Modeling and Querying Video Databases*

Cyril Decleir
LISI-INSA Lyon, Bâtiment 501
F-69621 Villeurbanne Cedex
cdecleir@lisi.insa-lyon.fr

Mohand-Saïd Hacid
LuFG Theoretical Computer Science
RWTH Aachen,
Ahornstraße 55, 52074 Aachen, Germany
hacid@cantor.informatik.rwth-aachen.de

Jacques Kouloumdjian
LISI-INSA Lyon, Bâtiment 501
F-69621 Villeurbanne Cedex
koulou@lisi.insa-lyon.fr

Abstract

Indexing video data is essential for providing content based access. This paper develops a data model and a rule-based query language for video content based indexing and retrieval. The data model is based on the notion of generalized strata, which can be seen as a set of intervals. Each interval can be analyzed to extract 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) in the domain of a video sequence, (2) video frames, called generalized strata, which contain these entities. To represent these information, our data model allows facts as well as objects and constraints. We present a declarative, rule-based, constraint query language that can be used to infer relationships about information represented in the model. In the video applications we are interested in, we wish to construct new generalized strata from old ones. To do this, our language has an interpreted function term (i.e., constructive term) to concatenate generalized strata. The language has a clear declarative and operational semantics.

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

1 Introduction

As a consequence of increased capabilities of computer software[1] and hardware, it is nowadays possible to store common human media, such as pictures, sounds, and more recently video streams. Nevertheless, storing is the primary but not the only service we are to expect from a computer. Its role should also concern *content indexing* and *retrieval*. Two main approaches are used:

1. Fully automated content indexing approach. Some fully automated research systems have been developed, among others, *VIOLONE* [22] and *JACOB* [13]. However, because of the weakness of content analysis algorithms, they focus on a very specific exploitation.

2. Computer aided content indexing approach (i.e., indices are provided by users (supported by some tools) by analyzing video content). Systems based on this approach are, among others, *OVID* [17], *AVIS* [1], and *VideoStar* [10].

A *database support* for video information will help *sharing* information among applications and make it available for analysis. Therefore, new technologies and techniques are required for organizing, storing, manipulating and retrieving by content video data. Although various 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

*This work is partially supported by France Télécom (through CNET/CCETT), research contract no. 96 ME 17.

[1]The combination of powerful compression techniques, such as *MPEG-I* [6] [7] and much recently *MPEG-II*, with mass-storage devices, made computers able to support video data.

set of pre-defined, ad hoc operators, often incorporated to SQL, and are not investigated in theoretical framework.

From *database point of view*, video data presents an interesting challenge in the development of data models and query languages. For example, the data model should be expressive enough to capture several characteristics inherent to video data, such as movements, shapes, variations, etc. The query language should allow some kind of reasoning, to allow, for example, virtual editing [14].

Despite the consensus of the central role video databases will play in the future, there is little research work on finding semantic foundations for representing and querying video information. This paper is a contribution in this direction. The framework presented here integrates formalisms developed in video, constraint object and sequence databases. The paper builds on the works of [1, 15, 10, 17, 20, 5, 16, 8] to propose a data model for video databases and a declarative, rule-based, constraint query language, that has a clear declarative and operational semantics. We make the following contributions:

1. We develop a simple and useful video data model on the basis of relation, object and constraint paradigms. We do not offer a fixed set of attributes, but users can freely describe and retrieve video scenes according to their viewpoints.

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

The model and the query language use the point-based approach to represent periods of time associated with generalized strata. First-order queries can then be conveniently asked in a much more declarative and natural way [21]. To our knowledge, this is the first proposal of a formal rule-based query language for querying video data.

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

2 Video Data Model: An Overview

In this section, we informally introduce our video indexing model.

Video indexing means that some kind of information is retrieved from a video document and stored in a database in order to be queried later. The model proposed here is designed in such a way that it allows to represent heterogeneous information describing video content.

An important problem that must be considered in video indexing is the *temporal* nature of the information. To deal with this problem, two approaches have been developed: *segmentation* and *stratification*. The segmentation (figure 1) consists in partitioning the video document time-line into interesting segments and annotating each segment according to what is occurring in the corresponding time-frame. This approach has been criticized mainly because it is not possible to extract a single information and precisely define its time boundaries. This fact leads Aguierre-Smith [2] to define the *stratification* (figure 2) approach. This approach allows to annotate interesting information by assigning it to a strata. A strata is a combination of a time interval and an annotation.

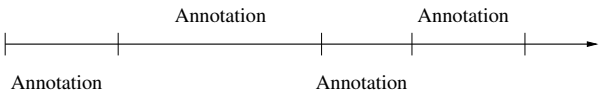

Figure 1. Segmentation

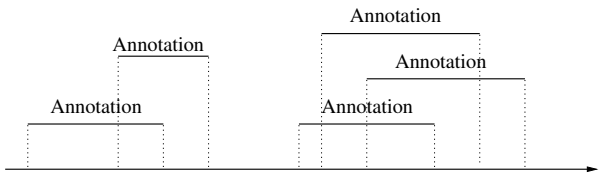

Figure 2. Stratification

Our proposal extends the stratification approach by assigning several (disjunction of) time intervals to an annotation. Basically, the model we propose makes use of two entities: (1) object of interest, which denotes information extracted from the video document. (2) generalized strata which is used to capture the temporal aspect of video data. A *generalized strata* is a set of time intervals which serves as temporal support for *objects of interest*.

