Implementing an Object-Oriented Deductive Database Using Temporal Reasoning

Nihan Kesim
Bilkent University

Marek Sergot
Imperial College

A general approach for temporal reasoning, the event calculus, has been modified and applied to the development of a historical deductive object base. The event calculus is a theory of time in first-order logic augmented with negation as failure. It is shown how an object-based variant of the event calculus may be used for representing changes to the states of objects. We first present the formulation and describe the maintenance of a historical object-oriented database by the use of events. The resulting formalization can be executed as a logic program. We then discuss the implementation of a practical database system based on the theory we develop. The additional detail needed to develop a system of realistic scale is outlined. The aim is to present the object-based event calculus as a unifying framework for the implementation of a deductive and object-oriented database system.

Deductive object-oriented databases are the most recent research area falling within the intersection of logic and databases. They aim at providing both database designers and users with a modelling tool which is semantically richer than the relational data model as well as using logic as a tool for the formalization of both the static and the dynamic aspects of databases.

Conventional deductive databases, i.e. those which have the relational data model as the underlying data model, are well-understood and several deductive database system prototypes have been implemented (Ceri, 1990; Chimenti, 1990; Ramamohanarao, 1988; Vieille, 1991). The transition from relational databases to deductive databases was smooth and natural because the relational data model has a simple and well-defined expression in terms of first-order logic (Gallaire, 1978). Therefore coupling a relational database to a logic programming language, such as Prolog, was natural and the result was an enhancing of the expressiveness of relational databases with deduction.

The perspective is totally different if object-oriented databases are considered instead. There is a conceptual mismatch between logic programming languages and the notion of object as imported from object-oriented programming languages. This conceptual mismatch raises several difficult problems that must be tackled for deductive object-oriented databases to be seen as a step forward. Nevertheless, deductive object-oriented databases have become a focus of interest because current deductive databases are based on the relational data model, and as a consequence they inherit the modeling shortcomings of the relational model for supporting non-business applications such as design activities and graphic data manipulation.

There have been many proposals on combining the deductive and object-oriented approaches. Some of the current work take deductive databases as the basis and extend the existing systems with some subset of object-oriented features (Abiteboul, 1988; Chimenti, 1990; Kuper, 1987; Zaniolo, 1985). Others take object-orientation as the basis and logic as the framework, and try to formalize object-oriented data modelling (Ait-Kaci, 1986; Chen, 1989; Kifer & Wu, 1989; Kifer & Lausen, 1989). There is also another stream of work which approaches the problem from a programming language point of view. The aim is to combine object-oriented programming and logic programming by extending the existing declarative languages (e.g. Prolog, Datalog) with object-oriented notions like methods, message passing and inheritance (Dalal, 1989; Fukunaga, 1986; McCabe, 1988). A detailed discussion of all these different approaches is presented in Fernandes (1993); Kesim (1994, 1993).
Current proposals mainly deal with structural aspects of objects by defining semantics of basic object-oriented features in a logical framework. However, there is little work on representing and dealing with the dynamic aspects of objects, such as state changes of objects. It is unclear how changes to object states, creation and deletion of objects, and changing the class of objects can be described in a deductive and object-oriented framework. In this paper we take a different approach by combining two kinds of models. We consider not only the structural aspects of objects but also their dynamic aspects. We use an object model to represent the structure of objects, and use a dynamic model to represent the behaviour of objects over time.

We have developed the object-based event calculus (OEC in short) to combine these two views of modelling the system. The OEC is a variant of the event calculus of Kowalski and Sergot (Kowalski, 1986), which is a theory for reasoning about time and change within a logic programming framework. The OEC is used to describe various temporal aspects of objects. Change is formulated in the context of a historical database which stores implicitly all past states of objects in the database. It is possible to determine which objects “exist” at which times and derive any past state of an object using the axioms of the OEC and a given set of event descriptions. It is also possible to keep and reason about different versions of an object at a time.

