AN INTELLIGENT FUZZY OBJECT-ORIENTED DATABASE FRAMEWORK FOR VIDEO DATABASE APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

NEZİHE BURCU ÖZGÜR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2007

Approval of the thesis:

AN INTELLIGENT FUZZY OBJECT-ORIENTED DATABASE FRAMEWORK FOR VIDEO DATABASE APPLICATIONS

submitted by **NEZİHE BURCU ÖZGÜR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Volkan Atalay _____________________
Head of Department, **Computer Engineering**

Prof. Dr. Adnan Yazıcı _____________________
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu _____________________
Computer Engineering Dept., METU

Prof. Dr. Adnan Yazıcı _____________________
Computer Engineering Dept., METU

Assoc. Prof. Dr. Gözde Bozdağı Akar _____________________
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Ahmet Coşar _____________________
Computer Engineering Dept., METU

Asst. Prof. Dr. Murat Koyuncu _____________________
Computer Engineering Dept., Atılım University

Date: 21.09.2007

ABSTRACT

AN INTELLIGENT FUZZY OBJECT-ORIENTED DATABASE FRAMEWORK FOR VIDEO DATABASE APPLICATIONS

Özgür, Nezihe Burcu
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Adnan YAZICI

September 2007, 143 pages

Video database applications call for flexible and powerful modeling and querying facilities, which require an integration or interaction between database and knowledge base technologies. It is also necessary for many real life video database applications to incorporate uncertainty, which naturally occurs due to the complex and subjective semantic content of video data. In this thesis study, firstly, a fuzzy conceptual data model is introduced to represent the semantic content of video data. UML (Unified Modeling Language) is utilized and extended to represent uncertain information along with video specific properties at the conceptual level. Secondly, an intelligent fuzzy object-oriented database framework is presented for video database applications. The introduced fuzzy conceptual model is mapped to the presented framework, which is an adaptation of the previously proposed IFOOD architecture. The framework provides modeling and querying of complex and rich semantic content and knowledge of video data including uncertainty. Moreover, it allows (fuzzy) semantic, temporal, (fuzzy) spatial, hierarchical, regional and trajectory queries, based on the video data model. We think that the presented conceptual data model and framework can be adapted to any application domain related to video databases.

Keywords: Conceptual data model, video data, uncertainty, fuzziness, UML, object-oriented modeling, video database applications, object-oriented databases, knowledge base systems, intelligent systems.

ÖZ

VİDEO VERİTABANI UYGULAMALARI İÇİN AKILLI, BULANIK VE NESNEYE DAYALI BİR VERİTABANI SİSTEMİ

Özgür, Nezihe Burcu
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Adnan YAZICI

Eylül 2007, 143 sayfa

Video veritabanı uygulamaları, veritabanı ve bilgi tabanı teknolojilerinin etkileşimini veya entegrasyonunu gerektiren esnek ve güçlü modelleme ve sorgulama özelliklerine ihtiyaç duymaktadır. Video verisinin karmaşık ve subjektif anlamsal içeriği sebebiyle doğal olarak oluşan belirsizliği de ele almak, gerçek yaşamdaki pek çok video veritabanı uygulaması için gereklidir. Bu tez çalışmasında, öncelikle video verisinin anlamsal içeriğinin gösterilmesi için bulanık, kavramsal bir veri modeli geliştirilmiştir. Video verisine ait özellikleri bulanık bilgi ile birlikte kavramsal düzeyde gösterebilmek için UML modeli genişletilmiştir. İkinci olarak, video veritabanı uygulamaları için akıllı, bulanık ve nesneye dayalı bir veritabanı sistemi geliştirilmiştir. Daha önceden geliştirilmiş olan IFOOD mimarisinin bir uyarlaması olan bu sistem üzerinde bulanık kavramsal video modelimiz kullanılmıştır. Sistem, belirsizlik içeren, karmaşık ve zengin video anlamsal içeriğinin ve bilgisinin modellenmesini ve sorgulanmasını sağlamaktadır. Ayrıca video veri modelimiz kullanılarak, (bulanık) anlamsal, zamansal, (bulanık) uzaysal, hiyerarşik, bölgesel, ve yörünge sorgularına olanak sağlanmaktadır. Geliştirilmiş olan kavramsal modelin ve sistemin video veritabanları ile ilgili herhangi bir uygulama alanına uyarlanabileceğini düşünmekteyiz.

Anahtar Kelimeler: Kavramsal veri modeli, video verisi, belirsizlik, bulanıklık, UML, nesneye dayalı modelleme, video veritabanı uygulamaları, nesneye dayalı veritabanları, bilgi tabanı sistemleri, akıllı sistemler.

To My Family

ACKNOWLEDGMENTS

I would like to present my deepest thanks to my thesis supervisor Prof. Dr. Adnan Yazıcı for his valuable guidance, motivation and support throughout this thesis study.

I want to thank Ceng Multimedia Research Group for their helpful advice and guidance.

I am very grateful to my family for all their patience and tolerance and to my friends who gave me support whenever I needed.

Finally, I would like to thank Levent Tanın for his never ending support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AVI	:	Audio Video Interleave
AVIS	:	Advanced Video Information System
EER	:	Extended Entity-Relationship
ER	:	Entity-Relationship
FKB	:	Fuzzy Knowledge Base
FOOD	:	Fuzzy Object-Oriented Database
IFOOD	:	Intelligent Fuzzy Object-Oriented Database
JMF API	:	Java Media Framework Application Programming Interface
LHS	:	Left Hand Side
MBR	:	Minimum Bounding Rectangle
MPEG	:	Moving Pictures Experts Group
OMG	:	Object Management Group
OVID	:	Object-Oriented Video Information Database
RHS	:	Right Hand Side
UML	:	Unified Modeling Language
XML	:	Extensible Markup Language

CHAPTER 1

INTRODUCTION

Research on video databases has gained more importance lately as a result of improvement in video technologies and increasing popularity of video applications. As conventional databases, video databases are also expected to provide basic facilities such as video modeling, querying and retrieval. The users of a video database are usually interested in the meaning of the videos. Therefore, information inherent to video data should be appropriately modeled and stored in a video database.

Semantic video modeling deals with high-level information in a video, which is perceived by a human. Due to the complexity inherent to video data, semantic video data modeling requires new techniques. There exist studies in the literature on semantic video modeling. Earlier models usually adopt the concept of segmentation or annotation [7, 8, 50] whereas recent studies [2, 3, 4, 6, 10, 11, 17, 18, 21, 47, 48] introduce more complex modeling approaches.

As in traditional databases, conceptual data modeling is an important step for semantic data modeling in video databases. The designed conceptual data model is then mapped to a logical data model. The conceptual model is an abstract representation of the semantic content of video data and it identifies the complex semantic entities (objects and events) and the relationships among the entities at the highest level.

In conceptual video modeling, it is not always possible to express the semantic video content in a precise way. Uncertainty may occur while describing the semantic entities or the relationships among the entities. The users of a video database application tend to construct queries, which contain uncertainty. Consider the following queries:

- Find the videos of female patients who are *quite old*
- Find the videos of fouls done during the game, which can be considered *extremely harsh.*

- Find the goal events, which happen *at the end of the game.*

The uncertain information should be taken into account in order to build a more expressive conceptual model and be able to handle queries, involving in some form of uncertainty.

The widely used ER/EER and UML [26] conceptual models do not allow to represent uncertain information despite their power and flexibility. There already exist some studies [1, 9, 16, 28] on building fuzzy object-oriented conceptual models enhancing EER and UML models with uncertain information. Among those, [28] defines new fuzzy constructs for the extended entity-relationship (EER) model. The studies included in [1, 9, 16] extend UML with some fuzzy constructs for some ordinary applications. Beside EER and UML, there are also other conceptual models used in the literature. For instance, [5, 15] introduce the conceptual ExIFO$_2$ model, which is a fuzzy object-oriented model supporting uncertainty representation. Another study in [27] extends a graph-based object model to represent imprecision and uncertainty, then utilizes the model to define a fuzzy object-oriented model. Although there are a number of studies on fuzzy conceptual modeling, only few studies incorporating fuzziness with conceptual modeling of video data exist. Among these, [11] proposes a fuzzy conceptual model for multimedia database applications. The model supports the representation of uncertainty at the attribute level, object/class level and class/subclass level.

Like any other complex applications, video database applications also require development of intelligent and powerful systems. The research efforts on the interaction or integration of database and knowledge base systems reflect the fact that such integration is becoming an important requirement for complex applications. Video databases are among those applications, which require powerful and flexible modeling and querying facilities. There are research efforts on utilizing knowledge-base systems or rule-based approach for video databases [32, 33, 34, 35, 48, 49]. Among these, [33] defines spatial inference rules which allow to deduce heterogeneous spatial relations by using existing ones. In [34, 49], an architecture for a video database application supporting spatio-temporal and semantic querying is proposed. The spatial relations are stored as Prolog facts in a knowledge base. They also define inference rules in order to reduce the number of facts for better performance. In [32], an SQL-like video query language, which has the capability to handle spatio-temporal queries, is proposed. The language is rule-based since it allows users to express spatial conditions in terms of Prolog-type predicates. The study included in [35], presents a framework for video modeling. For the object extractions and event type descriptions, rules are defined. In [48], a declarative, rule-based constraint query language is introduced to infer new relationships from the existing semantic information. None of the studies on video databases mentioned above provides a tightly coupled integration between database and knowledge base technologies and none of them considers uncertainty.

The research studies on the integration of database and knowledge base technologies mostly ignore uncertainty. Among the previous approaches handling uncertain information in databases and knowledge bases, we base our study on the IFOOD architecture proposed in [30]. IFOOD is an intelligent fuzzy object-oriented database architecture, which integrates a fuzzy object-oriented database and a fuzzy knowledge base. Environmental information system is chosen as the application domain in [30].

In this thesis study, we mainly focus on semantic modeling of video database applications by taking into account uncertainty inherent in video data. Firstly, we propose a new fuzzy conceptual video data model. The proposed model is generic and provides the base classes that can be adapted to any database application domain. It supports the hierarchical structure, and the spatial and temporal relations of video data. Object-oriented modeling [25] is suitable for conceptual modeling of video data, since it provides many abstraction facilities, such as inheritance, aggregation, encapsulation, etc.; thus, it can be used to represent complex video semantic entities, and the relationships among them. In this study, UML is utilized to represent video database applications conceptually. Since UML does not provide the necessary notations to represent uncertain information, we extend UML by introducing some special constructs. Beside the basic constructs introduced previously in studies [9, 10, 11], we additionally use fuzzy inheritance relationship constructor which represents class/subclass level uncertainty and add range and relevance definitions for fuzzy attributes. The extended UML is able to represent uncertainty at the attribute, object/class and class/subclass levels. We also introduce sequence constructor to represent the sequential nature of video data. We think that our fuzzy UML conceptual model can be used for any application domain with uncertain properties.

Secondly, we present an intelligent fuzzy object-oriented database framework based on integration of a fuzzy object-oriented database (FOOD) and a fuzzy knowledge base (FKB) for the video database applications. The presented framework is the adaptation of the IFOOD architecture [30] for video database applications. In this integrated environment, our fuzzy conceptual model is mapped to the logical FOOD model [14] to be used in the fuzzy object-oriented database. Semantic entities of video data are stored in the fuzzy object-oriented database along with uncertain information. A fuzzy knowledge-base system with fuzzy inference capability is used for handling knowledge about video data. We define rules to represent knowledge about temporal and spatial relations between the semantic video entities. Moreover, we define semantic fuzzy rules specific to a video database application domain, football game videos. Semantic fuzzy rules provide the inference of new information from the existing information stored in the database. The presented framework provides (fuzzy) semantic, temporal, (fuzzy) spatial, hierarchical, regional and trajectory queries, based on our video data model. We have also developed a prototype system for the presented framework. The prototype system provides the annotation and querying of video data for its users.

The main contributions of this thesis study are as follows:

1- A fuzzy conceptual video data model is proposed. The model handles the representation of uncertain information, which might naturally occur in video database applications. UML is extended with some special constructs to represent the conceptual model along with uncertainty. The model is a generic, object-oriented and semantic video data model. It represents the semantic entities and the relationships between the entities. Moreover, it considers the hierarchical structure of video data, and semantic, temporal, and spatial relations, which might occur between the semantic entities.

2- A new approach for handling uncertainty and fuzziness in video database applications is introduced. We model uncertainty in complex semantic entities of video data by introducing uncertain attributes, objects, classes, and rules. The presented framework, which is based on IFOOD, provides modeling and querying of rich semantic content and knowledge of video data including uncertainty in a flexible and powerful way.

3- By storing temporal, spatial and semantic rules in a knowledge base, we reduce the amount of information, which should be stored in a video database. The spatial and temporal relations between the video entities are inferred by the knowledge base. Fuzzy semantic rules can be used to infer semantic information instead of explicitly storing it in the database.

4- The presented framework supports a diversity of queries such as (fuzzy) semantic, temporal, (fuzzy) spatial, hierarchical, regional and trajectory queries. The framework provides an integrated environment in query evaluation. The integral parts of the framework, which are FOOD and FKB, are tightly coupled to allow flexible and intelligent querying facilities.

5- The conceptual video data model and the presented framework are all designed to be generic. Therefore, we think that our generic model and framework can be adapted to any application domain related to video databases.

The rest of the thesis is organized as follows: Chapter 2 explains the related previous studies. Our fuzzy conceptual video data model is introduced in Chapter 3. The model is explained by using UML diagrams. Chapter 4 presents the intelligent fuzzy object-oriented database framework and explains how the framework supports queries by giving examples. Chapter 5 presents the implementation of our prototype system. The prototype is explained by giving screenshots and the implementation tools that we used are discussed briefly. Finally, Chapter 6 provides conclusions and gives future directions.

CHAPTER 2

RELATED STUDIES

The research, which is done during the development phase of this thesis study, is summarized in this chapter. The organization of the chapter is as follows: The first section gives a literature survey about the studies on video modeling and querying. The second section explains the studies, which take a rule-based approach in video databases. In the third section, uncertainty in object-oriented databases and knowledge bases are discussed including the FOOD [14] model. The last section gives a brief description about the IFOOD [30] architecture.

2.1 Video Modeling & Querying

The users of a video database are usually interested in the semantic content of video data. Therefore, the rich semantic information in video data should be properly modeled in order to provide retrieval of video according to its semantic content. Due to the complexity of video data content, video modeling needs additional research efforts.

As discussed in [51], which provides a detailed survey on existing semantic video models, information inherent to video data can be classified into three main categories:

1- Low-level features, which include texture, color, shape etc.

2- Syntactic information, which includes the visual objects and temporal, spatial and spatio-temporal relations between these objects. Automatic or semi-automatic extraction algorithms exist which can extract the low-level features and syntactic information from video. Many spatio-temporal models [13, 52] exist in the literature to help modeling of syntactic information.

3- Semantic information which is the answer of “what is happening in the video?”. The semantic information in video data is what is perceived by humans and it is very difficult to extract it automatically.

There exist many studies on semantic video modeling. Earlier models usually utilize the concept of annotation, which consists of keywords and free text. Whereas, recent studies deal with the complex structure of semantic information and propose more complicated data models. Recent semantic video models represent the complex semantic entities such as objects and events; and the relationships among them at the highest level.

It is obvious that conventional relational database models are not adequate for the complex video semantic content. Additional research effort should be made to construct a more powerful video data model. Some of the studies on semantic video modeling are explained below briefly.

OVID (Object-oriented Video Information Database) proposed in [50], introduces a video-object model and a prototype video-object database system. A video frame sequence is considered as a video object, which has its own attributes. The model is schemaless so an object-oriented class hierarchy is not used as a database schema. Data is shared among video objects by using interval inclusion inheritance. A set of composition operations is used for video objects such as interval projection, merge, and overlap. A query language VideoSQL is also introduced to query video objects.

VideoStar proposed in [7], introduces a generic video model not only supporting semantics but also the structure of video documents. They use sequence-scene-shot hierarchy, which is a well-known method for representation of hierarchical structure of video data, and define shot as a contiguous sequence of frames representing a continuous action in time and space. Scenes are constructed by shots, which are related in time and space. The semantically closer scenes are combined to construct a sequence, which describes a continuing story. The model in [7] is represented by enhanced ER model. Annotations (Person, Location, and Event) are associated with video segments to support indexing. The annotation related classes may be extended for any application domain.

In [8], the authors propose an integrated data model and a query language. They also use sequence-scene-shot hierarchy and define an abstract class “Structure” for generalization of the classes “Sequence”, “Scene” and “Shot”. Figure 2.1 represents the hierarchical structure of video data, which is used in [7, 8]. The relations among the semantic entities are not considered in [7, 8].

AVIS (Advanced Video Information System) proposed in [47], introduces a formal video model. The model includes objects, events and activities. An activity is an event type and is used to group similar events. Special data structures such as association maps, frame segment trees, object arrays, activity arrays and event arrays are used to model video data semantically. Various semantic queries, such as elementary object, elementary activity, object occurrence and activity type occurrence, are supported by the model of AVIS. Beside these queries, the model also allows to construct compound queries including the relationships between objects and events, and to construct conjunctive queries, which group same type of queries.

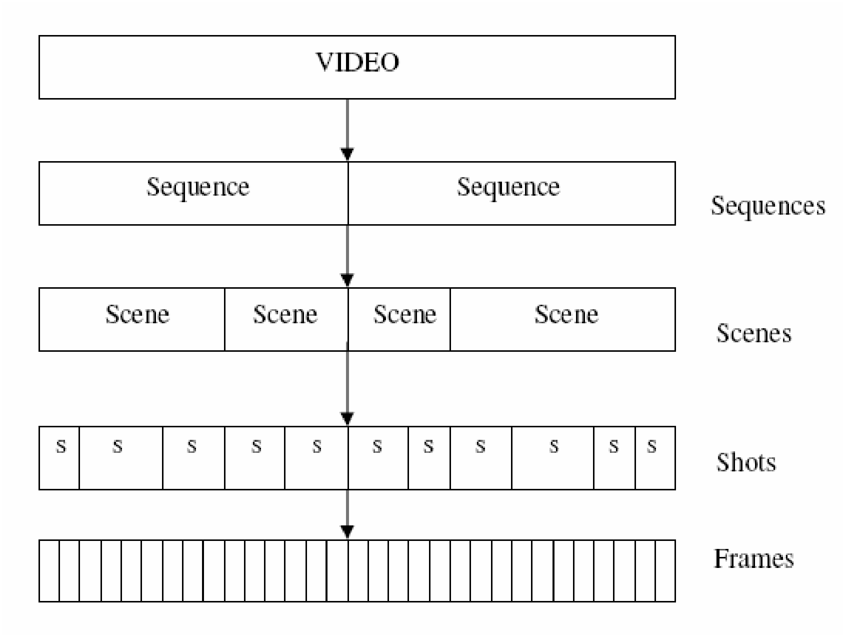


Figure 2.1: The Hierarchical Structure of Video Data

In [13], the data model of AVIS is enhanced to support spatio-temporal features of semantic video entities. The association map in AVIS model is enhanced to represent spatial properties of objects. Spatial data is stored together with the objects in a frame segment tree. An array structure of objects is presented to store spatial data. The spatio-temporal model introduced in [13] provides fuzzy spatial, fuzzy spatio-temporal, fuzzy object trajectory and regional queries of video data. Calculation formulas for membership values are introduced for fuzzy spatial queries.

In [3], a multimodal video data model is proposed. The model has a hierarchical structure for visual, auditory and speech contents, which provides querying video content not only at abstract groups but also at single events. In the hierarchy of the model, the shot structure is not used. A sequence is partitioned into scenes and scenes are associated with events. The model supports all the semantic queries in AVIS and fuzzy spatial, spatio-temporal queries in [13]. In addition to these queries, video data can be queried on visual and auditory content.

In [48], a video data model named as CoPaV^2 is proposed. Temporal cohesion is used for video segmentation, and objects and relationships among the objects are associated with a temporal cohesion. In addition to the semantic model, the authors also introduce a declarative, rule-based constraint query language to infer new relationships from the existing semantic information. Two layers; Semantic Layer and Feature & Content Layer are introduced to distinguish between the visual and semantic features of video data. The queries are divided into two forms: structural part and visual part. The queries related to the structural part are addressed to the Semantic Layer and are handled by FLORID (a deductive object-oriented database management system). The queries related to the visual part are addressed to the Feature & Content Layer and are handled by IBM’s QBIC system (image content-based retrieval system).

In [2], a semantic video data model, which handles semantic information in hierarchy, is proposed. In this model, video data has events, events have subevents and objects are modeled in every level of the hierarchy. Events correspond to sequences and subevents correspond to scenes. The model uses temporal cohesion to provide temporal management. The semantic model is used in BilVideo [49], which is a prototype video database management system consisting of four main parts: Fact-Extractor, Video Annotator, a web-based visual query interface and an SQL-like textual query language. BilVideo supports spatio-temporal queries by the help of a knowledge base and semantic, color, shape, texture queries by the help of an object-relational database.

In [4, 46], the SEBM model, which is a structural and event-based multimodal data model, is introduced. The model includes several sub models to model the structure of video data. Scene structure is not used, and sequence and scene have the same meaning. Video is segmented into shots automatically, according to the background changes, by using a Video Annotator tool. Then semantically closer shots are combined into sequences with manual assistance. Shots are partitioned into events and these events are associated with objects. The model supports content-based, (fuzzy) spatial and temporal queries. An XML database is used to store and query video data.

In [6], an integrated semantic and syntactic video model is proposed. The model is domain-independent and it handles both the high and low level features of video data. ER diagrams extended with object-oriented concepts are used to represent the model. The main entities in the model are object, event and actor. Objects may be involved in several events and have different roles in each event. Such event-specific information is stored in actor entity, which provides grouping of context-dependent properties of objects leading to efficient retrieval. Low-level object-object relations and trajectory information are also stored in actor entity. The instantiation of the model is represented by using graphs. Therefore, the queries of users are translated to those graphs. In query processing, the query graphs are matched with the instantiated ones in the database.

In [17], a generic video indexing model VIDEX, which integrates high-level and low-level content information, is proposed. It also supports sequence-scene-shot structuring. As low-level indexing, spatial and temporal features are supported. As high-level indexing, content-related event, object and location classes are provided. The content-related part of the model can be extended for any application domain. An example to the extension of VIDEX is given in [18, 19] which introduces SMOOTH, a distributed multimedia system. The basic classes used for high-level indexing which VIDEX provides are extended for a soccer-based implementation.

In [21, 22], SIRSALE system is presented. It is a video management system allowing video streams to be stored in distributed repositories. SIRSALE uses a modular model allowing structural (sequence-scene-shot) browsing and content annotation. The video indexing model consists of three parts. The first part deals with distribution of video documents, the second part handles the structural representation of video data, and the third part handles the content-based annotation. The third part can be changed for the needs of any specific application domain.

None of the video models mentioned so far deal with uncertain information inherent to semantic video content. There are very few studies, which enhance a video data model with uncertainty representation. Among these, in [10, 11], a conceptual data model based on ExIFO$_2$ model [5, 15] is introduced. ExIFO$_2$ is a fuzzy object-oriented data model including uncertain and fuzzy information representation. In [10, 11] uncertainty is modeled at the attribute level, object/class level and class/subclass level. The model includes the modeling of events, objects and the relationships among objects. Video is composed of a visual clip and an audio clip. A visual clip is constructed from sequential video frames. To represent this relationship, ExIFO$_2$ is extended with a sequence constructor. A visual clip has events describing the content of the clip and each event may have its own objects. Objects in an event have relationships (semantic or spatial) among each other. The studies [10, 11] also describe how the conceptual model is mapped to the logical FOOD [14] model which is a similarity-based fuzzy object-oriented database model where uncertainty is considered in attribute values, objects and class hierarchy.

2.2 Rule-Based Approach in Video Databases

Video data includes very rich and complex semantic information. Considering the semantic entities, there can be too many events, objects, and relationships among them in a single video. Storing such a huge amount of information could be very space consuming. In addition to that, it is not possible for an automatic extracting algorithm to capture all the spatial information in a video or it is very impractical for a human to provide all the semantic information manually. Therefore, mechanisms, which infer information from existing information, are needed for video databases.

There are research efforts on utilizing knowledge-base systems or rule-based approach for video databases [32, 33, 34, 35, 48, 49]. Among these, in [48] the authors introduce a declarative, rule-based constraint query language to infer new relationships from the existing semantic information. Semantic queries are handled by FLORID (a deductive object-oriented database management system). Therefore, a rule-based approach is used for handling semantic queries.

The study in [33] defines the spatial relations by using Allen's temporal relations [23]. Allen's temporal interval algebra represents relations in one dimension and it is extended to two-dimensional space. After defining the spatial relations in terms of Allen's temporal relations, the authors define a set of spatial inference rules, which allow inferring heterogeneous spatial relations by using existing spatial knowledge. However, as discussed in [49], the proposed system in [33] does not promise that a complete set of relations can be inferred from existing ones for all the object pairs. Therefore there are relations which can not be inferred and should be calculated when a query is handled.

In [34], architecture for a video database application supporting spatio-temporal and semantic querying is proposed. A rule-based system built on a knowledge base is used to store spatial relations. The spatial relations are stored as Prolog facts in the knowledge base. All of the spatial relations in two dimensions are supported and in addition to that, a set of three-dimensional spatial relations is used. The authors also define spatial inference rules in order to reduce the number of facts leading to considerable space savings and better performance. The facts stored in the knowledge base include single key frames instead of time intervals. There are two types of rules: query rules and extraction rules. Extraction rules deal with time intervals in order to extract the spatio-temporal relations from a video. The extraction is done in a semi-automatic way and MBRs (Minimum Bounding Rectangle) of video objects are annotated manually. The spatio-temporal relations are found automatically by using MBRs. Extracted relations are stored as facts in the knowledge base such as west(A, B, k) where A and B are video objects and k is the key frame. The facts stored in the knowledge base are later used in query processing by the query rules. The facts, which can be derived by rules, are not stored in the knowledge base. Instead they are inferred when a query is processed. If west(A, B, k) and west(B, C, k) are the two facts in the knowledge base, there is no need to store the fact west(A, C, k) since it can be derived by the inference rule: *west(A, B) and west(B, C) => west(A, C)*

The proposed rule-based architecture is embedded in the BilVideo [49], a prototype video database management system. The authors also introduce an SQL-like video query language in [32]. The language has the capability to handle spatio-temporal queries. The language is rule-based since it allows users to express spatial conditions in terms of Prolog-type predicates.

The rule-based architecture utilized in [34, 49] only define spatial inference rules and semantic inference rules are not considered.

The study included in [35], presents a framework for video modeling. Automatic definition of high-level concepts (video objects and events) by using the extracted features is supported by the proposed model. For the object extractions and event type descriptions, rules are defined. They give examples for soccer game video modeling and define rules for describing the soccer events.

2.3 Uncertainty in Object-Oriented Databases & Knowledge Bases

In this section, firstly the FOOD model is explained briefly. Secondly uncertainty issues in knowledge bases are discussed.

2.3.1 Fuzzy Object-Oriented Database (FOOD) Model

Complex objects might have uncertain information, which an object-oriented data model should consider. FOOD model, which is firstly proposed in [39] is a similarity-based data model. The study in [14] extends this model for a better representation of uncertainty. FOOD model is used as logical data model in this study. In this section, the FOOD model is briefly explained.

In FOOD model, regarding the representation of imprecise information, uncertainty is handled at three levels: attribute level, object/class level and class/superclass level.

2.3.1.1 Attribute Level Uncertainty

FOOD deals with three types of uncertainty at the attribute level. The first type being incomplete type occurs when the value of the attribute is specified as a range value. For example, the audience in a football game may take values as 10000 – 20000. This type of uncertainty is called “incompleteness.” The second type of uncertainty occurs when the value of the attribute is not known (unk), does not exist (dne) or there is no information on whether a value exists or not (ni). For example, the description of a video might not be known (unk), the description for a video might not exist (dne) or we might not know whether a description for a video exists or not (ni). This type of uncertainty is called “null”. The third type of uncertainty occurs when the value of the attribute is vaguely specified. This type of uncertainty is called “fuzzy”. For example, the weather condition in a football game can be specified with a fuzzy term “very hot”.

Each fuzzy attribute has a domain (the set of values the attribute may take) independent of its class. A domain consists of linguistic values which are called fuzzy terms. Range is a subset of the fuzzy attribute’s domain, representing the ideal values but the attribute can take any values from its domain. FOOD allows definition of relevance for fuzzy attributes, which is a real number between 0 and 1 reflecting the importance of the range definition of that fuzzy attribute in defining the boundaries of its class. Range and relevance are used to find the membership degree of an object to its class, and the membership degree of a class to its superclass(es). They are the same for each instance of a class.

A similarity relation, which is represented by a similarity matrix, is the basis for the similarity-based FOOD model. A similarity matrix defines similarities between every pair of the elements in a fuzzy domain. An example similarity matrix of a fuzzy attribute *age* is given in Table 2.1. The domain of the *age* attribute is {very old, old, young, very young, infant}

Table 2.1: Similarity Matrix for the Fuzzy Attribute *age*

age	very old	old	young	very young	infant
very old	1.0	0.7	0.0	0.0	0.0
old	0.7	1.0	0.0	0.0	0.0
young	0.0	0.0	1.0	0.8	0.1
very young	0.0	0.0	0.8	1.0	0.3
infant	0.0	0.0	0.1	0.3	1.0

Fuzzy attributes are multivalued; thus, they may take a set of values and these values are connected by one of the AND, OR, XOR semantics. The following representation is used for the multivalued attributes:

Logical Operator	**Representation**
AND	<…>
OR	{…}
XOR	[…]

Assume the attribute *weather*, which has the domain {cloudy, sunny, mild, cold, hot, windy}. The following representations are valid:

AND: Attribute *weather*'s value is <cloudy, cold>, which means the weather is cloudy and cold.

OR: Attribute *weather*'s value is {mild, hot}, which means the weather is mild or hot, or maybe both.

XOR: Attribute *weather*'s value is [hot, cold], which means the weather is hot or cold but not both.

Here, the meaning of XOR semantics is different from the logical operator XOR that returns true for an odd number of trues.

In FOOD model, the semantics are determined during the range definition of a fuzzy attribute. For example, consider a class C which has attributes a, b and c. The range definitions may be as follows:

$rng_c(a) = \{a_1, a_2, a_3\}$ where $dom_c(a) = \{a_1, a_2, a_3, a_4, a_5, \ldots, a_k\}$ for OR semantics

$rng_c(b) = <b_1, b_2>$ where $dom_c(b) = \{b_1, b_2, b_3, b_4, b_5, \ldots, b_k\}$ for AND semantics

$rng_c(c) = [c_1, c_2, c_3]$ where $dom_c(c) = \{c_1, c_2, c_3, c_4, c_5, \ldots, c_k\}$ for XOR semantics

2.3.1.2 Object/Class Level Uncertainty

Uncertainty at the object/class level refers to the existence of a partial membership of an object to its class. In FOOD model, the boundaries of a class might be uncertain since it has fuzzy attributes. Range of a fuzzy attribute indicates ideal values for that attribute. Since a fuzzy attribute may take any value from its domain regardless of its range definition, some objects are full members of their classes with a membership degree of 1 whereas some objects are member of their classes with a membership degree changing between 0 and 1. The values of fuzzy attributes of an object determine the membership degree of that object to its class. The closer the value of fuzzy attributes of an object to range definitions, the higher the object membership degree. Relevance of the fuzzy attributes and the similarity between the fuzzy attributes' values and their range definitions determine the membership degree of an object to its class. To find the object membership degree, the following formula is used:

$$\mu_C(o_j) = \frac{\sum INC(rng_C(a_i) / o_j(a_i)) * RLV(a_i, C)}{\sum RLV(a_i, C)}$$

In the formula, $INC(rng_C(a_i)/o_j(a_i))$ is the inclusion degree of the attribute a_i's value to its range. The calculation of the inclusion degree depends on the semantics of the attribute which might be one of AND, OR, XOR semantics. $RLV(a_i,C)$ is the relevance of the attribute a_i. The weighted-average is used to calculate the object membership degree.

Inclusion Formulas for Fuzzy Attributes

The calculation of the inclusion degrees for different semantics is explained below:

1- AND semantics:

AND semantics is strong since it requires that all of the values appear at the same time. The formula for AND semantics is as follows:

$$INC(rng_C(a_i)/o_j(a_i))= Min[Min[Max(\mu_S(x,y))],Min[Max(\mu_S(z,w)]],$$

$$\forall x \in rng_C(a_i),\ \forall y \in o_j(a_i),\ \forall z \in o_j(a_i),\ \forall w \in rng_C(a_i).$$

2- OR semantics:

When the values of an attribute get more dissimilar to each other, the degree of uncertainty increases. The formula for OR semantics is as follows:

$$INC(rng_C(a_i)/o_j(a_i)) = Min[Max(\mu_S(x,z)),Threshold(o_j(a_i))],$$

$$\forall x \in o_j(a_i), \forall z \in rng_C(a_i)$$

Here, the threshold value indicates the minimum level of similarity between the values of an attribute and it can be formulated as follows:

$$Threshold(o_j(a_i)) = Min[\mu_S(x,z)],\ \forall x, \forall z \in o_j(a_i)$$

3- XOR semantics:

With XOR semantics, only one of the attribute's values exist at a time. Assuming equal probabilities for the values of an attribute, the formula for XOR semantics is as follows:

$$INC(rng_C(a_i)/o_j(a_i)) = Avg[Max(\mu_S(x,y))], \qquad \forall x \in o_j(a_i), \forall y \in rng_C(a_i)$$

Inclusion Formulas for Incomplete Attributes

The domain and range of an incomplete attribute are defined by using ranges represented by two values such as {100 - 10000}. Considering the value of an incomplete attribute, there are five cases to calculate the inclusion degree of the value to the range of the attribute [5]. These cases are listed below:

$R[R_1..R_2]$ is the range of the attribute

$V[V_1..V_2]$ is the value of the attribute

$D[D_1..D_2]$ is the domain of the attribute

1- Value of the attribute is within the boundaries of the attribute's range.

In this case, inclusion degree is 1.

$INC(rng_C(a_i)/o_j(a_i)) = 1$, *where* $R_2 \geq V_2 > V_1 \geq R_1$

2- Value of the attribute is out of the boundaries of the attribute's range

In this case, inclusion degree is 0.

$INC(rng_C(a_i)/o_j(a_i)) = 0$, *where* $R_1 > V_2 > V_1 \geq D_1$ *or* $D_2 \geq V_2 > V_1 > R_2$

3- Attribute's range is the subset of the attribute's value.

In this case, inclusion degree is calculated by using the following formula:

$INC(rng_C(a_i)/o_j(a_i)) = (R_2 - R_1 + 1) / (V_2 - V_1 + 1)$, *where* $V_2 \geq R_2 > R_1 \geq V_1$

4- Attribute's range and attribute's value intersect at the lower side of the range.

In this case, inclusion degree is calculated by using the following formula:

$INC(rng_C(a_i)/o_j(a_i)) = (V_2 - R_1 + 1) / (V_2 - V_1 + 1)$, *where* $R_2 \geq V_2 \geq R_1 \geq V_1$

5- Attribute's range and attribute's value intersect at the upper side of the range.

In this case, inclusion degree is calculated by using the following formula:

$INC(rng_C(a_i)/o_j(a_i)) = (R_2 - V_1 + 1) / (V_2 - V_1 + 1)$, *where* $V_2 \geq R_2 \geq V_1 \geq R_1$

2.3.1.3 Class/Subclass Level Uncertainty

Uncertainty at the class/subclass level refers to the existence of a partial membership of a class to its superclass(es). This type of uncertainty indicates that the fuzziness occurs at the class inheritance hierarchy since a class hierarchy might not be constructed precisely in some cases. The membership degree of a class to its subclass is calculated by the following formula:

$$INC = Min(Max[\mu_S(x,y)]),\quad \forall x \in rng_{c1}(a_i),\ \forall y \in rng_{c2}(a_i),$$

where c2 is the subclass of the class c1

For further information about calculation of inclusion degrees and membership degrees, the readers may refer to [14].

2.3.2 Rule-Based Programming & Knowledge Bases

Rules are widely used to represent knowledge in various areas of computer science studies. Rule-based programming differs from procedural programming in a number of ways [40]. In procedural programming, a program consists of a set of instructions telling the computer what to do in what order, whereas in rule-based programming, a program is made up of a set of rules on a problem domain.

A rule engine is a program, which repeatedly checks whether to apply any rule for a set of facts. By using rule-based programming, much simpler and shorter programs can be written since you focus on the rules and do not deal with a long list of nested if statements (which you have to use in case of procedural programming).

Knowledge in a domain is represented by rules. A rule is simply an If-Then statement such as:

IF it snows THEN the weather is cold.

The "if" part, which is called predicate, is the left-hand side (LHS) of the rule. The "then" part is the right-hand side (RHS) of the rule, which is called actions.

A rule engine consists of three main parts: an inference engine, a rule base and a working memory. Figure 2.2 represents the architecture of a typical rule engine. The facts are stored in working memory and the rules are stored in the rule base. The inference engine checks repeatedly whether any rule can apply to any fact in the working memory by using pattern-matching process. If a fact matches with the "if" part of a rule (selection is done), the actions in the "then" part of the rule are executed.

There are various knowledge base systems developed. Among the most popular ones are CLIPS (C Language Integrated Production System) [42] and Jess (Java Expert System Shell) [37]. CLIPS is a knowledge base written in C language whereas Jess is a Java based rule engine. Both of them use the well-known Rete algorithm for pattern matching phase.

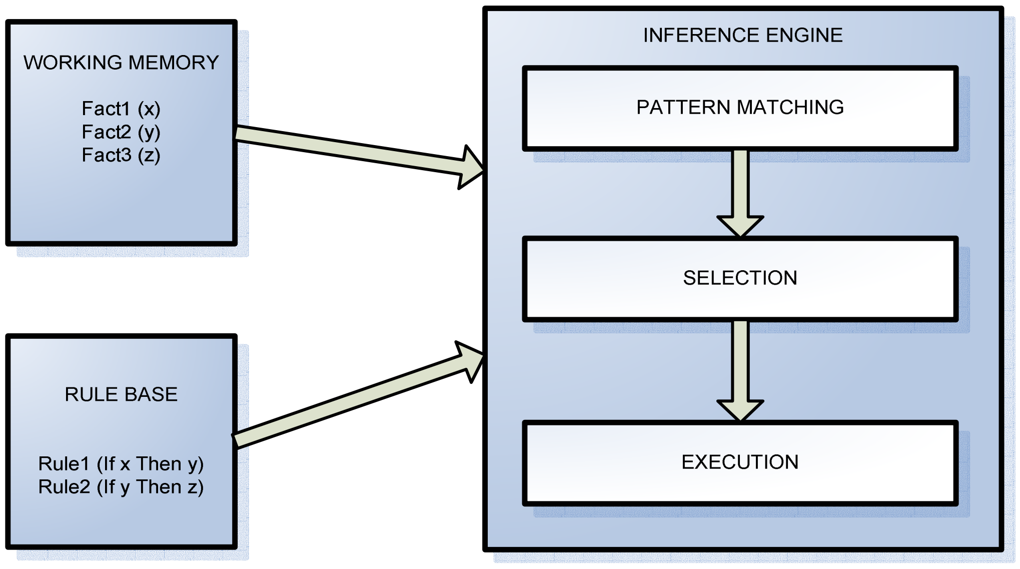


Figure 2.2: Representation of a Rule-Based System

2.3.3 Handling Uncertainty in Knowledge Bases

There have been previous studies [31, 43, 44] on building knowledge bases extended with uncertainty. FuzzyCLIPS [43] extends the CLIPS knowledge base to allow fuzzy rule processing. Both crisp and fuzzy terms are allowed. The extended version of Rete algorithm is used for the pattern-matching phase.

NRC FuzzyJ Toolkit & FuzzyJess [44] bring an extension to Jess rule engine in order to provide the capability of handling fuzzy concepts and reasoning. The study is mainly based on FuzzyCLIPS, but is developed in Java language. NRC FuzzyJ Toolkit can be used standalone or it can be integrated with Jess (leading to FuzzyJess).

Both of the studies, FuzzyCLIPS and FuzzyJess use the concept of certainty factor to handle fuzziness. Facts and rules can be defined with certainty factors. Membership functions, which map the crisp domains to fuzzy domains, are utilized for fuzzy processing. Either of the studies does not support the concept of similarity matching.

On the other hand, the fuzzy knowledge base, which is introduced in [31], extends the CLIPS knowledge base to allow uncertainty by using a similarity-based approach. The fuzzy knowledge base (FKB) is used in an integrated environment called IFOOD architecture [30]. IFOOD architecture uses the FOOD model, which is similarity-based. In order to integrate the FOOD with FKB, the fuzzy knowledge base should provide similarity matching. FKB allows using both crisp and fuzzy terms in a rule. The Rete algorithm is extended to allow similarity-based approach and is used in the inference mechanism of FKB.

2.4 Intelligent Fuzzy Object-Oriented Database (IFOOD) Architecture

Uncertainty is not usually considered in many research efforts on the integration of the database and knowledge base technologies. Among the previous approaches handling uncertain information in databases and knowledge bases, this study is based on the IFOOD architecture proposed in [30].

2.4.1 Architecture of IFOOD

IFOOD is an intelligent fuzzy object-oriented database architecture, which integrates a fuzzy object-oriented database (FOOD) and a fuzzy knowledge base (FKB). The representation of the IFOOD architecture is given in Figure 2.3.

FOOD is responsible for object management whereas FKB handles knowledge management. FOOD (fuzzy object-oriented database) model [14] is used as the logical model for the fuzzy object-oriented database. FKB has the fuzzy inference capability so it processes fuzzy rules.

FOOD and FKB are connected by a bridge, which provides all the interaction between these two parts. Bridge provides a unified view for the users of the architecture. There is a user interface, which is connected to bridge. The users of this architecture do not need to know any detail of the physical structure of the underlying system and they only communicate with the bridge. Users can create classes, objects, fuzzy rules and define fuzzy types, domains, similarity matrices and membership functions by using the user interface. The interface also allows querying of the system.

