

# KNOWEL: A Hypermedia Knowledge Editor

M. Vazirgiannis, A. Kalousis, M. Hatzopoulos  
Department of Informatics,  
University Campus, University of Athens,  
15771, Ilisia, Athens, Greece  
e-mail: {michalis, grad0058, mike}@di.uoa.gr

## Abstract

*A popular approach for organising and accessing information is the hypermedia approach. One of the features that is weak in current hypermedia systems is the semantics of the nodes and links. In principle nodes are classified into categories while links represent abstract relationships among concepts. In this paper we present the implementation of parts of an object oriented data model that represents hypermedia information networks integrating semantics and multimedia information. The implementation refers to the parts of the data model that represent fuzzy knowledge in hypermedia networks. Special emphasis is put to the fuzzy features of the relationships among concepts.*

## 1. Introduction

A popular approach for organising and accessing information is the hypermedia approach. The information in such an approach uses nodes to represent the information itself and links for the interconnections among nodes. The links represent the semantic relationships among the nodes. One of the features that is weak in current hypermedia systems is the semantics of the nodes and links. Nodes are classified into concept categories while links represent abstract relationships among concepts

Current proposed standards (MHEG, HYTIME) refer to structural organisation of hypermedia documents without special reference to their semantics. In general there is a lack of semantic organisation of hypermedia systems, especially as regards the semantics of links and nodes. Moreover an issue is the representation and manipulation of the uncertainty of the relationships that are represented by the links.

Today it is widely accepted that an important problem in AI is the representation of common-sense knowledge. Conventional knowledge representation schemes (predicate calculus and

related methods) are not well suited for representing common-sense knowledge [10]. Moreover the conventional approaches to knowledge representation (semantic networks, frames, predicate calculus and Prolog) are based on bivalent logic. A serious shortcoming of such approaches is their inability to come to grips with the issue of uncertainty and imprecision. [11]. Fuzzy Logic provides an effective conceptual framework for dealing with the problem of knowledge representation in an environment of uncertainty and imprecision. Fuzzy logic is applied to various areas of Artificial Intelligence such as: natural language understanding [6] or abductive inference [2].

In this paper we present a knowledge editor that may be utilised for the task of semantics enforcement over a hypermedia network. The representation of concepts and relationships exploits fuzzy logic for modelling the uncertainty of a hypermedia network. KNOWEL is based on an Object Oriented Model [9] that represents in an integrated and uniform way all the information layers involved in a hypermedia information system (multimedia objects and presentations, nodes, links and the semantics that are embedded in the hypermedia network)

