A Database Approach for Modeling and Querying Video Data

Mohand-Saïd Hacid, *Member*, *IEEE Computer Society*, Cyril Decleir, and Jacques Kouloumdjian, *Member*, *IEEE Computer Society*

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, and 2) video frames which contain these entities. To represent this information, our data model allows facts as well as objects and constraints. The model consists of two layers: 1) Feature & Content Layer (or Audiovisual Layer), intended to contain video visual features such as colors, contours, etc., 2) Semantic Layer, which provides the (conceptual) content dimension of videos. We present a declarative, rule-based, constraint query language that can be used to infer relationships about information represented in the model. Queries can refer to the form dimension (i.e., information of the Feature & Content Layer), to the content dimension (i.e., information of the Semantic Layer), or to both. A program of the language is a rule-based system formalizing our knowledge of a video target application and it can also be considered as a (deductive) video database on its own right. The language has both a clear declarative and operational semantics.

Index Terms—Content-based access of video, video database, video query, video indexing, video representation, rule-based query language, object-oriented modeling, constraint query language.

1 Introduction

With recent progress in compression technology, it is possible for computers to store huge amounts 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 e-mail, etc. Though only a partial list, these advanced applications need new techniques and tools for managing video data.

Video analysis and content retrieval based on semantics require multidisciplinary research effort in areas such as computer vision, image processing, data compression, databases, information systems, etc. (See, e.g., [68], [24]). 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 nonsequential 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 [35], [70] to describe the necessary semantic objects is not sufficient. Additional techniques are needed. As stated in [18], 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. The video data model must be powerful enough to allow both the expression of sophisticated content representation and its 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. Increasingly, users are demanding

- *M.-S. Hacid is with the Department of Computer Sciences, Purdue University, West Lafayette, IN 47907. E-mail: mshacid@cs.purdue.edu.*
- *C. Decleir and J. Kouloumdjian are with LISI-INSA, 20 av. A. Einstein, F-69621 Villeurbanne, France. E-mail: {cdecleir, koulou}@lisi.insa-lyon.fr.*

Manuscript received 16 Sept. 1999; revised 21 June 2000; accepted 18 July 2000.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 112565.

that video be managed by video database systems, similar to the way alphanumerical data is managed in existing databases, such as relational databases. This allows users to easily and effectively share, manipulate, store, query, retrieve, and browse such content-rich information resources. Much research has been done in the area of indexing and accessing video based on its visual features (color, shape, motion, etc.); however, relatively little research has been devoted to the problems of indexing, sharing, querying, and browsing the semantic content of video data. For applications, such as digital libraries[1] and distance learning, users are mostly interested in querying and retrieving the video in terms of "what the video is about" rather than "what its optical flow looks like."

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 [59], [73], [41]) 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 predefined, ad hoc operators, often incorporated to SQL, and, as far as we know, are not investigated in a *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 the 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 enable, for example, virtual editing [52], and should be able to perform exact as well as partial or fuzzy matching (see [5]).

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*.

1. There are many buzz-words for related activities including, but not limited to: multimedia databases, information mining, information warehouse, information retrieval, online information repositories, electronic library, operational image applications, imaging, world wide web (WWW), and wide area information services (WAIS) [32].

Paper outline. This paper is organized as follows: Section 2 summarizes the contributions of this work. Section 3 discusses related work and gives a brief summary of the object-oriented and temporal approaches proposed for modeling video data. Section 4 introduces the annotation process of video media. Section 5 proposes an original classification of some well-known video data models compared to ours, based on relevant criteria. Section 6 defines preliminary notions, mainly basic definitions, that will be used to specify constructs of our data model and query language. Section 7 formally introduces our video data model. Section 8 describes the underlying query language, defines the syntax, and develops its fixpoint semantics. By using the universal attachment, we show how to refer to the information of the Feature & Content Layer in queries. As an example, the language is used to express some interval relations. Section 9 gives the prototype implementation details. We conclude in Section 10 by anticipating the necessary extensions.

2 Contributions

Video information retrieval is complex since different types of information are associated with videos. These are:

1. Data which is not directly concerned with video content, but in some way related to it. It is referred to as content-independent metadata. Examples are *date*, *movie_title*, etc.
2. Data which refers to the visual content of videos. Two levels can be addressed:
   a. Data may refer to low/intermediate-level features like *color*, *texture*, *shape*, *spatial relationship*, etc., and their combination. It is also referred to as content-dependent metadata.
   b. Data may refer to content semantics. This data is also referred to as content-descriptive metadata. It is concerned with relationships of video entities with real-word entities or temporal events, etc.

Hence, video information can be regarded at several abstraction levels, from low-level perceptual clues to semantic aspects of content. Therefore, there is a need for data models and knowledge structures that organize visual information so as to provide effective retrieval at different levels of abstraction [12].

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 [55], [59], [39], [2], [53], [66], [14], [54], [33] 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. The proposed semantic models are too poor for video modeling. In general, they do not address relationships between contained semantics. We argue that associations are important in modeling real world and that these associations build other

high-level semantic units. We develop a simple video data model (in the spirit of semistructured data models), called *CoPaV*[2], on the basis of relation, object, and constraint paradigms. Objects of interest and relationships among objects can be attached to a temporal cohesion[3] either through attribute/value pairs or relations. We believe that our model is the first one combining objects and relations for specifying semantics of video data.

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. In addition, we wish to construct new sequences from old ones. To meet this goal, our language has an interpreted function term, that is, a constructive term to concatenate sequences. For example, consider a user who wants to compose a digital video presentation about the latest developments with economic reforms. First, the user searches a large collection of TV broadcasts for all video segments reporting on economic reforms and Simon, who is a notable economist. Then, the user may want to examine the context in which the segments have appeared: in the headline news or in a talk show, etc. Finally, the user chooses some segments and combines them such that they form a new video presentation that can be played out or stored. The originality of this language is that it integrates the declarative aspects of logical languages and constraint languages. Constraint languages are flexible tools for information retrieval.
3. In order to deal with queries addressing the content of both layers (i.e., Semantic Layer and Feature & Content Layer), we choose an appropriate interface between these two layers. The interface is inspired by the *universal attachment*, which is a mechanism for integrating diverse representation and reasoning methods into hybrid frameworks, found in artificial intelligence languages.
4. The data model we propose is characterized by the lack of any fixed and rigid schema, although, typically, video data has some implicit structure. An important and critical problem is the discovery of the structure implicit in our video data. This is especially important since video data are often accessed in an *explorative* or *browse* mode. For that, we propose to build a layer of classes on top of our data model. The classes are defined by a restricted form of rules and populated by computing a fixpoint.

The model and the query language use the point-based approach to represent periods of time associated with temporal cohesions. First-order queries can then be conveniently asked in a much more declarative and natural way [69]. There has been some previous research on the power of constraints for the implicit specification of temporal data [17].

To the best of our knowledge, this is the first proposal of an hybrid data model and a formal rule-based query language for modeling and querying video data and integrating in a single logical framework content-based retrieval of video data.

3 Related Work

With the advent of multimedia computers (PCs and workstations), the World Wide Web, and standard and powerful compression techniques,[4] 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* [72] and *JACOB* [9]. 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* [59], *AVIS* [2], and *VideoStar* [39].

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, [13], [16], [38], [43], [58].

In the framework of semantic modeling schemes for video retrieval by content, our work relates to several fields of research in databases and artificial intelligence. Semantic schemes attempt to model the meaning of video sequences, taking into account a number of aspects connected to what questions may be put, e.g., related to objects, spatial relationships between objects, events, and actions involving objects, temporal relationships between events and actions, user interaction. In the following, we shortly discuss the relationship to some of them regarding **temporal modeling** and **object-oriented modeling** of video data. We apologize if we left out other relevant references.

- **Temporal Modeling of Video Data**.
  Basically, video indexing means defining easy-to-retrieve meaningful[5] 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, videos have additional temporal and relational attributes and, in most cases, video data are hierarchical in structure. As a

2. *C*ombined *Pa*radigms for *V*ideo data.
3. A temporal cohesion is a set of pairwise nonoverlapping fragments in a video sequence.
4. Such as *MPEG-I* [29] and its successors.
5. But application dependent.

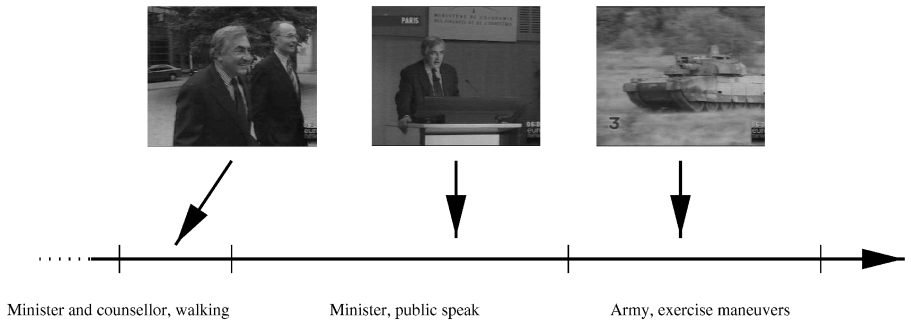

Fig. 1. Indexing by segmentation.

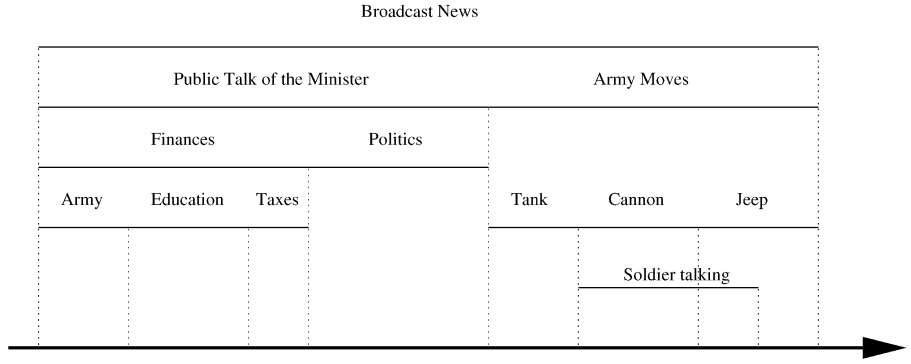

Fig. 2. Indexing by stratification.

consequence, a video indexing model should provide facilities to efficiently capture these additional attributes. Because of the temporal nature of video, information abstracting the video content is only true at a given time. Therefore, a temporal management of video information is required.