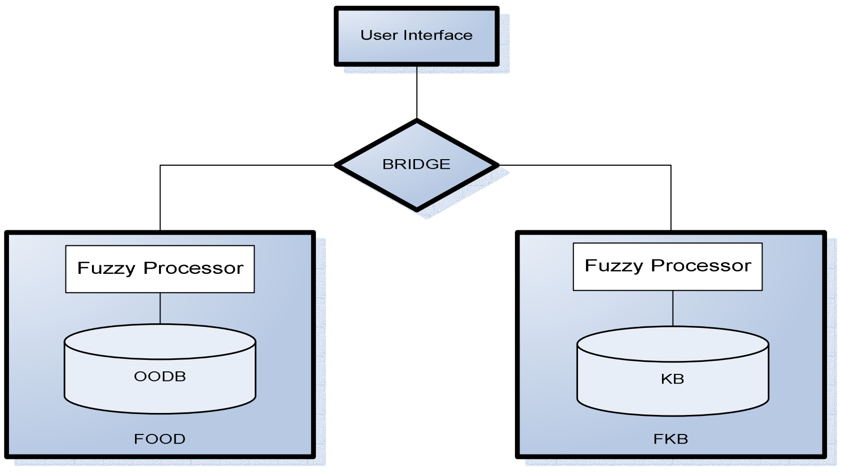


Figure 2.3: IFOOD Architecture

Since FOOD is used as the logical model, similarity matching is used to evaluate queries. At the database and knowledge base parts of the architecture, there is a fuzzy processor, which handles all of the operations related to uncertainty (such as object/class membership degree and inclusion degree calculations, similarity matching, etc.).

Fuzzy rules are used for derived attributes and virtual classes. A fuzzy rule might include both crisp and fuzzy conditions. Similarity matching is used to process fuzzy rules.

IFOOD allows representation of both fuzzy and crisp values for a fuzzy attribute. Some of the objects stored in database may take fuzzy value and some of the objects may take crisp value for the same fuzzy attribute. Therefore, the user can construct a query for the fuzzy attribute by specifying fuzzy or crisp query conditions. For this reason, membership functions are utilized to map crisp values to a fuzzy term. Membership functions help to find the inclusion of a crisp value to a fuzzy set. Membership functions, which are used in

IFOOD implementation, are right-decreasing, triangle, trapezoidal, elliptical, and right-increasing. They are also utilized in fuzzy rule processing. An example to a membership function definition in IFOOD is given below:

membershipof fuzzyTemp hot triangle 75, 100, 140;

Here, membership function "triangle" is associated with the fuzzy term "hot" which is defined for the fuzzy type "fuzzyTemp".

The study in [30] also proposes an object-oriented database language extended with declarative rules. The language provides definition of uncertain classes, objects and rules. As the application domain, environmental information system is choosen.

2.4.2 Fuzzy Knowledge Base (FKB) and Fuzzy Rules

The study in [30] utilizes the fuzzy knowledge base, which is proposed in [31]. FKB is capable of fuzzy inference. Knowledge is represented by IF-THEN rules where both of the consequent and antecedent parts consist of linguistic variables. As an example, "IF x is A THEN y is B" is a fuzzy rule where x and y are linguistic variables, A and B are fuzzy sets. "x is A" is the antecedent of the rule and it may have one or more clauses connected with fuzzy logical operators (AND, OR). "y is B" is the consequent of the rule.

Fuzzy rules are used to infer values of derived attributes. The inference mechanism of the fuzzy knowledge base, which is represented in Figure 2.4, uses similarity matching [14] during the pattern-matching phase.

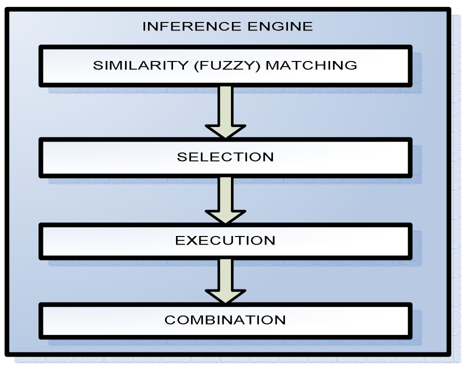


Figure 2.4: Inference Mechanism of IFOOD

The fuzzy and crisp attributes might be used together in a rule. If an antecedent predicate is defined with a crisp attribute, traditional pattern matching is used and if the matching is successful, the matching degree will be 1. If the predicate is defined with a fuzzy attribute and the value of an object's attribute is crisp, membership functions are utilized to find the membership degree of the crisp value to the fuzzy set defined in the rule.

2.4.3 Integration of FOOD and FKB

As mentioned before, IFOOD provides an integrated environment since it tightly couples a fuzzy object-oriented database and a fuzzy knowledge base. Queries are evaluated by the interaction of these two integral parts.

Users of the IFOOD architecture might construct queries including fuzzy, crisp conditions or a combination of both. When a query request is made, the bridge firstly sends the query to FOOD. The objects in FOOD, which satisfy the conditions, are sent back to bridge. If the query needs a rule firing, the bridge transfers those objects to the working memory of FKB. The rules are fired on the objects in FKB. Then, the bridge takes the result objects and sends them to the user interface.

The algorithm for the query evaluation, which is done by the bridge, is as follows:

1. If query needs virtual classes, then evaluate virtual class condition and get the satisfied objects.

2. If query does not need virtual classes, construct the crisp query and send it to FOOD, get the satisfied objects.

3. If query includes fuzzy predicates, then evaluate fuzzy predicates and get the satisfied objects.

4. If query needs rule processing, then transfer the satisfied objects to FKB and start its inference engine. Get the satisfied objects from the working memory of FKB.

5. Submit the satisfied objects to user interface.

More information about the query processing of the IFOOD architecture can be found in [30, 55].

2.4.4 Implementation of IFOOD

The authors in [30] also developed a prototype system. As the object-oriented database, they use Itasca, which is an object database management system. The knowledge base, which is used in the prototype, is CLIPS [42]. C++ programming language is used during the development phase. CLIPS is extended to support fuzzy inference. The objects are stored in Itasca and the rules are stored in CLIPS. The user interface and bridge are implemented in C++ environment.

CHAPTER 3

FUZZY CONCEPTUAL VIDEO DATA MODEL

People are usually interested in the semantic meaning of the videos they watch. They identify the videos by the objects appearing or the events happening in the video. Events and objects are the semantic entities of video data. The relationships between the semantic entities are also important to express the content of a video. Semantic video modeling deals with modeling the semantic entities (such as events and objects) and the relationships between those entities.

Video data has a more complex structure than other types of multimedia data, which are image, audio, and text, since video might consist of audio, image and text at the same time. Video has a temporal nature, since it is composed of sequential video frames streaming in a time interval. Events occur in a video within a time interval. Objects appear in a video in different time intervals several times. Objects may be involved in different events, with a different semantic role in each event.

Events have temporal relations between each other. Objects might have both temporal (according to the time intervals they appear in) and spatial (according to their position within a frame with respect to each other) relations between each other. Since objects might change their position within a time interval, we should also consider the spatio-temporality of video objects.

Video data is a sequence of chronologically ordered video frames. A set of video frames which share the same low-level features (such as background color) can be grouped together to form a video shot. Video shots, which represent a semantic content occurring at the same time and same place, construct a video scene. Semantically closer scenes, which represent a continuing story, are combined to form a video sequence. This hierarchical structure of video data helps us to model the semantic content more efficiently and to relate the semantic content with physical units of video data.

It is sometimes not possible to identify the semantic information in a video precisely or completely. There might be uncertain information, which also leads to uncertainty in query results for video databases. Therefore, uncertainty should be considered by a semantic video model.

In this study, we mainly focus on semantic modeling of video database applications by taking into consideration uncertainty inherent in video data, and propose a fuzzy conceptual video data model. The proposed model is generic and provides the base classes that can be adapted to any video database application domain. It supports the hierarchical structure of video data. Additionally, the model considers the semantic, temporal and spatial relations between the semantic entities (objects, events and actors). The model is object-oriented and UML is used to represent the conceptual model. UML is extended with some special constructs to represent uncertain information. The proposed model supports (fuzzy) semantic, temporal, (fuzzy) spatial, hierarchical, regional and trajectory queries. In this chapter, the proposed conceptual video model is explained in detail.

The rest of the chapter is organized as follows: The first section gives a brief description about UML. The second section describes the extensions done to the UML model and the third section introduces the proposed fuzzy conceptual data model. The fourth section discusses how the conceptual model is mapped to the FOOD model [14]. The application of the conceptual model for football game videos is presented in the fifth section. Finally, the last section gives exemplary queries to show how the model supports various types of queries.

3.1 UML (Unified Modeling Language)

UML [26] is a widely used standard of Object Management Group (OMG) for modeling systems by using object-oriented concepts. In UML, different types of diagrams may be used, such as class, sequence, collaboration, component, object, and state diagrams. In this study, UML class diagrams are utilized to represent the conceptual video data model. The class diagrams are extended to represent uncertain information.

Figure 3.1 gives a brief explanation for the basic UML notations, which are used in a class diagram.

3.2 Extending UML

In this study, UML class diagrams are extended to represent our fuzzy conceptual data model. UML model needs to be extended for two main reasons:

1- To represent uncertain information

2- To represent sequential nature of video data

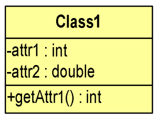
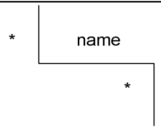
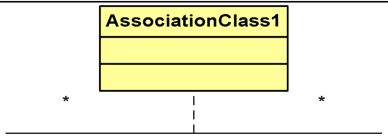
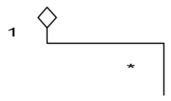
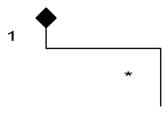
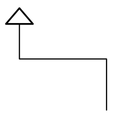
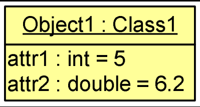
Notation	Description
 Class1 -attr1 : int -attr2 : double +getAttr1() : int	**Class** is a description of a set of objects, which share attributes and methods. The notation includes three parts: the name of the class, attributes and methods.
	**Association** is the semantic relationship between objects of two classes. It is usually bi-directional. The name of the association and the end multiplicities are also displayed.
	**Association class** represents additional information about an association between two classes. It cannot exist by itself.
	**Aggregation** is a special kind of association, which represents a whole/part relationship. An object of the "whole" class has objects of the "part" class.
	**Composition** is a special case of aggregation where an object of the "part" class may belong to only one object of the "whole" class at a time.
	**Generalisation** is the relationship between a more general class and a more specific class.
	**Object** is the instance of a class. The name of the object is written before the name of the class. The attribute values are also displayed.

Figure 3.1: Basic UML Notations

In this study, uncertainty is handled in three levels: attribute level, object/class level and class/superclass level, as defined in [14].

Attribute Level

There are three types of uncertainty at the attribute level [14]. The first type of uncertainty occurs when the value of the attribute is specified as a range value. For example, the audience in a football game may take values in the range 10000 – 20000. This type of uncertainty is called “incompleteness”. The second type of uncertainty occurs when the value of the attribute is not known (unk), does not exist (dne) or there is no information on whether a value exists or not (ni). This type of uncertainty is called “null”. For example, the description of a video might not be known, the description might not exist or we might not know whether the description exists or not. The third type of uncertainty occurs when the value of the attribute is vaguely specified. This type of uncertainty is called “fuzzy”. For example, the weather condition in a football game can be specified with a fuzzy term “very hot”.

To represent these three types of uncertainty at the attribute level, the uncertain data types defined in [9] are also used in this study. "UT_fy" represents fuzzy data type, "UT_nu" represents null data type and "UT_in" represents incomplete data type. We define three classes as in Figure 3.2, to represent these three uncertain data types. These data types are used in the class definitions in our fuzzy conceptual data model.

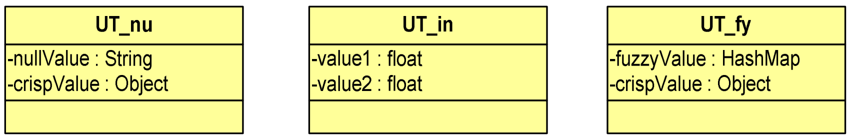


Figure 3.2: The Representation of Classes for Uncertain Data Types

To represent a class, which has uncertain information, the notation introduced in [9], is also used in this study. A tag “U” is placed in the left-hand side of the name compartment of a class if the attributes of that class have uncertain values. Figure 3.3 shows the representation of classes Video and Event, which have attributes having uncertain values.

The definition of a fuzzy attribute is as follows: <attribute name> : <type> <range> <relevance>. Each fuzzy attribute has a domain (the set of values the attribute may take) independent of its class. <range> is a subset of the fuzzy attribute’s domain, representing the ideal values but the attribute can take any values from its domain. <relevance> is a real number between 0 and 1 reflecting the importance of the fuzzy attribute in defining the boundaries of its class. <range> and <relevance> are used to find the membership degree of an object to its class, and the membership degree of a class to its superclass(es). The membership degrees are calculated by using inclusion formulas. Each class, which has a fuzzy attribute, has method definitions of "setRanges()" to set the ranges of fuzzy attributes, "setRelevances()" to set the relevances of fuzzy attributes, "calcMShip" to calculate the object-class membership degrees, and "calcCSCMShip" to calculate the class-subclass membership degrees. For further information about inclusion formulas to find membership degrees, readers may refer to [14].

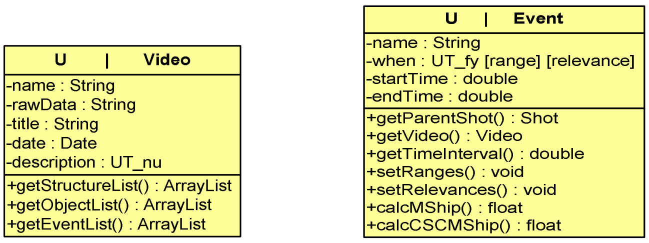


Figure 3.3: Representation of Uncertain Classes Video and Event

Object/Class Level

Uncertainty at the object/class level refers to the existence of a partial membership of an object to its class. If the boundaries of a class are uncertain, then the objects of this class may be a member of the class with a membership degree between 0 and 1. To represent object/class level uncertainty in our model, we extended UML class diagrams by adding a new notation "|U|" to the left-hand side of the name compartment of a class. This notation is similar to the "double-square" notation defined in [9]. Putting "|U|" to the class definition indicates the uncertainty which occurs when an instance of a class is a partially member of its class.

Class/Subclass Level

Uncertainty at the class/subclass level refers to the existence of a partial membership of a class to its superclass(es). This type of uncertainty indicates that uncertainty occurs at the class inheritance hierarchy since sometimes we might not be able to construct a class hierarchy precisely. To represent class/subclass level uncertainty, a fuzzy inheritance relationship constructor is introduced to the UML data model. The capital letter "F" is used to indicate that the inheritance relationship is fuzzy as represented in Figure 3.4.

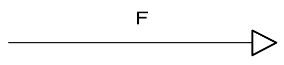


Figure 3.4: The Representation of the Fuzzy Inheritance Relationship

The example given in Figure 3.5 shows the new constructors added to the UML model to represent uncertainty at object/class and class/subclass levels. The class EndangeredSpecies is the superclass, the classes Butterfly, EndangeredBear, and Wolf are the subclasses. The inheritance relationship in this example is fuzzy; therefore, the classes Butterfly, EndangeredBear and Wolf are the subclasses of EndangeredSpecies with a membership degree, which may change between 0 and 1.

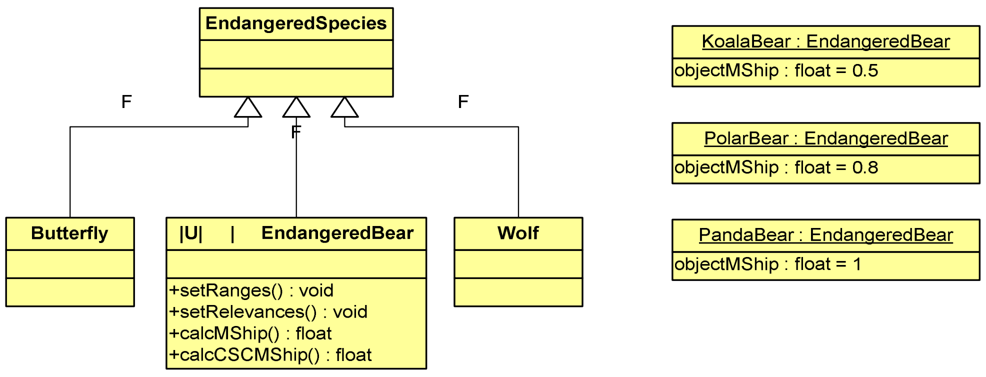


Figure 3.5: Object/Class and Class/Subclass Level Uncertainties

The class EndangeredBear has "|U|" notation indicating that it is also uncertain at the object/class level. The instances of EndangeredBear class are the members of the class with a membership degree, which may change between zero and one. As it is represented in Figure 3.5, there are three instances of the class EndangeredBear: KoalaBear, PolarBear and PandaBear. KoalaBear is an instance of the EndangeredBear class with a membership degree of 0.5, PolarBear is an instance with a membership degree of 0.8 and PandaBear is an instance with a membership degree of 1, which means it has a full membership to the EndangeredBear class.

Fuzzy aggregation/composition

The fuzzy conceptual data model has the constructors to represent fuzziness in aggregation and composition relationships. In a fuzzy aggregation or composition relationship, the constituent whose role is “part” is the part of the constituent whose role is “whole” with a membership degree between 0 and 1. The fuzzy aggregation or composition relationship is represented by using the notation “F” as shown in Figure 3.6 in the same manner with [9]. The difference between aggregation and composition relationships is explained in UML model [26].

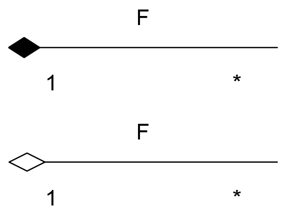


Figure 3.6: The Representation of Fuzzy Aggregation/Composition Relationships

Figure 3.7 represents an example for a fuzzy aggregation relationship, which may be necessary in various applications such as environmental information systems [9]. For instance, an instance of the class Chemical is a chemical substance such as carbon monoxide or carbon dioxide. The Chemical instances are aggregated to form a Pollutant instance. The membership degree of a Chemical instance in this whole-part relationship is fuzzy leading to a fuzzy aggregation. The values of fuzzy attributes *harmfulness* and *dose* determine the membership degree of Chemical instances to this aggregation relationship. The range of these attributes is specified as [low, medium, high] and the relevance of *harmfulness* is 0.8, whereas the relevance of *dose* is 0.7.

Video data can be thought of as a sequence of chronologically ordered video segments. In [10, 11], a new constructor “sequence” is introduced to the ExIFO$_2$ [5, 15] model with respect to the sequential nature of video data. The sequence constructor is a special case of aggregation where the constituents have a chronological order. In this study, a new relationship "sequence", similar to the sequence constructor defined in [10, 11] is added to the UML model. Representation of the "sequence" relationship is shown in Figure 3.8.

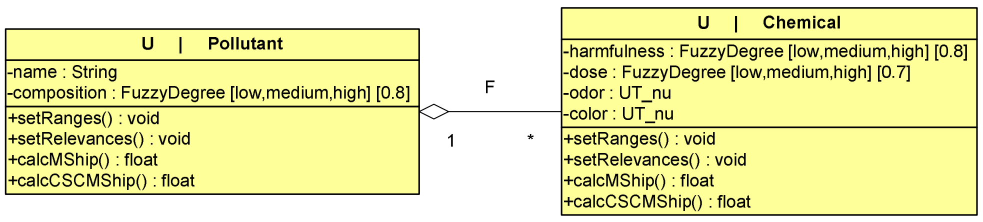


Figure 3.7: The Fuzzy Aggregation between Pollutant and Chemical Classes

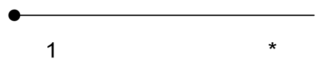


Figure 3.8: The Representation of "Sequence" Relationship

3.3 Conceptual Video Data Model

In this study, the fuzzy conceptual data model is represented by utilizing extended UML class diagrams. With respect to the hierarchical structure of video data, video is segmented into sequences, a sequence is segmented into scenes and a scene is segmented into shots. For example, in a football game video, a sequence may correspond to a continuing story such as the first half of the football game, a scene may correspond to a goal scene, and a shot corresponds to a free kick in that goal scene.

In the video model, the classes Sequence, Scene and Shot inherit from the class Structure. The Structure class corresponds to a video stream. The attributes *startTime* and *endTime* represent the time interval information of the corresponding video stream. A video is composed of sequential video streams; this relationship is represented in Figure 3.9. Sequences are sequential in a video whereas scenes and shots do not need to be sequential and they may overlap in the model. To represent the hierarchical structure of video data, an aggregation relationship between Sequence and Scene, and an aggregation relationship between Scene and Shot are defined as represented in Figure 3.10.

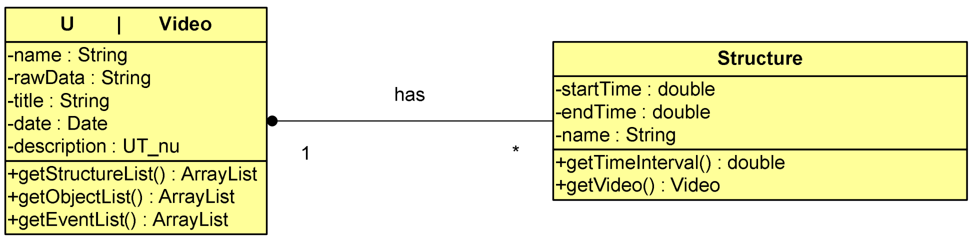


Figure 3.9: Sequence Relationship between Classes Video and Structure

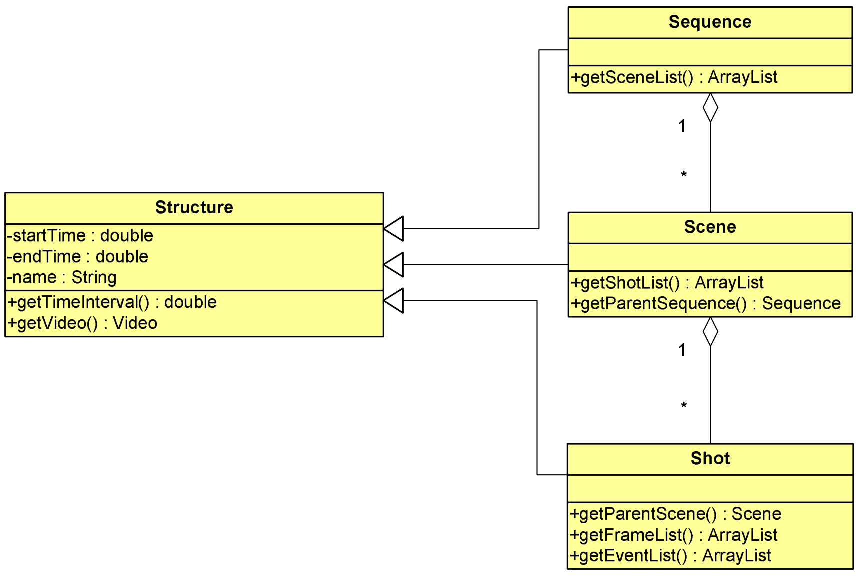

Sequence
+getSceneList() : ArrayList
1
*
Structure
-startTime : double
-endTime : double
-name : String
+getTimeInterval() : double
+getVideo() : Video
Scene
+getShotList() : ArrayList
+getParentSequence() : Sequence
1
*
Shot
+getParentScene() : Scene
+getFrameList() : ArrayList
+getEventList() : ArrayList

Figure 3.10: The Representation of the Hierarchical Structure of Video Data

In our model, video data has three main semantic entities: event, object, and actor. In the following sections, the classes related to these entities will be explained in detail.

Event

The class Event represents the information about what is happening in a video stream. It has three crisp attributes *name*, *startTime*, *endTime*, and an uncertain attribute *when*. The attribute *when* corresponds to the semantic time of the event. Sometimes we can not tell the exact time of an event in a video. We tend to use words like “at the beginning of the video” or “at the middle of the video”. Attribute *when* is used when we can not exactly define the time of the event. The attributes *startTime* and *endTime* represent the time interval in which the event occurs. In our model, events are defined within a shot. A shot may contain many events; this relationship is represented in Figure 3.11.

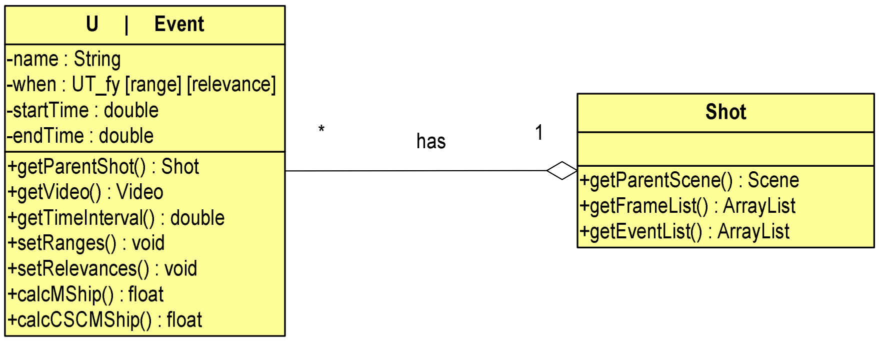


Figure 3.11: The Aggregation Relationship between Event and Shot

For a specific application domain, the class Event may be extended and many domain specific subclasses may be defined. The inheritance hierarchy of the class Event is represented in Figure 3.12. This hierarchy can be extended and there may be multiple levels in the class hierarchy.

Relations may exist among instances of Event class. There can be two types of relations among Event instances in our model.

1- Causal relations

We use a causal relation in case an event may be the cause of another event. Causal relations may be “causes” and “resultingFrom”.

2- Temporal relations

A video event occurs in a specific time interval, thus, there can be temporal relations among events. Temporal relations are defined according to Allen’s temporal algebra

[23]. Temporal relations utilized in this study are *before, meets, during, overlaps, starts, finishes, equal* and their inverses, except *equal.*

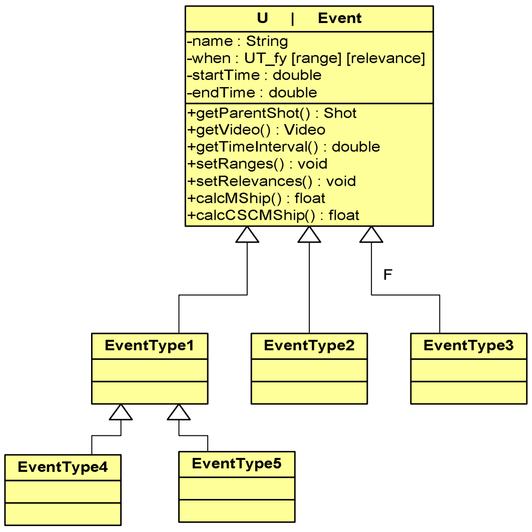


Figure 3.12: The Representation of Event Inheritance Hierarchy

Relations among events are represented in Figure 3.13. There is a many-to-many relationship from Event class to itself. EventToEventRelation is an association class (represented with a dashed line) and it implements the interface TemporalRelaton, which includes the methods corresponding to the temporal relations, used in this study. The representation of the TemporalRelation interface is shown in Figure 3.14. The interfaces SpatialRelation and TemporalRelation are similar to “SpatialObject” and “TemporalObject” of the VIDEX model [17].

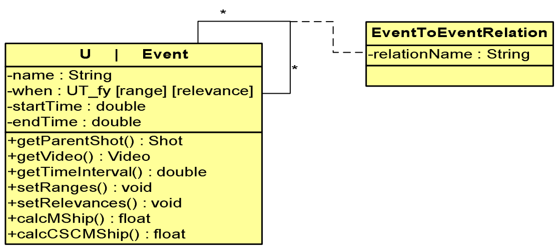


Figure 3.13: The Representation of EventToEventRelation Association Class

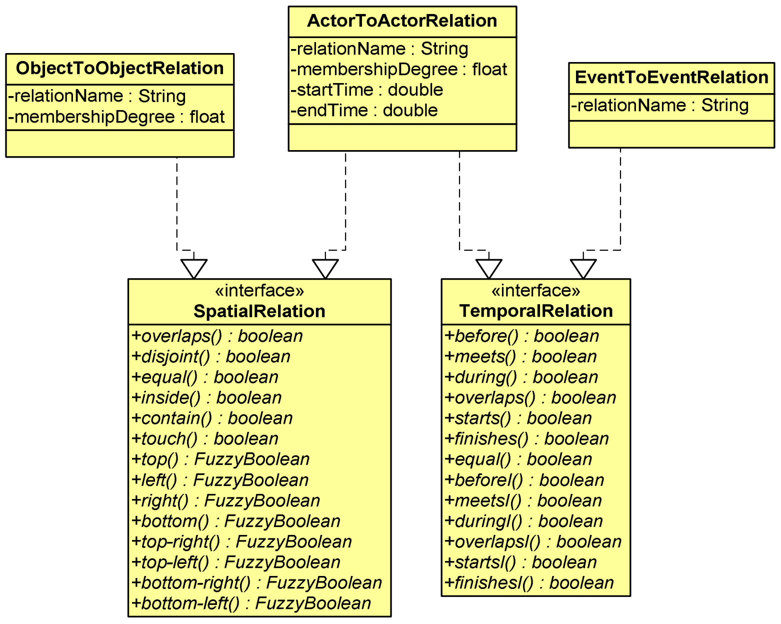


Figure 3.14: TemporalRelation and SpatialRelation Interfaces

In our model, there is also an aggregation between the classes Video and Event to provide accessing the events of a video more easily as represented in Figure 3.15.

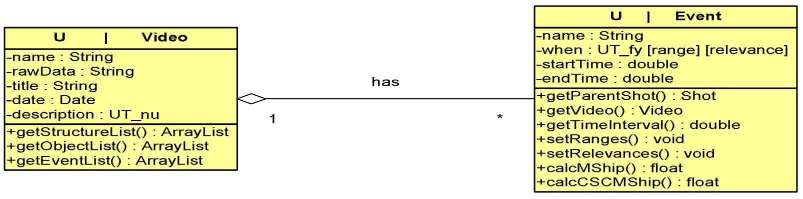


Figure 3.15: The Representation of the Relationship between Video and Event

Object

A video stream has objects, which correspond to meaningful semantic entities. An object may be a ball, a player, a tree, a person, etc. The class Object represents the video objects. It may have crisp or uncertain attributes reflecting event-independent properties of a video object. Event-specific attributes are stored in Actor entity initially defined in [6] and it will be explained in the next section in detail. The relationship between classes Video and Object is represented in Figure 3.16.

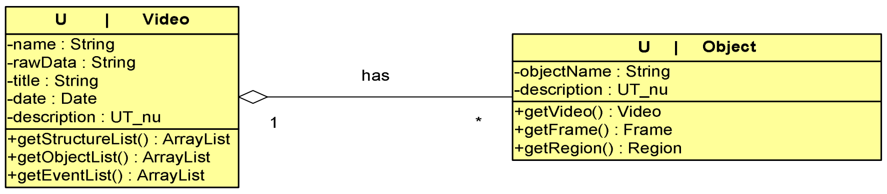


Figure 3.16: The Representation of the Relationship between Video and Object

For a specific application domain, the class Object may be extended and many domain specific subclasses may be defined. The inheritance hierarchy of the class Object is represented in Figure 3.17. The hierarchy can be extended and there may be multiple levels in the class hierarchy.

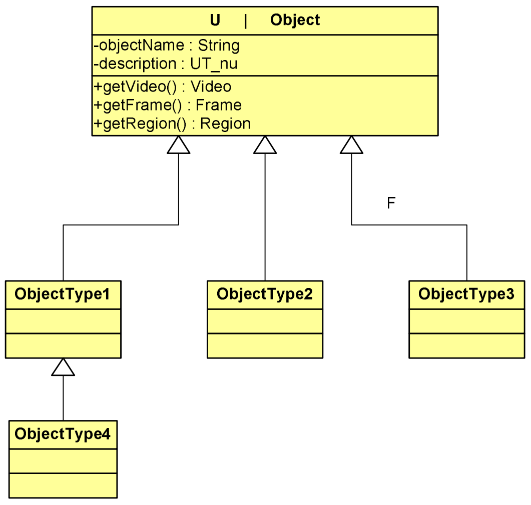


Figure 3.17: The Representation of Object Inheritance Hierarchy

Relations may exist among instances of Object. There are two types of relations among instances of Object in our model.

1- Semantic relations

These are the relations among instances of Object class, which are not event-specific, such as mother-of, friend-of, husband-of, etc…

2- Spatial relations

Spatial relations between instances of Object are only handled for image data. The spatial relations between Object instances in a video event (in other words between actors) are stored in ActorToActorRelation. An Object instance has location information within a video frame. Spatial relations occur according to the locations of two video objects with respect to each other. The notation of spatial relations, which was used in [13], is also used in our study. Spatial relations may be directional and topological.

Directional relations are strict directional relations (*top, left, right*, *bottom*) and mixed-directional relations (*top-right, top-left, bottom-right, bottom-left*).

Topological relations are *equal, inside, disjoint, touch, overlaps,* and *contain*.

Relations among Object instances are represented in Figure 3.18. There is a many-to-many relationship from Object class to itself. ObjectToObjectRelation is an association class and it implements the interface SpatialRelaton, which includes the methods corresponding to the spatial relations, used in this study. The type "FuzzyBoolean", which is used in SpatialRelation interface, is taken from the work [12]. The methods defined in SpatialRelation interface returns values in the range 0.0 and 1.0, representing fuzziness, which might occur in spatial relations.

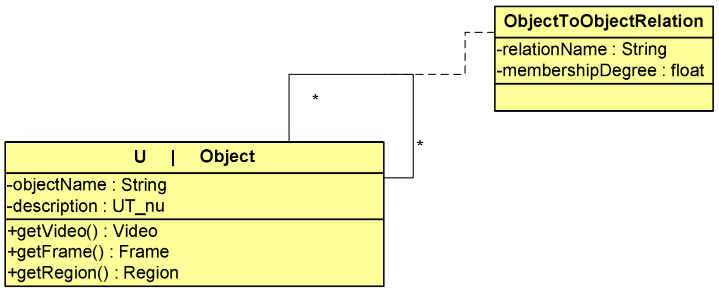


Figure 3.18: The Representation of Object class

The representation of the SpatialRelation interface is given in Figure 3.14 and the representation of the class, which we define for the type FuzzyBoolean, is given in Figure 3.19. Since a spatial relation may be fuzzy, there is a membership degree associated with the relation. The attribute *membershipDegree* in the class ObjectToObjectRelation is used to represent this membership degree.

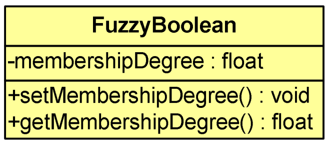

FuzzyBoolean
-membershipDegree : float
+setMembershipDegree() : void
+getMembershipDegree() : float

Figure 3.19: The Representation of the Class FuzzyBoolean

Actor

Video objects may be involved in different events. In each event an object is involved, it becomes an actor and takes semantic and linguistic roles. The concept of actor entity was introduced in [6]. The concept is integrated to our model. Video objects may have event-specific roles. For example, a video object may be a "news speaker" in one event and an "interviewer" in another event. Therefore, an object has a role, which is event-specific. Event-specific properties are stored in Actor entities in our model. There is an association between Event and Actor classes and there is an association between Actor and Object classes. An event may have more than one actor. An object may be involved in more than one event. These relationships are shown in Figure 3.20.

There are two attributes in Actor entity: *lingRole* and *semRole* as it is defined in [6]. *semRole* means the semantic role of an object in an event such as: scorer, kicker, etc. *lingRole* means the linguistic role of the object in an event. There are three linguistic roles: agent, object and recipient. Agent is doing the event, the object is the affected one from the event, and recipient is the indirect one. These linguistic roles are taken from Semantic DS of MPEG-7 [24].

Relations may exist among actors. There might be three types of relations among actors in our model:

1- Semantic Relations

Semantic relations describe the event-specific semantic interactions between two actors in an event.

2- Spatial Relations

Spatial relations describe the relationship between two actors according to their locations in the time interval of the event they occur. Spatial relations between actors are event-specific whereas spatial relations among objects are handled within a single video frame.

3- Temporal Relations

Temporal relations describe the relationship between two actors according to their occurrence during the time line.

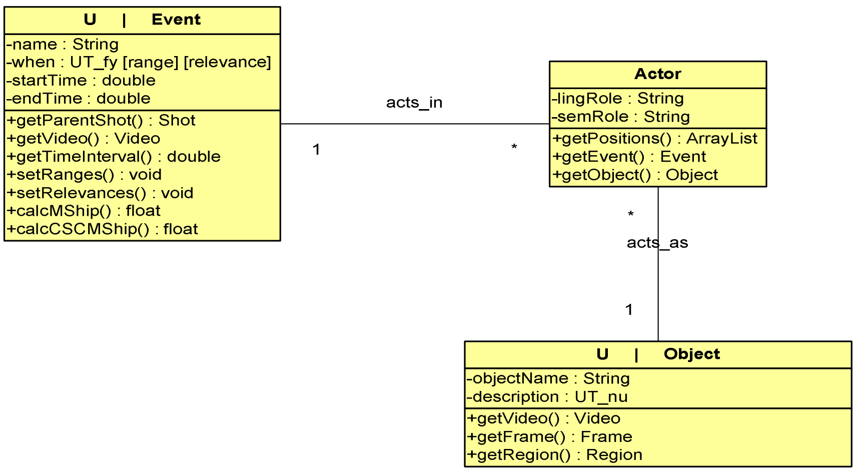


Figure 3.20: The Relationships among Event, Actor and Object

Relations among actors are represented in Figure 3.21. There is a many-to-many relationship from Actor class to itself. ActorToActorRelation is an association class and it implements both of the interfaces SpatialRelaton and TemporalRelation. The representation of this relationship is given in Figure 3.14. Since a spatial relation may be fuzzy, there is a membership degree associated with the relation. The attribute *membershipDegree* in the class ActorToActorRelation is used to represent this membership degree. The attributes *startTime* and *endTime* represent the time interval, which the relation represented with ActorToActorRelation class, holds within.

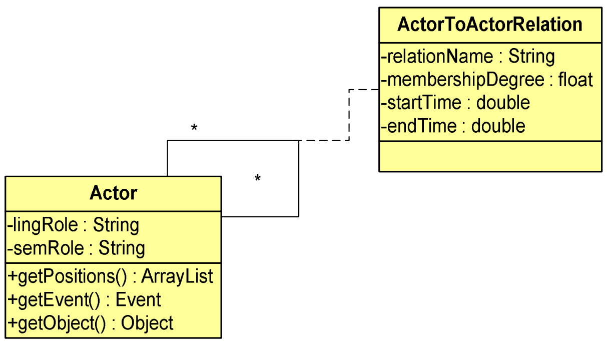


Figure 3.21: The Representation of ActorToActorRelation Association Class

An actor has a spatio-temporal property since it may change its position within the time interval of an event. Therefore, an actor has trajectory information in an event. To represent trajectory information, we define an aggregation between Actor and Position classes. The class Position indicates the position of an actor in a specific time interval within an event. Position class is associated with Region class. The Region class indicates the MBR (minimum bounding rectangle) of the actor in a specific time interval. The collection of Position instances gives the trajectory of the actor in a specific event. These relationships are represented in Figure 3.22. We store the trajectory information in Actor entity not in Object entity because the trajectory information is event-specific. The approach we use to store the trajectory information is similar to the one in [3] where the authors define a class named “MovingRegion” to store the trajectory information of a video object. “MovingRegion” aggregates “Location” which is the minimum bounding rectangle of an object.

A video shot consists of sequential video/audio frames. Therefore, there is a sequence relationship between Shot and Frame classes. Each frame has image and audio signal properties. Image specific properties are defined in the class Image and audio signal specific properties are defined in the class AudioSignal. These relationships are represented in Figure 3.23. The attributes of the classes Image and AudioSignal are taken from the study [11].

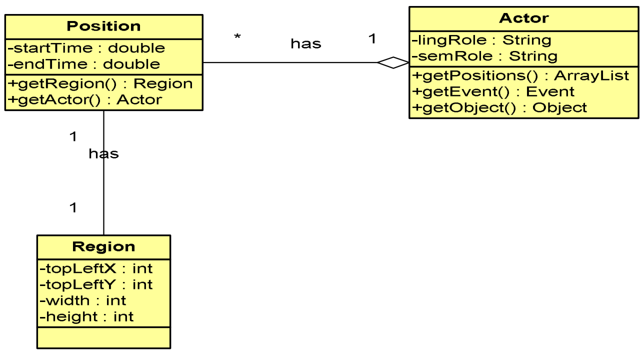


Figure 3.22: The Representation of the Actor Trajectory Information

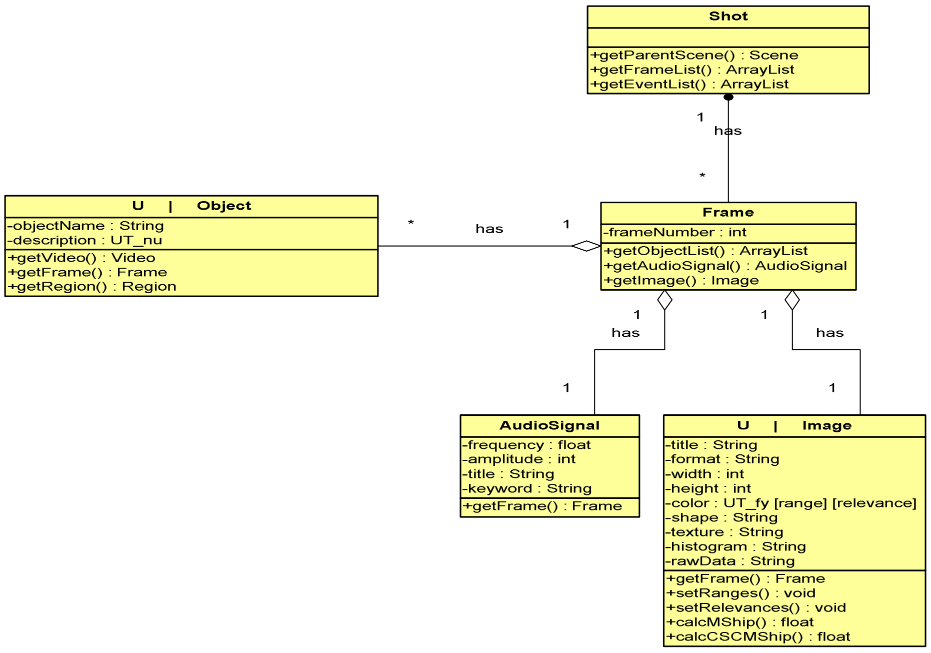


Figure 3.23: The Relationships between Shot, Frame, Image and Signal

A video frame has video objects associated with it. Therefore, an aggregation relationship is defined between Frame and Object as represented in Figure 3.23. The location information of an Object instance within a Frame is stored in a Region instance; thus, there is also an association between Object and Region. Spatial relations among the Object instances within a frame can be found by comparing their associated Region instances.