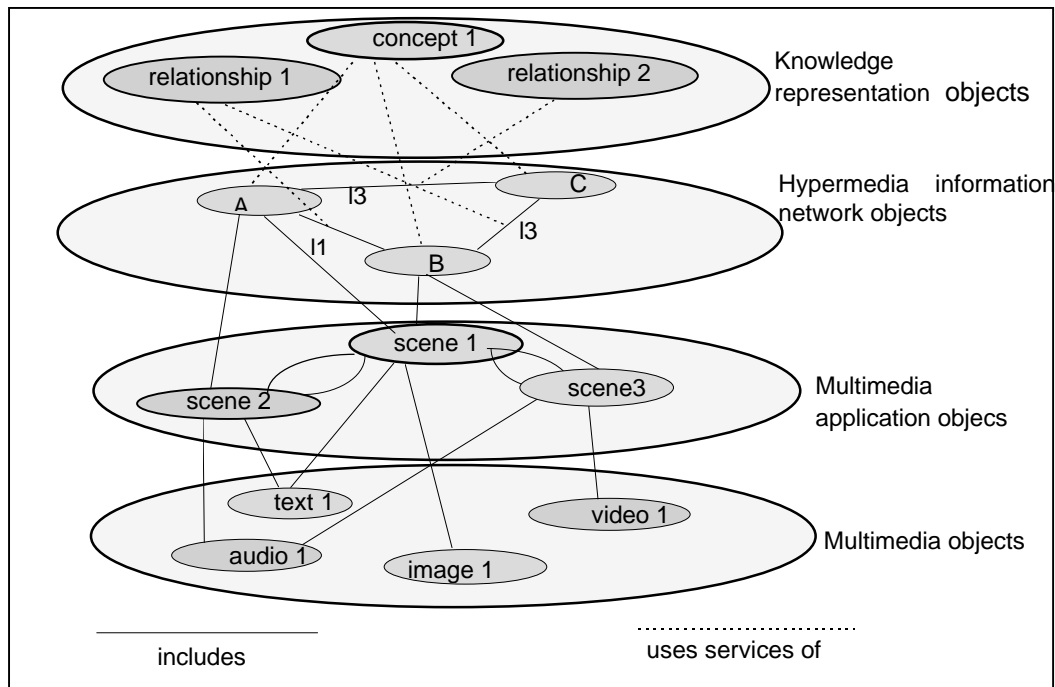
## 2. The data model

The data model [9] that served as the theoretical framework aims at representation of hypermedia information networks. More specifically the model represents the knowledge that is included in a hypermedia information network (in terms of concepts, relationships and weights), the hypermedia network itself (in terms of nodes and links), the multimedia presentations that are embedded in the nodes (in terms of scenes, scenario, spatial and temporal compositions). There is a vertical flow of information.

The information in a hypermedia information system may be classified into different layers (fig. 1.). the upper layer includes the objects that represent fuzzy knowledge (concepts, relationships,

and their fuzzy features)[8]. The lower layer represents the hypermedia information network in terms of nodes and links. The nodes inherit

semantic properties from concepts belonging to the upper layer while links inherit their properties from corresponding relationships.



**fig. 1. The information layers represented by the data model.**

Thus the hypermedia networks conforms to constraints imposed by the properties of concepts and relationships. The lower layer represents the interactive multimedia presentations that are included in the nodes of the hypermedia network. The presentations are represented in terms of scenes, scenarios [7]. The lowest layer of information represents the multimedia objects and their functionality as well as their transformations so that they can participate in an interactive presentation.

In KNOWEL we implemented the upper layer regarding knowledge. We will briefly describe the classes that are involved. Knowledge representation in the model is supported by the classes concept, relationship, weight, context and rules. These classes define a framework for concepts and the allowable relationships among them, modelling the related uncertainty features.

The knowledge network consists of instances of the class (concept), the relationships between the concepts instances. The uncertain features of the relationship instances are supported by the related instances of the class weight. The hypermedia network consists of the instances of the classes node and link and are classified in to types of concepts and relationships respectively.

The knowledge network imposes consistence constraints to the hypermedia network since every

node and link is associated to a concept and links respectively. For instance the relationship "husband\_of" is defined between the concepts "man" and "woman".

Thus a link must connect nodes of the concept type "man" and "woman" respectively. These constraints maintain the network in a consistent state assuming that the concepts, relationships and weights have been defined correctly.

The model is specified according to the object oriented approach as a class hierarchy. The model is defined according to a language that is based on the EBNF syntax (see APPENDIX A.). Moreover in the appendix there is the definition of fundamental entities for the model.

*Semantic Properties of a Concept.*

A semantic property of a concept has the following form (attribute - value,). We define a class for the representation of the semantic properties called Property:

```
class Property subclass of Object
attributes
    Attribute      domain STRINGS
    Value          domain VALUES
methods
    Attribute      return_attribute()
```

```

VALUE      return_value()
Attribute  set_attribute(a domain
           ATTRIBUTES)
VALUE      set_value(v domain
           VALUES)
end

```

The field attribute represents some semantic property while the field value is the corresponding. The list of properties may be exploited in order to classify and/or retrieve concepts and or nodes.

### 3.1 Class Concept

The nodes of a hypermedia network need to be classified into categories according to some common features. We call these categories concepts which are represented in the proposed model by the class concept. In general this class represents the semantic information that is included in the nodes of a hypermedia network.

