Database Support for Knowledge-Based Image Evaluation

Ingrid M. Walter
Peter C. Lockemann
Hans-Hellmut Nagel

Fakultät für Informatik, Universität Karlsruhe
Postfach 6980, D-7500 Karlsruhe, FRG

Abstract

The evaluation of images has to cope with vast amounts of raw data usually acquired within short periods of time. This is especially true for situations where sequences of images must be analyzed in order to determine motions, more complex actions, or to interpret entire processes comprising various combinations of actions. Database systems will become an essential part of the support structure for image evaluation systems in order to facilitate the management of raw data as well as of intermediate and final results. This contribution introduces a subproblem of image sequence evaluation, the extraction of so-called episodes, i.e. actions of relatively high semantic complexity. The extraction is based on a three-part database consisting of an object section, a process section, and a rule section. The contribution describes the abstraction level of episodes, introduces the system architecture of an episode extraction system EPEX, develops a suitable data model for the database of a case study of traffic scenes, determines its implementation in terms of the relational data model, and finally discusses the novel requirements that database systems must observe for image evaluation applications.

1. Introduction

Image evaluation – the interpretation of the contents of digitized images – takes on an ever increasing role wherever automatic control is to be achieved predominantly on the basis of sensory input, chiefly although not exclusively pictorial input. Application areas are quality control, robotics, automated manufacturing, nuclear reactor maintenance, traffic control, to name just a few. In all these applications, the sensory input has to be related to a so-called world model in order to determine or prompt the appropriate reactions of the technical system – motion, voice output, optical signals, and the like. The world models we are concerned with here are on a level of complexity that requires the collection and evaluation of long sequences of images, or in other words, they deal with nontrivial actions of real-world objects. In such a case, a large volume of data has to be acquired within a short amount of time. The world model itself – a collection of general descriptions of objects and motions, of evaluation rules, but also of the results of past evaluations – is maintained on a permanent basis. Taken together, all these requirements make a database system an ideal support component in image evaluation systems.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Database support for pictorial information systems and for image processing/evaluation systems is not new but an area that has received wide attention over the past ten years or so (for more recent surveys of the field, see [Chan81a, Chan81b, Tamm82, Hwan83, Benn84, Gutt84, Tamu84, Bour85, Chan85a, Chan85b]), though interestingly enough not so much from the database community. However, most of this work essentially deals with the maintenance of large numbers of individual images. Rarely are the images considered in sequence. On the other hand, we are not only concerned with sequences, but also with their evaluation to a rather high abstraction level for complex objects such as work pieces, robot arms, grippers, cars, and their motions in environments such as manufacturing cells and street intersections, respectively. Although these kinds of applications have captured much attention in the more recent past (for an overview see [Nage85b], for an example from medicine see [Niem85] and from robotics [Levi87]), to the authors' knowledge little effort has been extended so far in the direction of supporting them by database technology.

Obviously, the evaluation of complex motions from image sequences is an extremely ambitious goal. Some help is derived from the fact that there are experimental systems that attempt to reduce the digitized input to an intermediate level where geometric objects are recognized as such [Nage83, Nage85a]. Nonetheless, our first objective is to learn more about the basic mechanisms to be employed towards image evaluation on the level discussed before. Our specific issue is to determine whether and to what extent existing database technology will contribute towards the overall objective, or whether and which novel solutions must be found within database technology.

Our presentation will be organized as follows. We begin by making more precise what we mean by the level of abstraction of our world model, introducing on the way the concept of an episode (chapter 2). In the subsequent chapter 3 we give an overview of the process for extracting episodes from the pictorial material utilizing the world model, describe the system architecture chosen here, and outline the database organization. Chapter 4 concentrates on one particular aspect of the database, the object section, introduces a suitable data model, and illustrates it with an example from urban traffic analysis. The implementation of the data model for the object section in terms of a relational database system is sketched out in chapter 5. Chapter 6 will conclude with first experiences and future directions of work.

2. The abstraction level of episodes

2.1 Abstraction levels of image evaluation

The basic material for image evaluation are digitized images that have been obtained by discretizing the image plane and quantizing the recorded grey values. As it is well known from television, the temporal axis has been discretized, too, thus yielding an image for each consecutive interval along the time axis. We refer to the corresponding temporal sequence as an *image sequence*.

The digitized images are processed by a succession of evaluation steps, resulting in more and more abstract levels of description: the grey values are aggregated into lines, edges, vertices, these into more complex attributes, and so on until one arrives at a set of descriptions of geometric objects. The set forms what is called a (*geometric*) *scene description* (GSD) (see [Neum84]).

In evaluating an image sequence we now consider the sequence of geometric scene descriptions. What further abstraction levels are desirable? Take as an example a subway station where the dispatcher must decide on whether a train may depart. A dispatcher could hardly respond in time, if the image evaluation system provided him only with the coordinates of each single object (e.g., a person) on the platform: he would first have to translate the coordinates of all objects into concepts such as "all persons are a sufficient distance away from the train, and none is moving towards it". Our long-range objective is precisely the automatic abstraction to such a level, and the provision of information representative for that level.

Consequently, further levels of abstraction must be introduced over and above the geometric scene description, namely those that tie sequences of such descriptions together. One conceivable way would be as follows. The first new level concerns itself with the progression of coordinate values of the objects, the so-called *trajectories* which must be established even though the object in question may become hidden from time to time, e.g., when passed by another object. The next level relates several objects and their trajectories, identifying ensembles of objects but also dynamic relationships between objects such as one car passing another, or a person approaching a train door. We refer to these relationships as *actions* and Neumann and Novak [Neum86] speak of them as events. In natural language, these relationships are typically represented by simple motion verbs such as "move", "accelerate", "pass". A further level, and the one that concerns us here, is one where several actions interact, forming what we call an *episode*. Examples of episodes in that sense are "entering a train" (combining actions such as leaving the waiting position, moving towards a door (not necessarily in a straight line, with periods of hesitation), stepping into the door), or "waiting for a train" (combining actions such as walking up and down the platform, taking a seat, pulling out a paper, reading it, folding it again, ignoring a train (in case there are several lines leaving the station)). If their composition is not all too complex, one may again represent episodes by verbs, albeit with semantics that are considerably more complicated than in the case of elementary actions.