Finally, Figure 3.24 represents the complete generic model we propose for video database applications.

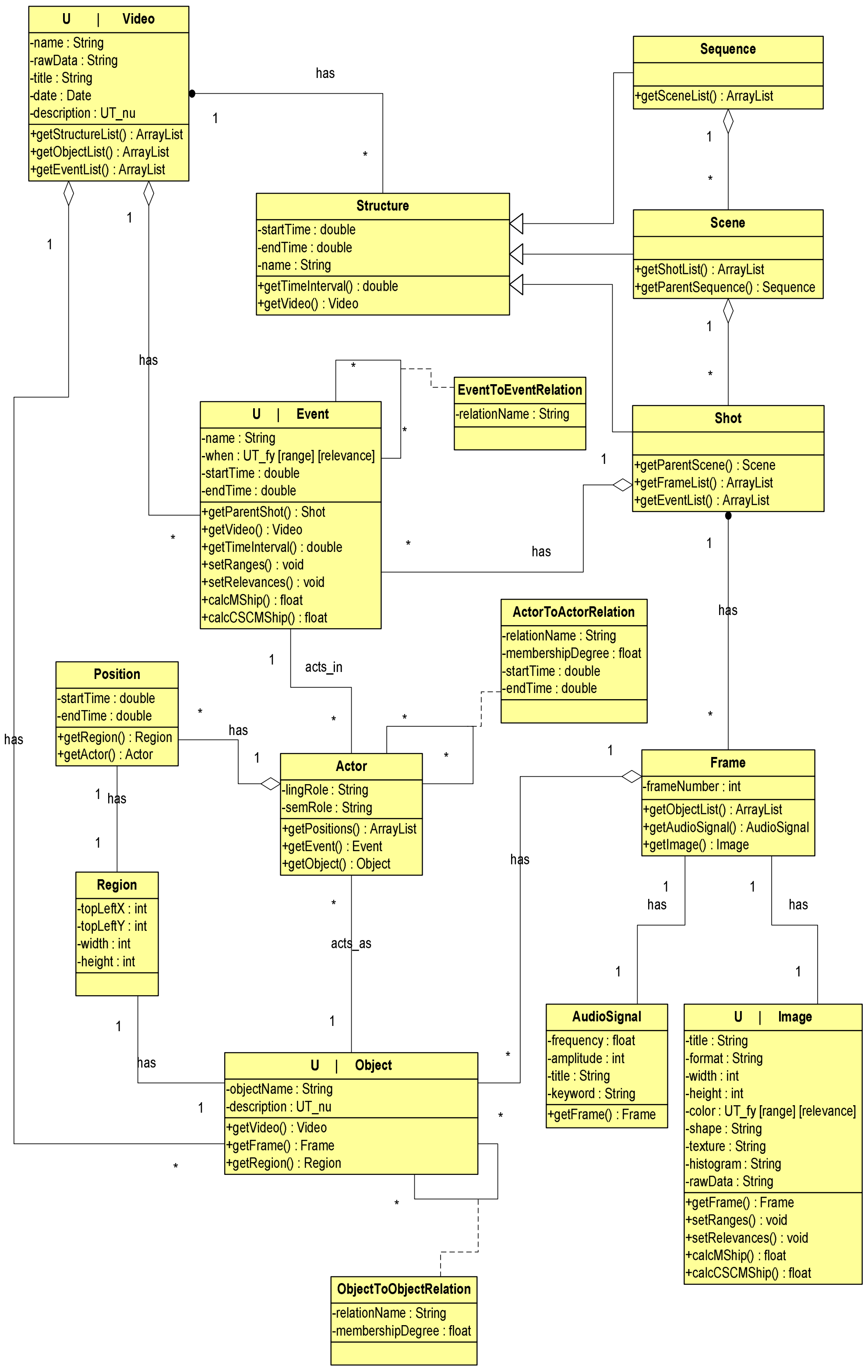


Figure 3.24: The Generic Conceptual Video Data Model Represented with the Extended UML Class Diagram

3.4 Mapping to FOOD

In this section, we give examples to mapping our conceptual model to a logical model. FOOD (fuzzy object-oriented data model) [14] is used as the logical model. The uncertain classes extend the class Fuzzy that provides basic methods “calcMShip” and “calcCSCMship” to calculate object membership degrees and class/subclass membership degrees respectively. The definition of the class Fuzzy is represented in Figure 3.25. Figure 3.26 represents the classes extending the class Fuzzy.

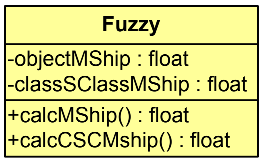


Figure 3.25: The Representation of the Class Fuzzy

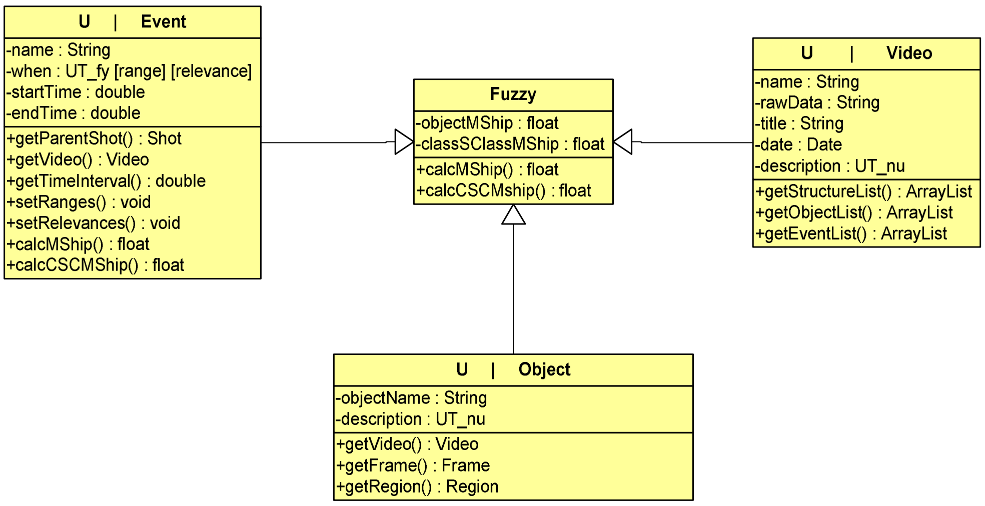
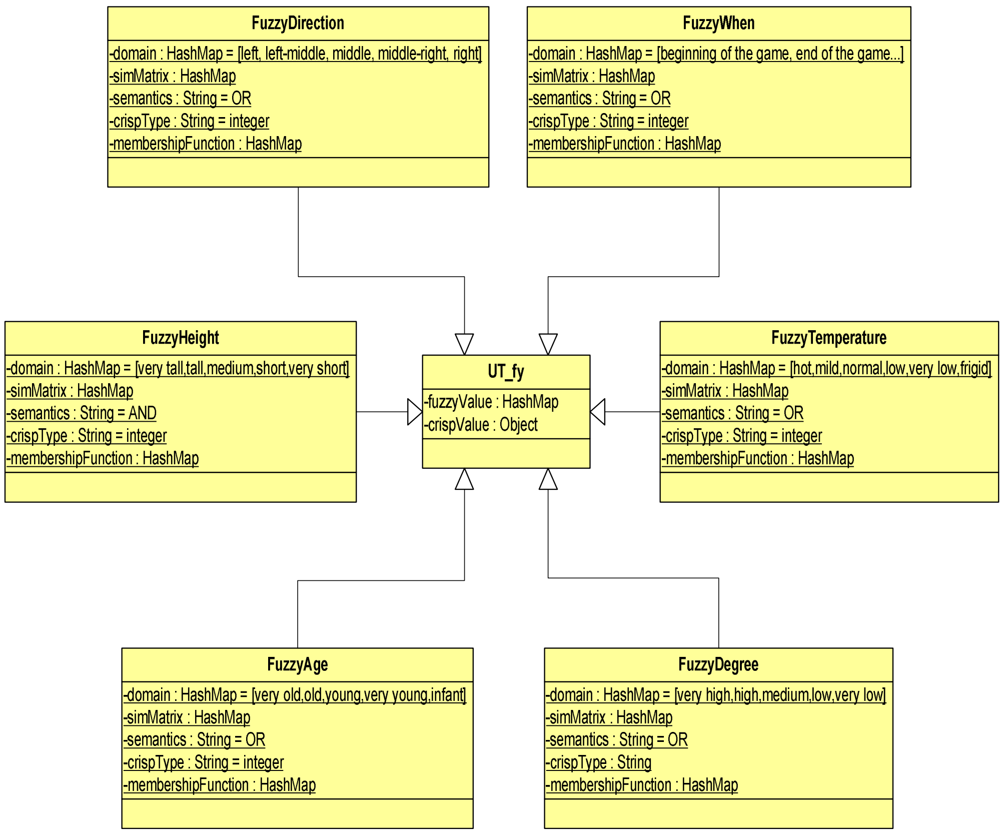


Figure 3.26: The Classes Extending the Class Fuzzy

New classes extending UT_fy are defined to store information specific to a fuzzy domain such as FuzzyTemperature, FuzzyAge, FuzzyHeight, FuzzyDirection, FuzzyWhen and FuzzyDegree. The representation of these classes is given in Figure 3.27. These classes store domain, similarity matrix, semantics, crisp type and membership function information related to a fuzzy domain. UT_fy is used for an abstraction between these classes.

The linguistic values in a fuzzy domain are called fuzzy terms. Similarity matrix is used to define similarities between every pair of the elements in a fuzzy domain. To find the inclusion of an attribute's value to a fuzzy set, inclusion formulas in [14] are used. Different inclusion formulas are employed depending on the semantics of a fuzzy attribute which might be one of AND, OR, XOR semantics. The attributes *domain*, *simMatrix*, *semantics, crispType and membershipFunction* are defined static, since they are class attributes and their values are the same for each instance of a class. It is meaningful that similarity matrix, semantics, domain and memberhsipFunction information is defined by domain experts. Example class definitions written in Java language are provided in Appendix A.



FuzzyDirection
-domain : HashMap = [left, left-middle, middle, middle-right, right]
-simMatrix : HashMap
-semantics : String = OR
-crispType : String = integer
-membershipFunction : HashMap
FuzzyWhen
-domain : HashMap = [beginning of the game, end of the game...]
-simMatrix : HashMap
-semantics : String = OR
-crispType : String = integer
-membershipFunction : HashMap
FuzzyHeight
-domain : HashMap = [very tall,tall,medium,short,very short]
-simMatrix : HashMap
-semantics : String = AND
-crispType : String = integer
-membershipFunction : HashMap
UT_fy
-fuzzyValue : HashMap
-crispValue : Object
FuzzyTemperature
-domain : HashMap = [hot,mild,normal,low,very low,frigid]
-simMatrix : HashMap
-semantics : String = OR
-crispType : String = integer
-membershipFunction : HashMap
FuzzyAge
-domain : HashMap = [very old,old,young,very young,infant]
-simMatrix : HashMap
-semantics : String = OR
-crispType : String = integer
-membershipFunction : HashMap
FuzzyDegree
-domain : HashMap = [very high,high,medium,low,very low]
-simMatrix : HashMap
-semantics : String = OR
-crispType : String
-membershipFunction : HashMap

Figure 3.27: The Representation of the Classes Defined for FuzzyDomains

3.5 An Application: Football Game Videos

We extend our generic model for a specific application domain: football game videos. Some of the classes are somewhat similar to the ones used in [20], which is designed for SMOOTH soccer application [18, 19]. In our generic model, we provide the base classes Event and Object, and then we extend them for the application domain, football games by adding new football game content specific classes.

Figure 3.28 represents the Event hierarchy for the application domain, football games. There are six classes extending Event: ShotEvent, Goal, Corner, Foul, Whistle and Penalty. The class FreeKick extends the class ShotEvent. The inheritance relationship between FreeKick and ShotEvent is fuzzy reflecting the fact that not every free kick is a shot in a football game. Some free kicks may be a pass.

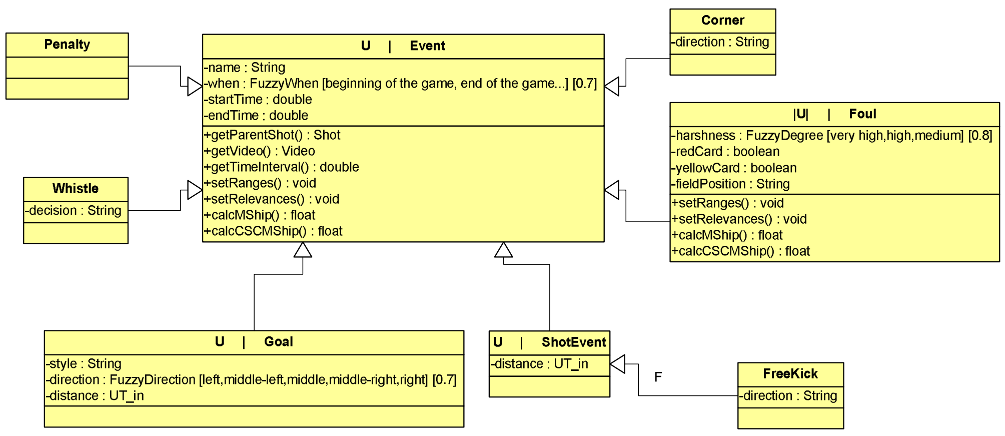


Figure 3.28: The Classes Which Inherit from Event in a Football Game

The range of the fuzzy attribute *when* in Event class is [beginning of the game, mid of the first half, end of the first half, beginning of the second half, mid of the second half, end of the game]. The *direction* attribute in the class Goal represents the information about the direction from which the goal is scored, which might take values from the range [left, left-middle, middle, middle-right, right]. The range values of the attribute *when* are taken from the model introduced in [11].

The class Foul has “|U|” notation indicating that it is uncertain at the object level. As it is represented in Figure 3.29, there are three instances of the class Foul: Kicking, Pushing and Hitting. Kicking is the instance of the Foul class with a membership degree of 0.9, Pushing is the instance of the Foul class with a membership degree of 0.2 and Hitting is the instance of the Foul class with a membership degree of 0.2. The value of the fuzzy attribute *harshness* decides the membership degree of these instances to the class Foul.

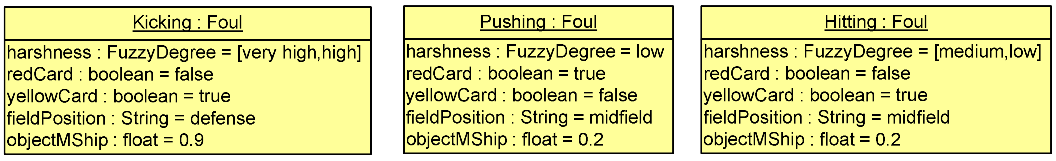

Kicking : Foul
harshness : FuzzyDegree = [very high,high]
redCard : boolean = false
yellowCard : boolean = true
fieldPosition : String = defense
objectMShip : float = 0.9
Pushing : Foul
harshness : FuzzyDegree = low
redCard : boolean = true
yellowCard : boolean = false
fieldPosition : String = midfield
objectMShip : float = 0.2
Hitting : Foul
harshness : FuzzyDegree = [medium,low]
redCard : boolean = false
yellowCard : boolean = true
fieldPosition : String = midfield
objectMShip : float = 0.2

Figure 3.29: The Instances of Foul Class with Different Membership Degrees

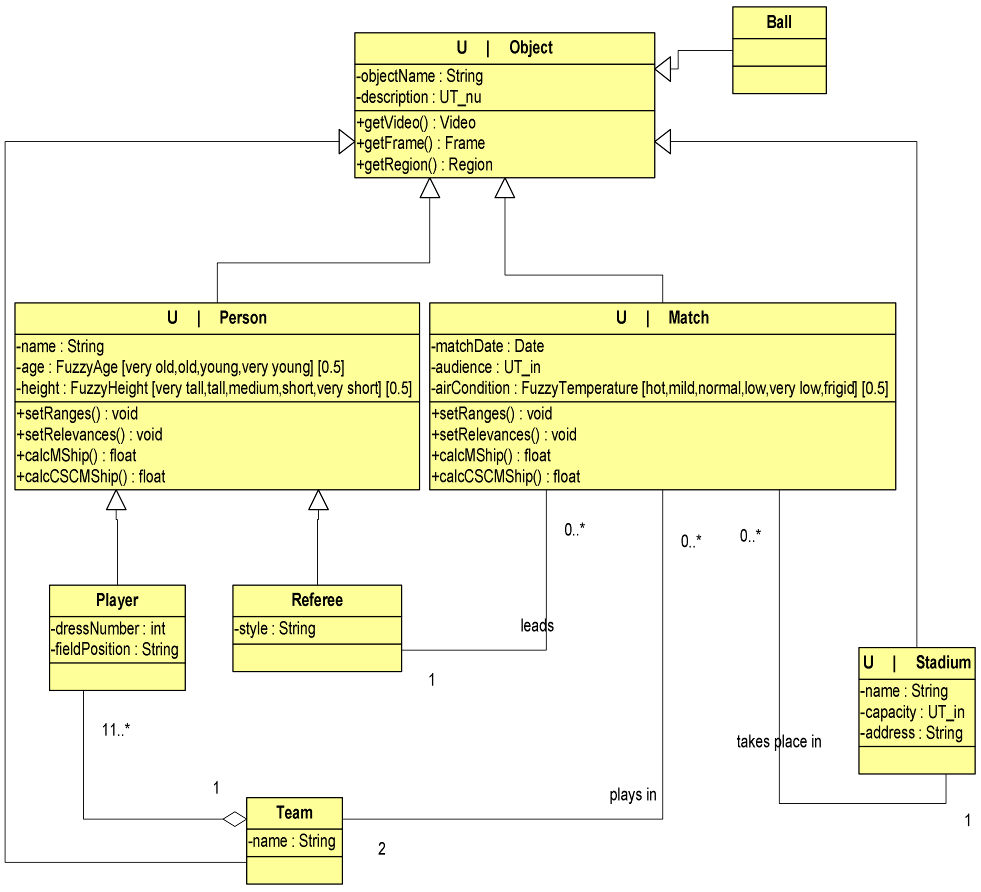
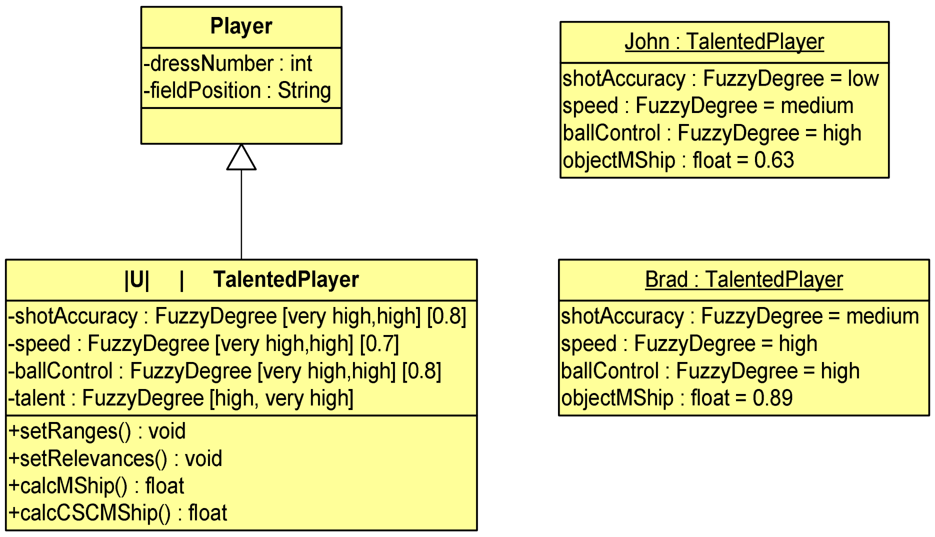


Figure 3.30: The Classes Which Inherit from Object in a Football Game

Figure 3.30 represents the Object hierarchy for the application domain, football games. There are five classes extending the class Object: Person, Match, Ball, Team and Stadium. Person, Match and Stadium are uncertain classes, whereas Ball and Team are crisp classes. The class Person has two uncertain attributes *age* and *height* of type fuzzy. The class Match has two uncertain attributes: *audience* of type incomplete and *airCondition* of type fuzzy. There are two classes extending the class Person: Player and Referee. There is an association between Referee and Match named "leads". There is an aggregation relationship between Team and Player meaning that a team consists of 11 and more players. There is an association between Team and Match named "plays in". There is an association between Match and Stadium named "takes place in".

Figure 3.31 represents an example to the object/class level uncertainty for the football games domain. The class TalentedPlayer, which extends the class Player, is uncertain at the object/class level. The values of the uncertain attributes *shotAccuracy*, *speed* and *ballControl* decide the membership degree of an object to TalentedPlayer class. The objects "John" and "Brad" are the instances of the class TalentedPlayer with the membership degrees 0.63 and 0.89 respectively. The attribute *talent* is a derived fuzzy attribute and its value is not stored explicitly but derived according to the values of the attributes *shotAccuracy*, *speed* and *ballControl* by processing a fuzzy rule. Fuzzy rules will be explained in Chapter 3.



Player
-dressNumber : int
-fieldPosition : String
|U| | TalentedPlayer
-shotAccuracy : FuzzyDegree [very high,high] [0.8]
-speed : FuzzyDegree [very high,high] [0.7]
-ballControl : FuzzyDegree [very high,high] [0.8]
-talent : FuzzyDegree [high, very high]
+setRanges() : void
+setRelevances() : void
+calcMShip() : float
+calcCSCMShip() : float
John : TalentedPlayer
shotAccuracy : FuzzyDegree = low
speed : FuzzyDegree = medium
ballControl : FuzzyDegree = high
objectMShip : float = 0.63
Brad : TalentedPlayer
shotAccuracy : FuzzyDegree = medium
speed : FuzzyDegree = high
ballControl : FuzzyDegree = high
objectMShip : float = 0.89

Figure 3.31: The Representation of the Class TalentedPlayer

We apply our model for an imaginary football game video, which is played between Fenerbahçe and Beşiktaş. Figure 3.32 represents the sequences and the scenes occurring in a specific football game video by utilizing UML object diagrams. The first sequence is “1st half of the game” and the second sequence is the “2nd half of the game”. The first half of the game has two scenes: “1st goal of the game” and “2nd goal of the game”.

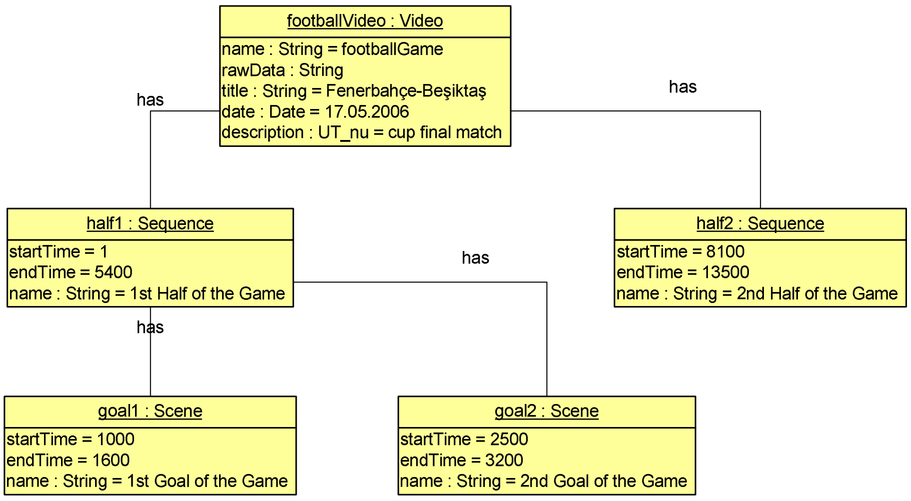


Figure 3.32: Representation of Sequences and Scenes of a Football Game Video

The details of the scene “1st goal of the game” are represented in Figure 3.33 by utilizing UML object diagrams. There are two shots in the scene: *corner* and *goal1*. The shot *corner* has only one event: *cornerEvent* and the shot *goal1* has only one event: *goalEvent*. There are two actors in *cornerEvent*: *kicker* and *kicked1*. *Kicker* is associated with a Player object and *kicked1* is associated with a Ball object. There are three actors in *goalEvent*: *kicked2*, *scorer1* and *goalkeeper1*. *Kicked2* is associated with a Ball object, *scorer1* is associated with a Player object and *goalkeeper1* is associated with a Player object.

The details of the scene “2nd goal of the game” are represented in Figure 3.34 by utilizing UML object diagrams. There are two shots in the scene: *foul* and *penalty*. The shot *foul* has only one event: *foulEvent* and the shot *penalty* has only one event: *penaltyEvent*. There are two actors in *foulEvent*: *committer* and *victim*. *Committer* is associated with a Player object and *victim* is associated with a Player. There are three actors in *penaltyEvent*: *kicked3*, *scorer2* and *goalkeeper2*. *Kicked3* is associated with a Ball object, *scorer2* is associated with a Player object and *goalkeeper2* is associated with a Player object.

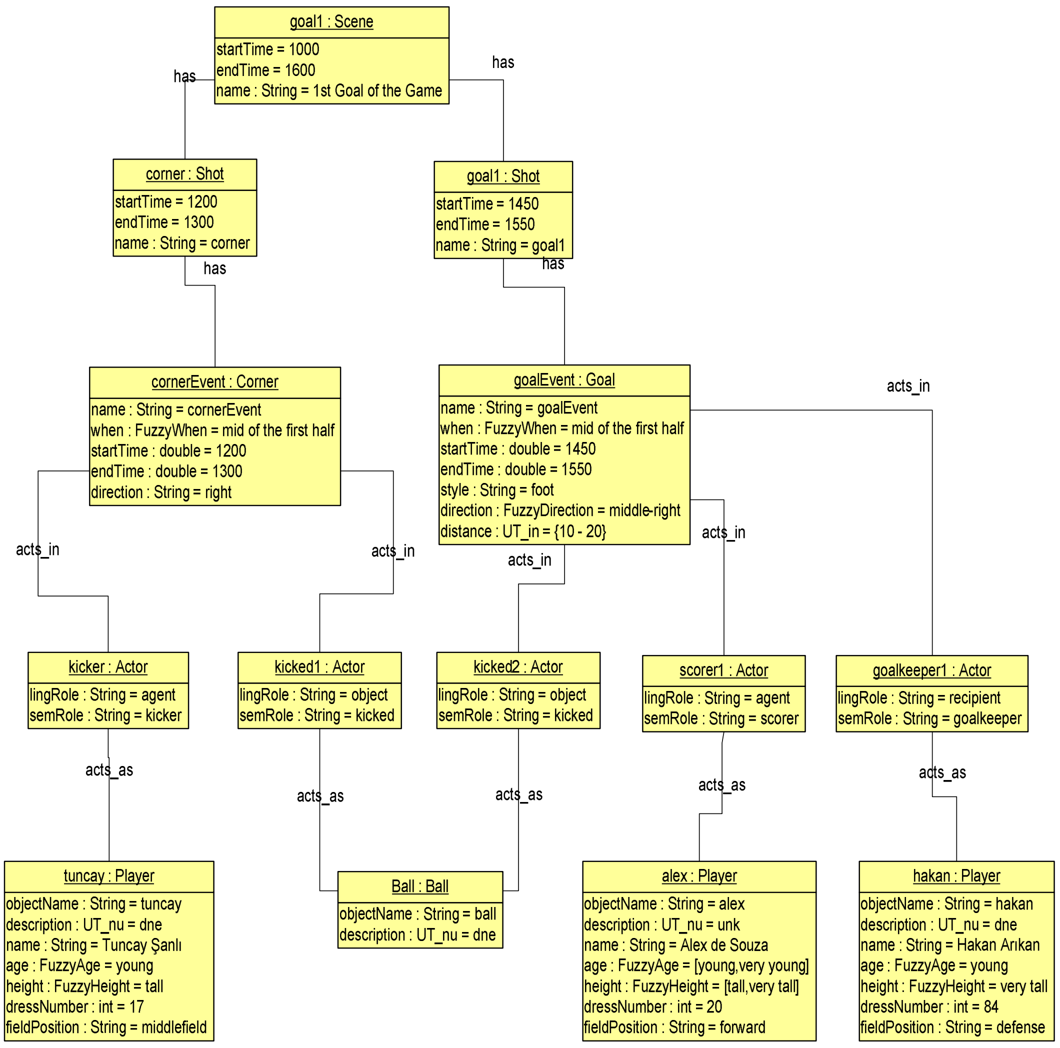

goal1 : Scene
startTime = 1000
endTime = 1600
name : String = 1st Goal of the Game
has
has
corner : Shot
startTime = 1200
endTime = 1300
name : String = corner
goal1 : Shot
startTime = 1450
endTime = 1550
name : String = goal1
has
has
cornerEvent : Corner
name : String = cornerEvent
when : FuzzyWhen = mid of the first half
startTime : double = 1200
endTime : double = 1300
direction : String = right
goalEvent : Goal
name : String = goalEvent
when : FuzzyWhen = mid of the first half
startTime : double = 1450
endTime : double = 1550
style : String = foot
direction : FuzzyDirection = middle-right
distance : UT_in = {10 - 20}
acts_in
acts_in
acts_in
acts_in
acts_in
acts_in
kicker : Actor
lingRole : String = agent
semRole : String = kicker
kicked1 : Actor
lingRole : String = object
semRole : String = kicked
kicked2 : Actor
lingRole : String = object
semRole : String = kicked
scorer1 : Actor
lingRole : String = agent
semRole : String = scorer
goalkeeper1 : Actor
lingRole : String = recipient
semRole : String = goalkeeper
acts_as
acts_as
acts_as
acts_as
acts_as
tuncay : Player
objectName : String = tuncay
description : UT_nu = dne
name : String = Tuncay Şanlı
age : FuzzyAge = young
height : FuzzyHeight = tall
dressNumber : int = 17
fieldPosition : String = middlefield
Ball : Ball
objectName : String = ball
description : UT_nu = dne
alex : Player
objectName : String = alex
description : UT_nu = unk
name : String = Alex de Souza
age : FuzzyAge = [young,very young]
height : FuzzyHeight = [tall,very tall]
dressNumber : int = 20
fieldPosition : String = forward
hakan : Player
objectName : String = hakan
description : UT_nu = dne
name : String = Hakan Arıkan
age : FuzzyAge = young
height : FuzzyHeight = very tall
dressNumber : int = 84
fieldPosition : String = defense

Figure 3.33: Representation of the Scene "1st goal of the game"

3.6 Querying the Model

The users of video databases might construct various queries such as (fuzzy) semantic, hierarchical, temporal, (fuzzy) spatial, regional, and trajectory queries. In this section, we give examples to these queries and explain how our conceptual model supports these queries.

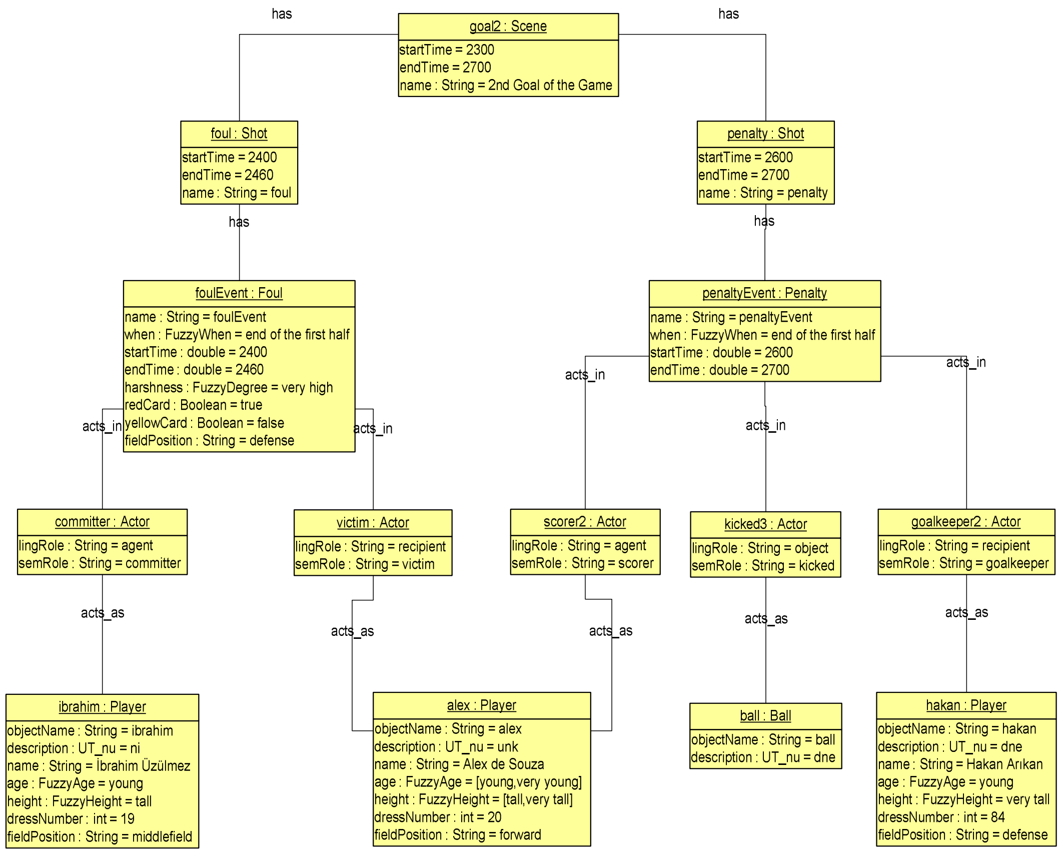

has
goal2 : Scene
startTime = 2300
endTime = 2700
name : String = 2nd Goal of the Game
has
foul : Shot
startTime = 2400
endTime = 2460
name : String = foul
penalty : Shot
startTime = 2600
endTime = 2700
name : String = penalty
has
has
foulEvent : Foul
name : String = foulEvent
when : FuzzyWhen = end of the first half
startTime : double = 2400
endTime : double = 2460
harshness : FuzzyDegree = very high
redCard : Boolean = true
yellowCard : Boolean = false
fieldPosition : String = defense
penaltyEvent : Penalty
name : String = penaltyEvent
when : FuzzyWhen = end of the first half
startTime : double = 2600
endTime : double = 2700
acts_in
acts_in
acts_in
acts_in
acts_in
committer : Actor
lingRole : String = agent
semRole : String = committer
victim : Actor
lingRole : String = recipient
semRole : String = victim
scorer2 : Actor
lingRole : String = agent
semRole : String = scorer
kicked3 : Actor
lingRole : String = object
semRole : String = kicked
goalkeeper2 : Actor
lingRole : String = recipient
semRole : String = goalkeeper
acts_as
acts_as
acts_as
acts_as
acts_as
ibrahim : Player
objectName : String = ibrahim
description : UT_nu = ni
name : String = İbrahim Üzülmez
age : FuzzyAge = young
height : FuzzyHeight = tall
dressNumber : int = 19
fieldPosition : String = middlefield
alex : Player
objectName : String = alex
description : UT_nu = unk
name : String = Alex de Souza
age : FuzzyAge = [young,very young]
height : FuzzyHeight = [tall,very tall]
dressNumber : int = 20
fieldPosition : String = forward
ball : Ball
objectName : String = ball
description : UT_nu = dne
hakan : Player
objectName : String = hakan
description : UT_nu = dne
name : String = Hakan Arıkan
age : FuzzyAge = young
height : FuzzyHeight = very tall
dressNumber : int = 84
fieldPosition : String = defense

Figure 3.34: Representation of the Scene "2nd goal of the game"

3.6.1 (Fuzzy) Semantic Queries

Semantic queries might be about the properties of events, objects and about the semantic relations between them. We can divide semantic queries to four categories as follows:

1- Querying about video properties such as:

- What is the date of the video played between "Galatasaray" and "Fenerbahçe":

  To answer this query, simply the matches related to each video are searched. If the teams playing in a match are "Galatasaray" and "Fenerbahçe", then the value of the *date* attribute of that video is the result.

- Find the videos having the title "2007 Champions League Final Match":

To answer this query, *title* attribute of each video is compared to the title specified in the query. The list of the videos satisfying the condition is the result.

2- Querying about event properties such as:

- Find the goal events happening between 10th and 100th seconds

  To answer this query, *startTime* and *endTime* attributes of each goal event are compared with the conditions specified in the query. The list of the goal events satisfying the condition is the result.

- Find the foul events with red cards

  To answer this query, *redCard* attribute of each foul event is compared to “true”. The list of the foul events satisfying the condition is the result.

- Find all fouls occurring between 10th and 200th seconds which can be considered as very harsh (very high degree of harshness) with a threshold of 0.7

  To answer this query, Foul events having the time interval within the given one are searched. For each foul event, inclusion degree of the value of *harshness* attribute to the fuzzy set [very high] is calculated. If it is greater than the threshold, that foul event is included in the result.

- Find the goal events which are scored by foot

  To answer this query, *style* attribute of each goal event is compared to “foot”. The list of the goal events satisfying the condition is the result.

- Find the goal events which are scored from the right side of the goalpost with a threshold of 0.6

  Here, *direction* is a fuzzy attribute in Goal class. To answer this query, inclusion degree of the value of *direction* attribute to the fuzzy set [right] is calculated. If the inclusion degree is greater than the threshold value, then that goal event is included in the result.

3- Querying about object properties such as:

- Find the players who are young and, very tall or tall with a threshold of 0.7

  Here, *age* and *height* are fuzzy attributes in Player class (inherited from Person class). To answer this query, inclusion degree of the value of *age* attribute to the fuzzy set [young] and inclusion degree of the value of *height* attribute to the fuzzy set [very tall OR tall] are calculated. If the inclusion degrees are greater than the threshold value, then that player object is included in the result.

- Find the matches played in a very cold weather with a threshold of 0.5

  Here, *airCondition* is a fuzzy attribute in Match class. To answer this query, inclusion degree of the value of *airCondition* attribute to the fuzzy set [very cold] is calculated. If the inclusion degree is greater than the threshold value, then that match object is included in the result.

- Find the matches played in a stadium having a capacity of {10000 - 20000} with a threshold of 0.5

  Here, *capacity* is an incomplete attribute in Stadium class. To answer this query, inclusion degree of the value of *capacity* attribute of Stadium objects associated with each Match object to the range {10000 - 20000} is calculated. If the inclusion degree is greater than the threshold value, then that Match object is included in the result list.

4- Querying about events and actors such as:

- Find the time interval in which a goal is scored with a distance {10m – 20m} from the goalpost at the end of the game with a threshold of 0.7

  Here, *distance* is an incomplete attribute and *when* is a fuzzy attribute (inherited from Event class) in Goal class. To answer this query, for each goal event, inclusion degree of the value of *distance* attribute to the range {10m – 20m} and inclusion degree of the value of *when* attribute to the fuzzy set [at the end of the game] are calculated. If the degrees are greater than the threshold, time interval of that goal event is included in the result.

- Find all talented players (with a threshold of 0.8) which are seen between 10th and 200th seconds

Events, which have the time interval within 10th and 200th seconds, are searched. For each found event, the TalentedPlayer objects, which are involved in that event, are found by using the actor list of the event. TalentedPlayer objects having object membership degrees greater than the threshold are included in the result.

- Find all corner events in which both of playerX and playerY are seen

  To answer this query, actors of the corner events are searched. If the actors of a corner event include the objects playerX and playerY, then that corner event is included in the result.

- Find all the other players in goal event in which playerX is the scorer

  To answer this query, actors of each goal event are searched. If the actors of a goal event include the object playerX with the semantic role “scorer”, then the player objects associated with all of the actors of that goal event are included in the result.

- Find the events in which playerX commits a foul against playerY

  Actors involved in foul events are searched. If the actors of a foul event include the objects playerX with semantic role “committer” and playerY with semantic role “victim”, that foul event is included in the result.

- Find the goals made against teamA in a match played in a very cold weather with audience from 10000 to 30000 people with a threshold of 0.6

  Matches are searched and the inclusion degree of the value of *airCondition* attribute to the fuzzy set [very cold] and the inclusion degree of the value of *audience* attribute to the range {10000 - 30000} are calculated. If the degrees are greater than the threshold, videos, which include those matches, are searched. For each goal event in the found videos, if the goal event involves a player with semantic role “goalkeeper” who is a member of teamA, that goal event is included in the result.

- Find all objects in the goal event which occurs between 10th and 200th seconds in the video

To answer this query, the goal events of the given video having a time interval within the specified time interval are selected. The actor list of the goal event is traversed. The related player objects are included in the result.

3.6.2 Hierarchical Queries

As mentioned before, our model supports the representation of the hierarchical structure of video data. Therefore, hierarchical structure of video content can be queried in the form of hierarchical queries such as:

- Find the goal scenes appearing in the “first half sequence” of the game

  To answer this query, Structure list of each video is searched. If the name of the sequence is “first half sequence”, then the scene list of that sequence is searched. The scenes with the name “goal” are included in the result.

- Find the corner shots appearing in the “1st goal scene”

  To answer this query, the Structure (sequence) list of each video is searched. Then the scene list of each sequence is searched. If the name of the scene is “1st goal scene”, then the shot list of that scene is searched. The shots having the name “corner” are included in the result.

- Find the events in the corner shot of the video “Fenerbahçe-Galatasaray”

  To answer this query, Scene list of each sequence, which is in the Structure list of the given video, is searched. For each found scene, if the shot list of the scene includes a shot with name “corner”, the event list of that shot is the result.

3.6.3 Temporal Queries

Video events might have temporal relations between each other. The temporal relations are represented by the instances of EventToEventRelation class. Therefore, we can construct temporal queries to find the temporal relations between events as follows:

- Find the foul events occurring before goal events

To answer this query, the foul events and the goal events are found. Then the temporal relation between each foul event – goal event pair is calculated. If the relation name is “before”, then the related foul event is included in the result.