Historically, the *segmentation* [18] 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. Fig. 1 shows a basic segmentation of broadcast news. It can be easily seen that the time-line of the document is partitioned into individual segments, each of them being associated with a handwritten description.

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

This approach was criticized by Aguierre-Smith and Davenport [3] mainly because its strict temporal partitioning results in rough descriptions of video documents. As a consequence, they introduced a method, called *stratification*, which allows annotating facts individually. In this approach, each element of interest is being associated with a single temporal descriptor, which is an interval called *stratum*, as shown in Fig. 2. One can see that stratification allows overlapping of descriptions, therefore allowing the user to specify several levels of descriptions. For example, Fig. 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 *temporal cohesions*. 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 temporal cohesion is defined as a set of nonoverlapping intervals providing time boundaries to a description. This allows the handling of all occurrences of an entity in a video document with a single object. 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 temporal cohesion, as shown in Fig. 3: Three objects of interest are defined and each of them has a temporal cohesion that traces its presence on the screen. Therefore, this allows, with a single identifier, for instance, "Reporter," to make reference to all occurrences of "Reporter" in the document.

- **Object-Oriented Modeling of Video Data**.
  Oomoto and Tanaka [59] proposed a schemaless 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

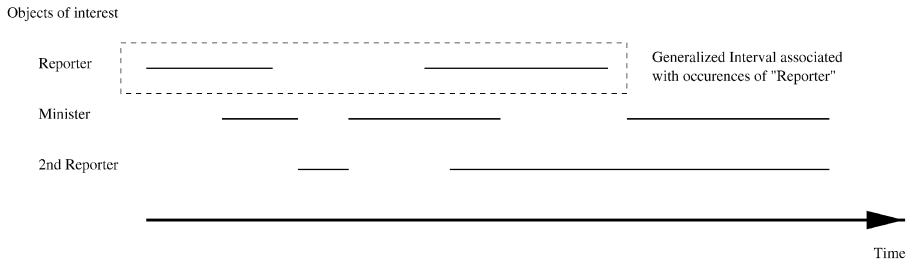

Fig. 3. Indexing based on temporal cohesion.

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 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. [2] 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 multivalued 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 that satisfy the specified query conditions.

Hjelsvold and Midtstraum [39] proposed a generic video data model. Their proposal combines ideas from the stratification [3] and the segmentation [18] 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.

Gibbs et al. [30] proposed an object-oriented approach to video databases. An audio/video database can be viewed as a collection of values (audio and video data) and activities (interconnectable components used to process values). Two abstraction mechanisms, temporal composition and flow composition, allow aggregation of values and activities. The database values (audio or video) are linear sequences of data elements so that the logical structure is not represented.

In [31], Gibbs et al. proposed an object-oriented data model for video data. The emphasis was on how to capture the aspects of time-based media (e.g., complex data encoding, compression, quality factors, timing) at the conceptual level. The authors proposed using conventional object-oriented data where these aspects are quantified and encoded as values of attributes. The model is supposed to be used by application programs to maintain and guarantee a certain quality of service.

Huang et al. [41] described the development of an experimental video database system which employs extended object-oriented features and techniques. By incorporating conceptual object clustering concepts and techniques, it enables users to dynamically form, among other things, video programs (or segments) from existing objects based on semantic features/index terms.

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

These semantic models are too poor for video modeling. In general, they do not address relationships between contained semantics. We argue that associations are important in modeling real world and that these associations build other high-level semantic units. In contrast to these proposals, we allow a more elaborated and structured description of the content of video sequences. Regarding the modeling, we have 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 other, either through attributes or through explicit relations, leading to more expressive relationships to link objects. Hence, video frames (which we call

temporal cohesions), as well as semantic objects (objects of interest in a temporal cohesion), are modeled and manipulated at the same level. Special queries, like temporal ones, can be expressed in a much more declarative manner. Additionally, none of these proposals addressed mixed queries (i.e., queries which refer to both Semantic Layer and Feature & Content Layer).

Despite these proposals and others on finding appropriate representation and indexing schemas for video data and systems architectures able to support such representations, there is little research work on finding semantic foundations for modeling and querying video data. We provide a *contribution in this direction*. We take a new look at the problem of modeling and querying video data and find that hybrid paradigms for databases lead to an interesting and promising approach.

4 Video Annotation

An annotation is descriptive information associated with part of a video document. The concept of annotation is very general. An annotation is any information which can be deduced from the content of a video document by a human or an algorithm. Thus, a text associated with a video segment is an annotation. A histogram of colors associated with an image is also an annotation.

An annotation is made up of information of localization and descriptive information. In the following, we describe these two kinds of information.

Information of Localization. By information of localization, we mean the information which makes it possible to attach descriptive information to a video stream. In its simplest form, information of localization can be a temporal interval to which one would attach, for example, descriptive information in the form of a free text. However, more complex forms can be considered, as in the case of an object motion [72], for example. Here, the information of localization will be the trajectory of the object, while descriptive information could be limited to a simple identifier (e.g., *car*).

If the annotation is done by the human, the aspect regarding localization of information is very simple. Because the annotation is strictly manual, one is restricted to the management of temporal intervals. Two annotation strategies can be identified according to whether one considers first the localization aspect or the description aspect of the annotations.

The first method consists of defining temporal intervals (therefore defining the localizations of information) and then specifying the associated descriptions. This method is called *segmentation*. Such a method allows us to split the video document into a set of disjoint temporal segments and then describe each segment individually. Thus, the segmentation rests on a temporal partitioning followed by a description of each interval. Usually, this temporal partitioning is based on the basic structural units (e.g., scene, shot) of the video document.

Contrary to the segmentation, the other approach considers the descriptive aspect of the annotations first. In other words, the annotator views the document until a relevant information is found. Then, he/she describes it and defines its temporal range. Thus, each information considered to be relevant has an associated time interval within a video stream.

Several authors proposed some improvements of these basic approaches (see, e.g., [39], [64], [21]). The resulting approaches are richer than classical ones because they are not limited to a strict partitioning of the document, but allow overlapping of annotations.

To summarize, one can consider that there are two levels of temporal management of video data:

- *Free temporal management*. This one includes the segmentation and the stratification. There is no constraint associated with this temporal management. The annotator is free to define segments and strata. However, in both cases, because the manual annotation is painful and, thus, inevitably short, the user tends to use the structures of the video document (e.g., shot, scene, sequence).
- *Temporal management based on the structure of video document*. Given that shots, scenes, and sequences are the structural units of an audio-visual document, their use in indexing is desirable because they make it possible to guide the annotator. For example, the models presented in [39], [64], [21] all include a hierarchy shot-scene-sequence. This means that the process of annotation is different according to the structural level we consider.

Information of Description. This information aims at describing part of the content of a video document which is supposed relevant. In the following, we distinguish two kinds of information of description:

- *High-level information*. The first applications of indexing video documents are based on a simple approach, namely the segmentation. With this approach, each segment is described individually by means of a natural language. This is the most obvious method of description: the use of free text. An example of use of such an approach is given in [18]. This approach allows complete freedom in the construction of descriptions. However, this is at the price of a weak exploitability of these annotations. Indeed, performing an automatic search in a free text is difficult and one is restricted to the search of simple keywords in annotations. In this case, traditional approaches regarding text retrieval can be used.

  Instead of using a free text, some systems impose a vocabulary resulting from a thesaurus. Davis [22] proposes an approach based on the definition of a visual language dedicated to the indexing of audio-visual documents. Descriptive information in this language is built by means of a series of icons composing an iconic sentence. The iconic vocabulary was created in such a way that the most common descriptions can be easily obtained. One notes the presence of icons specifically dedicated to cinematographic language (zoom, travelling, etc.). The advantage of such approaches is their ability to reduce

TABLE 1
Classification of Audio-Visual Information Models

Model	Level	Temporal Management	The knowledge which are used
Segmentation [18]	Specific	Segmentation	Not-specified
Stratification [3]	Specific	Stratification	Not-specified
VideoStar [39]	High	Segmentation	Kinematics
Corridoni *et al.* [20]	High	Segmentation	Kinematics
Rowe *et al.* [64]	High	Segmentation	Kinematics
OVID [59]	High	Stratification	Semantic
VRSS [18]	High and low	Segmentation	Physical
AVIS [2]	High	Stratification	Semantic
MediaStreams [22]	High	Stratification	Semantic
Algebraic Video [25]	High	Stratification	Kinematics
Jacob [9]	Low	Segmentation	Physical
Swanberg *et al.* [67]	Low	Segmentation	Physical and Kinematics
Violone [72]	Low	Stratification	Physical
CoPaV[23]	**High and low**	**Temporal cohesions**	**all**

the descriptive potential of the annotator by forcing her/him to use restricted vocabulary in order to overcome the later operational difficulties related to automatic search.

Another solution to the problem of the exploitability is the preliminary definition of attributes on which the analysis of the content of the video sequence will be based. It is possible to define metadata for audio-visual information. Such an approach is used in VideoStar [39].

- *Low-level information*. The management of low-level information fits naturally in the field of automatic processing of images and video. Releasing the human from the task of indexing video documents is a very desirable objective. However, no algorithm can currently compete with human possibilities in interpreting video information. Consequently, automatic extraction aims to assist the human in the task of indexing and thus to reduce the cost of this task. Given the performances of the algorithms at our disposal, automatic extraction aims at providing indices on audio-visual contents, which are called primary descriptors. These descriptors provide additional indications—sometimes secondary—on the content of an audio-visual document. So, for example, the systems *VISION* [50] and *Informedia* [37] use the sound track of an audio-visual document in order to carry out a more effective cutting in scenes. In general, any audio-visual indexing system having an automatic extraction module has an extractor of histograms of color (see, for example, [9], [28]).

5 Classification of Video Data Models

In this section, we give some relevant criteria for proposing a first classification of the data models for audio-visual information. We distinguish three main aspects:

- the management of the various levels for capturing audio-visual information;
- the management of the temporal dimension;
- the use of external knowledge.

