Towards Concept-Oriented Databases

Werner Dubitzky, Alex G. Büchner, John G. Hughes, David A. Bell

University of Ulster, at Jordanstown,
Faculty of Informatics,
Co. Antrim BT37 0QB, Northern Ireland
{w.dubitzky, ag.buchner, jg.hughes, da.bell}@ulst.ac.uk

Abstract

Case-based reasoning (CBR) systems define knowledge in terms of a memory or library of past cases and a retrieval mechanism that revolves around retrieving data *relevant* to a goal query. Additionally, such systems employ an adaptation component that transforms the retrieved data into a solution to the problem expressed by the original query. The combination of goal query and the subsequent solution transformation is referred to as *CBR goal query*. Goal queries are concerned with data that is *close* to the request expressed in the query. Conventional relational and object-oriented databases are usually concerned specific queries. Extending conventional object-oriented data models, this paper proposes a *concept-oriented data model* that provides a variety of mechanisms to support conventional goal and CBR goal queries. It is shown that such a concept-oriented data model could be used as the core for a more general knowledge base management system.

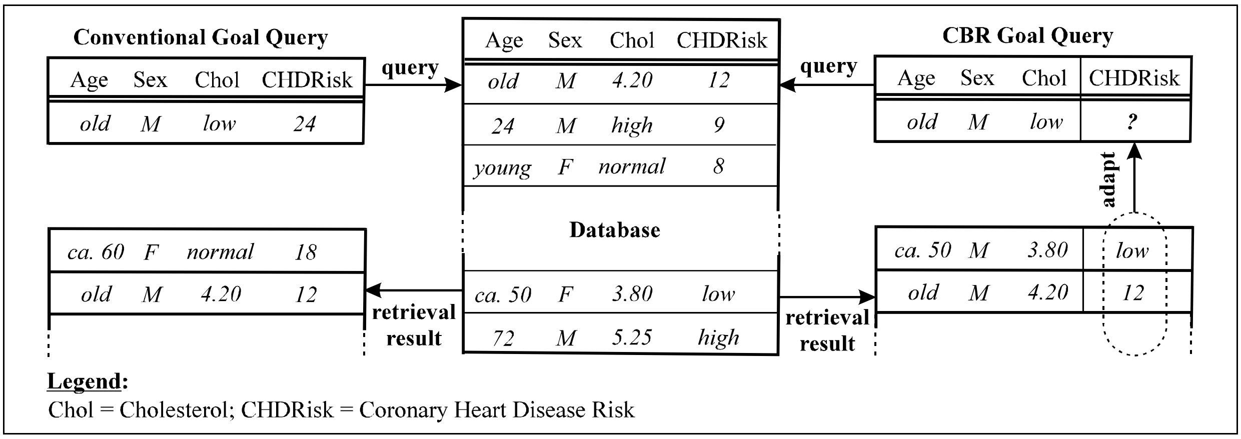
Keywords : data model, object-oriented databases, case-based reasoning, knowledge representation, reasoning under uncertainty

1 Introduction

Requests for data can be classified roughly into two kinds: *specific requests* and *goal requests* or goal queries [1,2]. A specific query establishes a rigid qualification, such a query is concerned only with data that matches it precisely. For example, “What is the income of John?”, “When does flight 0815 depart?”. If the database does not contain any data on John’s income or the departure time of flight 0815, a *nil* value should be returned. The user who issued the query is not interested in the income of somebody else or the departure time of a different flight. The specific query mechanism relies on a query language that is based on two-valued logic, which allows the implementation of efficient query processing strategies [2]. A goal query, on the other hand, describes a target that is concerned with data that is *close* or *similar* to that query. For example, "List *high* cholesterol patients that have a *low* coronary heart disease risk". If there are none, patients with medium cholesterol values and low coronary heart disease risk may also be of interest to the user who issued the query. Typical goal queries contain intrinsically imprecise predicates such as *high* cholesterol, *low* risk, *around 24* years, and so on. Such predicates represent sets whose boundaries are gradual or fuzzy rather than "sharp". It is difficult and impractical for query languages based on a two-valued logic to support goal queries of this kind. Since fuzzy goal queries frequently arise in engineering, business, medical, and other applications [3], various fuzzy relational database models and query languages have been investigated by the research community [4,5].

Case-based reasoning (CBR) or *exemplar-based reasoning* is a problem-solving or decision-making model that is concerned with solving new problems by *adapting* solutions that worked for *similar* problems in the past [2,6,7]. A CBR system could be viewed as an intelligent query and answering system or database that extends the notion of goal queries. In addition to relying on data that are close to the query specification, CBR systems also allow *new* information — that is, information that is not explicitly stored in the database — to be *inferred*. Simplified examples of typical CBR goal queries are: Query: "What heart disease risk score has a 42-year-old male patient whose cholesterol is very high?", Answer: "4.95", or Query: "Should a middle-aged applicant with an annual income of 29,000 be granted a 20 year fixed-rate mortgage?" Answer: "No".

At the query-definition level, the difference between CBR goal queries and a conventional goal queries is that CBR goal queries are composed of two parts, namely a (query/problem) *description* part, and a *solution* part. The description specifies the data that describes the problem at hand, and the solution specifies some unknown aspect or property of the current problem. At the query-processing level, a system must not only determine the data items that are similar to the description contained in the CBR query, but also generate an answer for the requested solution based on the retrieved items. To do so, a database that supports CBR goal queries must be equipped with a set of *solution transformation* or *solution adaptation* rules. The diagram in Figure 1 illustrates the difference between standard goal queries and CBR goal queries. Informally, the queries depicted in the diagram could be stated as follows: "List all old, male, low-cholesterol patients whose heart disease risk is 24." (conventional goal query), and "What is the heart disease risk of an old, male, low-cholesterol patient?" (CBR goal query).



Conventional Goal Query

Age	Sex	Chol	CHDRisk
old	*M*	*low*	*24*

Database

Age	Sex	Chol	CHDRisk
old	*M*	*4.20*	*12*
24	*M*	*high*	*9*
young	*F*	*normal*	*8*
ca. 50	*F*	*3.80*	*low*
72	*M*	*5.25*	*high*

CBR Goal Query

Age	Sex	Chol	CHDRisk
old	*M*	*low*	*?*

ca. 60	*F*	*normal*	*18*
old	*M*	*4.20*	*12*

ca. 50	*M*	*3.80*	*low*
old	*M*	*4.20*	*12*

Legend:
Chol = Cholesterol; CHDRisk = Coronary Heart Disease Risk

Figure 1: Conventional and CBR goal query.

As illustrated in Figure 1, the retrieval results for both types of goal queries is a list of data items closest to the query specification. However, for the CBR query, the list of closest items is determined based solely on the data contained in the description part of the query. The final answer to the solution part of the CBR goal query (depicted by the "?" symbol) is established from the solution parts of the closest retrieved items by means of a set of adaptation rules. Note, the diagram in Figure 1 also illustrates a feature that will be discussed in the paper, namely, *polymorphic attributes*. Polymorphic attributes allow instances of attributes to take on precise numeric values (e.g., Age = *72* years) as well as intrinsically imprecise values (e.g., Age = *old* or Age = *ca. 50* years).

Recently, there has been an increased interest in database research that is concerned with the integration of knowledge management, reasoning, and goal query capabilities into database systems [8,9,3,5,1]. However, the bulk of research in the field of conventional relational and object-oriented databases has been focusing on data and query models that revolve around specific data requests. It is therefore not surprising that the key modelling constructs in those models (i.e., relations and object classes) are based on the *classic concept model* [10].

CBR research, on the other hand, has been concentrated on investigating representation and query models that directly support the goal queries involved in problem solving, decision making, and other high-level reasoning tasks [2,11,12]. The typical core CBR process consists of three sub-processes:

1. *Retrieve* from a repository of past cases, called case base, those cases that are most *relevant* to the problem at hand (problem description and sought-for solution). Relevance, in CBR, is usually approximated by some measure of similarity (computational approach) or an appropriate indexing scheme (representational approach).
2. *Adapt* the solutions stored with the retrieved cases to the new problem and derive a solution for the new problem (solution transformation).
3. *Retain* the new problem-solving episode in the case base for future use (learning).

The first sub-process in the CBR process, namely the retrieval of relevant data, is arguably the most crucial one, as all other steps depend on it. It is not difficult to see that this retrieval sub-process is very similar to processing a conventional goal query (see above) in the database world. Many similarity-based and indexing-based models for effective and efficient case retrieval have been proposed in the CBR literature. For example, similarity measures that work on conventional crisp [13], fuzzy and vague [14,15], mixed crisp/fuzzy [16], and graph-structured [17] data or case representations, and indexing approaches such as the *memory organisation package* model [18], *generalised episode* or *context-plus-index* model [19], and the *concept-exemplar* model [20]. Apart from efficiency issues, the objective of both similarity-based and indexing-based retrieval models is to impose a *conceptual* organisation on the case base in order to ensure effective retrieval. So CBR system and database design are similar in the way that the system designer attempts to model the underlying domain concepts using higher-level modelling constructs such as relations or object classes (databases), and similarity or indexing models (CBR). The resulting conceptual structures and definitions form the basic framework to organise the underlying data and support effective retrieval of the data.

The work presented in this paper takes a fresh look at *concept modelling* in the light of conventional and CBR goal queries. Extending conventional object-oriented data models, it proposes a *concept-oriented data model* that facilitates CBR goal queries. Thus, concept-oriented database management systems could form the core of a general knowledge base management system capable of processing CBR goal queries. The (query) system architecture of such a system is illustrated in Figure 2.

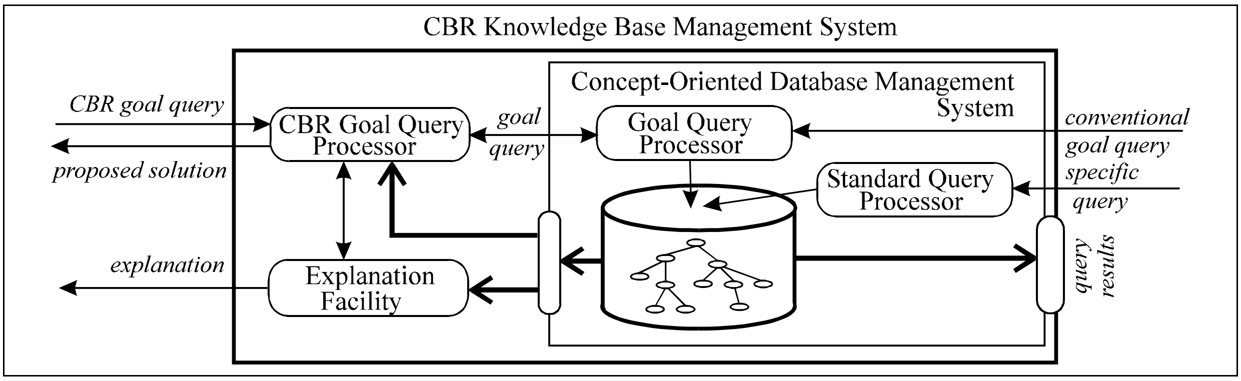


Figure 2: Basic query architecture of CBR Knowledge Base Management System.

It should be apparent that a key aspect of the proposed concept-oriented data model is the notion of a *concept*, and how concepts — and a data model based on concepts — could facilitate CBR goal queries. Also of interest is how the proposed concept notion differs from conventional concept models realised in traditional relational and object-oriented databases. The main part of the discussion below addresses the issues raised above.

Various data/knowledge models have been proposed which allow the handling of probabilistic and possibilistic uncertainty in some way or another. For example, multiple attribute decision making system models [14] (relevance weights and fuzzy sets), relational data models [4,3,5,21] (numeric intervals, fuzzy sets, user-provided probability estimates, rough sets, Dempster-Shafer theory), object-oriented information models [22] (fuzzy sets), and [23] (fuzzy sets, Dempster-Shafer theory). Common to these approaches is that a particular attribute may only take values of a single type (symbolic or numeric); attributes that permit symbolic values are usually associated with a numeric interval, a fuzzy set, or a similarity matrix. The approach presented in this paper is different in that it provides an integrated framework for representing and processing (*similarity* and *typicality*) precise numeric and imprecise probabilistic/possibilistic data. Another difference is that in the presented model goal requests are processed with respect to concept membership, and that this query processing mechanism is tightly integrated with the description of the data (schema) itself, rather than as a global procedure. Finally, the concept-oriented data model presented in this paper defines an explicit, systematic model for representing contextual effects on data. This concept is not treated in any form by the approaches discussed above.

