Toward Tractable Instantiation of Conceptual Data Models using Non-Semantics-Preserving Model Transformations

Matthew Nizol
Michigan State University
East Lansing, MI, USA
nizolmat@cse.msu.edu

Laura K. Dillon
Michigan State University
East Lansing, MI, USA
ldillon@cse.msu.edu

R.E.K. Stirewalt
LogicBlox, Inc.
Atlanta, GA, USA
kurt.stirewalt@logicblox.com

ABSTRACT

As a bridge from informal business requirements to precise specifications, conceptual models serve a critical role in the development of enterprise systems. Instantiating conceptual models with test data can help stakeholders validate the model and provide developers with a test database to validate their code. ORM is a popular conceptual modeling language due in part to its expressive constraint language. Due to that expressiveness, instantiating an arbitrary ORM model is NP-hard. Smaragdakis et al. identified a subset of ORM called ORM$^-$ that can be instantiated in polynomial time. However, ORM$^-$ excludes several constraints commonly used in commercial models. Recent research has extended ORM$^-$ through semantics-preserving transformations. We extend the set of ORM models that can be transformed to ORM$^-$ models by using a class of non-semantics-preserving transformations called constraint strengthening. We formalize our approach as a special case of Stevens' model transformation framework. We discuss an example transformation and its limitations, and we conclude with a proposal for future research.

Categories and Subject Descriptors

D2.5 [**Software Engineering**]: Testing and Debugging; H2 [**Database Management**]: Logical Design—*Data models*

General Terms

Verification, theory

Keywords

ORM, test data generation, model transformation, databases

1. INTRODUCTION

Enterprise decision making is increasingly data-driven. To properly design data-intensive enterprise systems that support this decision-making process, developers working in concert with business domain experts must first conceptually model the database. A conceptual model defines the real-world entities relevant to the system, the relationships among those entities, and constraints that clarify the semantics of the model. Beyond serving as a blueprint for the database schema, a conceptual model can aid in the validation of an enterprise system. Populating a conceptual model with data that adheres to the constraints can aid business stakeholders in validating system requirements. Moreover, test data generated from a conceptual model can be used by programmers to test the applications that use the database.

Test data generation from conceptual models is an active area of research [10, 7, 13]. One recent line of research has focused on the instantiation of models developed in the Object Role Modeling (ORM) language. ORM is a graphical conceptual modeling language that maps into first order logic [5]. ORM represents a domain as objects playing roles in relations that represent sets of facts; an expressive constraint language allows precise definition of the valid relations. The expressiveness of ORM empowers modelers, and ORM's intrinsic formality permits automated reasoning. Unfortunately, these two aspects of ORM are in conflict: due to the expressiveness of the constraint language, determining whether an arbitrary ORM model is satisfiable is an NP-hard problem [10].

Constraints that require comparisons of the individual objects playing roles in a relation are a significant factor in the complexity of the ORM satisfiability problem. Based on this observation, Smaragdakis et al. [10] defined ORM$^-$, a strict subset of ORM that only includes constraints related to the cardinalities of model elements. ORM$^-$ models can be checked for satisfiability in time polynomial in the model size and can be instantiated in time polynomial in the generated output size. However, a follow-up study of models developed at LogicBlox, Inc. found that most models developed for commercial use include constraints outside of ORM$^-$ [7]. To remedy this, recent research has extended ORM$^-$ to include commonly used features of ORM such as objectification and compound reference schemes. In particular, models containing compound reference schemes can be transformed into ORM$^-$ models via semantics-preserving transformations [7].

Still, there are several commonly used constraint types in ORM for which we do not know of a general semantics-preserving transformation to an ORM$^-$ model. We have observed that in some cases we can replace those constraints with more restrictive constraints and still obtain an instantiable ORM$^-$ model. This form of transformation, called

MiSE'14, June 2 – June 3, 2014, Hyderabad, India
ACM 978-1-4503-2849-4/14/06
http://dx.doi.org/10.1145/2593770.2593771

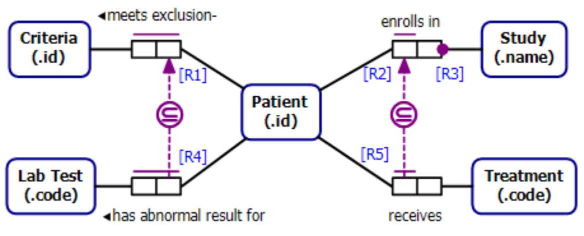

Figure 1: An example ORM model

constraint strengthening, can result in a model with fewer instances but that can be efficiently instantiated using the ORM$^-$ algorithm. If constraint strengthening produces an ORM$^-$ model with at least one instance, then that instance is also an instance of the source ORM model. Of course, the converse is not true: if the transformed model has no instance, we cannot conclude that the source ORM model is unsatisfiable. While this approach does not provide the same theoretical guarantee enjoyed by semantics-preserving transformations, our recent analysis of a real-world model suggests that it can be useful in a practical setting.

The contributions of this paper are threefold. First, we relate our work to the literature on model transformation by framing our problem as a restricted case of Stevens' framework [11]. Within this framework, we precisely categorize transformations as semantics-preserving, strengthening, and constraint strengthening. Second, we extend the set of ORM models that can be instantiated by the ORM$^-$ algorithm by applying constraint strengthening as a transformation step. While strengthening transformations have been discussed by Proper and Halpin [9] in the context of schema optimization, we are the first to extend the set of ORM models that can be transformed to ORM$^-$ models in this manner. Third, we discuss future research needed to theoretically extend and empirically validate the use of strengthening transformations for test data generation. In particular, we discuss the need for further research on issues such as traceability and triviality.