- Find the temporal relation between the corner event and the goal event occurring between 10th and 200th seconds in the video “Fenerbahçe-Galatasaray”

  To answer this query, firstly the goal and foul events of the given video are searched. If the time intervals of the events are within the given time interval, the temporal relation is found between the goal and foul events. The result is returned in the form of an EventToEventRelation instance.

- Find the goals which are scored after a corner shoot which is made by playerX

  To answer this query, firstly the actors of corner events are searched. If the actor list of a corner event includes playerX with the linguistic role “agent”, then that corner event is selected. The goal events in the video of the selected corner events are found. Then the temporal relation between each corner event – goal event pair is calculated. If the relation name is “before”, then the related goal event is included in the result.

- Find the penalties which happen after a foul committed by a player with name “Okan”, and of young age with a threshold of 0.7

  To answer this query, firstly the foul events are searched. The events having a player with name “Okan” as an actor with the linguistic role “agent” are selected. The inclusion degree of the value of the *age* attribute of the player to the fuzzy set [young] is calculated. If the degree is smaller than the threshold, that foul event is ignored. Then the penalty events in the same video of the found foul events are searched. If the temporal relation between the foul event and penalty event is “before”, that penalty event is included in the result.

3.6.4 (Fuzzy) Spatial Queries

When objects are involved in events, they become actors. Actors might have spatial relations between each other in an event. The spatial relations are represented by the instances of ActorToActorRelation class. Therefore, we can construct spatial queries to find the spatial relations between actors as follows:

- Find the spatial relation between playerX and playerY at the goal event of the video "Fenerbahçe-Galatasaray"

  To answer this query, position lists of the actors (playerX and playerY) participating in the goal event of the given video are searched. If the time intervals of two positions of two different actors are intersected, then the regions are compared to find the spatial relation. The result is returned in the form of an ActorToActorRelation instance.

- Find the player appearing on the left side of playerX in the foul event of the video "Fenerbahçe-Galatasaray" with a threshold of 0.8

  To answer this query, position lists of the actors in the foul event of the given video are searched. If the time intervals of two positions of two different actors are intersected and one of these two actors is playerX, regions are compared to determine whether the spatial relation in the intersected time interval is "left". If the membership degree of the relation is greater than the threshold, the player other than playerX is included in the result.

- Find the time interval (or event) in which playerX and playerY has the spatial relation "overlaps" in the video "Fenerbahçe-Galatasaray"

  To answer this query, firstly the actors related to playerX and playerY, which participate in the same event in the given video, are found. Then the position lists of the actors are searched. If the time intervals of two positions are intersected, then the regions are compared to find the spatial relation. If the relation is "overlaps" and the membership degree of the relation is greater than the threshold, the time interval (or the event) is included in the result.

3.6.5 Regional Queries

We store the region of an actor in a specific time interval in Position class. Position class is associated with Region class, which stores the information for the minimum bounding rectangle of an actor. Our model supports the regional queries defined in [3, 13]. We can construct regional queries as follows:

- Find the time interval in which playerX has the given region with a threshold of 0.9 in the foul event of the video "Fenerbahçe-Galatasaray"

  To answer this query, actors participating in the foul event in the given video are searched. If playerX is found as the actor, position list of the actor is traversed and regions are compared with the given region. For the region matching, the following formula, which is taken from the study in [3], is used:

$$\mu = \text{intersected\_area (Region1, Region2)} / \text{minimum\_area (Region1, Region2)} \qquad \text{(F-1)}$$

  If the membership degree (μ) is greater than the threshold for a region, the related time interval is included in the result.

- Find the regions of the ball in the goal event which is scored from the right side of the field in the video "Fenerbahçe-Galatasaray"

  To answer this query, Goal events in the given video are found. The inclusion degree of the value of the attribute *direction* to the fuzzy set [right] is calculated. If the degree is smaller than the threshold, that goal event is ignored. Actors participating in the found goal event are searched. If ball is found as the actor, position list of the actor is traversed and the regions are included in the result.

3.6.6 Trajectory Queries

Actors follow a path in an event since their position change with respect to time. The path of an actor is stored in a list of Position instances. Our model supports the trajectory queries defined in [3, 13]. We can construct trajectory queries as follows:

- Find the trajectory of playerX in goal event in the video "Fenerbahçe-Galatasaray"

  To answer this query, actors participating in the goal event in the given video are searched. If playerX is found as the actor, then position list of the actor is traversed and regions are included in the result. The result can be shown to the user by drawing all the regions and the path between them to the screen.

- Find the players having the given trajectory (specified with a starting region and an ending region given by query-by-sketch method) in the foul event in the video “Fenerbahçe-Galatasaray” with a threshold of 0.6

  To answer this query, actors participating in the foul event in the given video are searched. For each actor, position list of the actor is traversed. Regions are compared to the given starting and ending region. If matching of the starting region is successful, and the following regions reach to the ending region, the player (related to the actor) is included in the result.

CHAPTER 4

INTELLIGENT FUZZY OBJECT-ORIENTED DATABASE FRAMEWORK

Video data includes very rich and complex semantic information. To store such a huge amount of information could be very space consuming. Moreover, it is not possible to extract complete semantic information automatically or manually from a video. Therefore, mechanisms, which infer information from existing information, are needed for video databases.

Integration or interaction of database and knowledge base systems lead to more powerful and intelligent systems. The studies in the literature, which deal with such an integration, mostly ignore uncertainty. However, uncertainty might occur in many complex applications such as video databases. The previously proposed IFOOD [30] architecture provides a tightly coupled environment between database and knowledge base systems for complex applications by taking into consideration the uncertainty and fuzziness issues. The architecture models data and knowledge including uncertainty, which might occur in many cases.

In this study, an intelligent fuzzy object-oriented database framework for the video database applications is presented. The framework is based on the previously proposed IFOOD [30] architecture. The IFOOD architecture is adapted and implemented for video database applications. The conceptual model, which is introduced in Chapter 3, is mapped to the logical FOOD model [14] and used in the presented framework.

The rest of the chapter is organized as follows: The first section gives a description about the presented framework. The second section focuses on temporal, spatial and fuzzy semantic rules. Finally, the last section discusses the issues related to the integration of FOOD and FKB by giving query evaluation examples.

4.1 Intelligent Fuzzy Object-Oriented Database Framework

IFOOD architecture [30] mainly consists of two parts: a fuzzy object-oriented database (FOOD) and a fuzzy knowledge base (FKB).

In the object-oriented database part, the FOOD model [14] is utilized as the logical data model. In the previous chapter, how our conceptual video model is mapped to the FOOD model was explained. The object-oriented database provides storing the semantic video entities in the form of objects along with their uncertain features.

The fuzzy knowledge base is capable of not only crisp but also fuzzy inference. Fuzzy inference is done by using similarity-matching operations defined in FOOD model. Semantic fuzzy rules representing knowledge about a video database application domain are processed by FKB. Video databases also need handling temporal and spatial knowledge, which are stored in FKB in the form of temporal and spatial rules.

Existing database and knowledge base systems do not have the needed constructs to handle uncertainty. Therefore, new methods should be implemented and integrated to these systems. Fuzzy processors are developed at both of the object-oriented database and knowledge base parts, which handle all of the operations related to uncertainty. The fuzzy processor at the object-oriented database part provides methods for similarity matching, calculation of inclusion degrees, object membership degrees and class/subclass membership degrees. Whereas the fuzzy processor at the knowledge base part provides calculations of similarity matching, rule antecedent, consequent matching needed for the fuzzy inference mechanism.

In the framework, FOOD and FKB are connected through a bridge, which provides interoperability, and data flow between the database and the knowledge base. As it is mentioned before, uncertain classes, fuzzy data types and fuzzy domains, objects with both crisp and fuzzy attributes, similarity matrices and membership functions defined for the fuzzy data types, range and relevance information defined for the fuzzy attributes are stored in the fuzzy object-oriented database. The temporal, spatial and semantic fuzzy rules are stored in the fuzzy knowledge base. The objects, which FKB work on, are stored in FOOD. Therefore, the objects should be transferred to FKB in order to make FKB to process rules on those objects. This transfer is performed by bridge. FKB should also access the information needed for fuzzy inference. Bridge provides the needed information between FOOD and FKB.

There is a user interface above these systems, which is used for annotation and querying of video data. The user interface is connected to the bridge, which gives the system a unified view. The user of this framework does not need to know any detail of the underlying systems. Therefore, they may construct queries including both crisp and fuzzy conditions. Some queries might be so simple that only object-oriented database access would be sufficient whereas some queries might need rule firing. Bridge provides the integration between FOOD and FKB during the query evaluation.

4.2 Temporal, Spatial and (Fuzzy) Semantic Rules

Video data has semantic entities such as events and objects. Video events occur within a time interval; therefore, there exist temporal relations between events. Video objects have different positions in different time intervals; therefore, they have spatial relations between each other. Storing every temporal and spatial relation between the entities is very space consuming. The temporal relations may be inferred from the starting and ending frames of each event and the spatial relations may be inferred from the position information of each object. In this study, temporal and spatial inference rules are defined which are stored in the fuzzy knowledge base.

4.2.1 Temporal Rules

In this study, Allen's temporal interval algebra [23] is used to represent temporal knowledge. Allen defines an interval x as a pair [x-, x+] of time points. x- is the lesser end-point and x+ is the greater end-point assuming that x- < x+. By using this definition, 13 temporal relations, which may hold between two intervals, can be defined: *before, overlaps, during, meets, starts, finishes, equal* and their inverses except *equal*. These thirteen relations are enough to represent the temporal relationship between two intervals.

Table 4.1 represents the temporal relations between two intervals x and y which are defined by using end-points. Table 4.2 represents the temporal relations by showing their meanings.

Table 4.1: Temporal Relations (by using end-points)

Temporal Relation	Definition
before	x+ < y-
overlaps	(x- < y-) and (x+ > y-) and (x+ < y+)
during	(x- > y-) and (x+ < y+)
meets	x+ = y-
starts	(x- = y-) and (x+ < y+)
finishes	(y- < x-) and (x+ = y+)
equal	(x- = y-) and (x+ = y+)

Table 4.2: Temporal relations

Relation	Symbol	Symbol for Inverse	Meaning
X before Y	b	bi	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYYY
X starts Y	s	si	XXX YYYYYY
X finishes Y	f	fi	XXX YYYYYY
X equal Y	e	e	XXX YYY

Temporal rules are defined by using Allen's temporal relations and represented in Table 4.3. There is no need to define temporal rules for the inverse temporal relations since these relations may be inferred by the main temporal rules. By using temporal rules, temporal relations between video events can be inferred by the fuzzy knowledge base.

Table 4.3: Temporal rules

Temporal rules	Inverse property
x+ < y- => b(x,y)	b(x, y) => bi(y, x)
(x- < y-) and (x+ > y-) and (x+ < y+) => o(x,y)	m(x, y) => mi(y, x)
(x- > y-) and (x+ < y+) => d(x,y)	o(x, y) => oi(y, x)
x+ = y- => m(x,y)	d(x, y) => di(y, x)
(x- = y-) and (x+ < y+) => s(x,y)	s(x, y) => si(y, x)
(y- < x-) and (x+ = y+) => f(x,y)	f(x, y) => fi(y, x)
(x- = y-) and (x+ = y+) => e(x,y)	

4.2.2 Spatial Rules

The semantic video objects can have spatial relations among each other, which represent the relationship between the positions of objects in a specific time interval. The study in [33] defines the spatial relations by using Allen's temporal relations. Allen's temporal interval algebra represents relations in one dimension and it is extended to two-dimensional space. A two-dimensional space is represented by two orthogonal axes x and y. Spatial relations can be defined in terms of temporal relations in both of the x-axis and y-axis. A minimum bounding rectangle (MBR) can be used to represent the position of an object in a video frame. A minimum bounding rectangle is an imaginary rectangle covering all parts of an object and is represented by using two points: upper left corner and lower right corner in this study. These points are projected onto x-axis and y-axis and each projection is actually an interval. Spatial relations are defined by using temporal relations in both of the x-axis and y-axis.

The study in [13] uses the notations *left, right, top* and *bottom* instead of *west, east, north* and *south* respectively. In this study, the same notations are used. There are two main types of spatial relations in this study: directional and topological.

Table 4.4: Definition of Spatial Relations in Terms of Temporal Relations

Spatial Relation	Definition
A bottom B	$A_x\{b, bi, m, mi, o, oi, d, di, s, si, f, fi, e\}B_x$ and $A_y\{b, m\}B_y$
A top B	$A_x\{b, bi, m, mi, o, oi, d, di, s, si, f, fi, e\}B_x$ and $A_y\{bi, mi\}B_y$
A left B	$A_x\{b, m\}B_x$ and $A_y\{b, bi, m, mi, o, oi, d, di, s, si, f, fi, e\}B_y$
A right B	$A_x\{bi, mi\}B_x$ and $A_y\{b, bi, m, mi, o, oi, d, di, s, si, f, fi, e\}B_y$
A top-left B	($A_x\{b, m\}B_x$ and $A_y\{bi, mi, oi\}B_y$) or ($A_x\{o\}B_x$ and $A_y\{bi, mi\}B_y$)
A top-right B	($A_x\{bi, mi\}B_x$ and $A_y\{bi, mi, oi\}B_y$) or ($A_x\{oi\}B_x$ and $A_y\{bi, mi\}B_y$)
A bottom-left B	($A_x\{b, m\}B_x$ and $A_y\{b, m, o\}B_y$) or ($A_x\{o\}B_x$ and $A_y\{b, m\}B_y$)
A bottom-right B	($A_x\{b, m\}B_x$ and $A_y\{b, m, o\}B_y$) or ($A_x\{oi\}B_x$ and $A_y\{b, m\}B_y$)
A overlaps B	$A_x\{d, di, s, si, f, fi, o, oi, e\}B_x$ and $A_y\{d, di, s, si, f, fi, o, oi, e\}B_y$
A equal B	$A_x\{e\}B_x$ and $A_y\{e\}B_y$
A inside B	$A_x\{d\}B_x$ and $A_y\{d\}B_y$
A contain B	$A_x\{di\}B_x$ and $A_y\{di\}B_y$
A touch B	($A_x\{m, mi\}B_x$ and $A_y\{d, di, s, si, f, fi, o, oi, m, mi, e\}B_y$) or ($A_x\{d, di, s, si, f, fi, o, oi, m, mi, e\}B_x$ and $A_y\{m, mi\}B_y$)
A disjoint B	$A_x\{b, bi\}B_x$ or $A_y\{b, bi\}B_y$

Directional relations are *top, left, right, bottom, top-right, top-left, bottom-right, bottom-left*. Topological relations are *equal, inside, disjoint, touch, overlaps, contain*. Directional and topological relations are represented in Figure 4.1 and Figure 4.2 respectively. Table 4.4 represents the spatial relations, which are defined in terms of temporal relations, as defined in [13].

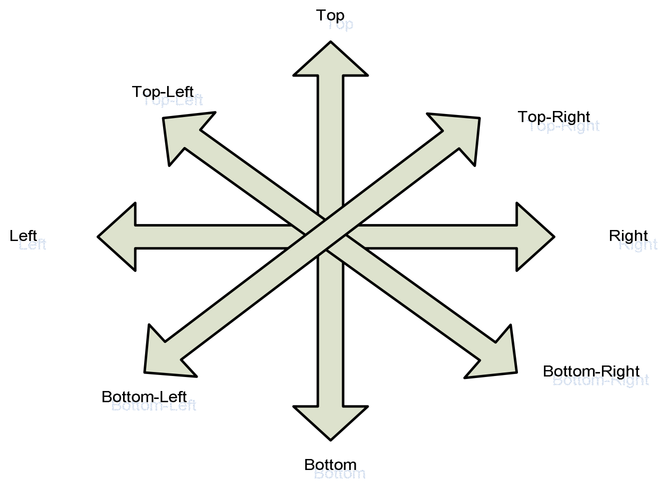


Figure 4.1: Directional Relations

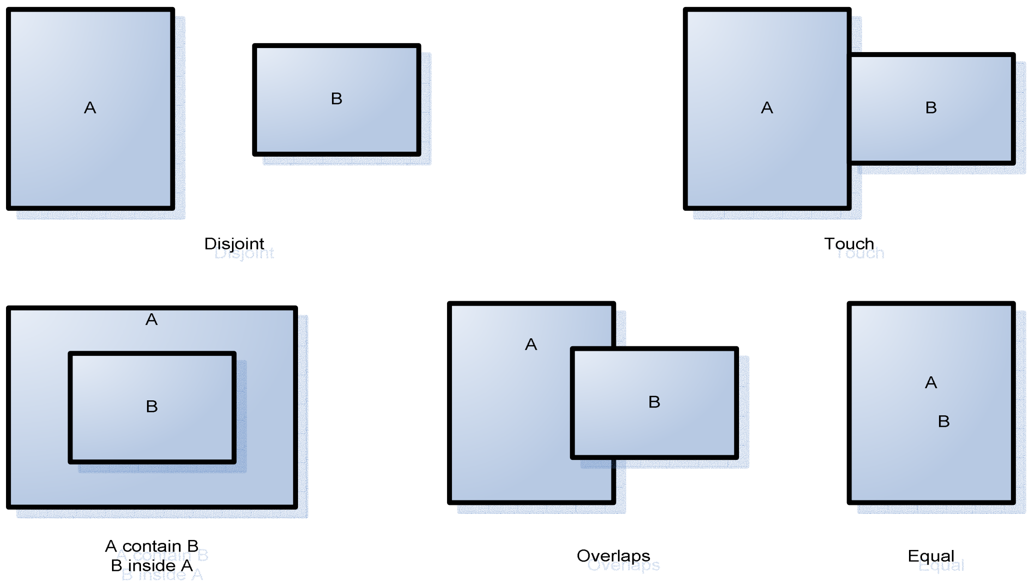


Figure 4.2: Topological Relations

Spatial rules are defined by using spatial relations as represented in Table 4.5. There is no need to define spatial rules for the relations *right, bottom, bottom-right, bottom-left, contain* and *disjoint* since these relations may be inferred by the rules defined for *left, top, top-left, top-right, inside* and *disjoint* respectively. The inferred spatial rules are shown in Table 4.6.

Table 4.5: Spatial Rules Which are Defined in Terms of Temporal Rules

Spatial Rules	
{b, bi, m, mi, o, oi, d, di, s, si, f, fi, e}(A_x, B_x) and {bi, mi}(A_y, B_y)	=> top(A, B)
{b, m}(A_x, B_x) and {b, bi, m, mi, o, oi, d, di, s, si, f, fi, e}(A_y, B_y)	=> left(A, B)
({b, m}(A_x, B_x) and {bi, mi, oi}(A_y, B_y)) or ({o}(A_x, B_x) and {bi, mi}(A_y, B_y))	=> top-left(A,B)
({bi, mi}(A_x, B_x) and {bi, mi, oi}(A_y,B_y)) or ({oi}(A_x, B_x) and {bi, mi}(A_y, B_y))	=> top-right(A, B)
{d, di, s, si, f, fi, o, oi, e}(A_x, B_x) and {d, di, s, si, f, fi, o, oi, e}(A_y, B_y)	=> overlaps(A, B)
{e}(A_x, B_x) and {e}(A_y, B_y)	=> equal(A, B)
{d}(A_x, B_x) and {d}(A_y, B_y)	=> inside(A, B)
({m, mi}(A_x, B_x) and {d, di, s, si, f, fi, o, oi, m, mi, e}(A_y, B_y)) or ({d, di, s, si, f, fi, o, oi, m, mi, e}(A_x, B_x) and {m, mi}(A_y, B_y))	=> touch(A, B)
{b, bi}(A_x, B_x) or {b, bi}(A_y, B_y)	=> disjoint(A, B)

Table 4.6: Inferred Spatial Rules

left(A, B)	=>	right(B, A)
top(A, B)	=>	bottom(B, A)
top-left(A, B)	=>	bottom-right(B, A)
top-right(A, B)	=>	bottom-left(B, A)
inside(A, B)	=>	contain(B, A)
disjoint(A, B)	=>	disjoint(B, A)

Directional spatial relations between two objects may be fuzzy according to the angle between the centers of the minimum bounding rectangles of the video objects. To find the membership degrees of directional spatial relations, the formulas in Table 4.7, which are taken from the study [13], are used. Here, x is the horizontal distance and y is the vertical distance between centers of two rectangles. The membership degree of directional spatial relations is calculated when the spatial rules are fired by the fuzzy processor of FKB.

Table 4.7: Formulas to Calculate Membership Degrees for Fuzzy Spatial Relations

Spatial Relation	Angle	Membership Degree
top	arctan(x/y)	1 – (angle/90)
left	arctan(y/x)	1 – (angle/90)
top-left	arctan(x/y)	1 - (abs(angle-45)/45)
top-right	arctan(y/x)	1 - (abs(angle-45)/45)

4.2.3 Semantic Fuzzy Rules

In this study, football game videos are used as the application domain and semantic fuzzy rules for football game videos are defined. An example to a fuzzy semantic rule is given below:

IF *shotAccuracy* is [very high](0.8) AND

speed is [high, very high](0.7) AND

ballControl is [high, very high](0.8)

THEN *talent* is very high (R-1)

Here, *shotAccuracy*, *ballControl, speed* and *talent* are fuzzy attributes of the class *TalentedPlayer*. *Talent* is a derived attribute, and its value is not stored in the database but inferred by this fuzzy rule. The values in parentheses are the threshold values for the matching degrees of each condition. The "if" part is called the antecedent of the rule. The "then" part is called the consequent of the rule.

The fuzzy inference mechanism utilized in this study is the same as the mechanism used in [30], which uses similarity matching [14]. The inference mechanism is represented in Figure 4.3.

In the pattern-matching phase, the matching degree of each condition in a rule antecedent is calculated by using similarity matrix and inclusion formulas. If the matching degree of each condition is greater than the specified threshold value, then the rule is activated. The overall matching degree of the rule antecedent is found by the following formula [30]:

For AND operator: $\mu_{antecedent} = \text{Min}(\mu_1, \mu_2, \dots, \mu_n)$

For OR operator: $\mu_{antecedent} = \text{Max}(\mu_1, \mu_2, \dots, \mu_n)$

Fuzzy inference mechanism quantifies the rule conclusion with a membership degree using an implication function. Among the various proposed implication functions, the Godelian's fuzzy implication function is used as follows:

$$t(x \text{ is } A \rightarrow y \text{ is } B) = \begin{cases} 1, & \mu_{A(x)} \leq \mu_{B(y)} \\ \mu_{B(y)}, & \mu_{A(x)} > \mu_{B(y)} \end{cases}$$

Here, $\mu_{A(x)}$ is the matching degree of the antecedent of the rule and $\mu_{B(y)}$ is the matching degree of the consequent of the rule.

The fuzzy and crisp attributes might be used together in a rule. If an antecedent predicate is defined with a crisp attribute, traditional pattern matching is used and if the matching is successful, the matching degree will be 1.

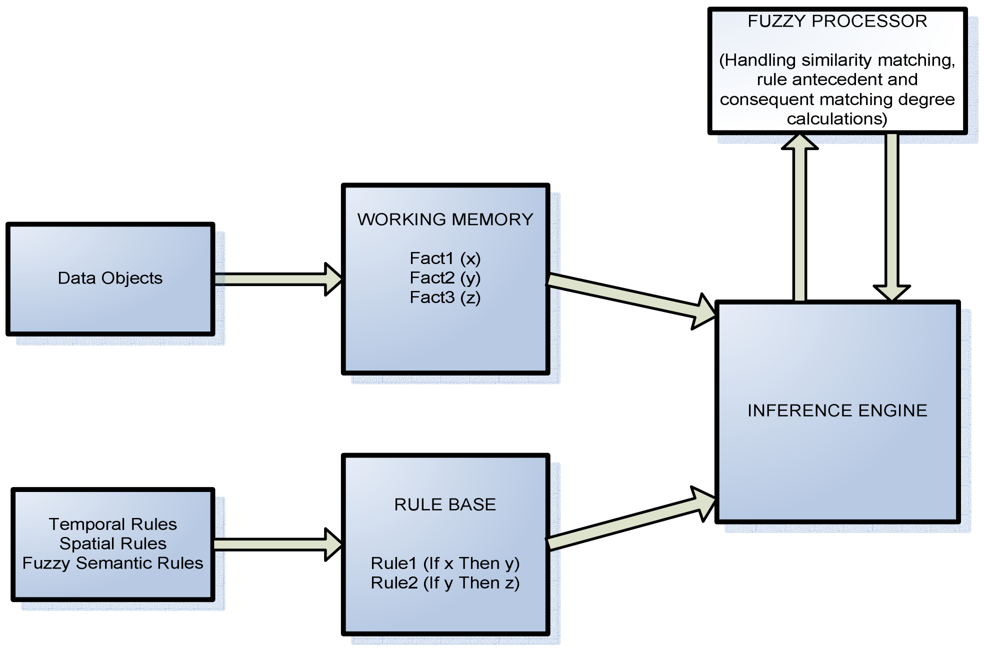


Figure 4.3: Representation of the Inference Mechanism

The fuzzy processor at the knowledge base part provides methods for similarity matching, rule antecedent and consequent matching degree calculations, which are needed for the fuzzy inference mechanism. The similarity matching method of the fuzzy processor uses inclusion degree calculations.

The value of a fuzzy attribute might be fuzzy or crisp. If a fuzzy rule is applied to an object having a fuzzy attribute with crisp value, then our similarity matching method employs membership functions which determine the fuzzy set to which a crisp value belongs. Membership functions are defined for each fuzzy term of a fuzzy domain. The membership functions utilized in this study are right decreasing, triangle, trapezoidal, elliptical, and right increasing.

4.3 Integration of FOOD and FKB

In this section, how the developed framework handles temporal, spatial and fuzzy semantic querying of video data in an integrated environment is explained.

When a query request is made, the bridge firstly sends the query to FOOD. The objects in FOOD, which satisfy the conditions, are sent back to bridge. If the query needs a rule firing, the bridge transfers those objects to the working memory of FKB. The rules are fired on the objects in FKB. Then, the bridge takes the result objects and sends them to the user interface. Representation of the query evaluation is given in Figure 4.4.

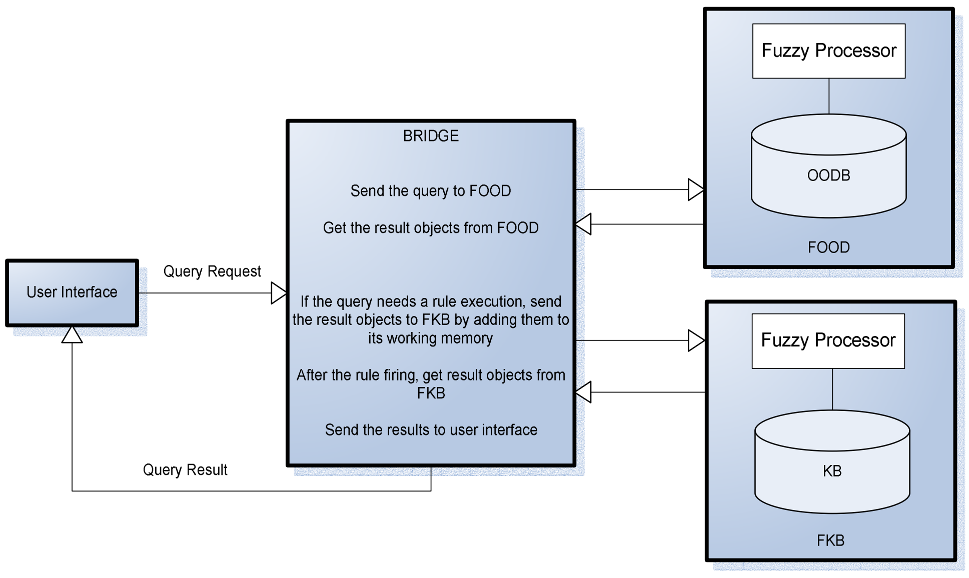

Fuzzy Processor
OODB
FOOD
BRIDGE
Send the query to FOOD
Get the result objects from FOOD
User Interface
Query Request
If the query needs a rule execution, send the result objects to FKB by adding them to its working memory
After the rule firing, get result objects from FKB
Send the results to user interface
Query Result
Fuzzy Processor
KB
FKB

Figure 4.4: Query Evaluation of the Framework

4.3.1 Temporal Query Evaluation

Assume that the user constructs a query to find the temporal relation between the goal event in which playerX scores a goal and the foul event occurring in a specified football game video. To answer such a query, our framework uses the temporal rules, which are stored in our FKB. Figure 4.5 represents the evaluation of a temporal query. The numbers in the figure show the execution order of the operations.

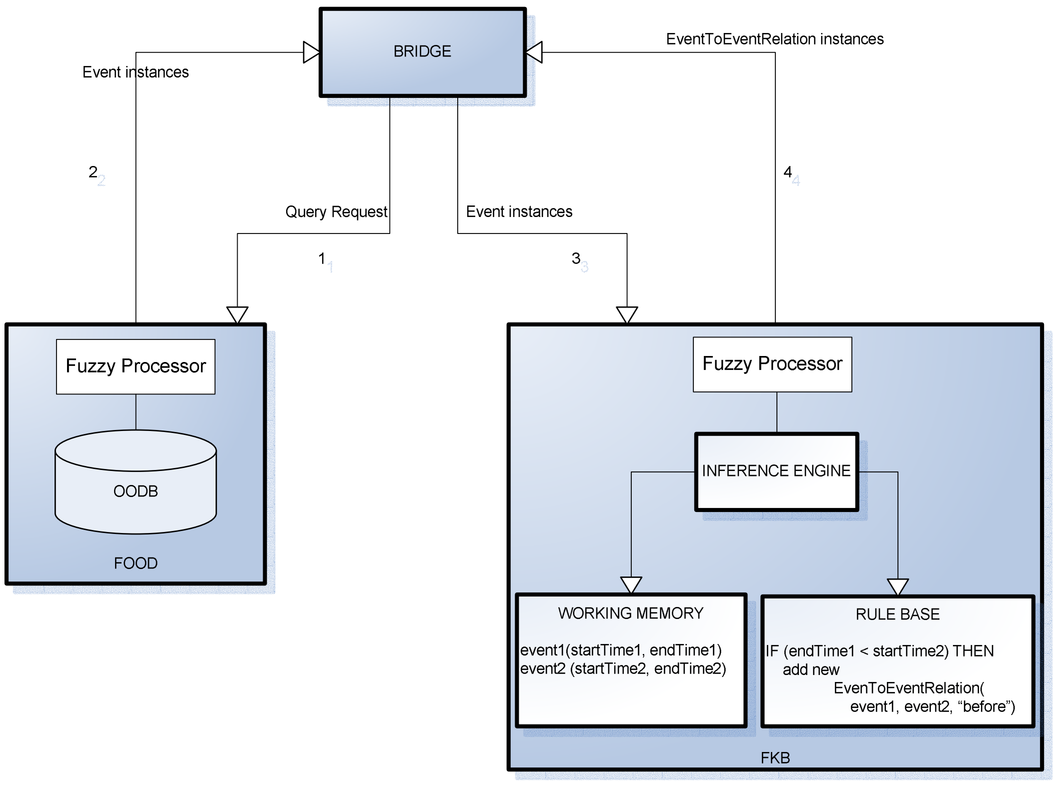


Figure 4.5: Evaluation of a Temporal Query

The algorithm for a temporal query evaluation is as follows:

1. Bridge sends the query to FOOD.
2. FOOD sends the Event instances satisfying the conditions given in the query to the bridge.
3. Bridge transfers the Event instances to the working memory of FKB.

4. The engine of FKB is run and the temporal rules are fired on the instances in the working memory of FKB.

5. If pattern matching is successful in a temporal rule, the rule adds a new EventToEventRelation instance to the working memory of FKB.

6. Bridge gets the EventToEventRelation instances from the working memory of FKB.

7. Bridge sends the EventToEventRelation instances to the user interface.

4.3.2 Spatial Query Evaluation

Assume that the user constructs a query to find the spatial relation between playerX and playerY in the goal event of a specified video. To answer such a query, our framework uses the spatial rules, which are stored in our FKB. The algorithm for a spatial query evaluation is as follows:

1. Bridge sends the query to FOOD.

2. FOOD sends the Position list of Actor instances satisfying the conditions given in the query to the bridge.

3. Bridge transfers the position list (a list of Position instances) of the Actor instances to the working memory of FKB.

4. The engine of FKB is run and the spatial rules are fired on the instances in the working memory of FKB.

5. If pattern matching is successful in a spatial rule, the rule adds a new ActorToActorRelation instance to the working memory of FKB. If the spatial relation is directional, membership degree of the relation is calculated by the fuzzy processor of FKB.

6. Bridge gets the ActorToActorRelation instances from the working memory of FKB.

7. Bridge sends the ActorToActorRelation instances to the user interface.

Figure 4.6 represents the evaluation of a spatial query. The numbers in the figure show the execution order of the operations.

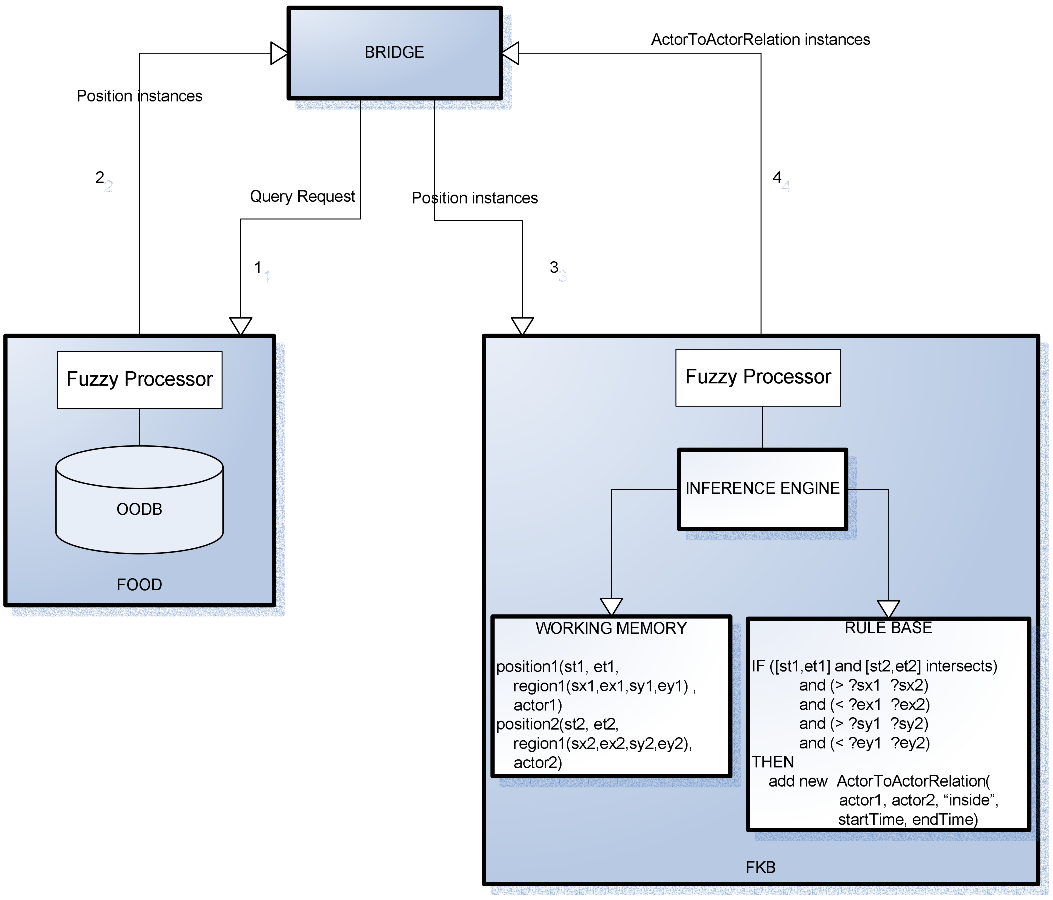
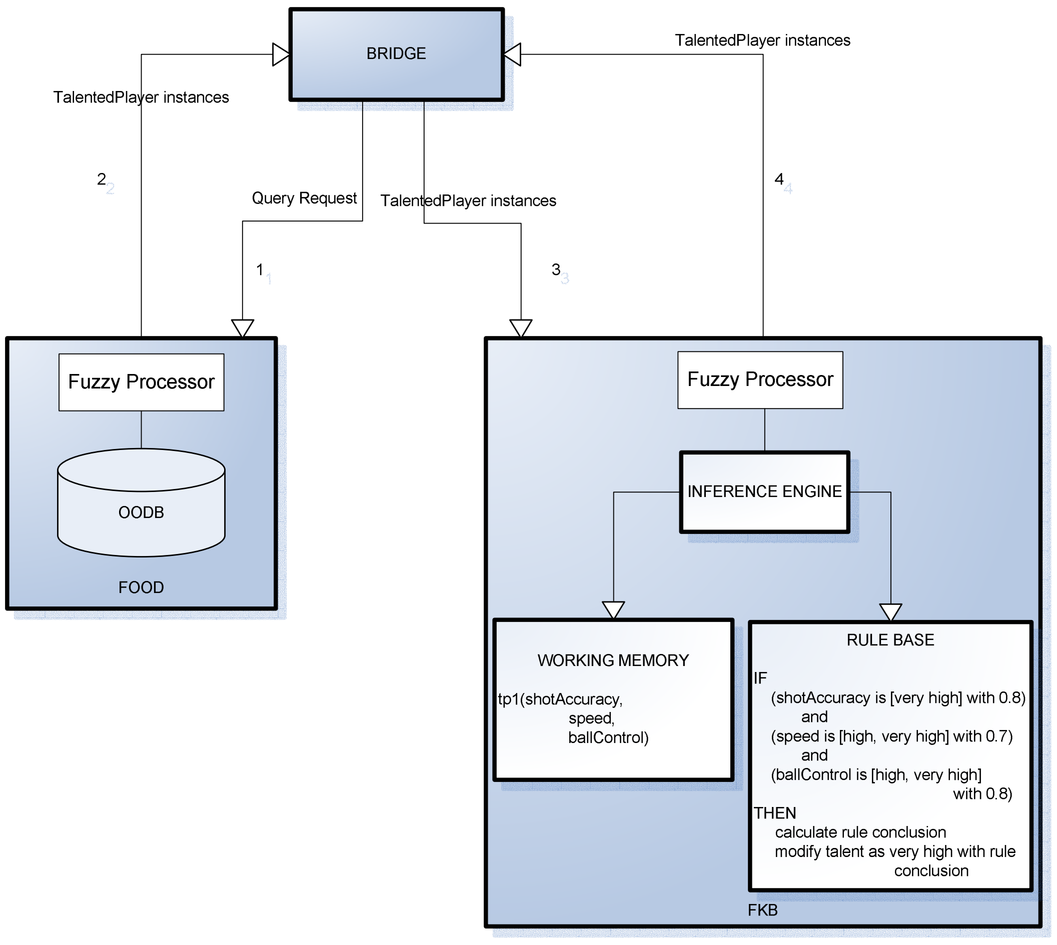

BRIDGE
ActorToActorRelation instances
Position instances
2
Query Request
Position instances
4
1
3
Fuzzy Processor
OODB
FOOD
Fuzzy Processor
INFERENCE ENGINE
WORKING MEMORY
position1(st1, et1, region1(sx1,ex1,sy1,ey1) , actor1)
position2(st2, et2, region1(sx2,ex2,sy2,ey2), actor2)
RULE BASE
IF ([st1,et1] and [st2,et2] intersects) and (> ?sx1 ?sx2) and (< ?ex1 ?ex2) and (> ?sy1 ?sy2) and (< ?ey1 ?ey2)
THEN add new ActorToActorRelation(actor1, actor2, "inside", startTime, endTime)
FKB

Figure 4.6: Evaluation of a Spatial Query

4.3.3 Semantic Query Evaluation

Besides processing the temporal and spatial rules, our FKB also has the capability to process semantic rules, which may include both crisp and fuzzy conditions. Semantic queries can consist of both fuzzy and crisp predicates. Sometimes a semantic query can be so simple that only object-oriented database would be enough whereas in some cases fuzzy inference would be needed.

Figure 4.7 represents the evaluation of a semantic query. The numbers in the figure show the execution order of the operations.



BRIDGE
TalentedPlayer instances
TalentedPlayer instances
2
Query Request
TalentedPlayer instances
4
1
3
Fuzzy Processor
OODB
FOOD
Fuzzy Processor
INFERENCE ENGINE
WORKING MEMORY
tp1(shotAccuracy, speed, ballControl)
RULE BASE
IF
(shotAccuracy is [very high] with 0.8)
and
(speed is [high, very high] with 0.7)
and
(ballControl is [high, very high] with 0.8)
THEN
calculate rule conclusion
modify talent as very high with rule conclusion
FKB

Figure 4.7: Evaluation of a Semantic Query

The algorithm for a semantic query evaluation is as follows:

1. Bridge sends the query to FOOD.
2. FOOD sends the objects satisfying the conditions given in the query to the bridge.
3. If the query needs rule firing (a derived attribute exists), bridge transfers the objects to the working memory of FKB.
4. The engine of FKB is run and the semantic fuzzy rules are fired on the objects in the working memory of FKB.
5. If pattern matching is successful in a semantic fuzzy rule, the rule modifies the object and sets the value of the derived attribute.

6. Bridge gets the objects from the working memory of FKB.

7. Bridge sends the objects to the user interface.

When the query is sent to FOOD (in the first step of the semantic query evaluation algorithm), it evaluates the following algorithm:

1. The query is executed for the crisp attributes at FOOD.

2. The objects satisfying the conditions for crisp attributes are returned by FOOD.

3. For each object in the list

3.1. If the membership degree of the object is smaller than the specified threshold, exclude that object from the list, go to step 3.

3.2. For each uncertain attribute value specified in the query

3.2.1. If the uncertain attribute is a derived attribute, go to step 3.2.

3.2.2. If the uncertain attribute's type is fuzzy

3.2.2.1. If the semantics of the uncertain attribute is AND, calculate the AND inclusion degree of the attribute's value to the value specified in the query.

3.2.2.2. If the semantics of the uncertain attribute is OR, calculate the OR inclusion degree of the attribute's value to the value specified in the query.

3.2.2.3. If the semantics of the uncertain attribute is XOR, calculate the XOR inclusion degree of the attribute's value to the value specified in the query.

3.2.2.4. If the calculated inclusion degree is smaller than the threshold specified in the query for that uncertain attribute, exclude the object from the result list.

