_1$ and P, at least one of the following cases applies:

- $A \in \mathcal{I}_1$. Then $A \in \mathcal{I}_2$, and thus $A \in T_P(\mathcal{I}_2)$;
- there exists a rule $r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$ in P and a valuation ν, based on $\mathbb{D}^{ext}_{I_1}$, such that $A = \nu(H)$, and $\forall i \in [1, n]\ \nu(L_i) \in \mathcal{I}_1$, and $\forall i \in [1, m] \nu(c_i)$ satisfiable. Following the Lemma 1, ν is also a valuation based on $\mathbb{D}^{ext}_{I_2}$. Since $\mathcal{I}_1 \subseteq \mathcal{I}_2$, we have $\nu(L_i) \in \mathcal{I}_2$ $\forall i \in [1, n]$ and $\nu(c_i)$ satisfiable $\forall i \in [1, m]$. Hence $A \in T_P(\mathcal{I}_2)$.

□

Theorem 2 (Continuity) *The operator T_P is continuous, that is, if $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots$ are interpretations such that $\mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \mathcal{I}_3 \ldots$ (possibly infinite sequence), then $T_P(\bigcup_i \mathcal{I}_i) \subseteq \bigcup_i T_P(\mathcal{I}_i)$.*

Proof Let $\mathbb{I} = \bigcup_i \mathcal{I}_i$ and let A be an atom in $T_P(\mathbb{I})$. We must show that A is also in $\bigcup_i T_P(\mathcal{I}_i)$. At least one of the following two cases applies:

- $A \in \mathbb{I}$, i.e., $A \in \bigcup_i \mathcal{I}_i$. Then, there exists some j such that $A \in \mathcal{I}_j$. Thus, $A \in T_P(\mathcal{I}_j)$ and consequently $A \in \bigcup_i T_P(\mathcal{I}_i)$.
- There exists a rule $r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$ in P and a valuation ν on $\mathbb{D}^{ext}_{\mathbb{I}}$ such that $\nu(L_i) \in \mathbb{I}\ \forall i \in [1, n]$ and $\nu(c_i)$ satisfiable $\forall i \in [1, m]$. Since $\nu(L_i) \in \mathbb{I}$, there exists some j_i such that $\nu(L_i) \in \mathcal{I}_{j_i}$. In addition, since the $\mathcal{I}_k$ are increasing, there exists some l, such that $\mathcal{I}_{j_i} \subseteq \mathcal{I}_l$ for all j_i. Hence, $\nu(L_i) \in \mathcal{I}_l\ \forall i \in [1, n]$ and $\nu(c_i)$ satisfiable $\forall i \in [1, m]$. Let $V = var(L_1, \ldots, L_n)$ be the set of variables in the rule r, and let $\nu(V)$ be the result of applying ν to each variable in V. $\nu(V)$ is a finite subset of $\mathbb{D}^{ext}_{\mathbb{I}}$ since ν is based on $\mathbb{D}^{ext}_{\mathbb{I}}$. We have $\nu(L_i) \in \mathcal{I}_l\ \forall i \in [1, n]$ and $\nu(c_i)$ satisfiable $\forall i \in [1, m]$. Thus, $\nu(var(L_i)) \subseteq \mathbb{D}^{ext}_{\mathcal{I}_l}$ $\forall i \in [1, n]$. Then $A \in T_P(\mathcal{I}_l)$ $(A = \nu(H))$. Consequently $A \in \bigcup_i T_P(\mathcal{I}_i)$.

□

Lemma 3 *$\mathcal{I}$ is a model of P iff $T_P(\mathcal{I}) \subseteq \mathcal{I}$.*

Proof
" $\Rightarrow$ " If $\mathcal{I}$ is an interpretation and P a program, then let $cons(P, \mathcal{I})$ denote the set of all ground facts which are immediate consequences for $\mathcal{I}$ and P.

$$T_P(\mathcal{I}) = \{A \mid A \text{ is an immediate consequence for } \mathcal{I} \text{ and P}\}$$

For any element A in $cons(P, \mathcal{I})$, at least one of the following cases holds:

- $A \in \mathcal{I}$. By definition of immediate consequence;
- there exists a rule $r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$ in P, and a valuation ν such that $\forall i \in [1, n]\, \nu(L_i) \in \mathcal{I}$, and $\forall i \in [1, m]\, \nu(c_i)$ satisfiable, and $A = \nu(H)$. Since $\mathcal{I}$ is a model of P, $\mathcal{I}$ satisfies r ($\mathcal{I} \models r$), and then $A \in \mathcal{I}$. Thus, $T_P(\mathcal{I}) \subseteq \mathcal{I}$.