We consider these three criteria as dominating. The criterion, *management of various levels for capturing audio-visual information*, allows us to identify three categories of approaches:

1. low-level systems aiming at defining autonomous tools for indexing;
2. high-level systems intended to manage audio-visual information as it is perceived by a human;
3. mixed systems intended to build operational audio-visual applications on the basis of current technology.

We consider as second criterion the *temporal management* of the audio-visual information, mainly the stratification and the segmentation. These are two opposite approaches in modeling audio-visual information. The first one considers this information independently of the structure of the document, while the second is based on a structural approach of the content of an audio-visual document. Finally, the use of *external knowledge* is our last classification criterion. By fixing preliminary apprehension rules, this knowledge allows us to guide or improve the extraction of relevant information on the content of audio-visual document.

Table 1 gives a classification of some models according to previous criteria. We wish to point out that the models used here are intended to be illustrative, rather than comprehensive. We apologize if we left out other models.

We use the following legenda:

- **Model**: name and bibliographical reference of the model;
- **Level**: The studied models can be at various levels of apprehension. We distinguish three levels:
  - The high-level corresponds to a model specialized in the representation of manually indexed information;

- The low-level corresponds to a model specialized in the representation of primary characteristics (automatically extracted);
- There exist approaches not limited to one of these two levels. This is for two reasons: 1) because the model considers a particular aspect of audio-visual information as, for example, the segmentation or the stratification, and we use, in this case, the term "*specific*;" 2) because the model is concerned with the management of both levels of apprehensions and we use, in this case, the term "*high and low*."

- **Temporal Management**: We indicate to which kind of temporal management the different models are related. We distinguish the two traditional approaches, namely *segmentation* and *stratification*.
- **The used Knowledge**: We distinguish the different cases reported in [36] (physical knowledge, kinematics knowledge, and semantic knowledge) and we add the characteristic "*not-specified*." This one corresponds to the basic models: segmentation and stratification. Indeed, these two approaches are only directed toward the study of the temporal management of video information. The integration of knowledge is secondary for such models.

Please note that our data model: 1) extends the traditional approaches (i.e., segmentation and stratification) for logical partitioning of video sequences, and 2) is not *sensitive* to a particular knowledge; the reason is that the video data in the Feature & Content Layer, for example, can be specified and organized according to kinematics, physical, or semantic knowledge.

6 Basic Definitions

This section provides the basic concepts used to design our video data model and the underlying rule-based, constraint query language.

Definition 1 (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 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 [34].

In this paper, we restrict ourselves to constraints of the form $x\theta c$.

Definition 2 (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 [6] 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 [66].

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

Definition 3 (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 4 (Temporal Cohesion). *A temporal cohesion is a set of pairwise nonoverlapping intervals. Formally, a temporal cohesion can be represented as a disjunction of time intervals.*

7 Basic Video Formalism

In this section, we introduce our layered view of video data and show how different modeling paradigms (i.e., object, constraint, relation) are combined to support modeling of this data.

7.1 Introduction

Digital video is content-rich information carrying media of massive proportion. In fact, the data volume of video is about seven orders of magnitude larger than a structured data record [26]. Video data also carries vital temporal and spatial information. Moreover, the structure of video data and the relationships among them is very complex and ill-defined. These unique characteristics pose great challenges for the management of video data in order to provide efficient and content-based user access. We propose a video data model that is based on logical video segment layering, video annotations, and associations between them. The model supports user queries and retrieval of the video data based on its semantic content. In addition, the resulting database is schemaless; in other words, so-called schema is not fixed and require the ability to be dynamically changed at any time. This property makes our model have certain similarities to semistructured data models (see, among others, [1], [15].

In the following, we propose two layers for representing video content (Fig. 4)

1. *Feature & Content Layer*. It contains video visual features (e.g., color, shape, motion). This layer is characterized by a set of techniques and algorithms

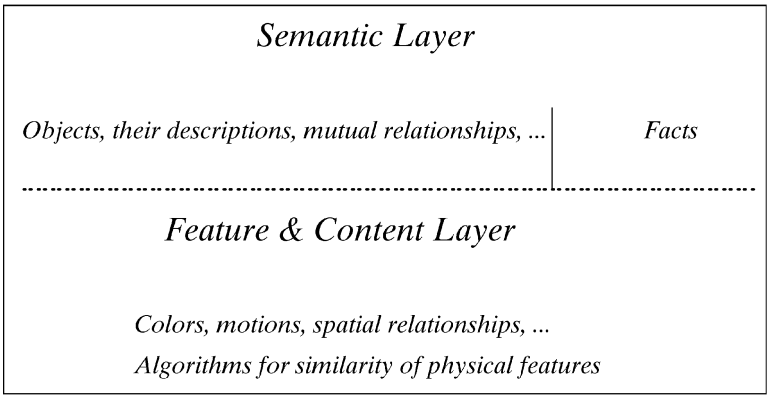

Fig. 4. Two layers for video content.

allowing the retrievial of video sequences based on the similarity of visual features.

2. *Semantic Layer*. This layer contains objects of interest, their descriptions, and relationships among objects, based on extracted features. It constitutes what we call the extensional part of a video database. Objects in a video sequence are represented in the object layer as visual entities. Instances of visual objects consist of conventional attributes (e.g., name, actor-ID, date, etc.).

In the following section, we start by introducing a data model for capturing information of the semantic layer, then we give a rule-based constraint query language for querying this layer. In Section 8.4, we extend this query language to accommodate the Feature & Content Layer.

7.2 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 oids at logical level. We have essentially two types of objects: 1) temporal cohesion 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 the 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 temporal cohesions.

Furthermore, in order to be able to associate a time interval to a temporal cohesion 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 5 (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 6 (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 $o.A_i$. Note that v_i could be a constraint (see Definition 5) specifying time boundaries in cases where the object of interest is a sequence.

Example. Let us see how the example given in [2] can be modeled in our framework. First, let us recall the example (extracted from [2]): It concerns the movie "*The Rope*" by Alfred Hitchcock. This movie has 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 receive any punishment for it. Hence, they kill their friend David and hide him inside a chest in the living room. As a sign of committing the perfect crime, 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 in which 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 temporal cohesions:

1. The first (gi_1) corresponds to the period of time in the sequence when 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 when the party is given. This interval contains objects: Philip, Brandon, David, Janet, Kenneth, Kentley, Atwater, Rupert Cadell, and Chest.

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

$$gi_1 = (id_1, [entities : \{o_1, o_2, o_3, o_4\}, duration : (t > a_1 \wedge t < b_1), subject :'' murder,'' victim : o_1, murderer : \{o_2, o_3\}]).$$

$$gi_2 = (id_2, [entities : \{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8, o_9\}, duration : (t > a_2 \wedge t < b_2), subject : ''\text{Giving a party},'' host : \{o_2, o_3\}, guest : \{o_5, o_6, o_7, o_8, o_9\}]).$$

$$o_1 = (id_3, [name :'' David,'' role :'' Victim'']$$
$$o_2 = (id_4, [name :'' Philip,'' realname : ''\text{Farley Granger},'' role :'' Murderer'']$$
$$o_3 = (id_5, [name :'' Brandon,'' realname : ''\text{John Dall},'' role :'' Murderer'']$$
$$o_4 = (id_6, [identification :'' Chest'']$$
$$o_5 = (id_7, [name :'' Janet,'' realname : \text{Joan Chandler}'']$$
$$o_6 = (id_8, [name :'' Kenneth,'' realname :'' \text{Douglas Dick}'']$$
$$o_7 = (id_9, [name :'' Mr.Kentley,'' realname : ''\text{Cedric Hardwicke}'']$$
$$o_8 = (id_{10}, [name :'' Mrs.Atwater,'' realname : ''\text{Constance Collier}'']$$
$$o_9 = (id_{11}, [name :''\text{ Rupert Cadell},'' realname : ''\text{James Stewart}'']$$
$$in(o_1, o_4, gi_1)$$
$$in(o_1, o_4, gi_2)$$
$$\ldots$$

The first statement says that the temporal cohesion 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 temporal cohesions 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 temporal cohesion 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 temporal cohesion 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 [14] 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.

Fig. 5 shows the (complex) structure we are going to deal with. The two graphs gi_1 and gi_2 are both sequences sharing some objects, but they differ in how they are annotated.

8 Rule-Based, Constraint Query Language

Most video retrieval approaches either treat video as textual information, based on the assumption that video has been annotated, or as a raw image sequence signal to which signal processing techniques are applied. Two main approaches are used for retrieving audio-visual information, namely *Semantic Queries* (or *query-by-subject*) and *Audio-visual Queries*.

- *Semantic Queries*. In "query-by-subject," a keyword representing a semantic content is specified. The semantic content of video data is extracted from raw data in order to evaluate a query. A simple way for managing the semantic content of video data is to annotate it with text. Examples that provide content-based retrieval for a video database through textual annotation are [51], [71], [59]. In these studies, a textual description representing semantic contents is assumed to be defined for video data by a human. These studies focus on "retrieval-by-content," which cannot be extracted from a video data through image processing. Such kinds of information include, for example, the name of a road in a map, the name of a person appearing in a news video. In these systems, content-based retrieval for video data is internally replaced by a keyword retrieval for annotation. One of the advantages of this method is that it can easily be implemented. However, this approach is not practical in rich-information resources such as large video databases.