3.2.3. If the uncertain attribute's type is incomplete

3.2.3.1. Calculate the incomplete inclusion degree of the range of the attribute's value to the range specified in the query.

3.2.3.2. If the calculated inclusion degree is smaller than the threshold specified in the query for that uncertain attribute, exclude the object from the result list.

3.2.4. If the uncertain attribute's type is null

3.2.4.1. If the attribute's crisp value and null value is not equal to the value specified in the query, exclude the object from the result list.

4. Return the result list to the user interface.

Query evaluation will be explained for four cases. In the first case, only fuzzy attributes are queried and there is no need for rule firing. In the second case, a derived attribute is queried, so a rule firing is necessary. In the third case, fuzzy attributes and derived attributes are queried at the same time. In the fourth case, an example to incomplete inclusion degree calculation and usage of a membership function is given.

The similarity matrices used in the examples are represented in Table 4.8 and Table 4.9. Suppose that the objects shown in Table 4.10 exist in the fuzzy object-oriented database.

FuzzyDegree and FuzzyAge types have OR semantics; therefore, the inclusion formula for the OR semantics is used. The formula for OR semantics is as follows:

$$INC(rng_C(a_i)/o_j(a_i)) = Min[Max(\mu_S(x,z)),Threshold(o_j(a_i))],$$

$$\forall x \in o_j(a_i), \forall z \in rng_C(a_i)$$

Here, the threshold value indicates the minimum level of similarity between the values of an attribute and it can be formulated as follows:

$$Threshold(o_j(a_i)) = Min[\mu_S(x,z)], \ \forall x, \forall z \in o_j(a_i)$$

1- Consider the semantic query below:

"Find the *very young* (with a threshold of 0.7) players with the *medium* degree of shotAccuracy (with a threshold of 0.7) having an object membership degree above 0.4.

Here, *shotAccuracy* and *age* are fuzzy attribute in class TalentedPlayer. Assume that we have the TalentedPlayer objects shown in Table 4.10 in the fuzzy object-oriented database. Consider the similarity matrix for *shotAccuracy* attribute, which is represented in Table 4.8, and the similarity matrix for *age* attribute, which is represented in Table 4.9.

Table 4.8: Similarity Matrix for *shotAccuracy*, *ballControl, speed* and *talent*

FuzzyDegree	very high	high	medium	low	very low
very high	1.0	0.9	0.6	0.2	0.2
high	0.9	1.0	0.7	0.2	0.2
medium	0.6	0.7	1.0	0.2	0.2
low	0.2	0.2	0.2	1.0	0.5
very low	0.2	0.2	0.2	0.5	1.0

Table 4.9: Similarity Matrix for the Fuzzy Attribute *age*

FuzzyAge	very old	old	young	very young	infant
very old	1.0	0.7	0.0	0.0	0.0
old	0.7	1.0	0.0	0.0	0.0
young	0.0	0.0	1.0	0.8	0.1
very young	0.0	0.0	0.8	1.0	0.3
infant	0.0	0.0	0.1	0.3	1.0

Table 4.10: The TalentedPlayer Objects in the Fuzzy Object-Oriented Database

object id	**age**	**shotAccuracy**	**speed**	**ballControl**	**object membership degree**
o1	young	high	very high	high	1
o2	very young	high, very high	high	high, very high	0.93
o3	old	high	medium	medium	0.8
o4	young	low	low	medium	0.37

The algorithm of the evaluation of this semantic fuzzy query is as follows:

1. Bridge sends the query to FOOD to find the TalentedPlayer objects.

2. FOOD sends the found TalentedPlayer objects, which have object membership degrees greater than the specified threshold (0.4) to bridge. Since the object membership degree of o4 (0.3) is smaller than the threshold value, o4 is ignored.

3. Bridge uses the inclusion formulas implemented in the fuzzy processor of FOOD to find the inclusion degree of the values of *age* attribute of TalentedPlayer objects (o1, o2, and o3) to the fuzzy set “very young”. Considering the three objects in the fuzzy object-oriented database:

   For o1: μ_a(young, very young) = 0.8

   For o2: μ_a(very young, very young) = 1

   For o3: μ_a(old, very young) = 0

   Only o1 and o2 satisfy the threshold value given for *age* attribute.

4. Bridge uses the inclusion formulas implemented in the fuzzy processor of FOOD to find the inclusion degree of the values of *shotAccuracy* attribute of TalentedPlayer objects (o1 and o2) to the fuzzy set “medium”. Considering the objects o1 and o2 in the fuzzy object-oriented database:

   For o1: μ_{sa}(high, medium) = 0.7

   For o2: Min [Max (μ_{sa}(high, medium), μ_{sa}(very high, medium)),

   μ_{sa}(high, very high)] = Min[Max (0.7, 0.6), 0.9] = Min[0.7, 0.9] = 0.7

   Both of the o1 and o2 satisfy the threshold value given for *shotAccuracy* attribute.

5. Bridge sends the result objects (o1 and o2) to the user interface.

2- Consider the semantic query below:

“Find the players with *very high* degree of talent (with a threshold of 0.6) which score a goal in a goal event”.

Here, *talent* is a derived attribute in class TalentedPlayer. The query needs a fuzzy rule firing to find the value of *talent* attribute. Assume that we have the rule (R-1) in the fuzzy knowledge base for the derived attribute *talent*. Assume that we have the TalentedPlayer objects shown in Table 4.10 in the fuzzy object-oriented database and all these objects have the role “scorer” in a goal event.

IF *shotAccuracy* is [very high](0.8) AND

speed is [high, very high](0.7) AND

ballControl is [high, very high](0.8)

THEN *talent* is very high (R-1)

The algorithm of the evaluation of this semantic fuzzy query is as follows:

1. The bridge sends the query to FOOD to find the TalentedPlayer objects, which appear in a goal event. The bridge transfers the found TalentedPlayer objects (o1, o2, o3, and o4) to the working memory of FKB.

2. When the engine of FKB is run, the semantic fuzzy rules will fire for the objects in working memory. For the first condition of the rule (R-1), the values of *shotAccuracy* attribute are compared.

   For o1: μ_{sa}(high, very high) = 0.9

   For o2: Min[Max(μ_{sa}(high, very high), μ_{sa}(very high, very high)),

   μ_{sa}(high, very high)] = Min[Max(0.9, 1), 0.9] = Min[1, 0.9] = 0.9

   For o3: μ_{sa}(high, very high) = 0.9

   For o4: μ_{sa}(low, very high) = 0.2

   The objects o1, o2 and o3 satisfy the first condition of the rule.

3. For the second condition of the rule, the values of *speed* attribute are compared.

   For o1: Max(μ_{sp}(very high, high), μ_{sp}(very high, very high)) = Max(0.9, 1) = 1

   For o2: Max(μ_{sp}(high, high), μ_{sp}(high, very high)) = Max(1, 0.9) = 1

   For o3: Max(μ_{sp}(medium, high), μ_{sp}(medium, very high)) = Max(0.7, 0.6) = 0.7

   The objects o1, o2 and o3 satisfy the second condition of the rule.

4. For the third condition of the rule, the values of *ballControl* attribute are compared.

   For o1: Max(μ_{bc}(high, high), μ_{bc}(high, very high))=Max(1, 0.9) = 1

   For o2: Min[Max(μ_{bc}(high, high), μ_{bc}(high, very high),

   μ_{bc}(very high, high), μ_{bc}(very high, very high)), μ_{bc}(high, very high)]

   = Min[Max(1, 0.9, 0.9, 1), 0.9] = Min[1, 0.9] = 0.9

   For o3: Max(μ_{bc}(medium, high), μ_{bc}(medium, very high))=Max(0.7, 0.6) = 0.7

   The objects o1 and o2 satisfy the third condition of the rule.

5. o3 does not satisfy the second condition since the matching degree 0.7 is smaller than the threshold value 0.8 specified in the rule definition. Therefore, only o1 and o2 satisfy the rule conditions. The overall matching degree of the rule antecedent for o1 and o2 is found according to the formula explained in Section 4.2.3 as follows:

   For o1: $\mu_{antecedent}$ = Min(0.9, 1, 1) = 0.9

   For o2: $\mu_{antecedent}$ = Min(1, 1, 1) = 1

6. The matching degree of the consequent is

   μ_{talent} (very high, very high) = 1.

   Since the matching degree of the consequent is greater than the matching degree of the antecedent for object o1, the membership degree of the rule conclusion is 1.0 for o1 according to the implication formula explained in Section 4.2.3. The matching degree of the consequent is equal to the matching degree of the antecedent for object o2, so the membership degree of the rule conclusion is 1.0 for o2 according to the implication formula.

   Since the rule conclusion is greater than the threshold value 0.6 specified for the *talent* attribute in the query, o1 and o2 will be included in the result list.

7. The bridge takes the Player objects from FKB. Only o1 and o2 are in the result list. Then the bridge sends the result objects to the user interface.

3- Consider the semantic query below:

"Find the *young or very young* (with a threshold of 0.7) players with *medium* degree of talent (with a threshold of 0.6) which score a goal in a goal event".

Here, *talent* is a derived attribute and *age* is a fuzzy attribute in class TalentedPlayer. The query needs a fuzzy rule firing to find the value of *talent* attribute. Assume that we have the rule (R-1) in the fuzzy knowledge base for the derived attribute *talent*. Assume that we have the TalentedPlayer objects shown in Table 4.10 in the fuzzy object-oriented database and all these objects have the role "scorer" in a goal event.

The algorithm of the evaluation of this semantic fuzzy query is as follows:

1. The bridge sends the query to FOOD to find the TalentedPlayer objects with young age and appear in a goal event. FOOD uses the inclusion formulas to find the inclusion of the values of *age* attributes of players to the fuzzy set [young, very young]. Considering the four objects in our fuzzy object-oriented database:

   For o1: Max (μ_a(young, young), μ_a(young, very young)) = Max (1, 0.8) = 1

   For o2: Max (μ_a(very young, young), μ_a(very young, very young))

   = Max (0.8, 1) = 1

   For o3: Max (μ_a(old, young), μ_a(old, very young)) = Max (0, 0) = 0

   For o4: Max (μ_a(young, young), μ_a(young, very young)) = Max (1, 0.8) = 1

   The objects o1, o2, and o4 satisfy the threshold value given for *age* attribute. Therefore, the bridge transfers these three objects to the working memory of FKB.

2. For the first condition of the rule (R-1), the values of *shotAccuracy* attribute are compared.

   For o1: Max(μ_{sa}(high, very high)) = Max(0.9) = 0.9

For o2: Min[Max(μ_{sa}(high, very high), μ_{sa}(very high, very high)),

μ_{sa}(high, very high)] = Min[Max(0.9, 1), 0.9] = Min[1, 0.9] = 0.9

For o4: Max(μ_{sa}(low, very high)) = Max (0.2) = 0.2

o4 does not satisfy the first condition since the matching degree 0.2 is smaller than the threshold value 0.8 specified in the rule definition. Therefore, only o1 and o2 satisfy the first rule condition.

3. For the second condition of the rule, the values of *speed* attribute are compared.

   For o1: Max(μ_{sp}(very high, high), μ_{sp}(very high, very high)) = Max(0.9, 1) = 1

   For o2: Max(μ_{sp}(high, high), μ_{sp}(high, very high)) = Max(1, 0.9) = 1

   The objects o1 and o2 satisfy the second condition of the rule.

4. For the third condition of the rule, the values of *ballControl* attribute are compared.

   For o1: Max(μ_{bc}(high, high), μ_{bc}(high, very high))=Max(1, 0.9) = 1

   For o2: Min[Max(μ_{bc}(high, high), μ_{bc}(high, very high),

   μ_{bc}(very high, high), μ_{bc}(very high, very high)), μ_{bc}(high, very high)]

   = Min[Max(1, 0.9, 0.9, 1), 0.9] = Min[1, 0.9] = 0.9

   The objects o1 and o2 satisfy the third condition of the rule.

5. The overall matching degree of the rule antecedent for o1 and o2 is found according to the formula explained in Section 4.2.3 as follows:

   For o1: $\mu_{antecedent}$ = Min(0.9, 1, 1) = 0.9

   For o2: $\mu_{antecedent}$ = Min(1, 1, 1) = 1

6. The matching degree of the consequent is

   μ_{talent} (medium, very high) = 0.6.

Since the matching degree of the consequent is smaller than the matching degree of the antecedent, the membership degree of the rule conclusion is 0.6 for the objects o1 and o2 according to the implication formula explained in Section 4.2.3. Since the rule conclusion is equal to the threshold value 0.6 specified for the *talent* attribute in the query, o1 and o2 will be included in the result list.

7. The bridge takes the Player objects from FKB. Only o1 and o2 are in the result list. Then the bridge sends the result objects to the user interface.

4- Consider the semantic query below:

"Find the videos of football matches played with an audience of *{10000 - 40000}* (with a threshold of 0.6) and played in a *mild* weather (with a threshold of 0.7)

Here, *audience* is an incomplete attribute and *airCondition* is a fuzzy attribute in class Match. The query needs calculation of inclusion degree for incomplete attributes and fuzzy attributes. The formulas to find inclusion degrees for incomplete attributes are previously explained in Chapter 2. Assume that we have the Match objects shown in Table 4.11 in the fuzzy object-oriented database. The similarity matrix for *airCondition* attribute is given in Table 4.12. The attribute *airCondition* has OR semantics so inclusion degree formula for OR semantics is used.

Table 4.11: The Match Objects in the Fuzzy Object-Oriented Database

object id	**audience**	**airCondition**
o1	{20000 - 30000}	45
o2	{7000 - 8000}	low
o3	{7000 - 50000}	30
o4	{7000 - 20000}	mild, hot
o5	{30000 – 50000}	very low, frigid

Table 4.12: Similarity Matrix for the Fuzzy Attribute *airCondition*

FuzzyTemperature	hot	mild	normal	low	very low	frigid
hot	1.0	0.7	0.2	0.2	0.2	0.2
mild	0.7	1.0	0.2	0.2	0.2	0.2
normal	0.2	0.2	1.0	0.7	0.7	0.7
low	0.2	0.2	0.7	1.0	0.7	0.7
very low	0.2	0.2	0.7	0.7	1.0	0.7
frigid	0.2	0.2	0.7	0.7	0.7	1.0

Consider the followings:

$R[R_1..R_2]$ is the range of the incomplete attribute

$V[V_1..V_2]$ is the value of the incomplete attribute

The algorithm of the evaluation of this semantic fuzzy query is as follows:

1. The bridge sends the query to FOOD to find the Match objects with audience of {10000 - 40000}. FOOD uses the inclusion formulas to find the inclusion of the values of *audience* attribute of Match objects to the range {10000 - 40000} Considering the four objects in our fuzzy object-oriented database:

   For o1: INC = 1, since $R_2 \geq V_2 > V_1 \geq R_1$

   For o2: INC = 0, since $R_1 > V_2 > V_1$

   For o3: INC = (40000 – 10000 + 1) / (50000 – 7000 + 1) = 0.69

   by the formula:

   $INC(rng_C(a_i)/o_j(a_i)) = (R_2 - R_1 + 1) / (V_2 - V_1 + 1),\ \text{where}\ V_2 \geq R_2 > R_1 \geq V_1$

   For o4: INC = (20000 – 10000 + 1) / (20000 – 7000 + 1) = 0.76

   by the formula:

   $INC(rng_C(a_i)/o_j(a_i)) = (V_2 - R_1 + 1) / (V_2 - V_1 + 1),\ \text{where}\ R_2 \geq V_2 \geq R_1 \geq V_1$

   For o5: INC = (40000 – 30000 + 1) / (50000 – 30000 + 1) = 0.5

   by the formula:

   $INC(rng_C(a_i)/o_j(a_i)) = (R_2 - V_1 + 1) / (V_2 - V_1 + 1),\ \text{where}\ V_2 \geq R_2 \geq V_1 \geq R_1$

   The objects o1, o3, and o4 satisfy the threshold value for *audience* attribute.

2. For the second condition of the query, the values of *airCondition* attribute of the objects o1, o3 and o4 are compared with the fuzzy value *mild*. Consider the following membership function is defined for the fuzzy term *mild*:

triangle 25, 30, 35

The algorithm to find the membership degree of a crisp value to a fuzzy set for the membership function *triangle,* which is taken from the IFOOD [30] implementation, is given below:

Let *number* be the crisp value and let arg1, arg2, arg3 be the parameters of the membership function *triangle* (arg1 = 25, arg2 = 30 and arg3 = 35 in our example):

- if number < arg1, membership degree is 0
- if number ≥ arg3, membership degree is 0
- if number ≥ arg1 and number < arg2,

  membership degree = (number – arg1) / (arg2 – arg1)
- if number ≥ arg2 and number < arg3,

  membership degree = (arg3 – number) / (arg3 – arg2)

For o1: number = 45, membership degree is 0 since number > arg3

For o3: number = 30, membership degree = (35-30) / (35 - 30) = 1,

since number ≥ arg2 and number < arg3,

For o4: $\text{Min}[\text{Max}(\mu_{ac}(\text{mild, mild}), \mu_{ac}(\text{hot, mild})), \mu_{ac}(\text{hot, mild})]$

$= \text{Min}[\text{Max}(1, 0.7), 0.7] = \text{Min}[1, 0.7] = 0.7$

The objects o3 and o4 satisfy the threshold value for the attribute *airCondition*.

3. Only o3 and o4 are in the result list. Then the bridge sends the result objects to the user interface.

CHAPTER 5

IMPLEMENTATION

We developed a prototype system of our framework by using Java programming language and used it for a football game video database application.

A fuzzy object-oriented database (FOOD) and a fuzzy knowledge base (FKB) are integrated in our framework. As the object-oriented database, db4o [36], which is an open source object database developed in Java, is used. Since a fuzzy object-oriented data model is used for the object-oriented database, we developed methods for calculating inclusion degrees and similarity matching which are included in the fuzzy processor of FOOD. As the knowledge base, Jess [37], which is a rule engine for the Java platform, is used. Since Jess does not provide any construct for fuzzy inference, we developed methods, which are included in the fuzzy processor of FKB, to handle similarity matching.

FOOD and FKB are connected by a bridge. There is a user interface above this integrated environment, which provides the users annotation and querying facilities. The users do not need to know any detail of the underlying system. The user interface, bridge and fuzzy processors are developed in Java programming language.

To play videos in our user interface, JMF API (Java Media Framework Application Programming Interface) [45] is used. The JMF API provides various video control facilities (such as playing, pausing).

The users of the framework can annotate football game videos by using the user interface. The IBM MPEG-7 Annotation Tool [38] is used to segment a video into video shots automatically. The output of the IBM MPEG-7 Annotation Tool is then parsed by our user interface and the shots detected are stored in our object-oriented database.

The framework also provides the user querying facilities. The supported queries are (fuzzy) semantic, temporal, (fuzzy) spatial, regional, trajectory and hierarchical queries.

Queries are performed in an integrated environment and the rules in FKB are fired whenever needed.

The rest of the chapter is organized as follows: The implementation tools, which are utilized in the prototype, are explained briefly in the first section. In the second section, the information about the architecture of our framework is given. The third section describes the annotation module of the prototype and the last section describes the querying facilities of the prototype system.

5.1 Implementation Tools

5.1.1 db4o

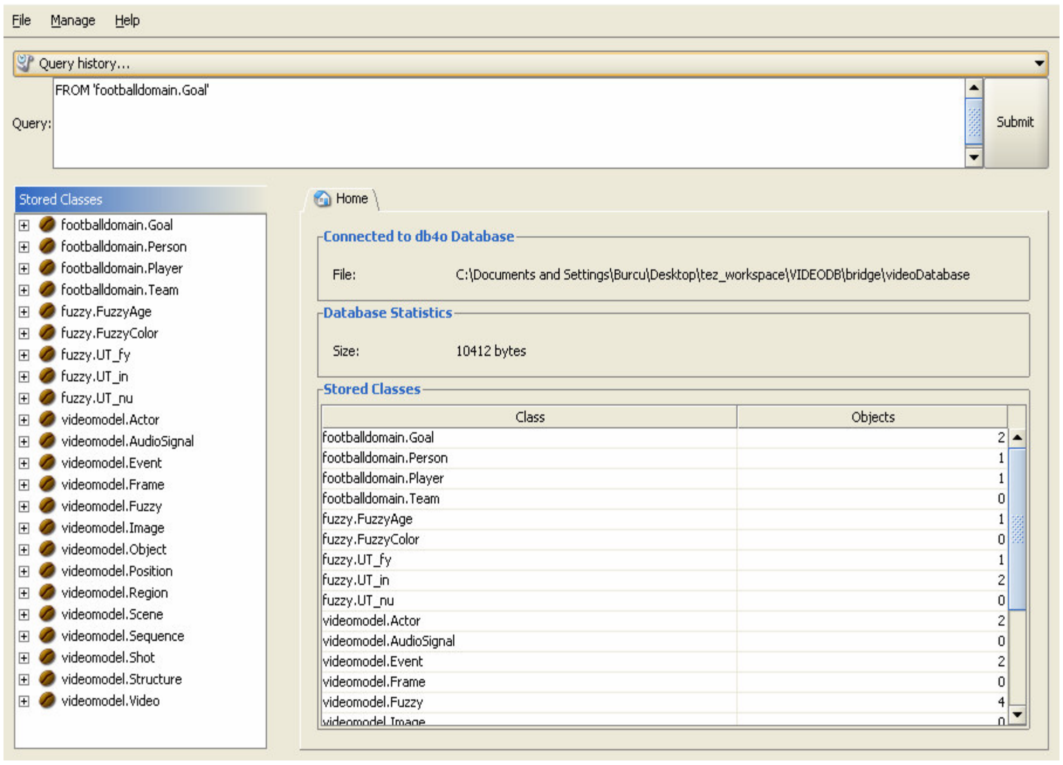
db4o [36] is an open source object database developed for both Java and .Net developers. It allows the users to store complex objects by providing a strong integration with object-oriented programming languages. It eliminates the use of object-relational mapping which most of the developers should deal with in object-oriented programming. By using db4o, there is no need to design an additional database schema since the class model becomes the database schema of your application. db4o also allows updating and refactoring the class schema.

It also supports Native Queries (NQ), which allows the developers to construct queries by using the programming languages eliminating the use of a data access API. A sample native query is given below:

```
List<Video> videos
        = db.query<Video>(new Predicate() {
 public boolean match(Video video)
 {
     return video.getTitle = = “Fenerbahçe-Galatasaray Turkey Cup Match”;
}} )
```

Beside NQ, db4o also provides the object-oriented query APIs: Query by Example (QBE) and S.O.D.A. S.O.D.A. is a low-level querying API and allows creating dynamic queries at runtime by providing direct access to the nodes of query graphs. An example to a S.O.D.A query is given below:

Query query = db.query();
query.constrain(Video.class);
query.descend("title").constrain("Fenerbahçe-Galatasaray Turkey Cup Match");
ObjectSet result = query.execute();



```
FROM 'footballdomain.Goal'
```

Stored Classes

- footballdomain.Goal
- footballdomain.Person
- footballdomain.Player
- footballdomain.Team
- fuzzy.FuzzyAge
- fuzzy.FuzzyColor
- fuzzy.UT_fy
- fuzzy.UT_in
- fuzzy.UT_nu
- videomodel.Actor
- videomodel.AudioSignal
- videomodel.Event
- videomodel.Frame
- videomodel.Fuzzy
- videomodel.Image
- videomodel.Object
- videomodel.Position
- videomodel.Region
- videomodel.Scene
- videomodel.Sequence
- videomodel.Shot
- videomodel.Structure
- videomodel.Video

Connected to db4o Database

File: C:\Documents and Settings\Burcu\Desktop\tez_workspace\VIDEODB\bridge\videoDatabase

Database Statistics

Size: 10412 bytes

Stored Classes

Class	Objects
footballdomain.Goal	2
footballdomain.Person	1
footballdomain.Player	1
footballdomain.Team	0
fuzzy.FuzzyAge	1
fuzzy.FuzzyColor	0
fuzzy.UT_fy	1
fuzzy.UT_in	2
fuzzy.UT_nu	0
videomodel.Actor	2
videomodel.AudioSignal	0
videomodel.Event	2
videomodel.Frame	0
videomodel.Fuzzy	4
videomodel.Image	0

Figure 5.1: An Example Screenshot of the Object Manager GUI

QBE allows constructing queries by simply using the setter methods of the Java classes as follows:

Video video = new Video();
video.setTitle("Fenerbahçe-Galatasaray Turkey Cup Match");
List videos = db.get(video);

db4o developer community also provides Object Manager GUI which is a visual tool providing browsing, querying and editing of the objects and classes stored in a db40 object database. An example screenshot of Object Manager GUI is given in Figure 5.1.

5.1.2 Jess

Jess [37] is a Java based rule engine and scripting language, which is used for building expert systems. It is developed at Sandia National Laboratories. The syntax of the Jess language is very similar to the syntax of Lisp language. Since Jess is written in Java, it can be easily integrated with the applications developed with Java technology.

As it is explained in [40], the inspiration for the development of Jess is the CLIPS expert system shell [42], which is an open source rule engine developed by C language. Although the rule languages of these two rule engines are very similar, their implementations differ in so many ways. Different from CLIPS, Jess is dynamic and all the Java APIs can be accessed from Jess.

Jess uses the well-known Rete algorithm for pattern matching. The Rete algorithm is implemented by building a network of interconnected nodes, which is called Rete network.

Like a typical rule engine, Jess consists of a rule base, an inference engine and a working memory. The rule base stores a set of rules and the working memory consists of a set of facts. A rule is similar to an "if –then" statement in a procedural programming language but differently, the Jess rules are executed whenever the left-hand side of the rule is satisfied. The inference engine continuously checks whether any rule in the rule base can be applied to a fact in the working memory. This process is called pattern matching. If the pattern in the left-hand side of the rule is matched with a fact in the working memory, then the actions in the right-hand side of the rule are performed.

The main features of the Jess language, which are used in this study, are explained briefly in this section.

Variables: Like an ordinary programming language, Jess language also provides variable definitions. A variable is defined with a question mark (?) character in Jess. A value can be assigned to a variable by using *bind* function as follows:

(bind ?x "value of x")

Jess also provides creating global variables, which are not destroyed by a reset command as follows:

*(defglobal ?*gv* = 3)*

Facts: The working memory of Jess contains a set of facts which rules can be applied. Each fact has a template, which consists of a name and a set of slots. The template of a fact is similar to the class of a Java object and the slots are similar to the attributes of a Java class. An example template and fact can be created as follows:

```
(deftemplate person (slot name) (slot surname) (slot age))

(person (name “John”) (surname “Doe”) (age 25))
```

Functions: Jess provides the definition of your own functions like any other programming language. The created function can be called in the right-hand side of a Jess rule. An example function definition in Jess is as follows:

```
(deffunction smaller (?v1 ?v2)
        (if (< ?v1 ?v2)  then (return ?v1)
        else (return ?v2)
)
```

Java reflection: Jess can directly access and manipulate the Java objects of an application. To use an object of a Java class, you import the package of the class as follows:

```
(import videomodel.*)
```

Once you import the package, you should define templates corresponding to the Java class you want to use as follows:

```
(deftemplate Event (declare (from-class Event)))
```

Now, you can create new Event objects as follows:

```
(bind ?e = (new Event))
```

To call a method of Event class, the construct *call* is used:

```
(call ?e setName “eventname”)
```

Rules: Jess provides rule definitions by using *defrule* construct as follows:

```
(defrule before
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) (< ?e1  ?s2)
                    (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "before")))
```

If the left-hand side (LHS) of the rule matches with an Event object in the working memory, the right-hand side (RHS) of the rule is executed and a new EventToEventRelation object is added to the working memory.

5.1.3 Java Media Framework API (JMF)

JMF API [45] provides adding time-based multimedia such as audio and video to the applications developed in Java. It supplies all the methods to control video streaming such as playing, pausing, restarting and stopping the video stream, positioning the stream at a specified time, disabling and enabling the sound of the video. Despite JMF supports different video formats, it brings difficulties when it is used with video formats other than AVI. Therefore, our prototype works best with the AVI videos.

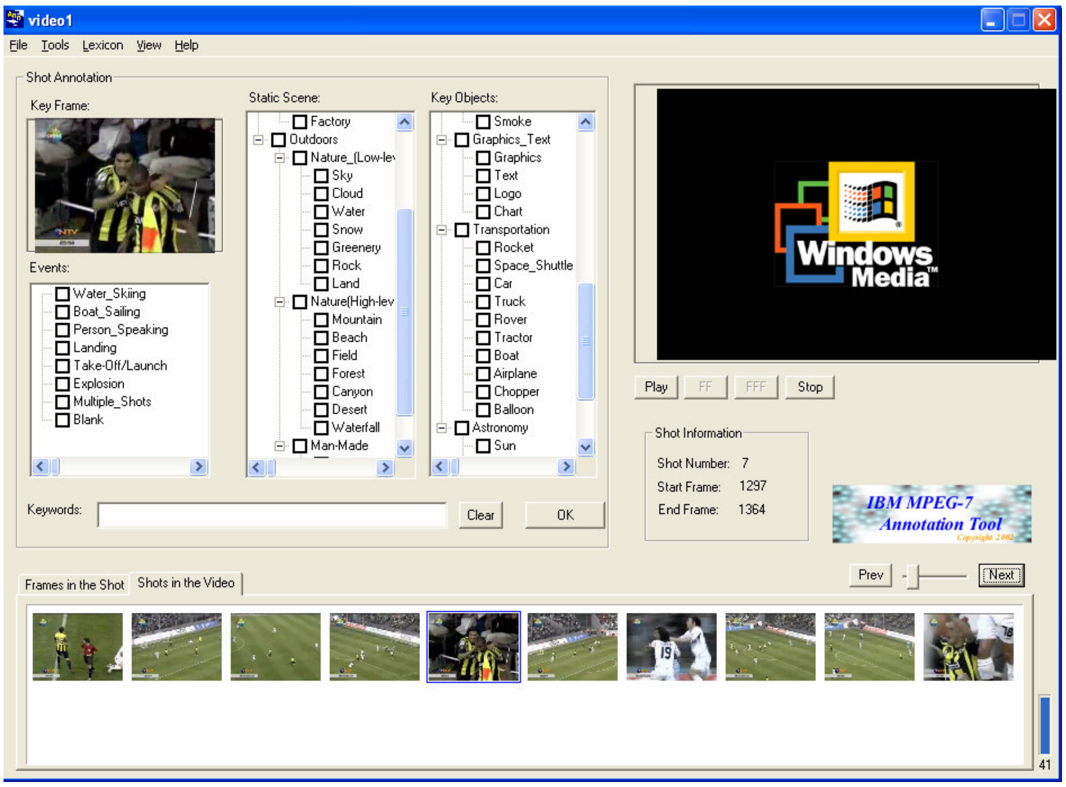
5.1.4 IBM MPEG-7 Annotation Tool (VideoAnnEx)

IBM MPEG-7 Annotation Tool [38] provides annotation of video sequences with MPEG-7 metadata. The tool segments a video sequence into its shots by detecting scene cuts, dissolutions and fading. The detected video shots can be annotated with static scene, key object, event descriptions and other lexicon sets. The annotated video shots are stored as MPEG-7 descriptions in an XML file. Therefore, the input of the tool is a MPEG video sequence and the output of the tool is a MPEG-7 XML file.

The tool also allows opening a previously constructed MPEG-7 XML file to display the annotations and video shots related to a video sequence. Customized lexicons can also be created by using the tool.

An example screenshot of the IBM MPEG-7 Annotation Tool is shown in Figure 5.2. As represented in the figure, there are three sections. The first one is on the upper right corner of the window and is used to play the video shots. The second one is on the upper left corner of the window and consists of the sections, which are used for shot annotation. The third one is at the bottom of the window and is used to display the detected shots or to display the frames within a single shot. There is also an additional pop-up window for region annotation, which is not shown in Figure 5.2.

IBM MPEG-7 Annotation Tool accepts MPEG formatted videos whereas JMF API works best with AVI formatted videos. A video format converter is used to convert videos from one format to the other. MPEG formatted versions are only used for automatic shot annotation purposes. AVI formatted versions are annotated by using the user interface.



Shot Information
Shot Number: 7
Start Frame: 1297
End Frame: 1364

Figure 5.2: An Example Screenshot of the IBM MPEG-7 Annotation Tool

5.1.5 Xerces2 Java Parser

Xerces2 Java Parser [41] is an XML parser of the Apache Xerces family. Xerces2 parses XML documents according to the XML 1.1 Recommendation. In this study, Xerces2 parser is used to parse the Mpeg-7 XML files which the IBM MPEG-7 Annotation Tool produces as its output.

5.2 Architecture

The architecture of the prototype system is shown in Figure 5.3. The architecture consists of four main parts:

1- FOOD
2- FKB
3- Bridge
4- User Interface

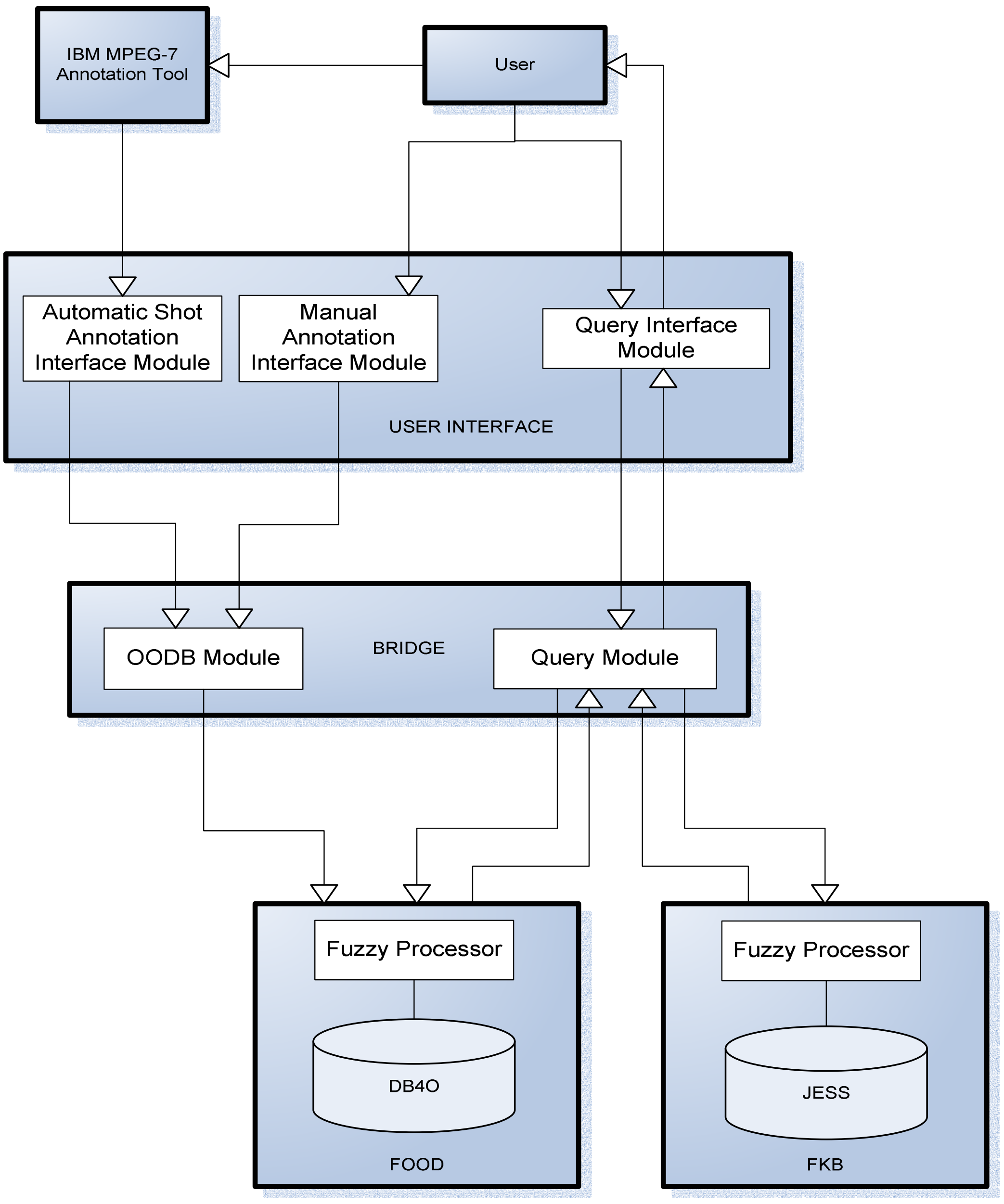


Figure 5.3: Representation of the Architecture of the Prototype System

The bridge consists of two modules: OODB Module and Query Module. OODB Module is responsible for storing the complex objects of video data. Query Module is responsible for processing queries, connecting FOOD and FKB, combining the query results and sending the results to user interface.

The user interface consists of three modules: Automatic Annotation Interface Module, Manual Annotation Interface Module and Query Interface Module.

Automatic Annotation Interface Module gets the output of IBM MPEG-7 Annotation Tool as its input. It parses the MPEG XML file, and stores the shot information in the database. This module includes scene annotation interface.

Manual Annotation Interface Module consists of annotation interfaces, which are used to annotate a football game video. This module includes video annotation, object annotation, sequence annotation and event annotation interfaces.

Query Interface Module consists of querying interfaces, which are used to query the data stored in the database. This module includes semantic, temporal, spatial, spatio-temporal, hierarchical query and class hierarchy interfaces.

5.3 Annotation

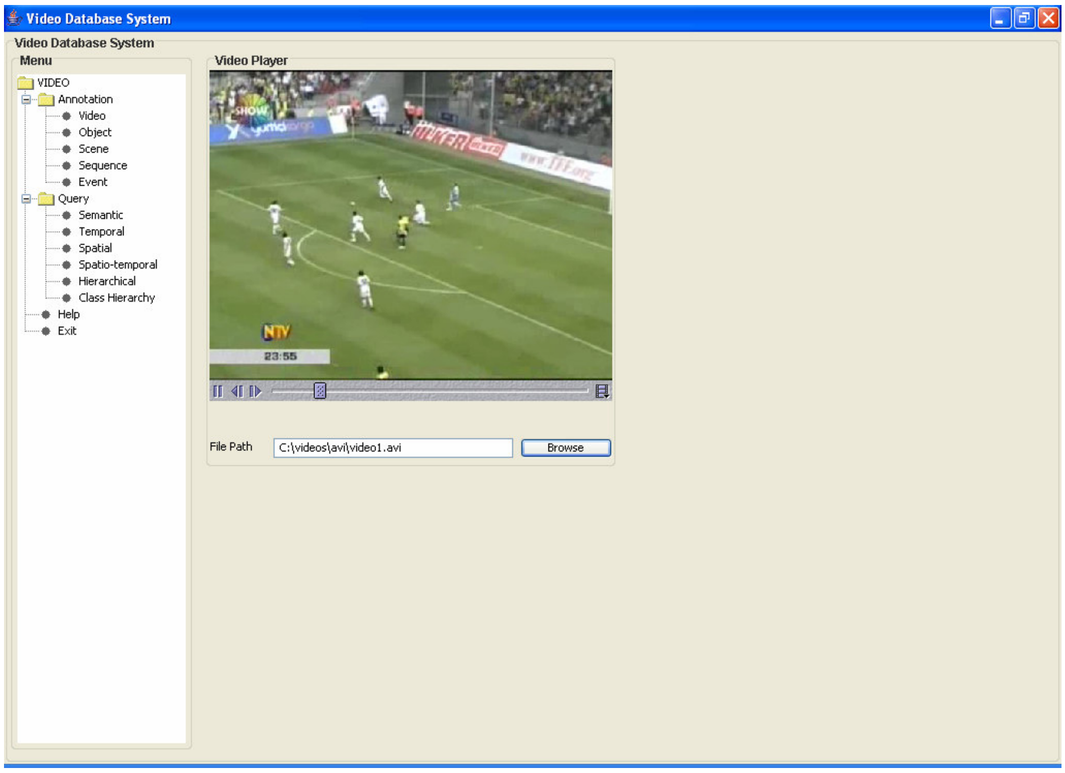
A football game video can be annotated and stored in our database by using the annotation interfaces. In annotation process, the objects of the classes in our model explained in Chapter 3 are created and stored in the object-oriented database.

In all of the annotation interfaces, a video player component appears in the window as shown in Figure 5.4. The user can select a video from the file system by using the *Browse* button in the video player. The player provides various opportunities such as pausing, restarting and stopping the video, positioning the video stream at a specified time, disabling and enabling the sound of the video.

When a video is selected, it becomes the active video and all the annotation, which is done after the selection, will be for the selected video. The user can display different annotation interfaces by selecting it from the menu on the left-hand side of the window.

5.3.1 Video Annotation

Video Annotation Interface is used for providing video information such as name, title, date and description of video. When information is entered and the *Save* button is pressed, a new instance of Video class is created and stored in the object-oriented database. Video Annotation Interface is shown in Figure 5.5.



File Path C:\videos\avi\video1.avi

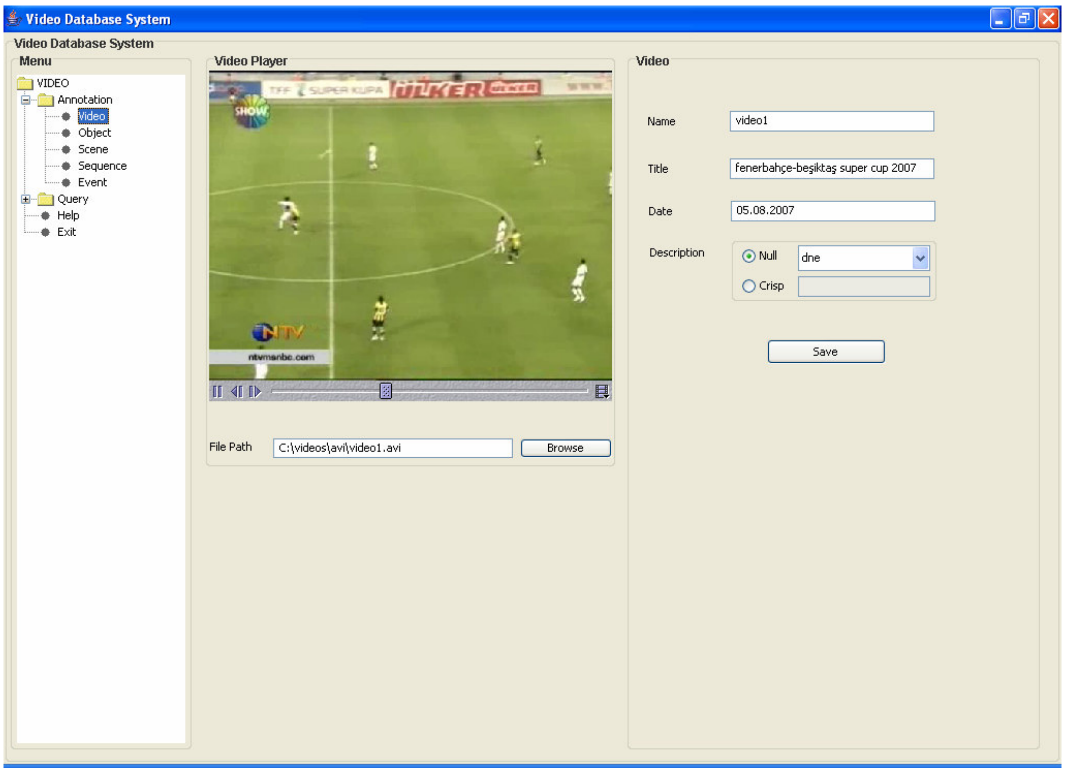
Figure 5.4: Video Player of the Prototype System

Figure 5.6 shows the panel which is used for null attributes, eg. *description*. Since the value of a null attribute can be null (dne, ni, unk) or crisp, there are two sections in the panel to choose one of them.