2.2 The concept of an episode

For what follows it is important to give a more precise characterization of an episode. From what was said above, we distinguish an episode by the following properties.

1. An episode is a combination of actions. Actions are either elementary, i.e. are directly derived from sequences of geometric scene descriptions, or are composed of such actions.
2. There is some minimum number of actions (presumably depending on the application) that make up an episode. Witness the examples given in chapter 2.1.
3. There is a certain variability to the action structure of an episode. The order of actions may vary within limits, and some actions may be missing altogether. Take as an example the episode of a person waiting for a train. The person may first elect to sit on the bench, then get up and walk, or vice versa, or may even take turns at doing both. S/he may read a paper, or s/he may not.
4. The decision on which actions to combine into an episode requires knowledge beyond the one gained from the pictorial input. For example, whether ignoring a train passing the station constitutes part of waiting for a train depends on knowledge of whether there is a single line or more serving the station. Similar information, e.g., in a system for warning persons not to step too close to the tracks, is the knowledge that trains move on tracks and may endanger persons getting too close to the tracks.
5. The knowledge needed to associate actions with an episode also allows us to infer the past and the future of episodes from a limited observation of actions. For example, from the facts of a person waiting for a train and ignoring a passing train we infer that s/he is going to wait for (at least) another train. From the observation of a train leaving the station we infer that the episode of this train serving the station has come to an end.

It should have become clear, then, that the information required for the determination of a particular episode is given in terms of objects and trajectories extracted from the pictorial input, of general knowledge about the constraints objects and trajectories have to meet, of rules that determine actions, and of rules that control the combination of actions into episodes.

3. Extraction of episodes

3.1 The object section

The basic assumption underlying our system approach is that the image material has already been preprocessed into a geometric scene description. The GSD includes descriptions of all objects occurring in the image sequence, each description containing information that can be extracted from the images, such as object type, coordinates, boundaries, colour. Some of these properties may vary from one image to the next such as coordinates, boundaries, or even colour as in the case of a traffic light. As a consequence, we further assume that the preprocessing also identifies one object from one image to the next so that the GSD includes temporal information such as trajectories of objects or histories of colour changes.

The general knowledge about GSD objects is usually expressed on classes of similar objects. It is utilized when inferring the actual actions and episodes that take place on the level of specific (GSD) objects. Within the class level, the concept of generalization [Smit77] plays an essential role.

Generalization as it is needed here is stronger than what has traditionally been provided by single-level typing in database schemas or even the generalization mechanisms in semantic data models. Generalization may extend across a number of levels, e.g., vehicles may have subclasses streetcars and motor vehicles, the latter motor cars and trucks. Relationships, or more generally constraints, may be established on any of the levels, e.g., streetcars move on tracks, motor cars may hold a maximum of six persons. Another useful abstraction mechanism is aggregation or subset formation which may again occur on any level, e.g., the aggregation of streetcars and motor cars to

vehicles for transporting persons. Generalization, or its converse, specialization, is governed by strict type inheritance [Card86] whereas aggregation is not.

We use the term *object classes* for the general knowledge on potentially existing objects of a chosen domain and the term *object specimens* for the GSD part. Object classes and object specimens form the *object section* of the system. The connection between the two levels is established by the class-membership-relation. The object specimens are constructed via an image analysis system developing the GSD, the object classes must be determined and entered by the system user. Fig. 1 gives a diagrammatic outline.

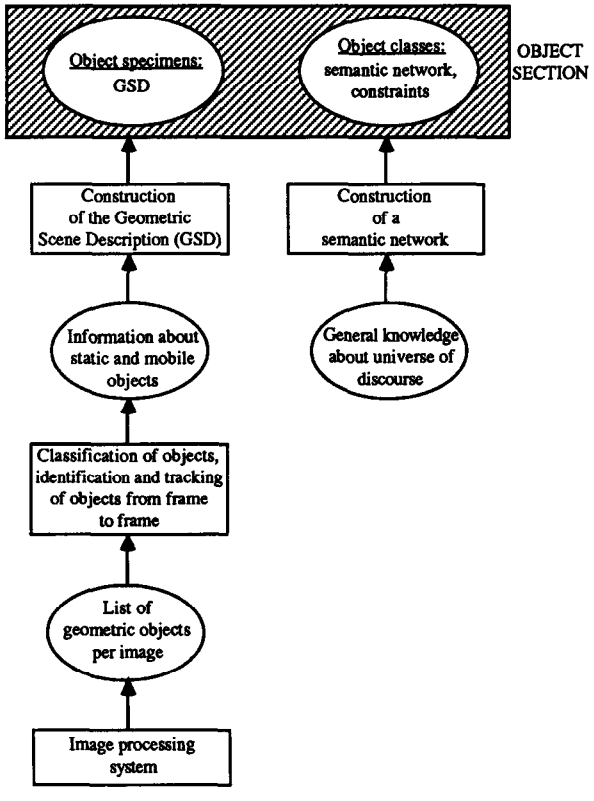

Fig. 1: Construction of the object section

3.2 The process section

As discussed in chapter 2, the information to be gleaned from the image data is the existence or non-existence of certain episodes or, more ambitiously, the enumeration of all episodes in a given image sequence. The data of the object section are merely the pieces from which the actions and episodes are stitched together.