" $\Leftarrow$ " Let $\mathcal{I}$ be an interpretation and P be a program. Let $r : H \leftarrow L_1, \ldots, L_n, c_1, \ldots, c_m$ be any rule in P and ν any valuation. If $\forall i \in [1, n]\, \nu(L_i) \in \mathcal{I}$ and $\forall i \in [1, m]\, \nu(c_i)$ satisfiable, then $\nu(H) \in T_P(\mathcal{I})$. Because $T_P(\mathcal{I}) \subseteq \mathcal{I}$, we have $\nu(H) \in \mathcal{I}$, and then $\mathcal{I}$ satisfies r ($\mathcal{I} \models r$). Hence $\mathcal{I} \models P$.

□

Lemma 4 *Each fix-point of T_P is a model for P.*

Proof Follows immediately from Lemma 3.

□

Theorem 3 *Let P be a program and $\mathcal{I}$ an input such that the minimal model for P exists, then the minimal model and the least fix-point coincide.*

Proof Let P be a program and $\mathcal{I}$ an interpretation, Let us denote by $P(\mathcal{I})$ the minimal model of P containing $\mathcal{I}$. According to lemma 3, $T_P(P(\mathcal{I})) \subseteq P(\mathcal{I})$. T_P is monotonic, so $T_P(T_P(P(\mathcal{I}))) \subseteq T_P(P(\mathcal{I}))$, and then $T_P(P(\mathcal{I}))$ is a model of P containing $\mathcal{I}$. As $P(\mathcal{I})$ is the minimal model containing $\mathcal{I}$, we have $P(\mathcal{I}) \subseteq T_P(P(\mathcal{I}))$. As $P(I)$ is a fix-point of P and also a minimal model of P, each fix-point of T_P containing $\mathcal{I}$ is a model of P containing $P(\mathcal{I})$. Thus $P(\mathcal{I})$ is the minimal model of P containing $\mathcal{I}$.

□

For Datalog with set order constraints, queries are shown to be evaluable bottom-up in closed form and to have DEXPTIME-complete data complexity [36]. For a rule language with arithmetic order constraints, the answer to a query can be computed in PTIME data complexity [37]. As a consequence, we obtain a lower bound complexity for query evaluation in our rule based query language.

7 Conclusion and Future Work

There is a growing interest in video databases. We believe that theoretical settings will help understanding related modeling and querying problems. This will lead to the development of powerful systems for managing and exploiting video information.

In this paper, we have addressed the problem of developing a video data model and a formal, rule-based, constraint query language that allow the definition and the retrieval by content of video data. The primary motivation of this work was that objects and time intervals are relevant in video modeling and the absence of suitable supports for these structures in traditional data models and query languages represent a serious obstacle.

The data model and the query language allow (a) an abstract representation of the visual appearance of a video able to support modeling and retrieval techniques; (b) a semantic data modeling styled representation of the video content, independent from how the content information is obtained; (c) a relational representation of the association between objects within a video sequence.

This paper makes the following contributions. (1) We have developed a simple and useful video data model that integrates relations, objects and constraints. Objects allow to maintain an object-centered view inherent to video data. Attributes and relations allow to capture relationships between objects. It simplifies the indexing of video sequences. (2) We have developed a declarative, rule-based, constraint query language to reason about objects and facts, and to build new sequences from others. This functionality can be useful in virtual editing for some applications. The language provides a much more declarative and natural way to express queries.

Due to the complex nature of video queries, the query language presents a facility that allows a user to construct queries based on previous queries. In addition, as all properties inherent to image data are also part of video data, the framework presented here naturally applies to image data.

There are many interesting directions to pursue.

- An important direction of active research is to extend our framework to incorporate abstraction mechanisms such as *classification*, *aggregation*, and *generalization*.
- Another important direction is to study the problem of sequence presentation. Most existing research systems use template-based approach [3] to provide the automatic sequencing capability. With this approach, a set of sequencing templates is predefined to confine the user's exploration to a certain sequencing order. The problem is that this approach is domain-dependent and relies on the availability of a suitable template for a particular query. We believe that a framework based on declarative graphical (visual) languages [35] will offer more possibilities and flexibility in the specification of sequence presentations.

