A Logic Framework for a Semantics of Object Oriented Data Modelling

O. De Troyer
Tilburg University, Infolab
P.O.Box 90153
NL-5000 LE Tilburg
The Netherlands
Phn: +31-13-662432
e-mail: detroyer@kub.nl

R. Meersman
Free University of Brussels (VUB)
Computer Science Department
Pleinlaan 2 --Bldg. FG/10
B-1050 Brussels, Belgium
Phn: +32-2 629 3819
e-mail: meersman@vub.ac.be

Abstract
We describe a (meta) formalism, called Data Modelling Logic (DM logic), for defining a variety of (object oriented) data models in a unified framework based on first-order logic. Using NORM, an OO model, we illustrate how essential OO properties such as information hiding, encapsulation, inheritance and behavior may be generically described, as well as the fundamental distinction with object-oriented programming, namely persistence. A formal semantics for these concepts can so be given independently of the chosen data model. DM logic has been demonstrated in earlier work to adequately support "classical" data models such as (E)ER, NIAM, and the Relational Model, and so-called lossless transformations between them. By "programming" an OO data model in DM Logic, it should become possible to arrive at objective relationships between (OO and other) data modelling techniques.

1 Introduction
As it becomes increasingly important to connect heterogeneous information systems with the purpose of interoperation, it becomes necessary to deal within the same context with different specification techniques and methodologies. This in turn indicates the need for a common formalism in which one can define different and multiple modelling and representation languages.

The approach we follow is based on a first-order logical formalism, called DM-Logic (Data Modelling Logic), an extension and refinement of the one used in [3]. There it is shown how one may describe NIAM [12], (E)ER [9] and the "vanilla" relational data model within this same representation. In this paper we extend the formalism with a specification method for the "dynamics" of a conceptual schema and demonstrate that it is possible to express encapsulation and information hiding.

We have deliberately adopted a model-theoretic approach [10] although a proof-theoretic approach could perhaps have yielded a simpler and more uniform notation as is e.g. apparent in F-logic [7]. We felt however the extra mile was justified as classical databases and information systems almost exclusively adopts a model theoretic approach; this is appreciated in general as more intuitive because of the "natural" distinction of schema and "data". For the same reason the description of behaviour through methods is done using an imperative language involving "classical" instruction such as assignment, choice, loops, etc.. But we shall pay only token attention to these instructions as we want to concentrate rather on the distinction between instructions that cause a state transition and those that do not.
To illustrate the formalism, we describe it by means of an existing object-oriented specification language NORM [2]. It is not our goal here to (re)define NORM but rather to illustrate the descriptive power and principles of DM-Logic. Therefore a number of simplifications will be adopted.

We also do not have the ambition, at least within the scope of this paper, to either achieve deep results in logic or fundamental new results in OO modelling itself. Rather we pursue what we believe is a new and consistent approach in (meta-)modelling that allows to describe exactly and in a comprehensive manner the many complex aspects of "real-life" data modelling, especially object oriented, in a unifying framework that is semantically sound.

The paper is organized as follows. First we give a small introduction to NORM. Then we describe NORM in terms of DM Logic and give the semantics of a NORM-schema as well as an OT-schema; in particular the concept of state is introduced as the locally defined realization of persistency. Finally an OB-schema is defined as an "application-dependent" schema having a "global" state and which "uses" OT-schema's in a well-defined aggregation; we show that the local states can be made to behave properly in this aggregation, without losing information hiding.

2 A Tourist's Introduction to NORM

NORM is intended for conceptual modelling and is an object oriented version of the binary Object Role Model (ORM) (e.g. [6], [4], also known as NIAM [12]). NORM combines the assets of ORM with the advantages of object orientation.

Contrary to most OO models which define the state of an Object Type (OT) by means of attributes, the state of an OT is defined by a complete schema, the so-called Object Type schema (OT-schemas). The OT defined by this schema is called the focus. As an example consider the OT-schema with focus Document, its graphical representation is given in fig. 1 and reads as follows: Each Document has a Size and may have a Title. A Document may contain several Keywords. A Document also has a maximal Size which is the same for all Documents. The Size is expressed in bytes (ByteSize). ByteSize and Title are lexical types. Document, Size and Keyword are non-lexical types. Size is a local type while Keyword refers to the OT-schema with focus Keyword. Fig. 2 gives the graphical representation of the Keyword-schema. This schema also refers to a third OT-schema Topic which is not shown here.

Behavior is specified by means of methods. Example 1 shows the (type-)method create_empty for Document. NORM uses pre- and post-conditions to express the semantics of a method. A body may also be given to express the procedural semantics but this is not needed. NORM allows to construct methods using role names and type names in a well-defined way, e.g. Keyword-appearing_in to obtain all instances of Keyword which are related to a Document instance by means of the role appearing_in. NORM also supports relation methods and triggers. Relation methods are methods which are defined for relations instead of for object types. The corresponding messages are sent to relation instances. Triggers allow to express dependencies between the activation of messages. Due to lack of space, they are not treated in this paper.

To make methods and possibly also relations visible outside the OT-schema in which they are defined, they should explicitly be exported. Exported concepts may be