The remainder of the paper is organised as follows. Section 2 discusses the role of concepts for CBR. Section 3 and its sub-sections describe in some detail the three most commonly known concept models, and how the models relate to the various aspects of the envisaged goal query-capable concept-oriented data model. In Section 4, an extension to the *prototype concept model* is proposed as the conceptual foundation for the proposed concept-oriented data model. Based on the findings in Section 4, Section 5 sets forth the description of the concept-oriented data model. Section 6 discusses issues concerning a knowledge base management framework based on concept-oriented databases. The same section summarises some of the results that have been obtained through experiments with various components of the proposed model. Section 7 concludes the paper with some critical observations. Throughout the paper, an example from the medical domain for illustration purposes.

2 Concepts and Case-Based Reasoning

Pure similarity-based case base organisation and retrieval schemes model concepts *implicitly* via the pair (C,*sim*), such that $sim(x,y)$: $U \times C \rightarrow [0,1]$, $C \subset U$, and $x, y \in U$, where U is the *universe* of all cases and C reflects the cases stored in the case base. Such schemes normally evaluate all cases in the case base sequentially. The main disadvantages of this approach are [24]:

- for a large number of cases and case features the computational costs for retrieval may be too high;
- normally, ancillary knowledge (e.g., adaptation knowledge, uncertainty management knowledge) should be stored with groups of cases to which it applies; the pure similarity-based approach makes this difficult;

- to define a useful similarity measure which is globally applicable constitutes a significant knowledge engineering task;
- it is difficult to incorporate context effects in this approach; and
- case base maintenance and user acceptance is not easy in purely similarity-based CBR systems.

Organising the cases stored in a case base around an *explicit* model of concepts, like *memory organisation packages* [30] or *generalised episode* models [22], can eliminate or alleviate the shortcomings (see above) associated with the similarity-based approach. In particular, it can significantly improve retrieval efficiency, and provide a framework for consistently incorporating other types knowledge (adaptation, uncertainty, management, deep domain knowledge, and so on) into the CBR system. Such a concept approach can be viewed as kind of indexing model. The drawback of such an approach is that it requires a substantial amount of knowledge acquisition and engineering. Coping with uncertainty (i.e., incomplete and imprecise information) is arguably the key problem encountered in any knowledge acquisition exercise. A considerable part of the discussion in this paper is devoted to the issue of modelling uncertainty.

The case base representation model put forth in this paper revolves around the representation and organisation of cases around concepts. A concept or class captures the notion that many objects or events are alike in some important respects, and can therefore be thought of in the same or similar ways [10,25]. Concepts and the relationships between concepts constitute very expressive modelling constructs for capturing *deep*, *general* domain knowledge, as opposed to shallow, general (e.g., rules) or shallow, specific (e.g., cases) associational knowledge [26]. For an intelligent goal query retrieval mechanism that is to achieve accurate and useful results in information-intensive application areas (e.g., medical diagnosis, investment planning), a tight *integration* of deep, general domain knowledge is desirable [1,27]. Therefore, by placing the concept notion in the centre of a goal query-capable model, a tight coupling of the data or cases with deep, general domain knowledge is guaranteed. Furthermore, through the concept-recognition function (see Section 3) that is inherent in complete concept models, an adequate retrieval (and storage) framework is provided, which is closely connected to deep, general domain knowledge reflected by concepts.

3 Concepts and Concept Properties

How people acquire and process concepts has been, and still is, researched quite intensively [10,25]. These studies revolve around the *structure* and *function* of concepts, and how concepts could be modelled. The three major concept "views" or models proposed in the literature are: the *classical model*, the *prototype model*, and the *exemplar model*. The prototype and exemplar models are meant to address the problems found in the more and more frequently criticised classical model, which dates back to Aristotle. In the classical model it is held that all instances of a concept share common properties, and these common properties are *necessary* and *sufficient* to define the concept. Virtually all current object-oriented and relational data models realise the classical concept model or a subset thereof. The prototype model, on the other hand, assumes that instances of a concept vary in the degree to which they share certain properties, and as a result vary in the degree to which they represent the concept. Finally, the exemplar model, which constitutes a more extreme departure from the classical model, holds that there is no single representation of an entire concept or class, but only specific representations of the concept's exemplars or instances.

Intensional concept *summaries* and extensional concept *exemplars* are characterised by *properties* (or *attributes*). Properties can be divided into *holistic* and *component* types [28,10]. The former kind of property is somewhat restricted to certain types of applications, it is therefore not further considered in this work. A component property is one that helps to describe a concept but does not usually constitute a complete description of the concept. Component properties describe *partial* or *local* and (abstracted) *global* aspects of a concept. Usually, two types of component properties are distinguished: *featural properties* and *dimensional* properties. Featural properties are sometimes called qualitative or symbolic properties, and dimensional properties are sometimes referred to as quantitative or numeric properties.

Although it is debated which of the concept models one should endorse, there exists some agreement on the function of concepts. Concepts provide a *taxonomy* of entities and events in the world, and concepts are used to express *relations* between classes in that taxonomy.

One sub–function of the taxonomic function has received much attention in the literature, namely, the *categorisation* function. A concept's categorisation function is concerned with the concept's role as a pattern–recognition device. This means that concepts are used to classify or categorise novel entities and draw inferences about such entities. To possess a concept, *C*, is to know something about the properties of the instances of *C*, this knowledge can be used to categorise novel entities. Once an object, *x*, is recognised as a member of a concept, *C*, the knowledge about the properties of *x* serves as basis to infer certain, not directly perceptible attributes of *x*. This inference process can be considered as a type of *reasoning* [26].

Because traditional relational and object-oriented data models are designed to support specific queries and, possibly, conventional goal queries, there is no such thing as a categorisation function of a relation (relational data model) or a class (object-oriented data model). For the concept-oriented data model proposed in this work the categorisation function is of particular interest, as it constitutes the key mechanism for processing CBR goal queries.

3.1 The three Concept Models

The three main concept models that are commonly studied and referenced in the literature are the classical, the prototype, and the exemplar model. This section and its sub-sections review the main properties of these models in the context of CBR goal queries and concept-oriented extensions to data models.

3.1.1 The Classical Concept Model

In the classical view of concepts it is held that the representation of a concept is a *summary* description of the entire class, rather than descriptions of various exemplars or subordinates of that class. This approach of condensing a concept into a single summary representation has the advantage that little information needs to be stored. Two additional fundamental assumptions underlying the classical concept model hold (1) that the properties that represent a concept are singly *necessary*, and (2) that jointly *sufficient* to define that concept.

Necessary and sufficient properties are sometimes referred to as *defining* properties. A singly necessary property requires that each exemplar of that concept *must* have this property. To illustrate, every legal chess position must involve exactly two kings (a white and a black king). A set of jointly sufficient properties demands that every object having that set must be an instance of the concept in question.

The classical model has been heavily criticised throughout its existence. Two of its most severe defects are (1) its inability to handle abstract properties, and, more importantly (2) the fact that for many concepts natural, scientific, artificial, and ontological concepts a commonly agreed set of defining properties cannot be found. Take, for example, the concept of a *Game*. What is a necessary property for this concept? It cannot be *competition between teams* or even *competition between two individuals*, for there are games that can be played by a single person, e.g., solitaire.

By insisting on defining properties, the classical models deny the fact that in most but a few cases the *relationships* between a concept and its properties are characterised by some degree of uncertainty [12,7]. Instead, for example, to define the *Game* concept by the necessary property *competition between teams*, it is more accurate and epistemological adequate to replace this property by something like *usually a competition between teams*. Traditional relational and object-oriented data models describe a concept, *C*, via attributes that are singly *necessary* and jointly *sufficient* to define *C*. For the purpose of adequately modelling many domain knowledge concepts (i.e., scientific, ontological, artificial, and natural concepts such as *Angina*, *Disease*, *Risk*, etc.) such models are of little use. Therefore, data models supporting goal queries and CBR goal queries based on domain knowledge representations should not be based on the classical concept model. What is needed is an expressive concept model that is more adequate for modelling domain knowledge.

3.1.2 The Prototype Concept Model

Paralleling the classical model, the first assumption of the prototype concept model holds that a concept is represented by an intensional summary description of the entire class. But, in contrast to the classical model, the second assumption of the prototype view holds that the representation of a concept *cannot* be restricted to a set of necessary and sufficient properties. Rather, the properties that represent a concept are the *salient* or *important* ones that have a significant *probability* of occurring in exemplars of the concept. The importance of a property and its probability of occurring in concept exemplars is explicitly reflected in the concept representation by a *conditional probability*. For dimensional properties (see Section 3.2.2) this conditional probability also reflects the importance of *value variations* in the associated dimension. Furthermore, in contrast to featural properties (see Section 3.2.1), which describe *modal* characteristics, dimensional properties define a *mean value* which expresses the *tendency* of values in the corresponding dimension. Both, the conditional probability and the *mean value*, play a crucial role for the categorisation function of concepts (see below). Table 1 illustrates how featural and dimensional properties are used in conjunction with conditional probabilities and mean values to define the *Game* concept.

Table 1: Prototype representation of the *Game* concept.

Property Name	**Conditional Probability**	**Mean Value**	**Property Type**
name	1.00	(n/a)	character string
has winner	0.90	(n/a)	featural
competition between teams	0.80	(n/a)	featural
duration	0.50	70 [minutes]	dimensional

Clearly, the conditional probability reflects the uncertainty associated with the relationship between a concept and the corresponding property. The conditional probability is a crucial construct when it comes to classifying a new entity.

The categorisation procedure of the *classical* model requires that an entity, x, to be recognised as an exemplar of a concept, C, contains values for *all* defining properties defined in the summary representation of C. Due to the fact that the *prototype* model allows for non–necessary properties (explicitly represented by conditional probabilities), its processing requirements for the classification function differ from that of the classical view. In the prototype model, an entity, x, is categorised as an instance of a concept, C, if, and only if, x possesses some critical sum, $m_C(x)$, of the “weighted” properties of C. This means that the prototype scheme, as opposed to the classical model, permits exemplars to be *partial* members of a concept (with $m_C(x)$ expressing the membership degree)! This notion of partial or gradual membership of occurrences or objects is not reflected in conventional relational and object-oriented data models. To a certain extent this is understandable, as these models do usually not support a relation- or a class-recognition function.

Revolving around the notion of conditional probability, modality, and central tendency (mean value), the prototype concept model sacrifices some economy of the classical model, however, it suggests an increased richness in abstracting information. More importantly, it offers an answer to the single most important failure of the classical view, namely the lack of progress in specifying defining properties for many natural, artificial, ontological, and scientific concepts. Because the prototype view does not require necessary and sufficient properties, this issue is immaterial.

3.1.3 The Exemplar Concept Model

In the exemplar model of concepts, the stand is taken that concepts are represented, at least in part, by some of their exemplars rather than by an abstract summary. Supported by experimental evidence, the critical claim in this view is that exemplars usually play the dominant role in categorisation because they are more accessible than the summary information [10]. The essential assumption in exemplar models is that exemplars used to explicitly represent a concept are *those* that share a critical number of properties with other exemplars of the concept. The concept-recognition function in exemplar models compares a new entity, x, to critical members of exemplars, E, used to model the concept, C. The entity x is considered a member of C if, and only if, it is sufficiently similar to a sufficient number of exemplars in E (the parameters associated with “sufficiently” are usually defined a priori). In contrast to the prototype model, which addresses the necessary-features and sufficient-features problems of the classical model via conditional probabilities represented at the concept summary level, the exemplar uses a highly disjunctive, multiple-exemplar representation for the same purpose. Since the exemplar representation of a concept, C, does not require that a property of one exemplar should be a property of other exemplars of C, the properties used to describe C need *not* be *necessary* ones. Furthermore, as the exemplar-based concept representations are highly disjunctive, there is *no* need for *sufficient* properties. The main criticism of the exemplar view is that it leaves little room for abstractions. Because its lack of capturing abstractions such model has the tendency to make every single instance an explicit part of the concept representation. This seems highly implausible and counter-intuitive, for such models would impose an inefficient processing and storage regime on the concept memory.