5.3.2 Object Annotation

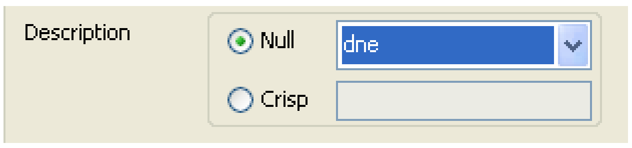
Object Annotation Interface is used for defining the video objects appearing in a video. Object Annotation Interface is shown in Figure 5.7.

There are four different object types: *Player*, *Ball*, *Match* and *TalentedPlayer*. When the object type is selected, the details panel below is refreshed for the selected object type and the needed fields for the selected type are displayed. As shown in Figure 5.7, the details of TalentedPlayer class (*Name*, *Age*, *Dress Number*, *Field Position*, *Shot Accuracy*, *Speed*, *and Ball Control*) are displayed in the *Player Details* panel. The fields *Object Name* and *Description* correspond to the attributes of the base class Object and exist for all object types.



Name: video1
Title: fenerbahçe-beşiktaş super cup 2007
Date: 05.08.2007
Description: [✓ Null] dne / Crisp
File Path: C:\videos\avi\video1.avi

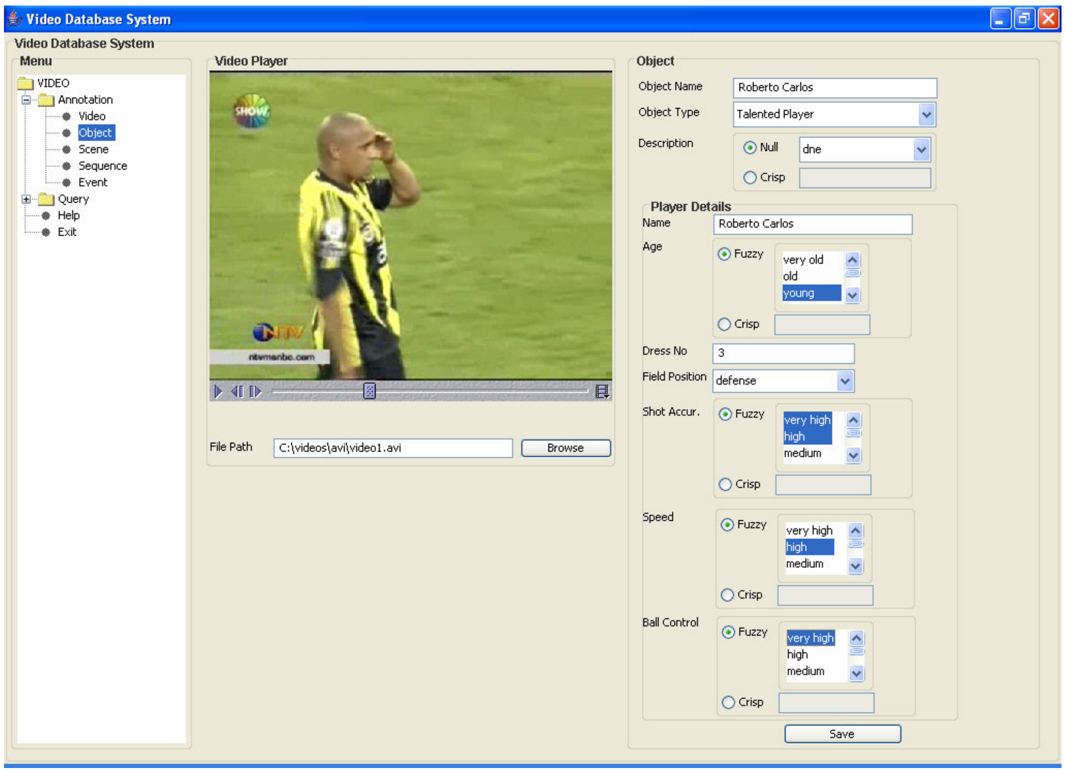
Figure 5.5: User Interface for Video Annotation



Description: [✓ Null] dne / Crisp

Figure 5.6: Definition of the Null Attribute *description*

When information is entered and the *Save* button is pressed, a new instance of TalentedPlayer class is created and stored in the object-oriented database. If another object type were selected, an instance of the corresponding class would be created. Since the class TalentedPlayer is uncertain at object/class level, the object membership degree is automatically calculated and stored when a TalentedPlayer instance is created and saved to database.



File Path C:\videos\avi\video1.avi

Object
Object Name: Roberto Carlos
Object Type: Talented Player
Description: [✓ Null] dne; Crisp

Player Details
Name: Roberto Carlos
Age: [✓ Fuzzy] very old, old, [✓ young]; Crisp
Dress No: 3
Field Position: defense
Shot Accur.: [✓ Fuzzy] [✓ very high], [✓ high], medium; Crisp
Speed: [✓ Fuzzy] very high, [✓ high], medium; Crisp
Ball Control: [✓ Fuzzy] [✓ very high], high, medium; Crisp

Figure 5.7: User Interface for Object Annotation

Figure 5.8 shows the panel which is used for fuzzy attributes, eq. *age*. Since the value of a fuzzy attribute can be fuzzy or crisp, there are two sections in the panel to choose one of them. Multiple fuzzy values can be selected since a fuzzy attribute is a multivalued attribute.

Figure 5.9 shows the panel which is used for incomplete attributes, eg. *audience*. Since the value of an incomplete attribute is a range with a starting and ending value, there are two sections in the panel for both of them.

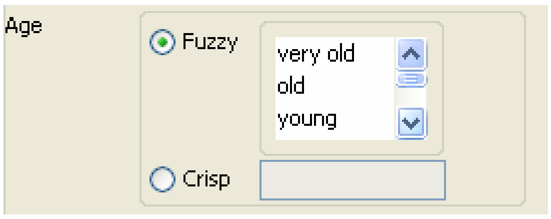


Figure 5.8: Definition of the Fuzzy Attribute *age*

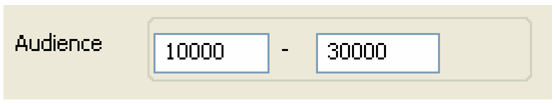


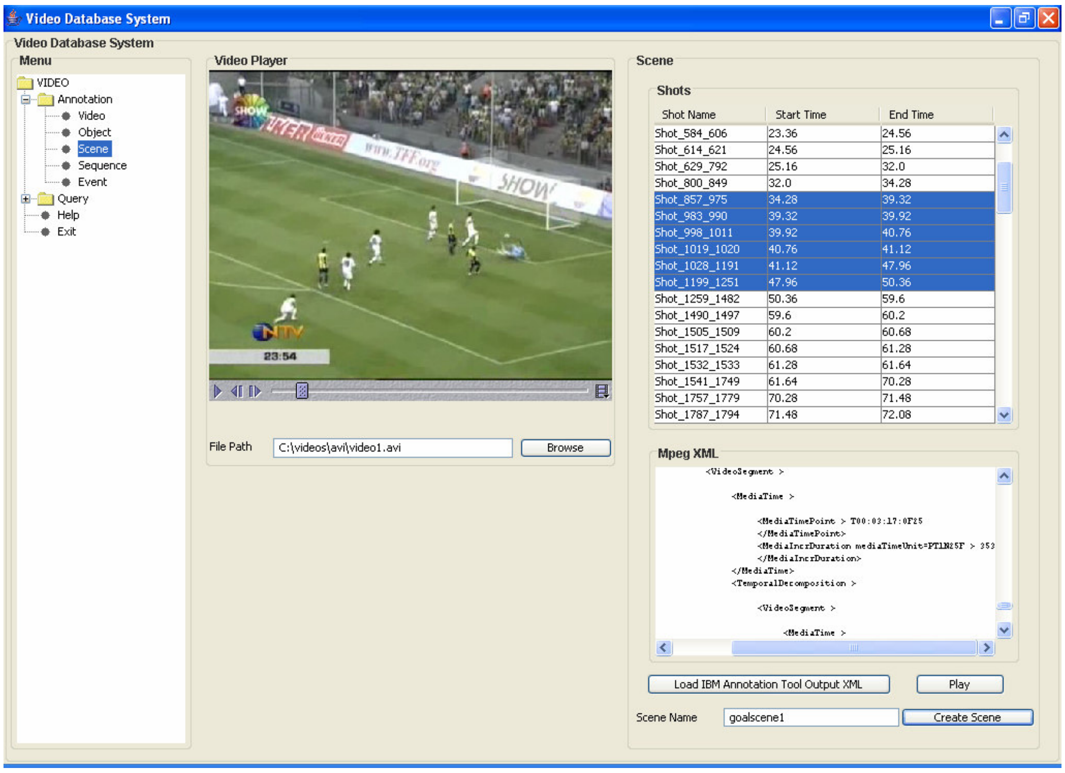
Figure 5.9: Definition of the Incomplete Attribute *audience*

5.3.3 Scene Annotation

Scene Annotation Interface is used for defining the video scenes and is shown in Figure 5.10. Before using this interface, the user should create the MPEG XML file of the corresponding video by using the IBM MPEG-7 Annotation Tool. An example MPEG XML output of the tool is provided for the reader in Appendix C.

The user can select the previously created XML file from the file system by using the *Load IBM Annotation Tool Output XML* button. After selecting the XML file, the content of the file is displayed in the *Mpeg XML* panel and the shots, which are created by IBM MPEG-7 Annotation Tool, are listed in the *Shots* panel with shot name, start time and end time information. If the user selects any shot from the list and presses the *Play* button, then the selected shot is played in the Video Player. Therefore, the user can play any shot and combine the semantically closer shots, which make up a video scene. IBM MPEG-7 Annotation Tool is utilized in this study in a similar way with the study included in [46].

If the user selects a number of shots from the shot list, enters the scene name and presses the *Create Scene* button, a new instance of the Scene class is created in the object-oriented database. The combined shots are also stored in the object-oriented database in the form of instances of the Shot class. The value of the *parentScene* attribute of the combined Shot instances is set to be the newly created Scene instance.



File Path C:\videos\avi\video1.avi

Shot Name	Start Time	End Time
Shot_584_606	23.36	24.56
Shot_614_621	24.56	25.16
Shot_629_792	25.16	32.0
Shot_800_849	32.0	34.28
Shot_857_975	34.28	39.32
Shot_983_990	39.32	39.92
Shot_998_1011	39.92	40.76
Shot_1019_1020	40.76	41.12
Shot_1028_1191	41.12	47.96
Shot_1199_1251	47.96	50.36
Shot_1259_1482	50.36	59.6
Shot_1490_1497	59.6	60.2
Shot_1505_1509	60.2	60.68
Shot_1517_1524	60.68	61.28
Shot_1532_1533	61.28	61.64
Shot_1541_1749	61.64	70.28
Shot_1757_1779	70.28	71.48
Shot_1787_1794	71.48	72.08

```
<VideoSegment >

    <MediaTime >

        <MediaTimePoint > T00:03:17:0F25
        </MediaTimePoint>
        <MediaIncrDuration mediaTimeUnit=PT1N25F > 353
        </MediaIncrDuration>
    </MediaTime>
    <TemporalDecomposition >

        <VideoSegment >

            <MediaTime >
```

Scene Name goalscene1

Figure 5.10: User Interface for Scene Annotation

5.3.4 Sequence Annotation

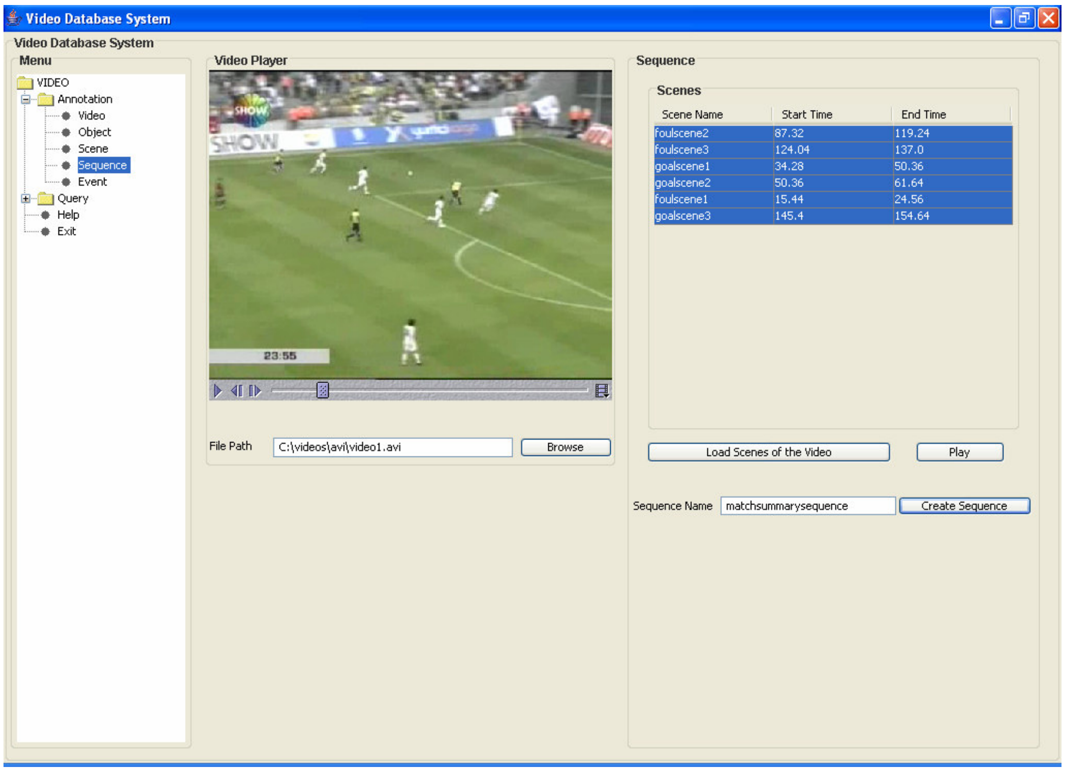
Sequence Annotation Interface is used for defining the video sequences and is shown in Figure 5.11. When the user presses the *Load Scenes of the Video* button, the previously created scenes are listed in the *Scenes* panel with scene name, start time and end time information. If the user selects any scene from the list and presses the *Play* button, then the selected scene is played in the Video Player. Therefore, the user can play any scene and combine the semantically closer scenes, which make up a video sequence.

If the user selects a number of scenes from the scene list, enters the sequence name and presses the *Create Sequence* button, a new instance of the Sequence class is created in the object-oriented database. The value of the *parentSequence* attribute of the combined Scene instances is set to be the newly created Sequence instance.

5.3.5 Event Annotation

Event Annotation Interface is used for defining video events happening within a video shot. Event Annotation Interface is shown in Figure 5.12.

There are two different event types: *Goal* and *Foul*. When the event type is selected, the details panel below is refreshed for the selected event type and the needed fields for the selected type are displayed. As shown in Figure 5.12, the details of Foul class (*Harshness*, Y*ellow Card*, *Red Card*, and *Field Position*) are displayed in the window.



File Path C:\videos\avi\video1.avi

Scene Name	Start Time	End Time
foulscene2	87.32	119.24
foulscene3	124.04	137.0
goalscene1	34.28	50.36
goalscene2	50.36	61.64
foulscene1	15.44	24.56
goalscene3	145.4	154.64

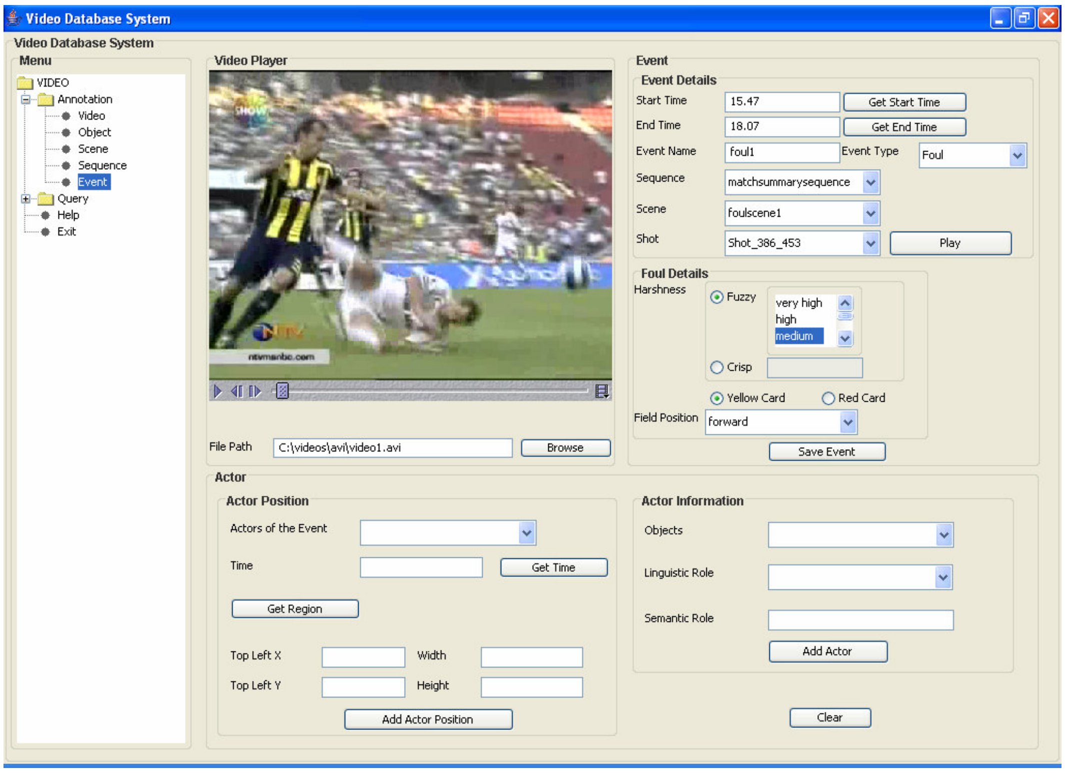
Sequence Name matchsummarysequence

Figure 5.11: User Interface for Sequence Annotation

The fields *Start Time*, *End Time* and *Event Name* correspond to the attributes of the base class Event and exist for all event types. When the user stops the Video Player at a specific time and presses the *Get Start Time* button, the time of the Video Player is displayed in the *Start Time* field. When the user stops the Video Player at a specific time and presses the *Get End Time* button, the time of the Video Player is displayed in the *End Time* field.

When Event Annotation window is loaded, the *Sequence* list is filled with the sequences of the selected video. When the user selects a sequence from the *Sequence* list, the scenes in the selected sequence are listed in the *Scene* list. When the user selects a scene from the *Scene* list, the shots in the selected scene are listed in the *Shot* list. When the user presses the *Play* button, the selected shot is played in the Video Player. This opportunity provides an easy way for the user to find an event in the selected video.

At the bottom of the window, there is *Actor* panel to enter information about the actors of an event. At the right-hand side of the *Actor* panel, there is *Actor Information* panel for selecting the objects, which have a role in the event. As represented in Figure 5.13, when the Event Annotation window is loaded, the *Objects* list is filled with the list of objects of the selected video. In the *Linguistic Role* list, the linguistic roles agent, object and recipient are listed.



File Path: C:\videos\avi\video1.avi

Start Time: 15.47
End Time: 18.07
Event Name: foul1
Event Type: Foul
Sequence: matchsummarysequence
Scene: foulscene1
Shot: Shot_386_453

Harshness: [✓ Fuzzy] medium / Crisp
[✓ Yellow Card] / Red Card
Field Position: forward

Actors of the Event:
Time:
Top Left X:
Width:
Top Left Y:
Height:

Objects:
Linguistic Role:
Semantic Role:

Figure 5.12: User Interface for Event Annotation

When the user selects an object from the *Objects* list, selects a linguistic role from the *Linguistic Role* list, enters the semantic role of the object and presses the *Add Actor* button, a new instance of the Actor class is created and the corresponding actor is added to the *Actors of the Event* list of the *Actor Position* panel. The annotation of two actors: *committer* and *victim* are represented in Figure 5.14 (a) and Figure 5.14 (b). Figure 5.15 shows the *Actors of the Event* list.

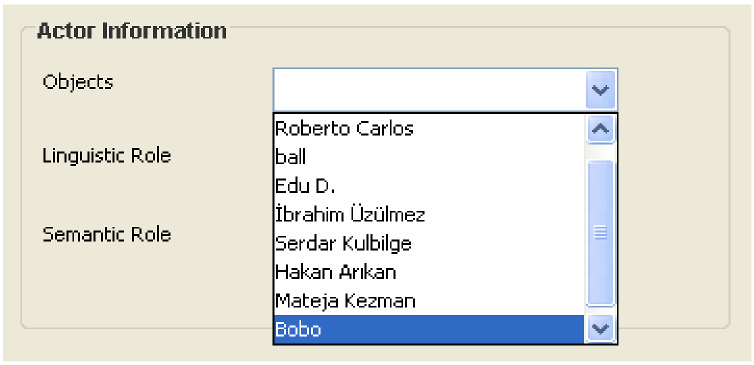
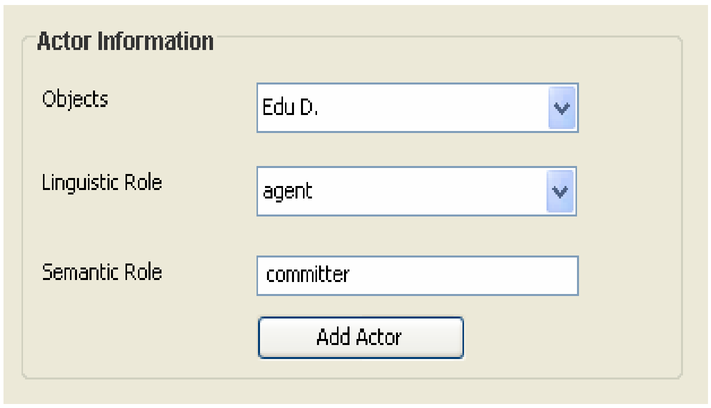


Figure 5.13: Object List of the Video

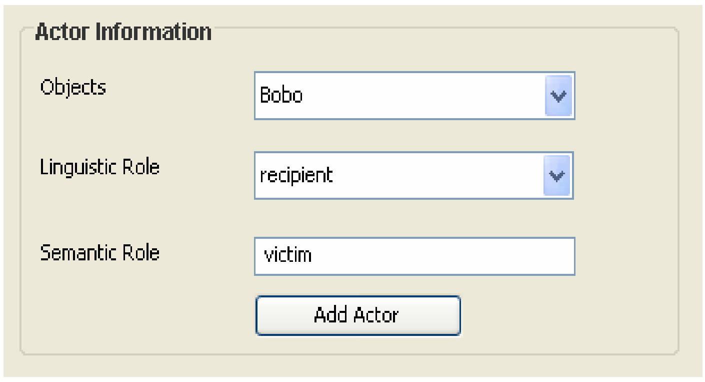
The *Actor Position* panel is used to enter the position information of the actors of the event. To enter position information for an actor in the event, the user stops the Video Player, selects the actor from the *Actors of the Event* list, and presses the *Get Time* button. The time of the Video Player is displayed in the *Time* field of the *Actor Position* panel. Then the user draws a rectangle on the Video Player, which bounds the corresponding object. After drawing the rectangle, the user presses the *Get Region* button and the coordinates of the rectangle are displayed in the *Top Left X*, *Top Left Y*, *Width* and *Height* fields of the *Actor Position* panel. If the user presses the *Add Actor Position* button, the user interface creates a new instance of Region and Position classes and adds the Position instance to the *positionList* attribute of the selected Actor instance. Figure 5.16 and Figure 5.17 show how the actor position information is entered for two different actors.

Finally, the user presses the *Save Event* button; a new instance of Foul class is created. If another event type were selected, an instance of the corresponding class would be created. The *event* attribute of the Actor instances is set to be the newly created Foul instance and the *parentShot* attribute of the Foul (inherited from the base class Event) instance is set to be the selected Shot instance from the *Shot* list of the *Event Details* panel. Then all the Actor, Position, Region and Foul instances are stored persistently in the object-oriented database.


Actor Information

Objects	Edu D.
Linguistic Role	agent
Semantic Role	committer

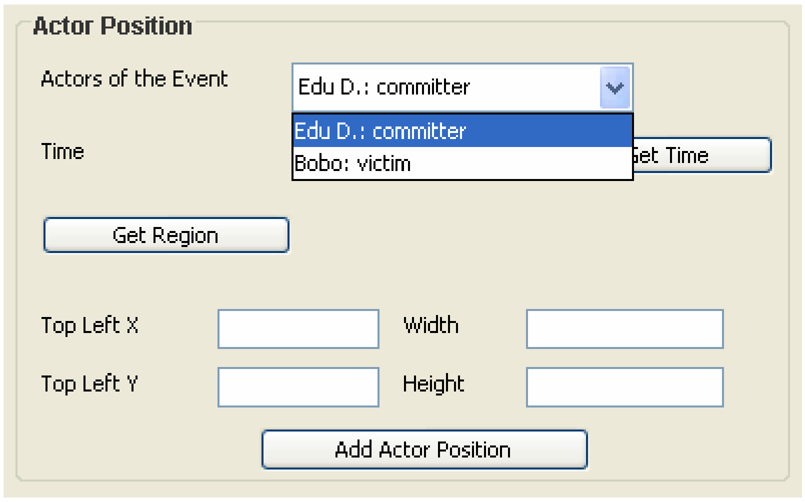
(a)


Actor Information

Objects	Bobo
Linguistic Role	recipient
Semantic Role	victim

(b)

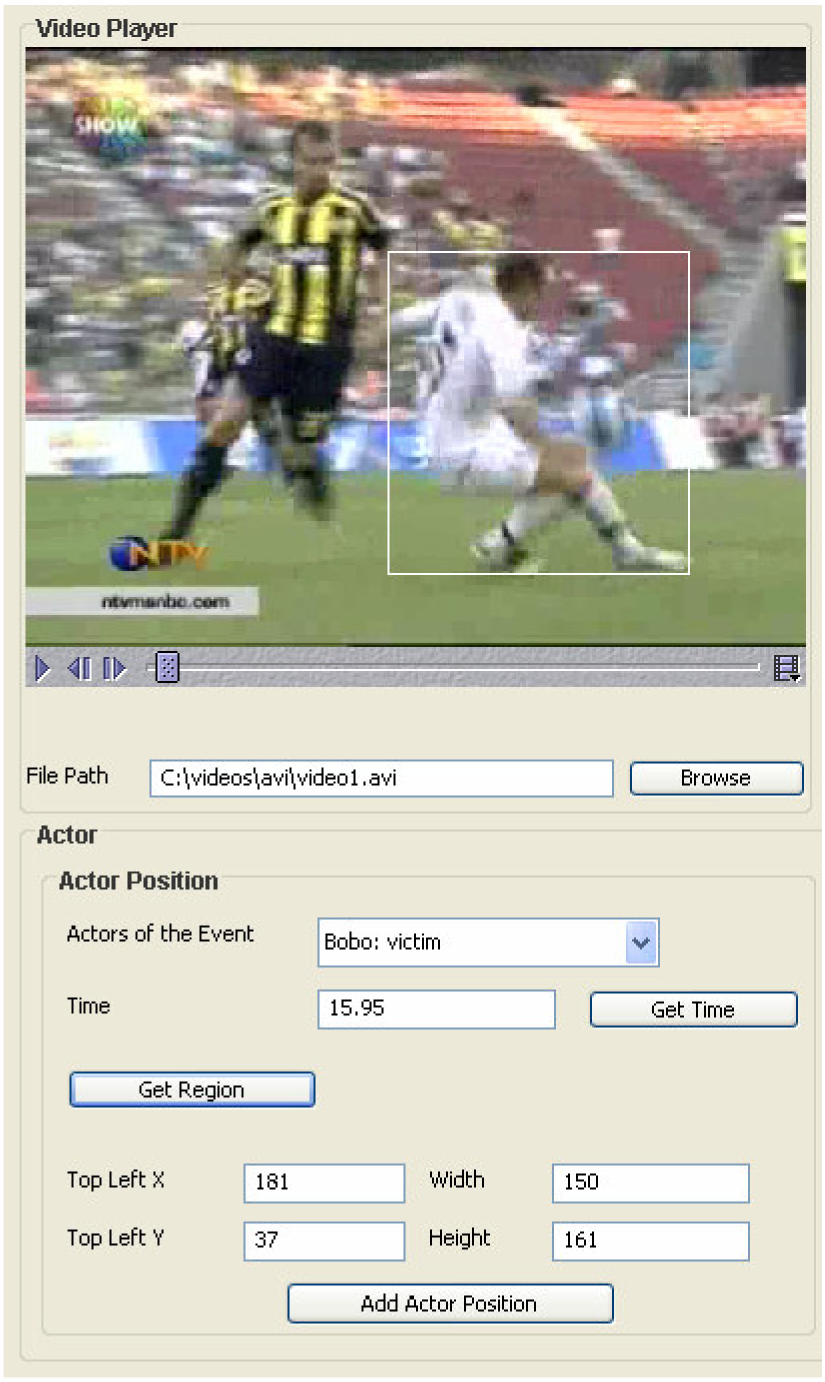
Figure 5.14: Actor Annotation


Actor Position

Actors of the Event	Edu D.: committer

Edu D.: committer
Bobo: victim

Figure 5.15: Actors of the Event



File Path: C:\videos\avi\video1.avi

Actors of the Event: Bobo: victim

Time: 15.95

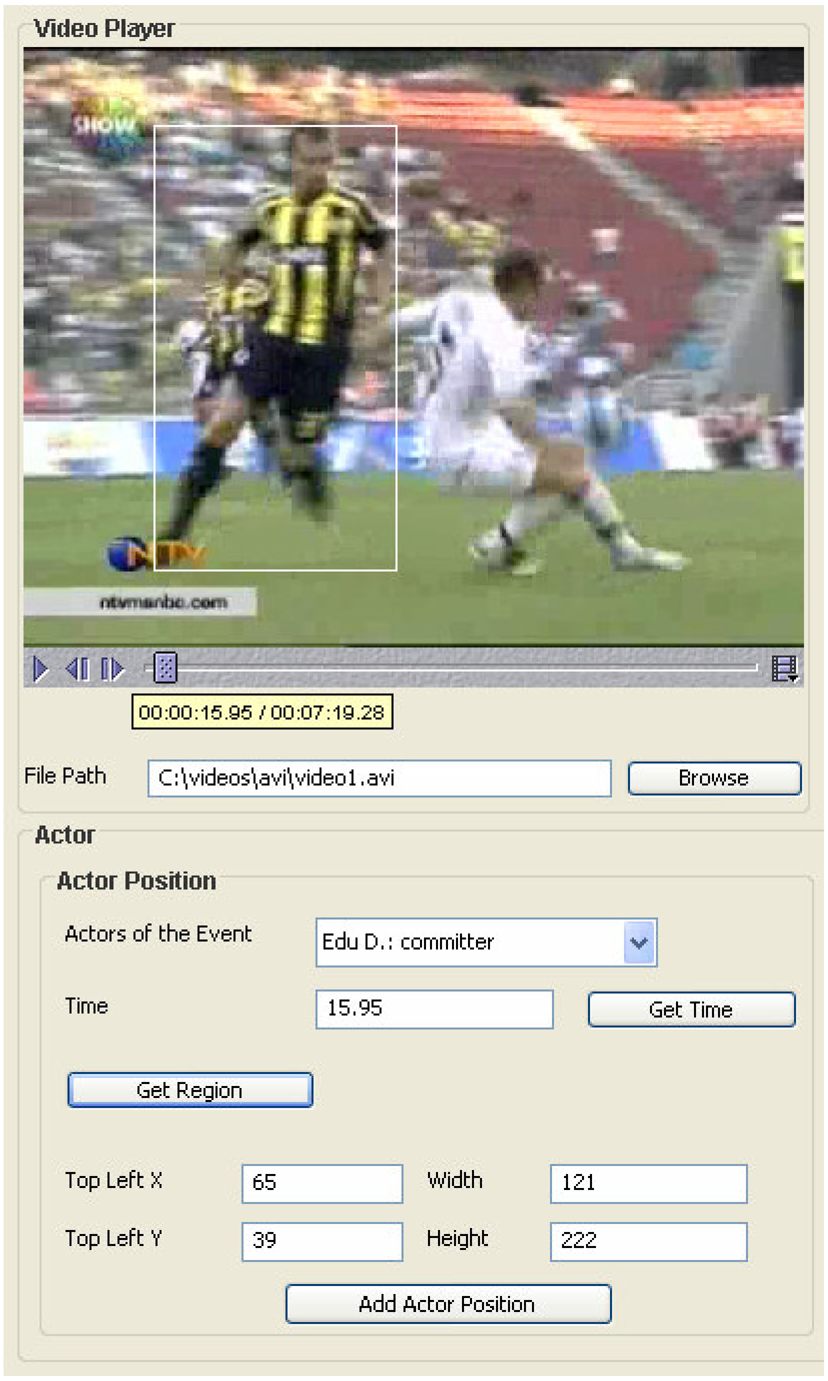
Top Left X: 181

Width: 150

Top Left Y: 37

Height: 161

Figure 5.16: Actor Position Annotation for the First Object



00:00:15.95 / 00:07:19.28

File Path: C:\videos\avi\video1.avi

Actors of the Event: Edu D.: committer

Time: 15.95

Top Left X: 65
Width: 121
Top Left Y: 39
Height: 222

Figure 5.17: Actor Position Annotation for the Second Object

5.4 Querying

The framework provides the user querying facilities. The supported queries are (fuzzy) semantic, temporal, (fuzzy) spatial, trajectory, regional and hierarchical queries. In this section, the query interfaces of our prototype system are explained and query examples are given. The temporal, spatial and (fuzzy) semantic rules are written in Jess language and are provided for the reader in Appendix B.

5.4.1 (Fuzzy) Semantic Queries

Semantic Query Interface provides the user to query the properties of Video, Object and Event instances stored in the database. Since some of the classes are uncertain and include uncertain attributes, the queries can be fuzzy. A generic query evaluation method is developed. The method is generic, since it does not access any football domain specific classes and data types. The method can be employed for any domain providing the base classes be used.

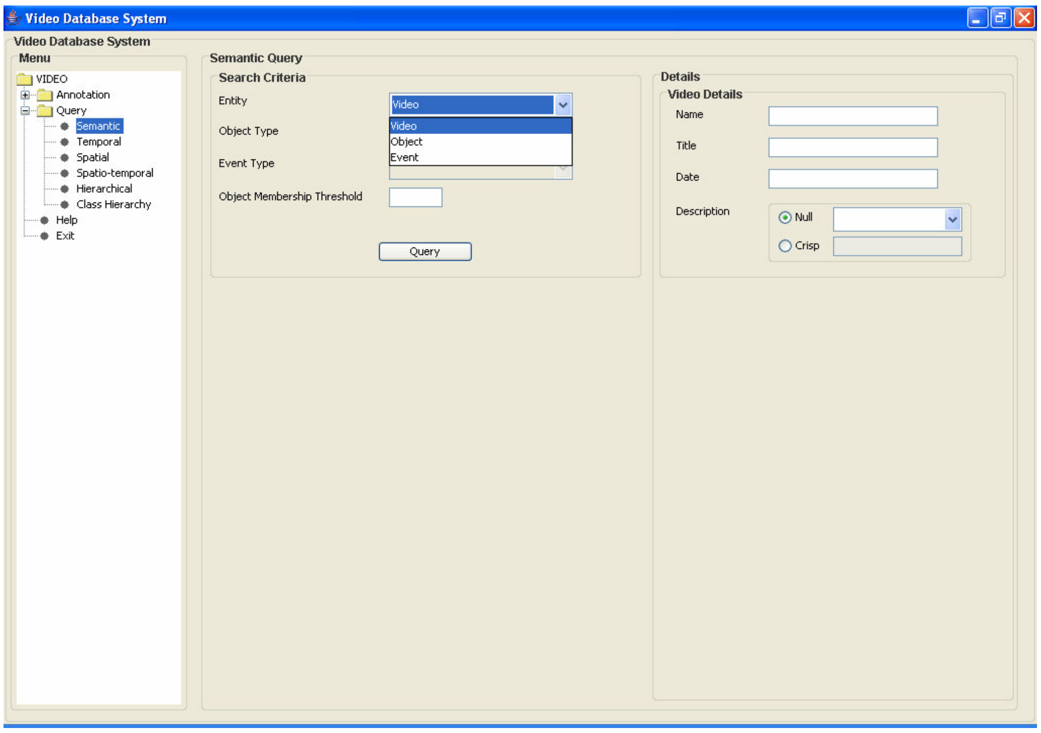
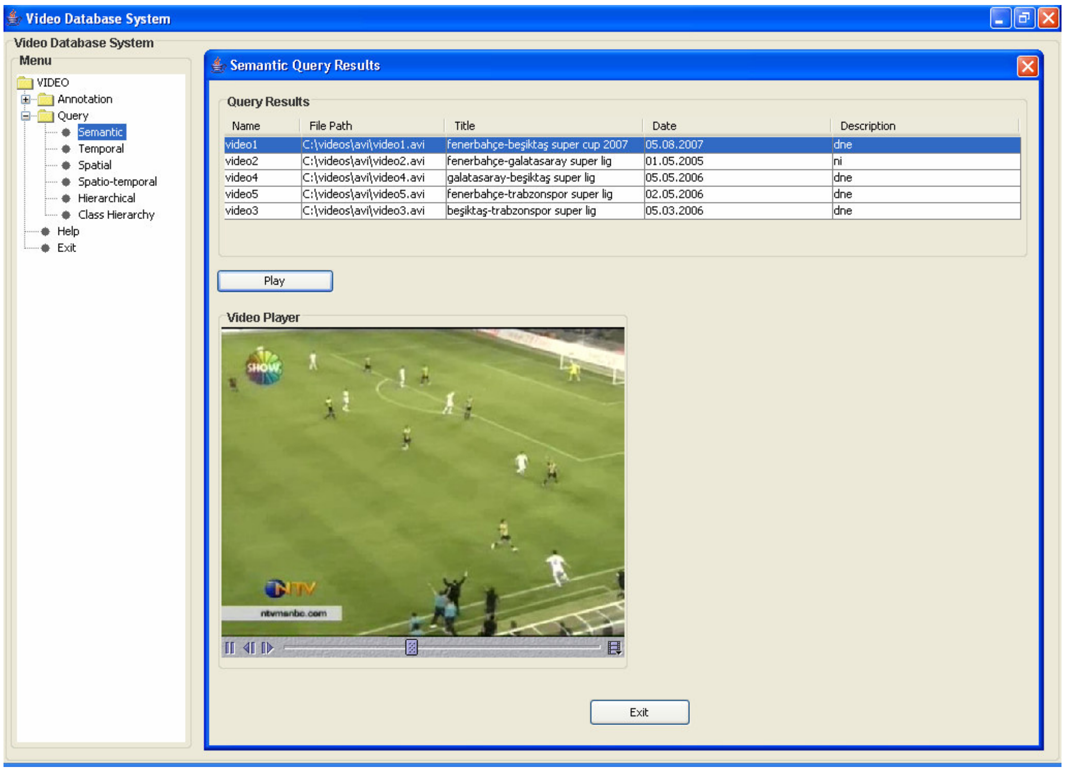


Figure 5.18: Semantic Query Interface

Semantic Query Interface is shown in Figure 5.18. There are three options in the *Entity* list: Video, Object and Event. If Video is selected from the list, *Video Details* panel is loaded as in Figure 5.18. The user might enter values to the fields *Name*, *Title*, *Date* and *Description*. When the *Query* button is pressed, the query request is sent to the bridge. The query is performed by using the query evaluation algorithm. The result list is displayed in the *Semantic Query Results* window as shown in Figure 5.19. In the *Semantic Query Results* window, the user can play a video from the result list by selecting the video and pressing the *Play* button.

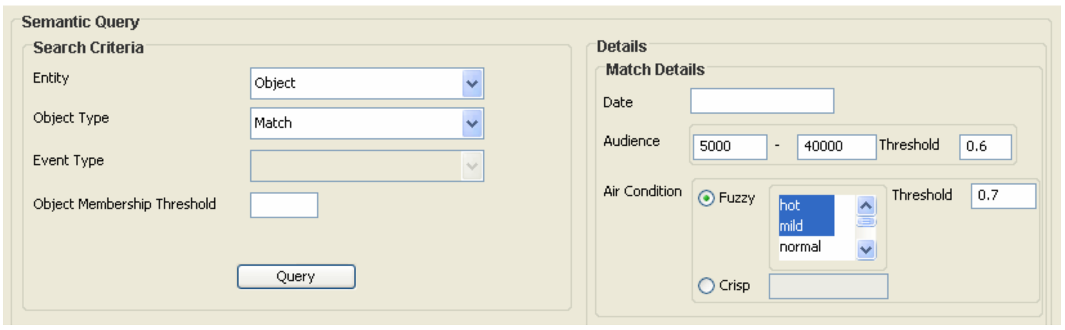


Name	File Path	Title	Date	Description
video1	C:\videos\avi\video1.avi	fenerbahçe-beşiktaş super cup 2007	05.08.2007	dne
video2	C:\videos\avi\video2.avi	fenerbahçe-galatasaray super lig	01.05.2005	ni
video4	C:\videos\avi\video4.avi	galatasaray-beşiktaş super lig	05.05.2006	dne
video5	C:\videos\avi\video5.avi	fenerbahçe-trabzonspor super lig	02.05.2006	dne
video3	C:\videos\avi\video3.avi	beşiktaş-trabzonspor super lig	05.03.2006	dne

Figure 5.19: Semantic Query Interface (Video is selected)

If Object is selected from the *Entity* list, the *Object Type* list becomes active, and the user can select the object type (Player, Match, Talented Player) from the list. When the user selects an object type, the details panel for the selected type is loaded as represented in Figure 5.20. The user might enter values to the fields in the *Details* panel. When the *Query* button is pressed, the query request is sent to the bridge. The query is performed by using the query evaluation algorithm. The result list is displayed in the *Semantic Query Results* panel. Figure 5.20 (a) shows the querying of Match objects and Figure 5.20 (b) shows the query results. As it can be seen in Figure 5.20 (b), the inclusion degrees for the attributes *audience* and *airCondition* are calculated and shown in the results.

