Abstract. This article summarizes mathematical formalisms used to describe object and object-oriented databases. Different approaches were used to describe object databases, object database models respectively. We focus on algebraic approaches including graphs, monoid comprehensions, monads and semi-monads, and applied type theory. Especially we introduce formal categorical approach to object database modeling. Several categorical models of object databases are described as well as models of some other formalisms using category theory. We use category theory as an unifying framework for any useful formalism, its model respectively. Furthermore, category theory is used in order to compare not only different formalisms for object oriented databases but also for different database paradigms comparison, i.e. object, relational, and XML. The vision of developing database framework using formal models and their transformations—based on the MDA concept from OMG—based on category theory is finally suggested.

1 Introduction

In 1970 Dr. Codd from IBM published his article about relational data model [8] and since that time the database world have used databases based on the relational data model [9].

When the object paradigm emerged there were also propositions how to use this paradigm to store data. This model was meant to be the successor of the older relational data model. But however it was never happened till now.

---

3 This work has been supported by the Ministry of Education, Youth, and Sports under Research Program MSM 6840770014 and by the grant of GACR No. GA201/06/0756.
In 1999 the W3C consortium published the XML standard. The database community then shifted their effort to the XML databases, native XML databases respectively. Today there is a great number of theoretical works, elaborations, and experiments behind this technology.

This article gives summary of formal models and approaches to the object databases. There are many distinctive formal approaches. We systematize and describe all known types of them. We introduce categorical models useful for modeling of object databases. Furthermore we demonstrate how to use category theory when developing a formal framework for description of different models of formalisms of object databases.

Today the main information portal aggregating information about object database products, their vendors, standards, applications, case studies, etc., in short about object databases, can be found at the resource portal for education and research of the Object Database Management System organisation ([http://www.odbms.org](http://www.odbms.org)).

### 1.1 Motivation

There are at least three different database paradigms:

1. **relational** which is most widely used, especially in the area of business for large data banks,
2. **object**—after the boom of object programming languages, Java especially, there are many applications needed to handle data in the format of objects and instances, and
3. **XML**—today there are many applications in the B2B area handling XML documents, storing, searching them etc.

The question of which technology to use in praxis was treated in greater detail in one of the previous work [30]. We can ask several (theoretical) questions about these approaches, e.g.:

- Which of these paradigm is the best one?
- Is there any special purpose allowing us to distinguish which of these database paradigms to use?
- Is there any difference at the high level of the data models?
- Are all the models “essentially the same thing” (from the categorical point of view, described e.g. in [22])?

We believe that the category theory can help us to answer a lot of these questions. The category theory is in mathematics widely used for 50 years in order to help to understand that different part of mathematics are in fact “essentially the same”. It was proved e.g. that any deductive system is equivalent to particular cartesian closed category (it is used to abbreviate this as CCC) [22]. Crole has proved that categorical semantics can unify different point of view of semantics in [10], etc.

We therefore emphasize the role of category theory as unifying framework for database models—formalisms description. Using category theory we can describe:
individual model of particular database formalism independently on the database paradigm,
– classes (better categories) of these models,
– interrelationships between the models themselves and between their classes.

Relational models were considered very carefully for nearly 40 years. But the models of object databases have their own long history as well. During last approximately 30 years there were proposed several formalisms and models of object databases. In this article we focus on these formalisms and models, and furthermore on their categorical description. Why?

There are some interesting questions about different existing formalisms:
– Are they comparable?
– How to compare them? How powerful are the languages derived from these formalisms?
– What language is possible to use to describe interrelationships between these formalisms?
– And there are more questions.

Furthermore we can be interested in the problems of relationships to the other models of other paradigms.

XML data models were treated in greater detail in [28], [29], and in [27].

**Terminological Note** Typically the term object-oriented database is used in the sense that there are not all the object properties present—the database management system does not support all of the capabilities of object paradigm. But then object databases have the capabilities of all considered object properties which typically are object instantiation, aggregation, inheritance and encapsulation (mandatory is that an object have to have the identity, and arbitrary, but nevertheless finite, number of attributes).

We will use the terms object-oriented and object database as synonyms. Because we will consider just the mandatory properties of object instances as they are described above.

**Paper Structure** This article is structured as follows. The section 2 summarizes known formalisms used for modeling object databases. Surprisingly there are many different formalisms used. The section 3 refers the categorical point of view of the database engineering. First we introduce some of the basic terms of the category theory and subsequently we deal with existing categorical approaches. Next section 4 treats of proposed models. We focus on categorical description. Last two sections concludes this paper summarizing what was done and what should be done in the future.