Usually, database systems make a clear distinction between meta-data level (schema) and actual data (occurrences or objects). The information contained in the schema reflects the concepts covered, while occurrences or objects are considered as members or instances of these concepts. Similarity-based CBR, on the other hand, does not explicitly define concepts in an a priori manner, concepts in such a system are defined implicitly by a pair of the form $(casebase, sim)$ (see Section 2). In that, similarity-based CBR is similar to exemplar-based concept models. In the light of a new problem or query, x, the retrieval mechanism (i.e., concept-recognition function) of a CBR system usually determines a set, $C \subset casebase$, of candidate cases similar to x. The set of retrieved cases, C, is interpreted as a (exemplar-based) concept of which with x is a member. Recall (Section 2), that possessing knowledge about a concept, C, and knowing that an entity, x, is a member of that concept, allows us to draw inferences or reason about x. Therefore, the exemplar concept model is relevant to CBR and hence to this discussion. Although the proposed concept-oriented data model (see below) essentially realises a prototype concept model, it is also capable of processing pure similarity-based queries. In that, it can be seen as incorporating some aspects of the exemplar concept model.

3.2 Concept Properties

Within a concept taxonomy, properties are used to *model* concepts, and to make apparent *relationships* between concepts [10]. Recall, a concept taxonomy is able to capture deep and general knowledge within a domain [12] (see Section 1).

Properties are employed to describe partial and global aspects of concepts. For example, *age* and *means of transport* are global properties, which describe the concept of a *Car* irrespective of other properties. Partial properties, on the other hand, refer to constituent parts of the entity in question, examples include *has engine*, *has 4 doors*, *has boot*, and so on.

Besides their conceptual value, so-called *shared* properties permit to represent a concept taxonomy *economically*. That is, shared properties correspond to the notions of reuse and inheritance in object-oriented data models. Of course, from the perspective of the concept-recognition function, a property should not be too general, that is, apply to too many concepts in a conceptual domain or taxonomy, for then it would be of little discriminatory value. Similarly, a too specific property barely contributes to conceptual richness and economy.

A concept may contain *abstract* or *functional* properties as well as atomic or perceptible (featural, dimensional) ones. Although this notion of abstract properties is important, this issue is not discussed in great detail here as it would go beyond the scope of this paper. Basically, abstract properties are composites or aggregates, which are formed from atomic properties (for more detail see [10,29]).

3.2.1 Featural Properties

One way to characterise the component properties of a concept is by *qualitative* or *featural* properties, see, for example, the *Game* properties *has winner* and *competition between teams* in Table 1. Featural properties, then, capture *qualitative* variations of concepts; two concepts differ with respect to a feature, *f*, if one concept is characterised by *f* and the other is not. Featural properties are similar to symbolic or nominal attributes in conventional data models. A major advantage of a feature-based representation regimes is their simplicity. An obvious shortcoming of the featural approach is that it cannot properly explain and handle quantitative variations of properties. Humans, on the other hand, often have a fairly accurate idea about such variations [25].

3.2.2 Dimensional Properties

Another way to represent concept properties is by *quantitative* or *dimensional* properties. The essential assumption of the dimensional approach is that concepts that have the same relevant dimensions can be represented as points in a multidimensional *metric space* (which has the desirable properties *minimality*, *symmetry*, and *triangular inequality*) [30]. Dimensional properties capture quantitative variations of concepts. For two concepts to differ with respect to a particular dimension, one concept must score a higher value on that dimension than the other. Within a prototype concept representation scheme, featural properties as well as dimensional properties carry a conditional probability value to reflect the property’s conditional probability of occurring in exemplars of the concept. Unlike featural properties, the conditional probability of a property can also be interpreted as reflecting the relevance of variations in the corresponding dimension. (Table 1 illustrates an example of a dimensional property, *duration*, used to define the *Game* concept.)

In contrast to featural properties, which model modality, dimensional properties are represented at the summary description level of a concept by means of a *mean value*. The mean value expresses the central tendency of the property's values along the corresponding dimension within the context of the encompassing concept. This mean value concept is used by the classification function in order to establish the *typicality* of property values. Two basic categorisation methods exist in conjunction with dimensional properties; the one that is of relevance to this work is stated in Definition 1.

Definition 1 To recognise an entity, *x*, as a member of a concept, *C*, the similarity, *sim*(*C*,*x*), between *C* and an *x* must be equal or greater than some concept-specific, predefined threshold, t_C. The similarity, *sim*(*C*,*x*), is computed from the "weighted" or accumulated-probability similarity (conditional probabilities) as follows:

$$sim(C,x) = \left(\sum_{i=1}^{|x|} w_i \;\; sim(m_i\,, x_i) \right) \Big/ \sum_{i=1}^{|x|} w_i \qquad (\ 1\ )$$

In equation (1), the symbol *i* represents an index to the *i*th dimensional property (type) in *x*, such that *i* = 1, 2, …, |*x*|, and x_i denotes the corresponding exemplar or instance value of the property indexed by *i*. Further, the terms w_i and m_i define the concept-specific conditional probability and concept-specific mean value of the property (type) indexed by *i*, such that w_i, $m_i \in [0,1]$. A particular concept, *C*, may only contain *some* of the properties in its summary representation that also occur in the query, *x*. In such cases, the data corresponding to index *i* in equation (1) is not evaluated, but the data corresponding to the next index, if any, is. Note, the term in the denominator serves to normalise the overall result.

Evidently, the advantage of the dimensional approach relates to the ability of dimensional properties to capture quantitative variations. A drawback of dimensional properties is that they do not support explicit relationships between properties such as the aggregation relation. Furthermore, although dimensions can be used to “simulate” qualitative properties like *has winner*, they, similar to their featural counterparts, cannot capture the possibilistic uncertainty inherent in *imprecise quantities* such as *approximately 200, around 50*, or *medium height*. Also, similar to the featural properties approach, dimensional properties are not able to represent context effects, for example, constraining the values a dimension may assume in a particular context.

4 Extending the Conventional Prototype Model

The classical concept model discussed in Section 3.1.1 insists on a set of necessary and jointly sufficient properties to describe a concept. Although this model is realised by most relational and object-oriented databases, this view of concepts is highly inadequate for modelling many ontological, scientific, artificial, and natural concepts. Successfully addressing the necessary-sufficient property dilemma of the classical view, the exemplar-based concept view suffers from problems that arise from the fact that it leaves little room for abstractions. Compared with the classical and exemplar concept models, the prototype view does not seem to be plagued by any of these major conceptual difficulties. The prototype concept model is therefore chosen as the conceptual foundation for the proposed concept-oriented data model. However, there are three aspects of the prototype concept model (as presented in Section 3.1.2) that could be improved: these are:

1) the model’s incoherent *dualistic property* model manifest in the feature-dimension dichotomy;
2) its inability of handling *possibilistic uncertainty* systematically; possibilistic uncertainty arises from incomplete or imprecise information manifest in featural properties; and
3) its lack of explicitly representing contextual effects or constraints associated with properties.

4.1 Reconciling Featural and Dimensional Properties

In principle the mechanisms and semantics of featural properties could be “simulated” by dimensional properties and dimensional properties could be “simulated” by featural properties [10]. However, these approaches are ad hoc and inefficient, they do not provide a coherent framework which integrates the two types of properties. This work proposes a *property concept frame* which — based on *fuzzy set theory* — provides a systematic framework for unifying featural (symbolic) with dimensional (numeric) properties. Not only is it possible to express three types of property values on such a concept frame, but it is also possible to compare different value types. This aspect is crucial for the proposed concept model, and is not found with related work, such as [31,32,14,15], where properties can either assume fuzzy or crisp values. Furthermore, property concept frames serve as a crucial platform extending the conventional prototype model of concepts to capture possibilistic uncertainty at the level of the concept summary definition level. This, to the best of our knowledge, constitutes a novel way of using fuzzy set theory for modelling *structured* concepts, and integrating incomplete knowledge in CBR. Finally, the concept frame mechanism also provides a platform to modelling concept-dependent semantics of properties, in other words, context effects. [33] Some aspects of property concept frames that relate to the general issue of unifying symbolic (featural) with numeric (dimensional) properties have been reported in [16]; for the sake of self-containment these are summarised in Section 4.1.1. In order to model context effects and the concept-recognition function (i.e., goal queries) within a concept-oriented data model two important extensions have been introduced to this property concept frame model, these are discussed in Section 4.2 and Section 4.3.

4.1.1 Concept Frames

To equip a property (type) with the capability to represent and process instances of different form, i.e., symbolic and numeric, the property and its instances are associated with a property concept frame. Based on such a concept frame, it is possible to express property instances in three different formats, namely, as *numeric*, *symbolic*, and *fuzzy predicate* values respectively. These formats facilitate the specification of property values or instances at various levels of certainty and expertise. To illustrate this property concept frame notion and the three basic property value formats consider the *cholesterol* property concept frame depicted in Figure 3.

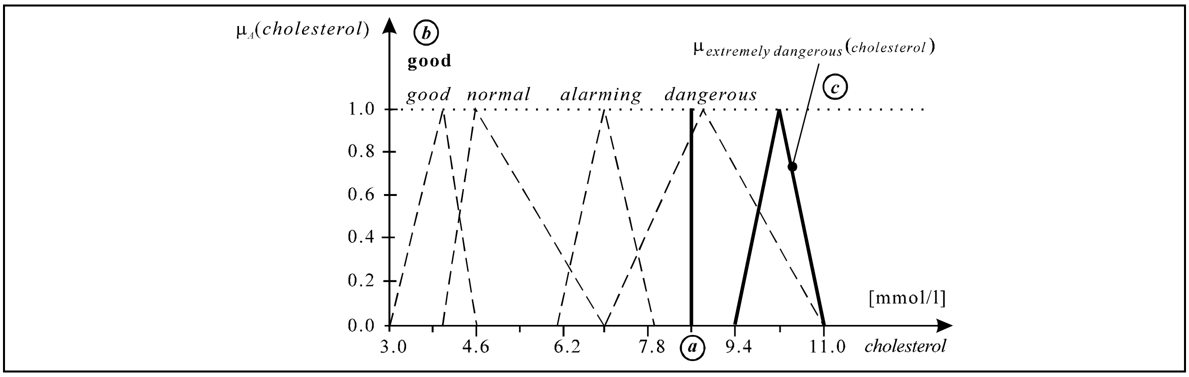


Figure 3: Property concept frame and three actual feature values.

A property concept frame defines a finite universe of discourse, U, which constrains the numeric values the associated property may assume. Henceforth, the universe of discourse, U, is always assumed to be a countable that is discrete universe. For example, in the cholesterol concept frame shown in Figure 3, the universe of discourse, U, is defined by the blood cholesterol concentration range [3.0,11.0] mmol/l. In addition to the universe of discourse, a property concept frame describes and a set of pre-defined conceptual categories that relate the underlying numeric domain (physical world) to their corresponding linguistic concepts (logic-based world). These concepts are represented via linguistic symbols and the corresponding membership functions that define fuzzy sets [31]. These membership functions are referred to as v-functions (v stands for value). For example, consider the linguistic term good and the associated (dashed-lined triangle below good) in Figure 3.

The linguistic concepts used to describe the *cholesterol* property concept frame are defined by the symbols *good*, *normal*, *alarming*, and *dangerous* respectively, and the corresponding v-functions (dashed triangles in Figure 3) $\mu_{good}(cholesterol)$, $\mu_{normal}(cholesterol)$, $\mu_{alarming}(cholesterol)$, and $\mu_{dangerous}(cholesterol)$.

By defining the linguistic labels for the underlying domain concepts and their meaning content (definition of the associated v-function), the designer of the property effectively formalises the knowledge held by the collaborating domain expert. It is important to understand that the concept frame components universe of discourse, linguistic labels defining domain concepts, and the associated v-functions constitute *design time* elements, which will remain constant at problem-solving time or run-time of the system. These components represent knowledge and are used at run-time to interpret data. Instance values of properties (see below), on the other hand, are created, used and, possibly, destroyed at run-time by system users.