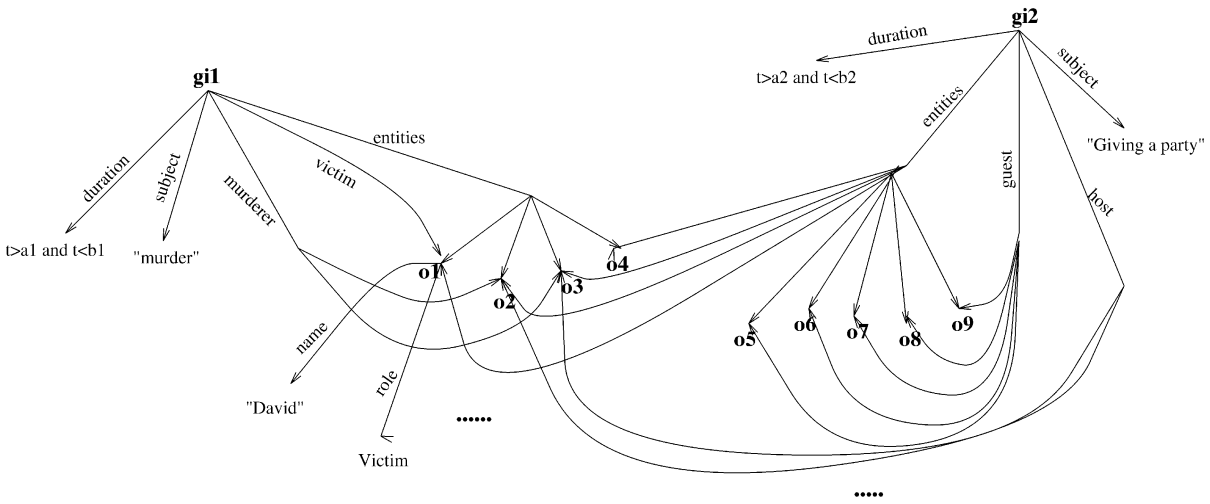

Fig. 5. Structure of the movie database extract.

- *Audiovisual Queries*. These queries depend only on the audio-visual information in the video and require a minimal degree of interpretation of the audio-visual signal in the video. The information needed to satisfy such queries can be derived by automatic or semiautomatic processing of the video data. For instance, the system QBIC [28] implements such possibilities.

In this paper, we propose a query language which allows us to specify queries referring to both Semantic and Audio-visual layers.

Many CLP dialects, and some of the related formalisms used in computational linguistics, provide for a combination of several "primitive" constraint languages. For example, in Prolog III [19], mixed constraints can be used to express lists of rational trees where some nodes can again be lists, etc. Rounds [63] introduces set-valued feature structures and non-well-founded sets, and many other suggestions for integrating sets into rule-based formalisms exist. The relationship between programming with constraints and database query languages was also investigated [44]. For a survey regarding problems we are faced with when combining constraints, see [11].

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,[6] on top of which relations can be defined by means of definite clauses.

This language has a model-theoretic and fixpoint semantics based on the notion of *extended active domain* of a database. The extended domain contains all temporal cohesion objects and their concatenations. The language has an interpreted function symbol for building new temporal cohesions 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 temporal cohesions I_1 and I_2.

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

Intuitively, a database in our model can be thought of as a graph (see Fig. 5) and a query in our query language can be viewed as specifying (constrained) paths in the graph.

8.1 Syntax

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 temporal cohesion objects.
2. A set $\mathcal{V}$ of variables called object variables and value variables and denoted by $X, Y, \ldots$;
3. A set $\tilde{V}$ of variables called temporal cohesion variables and denoted by $S, T, \ldots$.

If I_1 and I_2 denote temporal cohesion objects, temporal cohesion 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 [47] 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 for the definition of constructive rule).

6. For a formal definition of a constraint language, see [40].

Definition 7 (Predicate symbol). *We define the following predicate symbols:*

- *Each* $\mathrm{P} \in \mathcal{R}$ *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 temporal cohesion objects.*
- *A special unary predicate symbol* Object. *It can be seen as the class of all objects other than temporal cohesion objects.*

So, the only two extensional relations, besides those explicitly stated (i.e., $\mathcal{R}$), are Interval and Object.

Definition 8 (Atom). *If* P *is an* n*-ary predicate symbol and* $t_1, \ldots, t_n$ *are terms, then* $\mathrm{P}(t_1, \ldots, t_n)$ *is an atom. If* O *and* O' *denote objects, temporal cohesions, object variables, or temporal cohesion variables,* $Att_1, \ldots, Att_n$ *and* $Att'_1, \ldots, Att'_m$ *are attribute names, and* c *is a constant value, then*

$$O.Att_1 \ldots Att_n \; \theta \; c, O.Att_1 \ldots Att_n \; \theta \; O'$$

and

$$O.Att_1 \ldots Att_n \; \theta \; O'.Att'_1 \ldots Att'_m,$$

where θ *is one of* $=, <, \leq$ *(or their negation* $\neq, \geq, >$*) and* $n, m \geq 1$*, are called inequality atoms.*

Definition 9 (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 H 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 10 (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 11 (Program). *A program is a collection of range-restricted rules.*

Definition 12 (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 temporal cohesion, the set-valued attribute *entities* gives the set of semantic objects of interest in that temporal cohesion. 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 has been instantiated.

The query "list all temporal cohesions 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:

$$\begin{aligned} q(o) \leftarrow \; & Interval(G), Object(o), \\ & o \in G.entities, \\ & G.duration \Rightarrow (t > a \wedge t < b), \end{aligned}$$

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 temporal cohesions where the objects o_1 and o_2 appear together" can be expressed as:

$$\begin{aligned} q(G) \leftarrow \; & Interval(G), Object(o_1), \\ & Object(o_2), o_1 \in G.entities, \\ & o_2 \in G.entities \end{aligned}$$

or, equivalently, by:

$$\begin{aligned} q(G) \leftarrow \; & Interval(G), Object(o_1), \\ & Object(o_2), \{o_1, o_2\} \subseteq G.entities. \end{aligned}$$

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

$$\begin{aligned} q(O_1, O_2, G) \leftarrow \; & Interval(G), Object(O_1), \\ & Object(O_2), Rel(O_1, O_2, G), \\ & O_1 \in G.entities, \\ & O_2 \in G.entities. \end{aligned}$$

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

$$\begin{aligned} q(G) \leftarrow \; & Interval(G), Object(O), \\ & O \in G.entities, O.A = val. \end{aligned}$$

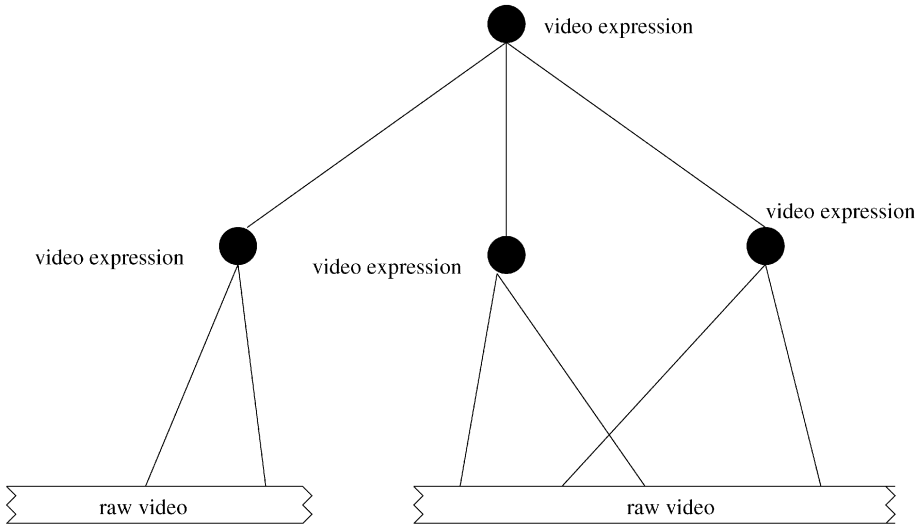

Fig. 6. Video expressions.

8.2 Inferring New Relationships

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

Example. Suppose we want to define the relation *contains*, which holds for two temporal cohesion 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:

$$\begin{aligned} contains(G_1, G_2) \leftarrow\ & Interval(G_1), \\ & Interval(G_2), \\ & G_2.duration \Rightarrow G_1.duration. \end{aligned}$$

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 temporal cohesions with their common objects, we write the following rule:

$$\begin{aligned} same\_object\_in(G_1, G_2, O) \quad \leftarrow\ & \\ & Interval(G_1), \\ & Interval(G_2), \\ & Object(O), \\ & O \in G_1.entities, \\ & O \in G_2.entities. \end{aligned}$$

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

$$\begin{aligned} concat\_temp\_cohesions(G_1 \otimes G_2) \quad \leftarrow\ & \\ & Interval(G_1), \\ & Interval(G_2), \\ & Object(o_1), \\ & Object(o_2), \\ & \{o_1, o_2\} \subseteq G_1.entities, \\ & \{o_1, o_2\} \subseteq G_2.entities. \end{aligned}$$

Our $\otimes$ constructor differs from the concatenation operation provided in [71]. First, $\otimes$ works on temporal cohesions while the concatenation operation of [71] works on atomic (i.e., elementary) segments. Second, with the concatenation operation, the temporal ordering of the component expressions (i.e., arguments) is preserved. Thus, if E_1 and E_2 are two video expressions, then the concatenation of E_1 and E_2 (written $E_1 \circ E_2$) defines the video expression where E_2 follows E_1. Video expressions are nodes that contain names of children nodes that are either video expressions or raw video segments (which are always the leaf nodes, as in Fig. 6). Raw video segments are created using the name of the raw video and a range within the raw video. Video expressions contain descriptive information about their contents.

Video expressions can be easily captured in our data model by restricting the temporal cohesion notion to a simple video segment.

8.3 Semantics

Our language has a declarative model-theoretic and a fixpoint semantics.

8.3.1 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 temporal cohesion variables. Let $\mathbf{V} = \mathcal{V} \cup \tilde{V}$.

Let var be a countable function that assigns to each syntactical expression a subset of $\mathbf{V}$ 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 13 (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 14 (Extensions). *Given a set $\mathbb{D}_3$ of temporal cohesion objects, the extension of $\mathbb{D}_3$, written $\mathbb{D}_3^{ext}$, 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 15 (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 $\mathbb{D}_\mathcal{I}^{ext}$, is the extension of $\mathbb{D}_\mathcal{I}$, that is, a subset of $\mathbb{D}_1 \cup \mathbb{D}_2 \cup \mathbb{D}_3^{ext}$.*

Definition 16 (Valuation). *A valuation ν is a total function from $\mathbb{V}$ to the set of elements $\mathbb{D}$. This is extended to be the 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 addition, if I_1 and I_2 are temporal cohesion terms, then $\nu(I_1 \otimes I_2) = \nu(I_1) \otimes \nu(I_2)$.*

Definition 17 (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 18 (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 19 (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 $\mathbb{D}_{\cap\chi}^{ext}$ 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 $\mathcal{I}$.□

8.3.2 Fixpoint Semantics

The fixpoint 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 fixpoint 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 temporal cohesions ($\mathbb{D}_3$). A constant temporal cohesion is an element of $\mathbb{D}_3$. We define $\mathbb{D} = \mathbb{D}_1 \cup \mathbb{D}_2 \cup \mathbb{D}_3$.