Figure 5.21 (a) shows the querying of TalentedPlayer objects and Figure 5.21 (b) shows the query results. As it can be seen in Figure 5.21 (b), the inclusion degrees of the fuzzy attributes *age*, *shotAccuracy*, *speed*, and *ballControl* are calculated and shown in the results. Since the class TalentedPlayer is also uncertain at the object/class level, a threshold for the object membership degrees can be specified. The membership degrees of the objects in the result list are also displayed in the *Semantic Query Results* window.


Semantic Query

Search Criteria

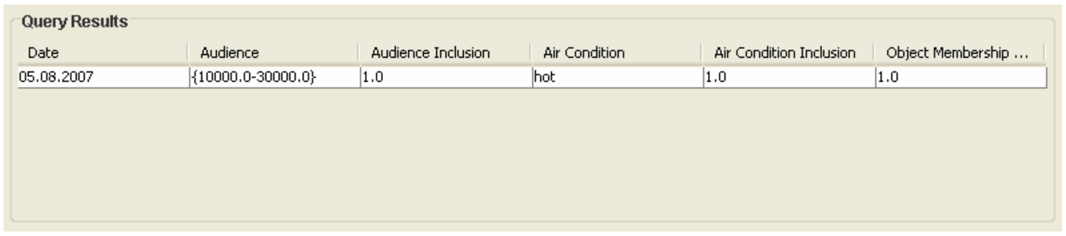
Entity	Object
Object Type	Match
Event Type	
Object Membership Threshold	

Details

Match Details

Date			
Audience	5000 - 40000	Threshold	0.6
Air Condition	[✓ Fuzzy] hot, mild (selected: hot, mild; normal) / Crisp	Threshold	0.7

(a) Query Specification


Query Results

Date	Audience	Audience Inclusion	Air Condition	Air Condition Inclusion	Object Membership ...
05.08.2007	{10000.0-30000.0}	1.0	hot	1.0	1.0

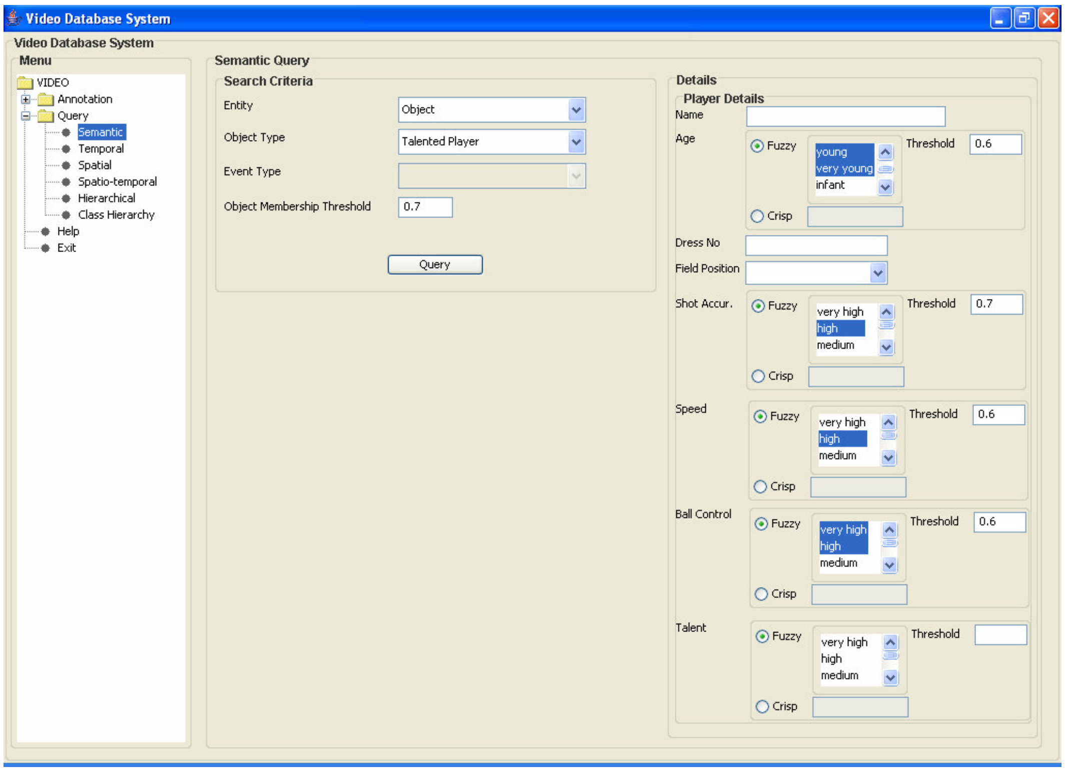
(b) Query Results

Figure 5.20: Semantic Query Interface (Object - Match is selected)

Figure 5.22 (a) shows the querying of TalentedPlayer objects by querying the value of a derived attribute *talent*. The value is specified as "very high" and the threshold is given as 0.7. The query results including the rule conclusion are represented in Figure 5.22 (b).

Figure 5.23 (a) shows the querying of TalentedPlayer objects by querying the value of a derived attribute *talent*. This time, the value is specified as "medium" and the threshold is 0.6. The query results including the rule conclusion are represented in Figure 5.23 (b).

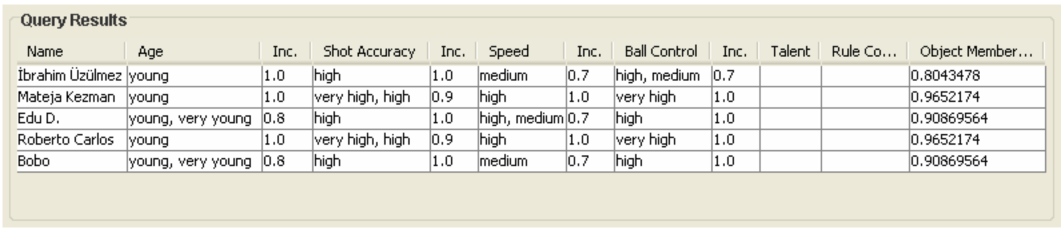
If Event is selected from the *Entity* list, the *Event Type* list becomes active, and the user can select the event type (Foul, Goal) from the list. When the user selects an event type, the details panel for the selected type is loaded as represented in Figure 5.24 (a). The user might enter values to the fields in the *Details* panel. When the *Query* button is pressed, the query request is sent to the bridge. The query is performed by using the query evaluation algorithm. The result list is displayed in the *Query Results* panel. Figure 5.24 (b) shows the query results for the class Foul. In the *Semantic Query Results* window, the user can play an event from the result list by selecting the event and pressing the *Play* button.



Entity: Object
Object Type: Talented Player
Event Type:
Object Membership Threshold: 0.7

Name:
Age: Fuzzy — young, very young (selected), infant — Threshold 0.6; Crisp
Dress No:
Field Position:
Shot Accur.: Fuzzy — very high, high (selected), medium — Threshold 0.7; Crisp
Speed: Fuzzy — very high, high (selected), medium — Threshold 0.6; Crisp
Ball Control: Fuzzy — very high, high (selected), medium — Threshold 0.6; Crisp
Talent: Fuzzy — very high, high, medium — Threshold; Crisp

(a) Query Specification

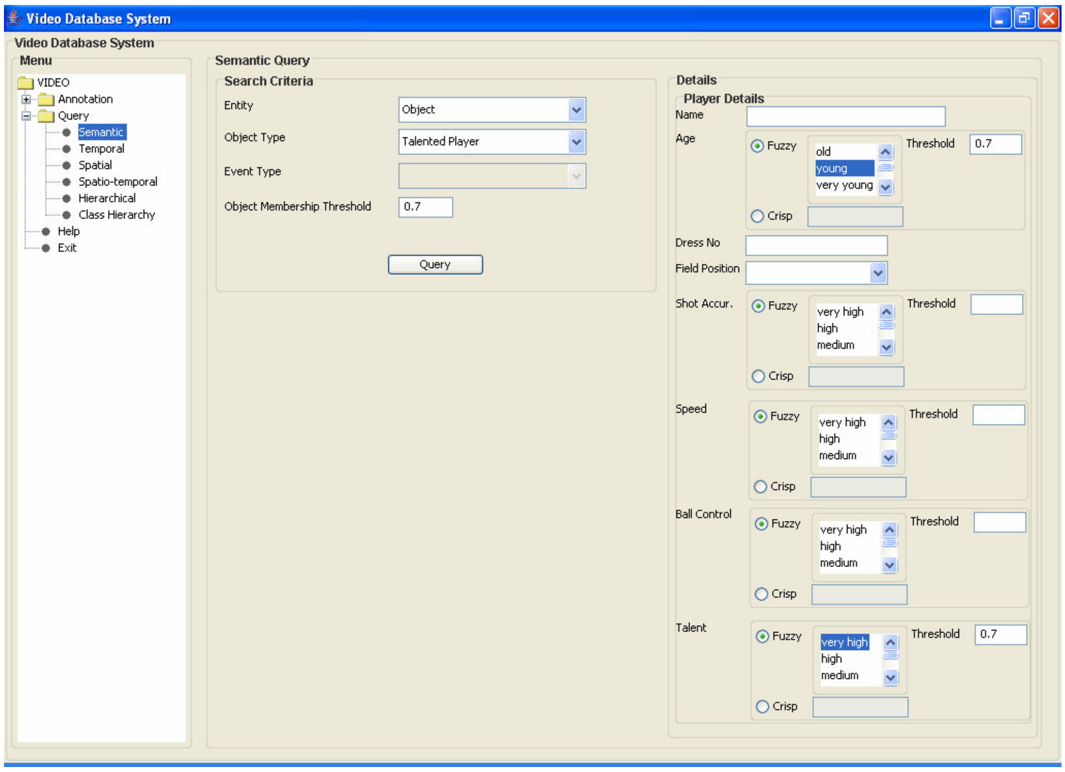

Query Results

Name	Age	Inc.	Shot Accuracy	Inc.	Speed	Inc.	Ball Control	Inc.	Talent	Rule Co...	Object Member...
İbrahim Üzülmez	young	1.0	high	1.0	medium	0.7	high, medium	0.7			0.8043478
Mateja Kezman	young	1.0	very high, high	0.9	high	1.0	very high	1.0			0.9652174
Edu D.	young, very young	0.8	high	1.0	high, medium	0.7	high	1.0			0.90869564
Roberto Carlos	young	1.0	very high, high	0.9	high	1.0	very high	1.0			0.9652174
Bobo	young, very young	0.8	high	1.0	medium	0.7	high	1.0			0.90869564

(b) Query Results

Figure 5.21: Semantic Query Interface (Object – TalentedPlayer is selected)

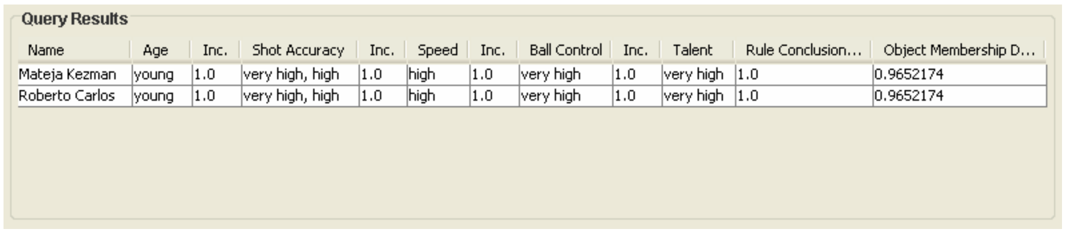
For the three types of entity, the user might enter a threshold value for the object membership degrees of the requested objects. This field is meaningful if the selected class is uncertain at the object/class level such as Foul or TalentedPlayer classes. If the field is left empty, its default value will be considered to be 0.



Entity	Object
Object Type	Talented Player
Event Type	
Object Membership Threshold	0.7

Name			
Age	[✓ Fuzzy] Crisp	old / [✓ young] / very young	Threshold 0.7
Dress No			
Field Position			
Shot Accur.	[✓ Fuzzy] Crisp	very high / high / medium	Threshold
Speed	[✓ Fuzzy] Crisp	very high / high / medium	Threshold
Ball Control	[✓ Fuzzy] Crisp	very high / high / medium	Threshold
Talent	[✓ Fuzzy] Crisp	[✓ very high] / high / medium	Threshold 0.7

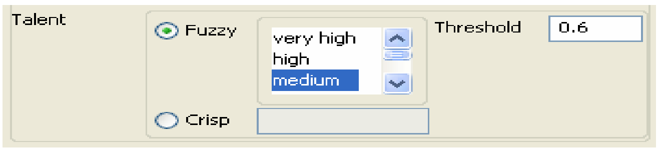
(a) Query Specification


Query Results

Name	Age	Inc.	Shot Accuracy	Inc.	Speed	Inc.	Ball Control	Inc.	Talent	Rule Conclusion...	Object Membership D...
Mateja Kezman	young	1.0	very high, high	1.0	high	1.0	very high	1.0	very high	1.0	0.9652174
Roberto Carlos	young	1.0	very high, high	1.0	high	1.0	very high	1.0	very high	1.0	0.9652174

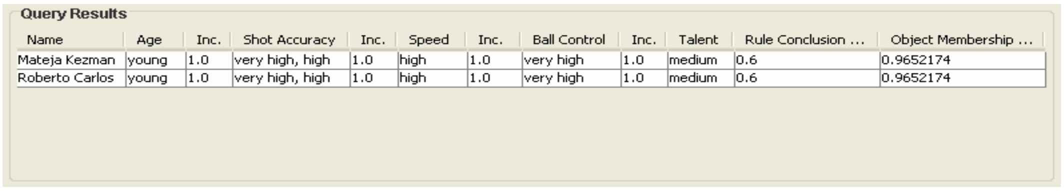
(b) Query Results

Figure 5.22: Semantic Query Interface (Object – TalentedPlayer is selected)



Talent	[✓ Fuzzy] medium
Threshold	0.6

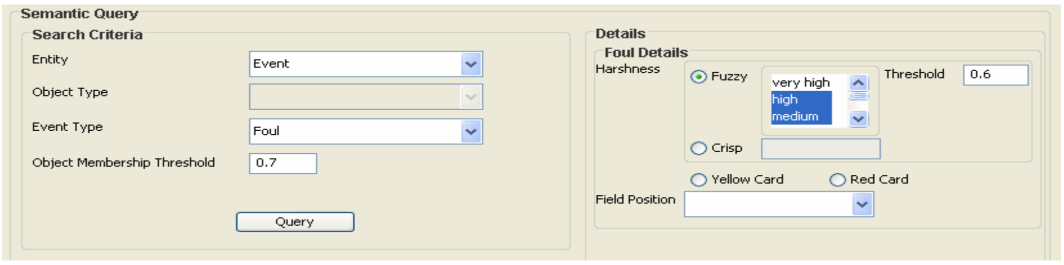
(a) Query Specification


Query Results

Name	Age	Inc.	Shot Accuracy	Inc.	Speed	Inc.	Ball Control	Inc.	Talent	Rule Conclusion ...	Object Membership ...
Mateja Kezman	young	1.0	very high, high	1.0	high	1.0	very high	1.0	medium	0.6	0.9652174
Roberto Carlos	young	1.0	very high, high	1.0	high	1.0	very high	1.0	medium	0.6	0.9652174

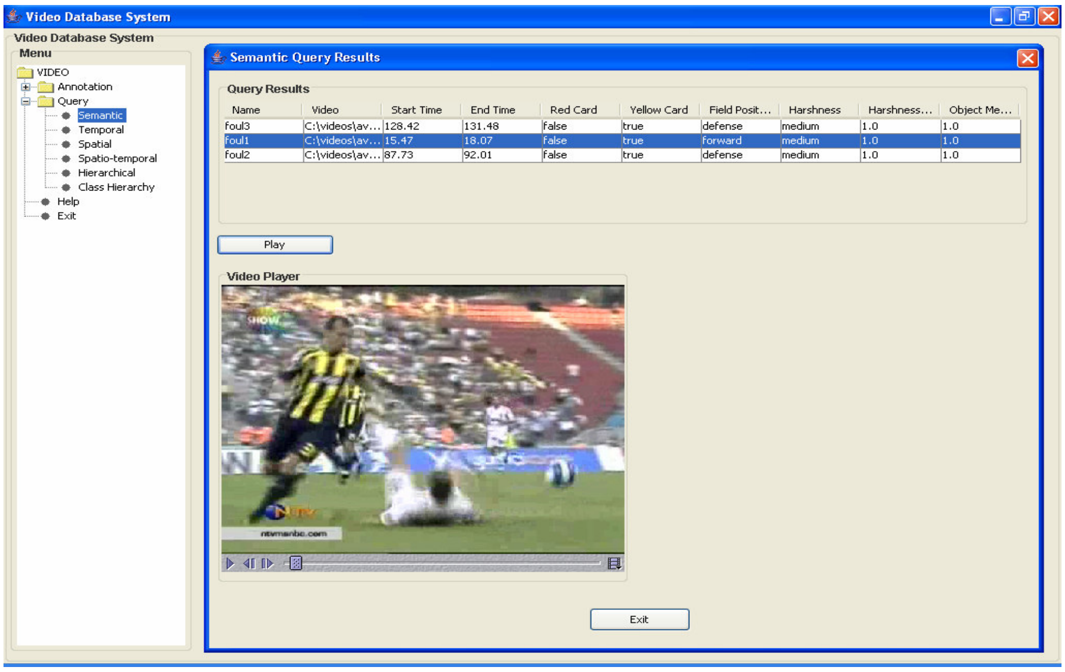
(b) Query Results

Figure 5.23: Semantic Query Interface (Object – TalentedPlayer is selected)



Entity	Event
Object Type	
Event Type	Foul
Object Membership Threshold	0.7
Harshness	[✓ Fuzzy] high, medium
Threshold	0.6
Field Position	

(a) Query Specification



Query Results

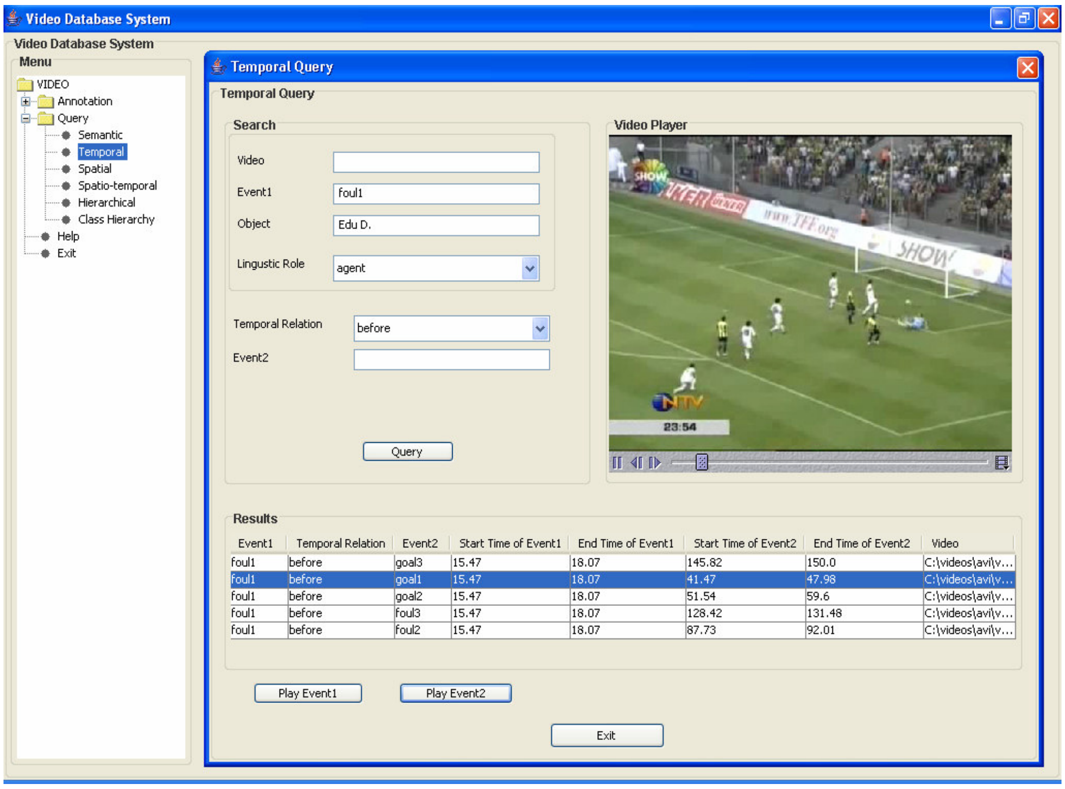
Name	Video	Start Time	End Time	Red Card	Yellow Card	Field Posit...	Harshness	Harshness...	Object Me...
foul3	C:\videos\av...	128.42	131.48	false	true	defense	medium	1.0	1.0
foul1	C:\videos\av...	15.47	18.07	false	true	forward	medium	1.0	1.0
foul2	C:\videos\av...	87.73	92.01	false	true	defense	medium	1.0	1.0

(b) Query Results

Figure 5.24: Semantic Query Interface (Event – Foul is selected)

5.4.2 Temporal Queries

Temporal Query Interface allows the users to construct temporal queries. The interface is shown in Figure 5.25. All of the search criterias are optional to perform a temporal query. If *Video* field is not left empty, the query is executed for the videos with the specified name. If that field is left empty, the query is executed for all of the videos.



Video:
Event1: foul1
Object: Edu D.
Lingustic Role: agent
Temporal Relation: before
Event2:

Event1	Temporal Relation	Event2	Start Time of Event1	End Time of Event1	Start Time of Event2	End Time of Event2	Video
foul1	before	goal3	15.47	18.07	145.82	150.0	C:\videos\avi\v...
foul1	before	goal1	15.47	18.07	41.47	47.98	C:\videos\avi\v...
foul1	before	goal2	15.47	18.07	51.54	59.6	C:\videos\avi\v...
foul1	before	foul3	15.47	18.07	128.42	131.48	C:\videos\avi\v...
foul1	before	foul2	15.47	18.07	87.73	92.01	C:\videos\avi\v...

Figure 5.25: Temporal Query Interface

When the user enters the needed information and presses the *Query* button, the constructed temporal query is executed. The results are displayed in the *Results* panel. When a record is selected from the *Results* panel and *Play Event1* button is pressed, the time interval of the Event1 is played in the Video Player on the right-hand side of the window. When the *Play Event2* button is pressed, the time interval of the Event2 is played in the Video Player.

Consider the temporal query represented in Figure 5.25:

Find all the events happening after the foul event (named as foul1) in which player Edu D. commits (has the linguistic role "agent") the foul.

The steps of the evaluation of this query are as follows:

1- The query is firstly sent to bridge and bridge sends the query to FOOD.

2- FOOD finds the Foul events in which the Player object with name "Edu D." is an actor having the linguistic role "agent".

3- FOOD also finds the Event instances, which occur in the same video of the found event in step 2.

4- The bridge transfers found Event instances to the working memory of FKB.

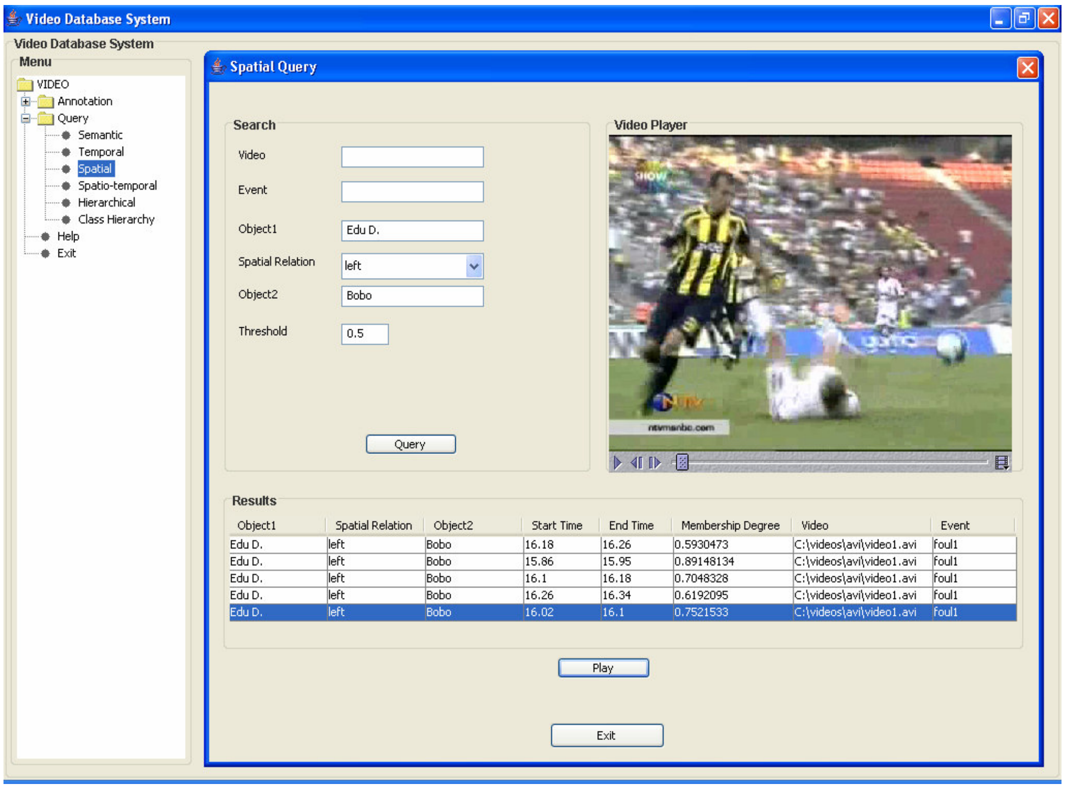
5- When the engine of FKB is run, the temporal rules will fire for the Event instances in working memory. If a rule is activated for two Event instances, it inserts a new EventToEventRelation instance to the working memory of FKB indicating the temporal relation between those two Event instances.

6- The bridge takes those newly added EventToEventRelation instances and sends the ones satisfying the temporal relation "before" to the user interface.

5.4.3 (Fuzzy) Spatial Queries

Spatial Query Interface allows the users to construct (fuzzy) spatial queries. The interface is shown in Figure 5.26. All the search criteria are optional to perform a spatial query. The threshold is for directional spatial queries, which can be fuzzy. If *Video* and *Event* fields are not left empty, the query is executed for the videos and video events with the specified names. If those fields are left empty, the query is executed for all of the videos and all of the events in those videos.

When the user enters the needed information and presses the *Query* button, the constructed spatial query is executed and the results are displayed in the *Results* panel. In the result list, the time intervals in which the selected spatial relation holds between the selected objects are displayed with their membership degrees. When a record is selected from the result list and *Play* button is pressed, the selected time interval is played in the Video Player on the right-hand side of the window.



Video:
Event:
Object1: Edu D.
Spatial Relation: left
Object2: Bobo
Threshold: 0.5

Object1	Spatial Relation	Object2	Start Time	End Time	Membership Degree	Video	Event
Edu D.	left	Bobo	16.18	16.26	0.5930473	C:\videos\avi\video1.avi	foul1
Edu D.	left	Bobo	15.86	15.95	0.89148134	C:\videos\avi\video1.avi	foul1
Edu D.	left	Bobo	16.1	16.18	0.7048328	C:\videos\avi\video1.avi	foul1
Edu D.	left	Bobo	16.26	16.34	0.6192095	C:\videos\avi\video1.avi	foul1
Edu D.	left	Bobo	16.02	16.1	0.7521533	C:\videos\avi\video1.avi	foul1

Figure 5.26: Spatial Query Interface

Consider the spatial query represented in Figure 5.26:

Find the time intervals in which player Edu D. appears on the left of player Bobo with a threshold of 0.5.

The steps of the evaluation of this query are as follows:

1- The query is firstly sent to bridge and bridge sends the query to FOOD.

2- FOOD finds the events in which the Player objects with names “Edu D.” and “Bobo” are actors.

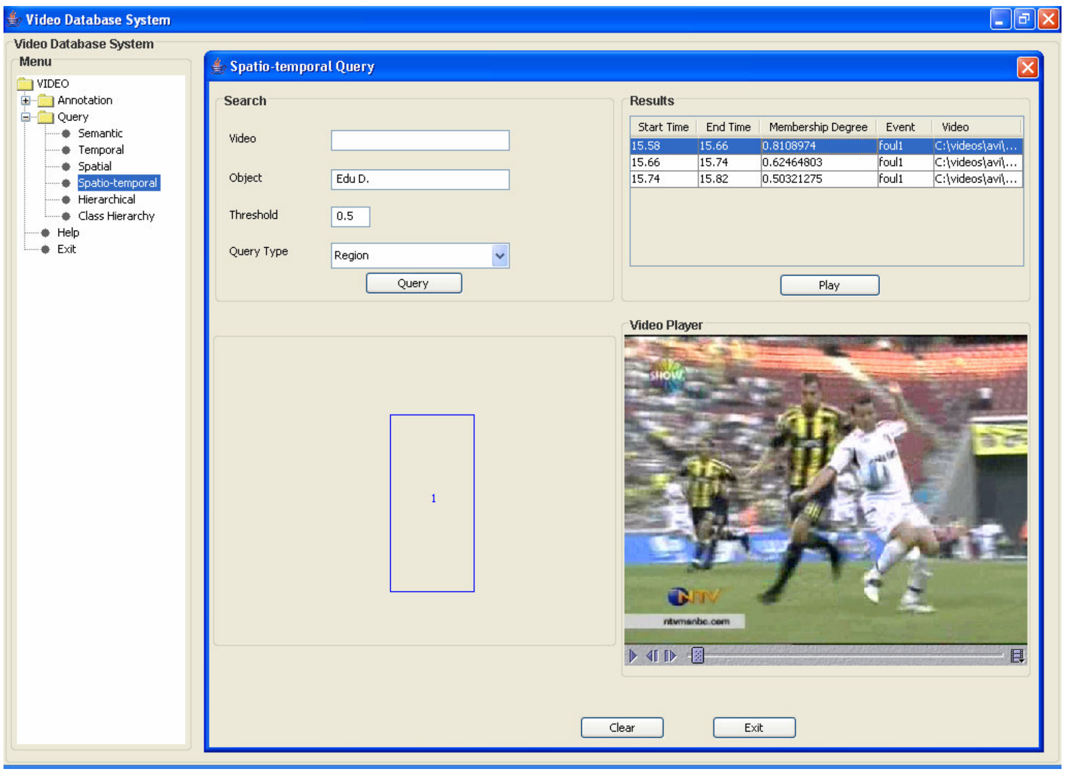
3- The bridge transfers the position list of the found Actor instances to the working memory of FKB.

4- When the engine of FKB is run, the spatial rules will fire for the Position instances in working memory. If a rule is activated for two Position instances, it inserts a new ActorToActorRelation instance to the working memory of FKB indicating the spatial relation between those two Position instances. If the relation is directional, membership degree is also calculated when the rule is executed.

5- The bridge takes those newly added ActorToActorRelation instances and sends the ones satisfying the specified spatial relation "left" and the specified threshold "0.5" to the user interface.

5.4.4 Spatio-temporal Queries

Spatio-temporal Query Interface allows the users to construct regional and trajectory queries. When *Region* is selected from the *Query Type,* the interface takes the Regional Query mode. The interface for regional query is shown in Figure 5.27. All the search criteria are optional to perform a regional query.



Video:
Object: Edu D.
Threshold: 0.5
Query Type: Region
Query

Start Time	End Time	Membership Degree	Event	Video
15.58	15.66	0.8108974	foul1	C:\videos\avi\...
15.66	15.74	0.62464803	foul1	C:\videos\avi\...
15.74	15.82	0.50321275	foul1	C:\videos\avi\...

Figure 5.27: Spatio-temporal Query Interface (Regional Query)

If *Video* and *Object* fields are not left empty, the query is executed for the videos and video objects with the specified names. If those fields are left empty, the query is executed for all of the videos and all of the objects in those videos.

The region is drawn in the drawing panel, which is on the left-bottom of the window. Since an exact region can not be specified, a threshold can be given which is used to eliminate the found regions which match with the specified region with a membership degree smaller than the threshold.

The region matching formula used to calculate the membership degree, which is taken from the study in [3], is as follows:

$$\mu = \text{intersected\_area (Region1, Region2)} \;/\; \text{minimum\_area (Region1, Region2)} \qquad \text{(F-1)}$$

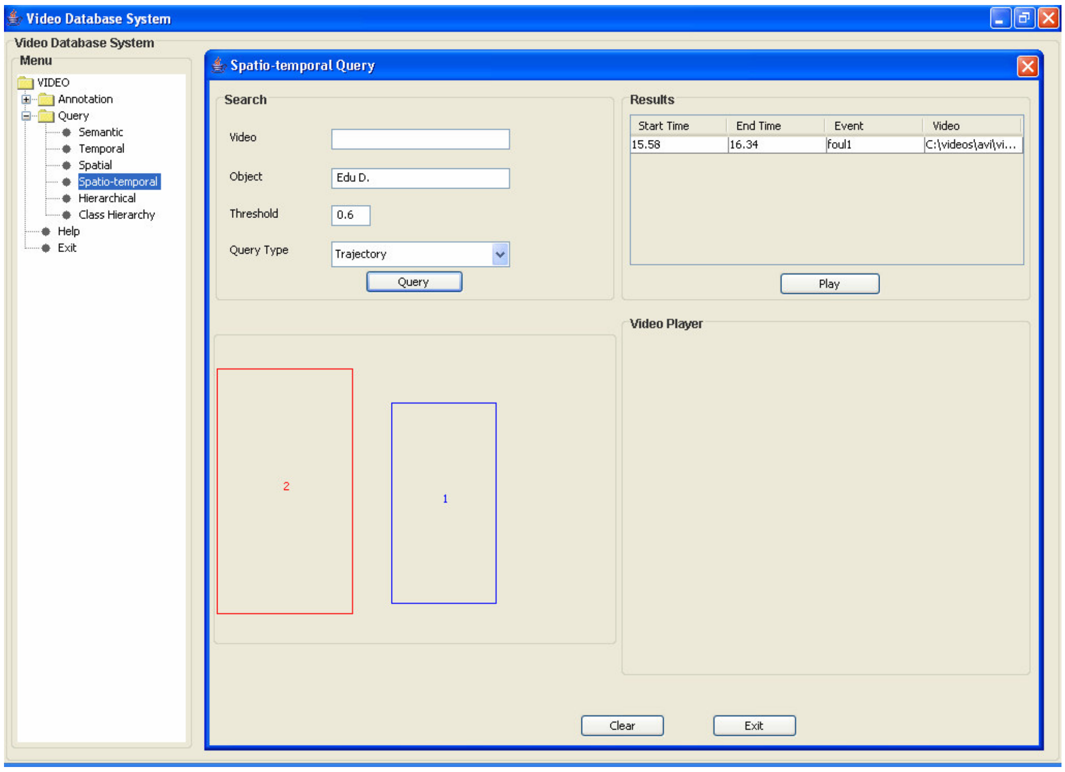
When the user enters the needed information and presses the *Query* button, the constructed regional query is executed and the results are displayed in the *Results* panel.

The following algorithm, similar to the one utilized in [3], is evaluated for regional queries:

```
1-  The query is firstly sent to bridge and bridge sends the query to FOOD.
2-  FOOD finds the actors according to the specified Video name and Object names.
3-  For each found actor
    3.1-  For each position of that actor
        3.1.1-  Get the region of the position
        3.1.2-  Find the membership degree of the region to the specified region by
                using the region matching formula (F-1).
        3.1.3-  If the membership degree is greater than the specified threshold, add
                that position with the found membership degree to result list.
4-  Send the result list to the user interface.
```

In the result list, the time intervals in which the regional query holds are displayed with their membership degrees. When a record is selected from the result list and *Play* button is pressed, the selected time interval is played in the Video Player.

When *Trajectory* is selected from the *Query Type,* the interface takes the Trajectory Query mode. The interface for trajectory query is shown in Figure 5.28 and the result of the query is represented in Figure 5.29. All the search criteria are optional to perform a trajectory query.



Video:

Object: Edu D.

Threshold: 0.6

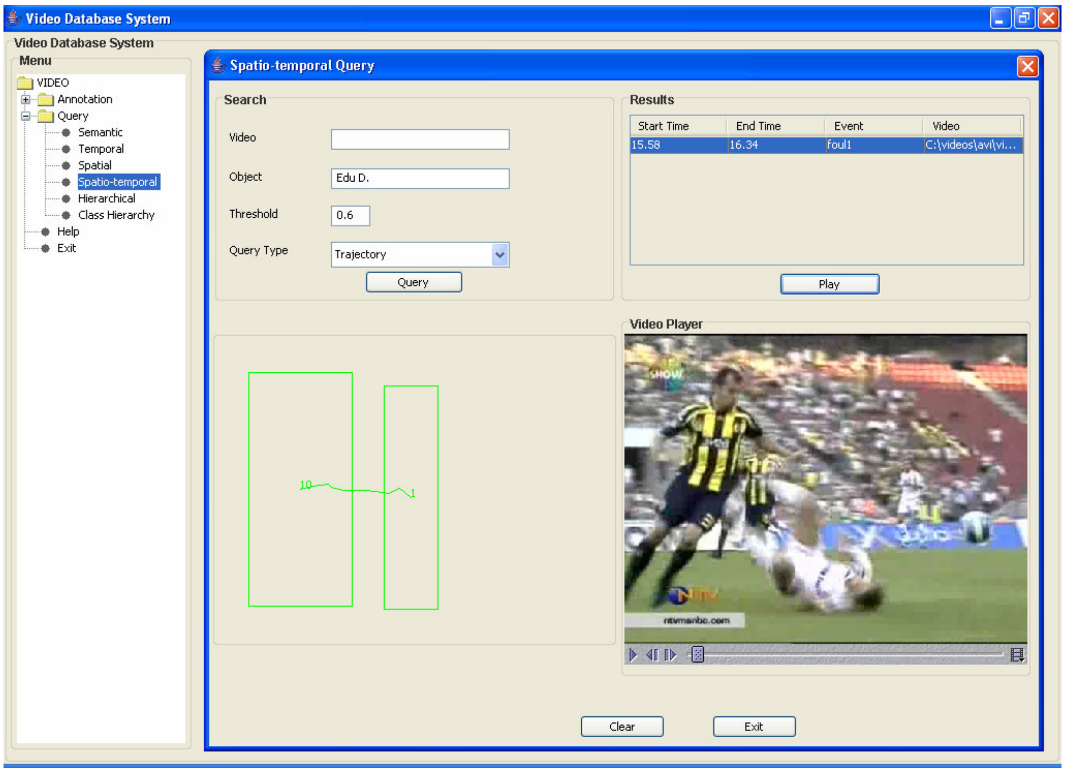
Query Type: Trajectory

Start Time	End Time	Event	Video
15.58	16.34	foul1	C:\videos\avi\vi...

Figure 5.28: Spatio-temporal Query Interface (Trajectory Query)

If *Video* and *Object* fields are not left empty, the query is executed for the videos and video objects with the specified names. If those fields are left empty, the query is executed for all of the videos and all of the objects in those videos.

This interface can be used to find the time intervals in which a specified trajectory holds for the specified objects in the given video. A trajectory is specified by drawing starting and ending regions, which can be drawn in the panel on the left-bottom of the window.



Video:

Object: Edu D.

Threshold: 0.6

Query Type: Trajectory

Start Time	End Time	Event	Video
15.58	16.34	foul1	C:\videos\avi\vi...

Figure 5.29: Spatio-temporal Query Interface (Trajectory Query Results)

Since exact regions can not be specified, a threshold can be given which is used to eliminate the found regions which match with the specified regions with a membership degree smaller than the threshold. The region matching formula (F-1) is used to calculate the membership degree.

When the user enters the needed information and presses the *Query* button, the constructed trajectory query is executed and the results are displayed in the *Results* panel.