When describing episodes, it is more natural to think of them as classes of similar episodes (and classes of actions as their constituents) than of individual episodes. The examples of chapter 2.1 and 2.2 clearly refer to classes of episodes. These classes are described in terms of their composition of actions of certain classes, the various orders in which these may take place, and an indication which actions are optional. Again, the classes must be constructed a-priori. In turn, they may be generalized or aggregated. Consequently, there is a large degree of structural conformity between the object section and the section of episodes and actions, or as we henceforth call it, the *process section.*

The basic difference between the two is on the level of specimens. In the object section all object specimens associated with the actual image sequence have been provided by the image evaluation routines and are the source for further processing. By contrast, in the process section the process specimens are the goal of all inferences: using the specimens and classes of the object section and the classes of the process section, actually occurring episodes are to be deduced. Hence, episode specimens will either not be kept permanently at all, or only if those already inferred from an image sequence are to be preserved for future reference.

Since we consider episodes to be associated with natural language verbs, the representation of episodes should take its cues from modern linguistic theories. One representation that comes particularly close to the models of the database world are Fillmore's case frames [Fill68] augmented by concepts introduced by Schank [Scha75, Scha77].

3.3 Rule world and system architecture

To infer episode specimens from object specimens via object and process classes requires a set of inference rules that tie them all together. In addition, control rules are needed that determine the strategies to be pursued during the inference process. The set of rules determines the *rule section* of the system.

There are close interdependencies between the three sections. One may keep the number of classes and their relationships small; much of the domain knowledge is then likely to end up in the form of rules. Or on the contrary, one may accumulate as much of the domain knowledge as possible within the classes, leaving to the rule section only very broad rules for interconnecting objects and processes. This is the approach that seems to prevail so far within the database community.

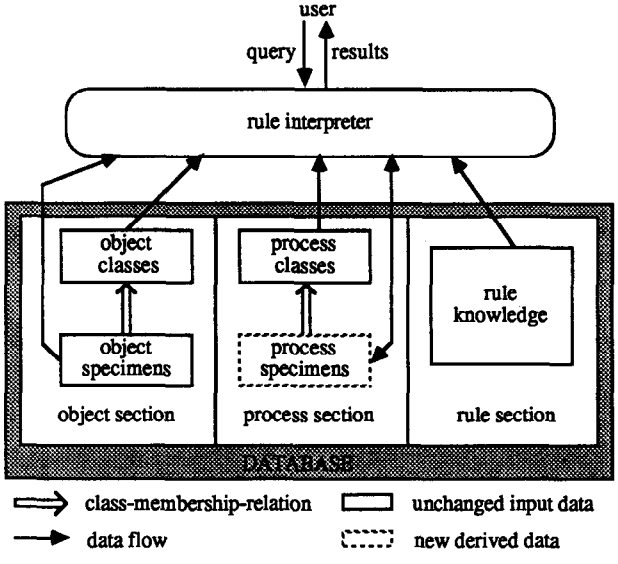

Fig. 2: The EPEX system structure

We propose a pragmatic approach. It is well-known that users, especially those that have to provide the domain knowledge and check out its representations, feel most comfortable with diagrams. Consequently, we set as a goal to capture as much of the domain semantics as possible by means of structural concepts, i.e. those for which fairly straightforward graphic representations exist. Since this also is the traditional approach to database interfaces, such a solution promises to make particularly good use of existing database technology and design methodology.

Indeed, it turns out that a similar approach can be taken for the rule section, allowing for easy storing and selective retrieval in a database system. The mechanism used is that of Augmented Transition Networks (ATN, [Wood70]). We shall not go into details in the remainder, though. The interested reader is referred to [Walt86].

The resulting architecture of the system, EPisode EXtraction (EPEX), is depicted in fig. 2.

4. Case study: A traffic scene

4.1 Selection of a data model

Like any modern high-level view of the application world, the approach to database modelling should be object-oriented. This is not the place to compare the multitude of object-oriented approaches to database systems (for a survey on more recent work see [Ditt86]). In particular, only those solutions are candidates that can be integrated with logic programming languages without too much difficulty, in order to meet the objectives of inference. We decided to choose the knowledge representation language KL-ONE [Brac85] as basis for the EPEX data model for the following reasons:

- It is in many respects similar to the widely used entity-relationship model.
- In addition, it supports the distinction between specimens (called individual concepts) and classes (called generic concepts) and includes an inheritance mechanism.
- Relationships including generalization/specialization can be established between generic concepts.
- Rule-based knowledge can easily be associated.
- Besides objects, case frames can easily be expressed.

We assume some acquaintance with KL-ONE so that we sketch only its general ideas and concentrate on some extensions that prove necessary for EPEX.

4.2 Basic concepts of the data model

As the EPEX data model is based on KL-ONE, we use its terminology. The basic structures are *generic concepts* and *individual concepts*, modelling classes and specimens, respectively. The class-membership-relation corresponds to the *individuation relation* of the data model.

Concepts are described by structure forming relations. The *generalization relation* forms hierarchies of concepts. Concepts within a hierarchy inherit attributes of the more general concepts. *Roles* describe binary relationships between concepts. Unlike the relations in the entity-relationship model [Chen76], roles are directed. As such they are associated with their respective domains and specified by *role descriptors*, which consist of the following features: value restriction (v/r), number restriction, role restriction, and role differentiation. In fig. 3 "truck" is a specialization of "motor vehicle". "Truck" restricts the value of the inherited role "driver" to "trucker" and the number of the inherited role "occupants" to (0,2). The *role fillers* of an individual concept I for a given role are all individual concepts standing in the role relationship to I. In fig. 5a the individual concept "green" is a role filler of the individual concept "signal 1" for the role "colour". The pair ("signal 1", "green") satisfies the conditions in the role descriptor.

In addition to these KL-ONE constructs, the EPEX data model provides some extensions necessary for our application. These are: default-value as a feature of the role descriptor, generic role filler, time-dependent individuation, time-dependent roles, and dependent roles. We will comment on time-dependency and dependent roles.