We are investigating these important research directions.

References

[1] Sibel Adali, Kasim S. Candan, Su-Shing Chen, Kutluhan Erol, and V. S. Subrahmanian. Advanced video information system: Data structures and query processing. *ACM-Springer Multimedia System*, 4:172–186, 1996.

[2] T. G. Aguierre Smith and G. Davenport. The stratification system, a design environment for random access video. Technical report, The Media Lab, M.I.T, 1992.

[3] T. G. Aguierre-Smith and N. C. Pincever. Parsing movies in context. In *Proceedings of USENIX (Summer 1991), USENIX Association, Berkeley, California*, pages 157–168, 1991.

[4] Gulrukh Ahanger, Dan Benson, and T.D.C. Little. Video query formulation. In Wayne Niblack and Ramesh C. Jain, editors, *Storage and retrieval for image and video database III (SPIE'95), San Jose, California*, pages 280 – 291, February 1995.

[5] Alexander Aiken and Edward L. Wimmers. Solving systems of set constraints (extended abstract). In IEEE Computer Society Press, editor, *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California*, pages 329–340, 1992.

[6] Timothy Arndt and Angela Guercio. An object-oriented multimedia database system with versioning and content-based retrieval. In Wayne Niblack and Ramesh C. Jain, editors, *Storage and retrieval for image and video database III (SPIE'95), San Jose, California*, pages 340 – 351, February 1995.

[7] Alberto Del Bimbo, M. Campanai, and P. Nesi. A three-dimensional iconic environment for image database querying. *IEEE Trans. on Software Engineering*, 19(1):997–1011, 1993.

[8] Alexander Brodsky and Yoram Kornatzky. The *lyric* language: Querying constraint objects. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD'95), San Jose, California*, pages 35–46, May 1995.

[9] N.S. Chang and K.S. Fu. Picture query languages for pictorial data-base systems. *Computer*, 14(11):23–33, November 1981.

[10] Jan Chomicki and Tomasz Imielinski. Relational specifications of infinite query answers. *ACM Transactions on Database Systems*, 18(2):181–223, June 1993.

[11] Tat-Seng Chua and Li-Qun Ruan. A video retrieval and sequencing system. *ACM Transactions on Information Systems (TOIS)*, 13(4):373–407, October 1995.

[12] Cyril Decleir, Mohand-Saïd Hacid, and Jacques Kouloumdjian. Modeling and querying video data: A hybrid approach. In *Proceedings of the IEEE Workshop on Content-Based Access of Image & Video Libraries (CBAIVL'98), Santa Barbara, CA, USA*, 1998.

[13] Cyril Decleir, Mohand-Saïd Hacid, and Jacques Kouloumdjian. Modeling and querying video databases. In *Proceedings 24th EUROMICRO 98 Conference Workshop on Multimedia and Telecommunications, Vasteras, Sweden*, 1998. To appear.

[14] Sadashiva Devadiga, David A. Kosiba, Ullas Gargi, Scott Oswald, and Rangachar Kasturi. A semiautomatic video database system. In Wayne Niblack and Ramesh C. Jain, editors, *Storage and retrieval for image and video database III (SPIE'95), San Jose, California*, pages 262 – 267, February 1995.

[15] Nevenka Dimitrova. The myth of semantic video retrieval. *ACM Computing Surveys*, 27(4):584–586, December 1995.

[16] D. Le Gall. Mpeg: a video compression standard for multimedia applications. *Communications of the ACM*, 1991.

[17] D. Le Gall. The mpeg compression algorithm: a review. *Communications of the ACM*, 1991.

[18] Stéfane Grumbach and Jianwen Su. Dense-order constraint databases. In *Proceedings of the 1995 Symposium on Principles of Database Systems (PODS'95), San Jose, California*, pages 66–77, June 1995.

[19] Stéfane Grumbach, Jianwen Su, and Christophe Tollu. Linear constraint query languages expressive power and complexity. In Daniel Leivant, editor, *Proceedings of the International Workshop on Logic and Computational Complexity (LCC'94), Indianapolis, IN, USA*, pages 426–446, October.

[20] Bernard J. Haan, Paul Kahn, Victor A. Riley, James H. Coombs, and Norman K. Meyrowitz. Iris hypermedia services. *Communications of the ACM*, 35(1):35–51, January 1991.