Property instances or *values* are specified when the system is in operational mode, these values may be provided by expert or novice users. Besides the design-time concept frame components, Figure 3 also demonstrates three examples of actual *cholesterol* values specified at *run-time*. The three values, labelled *a*, *b*, and *c*, are depicted in the diagram as bold-style objects (vertical line, linguistic symbol, and triangular membership function). The value $a = \langle 8.60 \rangle$ represents a *numeric*-format instance of the cholesterol property, it describes a crisp *cholesterol* value or quantity. The value *b* is provided in *symbolic* format, $b = \langle good \rangle$, it describes a *cholesterol* value as imprecise quantity or linguistic concept. Note, symbolic-format property values must be drawn from the set of linguistic symbols that are predefined for a particular property (type), thus, they are always connected with the corresponding v-function. Finally, the value *c* is specified using the *fuzzy predicate* format, $c = \langle extremely\, dangerous, \mu_{extremely\ dangerous}(cholesterol) \rangle$, it describes a *cholesterol* value as imprecise quantity or linguistic concept, *and* explicitly defines the *meaning content* of the corresponding symbol.

Of these three formats, the numeric format is the most obvious one, it is employed where quantifiable data is readily available.

To specify a property value linguistically or symbolically, the specifier of the value must choose one of the linguistic symbols associated with the v-functions of the corresponding concept frame, e.g., $value = \langle dangerous \rangle$. At a first glance this may appear somewhat restrictive. However, the concepts defined on a concept frame are carefully established in accordance with the underlying application semantics and constraints. Therefore, this format is useful for inexperienced users in situations where precise data is not available.

Finally, the fuzzy predicate format is intended for users who have considerable experience with the corresponding property and the associated concept defined via that property. The fuzzy predicate format provides a flexible mechanism which permits the user to specify a value by introducing the name (linguistic symbol) and explicitly defining the semantical content (membership function of fuzzy predicate value) of a value, concept, or quantity.

A crucial aspect of this unified property representation framework is the computation of similarity (or semantic equality) between the various possible format combinations. To illustrate, consider the similarity *sim*(*dangerous*,8.60) between the *cholesterol* values $\langle$*dangerous*$\rangle$ and $\langle 8.60 \rangle$ mmol/l. The method for this computation is described in detail in [16], it is based on Chen and Hwang's *crisp score method* [14]. As a consequence of the fuzzy-set-based unified representation of symbolic and numeric property values, the *triangular equality* constraint or property associated with the universe, *U*, does not hold (see also [14]). That is, $\forall x,y,z$ $(x,y,z \in U \wedge (1 - sim(x,y)) \leq (1 - sim(y,z))$ implies not $(1 - sim(x,y)) \leq (1 - sim(x,z))$).

4.2 Concept-Recognition Function and Possibilistic Uncertainty

The concept frame notion outlined in Section 4.1.1 constitutes a flexible framework for representing and processing (instance-instance similarity) concept properties and their instances. But what about representing possibilistic uncertainty or knowledge at the summary description level of prototype concepts definitions? The basic prototype concept model discussed in Section 3.1.2 provides an explicit model to consistently represent probabilistic uncertainty arising from incomplete knowledge about the relationships between domain-specific concepts. A possibility distribution defined over a property's universe discourse, *U*, would define how possible it is for certain property values, precise or imprecise ones, to occur. Once within the confines of physical laws, the notion of possibility carries a highly subjective connotation. For example, a physician that has never encountered or heard of a cholesterol value of 12.5 mmol/l may still argue, based on his background knowledge, that such a value is possible, and define how possible such a value might be according to his opinion. So a possibility distribution defined over a range of property values expresses an expert's knowledge about the property in terms of constraints. If such knowledge can be provided by an expert, it should be explicitly reflected in the prototype representation of concepts.

As Smith and Medin point out in [10], the mean value concept used to reflect the central tendency of dimensional properties appears to be an ad hoc solution. In the following it is proposed to replace this mean value construct by a more adequate concept that captures the *possibility* of values that may occur for a given property. It is important to recall, that, in contrast to property *values*, the mean value mechanism forms part of the ***summary definition*** of the encompassing concept!

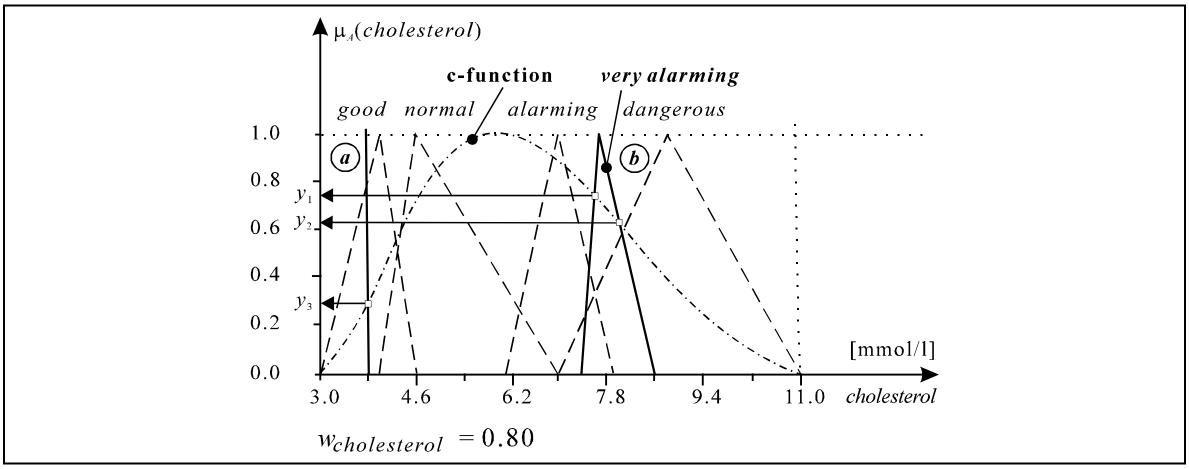


Figure 4: Central tendency and possible values (possibility distribution).

The diagram in Figure 4 illustrates this concept of a *possibility distribution* (dotted-dashed line) which is used in the proposed model to replace the ad hoc mean value mechanism of the conventional prototype concept model. The possible values of a property are modelled via a fuzzy set (possibility distribution) whose corresponding membership function is called *characterising function* or simply *c-function*. A key difference of this mechanism compared with the mean value approach is that the value tendency is not determined by an application-independent distance, such as the *Manhattan metric*, from the mean value. Instead, it is defined on the basis of an expert-defined possibility distribution that reflects the expert's knowledge about the underlying phenomena. Another important difference is that the characterising function, unlike the mean value, can be used in conjunction with both numeric and symbolic property values. This is not possible with standard classical and fuzzy numeric-symbolic schemes.

The bell-shaped, dashed-dotted line in Figure 4 illustrates the possibility distribution for values of the *cholesterol* property. Based on the c-function it is possible to determine the possibilistic *typicality* of *cholesterol* values. This is not only true for precise property values (numeric format) but also for imprecise values (symbolic, fuzzy predicate format). To illustrate this point, consider the numeric format value labelled *a*, and the fuzzy predicate format value labelled *b* in Figure 4. Given that the c-function is defined as $\mu_{ptyp}(\mathit{cholesterol})$, the possibilistic typicality, p_a of *a*, is determined by $p_a = \mu_{ptyp}(a) = y_3$, and the possibilistic typicality, p_b of *b*, is computed by the average membership degree as follows: $p_b = ½ (\mu_{ptyp}(b_{LEFT}) + \mu_{ptyp}((b_{RIGHT})) = ½ (y_1 + y_2)$. The c-function concept and the associated measures of typicality for crisp and imprecise property values are defined by Definition 2 to Definition 3.

Definition 2 The possibilistic typicality of values of a property, *P*, within the context of a concept, *C*, is represented via a *normal*, *convex* fuzzy set that is defined over the *universe of discourse*, U_P^C, by the *characterising function*, $m_P^C(x)$, as follows:

$$m_P^C(x) : U_P^C \rightarrow [0,1];\ x \in U_P^C$$

Clearly, the possibilistic typicality of precise numeric property instances, $x \in U_{ptyp}^C$, is computed by directly applying the $m_P^C(x)$. What remains to be formally defined is how the possibilistic typicality of *fuzzy*-valued (i.e., linguistic symbol, fuzzy predicate) property instances is computed. The reason for this is that membership functions defining such values may intersect more than once with the c-function.

Definition 3 Let *A* denote the linguistic label associated with an imprecise instance value (i.e., symbolic or fuzzy predicate format) of property *P*, and let U_P^C denote the concept-specific *universe of discourse* defined on *P* in *C* (where *A* is always given by $m_A^C(x) : U \rightarrow [0,1]$, $x \in U_P^C$). Further, let the characterising function, $m_P^C(x)$, define the possibility distribution over the values of *P*. Then, the possibilistic typicality, $p_P^C(A)$ of *A*, is defined as follows:

$$p_P^C(A) = \frac{1}{\left|\{x \middle| \mu_P^C(x) \wedge \mu_A^C(x)\}\right|} \sum_{x \in U_P^C} \mu_P^C(x) \wedge \mu_A^C(x) \qquad (2)$$

The first factor in equation (2) is used to normalise the typicality value based on the number of intersections between the membership of the fuzzy value with the c-function. The sum in the equation adds up the membership degrees of all intersection points.

Although the context aspect of the proposed extended prototype concept model have not been fully discussed (Section 4.3), the actual extensions made are now briefly summarised in Table 2.

Table 2: Conventional prototype concept view and extensions.

Conventional Prototype Model	Extended Prototype Model	Remarks
featural and dimensional properties	unified, uncertainty-capturing framework (concept frames)	allows to express and compare values of a single property type in precise / imprecise formats
mean value of dimensional properties	possibilistic typicality distribution (c-function)	allows to explicitly express knowledge about the constraints associated with property values
distance to mean value	possibilistic typicality	allows to establish the "central tendency" of both imprecise and precise property values
no model for context effects	concept-dependent v-functions and c-functions	permits to capture and process contextual effects

Comparing the *Game* concept summary definition example in Table 1 (which reflects the traditional prototype concept view) with the *Game* concept summary definition according to the extended model (Table 3), should help to illustrate the extended model. Note, for properties with no obvious numeric connotation, that is, universe of discourse, *U*, the unit interval serves as default universe of discourse. Also, a subscript is used for the c-functions to indicate the relationship to the corresponding concept; this notation is used to distinguish context-specific versions of a c-function.

Table 3: Extended prototype concept model representing the *Game* concept.

Property Name	Conditional Probability	Typicality Distribution (c-function)	Property Type (unified through f-functions)
name	1.00	(n/a)	character string
has winner	0.90	$m_{has-winner}^{Game}(x)$	unified featural/dimensional
competition between teams	0.80	$m_{competition-between-teams}^{Game}(x)$	unified featural/dimensional
duration	0.50	$m_{duration}^{Game}(x)$	unified featural/dimensional

4.2.1 Recognising Concept Members

Concept-recognition in the conventional prototype concept model involves measuring the distance between a new instance value (of a dimensional property) and the corresponding mean value represented at the summary-description level (see Definition 1). The extended prototype model replaces the mean value notion with a more expressive possibilistic typicality distribution (c-function). This has implications for the concept-recognition function of the extended model, as now the typicality degree of property values needs to be considered.

In the new model, an entity, *x*, is considered a member of a concept, *C*, if, and only if, the sum of the weighted (conditional probability) possibilistic typicality degrees exceeds a predefined threshold, t_C of C . Similar to the concept-recognition function of the conventional prototype concept and the exemplar concept model, this scheme allows an entity to be a member of one or more concepts, and concept membership is measured on a scale rather than being of binary type. The new model differs from its conventional counterpart by making concept membership dependent on two types of incomplete knowledge associated with concepts and their properties, namely, probabilistic and possibilistic knowledge. The precise computation of the concept membership degree for a single concept is defined below.