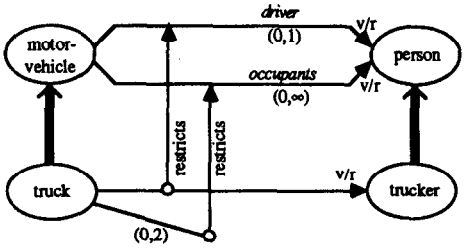

Fig. 3: Some KL-ONE constructs

The notion of (discrete) time plays an essential role in the evaluation of image sequences. In EPEX time-dependency occurs in two ways. First, specimens may belong to different classes during different time intervals. For example, a specimen "car herby" may belong to a class "normally occupied car" and then to a class "overcrowded car". In our data model this is expressed by augmenting the individuation relation with intervals of validity. This is illustrated in fig. 4.

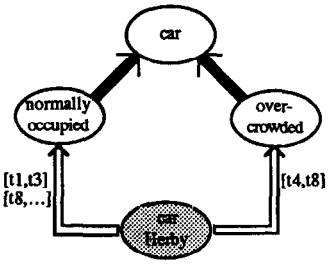

Fig. 4: Time-dependent individuation
(⇨ individuation; individual concepts are shaded)

Secondly, roles may be time-dependent. We represent this fact in the data model by augmenting the role descriptor with a time restriction using the values time-variable interval (var-i), time-variable point (var-p) and constant. An example described by time-variable interval is the role "colour" of "traffic-light" in fig. 5a. It is sufficent to know the moments of light change and the light immediately after the change, in order to deduce the light status at any time between two changes. In our example "colour" at [t4] is "green". The role "locus" in fig. 5b belongs to time-variable point. If the coordinates are not known at a given moment, one cannot use the latest value known before this moment.

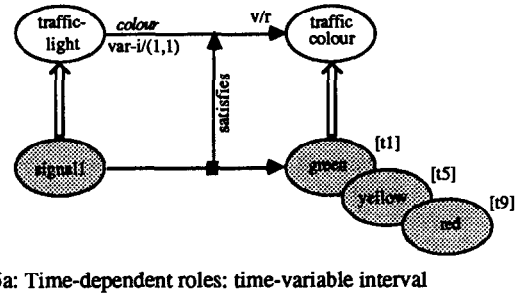

Fig. 5a: Time-dependent roles: time-variable interval

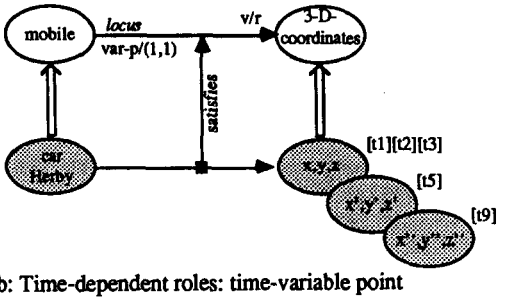

Fig. 5b: Time-dependent roles: time-variable point

The construct *dependent role* in the EPEX data model allows the representation of functional dependency between roles of the same concept. The role descriptor is extended by a function using role fillers of other roles of the same concept as input values and providing individual concepts compatible with the value restriction given in the role descriptor. The other features of the role descriptor are unchanged. An example using a dependent role is shown in fig. 6. The role fillers of the role "speed" are computed from fillers of the role "locus".

This extension for representing functional dependency allows also a natural representation of structural descriptions, which are defined in KL-ONE in a rather complicated way. We use only structural descriptions representing dependencies between roles of one concept. A structural description is a function, providing a truth value for a given set of role fillers. If the value is "false", the individuation relationship cannot be established. Structural descriptions are a special form of dependent roles – labelled *cond* – where the number restriction is (1,1) and the role filler must be "true". Structural descriptions are inherited like other roles. The structural description "$\subseteq$" in fig. 6 assures that a filler of "driver" is also a filler of "occupants".

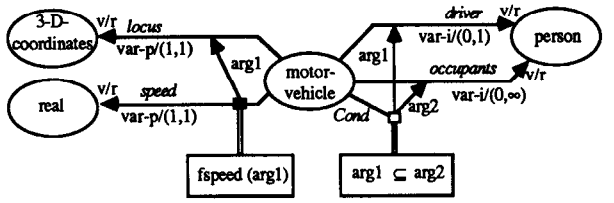

Fig. 6: Constructs "dependent-role" and "dependent-role as structural description"

For easy modelling of geometric objects we provide some basic generic concepts like 3-D-coordinates and polygons. Processes are modelled using exactly the same representation formalism. The generic concepts provided for them include primitives of the Conceptual-Dependency-Theory (CD-Theory) of Schank [Scha77] like "ptrans", "move", "ingest", "expel", as well as some additional ones.

The formalisms of the data model described so far are for modelling object data and process data. Rule data, the third part of the EPEX database, is based on the formalisms provided by ATN [Wood70]. ATNs describe the order of rule handling but say little about contents of the rules as they are domain dependent. Hence, user-written rules (actions and tests) must be attached. We will not go into much detail, but mention a few. An important class of rules represents knowledge about geometric relations between different objects. Typical are rules like "distance-less", "distance-more", "above-object". Another rule class represents time relations between processes. Rules like "before", "after", "during" are defined. Here we follow the notions of [Alle84].

4.3 Overview of the traffic scene

For a case study we chose the domain of traffic scenes. The scene we have modelled is a rather complicated traffic intersection in Karlsruhe, called "Durlacher Tor". Fig. 7 shows a sketch of part of the intersection, fig. 8 a camera view.

The intersection is formed of several multi-lane roads. Turning lanes for different directions are provided. There are separate lanes for pedestrians and bikes. Parking lanes and parking lots are within the wider range of the intersection. Streetcar-lanes cross the intersection.The traffic flow is regulated by traffic lights.