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

Defintion 20 (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 $\mathbb{D}_\mathcal{I}^{ext}$, 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 21 (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:*

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

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 $\mathbb{D}_{\mathcal{I}_1}^{ext}$, 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 $\mathbb{D}_{\mathcal{I}_2}^{ext}$. 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*

$$T_P\left(\bigcup_i \mathcal{I}_i\right) \subseteq \bigcup_i T_P(\mathcal{I}_i).$$

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 $\mathbb{D}_{\mathbb{I}}^{ext}$ 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 rule r and let $\nu(V)$ be the result of applying ν to each variable in V. $\nu(V)$ is a finite subset of $\mathbb{D}_{\mathbb{I}}^{ext}$ since ν is based on $\mathbb{D}_{\mathbb{I}}^{ext}$. 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,

  $$\nu(var(L_i)) \subseteq \mathbb{D}_{\mathcal{I}_l}^{ext} \forall i \in [1, n].$$

  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 fixpoint 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 fixpoint 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 fixpoint of P and also a minimal model of P, each fixpoint 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 [62]. For a rule language with arithmetic order constraints, the answer to a query can be computed in PTIME data complexity [66]. As a consequence, we obtain a lower bound complexity for query evaluation in our rule based query language.

8.4 Extension to Feature & Content Layer

Our extension to Feature & Content layer is based on the notion of Universal Attachment. Universal attachment [56] is a domain-independent mechanism for integrating diverse representation and reasoning methods into hybrid frameworks that contain a subsystem based on deduction over logical formulas. Predicate evaluation allows a deductive system to exploit external evaluation procedures by "attaching" programs to predicate and function symbols in a first-order language. When a symbol with an attachment is encountered during deduction, the attached program is invoked to calculate the appropriate value. As an example, we could attach the function symbol plus for a first-order language to the LISP function $+$. When the term plus(3,4) is encountered during the deduction process, the LISP program $(+\ 3\ 4)$ would be executed to evaluate it directly. Predicate evaluation has come to be referred to as procedural attachment.

If we can regard a piece of video data as a set of images, then *Query-by-Example* methods developed for images (see, for example, [28]) can be used to retrieve video data by audio-visual content.[7]

Thus, in our framework, similarity predicates will be implemented as foreign functions.

Example. Consider a travel agency which sells stays at resorts. This agency has a database containing both textual information and video clips (compact view) about resorts (such as cities, art galleries, lodging, etc.). Before selling a trip to a customer, he/she is invited to virtually discover his/her tour by referring to information contained in the database. Fig. 7 shows a simple fragment of such a database. For example, the sequence **S1** denotes a particular camp named "Green site," located in "Aachen" in "Germany," etc. Here, the attribute name **sequence** links an abstract object to a concrete object name (here **V1**), which denotes the name of the video sequence associated with the abstract name **S1**, and stored in the Feature & Content Layer.

Given this database, the query:

7. For example, [72] implemented a system which makes retrieval of video data possible by specifying the motion of an object observed in video data by giving an example. An example of an object motion is specified by making a mouse move and then a trajectory and velocity are sampled in accordance with the movement.

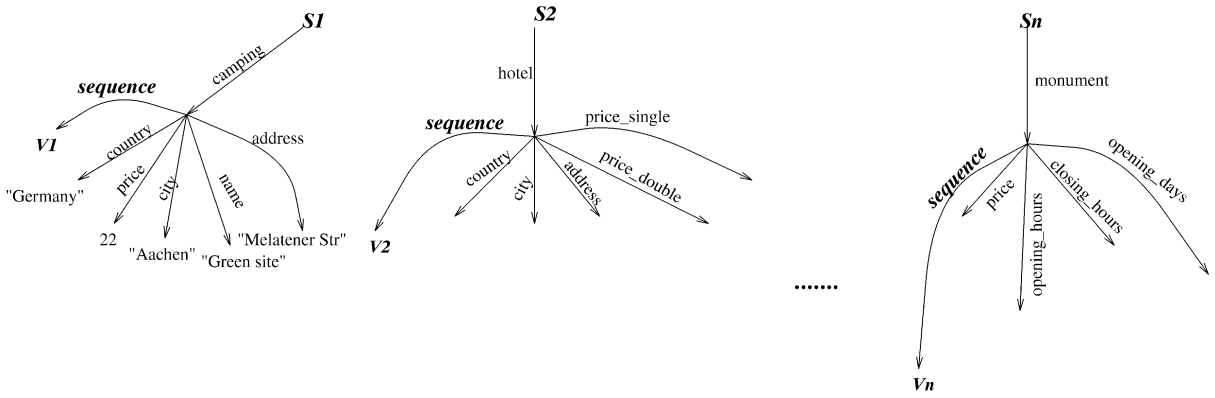

Fig. 7. A traveling database example.

$$Interval(X),\ V =$$
$$X.camping.sequence,\ X.camping.country\ =$$
$$''Germany'',\ X.camping.price < 45,\ \mathbf{same-color}(V, vd)$$

would be "Find a camping trip in Germany, with a price below 45, and the video clip (i.e., the filler of the sequence attribute, here V) of each camping object in the answer set is similar to the video clip vd regarding color." Here, vd is the name of a video clip stored in the Feature & Content Layer. **same-color** is a symbol with an attachment. The attached program will be executed in the Feature & Content Layer.

A query like the previous one is composed of two parts: a conjunctive part and a constraint part. The conjunctive part is evaluated in the Semantic Layer. The constraint part is evaluated in the domain in which the predicates of the constraint part are defined. In the previous example, **same-color** is a call to an external function implemented by a subsystem dealing with color. According to the operational semantics of the language, before the evaluation of the external function all its parameters (here, the variable V) are bound since the evaluation of the conjunctive part takes place first.

One of the assumptions we made is that the behavior of the subsystems implementing the external functions is hidden for the user. In other words, given a value for the variable V, the system will return *yes* if the video sequence denoted by the value of V presents a similarity in color distribution with the video sequence denoted by vd. The required grade for similarity is outside the control of the user and left to the administrator of the database. If we want users to specify the required grade for matching, then it will be used as a parameter of the function "same-color." In the presence of several (multimedia) external functions in a query, the decision for an object to be in the result of the query is taken according to (fuzzy) combining functions as described in [27].

8.5 Expressing Interval Relations

Usually, interval operators are used to specify the temporal constraints among video sequences. For example, the simple expression (drawn from [42]) (polar bear) BEFORE (fox) looks for a logical video segment annotated with polar, followed by a logical video segment annotated with fox. In the framework of the query language we propose, some of the temporal operators can be obtained by means of complex constraints involving $\Rightarrow, \neg, \wedge, \vee$, and others can be implemented thanks to the procedural attachment mechanism.

Example. The query

$$Interval(g), Interval(X),$$
$$X.duration \Rightarrow g.duration$$

would be "Find all temporal cohesions (i.e., logical segments) contained in the temporal cohesion g," and the query

$$Interval(g), Interval(X),$$
$$\neg(X.duration \wedge g.duration)$$

will retrieve all temporal cohesions which do not intersect with the given temporal cohesion g.

In [7], Allen proposes a formalism to represent relations between time intervals that is based on 13 disjoint basic relations on pairs of intervals. These relations correspond to a case analysis of the relative positions of the interval borders. In addition, he presents a consistency test for given sets of relations that is built around a transitive closure algorithm. This algorithm uses a big propagation table which has the following form: Given two basic interval relations $c(i_1, i_2), d(i_2, i_3)$, it says which basic relations could possibly apply to (i_1, i_3).

Recall that, in our framework, a simple interval considered as an ordered pair of real numbers $[a, b]$ is represented by the conjunction of two primitive dense linear order inequality constraints $a \leq t$ and $t \leq b$. Hence, given a temporal cohesion, represented by a complex constraint, we can easily find a set of ordered simple intervals associated with the representation in the form of constraints of our temporal cohesion. For instance, the list of simple intervals associated with the constraint

$$1 \leq t \leq 3 \vee 14 \leq t \leq 18 \vee 7 \leq t \leq 11$$

is $([1, 3], [7, 11], [14, 18])$. So, given such a list (which may include more than two elements), we call first_left the most left element (simple interval) of the list and first_right the most right element of the list. In addition, if X is an interval, then $X.left$ denotes the lower bound of X and $X.right$ denotes the upper bound of X.

Allen's 13 basic relations are binary relations on intervals. Given the above representation of temporal cohesions, we can easily express Allen's basic relations. Let I_1 and I_2 be two temporal cohesions.

I_1 **BEFORE** I_2 iff

$$I_1.\text{first\_right.right} < I_2.\text{first\_left.left},$$

I_1 **AFTER** I_2 iff

$$I_1.\text{first\_left.left} > I_2.\text{first\_right.right}.$$

A similar reasoning (possibly by considering all the elements of the lists) can be used to define the other relations.

So, given such programs defining Allen's interval relations, we can use them when expressing queries. For example, the query:

$$Interval(X), Interval(Y), Object(O), \\ \textbf{Before}(X.duration, Y.duration), \\ O \in X.entities, O \in Y.entities$$

will retrieve all pairs (X, Y) of sequences such that X is before Y in a video document and there exists at least one object O used in the annotation of both sequences. In this example, there is an external program "attached" to the predicate symbol **Before**.

8.6 Classification Rules for Video Data

By their very nature, the data represented in our data model do not come with a conceptual schema. However, adding a rich conceptual model to them is beneficial since it would make them more accessible to users. This is especially important since video databases are often accessed in an "exploratory" or "browse" mode, i.e., users not only query the data to find a particular piece of information, but also pose queries with the goal of having a better understanding of what information is available.

We build on the work of [57] to propose a layer of classes on top of our video data model that is an abstraction of this one. The classes are defined by rules and populated by computing a *greatest fixpoint*.

Clearly, the implicit structure in a particular video data set may be of varying regularity. Indeed, we should not expect, in general, to be able to perfectly type a video data set. The set of a perfect typing may be quite large, e.g., be roughly of the order of the size of the data set, which would prohibit its use and render it impractical, e.g., for graphical query interfaces. Thus, we consider approximate typings [57], i.e., an object does not have to fit its type definition precisely.