The definition of the class according to the aforementioned language follows:

```

class Concept : subclass of Object
attributes
  Name   domain STRINGS
  Date   domain DATE
  MMSceneId domain IDENTIFIERS
  Author domain LIST of STRINGS
  Properties domain LIST of IDENTIFIERS
  Type_of domain LIST of IDENTIFIERS
methods
  save(ofstream domain FILENAMES)
  retrieve(ifstream domain FILENAMES)
  display()
  LIST of IDENTIFIERS return_ancestors()
  LIST of IDENTIFIERS
    return_descendants()
  LIST of Properties
    get_inherited_properties()
  LIST of Properties
    get_defined_properties()
  LIST of IDENTIFIERS get_Type_of()
  LIST of IDENTIFIER
    get_relationships_as_source()
  LIST of IDENTIFIER
    get_relationships_as_target()
  LIST of IDENTIFIER
    get_inh_relationships_as_source()
  LIST of IDENTIFIER
    get_inh_relationships_as_target()
end

```

The presented set of methods is not exhaustive, only the most relevant ones were selected.

The attributes of the class include the name of the concept (name), a short description (description), a list of the authors (Author) and a list of the semantic properties of the concept (Properties). The attribute MMSceneId is the identifier of a multimedia presentation that is related to the concept.

A concept may be a subtype of another concept (i.e. the set of instances of a concept is a subset of the set of instances of another concept). For instance the concept "human" is, in principle, a subset of the concept "being". This requirement is represented in the class by the attribute Type\_of in which there is a list of identifiers of the concept that serve as supertypes for the current one.

The current concept inherits all the semantic properties of all its supertype concepts. Thus property inheritance is supported in the model.

The most interesting methods of the model follow:

- return\_ancestors() returns a list with the identifiers of the concept's superconcepts. This method is recursive and returns both the immediate and implicit superconcepts.
- return\_descendants() returns a list of the identifiers of the concepts that have the current concept as a superconcept.
- get\_inherited\_properties(): returns a list of the properties that are inherited from the superconcepts. This is a recursive collection of properties inherited from the explicit and implicit ancestors.
- get\_defined\_properties(): returns the list of properties that are defined within the current concept.
- get\_Type\_of() returns a list with the identifiers of the concept's immediate superconcepts.
- get\_relationships\_as\_source() that returns a list of the relationship identifiers that have the current concept as source
- get\_relationships\_as\_target() that returns a list of the relationship identifiers that have the current concept as target.
- get\_inh\_relationships\_as\_source() that returns a list of the relationship identifiers that the current concept inherits as source from its superconcepts.
- get\_inh\_relationships\_as\_target() that returns a list of the relationship identifiers that the current concept inherits as target from its superconcepts.

## 2.2 Class Weight

The class represents the measure and type of the semantic closeness (referred to as weight) between two concepts in terms of fuzzy values in different contexts. Weight is an abstract class having the subclasses: Nominal and Ordinal. Each of them corresponds to a different type of values indicating the weight. In the cases the weight values are transformed into fuzzy values using the appropriate transformation functions. The Nominal class represents an entity that is quantifiable and is expressed by a list of nominal values which are not ordered (for instance : [v1,v2,v3,v4,v5,v6] ). The list is mapped to the fuzzy domain using the auto\_select function [GUP88]. The ordinal class is a subclass of class Weight and is used in the cases of physical values that may be classified in categories according to set of lexical values. Such a set could be: {"small", "moderate", "big", "huge"}. The lexical values correspond to a set of regions of physical values measuring the entity described by the weight instance. The physical value coming from the real world is classified in a lexical value category according to the region it belongs (i.e. the year 1370 is classified in the category indicated by the lexical value "medieval" if historical period is the case). The physical value may also be transformed using the appropriate transformation function to a fuzzy one. The resulting value is the degree of belief (d.o.b.) that this physical value belongs to the set of values characterised by the specific lexical value. The description of the class follows:

```
class Weight subclass of Object
attributes
  L_Values domain LIST of STRING
  P_Values domain LIST of RANGE
  Equations domain LIST of f
  Inv_Equations domain LIST of f
  Weight_Type domain STRING
methods
  FLOAT get_fuzzy_value(v1 domain
    VALUE,n domain integer)
  VALUE get_physical_value(v1
    domain VALUE,
    n domain integer)
end
```

The attributes of the class value include the list of lexical values ( L\_Values) that are mapped to a set of physical values (P\_Values). The mapping is based on a set of transformation functions (Equation). The inverse procedure (de-fuzzification is achieved by the inverse functions (Inv\_Equation).

Moreover there is a lexical value featuring the Weight. (Weight\_Type).