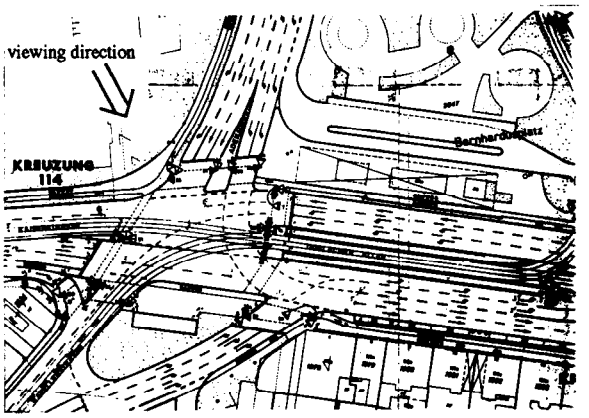

Fig. 7: The "Durlacher Tor"

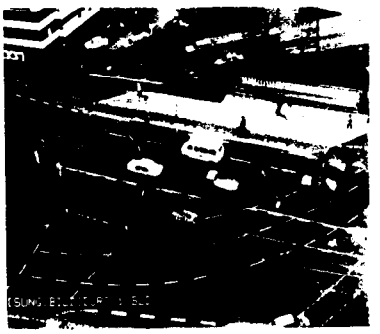

Fig. 8: Camera view of the "Durlacher Tor"

The descriptions of these features of the intersection together with the bordering buildings are the static components of the scene. Dynamic components are all moveable objects like cars, streetcars, persons, animals. Data on general knowledge of traffic scenes like the right of way form the third part of the object section.

For the process section, possible processes within the intersection area are to be modelled. These include simple processes like "drive" or "change-lane" as well as episodes like "turn", "traverse intersection" or "wait for streetcar".

4.4 Modelling the traffic scene

To model the traffic scene just mentioned, some 50 concepts with about 4 roles on average are needed. Lack of space forbids a complete exposition of these concepts. We just give the flavour for some parts of the object section representing knowledge about traffic scenes.The examples used in figs. 3 to 6 were taken from this object domain and show some modelling aspects. As mentioned in chapter 4.3 there are three subdivisions within the object section represented by three classes: mobile (moveable objects), stationary (static objects), and abstract (overall knowledge). The modelling of subclasses of mobile is rather straightforward. An example of a subclass of abstract is speed-limit, describing the maximum allowable speed for a movable object in a certain location.

The part of the object section we describe in more detail is from the stationary part, namely streets. Streets are fairly complex with quite a few interlocked concepts. The class street covers not only driving lanes but sidewalks, parking lanes, and cycle lanes. Streets are described not only by geometrical features like polygones for the street course but by abstract features like allowed users. Fig. 9 shows a small part of the modelling of street.

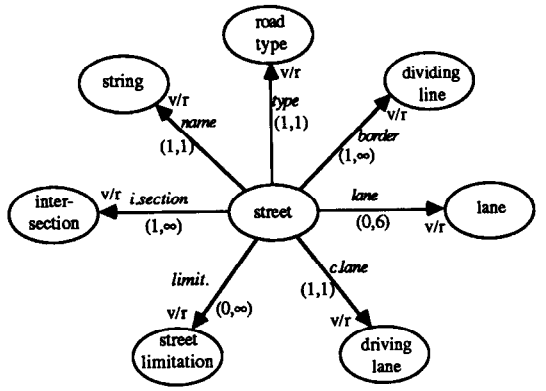

Fig. 9a: Modelling the class "street"

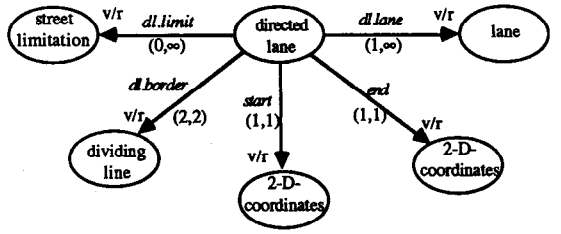

Fig. 9b: Modelling the class "directed lane"

5. Implementation by relational database

5.1 Extraction of episodes

No database system exists at present that offers KL-ONE as its data model. Consequently, the data model described in chapter 4 must be considered as a semantic model, to be mapped to a conventional data model. Among the possible choices, we selected for EPEX the relational data model.

The design of the relational schema depends on the characteristics of the retrieval sequences. A basis for predicting these is the extraction process for episodes which we describe briefly. The extraction is initiated by a query from the EPEX user. At this point, the EPEX database contains the object and process classes, the rules, and all object specimens (see fig. 2). In EPEX these data are used as mere input data and remain unchanged during the whole extraction process.

The extraction itself is guided by the rule interpreter. The result of the extraction is a set of all process specimens matching the query. These specimens and all process specimens generated during the extraction process may or may not be stored in the specimen level of the process section. Hence, process specimens are the only data in the database that change during extraction.

For the extraction the rule interpreter uses the information stored in the database. It starts by looking for an executable rule in the rule section. A rule is a transition between states. Transitions may be associated with the activity of extracting data – generic or individual – from the object and process sections. This may cause a certain amount of navigation from an episode schema to object classes to object specimens. Transitions may themselves depend on tests utilizing data provided by activities of the kind just mentioned. Failure of tests may cause backtracking within a rule, or selection of a new rule. One may also associate transitions with activities constructing episodes from the extracted and tested data in a stepwise fashion. Activities may also be used to interact with the user.

Clearly then, access to the database is both navigational – utilizing the structural links between episodes and objects and among objects – and associative based on role values. We also note that both types of access are affected by the necessity to observe inheritance. Associative access is served well by a relational database whereas navigation and inheritance seem to ask for an object-oriented DBMS. Nonetheless, in order to gain as much experience as possible within a short period, we decided to rely on a proven data model and a widely available DBMS.