In the following, we view our data model, in the style of [15], as a labeled directed graph. The nodes in the graph represent objects and the labels on the edges convey semantic information about the relationships between objects. The sink nodes (nodes without outgoing edges) in the graph represent atomic objects and have values associated with them. The other nodes represent complex objects. In a standard manner [57], we represent the graph using two base relations defined as follows:

link(FromObj, ToObj, Label): This relation contains all the edge information. $link(o_1, o_2, \ell)$ denotes an edge labeled ℓ from object o_1 to o_2.

atomic(Obj, Value): This relation contains all the value information. The fact $atomic(o, v)\}$ corresponds to object o being atomic and having value v.

We also require that 1) each atomic object has exactly one value, i.e., Obj is a key in relation atomic, and 2) each atomic object has no outgoing edges, i.e., the first projections of *link* and *atomic* are disjoint.

As in [57], we consider that a typing is specified by a program of a specific form. The only two extensional relations of the typing program are *link* and *atomic*. The intensional relations are all monadic and correspond to the various types defined by the program. For example, we can consider the following typing program for the database of Fig. 8. The representation in terms of a directed graph associated with this database is given Fig. 9.

$$\begin{aligned} movie(X) \quad &: - \quad link(X, Y, actor)\ \& \\ &\qquad actor(Y) \end{aligned}$$

$$\begin{aligned} actor(X) \quad &: - \quad link(Y, X, actor)\ \& \\ &\qquad movie(Y)\ \& \\ &\qquad link(X, Z, name)\ \& \\ &\qquad atomic(Z, Z') \end{aligned}$$

$$\begin{aligned} producer(X) \quad &: - \quad link(Y, X, producedBy)\ \& \\ &\qquad movie(Y)\ \& \\ &\qquad link(X, Z, name)\& \\ &\qquad atomic(Z, Z') \end{aligned}$$

$$\begin{aligned} director(X) \quad &: - \quad link(Y, X, director)\ \& \\ &\qquad movie(Y)\ \& \\ &\qquad link(X, Z, name)\ \& \\ &\qquad atomic(Z, Z'). \end{aligned}$$

Intuitively, o_5 is a movie, o_7, o_8 are actors, o_{10}, o_{11} are producers, and o_{11} is a director.

The EDBs are *link* and *atomic*. The IDBs are all monadic. Each IDB is defined by a rule of the form:

$$C(X) \ : - \ A_1 \& \ldots \& A_p$$

for some p. Each A_i, called a typed link, has one of the following forms:

1. $link(Y, X, \ell)\ \&\ C'(Y)$
2. $link(X, Y, \ell)\ \&\ C'(Y)$
3. $link(X, Y, \ell)\ \&\ atomic(Y, Z)$,

where ℓ is some constant (a label), X is the variable in the head of the rule, and Y, Z are variables not occurring in any other typed link of the rule.

The semantics of a program P of the form described above for a video database D is defined as the *greatest fixpoint* (see, for example, [8]) of P for D. Let M be an instance over the schema of P such that M coincides with D on {*link*, *atomic*}. Then, M is a fixpoint of P for D if, for each IDB c, $P(M)(c) = M(c)$. It is the *greatest fixpoint* if it contains any other fixpoint of P for D.

link	*FromObj*	*ToObj*	*Label*
	o_1	o_2	video
	o_2	o_3	duration
	o_2	o_4	theme
	...	...	...
	o_1	o_5	video
	o_5	o_6	category
	o_5	o_7	actor
	o_7	o_7'	name
	o_5	o_8	actor
	o_8	o_8'	name
	o_5	o_9	title
	o_5	o_{10}	producedBy
	o_{10}	o_{10}'	name
	o_5	o_{11}	directedBy
	o_{11}	o_{11}'	name
	...	...	...
	o_1	o_{12}	video
	o_{12}	o_{11}	producedBy
	...	...	...
	o_1	o_{13}	video
	o_{13}	o_{14}	newscaster
	o_{14}	o_{14}'	name
	o_{13}	o_{15}	category
	...	...	...

atomic	*Obj*	*Value*
	o_3	'120'
	o_4	"The second world war"
	o_6	"Space exploration"
	o_7'	"Tom Hanks"
	o_8'	"Bill Paxton"
	o_9	"Appolo 13"
	o_{10}'	"Brian Grazer"
	o_{11}'	"Ron Howard"
	o_{14}'	"Jeanne Meserve"
	o_{15}	"News"
	...	...

Fig. 8. Objects and values.

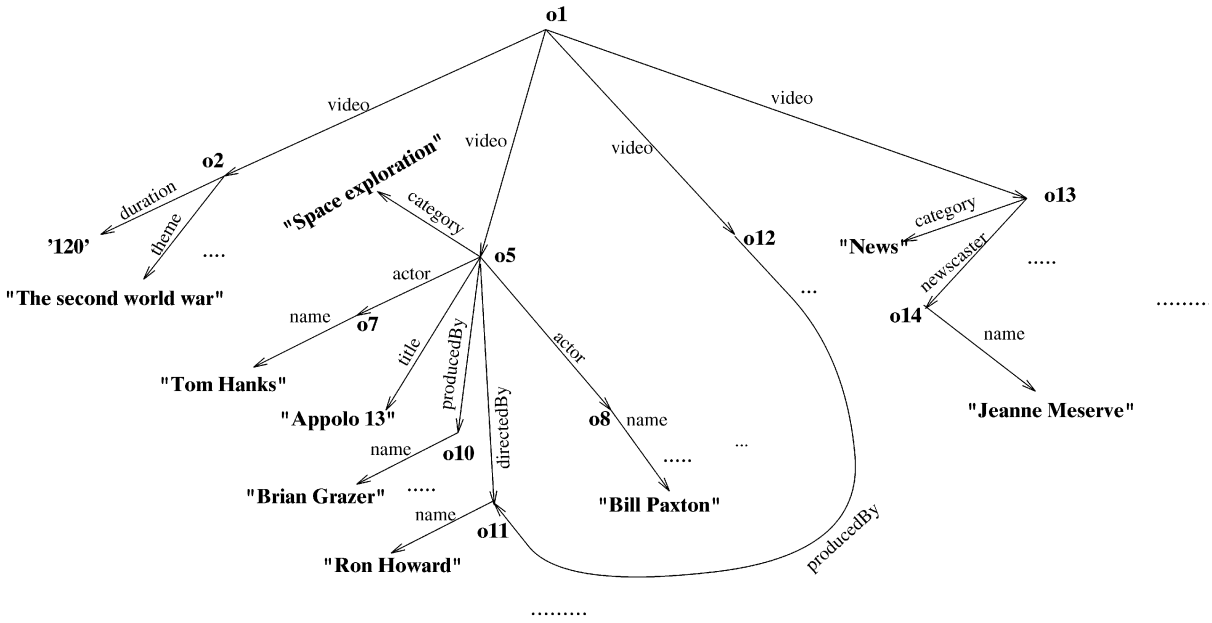

Fig. 9. A fragment of a movie database.

9 Implementation

We have implemented a part of the query language in the form a metainterpreter written in C++ on top of the deductive object-oriented database management system FLORID[8] and the image content based retrieval system QBIC.[9] When designing this implementation, the objective was to provide an idea on how the main aspects of the language can be implemented and architectured. We restricted the language to embed only dense linear order constraints and external functions in the form of calls to QBIC functions. We believe that the extension to set order constraints does not introduce additional difficulty.

FLORID is an implementation of a programming system based on the concepts of F-logic. Proposed by Kifer et al. [46], F-logic is designed as a logical language accounting in

8. F-LOgic Reasoning In Databases.
9. Query By Image Content.

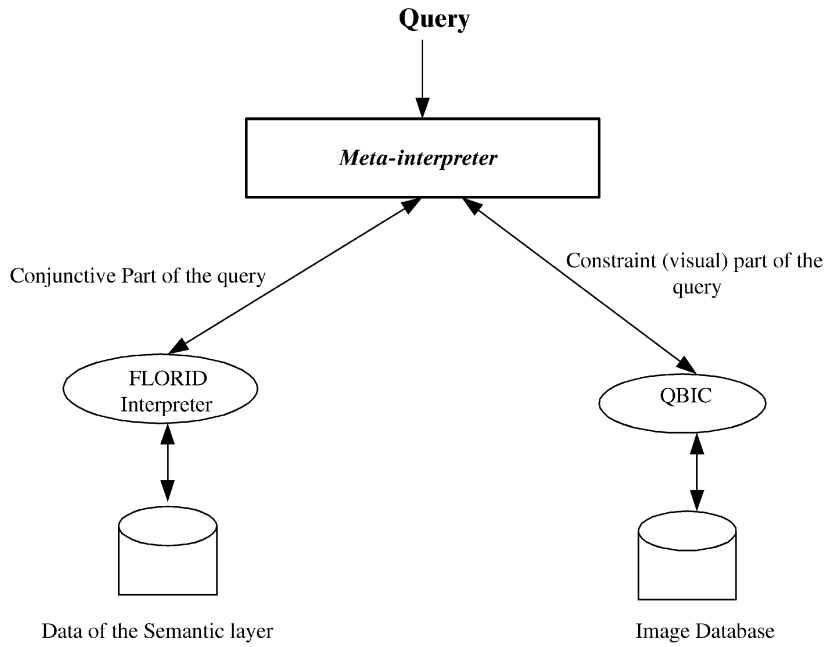

Fig. 10. Architecture of the prototype.

a clean, declarative fashion for most of the structural aspects of object-oriented data modeling. FLORID provides a seminaive evaluation component.

QBIC is an IBM technology that allows to query collections of images by their content. "Query by Content" means one can query a collection of images in order to locate images that are similar to the query image, where similarity can be based on color, texture, or other image properties. QBIC supports commonly used image formats, i.e., bmp, gif, jpg/jpeg, pgm, ppm, tiff/tif, and tga.

QBIC has predefined methods to compute image properties. Each method is implemented as a C++ class which can be invoked from any program. Examples of methods are:

- **QbColorFeatureClass**: computes the average RGB colors for images. Image similarity is based on these three average color values.
- **QbColorHistogramFeatureClass**: computes the color distribution for each image in a predefined 256 color space. Image similarity is based on the similarity of the color distribution.
- etc.