## 2 Known Formalisms Summary

Today there are many formalisms which help us to understand, describe, and formalize the object-oriented databases. There are several approaches useful when
describing model of object database. We will discuss just object databases from the point of view of their data models.

From the wide variety of formalisms we are especially pinpointing

- ontology-based approaches,
- monoid comprehension approach,
- monad-based description and
- type theory approach among others.

Some of the formalisms we treat later in this section are digestedly summarized in the paper of Buneman et. al [4].

Probably most important and interesting results published Leonidas Fegaras. He was long time interested in the research of the object databases formal models and formal languages. He suggested and used the method of monoid comprehensions which was used by many other researchers (as we’ll see later). Fegaras’ work recapitulation can be found at [15]. Fegaras’ works in the area of the object databases have culminated in λ-DB—an object database implementation with interface to C++ programming language. A digestive source can be found at [15], http://lambda.uta.edu respectively.

First formalism, we treat here, is an lambda calculus based model. The result of Leonidas Feagaras long time work in the area of object databases. We should note here that nowadays he focuses on XML databases and data streams.

2.1 lambda-DB: An ODMG-Based Object-Oriented DBMS

The heading of this subsection brings the name of Fegaras’ article, which we describe in some detail now.

Object databases have to compete with relational databases. Object database management systems in praxis must meet the performance requirements of at least as high-performance query evaluation engine as it is in case of relational database management system.

Object technology can use relational and can benefit from the already proven techniques as e.g. relational query-optimization technology is. Object languages include features like object identity, methods invocation, encapsulation, subtype hierarchy, user-defined type constructors, large multimedia objects, multiple collection types, arbitrary nesting of collections, and nesting of query expressions. But how to handle these features?

In [15] Fegaras describe implementation of ideas of monoid comprehensions in the λ-DB which is an Object-Oriented Database Management System based on the ODMG standard as it is described e.g. in [3]. We will focus on monoid comprehensions in greater detail later in this section.

VOODOO: A Visual Object-Oriented Database Language for ODMG OQL

In 1999 Fegaras in [13] introduced visual language to express ODMG OQL queries. The language is expressive enough to allow most types of

- query nesting,
Formal Models of Object-Oriented Databases

- aggregation,
- universal and existential quantifications,
- group-by, and
- sorting,

and at the same time is uniform and simple to learn and use.

This visual language is strongly typed in the sense that constructed queries are always type-correct. There is sufficient type information displayed by the system that guides every stage of the query construction. In this visual language is used only one generic visual construct, called a template, instead of inventing a new one for each OQL syntactic feature.

2.2 Older Fegaras’ Works

We can figure out that in the background of all Fegaras’ works dealing with object databases is the monoid comprehension calculus. This algebraic approach seems to be powerful enough to solve the object databases basic problems—collections of objects, and all the object interrelationships in general.

Query unnesting in object-oriented databases In 1998 in SIGMOD conference Fegaras [11] have treated query optimization in object-oriented databases. One of the most challenging problems is query unnesting. The embedded query can take any form, including aggregation and universal quantification.

The lack of a uniform algebra that would treat all operations, including aggregation and quantification, in the same way was the primer reason why other approaches failed. The system proposed by Fegaras uses the monoid comprehension calculus as an intermediate form for queries in object databases. The monoid comprehension calculus treats operations over multiple collection types, aggregates, and quantifiers in a similar way, resulting in a uniform way of unnesting queries, regardless of their type of nesting.

Such a universal way of description—algebraic—was proven to be very useful. Later in the next section we generalize this idea.

Optimizing Queries with Object Updates Object-oriented databases provide powerful data abstractions and modeling facilities. But they usually lack a suitable framework for query processing and optimization as it is in the case of relational databases. The presence of object identity and destructive updates are needed to be considered.

A formal framework for optimizing object-oriented queries and updates in the presence of side effects. Fegaras [12] developed and used language extension to the monoid comprehension calculus for these purposes. His method is based on denotational semantics, which is often used to give a formal meaning to imperative programming languages.

This method not only maintains referential transparency, which allows us to do meaningful query optimization, but it is also practical for optimizing object queries since it allows the same optimization techniques applied to regular queries to be used with minimal changes for object queries with updates.
**Optimizing object queries using an effective calculus** To let object databases be as successful as relational systems are they would meet the performance requirements of many non-traditional applications. To achieve this goal there is needed to develop an effective query optimizer.

Fegaras and Maier in [14] proposed an effective framework with a solid theoretical basis for optimizing object query languages. The proposed monoid comprehension calculus, captures most features of ODMG OQL [1] and is a good basis for expressing various optimization algorithms concisely.