5.2 The relational schema

The natural impulse is to consider specimens, i.e. individual concepts, as instances of the classes, i.e. generic concepts, and, consequently, to treat the generic concepts as types to be collected into a database schema. On second thought, however, this approach is entirely unacceptable. There are a number of reasons:

- The generic concepts are as much the subject of processing (primarily retrieval) as are the individual concepts. In particular, the abstraction mechanisms such as generalization, aggregation, and the relationships (roles) between them are exploited extensively, so that it remains unclear where the database schema level would have to be placed.
- Inheritance of properties may carry across all abstraction levels from top to bottom [Card84].
- A concept may have more than one supertype (polymorphism [Card86]).
- Generalization may vary over time.

The most radical solution – one followed, e.g., in two KL-ONE front-ends to NF2-based DBMS [Reim86, Börn87] – stores the entire generic data, like the individuals, as tuples of some very few relations. These relations are devoid of any world model semantics. They essentially have a bookkeeping character in much the same way as have data dictionary relations for representing relational schemas [Mart80].

Basically, for the generic concepts we follow the same path [Ikke87]. However, we utilize a larger number of relations in order to clearly separate the different aspects of the data model. Most important, in order to speed up reading access, we store information that is due to inheritance redundantly rather than derive it anew on each access. Because of the predominance of this type of access we are willing to pay for the ensuing overhead on updates. The basic ideas of our solution for the generic data are the following:

- Every concept is assigned a permanent, system-generated identifier when it is declared to the system for the first time. The identifier is used to establish the links between tuples of different relations much like surrogates [Codd79] or references [Hask81].
- The generic information is considered time-independent. Consequently, temporal information will be carried with the individuals, and only with these.
- The generalization hierarchy is represented in a separate relation HIERARCHY connecting super- and subconcepts. This relation is mostly used for access to generic concepts. For fast access to individual concepts, the transitive closure is (redundantly) represented by a second relation COMPLETE-HIERARCHY.
- The fundamental relation CONCEPT-ROLES relates roles to their respective concepts. It contains the name of the source concept of the role, role descriptor information together with inheritance information, and the name of the relation where the corresponding individual concepts are stored. In this way, the directedness of roles is mapped into the relational model. The CONCEPT-ROLES relation is the main carrier of redundant information as indicated above: we duplicate inherited roles and store them with each subconcept. This means that the role descriptor information has to be

computed during creation or update of the subconcept from the entire hierarchy above it. The same holds for restricted or differentiated roles, which traditionally would be referred to as consistency constraints. This may cause difficulties with later updates of the subconcept, because it is not in general decidable where the locally stored information originally came from. We therefore provide a second relation LOCAL-CONCEPT-ROLES, which contains such administrative knowledge. A similar approach is taken for most of the other relations of the generic level, to be discussed below.

- The ensuing redundancy in CONCEPT-ROLES must, of course, be observed and controlled during database updates. For support, a relation REDUNDANCIES is added.
- The relation DEFAULTS deals with default values.
- Finally on the generic level, the relations FUNCTION, FUNCTION-PARAMETERS, STRUCTURAL-DESCRIPTIONS, and STRUCT-DESC-PARAMETERS describe in detail the functions and structural descriptions, respectively.

The relations for the generic concepts are summarized in fig.10. In fact, their design is less critical than that for the individual concepts because it is there where the masses of data arise. Again, we just sketch the basic ideas somewhat similar to work on a KEE database [Abar86].

- Originally we had planned to take on this level a more object-oriented approach by creating a tuple for each individual concept. If we also declared a relation for each generic concept, comprising all time-independent and single-valued roles, such a tuple would have to be entered into each relation representing a superconcept for it. It would contain the name of an individual concept individuating the generic concept and a role filler for each role. In this way most of the information pertaining to a given individual could be accessed in a single step. Note that this approach implies the dynamic declaration, creation and deletion of relations as well as some mechanism for automatically naming them. The main obstacle, however, was the huge amount of redundancy arising from role inheritance. Unlike on the generic level, this redundancy appeared to us to be intolerable on the individual level. Furthermore, the majority of the roles in our application turned out to be multi-valued and/or time-dependent. Once one decides to drop inherited roles from the representation of individuals, however, the object-oriented approach makes less sense. Instead a separate, dynamically created relation is provided for each local role of a generic concept. The relation name consists of the concept name and the role name. The relation contains the name of the individual concept, its filler value (which, for example in case of geometric coordinates, may consist of several components), and begin and end in case of time dependency. They are essentially history relations [Klop83]. Each tuple in this relation satisfies the conditions in the role descriptor on the generic level. An example is given in fig. 11.
- An individual concept is associated with the generic data via a tuple (or several tuples in case of multiple hierarchies) of a predefined relation IND-CONCEPT; the association may again be time-dependent. The relation is used for two purposes: to check consistency of a local filler value with the associated domain (type inheritance), and to derive non-local filler values (value inheritance). Incidentally, the inheritance rules are fairly complicated whenever on the way up from the individual object a polyhierarchy is encountered, because filler values may then depend on the sequence of superconcepts that forms the current context.

Fig. 11 gives an example for the individual object concepts of fig.5a.