The most interesting methods of this class are:

- get\_fuzzy\_value(v1,n) that converts the physical value v1 into a fuzzy value for the nth lexical value using the corresponding transformation function.
- get\_physical\_value(f,n) that converts the fuzzy value f into a physical value for the nth lexical value using the corresponding inverse transformation function.

## 2.3. Relationship class

This class models the allowable relationships among concepts. It covers all the kinds of relationships among concepts referred to above: proximity, value and links. It is based on fuzzy functions of graded membership for storage and transformation of physical and lexical values.

The relationships are defined between a pair of concepts:

```
concept_pair = (“ identifier ”, identifier ”)
```

The description of the class follows:

```
class Relationship subclass of Object
attributes
  Relname          domain STRINGS
  concept_pairs    domain LIST of
                  concept_pair
  Weight_Type     domain
                  IDENTIFIERS
methods
  Apply_Energy_Metric(S1 domain LIST
    of IDENTIFIERS, n domain
    INTEGERS)
  get_concept_pairs()
  get_inherited_concept_pairs()
end
```

An instance of the class connects a source concept (SourceConcept) and a target concept (TargetConcept). These concept form concept\_pair. The attribute concept\_pairs is the list of the concept pairs that are allowed to be connected with the current relationship.

The semantics of the relationship are denoted by the attribute Relname, while the uncertainty features and manipulation of this relationships are represented by the related instance of the class Weight (WeightType).

There is a set of methods that represent actions on the semantic content of a relationship and are

inspired from corresponding methodologies from fuzzy logic]. The method `Apply_Energy_Metric(S1, n)` returns the total measure of information (that can be used as an overall degree of belief) that is included in the set of nodes `S1` as regards the `n`th lexical value of the weight that is related to the current relationship. If no parameters are provided then this method is applied to all the links that are typed in the current relationship in the hypermedia network.

The method `get_concept_pairs()` returns a list of concept pairs that may be connected by this relationships. The method `get_inherited_concept_pairs()` returns a list of concept pairs that are connected with some relationship and are ancestors of the concepts in the `concept_pairs` list.

## 2.4 Rule class

The rules are the elements of the model that facilitate manipulation of a hypermedia network as well as apply reasoning procedures. There are rules that: select nodes and links that fulfil certain conditions; trigger other actions (data manipulation, link creation or modification etc.); and impose integrity constraints. Moreover rules may be used in order to invoke methods from other classes depending upon conditions that are fulfilled or not, and serve inference procedures. This class is not yet implemented in KNOWEL so we will not present it in detail here.

## 3. The editor

The KNOWEL editor implements the higher layer of the model that is mentioned before. The program enables the author create and manipulate a fuzzy knowledge network in terms of:

- creating and modifying concepts
- creating and manipulating weights, which manage uncertainty of the degree to which a physical value belongs to a set or not.
- creating and modifying relationships among concepts and assign uncertainty management as defined in weight instances.
- evaluate the uncertainty of physical values

### 3.1. Creating and modifying concepts

The author may create and modify concepts (fig. 2.). The concepts correspond to the concept class of the aforementioned model and consists of

- a name
- a short textual description

- a list of authors
- indication of the abstraction level in which the concept resides (abstraction level)
  - the list of concepts from which the current concept inherits properties (`Type_of`)
  - the list of semantic properties of the current concept (`Properties`).

fig. 2. Concept creation/modification dialog box. Concept attributes may be seen.

#### *abstraction level*

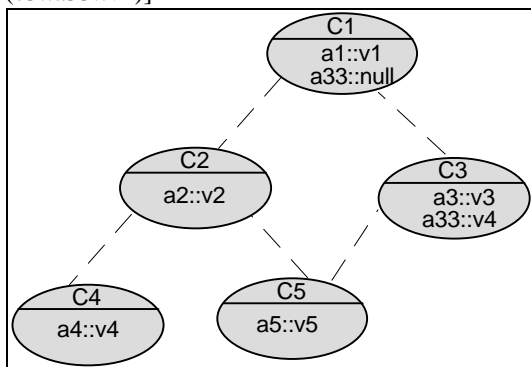
An important feature of KNOWEL is the classification of the concepts into categories called abstraction levels. The abstraction levels refer to the degree of generality of a concept. The most general concepts are classified in the abstraction level 0 while most specific one are classified in lower levels (1,2...). Once the concept is classified into an abstraction level it may define its semantic ancestors, other concepts that will be included in the `Type_of` list. These concepts can only belong to higher abstraction levels and the current concept inherits all their properties.

fig. 3. Properties dialog box.