Definition 4 Let *C* denote a concept, and $P = \{1, \ldots, i, \ldots, n\}$ denote the properties that define *C*. Further, let $W = \{ w_1^C, \ldots, w_i^C, \ldots, w_n^C \}$ denote the corresponding concept-specific conditional probabilities associated with the properties *P* in *C*, and $M = \{ m_1^C(x), \ldots, m_i^C(x), \ldots, m_n^C(x) \}$ the associated c-functions. Finally, let the set $X = \{x_1, \ldots, x_j, \ldots, x_m\}$ denote the property instance values (numeric, symbolic, fuzzy predicate format) describing a new entity or goal query, *q*. Then, the concept membership degree, $m_C(q)$, of *q* in *C* is defined as follows:

$$m_C(q) = \left(\sum_{j=1}^{|x|=m} w_j^C \; p_j^C(x_j) \right) \Big/ \sum_{j=1}^{|x|=m} w_j^C \qquad (3)$$

such that $x_j \in X$, and (see also Definition 2 and Definition 3)

$$p_j^C(x_j) = \begin{cases} p_j^C(x_j), \text{if } x_j \text{ is a numeric property} \\ p_j^C(A), \text{if } x_j \text{ is a linguistic term or fuzzy predicate value} \end{cases} \qquad (4)$$

Clearly, equations (3) and (4) have to rely on a mapping mechanism, based on property *names*, that locates the property values/types in *X* and relates them to the corresponding property *definitions* in *C*. Details of this mechanism are discussed in [29].

4.3 Representing Context — Contextual Effects on Properties and Classification

A desirable aspect of concept properties is that they should be general. Generality, however, often leads to ambiguity. To illustrate this point, consider the sentence “We need a bird for dinner”. This sentence establishes a *context* that limits or constrains the *Bird* concept to subordinate concepts like *Turkey*, *Duck*, *Chicken*, and so on. In general, a context can be used to distinguish or *discriminate* items based on a given set of descriptors, or to *emphasise* certain features or feature combinations in a description (the latter is called *contextual priming*) [34,35]. An expressive concept or data model should be able to capture the contextual idiosyncrasies of the representational primitives (properties) used to describe higher-level entities such as concepts, classes, or relations.

In the proposed concept-oriented data model, a context is defined by a terminal concept (called *concrete concept*) in the taxonomic concept hierarchy (database schema) that models the knowledge of the associated application (see also Section 5). The rationale for this becomes obvious if one looks at a *shared* properties. Shared properties are defined at the level of an *abstract* concept, *A*, (non-terminal node) in the taxonomic concept hierarchy. By virtue of property inheritance, all *concrete* concepts (terminal nodes in hierarchy) that are derived from *A* are effectively described by the shared properties specified for *A* plus a set of additional properties. However, at the level of a concrete concept, *C*, aspects of inherited properties, such as conditional probability or c-function, may have to be interpreted in relation to all other properties (context) that define *C*. An example should help to illustrate this.

Consider the two concrete concepts *Angina* and *Asthma* in Figure 6. Based on the *Cholesterol* property defined by their supertype *HeartLungCondition*, both concepts are, among other properties, defined by a *Cholesterol* property. This representation of *Angina* and *Asthma* using a shared property promotes conceptual clarity (shared properties make apparent relationships between concepts) and economy. Now it could be argued that from a medical point of view (application semantics) *Cholesterol* contributes (conditional probability) more to the concept definition of *Angina* than it does to that of *Asthma*. For example, w_{Chol}^{Angina} = 0.80 and w_{Chol}^{Asthma} = 0.70. To put it another way, the conditional probability aspect of *Cholesterol* is interpreted differently within the *context* of the two concrete concepts that are being defined. The property concept frames depicted in Figure 5 illustrate how other aspects of shared properties, such as c-function, universe of discourse, etc., could be subject to varying contextual interpretations and effects (dashed lines depict v-functions and dashed-dotted lines c-functions).

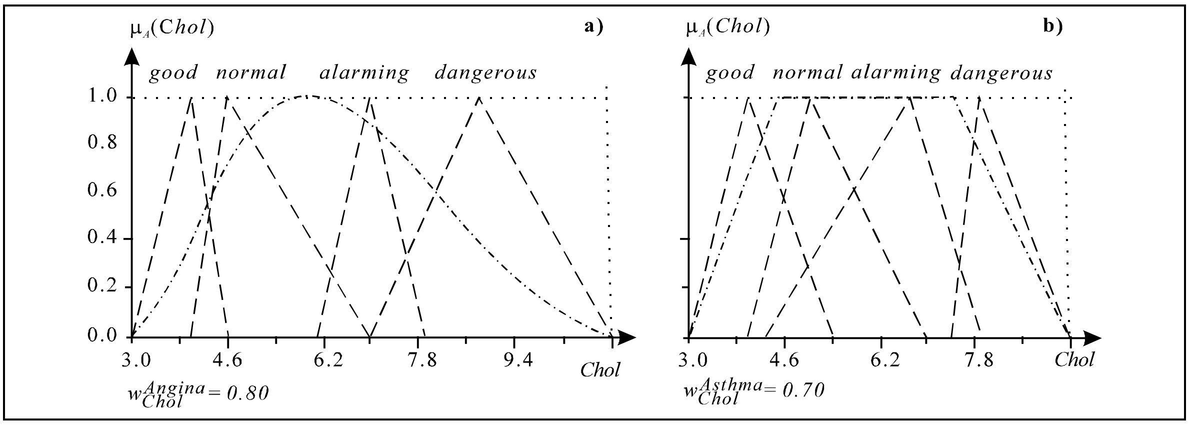


Figure 5: Contextualised *Cholesterol* property concept frames for *Angina* (a) and *Asthma* concept (b).

With this extension to the concept model it is possible to capture and process contextual idiosyncrasies of shared properties without losing the desirable generality of shared properties. The notations, definitions, and concept-recognition discussed in Section 4.2.1 have tacitly taken this context model into consideration. So there is no need to re-define the general concept-recognition model. However, no performance questions of the proposed concept-recognition model have been asked so far. This issue will be discussed in the next section.

4.4 Speeding up Concept-Recognition

Given a set of concrete concepts in a database or knowledge base schema, the basic concept-recognition algorithm would traverse all concrete concepts and evaluate the membership degrees of a new entity or goal query. Potentially, with a large number of concepts and a large average number of properties per concept, performance could become an issue. However, the fact that concepts are described hierarchically in terms of concrete *and* abstract concepts, gives rise to a pruning scheme, which would allow the concept-recognition algorithm to by-pass portions of the concept schema that are not relevant to the goal query.

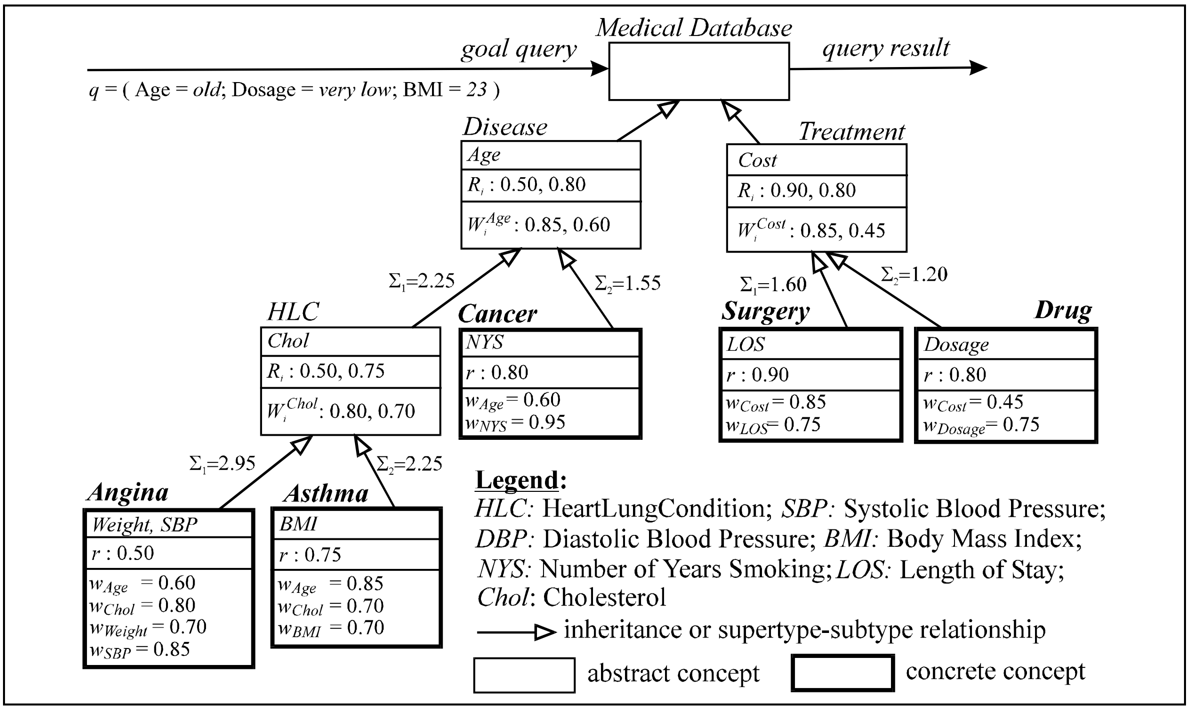


Figure 6: Example concept taxonomy (database schema) and goal query, *q*.

Consider the concept schema and the goal query, *q*, depicted the diagram in Figure 6. The schema in the diagram depicts three abstract concepts (*Disease*, *Treatment*, and *HeartLungCondition*), and five concrete concepts (*Cancer*, *Surgery*, *Drug*, *Angina*, and *Asthma*). In order to describe these concepts, a total of nine properties are used. Three of these are shared properties represented at the level of abstract concepts (i.e., *Age*, *Weight*, and *Cost*), and six are non-shared properties associated with concrete concepts (i.e., *Number of Years Smoking*, *Length of Stay*, *Dosage*, *Cholesterol*, *Systolic Blood Pressure*, and *Body Mass Index*). Each non-shared property is associated with exactly one concept frame. Each abstract property, *P*, on the other hand, is associated with as many concept frames as there are concrete concepts organised on lower levels of abstraction seen from *P*. For example, there are two *Cholesterol* concept frames, one is stored with the *Angina* concept and another with the *Asthma* concept. In Figure 6 only the conditional probability values of the concept frames is shown, other concept frame details are omitted for reasons of clarity. In addition to concept frames, concrete concepts also represent retrieval and storage thresholds. These thresholds are used as cut-off criteria for the concept-recognition process. Only the concept membership thresholds for retrieval are portrayed in the diagram. For example, r = 0.50 for the *Angina* concept and r = 0.75 for *Asthma*.

In order to allow efficient processing of goal queries like the one illustrated in Figure 6, four additional types of elements are necessary, namely, *propagated maximal conditional probabilities*, *propagated minimal storage* and *retrieval concept membership thresholds*, the *propagated minimal conditional probability sum*, and the *propagated maximal possibilistic typicality*. The first three of these elements are established at design time, so they remain constant at run-time, and the fourth element is dynamically established at run-time for each new query. The reason for calling these elements "propagated" is that they are evaluated as a result of a process that traverses the hierarchy in the direction of increasing abstraction levels.

Firstly, each abstract concept, A, stores a set of propagated minimal storage thresholds, $S = \{S_1, \dots, S_i, \dots, S_n\}$, and a set of propagated minimal retrieval thresholds, $R = \{R_1, \dots, R_i, \dots, R_n\}$. Each of these threshold values corresponds to a branch, i, leading to the corresponding sub-concept, K_i. of A. In Figure 6, only the retrieval thresholds are depicted. For example, the expression "R_i: 0.50, 0.80", which is defined at the abstract concept *Disease*, indicates the minimal propagated threshold along the sub-branch leading to the sub-concepts *HeartLungCondition* (first value: 0.50) and *Cancer* (second value: 0.80). For an abstract concept, A, the propagated minimal retrieval concept membership threshold, R_i, relating to sub-concept S of A along sub-branch i, is computed by min(r_1, r_2…, r_m), where r_1, r_2…, r_m represent the retrieval concept membership thresholds stored at the concrete concepts organised in the hierarchy below A along i. For example, for the concept *Disease*, R_1 = min(0.50, 0.75) = 0.50, and R_2 = min(0.80) = 0.80. Similar considerations apply to the storage threshold.