The formulation of the OEC has been presented in Kesim (1992) and the main idea has been extended to explore other temporal aspects of objects and classes, such as versioning of objects and schema evolution (Kesim & Sergot, 1993; Kesim, 1994). In this paper we mainly discuss the implementation aspects of a practical system based on the proposed framework. A direct coding of the OEC as a logic program has some efficiency problems due to the reasoning behind the theory. It generates a large search space and redundant computations in the evaluation of queries. These problems can be solved by a more sophisticated implementation which makes use of the derived conclusions. Instead of deriving the states of objects only when they are needed, all known facts about the objects can be stored in an extensional database and query evaluation can be performed on this database. In this paper we present such an implementation of the OEC to be used as a basis for a practical temporal deductive object base. We describe a conceptual architecture of such a system where the persistent storage of objects can be maintained as a relational database and an object-oriented layer can be built on top as an interface.

The database which keeps the history of objects will be very large in size. The problems encountered in implementing such a system is closely related to the problems in implementing temporal relational databases. Many efficient storage and indexing techniques have been proposed for temporal relational databases (Snodgrass, 1986; Tansel, 1993). These techniques can be applied in the implementation of the persistent object store. For instance in Jensen (1991), an implementation model is presented for the standard relational data model extended with transaction time where updates to relations are entered as time-stamped change requests into backlogs. The backlogs of Jensen et al. are very similar to the structure of our object store. However, the actual application of these techniques to the OEC is left as a future work.

The rest of the paper is organized as follows. Section 2 defines the basic modelling framework in which events and objects are related. In Section 3 we summarize the formulation of the object-based event calculus and discuss how it can be applied to describe different temporal aspects of objects including versioning of objects. Section 4 discusses the computational issues related to the implementation and identifies some efficiency considerations. Section 5 gives an overview of the architecture of a practical database system based on the OEC. In Section 6 we discuss the usage of the lemma generation approach in implementing an external database to store objects. We describe an efficient query evaluation technique and discuss the maintenance of the underlying database every time a new event is recorded into the system. Section 7 presents an evaluation of the current implementation and we conclude the paper in Section 8.

Events and Objects

We combine two models to describe a deductive object-oriented database: a dynamic model and an object model. The dynamic model is used to describe those aspects of the system concerned with time and change. The object model, on the other hand, describes the static structure of objects in a system - their identity, their relationships to other objects, their attributes, and classes.

The major dynamic modelling concepts are events, which represent external stimuli, and states, which represent values of objects. An event is something that happens at a point of time, such as user depresses left button or flight 123 departs from Chicago. An event has no duration. Of course, nothing is really instantaneous; an event is simply an occurrence that is fast compared to the granularity of the time scale of a given abstraction. A state, however has a duration; it occupies an interval of time. Events and states are duals of one another; an event separates two states, and a state separates two events. Thus a state is assumed to persist into the future until it is terminated by an event.

The state of objects is described by attributes which hold data values or identities of other objects. A state specifies the response of the object to input events. A state change corresponds to changing the value of any of the attributes of the object. For instance, a person moves to a new place, the value of the address attribute changes; if a car is painted, the color attribute changes accordingly. Thus, the state of an object depends on the past sequence of events that affects it.

We take a simple object model to describe the state of
Related Content

A Database Project in a Small Company (or How the Real World Doesn't Always Follow the Book)
www.igi-global.com/chapter/database-project-small-company-real/7927?camid=4v1a

Labelling-Scheme-Based Subgraph Query Processing on Graph Data
Hongzhi Wang, Jianzhong Li and Hong Gao (2012). Graph Data Management: Techniques and Applications (pp. 142-174).
www.igi-global.com/chapter/labelling-scheme-based-subgraph-query/58610?camid=4v1a

An MDA Compliant Approach for Designing Secure Data Warehouses
www.igi-global.com/chapter/mda-compliant-approach-designing-secure/7935?camid=4v1a

Collaborative Information Management System for Science Domains
www.igi-global.com/chapter/collaborative-information-management-system-science/7983?camid=4v1a