*properties*

A strong feature of KNOWEL is the mechanism for definition and propagation of semantic properties of concepts. The properties are pairs of the form <attribute, value> where *attribute* is a semantic attribute and *value* is the value of this *attribute* in the current concept. There is a list of existing attributes but the author may as well define a new one (fig. 3). In this case the value may be an arbitrary value that is provided by the user. As we will describe later in the paper the properties may be inherited to other concepts through an inheritance scheme.

The properties of a concept are either defined (these are the properties that are defined within the concept) or inherited (these are the properties that are inherited from the ancestor concepts defined in the *Type\_of* attribute). The inherited properties are referred within the concept in which they were firstly defined so this resolves cases where a conflict might occur in the case of multiple inheritance. For instance (fig. 4.) the inherited properties of concept c5 are: [(a5,v5)] while the inherited properties are: [(c1::a1::v1) (c1::a33::null), (c2::a2::v2), (c3::a33::v4)]



**fig. 4. An inheritance hierarchy for property propagation**

*inheritance features and constraints*

KNOWEL enables single and multiple inheritance of properties through the *Type\_of* attribute. The current concept inherits all the semantic properties of its ancestors, namely the concepts that are referred in the *Type\_of* list. There is a list of constraints for property inheritance:

- a property has the default value (*null*) until the user changes it
- the value of an inherited property may not be changed unless if it is *null*
- it is not allowed to define a property having the same attribute with an inherited one

In the case of multiple inheritance some properties might be inherited from more than one concept. In this case the property is referred in the current concept with a reference to the concept where it was initially defined (see fig. 4.)

If a property is inherited either with *null* or with a specific value then the property with the *null* value is preferred so that the current concept may attach its own value.

*abstraction level modifications*

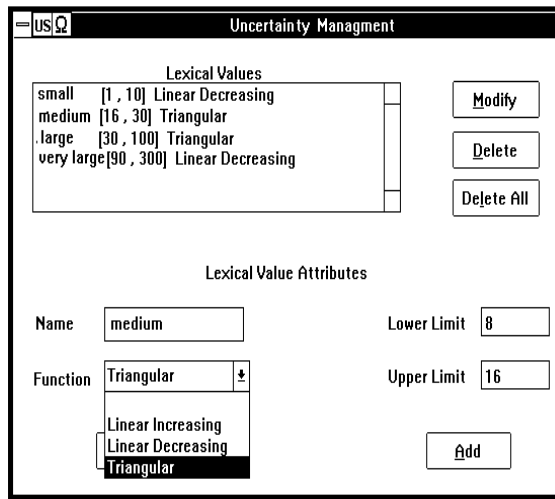
It is possible to change the abstraction level value of a concept. This causes reorganisation of the knowledge network and results in update of the *Type\_of* lists of related concepts and recalculation of inherited properties. The changes of abstraction levels as regards the concept itself, the ancestors and the descendants are summarised in Table 1.

	move concept to a more specific abstraction level	move concept to a more generic abstraction level
concept	- no change in the <i>Type_of</i> List - no change in the inherited properties	- ancestors in the same abstraction level are eliminated from the <i>Type_of</i> list - corresponding properties are eliminated from the inherited properties
ancestor	- no change in the <i>Type_of</i> List - No change in the inherited properties	- no change in the <i>Type_of</i> List
descendant	- if in the same abstraction level the concept is eliminated from the <i>Type_of</i> list - corresponding properties are eliminated from the inherited properties	- no change in the inherited properties - no change in the <i>Type_of</i> List - no change in the inherited properties