Secondly, each abstract concept, A, stores or represents otherwise a set, $M = \{M_1, \dots, M_i, \dots, M_n\}$, of propagated minimal conditional probability sums each of which corresponds to a sub-branch, i, leading to the corresponding sub-concept, K_i of A. The value for a sub-branch-specific *propagated* minimal conditional probability sum, M_i, is computed as follows:

$$M_i = \min\left(\sum_{j=1}^{m_X} w_j^X , \sum_{j=1}^{m_Y} w_j^Y , \ \mathbf{K}, \sum_{j=1}^{m_Z} w_j^Z \right) \qquad (5)$$

such that *X*, *Y*, …, *Z* represent all concrete concepts organised below *A* in the concept hierarchy along sub-branch *i*, and w_i^X, … , w_i^Y, …, w_i^Z represent the corresponding sets of conditional probabilities associated with the properties describing *X*, *Y*, …, *Z*. As an example consider the sum M_1 for the concept *Disease* (depicted in the diagram by $\Sigma_1 = 2.25$ close to the inheritance relation between and *HeartLungCondition*); $M_1 = \min\,[(w_{Age}^{Angina} + w_{Chol}^{Angina} + w_{Weight}^{Angina} + w_{SBP}^{Angina}), (w_{Age}^{Asthma} + w_{Chol}^{Asthma} + w_{BMI}^{Asthma})] = \min[(0.60 + 0.80 + 0.70 + 0.85), (0.85 + 0.70 + 0.70)] = 2.25$.

Thirdly, each abstract concept, *A*, stores a set of sub-branch-specific propagated maximal conditional probabilities, $\{W_1^x, …, W_i^x, …, W_n^x\}$, $\{W_1^y, …, W_i^y, …, W_n^y\}$, …, $\{W_1^z, …, W_i^z, …, W_n^z\}$, for each shared property, *x*, *y*, … , *z*, defined on *A*. For a shared property, *p* on *A*, the specific propagated maximal conditional probability, W_i^p, along sub-branch *i* is determined based on all conditional probabilities on the concrete concepts, *X*, *Y*, …, *Z*, that can be reached along sub-branch *i* as follows: $W_i^p = \max(w_p^X, w_p^Y, …, w_p^Z)$. For example, the propagated maximal conditional probability, W_1^{Age}, of concept *Angina* along the first sub-branch, works out as follows: $W_1^{Age} = \max(w_{Age}^{Angina}, w_{Age}^{Asthma}) = \max(0.60, 0.85) = 0.85$ (see Figure 6).

Fourthly, each abstract concept, *A*, represents a method that computes the sub-branch-specific propagated maximal possibilistic typicality, $\Pi_i(v(p))$, of property values, *v*(*p*); the property *p* is a shared property represented on *A*. In contrast to the pre-computed (design-time) values described above, $\Pi_i(v(p))$, is a measure that is derived at run-time within the concept-recognition process based on the value *v*(*p*). Usually, the value *v*(*p*) forms part of a goal query. For example, consider the value *v*(*Age*) = *old* in the goal query, *q*, depicted in Figure 6. The measure $\Pi_i(v(p))$ captures the maximal possibilistic typicality of a shared property value, *v*(*p*) (numeric, symbolic, or fuzzy predicate), along sub-branch *i* of *A*. Given the shared property, *p*, and the occurrence of a value, *v*(*p*) of *p*, at the level of an abstract concept, *A*, which is described by *p*, then $\Pi_i(v(p))$ along sub-branch *i* of *A* is defined as follows: $\Pi_i(v(p)) = \max(m_p^X(v(p)), m_p^Y(v(p)), \ldots\ m_p^Z(v(p)))$, where *X*, *Y*, …, *Z* that can be reached along sub-branch *i* of *A*, and $m_p^X(v(p)), m_p^Y(v(p)), \dots\ m_p^Z(v(p))$ are described by Definition 2 and Definition 3. The diagram in Figure 6 does not directly illustrate an example, because it only depicts design-time elements of a concept schema. However, given the shared property *Age* reflected on the abstract concept *Disease*, and the value *v*(*Age*) = *old* specified by the goal query *q*, Π_1(*v*(*Age*) = *old*) may evaluate to 0.77.

Initially, before a new goal query, *q*, is evaluated by the goal query processor, *q* is assigned a ***current concept membership degree***, $m_{any}(q) = 1$. For each sub-branch-specific copy of *q*, $m_{any}(q)$ is updated, i.e., reduced, in the light of the propagated values described above. Should the value $m_{any}(q)$ fall below the "expected" propagated sub-branch-specific retrieval or storage threshold, S_i or R_i, the respective sub-branch *i* can be "pruned".

Definition 5 Based on a goal query, *q*, and the propagated stored and dynamically derived values described above, it is possible to define a sub-branch-specific ***pessimistic typicality loss***, $L_i(q)$, along a sub-branch, *i*, of an abstract concept, *A*, as follows:

$$L_i(q) = \frac{1}{M_i}\sum_{p=1}^{k} W_i^p\left(1 - \Pi_i(v(p))\right) \tag{6}$$

The situation on the level of concrete concepts is very similar to that of abstract concepts, except that the *actual* typicality degrees, conditional probabilities, etc., are used for computation. For the sake of brevity, this is not discussed in further detail.

5 Modelling Concepts in Databases

The conceptual artefacts outlined in the previous sections are now embedded in an object-oriented database context. The main components that distinguish the extended prototype concept model (see Section 4) from current object-oriented data models are the following (henceforth, to be in line with accepted database terminology, the term *class* is used instead of concept, and *attribute* instead of property):

- explicit model for modelling *uncertain* associated with class-attribute (conditional *probability*);

- explicit model for expressing *possibilistic* uncertainty of attribute instances (v-function, value specification) and for representing prototype-centred classes (c-function) (others [22,36] provide mechanisms for expressing possibilistic uncertainty in databases);
- unified framework for integrating *symbolic* (and fuzzy predicate) with *numeric* attribute values through attribute concept frame;
- explicit model that supports goal queries based on the prototype-centred concept-recognion function (effectively, this can be viewed as an “object checkpoint” role of a class, complementing the classical “object factory” and “object ware house” [37] class roles); and
- explicit model for modelling *contextual idiosyncrasy* (context effects) of class attributes.

For the purpose of embedding concepts in an object data model, the ODMG Standard [38,39] has been chosen. However, none of the extensions made here are ODMG-specific, they can be applied to any object data model. Firstly, a concept-oriented data model is being designed, which specifies a meta model for constructing concept databases. Secondly, in order to declare concepts and concept frames, the ODMG object definition language (ODL) is extended. Lastly, for expressing conventional as well as CBR goal queries against a concept database, the ODMG object query language (OQL) is extended accordingly. Examples from the coronary heart disease domain are used throughout the section to illustrate the corresponding constructs.

5.1 Concept-Oriented Data Model

The object-oriented data model of the ODMG Standard [38] has been chosen to illustrate how the proposed concept-oriented constructs can be integrated into and extend existing object-oriented data models. Figure 7 provides a description of the resulting concept-oriented ODMG data model using Rumbaugh’s well-known graphical OMT notation. Grey-shaded areas indicate conceptualised additions to the original data model; italic fonts mark changes introduced to incorporate concept-oriented extensions; and plurals denote set-valued attributes, for example, the attribute *super_types* refers to a set of super types.

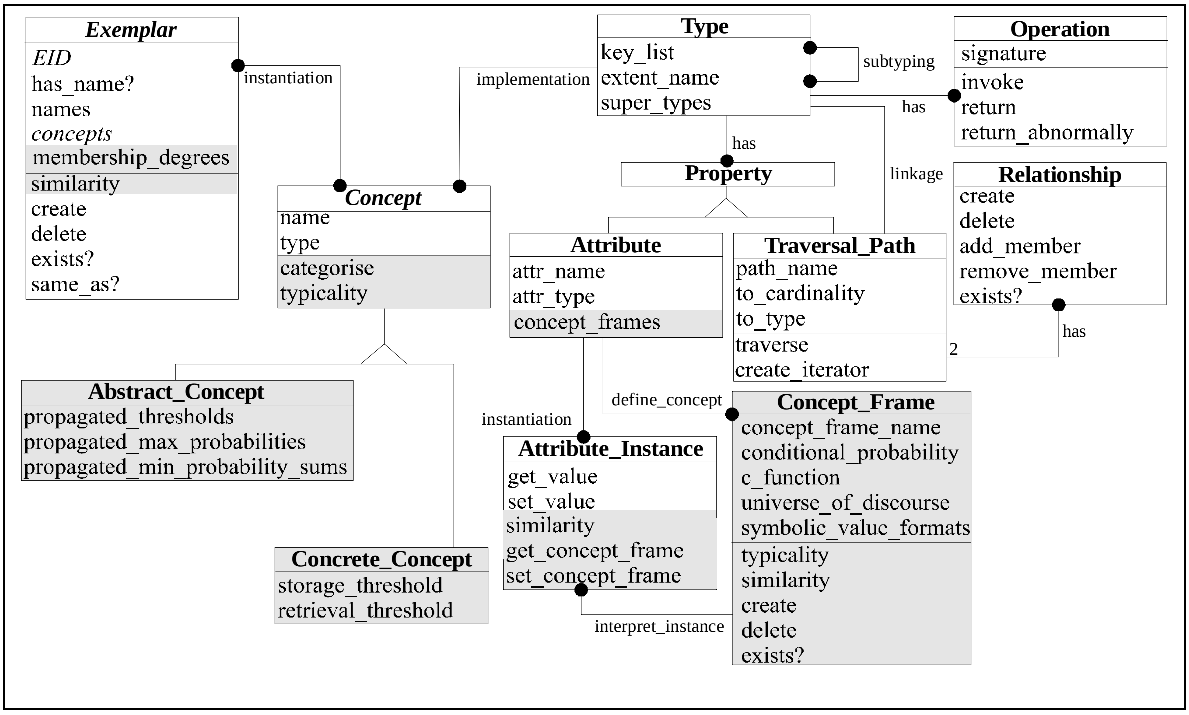


Figure 7: Extended, concept-oriented ODMG data model (partially adapted from [39]).

The *Object* type has been changed to its *Exemplar* counterpart, which, unlike traditional objects, can belong to more than one concept. Each instantiation is allotted a *membership_degree*, which indicates the degree of belonging of each exemplar to a concept. Furthermore, a similarity method has been added, which calculates an exemplar-to-exemplar similarity degree based on the similarities of exemplars' constituent attributes. The *Concept* meta data type represents the *name* and *type*, as well as the methods *categorise* and *typicality* respectively. The *Concept* type is the *Class* counterpart in databases, extended by the mentioned two methods. Its attributes and methods are inherited by the two sub-types *Abstract_Concept* and *Concrete_Concept* respectively (these correspond to the abstract and concrete concept constructs discussed in Section 4). An *Abstract_Concept* stores the propagated threshold values for storage and retrieval (S_i, and R_i), propagated maximal conditional probability degrees per shared property (w_i^p), and propagated minimal probability sums (M_i) — compare also with the corresponding components described in Section 4.3 and Figure 7. The *categorise* method is the central mechanism that controls the classification or class-recognition procedure. There exist two basic versions of the *categorise* method, both take as input a goal query, q. One version returns the names and membership degrees of the concrete concepts for which q is classified, the other returns an ordered set of concept instance-similarity pairs of the form (i,$sim(i,q)$). The instance with the highest similarity degree is the object or case which is most relevant to goal query *q*. Queries are described in detail in Section 5.3. The *categorise* method uses the propagated values (actual values on the level of *Concrete_Concept*) and the typicality method as described in Section 4.2 and Section 4.3.

The key component that permits the conceptualisation (concept definition), uncertainty representation, and unification of attribute value types is reflected by the component *Concept_Frame* (see Section 3.1.2 and Section 4). A concept frame is used for both the representation of a concept or class (concept-defining role) as well as the interpretation of attribute instances according to the context defined by a *Concrete_Concept*. This situation is modelled in the diagram by the relationships *define_concept* and *interpret_instance*. For non-shared attributes, the *define_concept* relation is one-to-one, for shared attributes it is one-to-many (context-specific versions). The cardinality of the *interpret_instance* relationship depends on whether the corresponding attribute is a shared or non-shared attribute, and on the number of attribute instances that are in existence at a given point in time.

