Applying Model Management to Classical Meta Data Problems

Motivation
- Design, integration and maintenance of application artifacts involves meta data manipulation
- Current method – map the models to an object oriented representation and manipulate using object-at-a-time primitives
- Instead, treat models and mappings as abstractions – use model-at-a-time and mapping-at-a-time operators to improve performance

Basic terminology
- Models
  - Set of objects and relationships between them that effectively model an application artifact e.g. DB schema
- Morphism
  - One-to-one mapping between two objects in two models
- Mapping
  - Between two models is also a model that has morphisms with each of the 2 models that undergoes mapping

Models, objects, morphisms, mapping

D1 (open) – Main Contributions?
- Brand spanking new approach to meta data management
- Old approach, with some added value (e.g. extension of operators)
- Unified description of operators under a common framework
- Detailed and formal descriptions of the operators
- Algorithms/means of computing the operators
- Proposed system with high level interface to model management
- Application of model management to meta data management problems
Operators

- Match – similarities between 2 models
  - Input – 2 models
  - Output – Mapping between models identifying equal or similar objects in models
- Elementary and Complex match operators
  - Elementary – simple definitions of equality
  - Complex – be able to identify exact matches and similar matches maybe using semantic knowledge

Complex Match Operator

Operators (2)

- Diff – difference between 2 models M1 & M2
  - Input – a model M1 and a mapping map1, which is result of match of M1 and M2
  - Output – a model M1’ with objects of M1 that are not referenced in the mapping map1, and a mapping to M1 that distinguishes support objects
  - Support objects – objects to provide model structural integrity

Diff operator

Operators (3)

- Merge
  - Input – 2 models to merge and a mapping
  - Output – Model with all objects in input models with objects that are equal collapsed into a single object and mappings between merged model and input models
- Compose
  - Creates a mapping by combining two other mappings – e.g. M1 map1 M2 and M2 map2 M3
  - composition of map1 and map2 is map1 between M1 and M3 (map2(map1(M1)) = map1(map2(M1)))

Operators (4)

- Apply – takes a model and a function f as input and applies f to each object in model
- Copy – takes a model as input and returns a copy of it
- DeepCopy – copies model and associated mapping
- ModelGen - takes a model A, and returns a model B based on A (typically B’s data model would be different than A’s) and a mapping between the two
- Enumerate - returns objects in model one at a time
Application in Schema Integration

- Problem
  - Suppose we have two databases with different schemas, \( S_1 \) and \( S_2 \), and we want to create an integrated schema \( S_3 \), as well as the mapping between \( S_1 \) and \( S_3 \), and \( S_2 \) and \( S_3 \).

Step 1

- First, identify overlapping information in \( S_1 \) and \( S_2 \)
- To achieve this, use Match operator to create a mapping between the two (via Complex matching, since they are likely to be independently developed schemas)

\[
\text{map}_{12} = \text{Match}(S_1, S_2)
\]

Match Result:

Step 2

- Use the identified overlaps to merge \( S_1 \) and \( S_2 \)
- To achieve this, we use Merge on \( S_1 \), \( S_2 \) and \( \text{map}_{12} \) to get the integrated schema and desired mappings

\[
\langle S_3, \text{map}_{13}, \text{map}_{23} \rangle = \text{Merge}(S_1, S_2, \text{map}_{12})
\]

Merge Result:

Step 3

- We have to deal with conflicts in the mapping (i.e. \( S_1 \) and \( S_2 \) having the same information, but represented differently)
- The conflicting objects are rooted under the object labelled by the symbol
- The object \( = \) is a placeholder for an expression property that relates conflicting objects and resolves the conflict
- Thus we have an integrated schema \( S_3 \) with its mappings to \( S_1 \) and \( S_2 \)
D2 (idea injection) Feasibility

- is it feasible? why or why not?
- is a subset feasible?
- is it feasible for certain applications/circumstances?