**Table. 1. The effects of abstraction level changes on the concept itself, its ancestors and descendants**

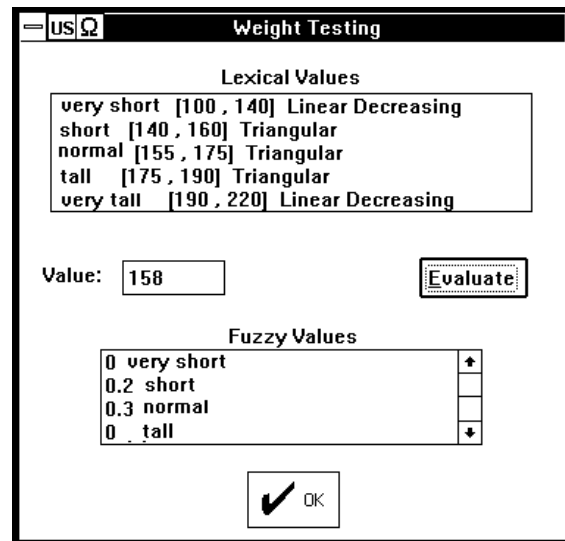
### 3.2. Creating and modifying weights

The system enables the author of the knowledge network define weights, which represent the fuzzy features of relationships and may be exploited for transforming physical values into fuzzy using a set of lexical values and the corresponding set of transformation functions. The weights are classified into nominal (where there is a set of lexical values which may not be ordered) and ordinal (where the set of lexical values may be ordered). In the case of ordinal weights the user may define the set of lexical values, attached with the domain of physical values and the transformation function (fig. 5.).



**fig. 5. Dialog box for definition of an ordinal weight**

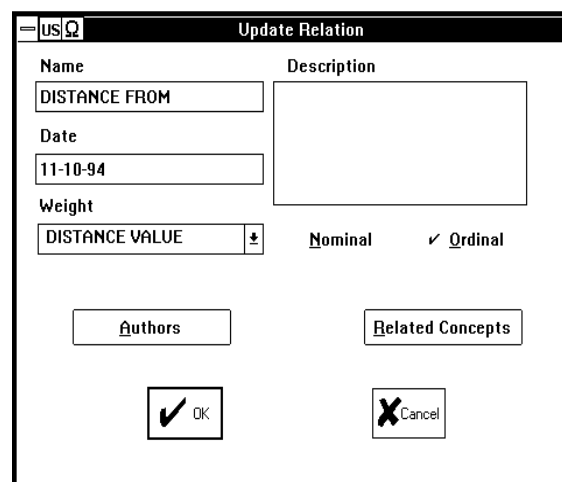
The user may evaluate the uncertainty of a physical value by selecting the appropriate choice. The system returns the degrees of belief that the given physical value belongs to the sets defined by the lexical values (fig. 6.).



**fig. 6. Dialog box for evaluation of a physical value that refers to a measure.**

### 3.3. Creating and modifying relationships

KNOWEL enables user define relationships among concepts. Each relationship (fig. 7) includes the name of the relationship, a short textual description, the list of authors, the list or related concepts and the weight that is attached to the relationship. The weight represents the fuzzy features of the relationship under definition. The related concepts may belong to any of the abstraction level.



**fig. 7. Dialog box for definition of a relationship**

An interesting issue is the way that relationships and concept pairs are inherited. In fig. 8. there a knowledge network defined with KNOWEL. For each of the concepts it is valid that it inherits relationships that have been defined in any of its semantic ancestors (e.g. woman - owns - car). On the other hand for each of the defined relationships it is valid that they are inherited to the concept pairs that are descendants of the initially connected concepts (e.g. lion-eats-woman).

The system has been implemented under Windows 3.11 using Borland C++/v. 3.1. As regards the graphical interface we used the OWL framework provided by this compiler. We also used the Container class hierarchy [1] that provides the classes for list and collection manipulation, something that was extensively used in the implementation of KNOWEL.