5.2 Exemplar Definition Language

To keep consistency with the ODMG model enhancements incorporated above, the ODMG-ODL has been extended to allow the definition of exemplars. Again, the extensions are not ODMG-specific, syntactical statements are just borrowed for convenience.

Generally, in ODMG every type is defined as an interface compatible with the OMG interface definition language (IDL). An interface consists of an *extent*, a set of *keys*, a set of *attributes* and *relationships*, as well as a set of *operation signatures*. Relatively few syntactical extensions are necessary to support all required constructs of the concept-oriented data model.

The following two sub-sections describe the necessary additions needed to support the declaration of abstract and concrete concepts as well as concept frames. A full BNF specification of the discussed ODMG extensions is given in Appendix A.1 to A.3.

5.2.1 Concept Declaration

At design-time, abstract as well concrete concepts can be declared, and at run-time all existing exemplars are assigned to one or more concrete concepts with their corresponding membership degrees. The declaration of both types of concepts contains a name and an IDL-conformant interface body, which holds information about attributes, relationships, exceptions and operations. Optionally, super-concepts that are extended or inherited and a list of types (containing extent and key information) can be specified[1]. Each abstract concept specifics one or more propagated thresholds, maximal probabilities, and minimal probability sums. Concrete concepts specify a storage and a retrieval threshold respectively. An example for each concept type should clarify the usage of the exemplar definition language for concept declaration purposes.

First, an abstract concept, called *HeartLungCondition*, representing heart and lung conditions is declared. *HeartLungCondition* inherits attributes and operations from the *Disease* concept, such as *Age*. The concept-specific settings are a propagated storage and retrieval thresholds for *Angina* and *Asthma*, propagated maximum probability for both conditions for the *cholesterol* attribute, and propagated minimum probability sums for both concrete concepts.

```
abstract concept HeartLungCondition extends Disease
    propagated_storage_threshold Angina 0.5
    propagated_retrieval_threshold Angina 0.5
    propagated_storage_threshold Asthma 0.75
    propagated_retrieval_threshold Asthma 0.75
    propagated_max_probability Angina cholesterol 0.8
    propagated_max_probability Asthma cholesterol 0.7
    propagated_min_probability_sum Angina 2.95
    propagated_min_probability_sum Asthma 2.25
    {
        attribute float cholesterol;
        relationship set <Cancer> has_also inverse Cancer::patient
    }
```

Next, two concrete concepts representing special heart-lung conditions are specified, namely the *Asthma* and *Angina* concepts. From the examples below it can be seen that each concept has different storage and retrieval thresholds, respectively. These concept-specific thresholds reflect the contextual idiosyncrasies of the corresponding concepts with regard to classifying new queries for the retrieval and storage operation respectively.

[1] Although the terms are regularly used interchangeably, ODMG distinguishes between types and classes (concepts). “Type is an abstract concept, while class has implementation connotations” ([39]), or in short, data values have type; objects (exemplars) have a class (concept).

```
concrete concept Asthma extends HeartLungCondition
    storage_threshold 0.5
    retrieval_threshold 0.5
    {
        attribute unsigned float weight
        attribute unsigned short systolic_blood_pressure
    }

concrete concept Angina extends HeartLungCondition
    storage_threshold 0.75
    retrieval_threshold 0.75
    {
        attribute unsigned float body_mass_index
    }
```

5.2.2 Concept Frame Declaration

Like the concept declaration extensions, the concept frame declarations support all semantic constructs outlined in Section 4. The concept frame name is the key, which links the concept frame to an attribute and consequently to its attribute instances. The conditional probability is specified as a real number from the unit interval [0,1]. The *c_function* is specified in the exemplar definition language as string representing a mathematical function, which has to be interpreted by the query processor. The *universe of discourse* specifies a range, while symbolic value formats declare *linguistic terms* and their corresponding v-functions. Each *symbolic expression* consists of a list of mathematical functions, which together form a linguistic term format definition. Using the *Cholesterol* concept frame depicted in Figure 5, part (a), the following concept frame is defined to illustrate the discussed notions.

```
concept frame cholesterol
{
    conditional_probability 0.80
    c_function "2.4 * exp(0.5 * (x – 0.01) ^ 2)"
    universe_of_discourse 3.00 11.0
    symbolic_expression good "1.25 * x – 3.75"   "– 1.25 * x + 5.75"
    symbolic_expression normal "1.25 * x – 5.00"   "– 0.46 * x + 3.20"
    symbolic_expression alarming "1.25 * x – 7.75"   "– 1.25 * x + 9.75"
    symbolic_expression dangerous "0.68 * x – 5.83"   "– 0.68 * x + 8.83"
}
```

5.3 Concept Query Language

After adding linguistic constructs to ODL, which allows the storage of conceptual information, its query counterpart has to be specified, too. The objective of formulating a query is express a goal, which contains a case description, a case solution, and, optionally observed outcomes or effects that ensued after applying the solution of previous cases. From a database point of view, the main objective is to keep as much of an existing query language as possible in order to keep underlying principles, such as programming language independence, computational completeness and declarativity. Another design objective is to re-use of implementation-specific components like optimisation mechanisms. As a result it has been decided to extend the ODMG object query language (OQL) with a GOAL keyword, keeping the remaining structure as is. The high-level BNF of the concept query language looks as follows (bold face fonts indicate the added concept-related parts; the BNF extensions are listed in full in Appendix A.3):

```
SELECT <projection>
FROM <concepts>
[WHERE <query>]
[GROUP BY <attributes>]
[HAVING <query>]
[ORDER BY <sort_criterion>]
[GOAL <conventional_goal_query> | <CBR_goal_ query>]

<conventional_goal_ query > :=
   <case_description> <retrieval_membership_degree> [similarity]

<CBR_goal_ query > :=
   <case_description> <retrieval_membership_degree> <case_solution> [case_outcome]
```

A conventional goal query requires a *case description*, which contains the attributes that are relevant for the case and the *retrieval membership degree*. The attributes can be in form of conventional attribute names, but, in order to conform to object database conventions, can also be in the form of relationships or methods. The result of such a query results in a set of classified cases, which match the case description and trigger the given threshold. In case the *similarity* flag is set, similarity will be measured and the result will be a set of ordered cases, usually smaller than the one after pure classification.

In addition to the *case description* and the *retrieval membership degree*, a CBR goal query requires a *case solution*, which has the same structure as the case description, namely a list of attributes. The optional *case_outcome* parameter acts as a flag, which indicates whether results from previous cases should be retrieved or not. Two examples should illustrate the usage of the concept query language, facilitating the query scenarios depicted in Figure 1.

```
select age, sex, cholesterol, CHDRisk
    from Asthma
    where age > 40
    goal description age, sex, cholesterol (0.7) similarity
```

Querying the concrete concepts of asthma diseases, the description part contains the attributes *Age*, *Sex* and *Cholesterol*, the retrieval membership degree is set to 0.7, and it is required that, after classifying the described cases, the second step of retrieval, namely similarity-based ranking, is undertaken, too.

```
select age, sex, cholesterol, CHDRisk
    from Asthma
    where age > 40
    goal description age, sex, cholesterol (0.7) solution prediction(3) case_outcome
```

This CBR goal query contains the same description part and retrieval membership degree as the query above. The solution part specifies a method that predicts the coronary heart disease risk after three years and the case_outcome flag indicates that previous cases have to be retrieved, too. A general constraint, which has to be considered at the implementation stage, is that the intersection of specified case description attributes and case solution attributes must be empty. That is no attributes can occur in the description and the solution part at the same time.

The principle of relational closure of the query language, that is every operator returns a relation as a result, is still given, which allows the nesting of queries. This might be useful when the above two queries have to be executed consecutively in order to combine conventional and CBR goal queries, for example,

```
select age, sex, cholesterol, CHDRisk
    from Asthma
    where ( select age, sex, cholesterol, CHDRisk
              from Asthma
              where age > 40
              goal description age, sex, cholesterol (0.7) similarity)
    goal description age, sex, cholesterol (0.7) solution prediction(3) case_outcome
```

The query constructs outlined above support conventional goal queries as well as CBR goal queries. To keep the query language and the underlying query processor as compact as possible, a query model that separates conventional from CBR goal query elements (e.g., adaptation) is desirable. Such a query model is illustrated in Figure 1 and Figure 2 respectively. This idea leads to a knowledge base management system architecture, which has as its core the concept-oriented database and query model outlined above.

6 Concept-Oriented Databases and Knowledge Base Management Systems

Case-based reasoning is a framework that revolves around a memory of past cases for solving new problems and interpreting new situations [7,11]. It can be shown that CBR systems and database systems — in particular *object-oriented* database management systems — have many features in common [40]. In both systems, the crucial function is to provide access to *relevant* information (objects, cases). However, in contrast to the database approach, where relevancy resides in the head of the user who issues a query, the case-based approach captures the notion of relevancy in the underlying (case) data model and the associated query processing strategy. CBR is therefore proposed as a framework for knowledge base management systems. The basis for this framework is the concept-oriented data model and its goal query capability discussed in previous sections.

The concept-oriented data model proposed in this work defines an extension to existing object-oriented data models. This model supports goal queries based on the prototype definition of the classes contained in a class schema, and a concept-recognition function used to actually assign new queries to relevant classes in the class schema. Data retrieval in the proposed model results in a set of relevant items that may originate from more than one class (extent) in the system. This set of potential candidates can be further analysed for the most promising ones based on the similarity measure associated with classes. So a goal query in the proposed model is carried out as a two-phase procedure, concept-recognition or classification plus case-to-case similarity-based filtering. The main advantage of such a query approach is its ability to cope with a potentially large number of cases. From a CBR perspective, such a two-phase retrieval architecture represents a mixed index/similarity approach. A case-knowledge base management system (KBMS) would use the concept-oriented data model as its core for data management an relevance-driven retrieval of cases. To separate the reasoning part from the goal-query part, a special query processor capable of adaptation and processing of CBR goal queries would form the key component of the KBMS (this situation is depicted in Figure 2).

Some of the crucial aspects of the proposed concept-oriented data model and the KBMS extensions have been tested in a prototype system. The objective of the system is to give initial advice to asymptomatic test subjects on their coronary heart disease (CHD) risk. The main point of this study was to have the subjects use the system and provide CHD-relevant data *without* first consulting a medic to establish various parameters such as *blood pressure*, *cholesterol*, *anxiety* etc. Based on the reference subjects in the database, which were monitored over a three year period, the subjects where given a risk estimate and advice.

The data used in the study was established for 83 male, middle-aged subjects in 1993 as well as in 1996, it consisted of several structured or composite and atomic CHD risk factors. The corresponding data structure and their expert-assigned conditional probabilities are reflected in Table 4 (for the sake of brevity a detailed description is omitted).

Table 4: CHD patient data (structured attributes in bold style, atomic in normal style).

Complex/Atomic Attribute	Conditional Probability	Property Structure
Chol	0.80	(*TChol*, *Trig*), (*LDL*, *HDL*)
Smoke	0.80	*Pipe*, (*nCPD*, *nYears*)
HCHD	0.60	*PHCHD*, *FHCHD*
PA	0.90	*PAW*, *PAL*, *PAS*
ST	0.60	*STW*, *STP*, *STH*, *STM*
BP	0.60	(*DBP*, *SBP*), *PHBP*
PHDiabetes	0.50	*n/a*
Age	0.80	*n/a*
BMI	0.35	*n/a*
BF%	0.35	*n/a*
Alc	0.15	*n/a*
SC	0.05	*n/a*

Chol *= cholesterol* ***Smoke*** *= smoking* ***HCHD*** *= history of coronary heart disease*
PA *= physical activity* ***ST*** *= stress* ***BP*** *= blood pressure*

In a series of retrieval experiments the precision of the risk factors in both the test and the training sets was modified. It was confirmed that the system could handle the introduced uncertainty in a robust and consistent fashion. Overall, the domain expert who helped to set up the case base was convinced that the system reflected his experience very accurately. He agreed that the way the system let him express the various types of knowledge and constraints about his subject was easy to use and understand, he also observed that a system of this kind would be very useful in the hands of a general practitioner.