The following algorithm, similar to the one utilized in [3], is evaluated for trajectory queries:

1. The query is firstly sent to bridge and bridge sends the query to FOOD.

2. FOOD finds the actors according to the specified Video name and Object names.

3. For each found actor

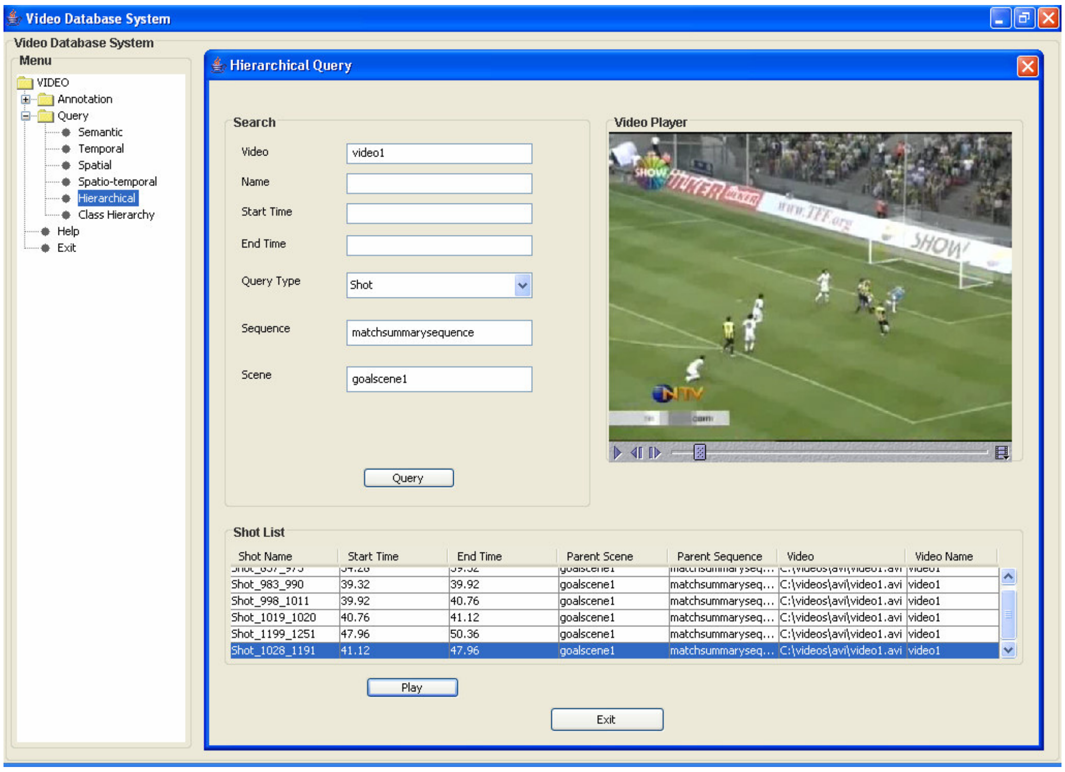
   3.1. Create a new trajectory list.

```
        3.2. For each position of that actor
            3.2.1. Get the region of the position
            3.2.2. If the trajectory list is empty
                3.2.2.1. Find the membership degree of the region to the specified
                         starting region by using the region matching formula (F-1).
                3.2.2.2. If the membership degree is greater than the specified
                         threshold, add that position to the trajectory list.
            3.2.3. Else (trajectory list is not empty)
                3.2.3.1. Find the membership degree of the region to the specified
                         ending region by using the region matching formula (F-1).
                3.2.3.2. If the membership degree is greater than the specified
                         threshold, add that position to the trajectory list. Add the
                         trajectory list to the result list. Create a new trajectory list.
                3.2.3.3. Else (If the membership degree is smaller than the specified
                         threshold), add that position to the trajectory list.
    4. Send the result list to the user interface.
```

In the result list, the time intervals in which the trajectory query holds are displayed with the event names. When a record is selected from the result list and *Play* button is pressed, the selected time interval is played in the Video Player. In addition, the exact starting and ending regions of the selected trajectory and the path between these two regions are drawn on the drawing panel as shown in Figure 5.29.

5.4.5 Hierarchical Queries

Hierarchical Query Interface allows the users to construct hierarchical queries. The Hierarchical Query Interface is shown in Figure 5.30 and Figure 5.31. Hierarchical Query Interface provides querying of the hierarchical structure of videos, such as finding sequences of a video, scenes of a specific sequence or video, shots of a specific scene, sequence or video or events of a specific shot, scene, sequence or video. At the same time; name, start time and end time properties of a shot, scene, sequence or event might be specified to narrow the search.

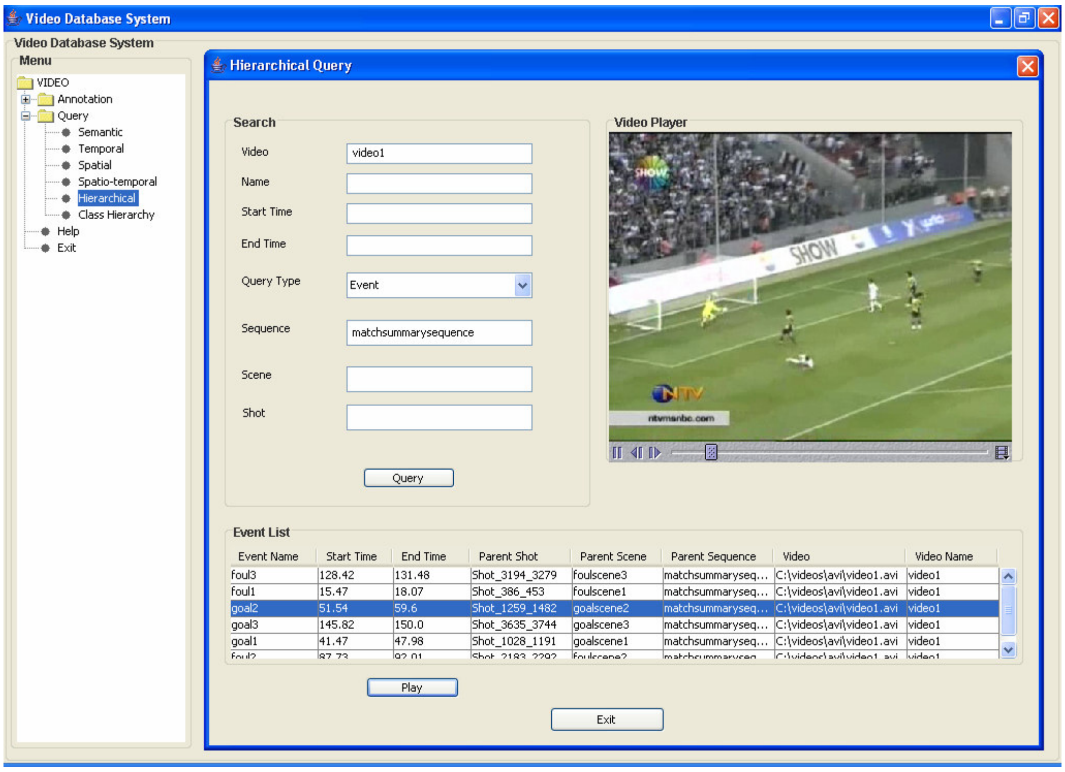


Video: video1
Name:
Start Time:
End Time:
Query Type: Shot
Sequence: matchsummarysequence
Scene: goalscene1

Shot Name	Start Time	End Time	Parent Scene	Parent Sequence	Video	Video Name
[illegible]	[illegible]	[illegible]	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
Shot_983_990	39.32	39.92	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
Shot_998_1011	39.92	40.76	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
Shot_1019_1020	40.76	41.12	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
Shot_1199_1251	47.96	50.36	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
Shot_1028_1191	41.12	47.96	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1

Figure 5.30: Hierarchical Query Interface (Shot is selected)

All the search criteria are optional to perform a hierarchical query. If *Video* field is not left empty, the query is executed for all the videos with the specified name. If that field is left empty, the query is executed for all of the videos. *Name* field is for the name of the sequence, scene, shot or event. The fields *Start Time* and *End Time* are for specifying the time interval of the queried sequence, scene, shot or event. The *Query Type* field includes the options: Sequence, Scene, Shot and Event. The query is performed only for the selected query type. If Sequence is selected, sequences with the specified properties are searched, if scene is selected, scenes with the specified properties are searched. The fields below the *Query Type* field (which are for entering parent *Sequence*, *Scene* and *Shot* names) are visible or invisible according to the selected query type. If Sequence is selected, none of them are visible. If Scene is selected, only *Sequence* field is visible. If Shot is selected, both of the *Sequence* and *Scene* fields are visible. If Event is selected, all of the fields are visible.



Video: video1
Name:
Start Time:
End Time:
Query Type: Event
Sequence: matchsummarysequence
Scene:
Shot:

Event List

Event Name	Start Time	End Time	Parent Shot	Parent Scene	Parent Sequence	Video	Video Name
foul3	128.42	131.48	Shot_3194_3279	foulscene3	matchsummaryseq...	C:\videos\avi\video1.avi	video1
foul1	15.47	18.07	Shot_386_453	foulscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
goal2	51.54	59.6	Shot_1259_1482	goalscene2	matchsummaryseq...	C:\videos\avi\video1.avi	video1
goal3	145.82	150.0	Shot_3635_3744	goalscene3	matchsummaryseq...	C:\videos\avi\video1.avi	video1
goal1	41.47	47.98	Shot_1028_1191	goalscene1	matchsummaryseq...	C:\videos\avi\video1.avi	video1
foul2	87.73	92.01	Shot_2183_2292	foulscene2	matchsummaryseq	C:\videos\avi\video1.avi	video1

Figure 5.31: Hierarchical Query Interface (Event is selected)

The following algorithm is evaluated for hierarchical queries:

1. The query is firstly sent to bridge and bridge sends the query to FOOD.

2. If the selected query type is Sequence

   2.1. Find the sequences with the specified properties (video, name, start time and end time).

   2.2. Result list is the list of found sequences.

3. If the selected query type is Scene

   3.1. The field for entering the parent sequence name becomes visible.

3.2. Find the scenes with the specified properties (video, name, start time, end time and parent sequence name).

3.3. Result list is the list of found scenes.

4. If the selected query type is Shot

4.1. The fields for entering the parent sequence name and parent scene name become visible.

4.2. Find the shots with the specified properties (video, name, start time, end time, parent sequence name and parent scene name).

4.3. Result list is the list of found shots.

5. If the selected query type is Event

5.1. The fields for entering the parent sequence name, parent scene name and parent shot name become visible.

5.2. Find the events with the specified properties (video, name, start time, end time, parent sequence name, parent scene name and parent shot name).

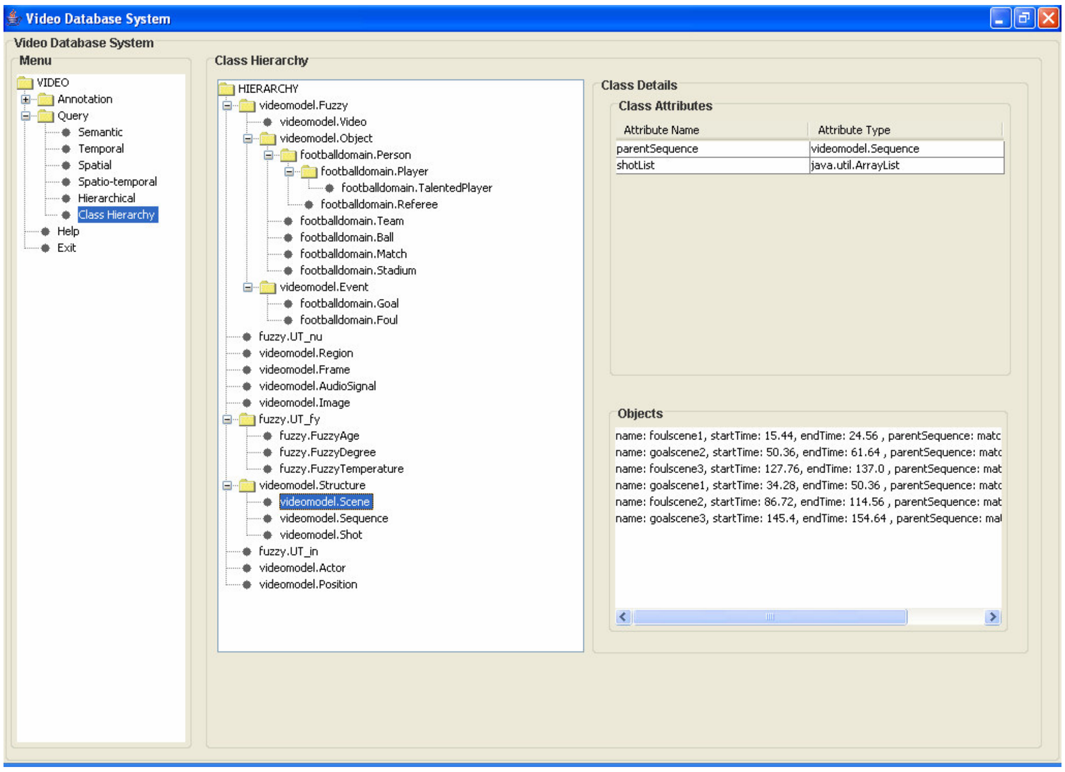
5.3. Result list is the list of found events.

6. Send the result list to the user interface.

In the result list, the found instances (sequence list, scene list, shot list or event list depending on the selected query type) are displayed with the start time and end time properties. When a record is selected from the result list and *Play* button is pressed, the selected time interval is played in Video Player as shown in Figure 5.30 and Figure 5.31.

5.4.6 Class Hierarchy

This interface can be used to display the content of the object-oriented database. In the tree menu on the left-hand side of the window, the class hierarchy is represented. When an object is stored in db4o, its class definition is automatically stored in the database. The db40 reflection API is used to get the class and class hierarchy information from the database. The Class Hierarchy Interface is shown in Figure 5.32.



Attribute Name	Attribute Type
parentSequence	videomodel.Sequence
shotList	java.util.ArrayList

```
name: foulscene1, startTime: 15.44, endTime: 24.56 , parentSequence: matc
name: goalscene2, startTime: 50.36, endTime: 61.64 , parentSequence: matc
name: foulscene3, startTime: 127.76, endTime: 137.0 , parentSequence: mat
name: goalscene1, startTime: 34.28, endTime: 50.36 , parentSequence: matc
name: foulscene2, startTime: 86.72, endTime: 114.56 , parentSequence: mat
name: goalscene3, startTime: 145.4, endTime: 154.64 , parentSequence: mal
```

Figure 5.32: Class Hierarchy Interface

When a class is selected in the tree, the attribute information of the selected class is displayed in the *Class Attributes* panel on the right-hand side of the window. The instances of the selected class are also displayed in the *Objects* panel on the right-hand side of the window. The instances are displayed with their attribute values.

CHAPTER 6

CONCLUSIONS

In this study, we firstly introduce a fuzzy conceptual data model for video databases. The model supports uncertainty, which might naturally occur in various applications of video data. The UML model is utilized to represent our conceptual model. We extend UML model to represent uncertain information and video specific properties. Our semantic model does not only support hierarchical structure of video data but also handles spatial and temporal relations between semantic entities.

Secondly, we present an intelligent fuzzy object-oriented database framework for video database applications. The presented framework is the adaptation of IFOOD architecture and provides an integration of a fuzzy object-oriented database and a fuzzy knowledge base. The framework allows modeling complex and rich semantic content of video data by incorporating uncertain information in the fuzzy object-oriented database. Our conceptual model is mapped to logical FOOD model and used in the fuzzy object-oriented database. To represent knowledge about video data, spatial, temporal and fuzzy semantic rules are defined, and these rules are processed by the fuzzy knowledge base. Fuzzy object-oriented database and fuzzy knowledge base are integrated by a bridge providing powerful and intelligent video retrieval.

We have also developed a prototype system of the presented framework, which provides the user annotation and querying of video data. The framework supports a diversity of queries related to video data such as (fuzzy) semantic, temporal, spatial, hierarchical, regional and trajectory queries. Our framework may easily be applied for any knowledge intensive video database application involving uncertain information.

The presented framework is developed for video databases. A future work is to extend the framework in order to handle other multimedia data such as image, audio and text. Our generic conceptual model can be extended for this purpose.

Another future direction is to incorporate an automatic extraction algorithm to the presented framework. It is obvious that manually annotating a video is very difficult, time consuming and impractical. Although, we utilize IBM MPEG-7 Annotation Tool, which provides automatic shot annotation, all the other semantic information is obtained manually. It would be very helpful to use an algorithm that automatically detects objects, positions of the objects and even events happening in a time interval. Automatic extraction is a very open research area since extracting high-level semantic information (such as to extract the events happening, the meaning of the sequences and scenes) demands challenging work.

Considering the automatic extraction algorithms, in the future, the knowledge base part of our framework may be used to process rules for automatic extraction of event and object entities in video data, which can lead to development of more intelligent video database applications.

The number of the rules stored in the knowledge base takes an important role for the performance of the framework. Since there might be a huge number of videos in a video database, scalability becomes an important problem. Research efforts on improving the performance of the framework would be needed in the future, such as developing special index structures, trying to use a minimum number of rules, etc.

We use JMF API to provide playing and controlling videos in our user interface. Since JMF API does not support every video format or every video codec properly, it brings some problems. A future work would be needed to overcome these video format issues.

To represent how our model and framework can be applied to a video database application, we prefer to use football game videos. Although it is a very popular domain among the studies on video databases, it does not involve sufficient uncertain information. Choosing a different knowledge intensive domain such as news or medical videos, which might naturally bring much more uncertainty, would be more illustrative to show the usability and applicability of our video data model and framework. As a future direction, our conceptual model and framework can be extended for another domain. Since we designed them to be generic, we think that it is easy to apply the model and the framework for any application domain.

REFERENCES

[1] Ma, Z. M., “Fuzzy Information Modeling with the UML”, In: Advances in Fuzzy Object-Oriented Databases: Modeling and Applications, IDEA Group Publishing, 2004.

[2] Arslan U., “A Semantic Data Model and Query Language for Video Databases”, M. Sc. Thesis, Department of Computer Engineering, Bilkent University, Ankara, Turkey, January 2002.

[3] Durak N., “Semantic Video Modeling and Retrieval with Visual, Auditory, Textual Sources”, M. Sc.Thesis, Department of Computer Engineering, METU, Ankara, Turkey, September 2004.

[4] Oztarak H., A. Yazici, "Structural and Event Based Multimodal Video Data Modeling", ADVIS 2006, LNCS 4243, pp. 264-273, 2006.

[5] Yazici A., and A. Cinar, “Conceptual Modeling for the Design of Fuzzy OO Databases”, Knowledge Management in Fuzzy Databases, Edited by O. Pons, A. Vila and J. Kacprzyk, Physica-Verlag, Heidelberg and New York, Vol. 39, pp. 12-35, 2000.

[6] Ekin A., A. M. Tekalp, and R. Mehrotra, “Integrated Semantic–Syntactic Video Modeling for Search and Browsing”, IEEE Transactions on Multimedia, Vol. 6, No. 6, December 2004.

[7] Hjelsvold R. and R. Midtstraum, “Modelling and Querying Video Data”, Proceedings of the 20th International Conference on Very Large Databases, pp. 686-694, 1994.

[8] Choi Y.I., Y.M. Park, H.S. Lee, and S.I. Jin, “An Integrated Data Model and a Query Language for Content-Based Retrieval of Video”, Proceedings of the 4th International Workshop on Advances in Multimedia Information Systems, pp. 192-198, September 1998.

[9] Yazici A., Q. Zhu, and N. Sun, “Semantic Data Modeling of Spatiotemporal Database Applications”, International Journal of Intelligent Systems, Vol. 16, No. 7, pp. 881-904, 2001.

[10] Aygun R.S., A. Yazici, and N. Arica, “Conceptual Data Modeling of Multimedia Database Applications”, “Proceedings of the 4th International Workshop on Multimedia Database Management Systems, pp. 182-189, 1998.

[11] Aygun R. S., and A. Yazici, “Modeling and Management of Fuzzy Information in Multimedia Database Applications”, Multimedia Tools and Applications, Vol. 24, No. 1, pp. 29-56, 2004.

[12] Nepal S., M.V. Ramakrishna, and J.A. Thom, “A Fuzzy Object Query Language (FOQL) for Image Database”, Proceedings of Sixth International Conference on Database Systems for Advanced Applications, pp. 117-124, 1999.

[13] Köprülü M., N.K. Cicekli, A. Yazici, “Spatio-temporal Querying in Video Databases”, Information Sciences 160(1-4), pp. 131-152, 2004.

[14] Yazici A., R. George, and D. Aksoy, “Design and Implementation Issues in the Fuzzy Object-Oriented Data Model”, Information Sciences (Int. Journal), Vol. 108, No. 1-4, pp. 241-260, July 1998.

[15] Yazici A, and A. Cinar, “Conceptual Design of Fuzzy Object-Oriented Databases”, Proceedings of the 1998 Second International Conference on Knowledge-Based Intelligent Electronic Systems, Vol. 2, pp. 299-305, 1998.

[16] Sicilia, M.A., E. Garcia, and J.A. Gutierrez, “Introducing Fuzziness in Existing Orthogonal Persistence Interfaces and Systems”, In: Advances in Fuzzy Object-Oriented Databases: Modeling and Applications, IDEA Group Publishing, 2004.

[17] Tusch R., H. Kosch, and L. Böszörményi, “VIDEX: An Integrated Generic Video Indexing Approach”, Proceedings of the 8th ACM Multimedia Conference, Los Angeles (USA), ACM Press, pp. 448-451, October-November 2000.

[18] Kosch H., L. Böszörményi, A. Bachlechner, B. Dörflinger, C. Hanin, C. Hofbauer, M. Lang, C. Riedler, and R. Tusch, “SMOOTH - A Distributed Multimedia Database System”, VLDB'2001 in Rome (Italy), pp. 713-714, 2001.

[19] Kosch H., L. Böszörményi, R. Tusch, A. Bachlechner, B. Dörflinger, C. Hofbauer, and C. Riedler, “The SMOOTH Video DB - Demonstration of an Integrated Generic Indexing Approach”, Proceedings of the 8th ACM Multimedia Conference, Los Angeles (USA), ACM Press, pp. 495-496, October-November 2000.

[20] Böszörményi L., H. Hellwagner, and H. Kosch. “Multimedia Technologies for E-Business Systems and Processes”, Proceedings of Elektronische Geschaftsprozesse (E-

Business Processes), pp. 471-481, Klagenfurt, Austria, IT Verlag für Informationtechnik, September 2001.

[21] Mostefaoui A, L. Favory, and L. Brunie, "SIRSALE: A Large Scale Video Indexing and Content-Based Retrieving System", ACM Multimedia Conference, Juan les pins, France, pp. 251–254, December 2002.

[22] Mostefaoui, A, "A Modular and Adaptive Framework for Large Scale Video Indexing and Content-Based Retrieval: the SIRSALE System", Software – Practice & Experience, Vol. 36, No. 8, pp. 871-890, July 2006.

[23] Allen, J. F., "Maintaining Knowledge About Temporal Intervals", Communications of the ACM 26 (11), pp. 832-843, 1983.

[24] ISO/IEC Committee Draft 15 938-5 Information Technology-Multimedia Content Description Interface: Multimedia Description Schemes, ISO/IEC/JTC1/SC29/WG11/N3966, March 2001.

[25] Rumbaugh J., M. Blaha, et al., Object-Oriented Modeling and Design, Prentice-hall, NJ, 1991.

[26] Booch, G., J. Rumbaugh and I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, Reading MA, 1999.

[27] Bordogna G., G. Pasi, and D. Lucarella, "A Fuzzy Object-Oriented Data Model for Managing Vague and Uncertain Information", International Journal of Intelligent Systems, Vol. 14, pp. 623–651, 1999.

[28] Ma Z. M., W.J. Zhang, W.Y. Ma, and Q. Chen, "Conceptual Design of Fuzzy Object-Oriented Databases Using Extended Entity-Relationship Model", International Journal of Intelligent Systems, Vol. 16, pp. 697-711, 2001.

[29] Yazici, A., and R. George, Fuzzy Database Modeling, Physica-Verlag, NY, 1999.

[30] Koyuncu, M. and A. Yazici, "IFOOD:An Intelligent Fuzzy Object-Oriented Database Architecture", IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 5, pp. 1137-1154, 2003.

[31] Koyuncu, M., and A. Yazici, "A Fuzzy Knowledge-Based System for Intelligent Retrieval", IEEE Transactions on Fuzzy Systems, Vol. 13, pp. 317-330, 2005.

[32] Donderler, M.E., O. Ulusoy, and U. Gudukbay, “Rule-Based Spatiotemporal Query Processing for Video Databases”, The VLDB Journal, Vol. 13, No. 1, pp. 86-103, 2004.

[33] Li, J.Z., M.T. Ozsu, and D. Szafron, “Spatial Reasoning Rules in Multimedia Management Systems”, Proceedings of International Conference on Multimedia Modeling, pp. 119-133, Toulouse, France, 1996.

[34] Donderler, M.E., O. Ulusoy, and U. Gudukbay, “A Rule-Based Video Database System Architecture”, Information Sciences, Vol. 143, No. 1-4, pp. 13-45, 2002.

[35] Petkovic, M. and W. Jonker, “A Framework for Video Modelling”, Eighteenth IASTED International Conference Applied Informatics, Innsbruck, Austria, 2000.

[36] db4o, Open Source Object Database, http://www.db4o.com, Last date accessed: September, 2007.

[37] Jess, the Rule Engine for the Java Platform, http://herzberg.ca.sandia.gov/jess, Last date accessed: September, 2007.

[38] IBM MPEG-7 Annotation Tool, http://www.alphaworks.ibm.com/tech/videoannex, Last date accessed: September, 2007.

[39] George R., F.E. Petry, and B.P. Buckles, “Modeling Class Hierarchies in the Fuzzy Object Oriented Data Model”, Fuzzy Sets and Systems Vol. 60, No.3, pp. 259-272, 1993.

[40] Friedmann-Hill, Ernest, Jess In Action: Java Rule-Based Systems, Manning Publications, 2003.

[41] Xerces2 Java Parser, http://xerces.apache.org/xerces2-j/, Last date accessed: September, 2007.

[42] CLIPS (C Language Integrated Production System) Expert System Shell, http://www.ghg.net/clips/CLIPS.html, Last date accessed: September, 2007.

[43] FuzzyCLIPS, http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyClips/fuzzyCLIPSIndex2.html, Last date accessed: September, 2007.

[44] NRC FuzzyJ Toolkit & FuzzyJess,
http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html,
Last date accessed: September, 2007.

[45] Java Media Framework API (JMF), http://java.sun.com/products/java-media/jmf/, Last date accessed: September, 2007.

[46] Oztarak H., "Structural and Event Based Multimodal Video Data Modeling", M. Sc.Thesis, Department of Computer Engineering, METU, Ankara, Turkey, December 2005.

[47] Adali S., K.S. Candan, S.S. Chen, K. Erol, and V.S. Subrahmanian, "The Advanced Video Information System: Data Structures and Query Processing", Multimedia Systems, Vol. 4, pp. 172-186, 1996.

[48] Hacid M.S., C. Decleir and J. Kouloumdjian, "A Database Approach for Modeling and Querying Video Data", IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 5, pp. 729-750, 2000.

[49] Donderler M.E., E. Şaykol, U. Arslan, and O. Ulusoy, "BilVideo: Design and Implementation of a Video Database Management System", Multimedia Tools and Applications, Vol. 27, No. 1, pp. 79-104, September 2005.

[50] Oomoto E. and K. Tanaka, "OVID: Design and Implementation of a Video-Object Database System", IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 4, pp. 629-643, August 1993.

[51] Wang Y., C. Xing, and L. Zhou, "Video Semantic Models : Survey and Evaluation", International Journal of Computer Science and Network Security, Vol. 6, No. 2, pp. 10-20, 2006.

[52] Li J.Z., M.T. Özsu, and D. Szafron, "Modeling of Moving Objects in a Video Database", Proceedings of IEEE International Conference on Multimedia Computing and Systems, Ottawa, Canada, pp. 336-343, 1997.

[53] Van Gyseghem, N., R. De Caluwe, and R. Vandenberghe, "UFO:Uncertainty and Fuzziness in an Object-Oriented Model", Second IEEE International Conference on Fuzzy Systems, Vol. 2, pp. 773-778, 1993.

[54] Yazici A. and M. Koyuncu, "Fuzzy Object-Oriented Database Modeling Coupled with Fuzzy Logic", Fuzzy Sets and Systems, Vol. 89, No. 1, pp. 1-26, July 1997.

[55] Koyuncu M., A. Yazici, and R. George, “Flexible Querying in an Intelligent Object-Oriented Database Environment”, Proceedings of the Fourth International Conference on Flexible Query Answering Systems, Warsaw, Poland, pp. 75–84, October 2000.

APPENDIX A

EXAMPLE JAVA CLASS DEFINITIONS

```
public class UT_fy
{
      private HashMap fuzzyValue;

      private Object crispValue;
}

public class UT_in
{
      private Float value1;
      private Float value2;
}

public class UT_nu
{
      private String nullValue;
      private Object crispValue;
}

public class FuzzyDegree extends UT_fy
{
      private static HashMap domain;
      private static HashMap simMatrix;
      private static String semantics;
      private static String crispType;
      private static HashMap membershipFunction;

      public static class FuzzyTerms
      {
            public static String very_high = "very high";
            public static String high = "high";
            public static String medium = "medium";
            public static String low = "low";
            public static String very_low = "very low";
      }
      public static String getSemantics() {
            return semantics;
      }
      public static HashMap getSimMatrix() {
            return simMatrix;
      }
}

public abstract class Fuzzy
{
      private float objectMShip;
      private float classSClassMShip;

      public void setObjectMShip(float objectMShip) {
            this.objectMShip = objectMShip;
      }
      public void calcMShip(HashMap fuzzyAttributes) {
            this.setObjectMShip(FuzzyProcessor
                  .findObjectClassMembership(fuzzyAttributes));
      }
}

public class Video extends Fuzzy
{
      private String name;
      private String rawData;
      private String title;
      private String date;
      private UT_nu description;
      private ArrayList structureList;
      private ArrayList objectList;
}

public class Object extends Fuzzy
{
      private String objectName;
      private UT_nu description;
      private Region region;
      private Video video;
      private Frame frame;
}

public class Event extends Fuzzy
{
      private String name;
      private UT_fy when;
      private double startTime;
      private double endTime;
      private Shot parentShot;
      private Video video;
}

public class Actor
{
      private String lingRole;
      private String semRole;
      private Event event;
      private Object videoObject;
      private ArrayList positionList;
}

public class Position
{
      private double startTime;
      private double endTime;
      private Region region;
      private Actor actor;
}
```

```
public class Foul extends Event
{
      private FuzzyDegree harshness;
      private boolean redCard;
      private boolean yellowCard;
      private String fieldPosition;
      private static HashMap rangeHarshness;
      private static float relevanceHarshness;

      public static HashMap getRangeHarshness() {
            return rangeHarshness;
      }

      public static float getRelevanceHarshness() {
            return relevanceHarshness;
      }

      public static void setRanges()
      {
            rangeHarshness = new HashMap();
            rangeHarshness.put(new Integer(0),
                 FuzzyDegree.FuzzyTerms.very_high);
            rangeHarshness.put(new Integer(1),
                 FuzzyDegree.FuzzyTerms.high);
            rangeHarshness.put(new Integer(2),
                 FuzzyDegree.FuzzyTerms.medium);
      }

      public static void setRelevances() {
            relevanceHarshness = new Float(0.8).floatValue();
      }

      public void calcMShip()
      {
            UncertainAttribute atr = new UncertainAttribute();
            atr.setAttributeRange(getRangeHarshness());
            atr.setUtFyValue(harshness);
            atr.setRelevance(getRelevanceHarshness());
            atr.setSemantics(FuzzyDegree.getSemantics());
            atr.setSimMatrix(FuzzyDegree.getSimMatrix());
            HashMap hashMap = new HashMap();
            hashMap.put(new Integer(0), atr);
            super.calcMShip(hashMap);
      }
}

public class TalentedPlayer extends Player
{
      private FuzzyDegree shotAccuracy;
      private FuzzyDegree speed;
      private FuzzyDegree ballControl;
      private FuzzyDegree talent;
      private static HashMap rangeShotAccuracy;
      private static float relevanceShotAccuracy;
      private static HashMap rangeSpeed;
      private static float relevanceSpeed;
      private static HashMap rangeBallControl;
      private static float relevanceBallControl;

      public static HashMap getRangeShotAccuracy() {
            return rangeShotAccuracy;
      }

      public static float getRelevanceShotAccuracy() {
            return relevanceShotAccuracy;
      }

      public static HashMap getRangeSpeed() {
            return rangeSpeed;
      }

      public static float getRelevanceSpeed() {
            return relevanceSpeed;
      }

      public static HashMap getRangeBallControl() {
            return rangeBallControl;
      }

      public static float getRelevanceBallControl() {
            return relevanceBallControl;
      }

      public static void setRanges()
      {
            rangeShotAccuracy = new HashMap();
            rangeShotAccuracy.put(new Integer(0),
                 FuzzyDegree.FuzzyTerms.high);
            rangeShotAccuracy.put(new Integer(1),
                 FuzzyDegree.FuzzyTerms.very_high);

            rangeSpeed = new HashMap();
            rangeSpeed.put(new Integer(0),
                 FuzzyDegree.FuzzyTerms.high);
            rangeSpeed.put(new Integer(1),
                 FuzzyDegree.FuzzyTerms.very_high);

            rangeBallControl = new HashMap();
            rangeBallControl.put(new Integer(0),
                 FuzzyDegree.FuzzyTerms.high);
            rangeBallControl.put(new Integer(1),
                 FuzzyDegree.FuzzyTerms.very_high);
      }

      public static void setRelevances()
      {
            relevanceShotAccuracy = new Float(0.8).floatValue();
            relevanceSpeed = new Float(0.7).floatValue();
            relevanceBallControl = new Float(0.8).floatValue();
      }

      public void calcMShip()
      {
            HashMap hashMap = new HashMap();
            UncertainAttribute atr = new UncertainAttribute();
```

```
            atr.setAttributeRange(getRangeShotAccuracy());
            atr.setUtFyValue(shotAccuracy);
            atr.setRelevance(getRelevanceShotAccuracy());
            atr.setSemantics(FuzzyDegree.getSemantics());
            atr.setSimMatrix(FuzzyDegree.getSimMatrix());
            hashMap.put(new Integer(0), atr);
            atr = new UncertainAttribute();
            atr.setAttributeRange(getRangeSpeed());
            atr.setUtFyValue(speed);
            atr.setRelevance(getRelevanceSpeed());
            atr.setSemantics(FuzzyDegree.getSemantics());
            atr.setSimMatrix(FuzzyDegree.getSimMatrix());
            hashMap.put(new Integer(1), atr);
            atr = new UncertainAttribute();
            atr.setAttributeRange(getRangeBallControl());
            atr.setUtFyValue(ballControl);
            atr.setRelevance(getRelevanceBallControl());
            atr.setSemantics(FuzzyDegree.getSemantics());
            atr.setSimMatrix(FuzzyDegree.getSimMatrix());
            hashMap.put(new Integer(2), atr);
            super.calcMShip(hashMap);
      }
}
```

APPENDIX B

EXAMPLE RULES IN JESS LANGUAGE

Temporal Rules

```
(import videomodel.*)

(deftemplate Event (declare (from-class Event)))
(deftemplate EventToEventRelation (declare (from-class
                                          EventToEventRelation)))

(defrule before
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) (< ?e1  ?s2)
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "before")))

(defrule meets
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) (= ?e1  ?s2)
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "meets")))

(defrule equal
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) ( and (= ?s1  ?s2)(= ?e1  ?e2))
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "equal")))

(defrule during
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) ( and (> ?s1  ?s2)(< ?e1  ?e2) )
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "during")))

(defrule starts
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) ( and (= ?s1  ?s2)(< ?e1  ?e2) )
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "starts")))

(defrule finishes
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) ( and (> ?s1  ?s2)(= ?e1  ?e2) )
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "finishes")))

(defrule overlaps
?p1 <- (Event (startTime ?s1) (endTime ?e1) (OBJECT ?event1))
?p2 <- (Event (startTime ?s2) (endTime ?e2) (OBJECT ?event2))
(test (and (neq ?p1 ?p2) ( and (< ?s1  ?s2) (> ?e1 ?s2) (< ?e1
?e2) )
           (eq (get ?event1 video) (get ?event2 video))))
    => (add (new EventToEventRelation ?event1 ?event2 "overlaps")))

(defrule beforeInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "before"))
    => (add (new EventToEventRelation ?e2 ?e1 "beforeInverse")))

(defrule meetsInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "meets"))
    => (add (new EventToEventRelation ?e2 ?e1 "meetsInverse")))

(defrule equalInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "equal"))
    => (add (new EventToEventRelation ?e2 ?e1 "equalInverse")))

(defrule duringInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "during"))
    => (add (new EventToEventRelation ?e2 ?e1 "duringInverse")))

(defrule startsInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "starts"))
    => (add (new EventToEventRelation ?e2 ?e1 "startsInverse")))

(defrule finishesInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "finishes"))
    => (add (new EventToEventRelation ?e2 ?e1 "finishesInverse")))

(defrule overlapsInverse
?p1 <- (EventToEventRelation (event1 ?e1) (event2 ?e2)
(relationName "overlaps"))
    => (add (new EventToEventRelation ?e2 ?e1 "overlapsInverse")))
```

Spatial Rules

```
(import bridge.*)
(import videomodel.*)

(deftemplate PositionFact (declare (from-class PositionFact)))
(deftemplate PositionToPositionRelation (declare (from-class
                                   PositionToPositionRelation)))
(deftemplate ActorToActorRelation (declare (from-class
                                  ActorToActorRelation)))

(defglobal ?*fp* = (new FuzzyProcessor))


(defrule during
?p1 <- ( PositionFact   (startTime ?start1) (endTime ?end1)
                        (startX ?sx1) (endX ?ex1)
                        (startY ?sy1) (endY ?ey1) (actor ?a1))
?p2 <- ( PositionFact   (startTime ?start2) (endTime ?end2)
                        (startX ?sx2) (endX ?ex2)
                        (startY ?sy2) (endY ?ey2) (actor ?a2))
(test (and (and (neq ?p1 ?p2)
                (eq (get ?a1 event) (get ?a2 event)) (neq ?a1 ?a2))
                (and (> ?start1  ?start2)(< ?end1  ?end2))))
      =>
            (bind ?startTimeResult
                  (intersectionStart ?start1 ?start2 "during"))
            (bind ?endTimeResult
                  (intersectionEnd ?end1 ?end2 "during"))
            (add (new PositionToPositionRelation
                  ?startTimeResult ?endTimeResult
                  ?sx1 ?ex1 ?sy1 ?ey1 ?a1 ?sx2 ?ex2 ?sy2 ?ey2 ?a2)
))

(defrule left
?p1 <- (PositionToPositionRelation
            (startTimeResult ?startTimeResult)
            (endTimeResult ?endTimeResult)
            (startX1 ?sx1) (endX1 ?ex1)
            (startY1 ?sy1) (endY1 ?ey1) (actor1 ?a1)
            (startX2 ?sx2) (endX2 ?ex2)
            (startY2 ?sy2) (endY2 ?ey2) (actor2 ?a2))

(test (or (< ?ex1  ?sx2) (= ?ex1  ?sx2) ))

      =>
         (bind ?mShipDegree (call ?*fp* findSpatialMShipDegree
                                  ?sx1 ?ex1 ?sy1 ?ey1
                                  ?sx2 ?ex2 ?sy2 ?ey2 "left"))

        (add (new ActorToActorRelation
                        ?startTimeResult ?endTimeResult
                        "left" ?a1 ?a2 ?mShipDegree) ))

(defrule right
?p1 <- (ActorToActorRelation
            (startTimeResult ?startTimeResult)
            (endTimeResult ?endTimeResult)
            (relationName "left")
            (actor1 ?a1) (actor2 ?a2)
            (membershipDegree ?mShipDegree))

      => (add (new ActorToActorRelation
                        ?startTimeResult ?endTimeResult
                        "right" ?a2 ?a1 ?mShipDegree) ))
```

```
(defrule inside
?p1 <- (PositionToPositionRelation
            (startTimeResult ?startTimeResult)
            (endTimeResult ?endTimeResult)
            (startX1 ?sx1) (endX1 ?ex1)
            (startY1 ?sy1) (endY1 ?ey1) (actor1 ?a1)
            (startX2 ?sx2) (endX2 ?ex2)
            (startY2 ?sy2) (endY2 ?ey2) (actor2 ?a2))
(test ( and (> ?sx1  ?sx2)(< ?ex1  ?ex2)
            (> ?sy1  ?sy2)(< ?ey1  ?ey2) ))

      => (add (new ActorToActorRelation
                        ?startTimeResult ?endTimeResult
                        "inside" ?a1 ?a2) ))

(defrule contain
?p1 <- (ActorToActorRelation
            (startTimeResult ?startTimeResult)
            (endTimeResult ?endTimeResult)
            (relationName "inside")
            (actor1 ?a1) (actor2 ?a2))

      => (add (new ActorToActorRelation
                        ?startTimeResult ?endTimeResult
                        "contain" ?a2 ?a1) ))
```

Fuzzy Semantic Rules

```
(import footballdomain.*)
(import bridge.*)


(deftemplate TalentedPlayer (declare (from-class TalentedPlayer)))
(deftemplate Result (declare (from-class Result)))


(defglobal ?*fp* = (new FuzzyProcessor))


(defrule findTalent
?obj <- (TalentedPlayer
                        (shotAccuracy ?sa&:( and (neq ?sa nil)
                        (>= (call ?*fp* similarityMatch (create$
                                    "very high") ?sa) 0.8)))
                        (speed ?sp&:( and (neq ?sp nil)
                        (>= (call ?*fp* similarityMatch (create$
                                    "very high" "high") ?sp) 0.7)))
                         (ballControl ?bc&:( and (neq ?bc nil)
                        (>= (call ?*fp* similarityMatch (create$
                                    "very high" "high") ?bc) 0.8)))
                        (talent ?talent)
                        (OBJECT ?tp)
        )
      =>
         (bind ?result (call ?*fp* ruleResult
                              (create$ "very high")
                              (create$
                               (call ?*fp* similarityMatch (create$
                                        "very high") ?sa)
                               (call ?*fp* similarityMatch (create$
                                        "very high" "high") ?sp)
                               (call ?*fp* similarityMatch (create$
                                        "very high" "high") ?bc)
                              )
                              ?talent
                         ))

          (if (= ?result "true")  then (add (new Result ?tp)))
)
```

APPENDIX C

EXAMPLE OUTPUT OF IBM MPEG-7 ANNOTATION TOOL

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">
      <Video>
        <TemporalDecomposition>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:00:0F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">88</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:01:18F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:03:17F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">287</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:09:10F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:15:11F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">68</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:16:19F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:18:11F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">41</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:19:6F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:20:9F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">53</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:21:10F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:22:19F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">8</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:22:22F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:23:9F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">23</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:23:20F25</MediaTimePoint>
```

```
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:24:14F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">8</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:24:17F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
          <VideoSegment>
            <MediaTime>
              <MediaTimePoint>T00:00:25:4F25</MediaTimePoint>
              <MediaIncrDuration
                  mediaTimeUnit="PT1N25F">164</MediaIncrDuration>
            </MediaTime>
            <TemporalDecomposition>
              <VideoSegment>
                <MediaTime>
                  <MediaTimePoint>T00:00:28:10F25</MediaTimePoint>
                </MediaTime>
              </VideoSegment>
            </TemporalDecomposition>
          </VideoSegment>
        </TemporalDecomposition>
       </Video>
    </MultimediaContent>
  </Description>
</Mpeg7>
```