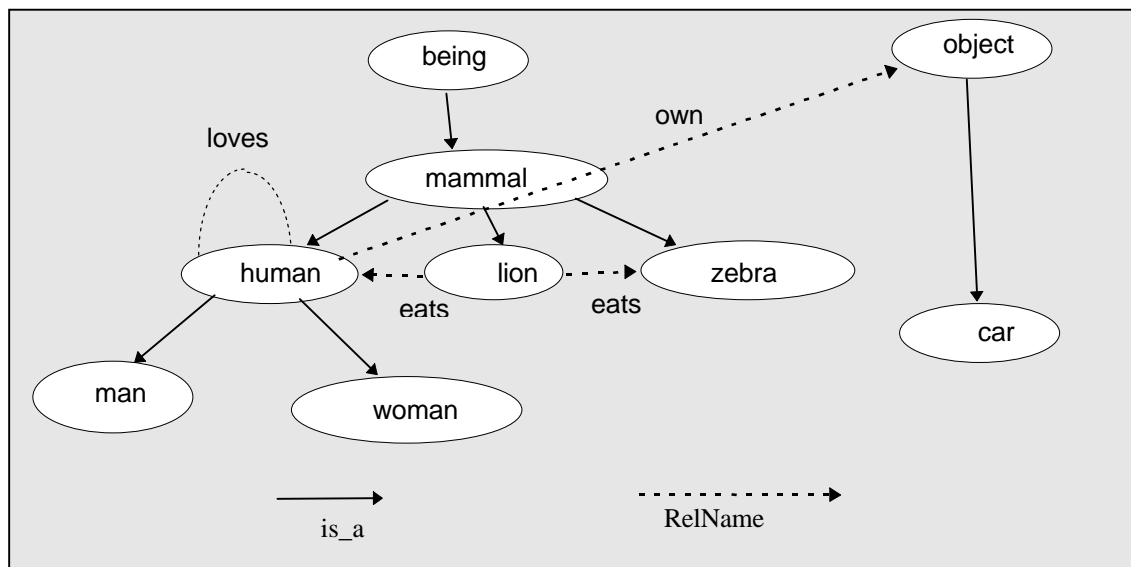


fig. 8. A sample knowledge network defined with KNOWEL

#### 4. Related Work

A system that proposes a model for hypermedia documents is the VODAK system [5] which is a general purpose OO DBMS. Parts of it have been adopted for representation of hypermedia documents. Using the VML language the user may define the hypermedia network in terms of metaclasses, categories. The issues that are not addressed by the VODAK systems are the following:

- The uncertainty of links is not supported
- Manipulation of the hypermedia network is not supported
- Only the operational semantics are addressed and there is not representation of the content and behaviour of the nodes as regards multimedia data.

Another model is HDM [3] which is an attempt for modelling hypertext applications in a more expressive way than the traditional scheme node-link. The applications are represented by 'Type' and an 'Instance'. Unlike other systems objects and relationships in HDM are first class objects having internal structure. This feature promotes the

navigation features of the system. The structure named Object and Relationship allow the definition of organisation of hypermedia networks. They may store meta-information enabling thus semantic classification and inference in the hypermedia network. The Dexter Hypertext Reference Model [4] aims at definition of a specification for modelling the structure and functionality of hypertext systems. The basic structural elements are the nodes the links and the anchors. This model deliberately does not represent the content of the node in order that the system remains open.

In the aforementioned models there are some issues that are not addressed. For instance there is no adequate representation and manipulation of the link semantics. Moreover none of these systems represents the uncertainty of links.

#### 5. Conclusion

The proposed system is the implementation of the first layer from the architecture shown in fig. 1.) We intend to implement other layers in order to evaluate the integrated model in real world hypermedia applications.

The model proposed, is one of the first attempts to model all the information layers in a hypermedia

system and tries to overcome the limitations of existing models which come with a fixed set of predefined modelling primitives that fit well only into the original application domains. Other proposed standards (MHEG, HYTIME) refer to structural organisation of hypermedia documents without special reference to their semantics.

The knowledge editor implements a knowledge representation scheme that embeds fuzzy features as regards relationships between concepts. This scheme is applicable to hypermedia information networks and may be used as a constraint framework to enforce consistency in hypermedia information networks.

KNOWEL may be used to

- define concepts and their semantic properties
- define relationships and their fuzzy features that arise from weights that represent the
- fuzzy features of physical entities.

The consistency of the knowledge network is assured by the mechanisms that update the properties of each concept while it is updated (change of abstraction level results in modification of the inherited properties). Also multiple inheritance scheme is supported.

KNOWEL may be used for experimental purposes for the definition of knowledge networks with uncertainty features. Eventually decision support procedures may be supported when the rules are implemented.

