ABSTRACT

While some information modeling approaches (e.g., the Relational Model and Object-Role Modeling) are typically formalized using first-order logic, other approaches to information modeling include support for higher-order types. There appear to be three main reasons for requiring higher-order types: (1) to permit instances of categorization types to be types themselves (e.g., the Unified Modeling Language introduced power types for this purpose); (2) to directly support quantification over sets and general concepts; (3) to specify business rules that cross levels/meta levels (or ignore level distinctions) in the same model. As the move to higher-order logic may add considerable complexity to the task of formalizing and implementing a modeling approach, it is worth investigating whether the same practical modeling objectives can be met while staying within a first-order framework. This chapter examines some key issues involved, suggests techniques for retaining a first-order formalization, and makes some suggestions for adopting a higher-order semantics.

INTRODUCTION

This chapter evaluates the advisability of using higher-order types in information models. In case the reader is untutored in logic, we first clarify the notion of higher-order type. First-order logic quantifies over individuals only, not predicates. For example, the constraint “Each Person was born on at most one Date” may be formalized in first-order typed logic thus:
\[ \forall x: \text{Person} \exists ^{0..1} y: \text{Date} \; x \text{ was born on } y \]

Here the types “Person” and “Date” appear as unary predicates, “\( \forall \)” is the universal quantifier (“for each”) and “\( \exists ^{0..1} \)” is the “there exists at most one” quantifier. The quantification is over individual people and individual dates. Second-order logic allows quantification over first-order predicates (as well as individuals). For example, the rule “Each asymmetric predicate is also irreflexive” may be formalized in second-order thus:

\[ \forall R: \text{Predicate} (\text{Asymmetric } R \rightarrow \text{Irreflexive } R) \]

Third-order logic allows quantification over second-order predicates and so on. All logics whose order is above first-order are called higher-order logics. Since types may be formalized by unary predicates, if an instance of type \( A \) is itself a type, then \( A \) is a higher-order type.

Codd (1970) used first-order logic to formally underpin the relational model of data. Subsequently, most formalizations of information modeling approaches restricted their logical foundations to first-order. This is the case for Entity Relationship (ER) modeling (Chen, 1976), as well as Object-Role Modeling (ORM) and its variants (e.g., Bakema, Zwart & van der Lek, 1994; Halpin, 1998; Halpin, 2001). A full formalization of ORM’s fact-oriented (attribute-free) approach to information modeling was first provided by Halpin (1989), with alternative formalizations supplied later by De Troyer and Meersman (1995) and ter Hofstede, Proper, and van der Weide (1993). In contrast, the Unified Modeling Language (UML) introduced the notion of powertypes, whose instances may themselves be types, thus requiring higher-order semantics (OMG, 2003a; OMG 2003b; Rumbaugh, Jacobson & Booch, 1999).

There appear to be three main arguments for requiring higher-order types to logically underpin information modeling semantics:

- to allow one to think of instances of certain categorization types (e.g., AccountType, CarModel) as being types themselves (as for UML powertypes);
- to formalize very directly the semantics of flexible data structures where attribute entries may themselves denote sets or general concepts (e.g., object-relational tables in nonfirst normal form); and
- to allow one to specify business rules that seem to cross levels/meta levels (or ignore level distinctions) in the same model (e.g., the Finance department is responsible for defining the possible values of AccountType).

As the move to higher-order logic may add considerable complexity to the task of formalizing and implementing a modeling approach, it is worth investigating...