[21] Arun Hampapur, Ramesh Jain, and Terry E. Weymouth. Indexing in video databases. In Wayne Niblack and Ramesh C. Jain, editors, *Storage and retrieval for image and video database III (SPIE'95), San Jose, California*, pages 292 – 306, February 1995.

[22] K. Hirata and T. Kato. Query by visual example. In *Proceedings of the Third International Conference on Extended Database Technology (EDBT'92), Vienna, Austria*, pages 56–71, March 1992.

[23] Rune Hjelsvold and Roger Midtstraum. Modelling and querying video data. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94), Santiago, Chile*, 1994.

[24] Markus Höhfeld and Gert Smolka. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, Germany, October 1988.

[25] Liusheng Huang, John Chuang-Mong Lee, Qing Li, and Wei Xiong. An experimental video database management system based on advanced object-oriented techniques. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage and retrieval for image and video database IV (SPIE'96), San Jose, California*, pages 158 – 169, February 1996.

[26] T. Joseph and A.F. cardenas. Picquery: A high level query language for pictorial database management. *IEEE Trans. on Software Engineering*, 14(5):630–638, May 1988.

[27] Michael Kifer and James Wu. A logic for object-oriented logic programming (maier's o-logic revisited). In *Proceedings of the 1989 Symposium on Principles of Database Systems (PODS'89), Philadelphia, Pennsylvania*, pages 379–393, March 1989.

[28] M. La Cascia and E. Ardizzone. Jacob : Just a content-based query system for video databases. In *ICASSP-96, Atlanta*, 1996.

[29] Wendy E. Mackay and Glorianna Davenport. Virtual video editing in interactive multimedia applications. *Communications of the ACM*, 32(7):802–810, July 1989.

[30] Sherry Marcus and V. S. Subrahmanian. Foundations of multimedia database systems. *Jounal of the ACM*, 43(3):474–523, 1996.

[31] Giansalvatore Mecca and Anthony J. Bonner. Sequences, datalog and transducers. In *Proceedings of the 1995 Symposium on Principles of Database Systems (PODS'95), San Jose, California*, pages 23–35, May 1995.

[32] Carlo Meghini. Towards a logical reconstruction of image retrieval. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage and retrieval for image and video database IV (SPIE'96), San Jose, California*, pages 108–119, February 1996.

[33] W. Niblack, R. Barber, W. Equitz, M. Flickner, D. Petkovic, and P. Yanker. The qbic project: Querying images by content using color, texture, and shape. In *IS&T/SPIE Symposium on Electronic Imaging;Science and Technology, San Jose, California*, February 1993.

[34] Eitetsu Oomoto and Katsumi Tanaka. Ovid: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629–643, August 1993.

[35] Jan Paredaens, Peter Peelman, and Letizia Tanca. G-log: A declarative graphical query languages. In Claude Delobel, Michael Kifer, and Yoshifumi Masunaga, editors, *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases (DOOD'91), Munich, Germany*, volume 566 of *LNCS*, pages 108–128, December 1991.

[36] Peter Z. Revesz. Datalog queries of set constraint databases. In Georg Gottlob and Moshe Y. Vardi, editors, *Proceedings of the 5th International Conference on Database Theory (ICDT'95), Prague, Czeck Republic*, pages 425–438, January 1995. LNCS 893.

[37] Divesh Srivastava, Raghu Ramakrishnan, and Peter Z. Revesz. Constraint objects. In *Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming (PPCP'94)*, number 874 in LNCS, pages 218–228. Springer Verlag, 1994.

[38] ACM TOIS. Special issue in video information systems. *ACM Transactions on Information Systems (TOIS)*, 13(4), October 1995.

[39] David Toman. Point vs. interval-based query languages for temporal databases. In *Proceedings of the 1996 Symposium on Principles of Database Systems (PODS'96), Montréal, Canada*, pages 58–67, June 1996.

[40] Yoshinobu Tonomura. Video handling based on structured information for hypermedia systems. In *Proceedings of the International Conference on Multimedia Information Systems'91, Singapore*, pages 333–344. McGraw Hill, New York, 1991.

[41] A. Yoshitaka, Y. Hosoda, M. Yoshimitsu, M. Hirakawa, and T. Ichikawa. Violone : Video retrieval by motion example. *Journal of Visual languages and Computing*, 7:423–443, 1996.

[42] Aidong Zhang and Shavinder Multani. Implementation of video presentation in database systems. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage and retrieval for image and video database IV (SPIE'96), San Jose, California*, pages 228 – 238, February 1996.