Towards a Formalization of Variability in Conceptual Models of Information Systems

Jan Paredaens Jan Hidders Jan Verelst
University of Antwerp
jan.{paredaens,hidders,verelst}@ua.ac.be

Abstract

Building conceptual models is an increasingly important part of information systems development. It is well-known that, for a given set of data requirements, multiple data models can be designed. From both theoretical and practical viewpoint, it is relevant to define whether these models can be considered equivalent. Therefore, we describe an approach based on formally defining the term schema equivalence indicating that two data models contain the same information; a schema is considered dominant if it contains the same or more information than another schema.

1 Introduction

In the analysis phase of information systems, conceptual models are built of the relevant part of the real world (also called Universe of Discourse - UoD). We use the term variability to indicate that, of a certain UoD, different conceptual models can be built in a given modeling technique, such as UML class diagrams. In this paper, we will refine the rather broad term variability by defining the term equivalence of data models. Data models are considered equivalent if they contain the same information, as illustrated in Fig. 1.

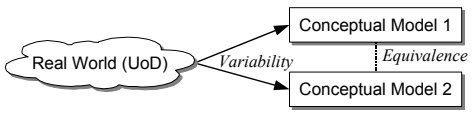

Figure 1: Variability and Equivalence

Verelst [**?**] proposed a framework containing three types of variability that are relevant to studying the evolvability of information systems. However, the definitions of these types are rather informal, and are therefore to a certain extent vague. Construct variability occurs when a given concept in the UoD is modeled using different constructs of the modeling technique. An example is when the price of a book is modeled as an attribute versus an entity. Vertical abstraction variability occurs when concepts in the UoD are modeled at different levels of abstraction (or genericity). For example, two types of employees can be modeled using a subtype hierarchy or using meta-modelling as illustrated in Fig. 2. It can be argued that variant B is more generic than variant A because certain changes in the UoD can be represented in the model without needing change. For example, the addition of a third type of employee requires changes to variant A but not to variant B. Horizontal abstraction variability occurs when concepts in the UoD are modeled using different properties. For example, a set of people and animals are first modeled in entities Human and Animal, and secondly as Red-eyed creatures and Brown-eyed creatures.

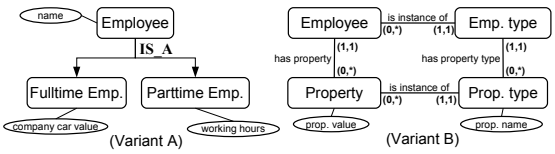

Figure 2: Vertical abstraction variability

This paper constitutes a first step in our attempt to provide more precise, formal definitions of these three variability types. In this first step, we define a way to compare the information content in two or more conceptual models. We formalize this in the notion of schema equivalence. Schemas are considered equivalent when they contain the same information; a schema is considered dominant if it contains the same or more information than another schema. Equivalence can be used to prove construct and horizontal abstraction variability. Dominance can be used to prove vertical abstraction variability.

2 Formalisation

We begin with the introduction of a very simple and general datamodel that is based on graphs, and then we introduce the formal notion of ontological dominance for this datamodel.

The first notion we introduce is that of *schema graph* which defines the structural skeleton of a schema, i.e., it defines which classes and (binary) relations are present and how they are connected. A *schema graph* is formally defined as a tuple $G = (C, R, s, t)$ where C is a finite set of *class names*, R a finite set of *relation names* disjoint with C and $s : R \to C$ and $t : R \to C$ the functions that give the *source class* and *target class* of a relation, respectively. A schema graph $G_1 = (C_1, R_1, s_1, t_1)$ is said to be a *subgraph* of a schema graph $G_2 = (C_2, R_2, s_2, t_2)$ if $C_1 \subseteq C_2$, $R_1 \subseteq R_2$, $s_1 \subseteq s_2$ and $t_1 \subseteq t_2$.

The next definition is that of *schema graph instance* which defines how an instance of a schema graph will look. It consists of a set of objects and two interpretation functions. The first interpretation function maps all the class names to the extension of the corresponding class, i.e., the set of objects that are members of this class. The second interpretation function maps relation names to the extension of the corresponding relations, i.e., the set of pairs of objects that are in that relation. An *instance* of a schema graph $G = (C, R, s, t)$ is formally defined as a tuple $I = (O, \mathcal{C}, \mathcal{R})$ where O is a finite set of objects, $\mathcal{C} : C \to 2^O$ the interpretation function for class names and $\mathcal{R} : R \to 2^{O \times O}$ the interpretation function for relation names such that $\mathcal{R}(r) \subseteq \mathcal{C}(s(r)) \times \mathcal{C}(t(r))$.

Two instances $I_1 = (O_1, \mathcal{C}_1, \mathcal{R}_1)$ and $I_2 = (O_2, \mathcal{C}_2, \mathcal{R}_2)$ of the same schema graph $G = (C, R, s, t)$ are said to be *isomorphic* if there is a one-to-one function $h : O_1 \to O_2$ such that (1) for all $c \in C$ it holds that $\mathcal{C}_2(c) = \{h(o) \mid o \in \mathcal{C}_1(c)\}$ and for all $r \in R$ it holds that $\mathcal{R}_2(r) = \{(h(o_1), h(o_2)) \mid (o_1, o_2) \in \mathcal{R}_1(r)\}$.

Now we introduce the notion of *schema* which consists of a schema graph that roughly defines the allowed instance plus a set of constraints that should also hold for instances of the schema. How constraints are exactly specified is left open for the moment, we only specify that they are predicates which either do or do not hold for instances of the schema graph. A *schema* is formally defined as a pair $S = (G, K)$ where G is a schema graph and K a finite set of constraints which are unary predicates defined over instances of G. An *instance* of this schema is then formally defined as an instance of G that satisfies all the constraints in K.

The central problem we need to tackle is that the equivalence depends on certain background knowledge such as the fact that if an object in the class *Creature* has a *has* relationship with an *Eye_Color* object that in turn has a *name* relationship with the value "red" then the *Creature* object is an instance of the class *Red_Eyed_Creature*. We will assume that this knowledge is modeled in a global schema in which all these classes are represented and for which constraints are specified that express this fact. So if we ask the relationship between two schemas then we ask this with respect to a common super-schema.

Given a subgraph G' of schema graph G it is easy to see that given an instance I of G we can restrict it to the classes and relations in G' and obtain an instance of G'. This restriction of I to G' is denoted as $I[G']$.

Finally we define what *ontological dominance* means. Given a schema $S = (G, K)$ and two schema graphs G_1 and G_2 which are subgraphs of G we say that G_1 *dominates* G_2 if for each two instances I_1 and I_2 of S it holds that if $I_1[G_1]$ and $I_2[G_1]$ are isomorphic then so are $I_1[G_2]$ and $I_2[G_2]$.

3 Applications

The future results of our research are useful in several contexts. First, Halpin and Proper [**?**] study the problem of equivalent database schemas in order to improve database performance at conceptual, logical and internal levels (e.g. query response times). Second, our results could be used to check whether two modelers arrive at equivalent results when modeling a UoD, thereby improving the odds that the models are "correct" models of the UoD. A third, more practical application of our findings is conversion of databases. The conversion from one database to another can possibly be shown to be successful by proving that both databases are equivalent.

References

[1] T. Halpin and H. Proper. Database schema transformation and optimization. In *Lecture notes in computer science*, pages 191–203. Springer Verlag, 1995.

[2] J. Verelst. Variability in conceptual modeling. Technical Report Technical Report RPS-2004-019, University of Antwerp, 2004.