2.1 Example

To illustrate the underlying paradigms of our model, we will use a French TV news broadcast from channel *France 2* of July, 13, 1996. In this document, we focus on a sequence dealing with a stay of Nelson Mandela in France. This sequence comprises three main parts: after being introduced by broadcast speaker, the sequence starts with a meeting between Presidents Chirac and Mandela at the Rambouil-

let's Castle, in France. Then, a brief recall of Mandela's life from his trial to his election as South Africa's President is shown. In conclusion, the sequence ends with an extract in the Rambouillet's castle dealing with difficulties occurring in South Africa after apartheid's end. This report is presented in a segmentation-style in figure 3.

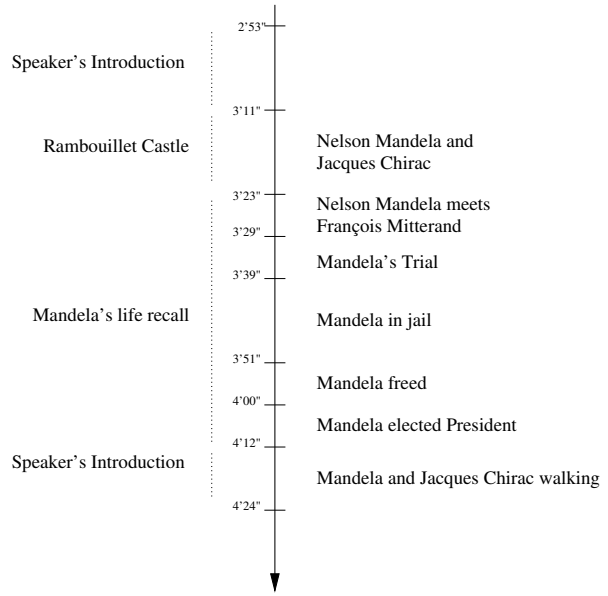

Figure 3. Report overview

To index this report, we consider three main entities: Nelson Mandela, Jacques Chirac and the TV speaker.

$o_1 = (id_1, [name : \text{"Nelson Mandela"}, status : \text{"South Africa President"}])$
$o_2 = (id_2, [name : \text{"Jacques Chirac"}, status : \text{"French President"}, wearing : \text{"Suit"})$
$o_3 = (id_3, [name : \text{"Bruno Masure"}, status : \text{"TV Speaker"}, works_at : \text{"France 2"}, wearing : \text{"Shirt"}])$

Now, let us see how these objects of interest are related to generalized strata. Consider three generalized strata (figure 4):

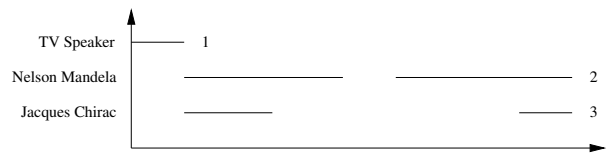

Figure 4. Generalized strata

$i_1 = (id_6, [timeframe : (t > 2.53 \wedge t < 3.11)])$
$i_2 = (id_7, [timeframe : (t > 3.11 \wedge t < 3.23) \vee (t > 3.51 \wedge t < 4.24)])$
$i_3 = (id_8, [timeframe : (t > 3.11 \wedge t < 3.23) \vee (t > 4.12 \wedge t < 4.24)])$

where t is a temporal variable.

To link objects of interest to generalized strata, we simply extend the description of generalized strata.
$i_1 = (id_6, [entities : o_3, timeframe : (t > 2.54 \wedge t < 3.10)])$
$i_2 = (id_7, [entities : o_1, timeframe : (t > 3.11 \wedge t < 3.34) \vee (t > 3.51 \wedge t < 4.24)])$
$i_3 = (id_8, [entities : o_2, timeframe : (t > 3.11 \wedge t < 3.34) \vee (t > 4.12 \wedge t < 4.24)])$

Besides attributes, we allow relations to be defined among entities.

3 Basic Definitions

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

Definition 1 (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, $=$, $\leq$ and $<$ are interpreted respectively as elements, equality, the dense order, and the irreflexive dense order of $\mathcal{D}$.*

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 [9].

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 [4], involving $\in$, $\subseteq$, and $\supseteq$, but no set functions such as $\cup$ and $\cap$.

Satisfaction and entailment of conjunctions of set-order constraints can be solved in polynomial-time using a quantifier elimination algorithm given in [20].

Definition 3 (Interval) *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 $\mathbf{R}$. 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 (Generalized strata) *A generalized strata is a set of pairwise non overlapping intervals. Formally, a generalized strata can be represented as a disjunction of intervals.*

4 Rule-Based, Constraint Query Language

In this section, we present the declarative, rule-based query language that can be used to reason with facts and objects in our video data model. The language consists of two constraint languages[2] on top of which relations can be defined by means of definite clauses.