### 2.3 CCUBE

The CCUBE is a constraint object-oriented database system, described in detail in [5]. Using constraints is useful to provide a flexible and uniform way to:

- represent diverse data capturing spatio-temporal behavior,
- complex modeling requirements,
- partial and incomplete information etc.

Constraint approach have been used in a wide variety of application domains. Constraint databases integrate data captured by constraints in databases. CCUBE is the first constraint object-oriented database system. The CCUBE should be used for the implementation of high-level constraint object-oriented query languages as well as for directly building software systems requiring extensible use of constraint database features. The CCUBE data manipulation language, Constraint Comprehension Calculus, is based on monoid comprehensions. The data model for the constraint calculus is based on constraint spatio-temporal (CST) objects that may hold spatial, temporal or constraint data, conceptually represented by constraints. Monoid is a generalization of collection and aggregation types in this case. Authors of claims that CCUBE constraint calculus guarantees polynomial time data complexity.

We should emphasize here that the constraint calculus is based on monoid comprehensions. To describe monoid comprehensions is therefore the basic task to do.

### 2.4 A Graph-Oriented Object Database Model—GOOD

Gyssens et. al [19] introduced GOOD—a graph-oriented object database model as a theoretical basis for object databases. In the GOOD model the database scheme, as well as the instances, are represented by a graph. The data manipulation is by a graph transformations.

From the categorical point of view we have to address graph structures with the transformations.

### 2.5 Query processing with description logic ontologies over object-wrapped databases

In 2002 in [25] was presented an approach to answering queries over an ontology modelled using a description logic. The ontology is used as a global schema, which
provides a declarative description of the concepts of the domain, the instances of which are stored in (potentially many) object-wrapped sources.

Queries are expressed here using terms from the vocabulary of the ontology, and are translated into an equivalent calculus expression, which references only the objects available in the source databases. The query is then optimized on the basis of information from the ontology and the source databases.

This approach uses of the expressive $\mathcal{ALCQI}$ description logic, which supports both ontology definition and query expression. Furthermore uses the adoption of a global-as-view approach to relating the ontology to the sources. Finally this approach is based on the use of the ontology to direct semantic optimization of queries phrased over specific sources.

We note here that the approach is being developed in, and is illustrated using examples from, bioinformatics.

Not suprisingly the query itself and query optimization process is again based on monoid comprehension calculus.

### 2.6 Tractable Query Languages for Complex Object Databases

In 1999 in [17] was considered the expressiveness and complexity of several calculus-based query languages for complex objects. Concerning with the complexity of queries on databases of complex objects raises new issues specific to complex objects. For instance, it is shown that the way the database makes use of its higher-order types has direct impact on query complexity.

The languages for object queries can have the form of:

- extension of relational calculus,
- extension of relational algebra,
- deductive languages.

All of them use higher-order types. The work [17] is based on extensions based on relational calculus.

### 2.7 Translating Object Query Language

The paper [20] investigated implementation means for object query language OQL. The formal background dwell in universal algebra, monoids (collection monoid and primitive monoid), and monoid comprehensions.

Again, from the categorical point of view, it is necessary to find meaningful representation of monoid comprehensions and find out their interrelationships.

### 2.8 Monad approach

Investigating aspects of homological algebra in the 1950s, category theorists uncovered the concept of a monad, which among others generalizes set, bag, and list types.
Investigating programming languages based on lists in the 1970s, functional programmers adapted from set theory the notion of a comprehension, which expresses iteration over set, bag, and list types.

In the early 1990s, a precise connection between monads and comprehension notation was uncovered by Wadler [31].

Eugenio Moggi has applied monads [24] to describe properties of programming languages particularly using lambda calculus. Syntax of comprehensions is described in detail in [6].

The result of these works we can among other results see in the NRA—Nested Relational Algebra [7].

The work [16] uses all these ideas to develop the formal model for semi-structured data, XML documents particularly. From our (categorical) point of view, monads are endo-functors.

In the next section we treat of categorical models of formalisms and models of object databases. First we explain some basic terms from the category theory. Subsequently we describe known uses of category theory in the field of databases, especially in the area of object databases.

3 Categorical Approach

Our approach is quite different from all the other approaches described in the last section. We use mathematical category theory as a formal framework serving to describe not only object databases but also to describe other database models. We have focused on XML databases in [28] among others.

3.1 Category Theory Primer

Definition 1. A category \( \mathcal{C} \) consists of

- objects (denoted by \( A, B, C, \ldots \)) and
- morphisms between them (denoted by \( f : A \to B, g : B \to C, \ldots \)).

