This article (further referred to as Math-I), and the next one (further referred to as Math-II, see p. 359), form a mathematical companion to the article in this encyclopedia on Generic Model Management (further referred to as GenMMt, see p. 258). Articles Math-I and II present the basics of the arrow diagram machinery that provides model management with truly generic specifications. Particularly, it allows us to build a generic pattern for heterogeneous data and schema transformation, which is presented in Math-II for the first time in the literature.

INTRODUCTION

Generic MMt (gMMt) is a novel view on metadata-centered problems manifested by Bernstein, Halevy, and Pottinger (2000). The main idea is to implement an environment where one could manipulate models as holistic entities irrespectively of their internal structure. It can be done only within an integral framework for abstract generic specifications of relations between models and operations with models. However, building truly generic specifications presents a problem of a new kind for the DB community, where such familiar tools as first-order logic or relational algebra cannot help. Amongst the most pressing gMMt problems listed in Bernstein (2003)—the most complete presentation of gMMt agenda to date—almost all are specification problems.

Fortunately, the community does not have to develop the desired framework from scratch. As is manifested in GenMMt, appropriate methodology and specification techniques are already developed in mathematical category theory (CT) and waiting to be adapted to gMMt needs. The goal of Math-I and Math-II is to outline foundations of the categorical approach to MMt and demonstrate how it works in a few simple examples.

Our plan is as follows. The next section of the article describes Kleisly arrows—the main vehicle of the approach. The section following it presents a simple example of model merge as a sample demonstrating how the categorical treatment of MMt procedures works. We begin by representing models in a graph-based format called sketch and describe a general pattern for sketch merging. After that, we reformulate the pattern in abstract terms to make it applicable to any sort of models, not necessarily sketches, although models are still supposed to be similar (be instances of the same metamodel). The last section summarizes this work by describing a generic arrow framework for homogeneous MMt.

The next step—managing models’ heterogeneity, particularly data and schema translation—is a highly nontrivial issue, a truly generic solution to which is often considered to be impossible (Bernstein, 2003). Nevertheless, it does exist and is presented (for the first time in the literature) in Math-II. Based on some category theory ideas, data and schema translation is specified in an abstract generic way via the so-called Grothendieck mappings. After that, homogeneous MMt specifications presented in Math-I can be considered as abbreviations for heterogeneous specifications, where arrows are interpreted as Grothendieck mappings. Particularly, the abstract pattern of homogeneous model merge developed in Math-I can be applied for heterogeneous merge as well. The last section of Math-II summarizes the results and outlines a general mathematical framework underlying generic MMt specifications.

MODEL MAPPINGS AS KLEISLY ARROWS

Do Model Mappings Really Map?

In the literature, the term “model mapping” usually refers to a specification of correspondence between models. It was noted by many authors that a general format of such correspondence must involve derived items of the models in question. In recent surveys, such as Lenzerini (2002), it became common to describe a mapping between schemas $S$ and $T$ as a set of assertions of the type $Q_S \sim Q_T$, where $Q_S$ and $Q_T$ are some queries over schema $S$ and $T$, respectively, and $\sim$ denotes some comparison operator between query outputs. It is assumed that queries (operations) are terms formed by strings of operation symbols and variables, and assertions are formulas, i.e., strings composed from terms and logical operators.

This framework is typical for string-based algebra and logic familiar to the community, but it has a few inherited
Mathematics of Generic Specifications for Model Management, I

drawbacks for generic MMt applications. First of all, string-based terms and formulas are inadequate for expressing queries and constraints over graph-based models like ER or UML diagrams. In addition, it is unclear how to define composition of mappings in this setting. Not surprisingly, in all concrete applications and implementations of this framework, it is specialized for working with relational models; see Fagin, Kolaitis, Miller, and Popa (2003); Madhavan and Halevy (2003); and Velegrakis, Miller, and Popa (2003).

A more general framework is proposed in Bernstein et al. (2000). The main idea is to reify mappings as models and to attach correspondence assertions to elements of these model mappings (see Figure 2 in GenMMt). In this way model mappings become spans (whose legs, i.e., projection arrows, are model morphisms), and their composition is defined as composition of spans. Though semantically clear, span composition is much more complicated than composition of arrows, and the problem of composing expressions still persists. A common drawback of both approaches is that queries appear as something foreign to the models and need a special interface (Problem 3 in Table 2 in GenMMt).

All these problems are eliminated as soon as we consider the issue in the generic framework of categorical algebra developed in CT in the 60th-70th (see Manes, 1976, for a basic account). The key observation is that operations/queries against the model can be denoted by expressions still persists. A common drawback of both approaches is that queries appear as something foreign to the models and need a special interface (Problem 3 in Table 2 in GenMMt).

In our abstraction efforts in the right column, we can go even further and consider query expressions attributed to arrows rather than to models, as it is shown by Diagram (b3). In this way we come close to the notation with which we began our discussion, but note that in our case a figurative arrow \( S \sim T \) is just an abbreviation of pair \((f, Q)\), with \( Q \) a set of queries to the target schema and \( f:S \rightarrow \text{der}^Q T \) a (functional) mapping of the source schema.

An essential feature of the specifications in Figure 1 is that the arrows are functional mappings that send each item from the source to a single item in the target. We do not need to reify mappings by new intermediate models nor do we need to consider many-to-many relationships between models. Although reification of model correspondence by intermediate models (spans) is sometimes necessary, many MMt tasks can be specified with pure functional mappings (morphisms). From now on, we will use the term model mapping in the sense of functional mapping between models.

In our abstraction efforts in the right column, we can go even further and consider query expressions attributed to arrows rather than to models, as it is shown by Diagram (b3). In this way we come close to the notation with which we began our discussion, but note that in our case a figurative arrow \( S \sim T \) is just an abbreviation of pair \((f, Q)\), with \( Q \) a set of queries to the target schema and \( f:S \rightarrow \text{der}^Q T \) a (functional) mapping of the source schema.

---

Figure 1. Specifying correspondences between models via Kleisly morphisms

<table>
<thead>
<tr>
<th>Table</th>
<th>PERSON</th>
<th>EMPLOYEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN</td>
<td>NAME</td>
<td>STRING</td>
</tr>
<tr>
<td>NAME</td>
<td>STR</td>
<td>INT</td>
</tr>
</tbody>
</table>

(a1) relational schemas

(b1) convenient visualization

Concrete specifications

Formal abstract specifications