HIERARCHY (subconcept, superconcept)
COMPLETE-HIERARCHY (subconcept, superconcept)
CONCEPT-ROLES (concept, role-name, ind-relation-name, time-type, inheritance, value-res, min-numb, max-numb, function, default)
REDUNDANCIES (concept, role-name, value-res)
DEFAULTS (concept, role-name, default-value)
FUNCTIONS (concept, role-name, f-name)
FUNCTION-PARAMETERS (concept, role-name, par-no, par-role)
STRUCTURAL-DESCRIPTIONS (concept, cond-no, condition)
STRUCT-DESC-PARAMETERS (concept, cond-no, par-no, par-role)
LOCAL-CONCEPT-ROLES (concept, role-name, time-type & inheritance, value-res, min-numb, max-numb, function, default)
LOCAL-DEFAULTS (concept, role-name, default-value)
LOCAL-FUNCTIONS (concept, role-name, f-name)
LOCAL-FUNCTION-PARAMETERS (concept, role-name, par-no, par-role)
LOCAL-STRUCTURAL-DESCRIPTIONS (concept, cond-no, condition)
LOCAL-STRUCT-DESC-PARAMETERS (concept, cond-no, par-no, par-role)

Fig. 10: Relational schema for the generic level

IND-CONCEPT (individual, concept, begin, end)
TRAFFIC-LIGHT/COLOUR (individual, filler, begin, end)

IND-CONCEPT

INDIVIDUAL	CONCEPT	BEGIN	END
signal1	traffic-light	t1	t∞
green	traffic-colour	t1	t∞
yellow	traffic-colour	t1	t∞
red	traffic-colour	t1	t∞

TRAFFIC-LIGHT/COLOUR

INDIVIDUAL	FILLER	BEGIN	END
signal1	green	t1	t5
signal1	yellow	t6	t9
signal1	red	t10	...

Fig. 11: Example of the relations in the individual object section for fig. 5a

Because of the different treatment of generic and individual data, two sets of operators are provided, generic operators and individual operators. Generic operators are part of either a Generic Definition Language (GDL), creating and updating generic data, and a Generic Evaluation Language (GEL), reading generic data. There are two reasons for the division into GDL and GEL. First, GDL operators have a tremendous impact on the individual data whereas GEL operators have not. The creation of a new role of a concept implies the creation of a relation on the individual level. The deletion of a role within a generic concept may cause the deletion of an entire relation within the individual level. Secondly, during the evaluation phase the EPEX system has only reading access to the generic data, that means it may use only the GEL operators. For related work in schema evolution see [Bane86].

The individual operators form the Individual Manipulation Language (IML). The IML is an SQL-like language, though it was necessary to add functions dealing with the multiple values of role fillers. Moreover, special predefined geometric and temporal conditions are provided that can be used in the where-clause. For example, the geometric condition "in-circle" checks for maximum distance of two 3-D-coordinates. IML operators use the generic data for access. For example, a retrieval operation "select all mobile concepts where locus [t1] = (x,y,z)" uses COMPLETE-HIERARCHY to determine the concepts involved (mobile and all its subconcepts), and the relation CONCEPT-ROLES to determine the affected role-relations. With this information, individual concepts are selected from IND-CONCEPT and the associated role fillers from the relation representing the role "locus" of the generic concept "mobile" (see fig. 5b).

To summarize, assertions, consistency conditions, and generalization information are stored as data in the database. The IML uses this information for retrieval operations. Information normally represented in a schema or data dictionary is no longer separated from the data in the database. This means that update operations for generic concepts (GDL) have to deal with massive consistency problems. Changes within generic concepts may be followed by changes of relations for generic and individual concepts. Because the generic level in EPEX is considered stable during the episode extraction process, the associated problems will affect performance only in a controlled way.

6. Conclusions

The project discussed in this paper carries us to the frontiers of image evaluation. In it, databases play a modest although vital support role. From the standpoint of database research, however, this is an application opening a Pandora's box of novel problems. Among them are blurring of the distinction between schema and database, new kinds of access characteristics that may require novel access techniques, large-scale acquisition of sensory data and their structuring according to the rules of the database, and a high degree of dynamic declaration of relations. Perhaps the toughest challenge is expected to be in the realm of performance. It is an open question whether one would have, in the long run, to resort to object-oriented DBMS [Ditt86].

The present solution is based on the relational DBMS ORACLE, not the least because one should study the problems just discussed in the context of existing database technology, but also because EPEX will be implemented in CommonLisp to which ORACLE may be attached relatively easily. The EPEX system is currently under construction. The data model has been implemented, the operators are still under construction. The traffic scene discussed in chapter 4 has been described within the data model. The outline of the rule base is available and the rule interpreter has been implemented. Integration of the different parts is about to commence, and the connection to a system for providing the traffic data in the form of GSD must be established in the near future. Large-scale testing of EPEX is planned for 1988. By that time, object-oriented DBMS should become available so that work will start to base EPEX on such a system. This should allow us to study whether object-oriented DBMS will indeed provide better performance for image evaluation applications.

Acknowledgement. The authors are grateful to M. Ikker for his contribution to the design of the traffic scene data model and its implementation in a relational system.

Literature

[Abar86] R.M. Abarbanel, M.D. Williams: "A Relational Representation for Knowledge Bases." Proc. 1th Intern. Conf. on Expert Data Base Systems, Charleston, April 1986

[Alle84] J.F. Allen: "Towards a General Theory of Action and Time." Artificial Intelligence 23 (1984), 123-154

[Bane86] J. Banerjee, H.-J. Kim, W. Kim, H.F. Korth: "Schema Evolution in Object-Oriented Persistent Databases." Proc. 6th Advanced Database Symposium, Aug. 1986, 23-31

[Benn84] W. Benn, B. Radig: "Retrieval of Relational Structures for Image Sequence Analysis." Proc. 10th Int. Conf. on Very Large Data Bases 1984, 533-536

[Börn87] S. Börner, R. Studer: "An Approach to Manage Large Inheritance Networks." IBM Germany, LILOG-Report No. 8, 1987

[Bour85] P. Boursier: "Image Data Bases: A Status Report." Proc. IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management 1985, 355-358

[Brac85] R.J. Brachman, J.G. Schmolze: "An Overview of the KL-ONE Representation System." Cognitive Science 9 (1985), 171-216

[Card84] L. Cardelli: "A Semantics of Multiple Inheritance." in: Semantics of Data Types, Lecture Notes in Computer Science 173, Springer Verlag 1984, 51-67