The proposed system may be extended in the following directions:

*implementation and integration of the other layers*

There is a great potential in the implementation of the layer of the model that refers to hypermedia information networks. Then an important issue will be the mappings between the two layers (fuzzy knowledge and hypermedia information network). It is apparent that the fuzzy knowledge network will serve as a constraint framework which will impose consistency in the hypermedia network. Moreover it will be possible to maintain the network so that it remains in a consistent state.

*navigation semantics*

An apparent extension of KNOWEL are browsing and navigation facilities in order that the knowledge network is exploitable to the user. Moreover visualisation of the information network would aid in viewing the nodes and links classified in the various abstraction levels.

*manipulation of the knowledge network*

Another extension would be the ability to delete a concept or a link. This action results in an update transaction during which the knowledge network is updated and reorganised so that it remains in a consistent state after the deletion

## REFERENCES

- [1] Borland C++ v3.1, Programmer's Reference Manual, 1991
- [2] R.Fox, J. Josephson, " Peirce: A tool for Abductive Inference", proceedings of IEEE/TAI'94 Conference, 11/1994, pp. 212-218.
- [3] F. Garzoto, P.Paolini, D. Shwabe. "HDM - A Model Based Approach to Hypermedia Application Design", ACM - TOIS, vol 11(1), p 1-26.
- [4] F. Halasz, M. Swartz, "The Dexter Reference Model", in proceedings of 1st Hypertext NIST Standardization Workshop, (Gaithersburg, MD, 1990), pp. 95-104
- [5] W. Klas, K. Aberer, E. Neuhold, "Object Oriented Modelling for Hypermedia systems Using the VODAK Model Language", NATO - ASI on OODBMSs, Springer Verlag, 1994, pp. 389-433.
- [6] S. Tano et. al." Three Layered Fuzzy Inference and Self Wondering Mechanism as Natural Language Processing Engine of FLINS", proceedings of IEEE/TAI'94 Conference, 11/1994, pp. 212-218.
- [7] M. Vazirgiannis, M. Hatzopoulos, "A Script Based Approach for Interactive Multimedia Applications", Proceedings of the MMM (International Conference on Multi-Media Modeling) Conference, Singapore, (November 9-12, 1993).
- [8] M. Vazirgiannis, K.Petrou, A. Tsompanidis, M.Hatzopoulos, "An Object Oriented Framework for Knowledge representation based on Fuzzy sets", Intelligent and Fuzzy Systems Journal, Wiley.
- [9] M. Vazirgiannis, "An Object Oriented Model For Representation Of Hypermedia Information Networks, Enriched With Fuzzy Knowledge Structures", PhD Thesis, Dept. of Informatics, University of Athens, 12/94.
- [10] L.Zadeh, "Commonsense Knowledge Representation Based on Fuzzy Logic", IEEE/ Computer Journall, October 83, pp.61-65.
- [11] L.Zadeh, "Knowledge Reoresentation Based in Fuzzy Logic", IEEE/ Transactions on Data and Knowledge engineering, March 89, pp.61-65.
- [12] M.M. Gupta et al., "Sinusoidal - Based cognitive mapping functions", in Fuzzy Logic in Knowledge Based systems, decision and control, North-Holland, pp. 69-92, 1988.

## APPENDIX A.

The following language aims at description of the model classes and its definition has been based on the EBNF notation :

```
class =
    "class" class_name [ "subclass of"
    classes]
    "attributes" attribute_specifications
    "methods" method_specifications
    "end"

classes = class_name {"," class_name}
class_name = STRING
attribute_specifications = attribute {attribute}
attribute = STRING "domain" class_name
method_specifications = {method}
method = return_object_name method_name "("
[parameter_list]"")
return_object_name = class_name | "LIST of"
class_name
method_name = STRING
parameter_list = parameter {"," parameter}
parameter = par_name "domain" class name
```

### *Definition of semantic attributes*

```
ATTRIBUTES
= {ATTRIBUTE}
```

```
ATTRIBUTE
= STRING
```

### *Definition of the attribute values*

```
VALUES
= {VALUE}
```

```
VALUE
=     STRING
|     NUMBER
|     RANGE
|     LIST
```