Deductive Object Databases

Elisa Bertino	Giovanna Guerrini
Dipartimento di Scienze	Dipartimento di Informatica e
dell'Informazione	Scienze dell'Informazione
Università di Milano (Italy)
bertino@hermes.mc.dsi.unimi.it
Università di Genova (Italy)
guerrini@disi.unige.it

Danilo Montesi *
Departamento de Informatica
Universidade de Lisboa (Portugal)
danilo@di.fc.ul.pt

Abstract

This paper proposes a new approach to model deductive object databases. Each object database is described by means of a Datalog language extended with extensional updates, called U-Datalog. Each object can change its state and cooperate with other objects. We introduce an extension of U-Datalog to approach the problem of composition among object databases. It can be used for modular database design and for cooperation among databases. The resulting language has a clear semantics for the evolution of objects and for modeling the transactional behavior of the resulting database. Finally, we describe some architectural issues of the prototype which has been developed.

Keywords: Logic languages, object based paradigm, composite databases, transactional behavior.

1 Introduction

In the last few years the object-oriented paradigm has being widely used in several areas of computer science, such as programming languages, databases, software engineering, user interfaces. In particular, the notion of modularity, which is already widely accepted in the programming language field, is an important notion also in the database field. The need for considering federated databases rather than a centralized one is motivated by several issues. The role played by databases in Artificial Intelligence, when an expert system requires
processing large volumes of data and rules, is one of them. In the specific case of a distributed expert system the distribution of data/rules requires to consider the distribution of the database and thus the development of cooperative databases. Moreover for some complex application domains (like, e.g. the Computer Integrated Manufacturing applications), the database system is inherently distributed. Many applications integrating and using data and services of local database systems [13] are designed and needed. In all cases, the specification of a federated system presents strong analogies with the specification of composite (database) systems. In [22], such concerns motivate the identification of "objects" each of them incorporating some knowledge (like facts, rules, and constraints) and being a unit of design that can be composed with other objects.

In this paper we consider the above problem in the context of deductive databases. The motivation for this choice is related to the formal model underlying deductive databases, which provides a formal behavior and also a computational model. Note that we will consider an extension to Datalog considering updates and transactions. The language in which our modules are expressed is in fact Update Datalog (U-Datalog) [10], an extension of Datalog which allows the specification of updates in rule bodies. The relevant characteristic of U-Datalog is that updates are not executed as soon as they are evaluated, rather they are collected in a set and executed altogether at the end of the refutation process, if this process succeeds and the set is ground and consistent (i.e. it does not contain complementary updates on the same fact).

The language we propose in this paper (called Obj-U-Datalog) is based on the notion of object. Each object is an U-Datalog database. Each object has a state (a set of facts) and a set of methods. Methods may also contain update atoms to modify the object state. In such a way we model objects with OIDs, state, behavior and state evolution. Moreover the computational model of our language is based on cooperation among objects through message passing. In a conventional deductive database, in fact, the evaluation of a query may make use of all facts and rules in the database. In an Obj-U-Datalog database, by contrast, facts and rules are grouped in objects, and the evaluation of a query is performed in a specific object. During the evaluation process an object may request the evaluation of a subquery to another object (through a message call), in such a case the computation moves to another object (we will also say that a context switch has happened).

Our approach to deductive object databases relates to previous work on logical object databases as follows. Some of these approaches do not consider state evolution of deductive objects [12, 19, 27, 28, 24, 2, 15, 21, 23]. Others consider state evolution [5, 16] but objects have the granularity of terms. By contrast we consider an object with larger granularity, that is a theory, i.e. a set of logical clauses. Moreover many of the considered approaches do not consider the behavioral component of objects (i.e. the methods). We think that this is an important issue because it overcomes the dichotomy between data and opera-