[Card86] L. Cardelli, P. Wegner: "On Understanding Types, Data Abstraction, and Polymorphism." Tech. Rep. No. CS-85-14, Brown University, 1986

[Chan81a] S.-K. Chang, T.L. Kunii: "Pictorial Data-Base Systems." Computer 14:11 (1981), 13-21

[Chan81b] N.S. Chang, K.S. Fu: "Picture Query Languages for Pictorial Data-Base Systems." Computer 14:11 (1981), 23-33

[Chan85a] S.-K. Chang: "Image Information Systems." Proc. IEEE 73 (1985), 754-764

[Chan85b] S.-K. Chang, E. Jungert, S. Levialdi, G. Tortora, T. Ichikawa: "An Image Processing Language with Icon-Assisted Navigation." IEEE Trans. Software Engineering, SE-11 (1985), 811-819

[Chen76] P.P. Chen: "The Entity-Relationship Model: Toward a Unified View of Data." ACM Trans. Database Sys. 1 (1976), 9-36

[Codd79] E.F. Codd: "Extending the Relational Model to Capture More Meaning." ACM Trans. Database Sys. 4 (1979), 397-434

[Ditt86] K.R. Dittrich, U. Dayal (eds.): "Proc. 1986 International Workshop on Object-Oriented Database Systems." IEEE Computer Society Press, 1986

[Fill68] C. Fillmore: "The case for case." In: E.Bach, R.Harms (eds): Universals in Linguistic Theory. Holt, Rinehart and Winston, New York, 1968, 1-88

[Gutt84] A. Guttman: "R-Trees: A Dynamic Indexing Structure for Spatial Searching." Proc. ACM SIGMOD Conf. 1984, 47-57

[Hask81] R.L. Haskin, R.A. Lorie: "On Extending the Functions of a Relational Database System." IBM Res. Rep. RJ 3182, 11/81

[Hwan83] K. Hwang, K.S. Fu: "Integrated Computer Architectures for Image Processing and Database Management." IEEE Computer, Jan. 1983, 51-60

[Ikke87] M. Ikker: "Modellierung einer Datenbankschnittstelle für die Bildfolgenauswertung (Modelling of a Data Base Interface for Image Sequence Analysis)." Diploma Thesis, Fak. Informatik, Univ. Karlsruhe, 1987 (in German)

[Klop83] M.R. Klopprogge, P.C. Lockemann: "Modelling Information Preserving Databases: Consequences of the Concept of Time." Proc. 9th Int. Conf. on Very Large Data Bases, 1983, 399-416

[Levi87] P. Levi, J. Majundar: "Model-Based Robot Vision by Matching Scene Descriptions with Object Models from a CAD Modeller." Proc. 1987 Intern. Conference on Advanced Robotics (to be published)

[Mart80] R. Marti, J. Rebsamen, B. Thurnherr: "Meta Data Base Design - Consistent Description of a Database Management System." ETH Zürich, Institut für Informatik, Rep. 34, 1980

[Nage83] H.-H. Nagel: "Overview on Image Sequence Analysis." In: T.S.Huang (ed.): Image Sequence Processing and Dynamic Scene Analysis. NATO Advanced Study Institute Series 2, Springer-Verlag 1983, 2-39

[Nage85a] H.-H. Nagel: "Analyse und Interpretation von Bildfolgen (Analysis and Interpretation of Image Sequences) II." Informatik-Spektrum 8:6 (1985), 312-327 (in German)

[Nage85b] H.-H.Nagel: "Wissensgestützte Ansätze beim maschinellen Sehen: Helfen sie in der Praxis? (Knowledge-Based Approaches to Machine Vision: Do They Help in Practice?)." Informatik-Fachbericht 112, Springer-Verlag 1985, 170-198 (in German)

[Neum84] B. Neumann: "Natural Language Description of Time-Varying Scenes." Interner Bericht 105, Fachbereich Informatik, Universität Hamburg, 1984

[Neum86] B. Neumann, H.-J. Novak: "NAOS: Ein System zur natürlichsprachlichen Beschreibung zeitveränderlicher Szenen." Informatik Forschung und Entwicklung 1 (1986) 83-92 (in German)

[Niem85] H. Niemann, H. Bunke, I. Hofmann, G. Sagerer, F. Wolf, H. Feistel: "A Knowledge Based System for Analysis of Gated Blood Pool Studies." IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-7 (1985), 246-259

[Reim86] U. Reimer, H..-J. Schek: "A Frame Representation Model and the Mapping to NF^2 Relations." Working Paper, TH Darmstadt, 1986

[Scha75] R.C. Schank: "Conceptual Information Processing." North-Holland, Amsterdam, 1975

[Scha77] R.C. Schank. R. Abelson: "Scripts, Plans, Goals and Understanding." Lawrence Erlbaum, Hillsdale, N.J., 1977

[Smit77] J.M. Smith, D.C.P. Smith: "Database Abstractions: Aggregation and Generalization." ACM Trans. Database Sys. 2 (1977), 105-133

[Tamm82] M. Tamminen: "Efficient Spatial Access to a Data Base." Proc. ACM SIGMOD Conf. 1982, 200-206

[Tamu84] H. Tamura, N. Yokoya: "Image Database Systems: A Survey." Pattern Recognition 17 (1984), 29-43

[Walt86] I. Walter, P.C. Lockemann, H.-H. Nagel: "Ein Datenmodellentwurf für die Extraktion von Episoden aus Bildfolgen (Data Model Design for the Extraction of Episodes from Image Sequences)." Int. Rep. 3/86, Fak. Informatik, Univ. Karlsruhe, 1986 (in German)

[Wood70] W.A. Woods: "Transition Network Grammars for Natural Language Analysis." Comm. ACM 13:10 (1970), 591-606