At lower level, QBIC provides a set of abstract C++ classes and a few fully implemented classes to provide QBIC functionality, such as database connection and management, feature data computation, and so on. The QBIC abstract C++ classes are classes that use pure virtual functions to define the interface, forcing the derived classes to specify the implementation. Programs cannot create instances of abstract classes—only pointers to abstract classes are allowed.

The QBIC API is designed to be extensible which means that one can add new feature computation classes to QBIC.

As shown by Fig. 10, the meta-interpreter receives queries from the user. Queries are specified in the form of *structural part & visual part*. The structural part is intended to be evaluated by FLORID and the visual part is intended to be evaluated by QBIC. After splitting the query, the meta-interpreter makes a call to FLORID to execute the structural part on the database containing data of the Semantic Layer. The result of this evaluation is a set of tuples of objects. The result is stored in a file. The file is then analyzed and the results are propagated to the visual part. QBIC is then requested to evaluate the functions of the visual part of the query on the specified images.

10 Conclusion and Future Work

There is growing interest in video databases. As video libraries proliferate, aids to browsing and filtering become increasingly important tools for managing such exponentially growing information resources and for dealing with access problems. One of the central problems in the creation of robust and scalable systems for manipulating video information lies in representing video content. We believe that formal settings will help understanding related modeling and querying problems. This will lead to the development of intelligent systems in order to effectively disseminate, integrate, retrieve, correlate, and visualize 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 1) an abstract representation of the visual appearance of a video able to support modeling and retrieval techniques; 2) a semantic data modeling styled representation of the video content, independent from how the content information is

obtained; 3) a relational representation of the association between objects within a video sequence.

This paper makes the following contributions. 1) We have developed a simple video data model that integrates relations, objects, and constraints. Objects allow us to maintain an object-centered view inherent to video data. Attributes and relations allow us 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 for virtual editing in some applications. The language provides a much more declarative and natural way to express queries. Video database systems require a number of query languages, at different abstraction levels. On one hand, the final user should be enabled to perform "browsing" or "navigation" operations by means of graphical metaphores. On the other hand, the sophisticated user that needs to express more complex queries should be allowed to use a declarative, high-level language, such as a rule language, or an extension of SQL or OQL.

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.

As we said, our video data model has certain similarities to semistructured data models. The schema is absent. In this "unstructured" approach to data representation, each component or object is interpreted dynamically and may be linked to other components in an arbitrary way. In this case, data representation can be viewed as an edge-labeled rooted directed graph and the query language may be thought of as a tree-traversing language. Given such considerations, our work can take advantage of recent developments regarding semistructured data (see, e.g., [15]).

The part of a query that pertains to the Feature & Content Layer (through the mechanism of procedural attachment) is processed by specialized image processing procedures which are time consuming. In addition, the amount of information contained in the object layer is huge. To enable quick response to the queries, the strategy based on the use of materialized[10] views to compute answers to queries can turn out to be useful. Supporting materialized views to process and optimize queries in video databases is a recent work [10]. In the framework we presented here, the combination of techniques for rewriting queries using views (see, for example, [48]) with constraint propagation techniques [45] could constitute a promising approach for query optimization in video databases.

There are many interesting directions to pursue.

1. Due to the visual nature of video data, a user may be interested in results that are similar to the query. Thus, the query system should be able to perform exact as well as partial or fuzzy matching. The first investigations reported in [27], [60] constitute a nice basis.
2. It seems attractive to extend this work such that it can accommodate space and actions inherent to video data. The spatial dimension of video data is a very important aspect that should be considered if we want to provide a satisfactory environment for modeling and querying video data. Independently of video databases, some researchers tackled the problem of modeling and querying moving objects [65]. Modeling of moving objects in a video database was also considered [49]. The authors proposed a representation which supports a set of spatial topological and directional relations. We are working on a way to declaratively specify the trajectory of a single object and the relative spatial relation between multiple moving objects.
3. Another important direction is to study the problem of sequence presentation. Most existing research systems use a template-based approach [4] to provide automatic sequencing capability. With this approach, a set of sequencing templates is predefined to confine the user's exploration to a certain sequencing order. However, 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 [61] connected to our query language will offer more possibilities and flexibility in the specification of sequence presentations.
4. In order to exploit advances in computer vision allowing automatic extraction of shots/scenes from video sequences, a clustering technique could be used to group together, on the basis of semantic content, semantically related shots/scenes forming what we called temporal cohesion.

10. A materialized view is a query where a physical copy of each instance, answers to the query, is stored and maintained.

ACKNOWLEDGMENTS

The authors wish to thank Michel Martinez, Jean-Marc Petit, Behzad Shariat, and Farouk Toumani for their remarks.

REFERENCES

[1] S. Abiteboul, "Querying Semi-Structured Data," *Proc. 1997 Int'l Conf. Database Theory (ICDT '97),* pp. 1–18, Jan. 1997.
[2] 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 System,* vol. 4, pp. 172–186, 1996.
[3] T.G. Aguierre-Smith and G. Davenport, "The Stratification System: A Design Environment for Random Access Video," *Proc. Third Int'l Workshop Network and Operating System Support for Digital Audio and Video,* Nov. 1992.
[4] T.G. Aguierre-Smith and N. C. Pincever, "Parsing Movies in Context," *Proc. Summer 1991 Usenix Conf.,* pp. 157–168, June 1991.
[5] G. Ahanger, D. Benson, and T. Little, "Video Query Formulation," *Proc. Int'l Soc. Optical Eng., Storage and Retrieval for Image and Video Database III (SPIE '95),* W. Niblack and R.C. Jain, eds., pp. 280– 291, Feb. 1995.
[6] A. Aiken and E.L. Wimmers, "Solving Systems of Set Constraints (extended abstract)," *Proc. Seventh Ann. IEEE Symp. Logic in Computer Science,* I.C.S. Press, ed., pp. 329–340, 1992.