To establish some idea about the model's performance, 10,000 cases, each described by 21 attributes (7 concept frames with 5, 7 and 9 v-functions respectively) have been seeded randomly with values of the three data formats (numeric, symbolic, fuzzy predicate). On 100 different value distributions over all cases, the average time to process all 10,000 cases was 11.74 seconds (minimum 7.22 sec, maximum 16.63 sec). The test configuration was a 90 MHz Pentium PC, 32 MB RAM on a Windows NT4 platform.

7 Summary and Conclusions

This paper proposed a data model that extends conventional relational and classic/fuzzy object-oriented database models. The proposed model provides inherent support for conventional goal queries and CBR goal queries, it provides a coherent representation and processing framework for unifying symbolic and numeric data formats, and it permits the modelling of vague, imprecise data with respect to probabilistic and possibilistic uncertainty. Furthermore, based on a prototype representation of concepts or classes, the model defines an explicit, efficient, and effective concept-recognition-model.

As the model revolves around the notion of *concepts* (prototype concept model), it is called *concept-oriented data model*. It was demonstrated how this model can be defined as an extension to existing object-oriented data models (e.g., ODMG 2.0). Furthermore, it was shown that the model, together with well-established methods and mechanisms from the CBR field, could provide a robust framework for building knowledge base management systems. The crucial components of this model were tested and evaluated on medical data. The main results obtained from this work relate to the model's flexibility and adequacy for expressing, modelling, and processing (goal query) ontological, scientific, natural, and artificial concepts. The model allows the tight integration of various types of knowledge (deep, general domain knowledge, uncertain or incomplete knowledge, and knowledge about constraints and context) within the concept or data schema. As such the model addresses the epistemological adequacy issue known in the knowledge representation community [12]. In order to study the model's potential for a wide-spread acceptance in the mainstream database community, further integration work is needed, for example, as part of a heterogeneous database environment, and with regard to more conventional database functionality such as query optimisation, security, and so on. The current model for processing conditional probabilities could be reviewed in the light of rough set and / or evidence theory.

Appendix A: BNF Extensions of the ODMG ODL

Note, all expressions in bold style represent key words.

A1. Concept Declaration

<abstract_concept_dcl> ::=
  abstract concept <identifier>
  [: **extends** <scopedName>]
  [<inheritance_spec>]
  [<type_property_list>]
  <propagated_thresholds>
  <propagated_max_probabilities>
  <propagated_min_probability_sums>
  {<interface_body>}   // attributes, relationships, exceptions, and methods according to IDL

<concrete_concept_dcl> ::=
  concrete concept <identifier>
  [: **extends** <scopedName>]
  [<inheritance_spec>]
  [<type_property_list>]
  storage_threshold <floating_pt_literal>
  retrieval_threshold <floating_pt_literal>
  {<interface_body>}   // attributes, relationships, exceptions, and methods according to IDL

<propagated_thresholds> ::=
  <propagated_threshold> | <propagated_threshold> <propagated_thresholds>

<propagated_threshold> ::=
  propagated_storage_threshold <concept_name> <floating_pt_literal>
  propagated_retrieval_threshold <concept_name> <floating_pt_literal>

<propagated_max_probabilities> ::=
  <propagated_max_probability> | <propagated_max_probability>
  <propagated_max_probabilities>

<propagated_max_probability> ::=
  propagated_max_probability <concept_name> <attribute> <floating_pt_literal>

<propagated_min_probability_sums> ::=
  <propagated_min_probability_sum> | <propagated_min_probability_sum>
  <propagated_min_probability_sums>

<propagated_min_probability_sum> ::=
  propagated_min_probability_sum <concept_name> <floating_pt_literal>

A2. Concept Frame Declaration

<attrib_dcl> ::=
  [**readonly**] **attribute**
  <domain_type> <identifier>
  [**[**<positive_int_const>**]**]
  concept_frames <concept_frames>

<concept_frames> ::=
  <concept_frame_name> | <concept_frame_name> <concept_frames>

```
<concept_frame_dcl> ::=
    concept frame <string_literal>
    {
    conditional_probability <floating_pt_literal>
    c_function <function>
    universe_of_discourse <floating_pt_literal> <floating_pt_literal>
    <symbolic_value_formats>
    }

<symbolic_value_formats> ::=
    <symbolic_value_format> | <symbolic_value_format> <symbolic_value_formats>

<symbolic_value_format> ::=
    symbolic_expression <linguistic_term> <v_function>

<linguistic_term> ::= <string_literal>

<v_function> ::= <function> | <function> <v_function>

<function> ::= <string_literal>
```

A3. Concept Query Specification

```
<query> ::=
    select <projection_attributes>
    from <concepts>
    [where <query>]
    [group by <partition_attributes>]
    [having <query>]
    [order by <sort_criterion>]
    [goal <conventional_goal_query> | <CBR_goal_ query>]

<conventional_goal_query> :=
    description <case_descriptions> (<retrieval_membership_degree>) [similarity]

<CBR_goal_query> :=
      description <case_descriptions> (<retrieval_membership_degree>)
      solution <case_solutions> [case_outcome]

<case_descriptions> ::=
    <case_description> | <case_description> , <case_descriptions>

<case_solutions> ::=
    <case_description> | <case_description> , <case_descriptions>

<case_description> ::=
    <attribute_name> | <relationship_name> | <operation>

<retrieval_membership_degree > ::=
    <floating_pt_literal>
```

References

[1] A. Motro, “Extending the Relational Data Model to Support Goal Queries”, in *Expert Database Systems*, L. Kerschberg (editor), pp129-149, Benjamin/Cummings Pub. Co., Inc., 1987.

[2] C. Eastman, “Approximate Retrieval: A Comparison of Information Retrieval and Database Management Systems”, in *IEEE Data Engineering* 12 (2), pp41-45 (1989).

[3] N. Fuhr, “A Probabilistic Framework for Vague Queries and Imprecise Information in Databases”, in *Proc. 16th International Conference on Very Large Databases*, (1990).

[4] B.P. Buckles, F.E. Petry, “A Fuzzy Model for Relational Databases”, in *Fuzzy Sets and Systems*,7:213-226 (1982).

[5] F.E. Petry, *Fuzzy Databases: Principles and Applications*, Kluver Academic Pub., MA, 1996.

[6] J.L. Kolodner, *Case–Based Reasoning* (Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993).

[7] A. Aamodt, E. Plaza, “Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches”, in *AICOM*, vol. 7, 1 (1994).

[8] Kim, W. (editor), *Deductive and Object-Oriented Databases*, Elsevier Science Publishers B.V., (North-Holland), 1990.

[9] Donovan, H.; A logic to unify semantic-network knowledge systems with object-oriented database models, Journal of Object-Oriented Programming, 1993.

[10] E.E. Smith, and D.L. Medin, *Categories and Concepts* (Harvard University Press, 1981).

[11] J.L. Kolodner, *Case–Based Reasoning* (Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993).

[12] A. Aamodt, E. Plaza, “Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches”, in *AICOM*, vol. 7, 1 (1994).

[13] T.W. Liao, Z. Zhang, & C.R. Mount, “Similarity Measures for Retrieval in Case-Based Reasoning Systems”, *Applied Artificial Intelligence*, Vol. 12, pp. 267-288, 1998.

[14] S-J. Chen, C-L. Hwang, *Fuzzy Multiple Attribute Decision Making, Methods and Applications*, Springer Verlag, 1992.

[15] S-M. Chen, “Measures of Similarity between Vague Sets”, in *Fuzzy Sets and Systems*, vol. 74, 2, pp217-223 (1995).

[16] W. Dubitzky, A. Schuster, D.A. Bell, J.G. Hughes, K. Adamson, “How Similar is VERY YOUNG to 43 Years of Age? On the Representation and Comparison of Polymorphic Properties”, in *Proc. 15th Int. Joint Conf. on Artificial Intelligence*, pp226-231, Japan, 1997.

[17] F. Gebhardt, “Survey on Structure-Based Case Retrieval”, in *The Knowledge Engineering Review*, vol. 12:1, pp41-58, 1997.

[18] R.C. Schank, *Dynamic Memory: A Theory of Learning in Computers and People* (Cambridge University Press, 1982).

[19] J.L. Kolodner, *Retrieval and Organizational Strategies in Conceptual Memory* (Lawrence Erlbaum Associates, Hillsdale, N.J., 1984).

[20] B. Porter, R. Barreiss, R. Holte, “Concept Learning and Heuristic Classification in Weak Theory Domains, in *Artificial Intelligence*, 45 (1-2), pp229-263 (North-Holland, 1990).

[21] D.A. Bell, J.W. Guan, S.K. Lee, “Generalised union and project operations for pooling uncertain and imprecise information”, in *Data and Knowledge Engineering*, pp89-117, vol. 18 (1996).

[22] R. George, F.E. Petry, P.B. Buckles, “Modelling Class Hierarchies in the Fuzzy Object-Oriented Data Model”, in *Fuzzy Sets and Systems*, 60, 3 (1993).

[23] J.F. Baldwin, T.P. Martin, B.W. Pilsworth, *Fril—Fuzzy and Evidential Reasoning in Artificial Intelligence* (Research Studies Pr., 1995).

[24] D.B. Leake (editor), *Case-Based Reasoning: Experiences, Lessons & Future Directions* (MIT Press, MA, 1996).

[25] I. van Mechelen, J. Hampton, R.S. Michalski, P. Theuns, *Categories and Concepts: Theoretical Views and Inductive Data Analysis* (Academic Press, 1993).

[26] A. Aamodt, *A Knowledge-Intensive Approach to Problem Solving and Sustained Learning*, Ph.D. Thesis, Norwegian Institute of Technology, University of Trondheim, Norway, 1991.

[27] D. Patterson, W. Dubitzky, S.S. Anand, J.G. Hughes, "On the Automation of Case Base Development from large Databases", in Proc. AAAI 1998 Workshop: Case-Based Reasoning Integrations, pp126-130, US, 1998.

[28] A. Tversky, “Features of Similarity”, in *Psychological Review*, vol. 84, 4, pp327-252 (1977).

[29] W. Dubitzky, *Knowledge Integration in Case-Based Reasoning: A Concept-Centred Approach*, Ph.D. Thesis, Faculty of Informatics, University of Ulster, Northern Ireland, 1997.

[30] A. Hutchinson, *Algorithmic Learning* (Claredon Press, 1994).

[31] L.A. Zadeh, “Outline of a New Approach to the Analysis of Complex Systems and Decision Processes”, in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, No. 1, pp28-44, (Jan 1973).

[32] Peterson J., “Similarity of Fuzzy Data in a Case-Based Fuzzy System in Anaesthesia”, in *Fuzzy Sets and Systems*, pp247-262, vol. 85, 1997.

[33] A.G. Büchner, D.A. Bell, J.G. Hughes, “A Contextualised Object Data Model based on Semantic Values”, in *Proc. 11th Int’l. Conf. Parallel and Distributed Computing System*, pp171-176, 1998.

[34] R. Wilensky, “Knowledge Representation—a Critique and a Proposal”, in *Retrieval and Organizational Strategies in Conceptual Memory*, J.L. Kolodner (editor), pp15-28 (Lawrence Erlbaum Associates, Hillsdale, N.J., 1984).

[35] M. Brown, *A Memory Model for Case Retrieval by Activation Passing*, Ph.D. Thesis, Department of Computer Science, University of Manchester, UK, 1993.

[36] D. Dubois, H. Prade, J.P. Rossazza, “Vagueness, Typicality, and Uncertainty in Class Hierarchies”, in *International Journal of Intelligent Systems*, vol. 6, pp167-183 (1991).

[37] W. Kim, F.H. Lochovsky (editors), *Object-Oriented Concepts, Databases, and Applications* (Addison-Wesley, Reading, MA, 1989).

[38] R.G.G. Cattell, D. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, D. Wade (editors.) *The Object Database Standard: ODMG-93 Release 2.0*, Morgan Kaufmann Publishers, San Mateo, CA (1997).

[39] M.E.S. Loomis, “The ODMG Object Model”, in *Journal of Object-Oriented Programming*, 6(3), pp64-69 (1993).

[40] W. Dubitzky, J.G. Hughes, D.A. Bell, “A Generic, Object-Oriented Case-Knowledge Representation Scheme, and its Integration into a Wider Information Management Scenario”, in *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks*, vol. 13 (3), pp219-233 (Blackwell Publishers, UK, 1996).