These data are subject to obvious axioms expressing composition, its associativity, and existence of identity morphisms (units w.r.t. composition).

A paradigm category is the category \( \text{Set} \) of all sets and mappings. See [2] for more details.

Note here that the morphism is usually called arrow.

Definition 2. We call an object 1 of a category \( \mathcal{C} \) terminal iff there is exactly one arrow \( A \to 1 \) for each object \( A \) of \( \mathcal{C} \).

Now we define the concept of a cartesian closed category (usually denoted as CCC). It is proved [22] that CCC’s are essentially the same thing as simply typed \( \lambda \)-calculus. Although the following definition is rather technical, one may bear in mind that the category \( \text{Set} \) forms a paradigm example of a CCC.
Definition 3. A category $C$ is called a cartesian closed category (CCC) if it satisfies the following:

1. There is a terminal object $1$.
2. Each pair of objects $A$ and $B$ of $C$ has a product $A \times B$ with projections $p_1 : A \times B \to A$ and $p_2 : A \times B \to B$.
3. For every pair of objects $A$ and $B$, there is an object $[A \to B]$ and an arrow $\text{eval} : [A \to B] \times A \to B$ with the property that for any arrow $f : C \times A \to B$, there is a unique arrow $\lambda f : C \to [A \to B]$ such that the composite

$$\lambda f \times A \quad \text{eval}$$

$C \times A \longrightarrow [A \to B] \times A \longrightarrow B$ is $f$.

Definition 4. We define a functor as a function $F$ with polymorphic argument which translate object from one category to an object in another category and the same in the case of arrows, but preserving composition, its associativity, and identities. More precisely:

The domain of the function is a category $C$ and codomain is another (perhaps the same) category $D$. Therefore we write $F : C \to D$. The argument is either an object $A$ of $C$ and then $F(A) = B$, where $B$ is an object of $D$ or the argument is an arrow and then $F(f) = g$, where $f$ is an arrow of $C$ and $g$ is an arrow of $D$, where $f : X_1 \to X_2$ and $g : Y_1 \to Y_2$, and $F(X_1) = Y_1$, and $F(X_2) = Y_2$, and if $g \circ f$ is defined in $C$, then $F(g) \circ F(f)$ is defined in $D$, and finally $F(g \circ f) = F(g) \circ F(f)$.

Definition 5. Let us define the comprehension as follows

$$[e_0 | x_1 \gets e_1] = \text{for } x_1 \text{ in } e_1 \text{ do } e_0$$

Definition 6. The monad has to satisfy three laws.

First left unit law: $\text{for } v \text{ in } e_1 \text{ do } e_2 = e_2 \{ v := e_1 \}$, where $e_1$ is an unit type (e.g., is an element or a scalar constant). identity.

Second the right unit law: $\text{for } v \text{ in } e \text{ do } v = e$

Third the associative law:

$$\text{for } v_2 \text{ in } (\text{for } v_1 \text{ in } e_1 \text{ do } e_2) \text{ do } e_3 = \text{for } v_1 \text{ in } e_1 \text{ do } (\text{for } v_2 \text{ in } e_2 \text{ do } e_3)$$

3.2 Known Uses of Category Theory for Object Databases

Categorical Framework for Object-Oriented Database Model

Complicated situation arose in the area of object-oriented databases. Kolenčík in his thesis [21] developed categorical framework for object-oriented databases. This framework is nevertheless a framework for the meta-metamodel. He evolved the theory of objects, models, meta-models and meta-metamodels. Furthermore he is using UML as both, as a notation, and as a visual part of his framework itself.

We believe this work is too wide and elaborated and so far from practical use that we cannot use this directly. That is the basic reason we propose several different description in the next section.
Object Databases and the Semantic Web Another sophisticated approach can be seen in the work of Güttner [18]. His work aims to the basics of object databases and semantic web. He uses several models including the LDM of Kolenčík.

We do not treat this work in greater detail because its core lies in ontologies which are generalization of graphs as it is the category of graphs.

The Product Data Model In [26] is described categorical model called the Product Data Model suitable for object-relational databases of these properties:

1. a clear separation between extension (an object) and intension (a class),
2. encapsulation,
3. an orthogonal definition language for functions within a class,
4. constraints on class structures similar to e.g. BCNF,
5. the standard information system abstractions,
6. message passing facilities between methods,
7. a query language which provides results with closure, and finally
8. multi-level architecture compliant with ANSI/SPARC standards.

Again the Product Data Model is complicated enough to not to be possible describe evidently valid relationships between different models. We therefore also omit detailed description here and later in this paper.

In case of further interest we encourage the reader to study [26] in detail.