This language has a model-theoretic and fix-point semantics based on the notion of *extended active domain* of a database. The extended domain contains all generalized strata objects and their concatenations. The language has an interpreted function symbol for building new generalized strata (fragments of sequences) from others (by concatenating them) using the operator $\otimes$.

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

4.1 Syntax

To manipulate generalized strata, our language has an interpreted function symbol for constructing[3] complex term. Intuitively, if I_1 and I_2 are generalized strata, then $I_1 \otimes I_2$ denotes the concatenation of I_1 and I_2.

The language of terms uses three countable, disjoint sets:

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

If I_1 and I_2 are generalized strata objects, generalized strata 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 [12] 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 concatenation of I with I_1 or I_2 has as result I. This leads to the termination of the execution of constructive rules (see below the definition of constructive rule).

Definition 5 (Predicate symbol) *We define the following predicate symbols:*

- *let $\mathcal{R}$ be the set of relations of $\mathbb{D}_2^* \times \mathbb{D}_3$. Each $\mathrm{P} \in \mathcal{R}$ with arity n is associated with a predicate symbol P of arity n.*
- *a special unary predicate symbol $AnyInterval$. It can be seen as the class of all generalized strata objects.*
- *a special unary predicate symbol $AnyObject$. It can be seen as the class of all objects other than generalized strata objects.*

Definition 6 (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.*

Definition 7 (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.

[2] For a formal definition of a constraint language, see [11].
[3] for concatenating generalized strata.

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

Definition 8 (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 9 (Program) *A program is a collection of range-restricted rules.*

Definition 10 (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.

4.2 Semantics

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

4.2.1 Model-theoretic Semantics

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

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

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

Definition 11 (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 12 (Valuation) *A valuation ν is a total function from $\mathbb{V}$ to the set of elements $\mathbb{D}$. This is extended to be identity on $\mathbb{D}$ and then extended to map free tuples to tuples in the natural fashion. ν is extended to constraints in a straightforward way. In addition, if I_1 and I_2 are generalized strata terms, then $\nu(I_1 \otimes I_2) = \nu(I_1) \otimes \nu(I_2)$.*

Definition 13 (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 14 (Model of a Program) *Consider a program P. An interpretation $\mathcal{I}$ is said to be a model of a program 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 15 (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 $\mathcal{I}$.*

4.2.2 Fix-point Semantics

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

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

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

Definition 16 (Extensions) *Given a set $\mathbb{D}_3$ of generalized strata 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 17 (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 $\mathbb{D}_1 \cup \mathbb{D}_2 \cup \mathbb{D}_3^{ext}$.*

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

Definition 18 (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 19 (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:*

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

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

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

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

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

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

For Datalog with set order constraints, queries are shown to be evaluable bottom-up in closed form and to have DEXPTIME-complete data complexity [19]. For a rule language with arithmetic order constraints, the answer to a query can be computed in PTIME data complexity [20]. In the case of our language, in the presence of a constructive term combined with the use of two classes of constraints: (1) a restricted form of set constraints and (2) dense linear order inequality constraints, the computational complexity need to be considered in depth. This will be part of future work.

5 Conclusion and Future Work

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

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

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

There is an interesting problem we intend to pursue. It consists in studying the problem of sequence presentation. Most existing research systems use template-based approach [3] to provide the automatic sequencing capability. In this approach, a set of sequencing templates is predefined to confine the user's exploration to a certain sequencing order. The problem is that this approach is domain-dependent and relies on the availability of a suitable template for a particular query. We believe that a framework based on declarative graphical languages [18] will offer more possibilities and flexibility in sequence presentation. We are investigating this important research direction.

References

[1] S. Adali, K. S. Candan, S.-S. Chen, K. Erol, and V. S. Subrahmanian. Advanced video information system: Data structures and query processing. *ACM-Springer Multimedia System*, 1995.

[2] T. G. Aguierre Smith. Stratification : Toward a computer representation of the moving image. Technical report, The Media Lab, M.I.T, 1991.

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

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

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

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

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

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

[9] S. Grumbach, J. Su, and C. Tollu. Linear constraint query languages expressive power and complexity. In D. Leivant, editor, *Proceedings of the International Workshop on Logic and Computational Complexity (LCC'94), Indianapolis, IN, USA*, pages 426–446, Oct. 1994.

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

[11] 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.

[12] M. Kifer and J. Wu. A logic for object-oriented logic programming (Maier's o-logic revisited). In *Proceedings of the 1989 Symposium on Principles of Database Systems (PODS'89), Philadelphia, Pennsylvania*, pages 379–393, Mar. 1989.

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

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

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

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

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

[18] J. Paredaens, P. Peelman, and L. Tanca. G-log: A declarative graphical query languages. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases (DOOD'91), Munich, Germany*, volume 566 of *LNCS*, pages 108–128, Dec. 1991.

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

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

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

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