[7] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. ACM,* vol. 26, no. 11, pp. 832–843, 1983.
[8] K.R. Apt, *Logic Programming, Handbook of Theoretical Computer Science.* Elsevier, 1991.
[9] E. Ardizzone and M. L. Cascia, "Automatic Video Database Indexing and Retrieval," *Multimedia Tools and Applications,* vol. 4, no. 1, pp. 29–56, Jan. 1997.
[10] E. Ardizzone and M.-S. Hacid, "A Knowledge Representation and Reasoning Support for Video Data," *Proc. 11th IEEE Int'l Conf. Tools with Artificial Intelligence (ICTAI '99),* Nov. 1999.
[11] F. Baader and K.U. Schulz, "On the Combination of Symbolic Constraints, Solution Domains, and Constraint Solvers," *Proc. Int'l Conf. Principles and Practice of Constraint Programming (CP '95),* vol. 976, pp. 380–397, 1995.
[12] A.D. Bimbo, *Visual Information Retrieval.* Morgan Kaufmann, 1999.
[13] A.D. Bimbo, M. Campanai, and P. Nesi, "A Three-Dimensional Iconic Environment for Image Database Querying," *IEEE Trans. Software Eng.,* vol. 19, no. 1, pp. 997–1,011, Jan. 1993.
[14] A. Brodsky and Y. Kornatzky, "The *lyric* Language: Querying Constraint Objects," *Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '95),* pp. 35–46, May 1995.
[15] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu, "A Query Language and Optimization Techniques for Unstructured Data," *Proc. ACM SIGMOD Int'l Conf. (SIGMOD '96),* pp. 505–516, June 1996.
[16] N. Chang and K. Fu, "Picture Query Languages for Pictorial Database Systems," *Computer,* vol. 14, no. 11, pp. 23–33, Nov. 1981.
[17] J. Chomicki and T. Imielinski, "Relational Specifications of Infinite Query Answers," *ACM Trans. Database Systems,* vol. 18, no. 2, pp. 181–223, June 1993.
[18] T.-S. Chua and L.-Q. Ruan, "A Video Retrieval and Sequencing System," *ACM Trans. Information Systems (TOIS),* vol. 13, no. 4, pp. 373–407, Oct. 1995.
[19] A. Colmerauer, "An Introduction to PROLOG III," *Comm. ACM,* vol. 33, pp. 69–90, 1990.
[20] J.M. Corridoni and A.D. Bimbo, "Structured Digital Video Indexing," *Proc. 13th Int'l Conf. Pattern Recognition—ICPR '96,* 1996.
[21] J.M. Corridoni, A.D. Bimbo, D. Lucarella, and H. Wenxue, "Multi-perspective Navigation of Movies," *J. Visual Languages and Computing,* vol. 7, pp. 445–466, 1996.
[22] M. Davis, "Media Streams: An Iconic Visual Language for Video Annotation," *Proc. IEEE 1993 Symp. Visual Languages,* pp. 196–203, Aug. 1993.
[23] C. Decleir, M.-S. Hacid, and J. Kouloumdjian, "A Database Approach for Modeling and Querying Video Data," *Proc. 15th Int'l Conf. Data Eng., (ICDE '99),* Mar. 1999.
[24] N. Dimitrova, "The Myth of Semantic Video Retrieval," *ACM Computing Surveys,* vol. 27, no. 4, pp. 584–586, Dec. 1995.
[25] A. Duda, R. Weiss, and D.K. Gifford, "Content-Based Access to Algebraic Video," technical report, MIT Laboratory for Computer Science 1994.
[26] A.K. Elmagarmid and H. Jiang, *Video Database System: Issues, Products, and Applications.* Kluwer, 1997.
[27] R. Fagin, "Fuzzy Queries in Multimedia Database Systems," *Proc. 1998 ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS '98),* J. Paredaens, ed., pp. 1–10, 1998.
[28] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System," *Intelligent Multimedia Information Retrieval,* M.T. Maybury, ed., chapter 1, pp. 7–22, 1996.
[29] D.L. Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Comm. ACM,* 1991.
[30] S. Gibbs, C. Breiteneder, and D. Tsichritzis, "Audio/Video Databases: An Object-Oriented Approach," *Proc. Ninth IEEE Int'l Data Eng. Conf. (ICDE '93),* pp. 381–390, 1993.
[31] S. Gibbs, C. Breiteneder, and D. Tsichritzis, "Data Modeling of Time-Based Media," *Proc. 1994 ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '94),* pp. 91–102, 1994.
[32] H.M. Gladney, N.J. Belkin, Z. Ahmed, E.A. Fox, R. Ashany, and M. Zemankova, "Digital Library: Gross Structure and Requirements (Report from a Workshop)," Technical Report RJ 9840, IBM Research Report, May 1994.
[33] S. Grumbach and J. Su, "Dense-Order Constraint Databases," *Proc. 1995 Symp. Principles of Database Systems (PODS '95),* pp. 66–77, June 1995.
[34] S. Grumbach, J. Su, and C. Tollu, "Linear Constraint Query Languages Expressive Power and Complexity," *Proc. Int'l Workshop Logic and Computational Complexity (LCC '94),* D. Leivant, ed., pp. 426–446, Oct. 1994.
[35] B.J. Haan, P. Kahn, V.A. Riley, J.H. Coombs, and N.K. Meyrowitz, "IRIS Hypermedia Services," *Comm. ACM,* vol. 35, no. 1, pp. 35–51, Jan. 1991.
[36] H. Hampapur, R. Jain, and T.E. Weymouth, "Feature Based Digital Video Indexing," technical report, Univ. of Michigan, 1994.
[37] A.G. Hauptmann and M.A. Smith, "Text, Speech, and Vision for Video Segmentation: The Informedia Project," technical report, Carnegie Mellon Univ., 1995.
[38] K. Hirata and T. Kato, "Query by Visual Example," *Proc. Third Int'l Conf. Extended Database Technology (EDBT '92),* pp. 56–71, Mar. 1992.
[39] R. Hjelsvold and R. Midtstraum, "Modeling and Querying Video Data," *Proc. 20th Int'l Conf. Very Large Databases (VLDB '94),* pp. 686–694, 1994.
[40] M. Höhfeld and G. Smolka, "Definite Relations over Constraint Languages," LILOG Report 53, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, Germany, Oct. 1988. http://www.dfki.de/dfki/research.html.
[41] L. Huang, J.C.-M. Lee, Q. Li, and W. Xiong, "An Experimental Video Database Management System Based on Advanced Object-Oriented Techniques," *Proc. Int'l Soc. Optical Eng. Storage and Retrieval for Image and Video Database IV (SPIE '96),* K. Sethi and R.C. Jain, ed., pp. 158– 169, Feb. 1996.
[42] H. Jiang and A.K. Elmagarmid, "WVTDB—A Semantic Content-Based Video Database System on the World Wide Web," *IEEE Trans. Knowledge and Data Eng.,* vol. 10, no. 6, pp. 947–966, Nov./Dec. 1998.
[43] T. Joseph and A. Cardenas, "PICQUERY: A High Level Query Language for Pictorial Database Management," *IEEE Trans. Software Eng.,* vol. 14, no. 5, pp. 630–638, May 1988.
[44] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz, "Constraint Query Languages," *J. Computer and System Sciences,* vol. 51, no. 1, pp. 26–52, 1995.
[45] D.B. Kemp and P.J. Stuckey, "Analysis Based Constraint Query Optimization," *Proc. 10th Int'l Conf. Logic Programming,* D.S. Warren, ed., pp. 666–682, June 1993.
[46] M. Kifer, G. Lausen, and J. Wu, "Logical Foundations of Object-Oriented and Frame-Based Languages," *J. ACM,* vol. 42, no. 4, pp. 741–843, July 1995.
[47] M. Kifer and J. Wu, "A Logic for Object-Oriented Logic Programming (Maier's O-Logic Revisited)," *Proc. 1989 Symp. Principles of Database Systems (PODS '89),* pp. 379–393, Mar. 1989.
[48] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, and D. Srivastava, "Answering Queries Using Views," *Proc. 14th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS '95),* pp. 95–104, May 1995.
[49] J.Z. Li, M.T. Ozsu, and D. Szafron, "Modeling of Moving Objects in a Video Database," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems,* pp. 336–343, June 1997.
[50] W. Li, S. Gauch, J. Gauch, and K.M. Pua, "VISION: A Digital Video Library," *ACM Digital Libraries,* 1996.
[51] T.D.C. Little, G. Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh, "A Digital On-Demand Video Service Supporting Content-Based Queries," *Proc. First ACM Int'l Conf. Multimedia,* pp. 427–436, Aug. 1993.
[52] W.E. Mackay and G. Davenport, "Virtual Video Editing in Interactive Multimedia Applications," *Comm. ACM,* vol. 32, no. 7, pp. 802–810, July 1989.
[53] S. Marcus and V.S. Subrahmanian, "Foundations of Multimedia Database Systems," *J. ACM,* vol. 43, no. 3, pp. 474–523, 1996.
[54] G. Mecca and A.J. Bonner, "Sequences, Datalog and Transducers," *Proc. 1995 Symp. Principles of Database Systems (PODS '95),* pp. 23–35, May 1995.
[55] C. Meghini, "Towards a Logical Reconstruction of Image Retrieval," *Proc. In't Soc. Optical Eng., Storage and Retrieval for Image and Video Database IV (SPIE '96),* I.K. Sethi and R.C. Jain, eds., pp. 108–119, Feb. 1996.
[56] K.L. Myers, "Hybrid Reasoning Using Universal Attachment," *Artificial Intelligence,* vol. 67, pp. 329–375, 1994.
[57] S. Nestorov, S. Abiteboul, and R. Motwani, "Extracting Schema from Semistructured Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '98),* L.M. Haas and A. Tiwary, eds., pp. 295–306, June 1998.

[58] 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" *IS&T/SPIE Symp. Electronic Imaging; Science and Technology,* Feb. 1993.
[59] E. Oomoto and K. Tanaka, "OVID: Design and Implementation of a Video-Object Database System," *IEEE Trans. Knowledge and Data Eng.,* vol. 5, no. 4, pp. 629–643, Aug. 1993.
[60] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T.S. Huang, "Supporting Similarity Queries in MARS," *Proc. Fifth ACM Int'l Multimedia Conf.,* pp. 403–413, Nov. 1997.
[61] J. Paredaens, P. Peelman, and L. Tanca, "G-Log: A Declarative Graphical Query Languages," *Proc. Second Int'l Conf. Deductive and Object-Oriented Databases (DOOD '91),* C. Delobel, M. Kifer, and Y. Masunaga, eds., pp. 108–128, Dec. 1991.
[62] P.Z. Revesz, "Datalog Queries of Set Constraint Databases," *Proc. Fifth Int'l Conf. Database Theory (ICDT '95),* G. Gottlob and M.Y. Vardi, eds., pp. 425–438, Jan. 1995.
[63] W.C. Rounds, "Set Values for Unification Based Grammar Formalisms and Logic Programming," Research Report CSLI-88-129, Stanford Univ., 1988.
[64] L.A. Rowe, J.S. Boreczky, and C.A. Eads, "Indexes for User Access to Large Video Databases," *Proc. Storage and Retrieval of Image and Video Databases II,* 1995.
[65] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," *Proc. 13th Int'l Conf. Data Eng., (ICDE '97),* A. Gray and P.-A. Larson, eds., pp. 422–432, IEEE CS Soc., 1997.
[66] D. Srivastava, R. Ramakrishnan, and P.Z. Revesz, "Constraint Objects," *Proc. Second Int'l Workshop Principles and Practice of Constraint Programming (PPCP '94),* pp. 218–228, 1994.
[67] D. Swanberg, C.-F. Shu, and R. Jain, "Knowledge Guided Parsing in Video Databases," *Proc. Image and Video Processing Conf.; Symp. Electronic Imaging: Science & Technology,* vol. 1,908, pp. 13–24, Feb. 1993.
[68] Special Issue in Video Information Systems, *ACM Trans. Information Systems,* vol. 13, no. 4, Oct. 1995.
[69] D. Toman, "Point vs. Interval-Based Query Languages for Temporal Databases," *Proc. 1996 Symp. Principles of Database Systems (PODS '96),* pp. 58–67, June 1996.
[70] Y. Tonomura, "Video Handling Based on Structured Information for Hypermedia Systems," *Proc. Int'l Conf. Multimedia Information Systems '91,* pp. 333–344, 1991.
[71] R. Weiss, A. Duda, and D. K. Gifford, "Content-Based Access to Algebraic Video," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems,* pp. 140–151, May 1994.
[72] A. Yoshitaka, Y. Hosoda, M. Yoshimitsu, M. Hirakawa, and T. Ichikawa, "VIOLONE: Video Retrieval by Motion Example," *J. Visual Languages and Computing,* vol. 7, pp. 423–443, 1996.
[73] A. Zhang and S. Multani, "Implementation of Video Presentation in Database Systems," *Proc. Int'l Soc. Optical Eng., Storage and Retrieval for Image and Video Database IV (SPIE '96),* I.K. Sethi and R.C. Jain, eds., pp. 228– 238, Feb. 1996.

Mohand-Saïd Hacid graduated as an engineer in computer science from the University of Tizi-Ouzou, Algeria, in 1987, and received his PhD degree in computer science from the National Institute of Applied Sciences, Lyon, France, in 1991. He is currently an associate professor at the University Claude Bernard Lyon 1, Lyon, France. He has been a visiting researcher at the Theoretical Computer Science Laboratory, Aachen University of Technology, Germany, and at the Indiana Center for Database Systems, Purdue University, West Layfayette, Indiana. His research interests include knowledge representation and reasoning, data models and query languages for multimedia databases, semistructured databases, and multidimensional databases. He is a member of the IEEE Computer Society and ACM.

Cyril Decleir graduated as an engineer in computer science from the University of Marseille, France, in 1996, and received his PhD degree in computer science from the National Institute of Applied Sciences, Lyon, France, in 1999.

Jacques Kouloumdjian graduated as an engineer from the Ecole Centrale de Lyon, France, and received his PhD degree in nuclear physics in 1968. In 1970, he became a professor of computer science at the University Claude Bernard Lyon 1 and, in 1988, he joined the National Institute of Applied Sciences, Lyon, France. His research interests include deductive databases, multimedia databases, and database reverse-engineering. He is a member of the IEEE Computer Society and ACM.