In the next part we focus on categorical description of formalisms for XML.

3.3 Description of Formal Models of XML Databases

Most of the effort in XML databases were culminated in the work published at ADBIS 2008 [29]. We shortly summarize some interesting results here.

- The category of XML-λ is CCC.
  - XML-λ is introduced e.g. in Loupal’s work[23].
  - That any λ-calculus is CCC is proved e.g. in Lambek and Scott [22].
- $\mathcal{G}$raph is CCC
  - from e.g. [2]
- $\mathcal{T}$ree of trees is NOT CCC, because
  - there is not true the basic condition — existence of terminal object.
- category of HFS is CCC.
- The properties of a category of HNR are not known.

$\mathcal{T}$ree

- objects: tree (XML seen as a tree)
- morphisms: trees homomorphisms
- THERE IS NO Terminal Object ⇒ NOT a CCC
Formal Models of Object-Oriented Databases

$\mathcal{G}raph$
- **objects**: graphs (XML seen as an oriented graph)
- **morphisms**: graph homomorphisms
- $\mathcal{G}raph$ is CCC, as was proved e.g. in [2].

$\lambda$ML
- Given a typed $\lambda$-calculus $L$
- **objects**: types of $L$
- **morphisms** (arrows $A \rightarrow B$): equivalence classes of terms type $B$ with one free variable of type $A$
- $\lambda$ML–$\lambda$ is a CCC when applying proofs from [22]

$\mathcal{C}HFS$
- $\mathcal{C}HFS$ and $\mathcal{F}orrest$.
- **objects**: sets with nested sets
- **morphisms**: operations on sets
- $\mathcal{C}HFS$ is proved is a CCC [10]

Next section deals with categorical models of object databases we propose to use or just consider. Some models are not useful in praxis but they are a part of a testimony of what kind of point of view bring us necessary portion of usefulness.

4 Proposed Object-Oriented Formal Models

We define categorical model of object data model from different perspective.

**Definition 7.** A category $\mathcal{O}bj_L$ of all objects of a class $L$ consists of

- the collection of **objects**: instances
- the collection of **morphisms**: all functions which domain and codomain is an instance of a class $L$.

The problem with the definition above is that the morphism do not necessarily correspond to the operation of that class; neither static nor dynamic operations. But it could. This is strongly dependent on how the objects are interconnected, what methodology is used etc.

The identity, composability and associativity to verify is trivial. But to verify that there is a terminal object, cartesian products and the closedness property in this category is harder. That fact is not known till now.

**Definition 8.** A category $\mathcal{O}bjects$ of all categories of classes $\mathcal{O}bl_L$ consists of

- the collection of **objects**: categories $\mathcal{O}bl_L$
- the collection of **morphisms**: all functors between these classes.
These definitions are basically similar in Kolencik’s work and here. The meta-metamodel and others we can define in several ways. It seems to be very hard to prove CC’s property of above mentioned models.

Obviously another approach dwells in definition of a category of all objects of all classes. Which can be represented as general graph.

Definition 9. A category $\mathbf{Graph}$ of all graphs consists of

- the collection of objects: graphs
- the collection of morphisms: graph homomorphisms.

Verify that the definition above is a category is trivial. In [2] was proved that this category have the property CC. It is very interesting and even promising coincidence that the category of relations is also CCC. Is there any model of XML Database which would have the CC property? Would be possible to say that there are models of all three data model types and that their models are CCC?

Next section concludes this paper summarizing obtained results and proposed solutions.

5 Conclusions

Database modeling involves XML, Object, Relational, Hierarchical, and Network-based models. All these models can be seen from the point of view of category theory. In this paper we focus on object models and formalisms, their summary and categorical description. Several formal models, descriptions, and properties are introduced.

Categorical approach to the database modeling allows us to describe models themselves, as well as categories of these models. Furthermore we can describe, using category theory, another formal models and specifications. In addition it is possible to describe properties of these (existing) models each other.

In the next section we will consider possibilities of elaboration of categorical way of thinking and models construction.

6 Future Works

There is a lot of evidences which have to be explored and considered into the whole categorical framework. There are still new models and ideas which are needed to be ponder.

In the near future we plan to describe known formalisms of Relational, Object, and XML database paradigms using category theory. Then catalogize them and find out the methods of their transformations.

In the far future we plan to establish formal basis for interrelationships description.

MDA—Model-Driven Architecture—MDD—Model-Driven Development respectively, can be viewed from the category theory viewpoint. Conceptual and
logical models and their properties could be described using one unifying framework. Category Theory appears to be strong enough to gain this aim. We plan to develop unified framework for these different types of models based on category theory.

References