2. BACKGROUND

2.1 ORM and ORM$^-$

ORM [5] is a conceptual modeling language based on first order logic. An ORM model consists of a set of object types $\mathcal{O}$, a set of fact types $\mathcal{F}$, and a set of constraints $\mathcal{C}$. In that sense, an ORM model M is a 3-tuple $(\mathcal{O}, \mathcal{F}, \mathcal{C})$[1]. ORM models may be displayed graphically. We use Figure 1 and adopt some notational conventions from [7] to assist in the subsequent discussion of basic ORM terminology.

Each *object type* $O \in \mathcal{O}$ defines both a domain D_O and a reference scheme that identifies objects drawn from D_O. Object types are shown as rounded rectangles: solid edges indicate an *entity type*, which requires a reference scheme, while dashed edges indicate a *value type*, which is self-identifying. The reference scheme for an entity type can be indicated in parentheses. For example, in Figure 1, Patient is an entity type whose reference scheme is "(.id)," which implies the existence of a value type named PatientId used to identify Patients. An *object* is an element of the domain of an object type. A *population* of an object type O is a subset of D_O.

[1]This is a simplification. See [9] for a more complete formalism.

A *fact type* $F \in \mathcal{F}$ defines a relationship among one or more object types. A fact type is displayed as a set of contiguous boxes, each representing a *role* in that relationship played by the connected object type. For example, Patient and Study each play a role in the binary fact type "Patient enrolls in Study." We may enumerate the object types related by a fact type via subscripts; for instance, $F_{O_1,\ldots,O_k}$ denotes a fact type relating the k object types O_1 through O_k. Roles may be explicitly named in the diagram by bracketed text next to the role (e.g. $[R1]$). We may also denote the j^{th} role in fact type F_i as $R_{i,j}$. The object type playing role $R_{i,j}$ is identified by the function $Player(R_{i,j})$. A *fact* is an element of the domain of a fact type; in effect, a fact is a tuple of objects of the appropriate types. A *population* of a fact type is a subset of the domain of that fact type; thus, a population of a fact type is a relation.

A *constraint* $C \in \mathcal{C}$ restricts the *valid population* of one or more object or fact types. A constraint is said to *cover* the roles that it restricts. For example, the solid line above role $R2$ in Figure 1 is an internal uniqueness constraint, which covers $R2$. An internal uniqueness constraint asserts that an object may play the covered role at most once. In this case, each Patient may enroll in at most one Study. The dot on role $R3$ is a mandatory constraint, which specifies that each object in the population of that object type must play that role at least once. Thus, every Study must have at least one enrolled Patient. Because these constraints cover only one role, they are called *simple*; a constraint, however, can cover more than one role in a fact type. For example, Figure 1 contains an internal uniqueness constraint covering both roles of the "meets exclusion-" fact type. This constraint requires only that each (Criteria, Patient) tuple in the population of that fact type is unique; in effect, the fact type is an $m{:}n$ relationship. Some constraints, such as subset constraints, cover roles in more than one fact type. Figure 1 includes two subset constraints. A subset constraint is displayed as a subset symbol inside a circle connected by a dashed arrow from the subset role to the superset role. For example, the constraint from role $R5$ to role $R2$ specifies that each Patient who receives Treatment must be enrolled in a Study.

Each model M defines a set of instances $\mathcal{I}_M$. More precisely, an *instance* $I \in \mathcal{I}_M$ is a mapping of the object and fact types in model M to populations that satisfy all constraints in the constraint set $\mathcal{C}$. We denote the populations of an object type O and a fact type F for a given instance I as $I(O)$ and $I(F)$, respectively. Because populations are sets of objects or facts that must be drawn from the appropriate domain, we have that $I(O) \subseteq D_O$ and $I(F_{O_1,O_2,\ldots,O_k}) \subseteq D_{O_1} \times D_{O_2} \times \ldots \times D_{O_k}$. The population of role $R_{i,j}$ in fact type F_i, denoted as $\pi_{R_{i,j}}(I(F_i))$, is the projection of the fact type's population on that role.

We can define various levels of model *satisfiability* [6]. A model $M = (\mathcal{O}, \mathcal{F}, \mathcal{C})$ is *strongly satisfiable* if there exists at least one instance $I \in \mathcal{I}_M$ such that $I(F)$ is non-empty for all fact types $F \in \mathcal{F}$. A model $M = (\mathcal{O}, \mathcal{F}, \mathcal{C})$ is *weakly satisfiable*[2] if there exists at least one instance $I \in \mathcal{I}_M$ such that $I(O)$ is non-empty for all object types $O \in \mathcal{O}$. Clearly strong satisfiability implies weak satisfiability. Because our

[2]Jarrar calls this *concept satisfiability*

objective is to find at least one instance of a model for purposes of test data generation, we consider a model *satisfiable* if it is weakly satisfiable[3].

An ORM$^-$ model is an ORM model whose constraint set $\mathcal{C}$ is restricted to certain forms of uniqueness, frequency, mandatory, value, cardinality, and subtype constraints[4] [10]. Each of these constraint types places a restriction on the cardinality of the roles or object types it covers. The ORM$^-$ satisfiability algorithm generates a system of inequalities based on these constraints; a solution to the system represents cardinality assignments for each object type and role in the model. Because every constraint in ORM$^-$ can be expressed using inequalities that have at most one variable on the left hand side, the system can be solved in polynomial time using a fixpoint algorithm. Given the computed cardinalities, [10] then shows how to efficiently generate an instance for the model.

2.2 Model Transformation

One extension to ORM$^-$ proposed in [7] makes use of semantics-preserving model transformations to convert an ORM model containing compound reference schemes into an ORM$^-$ model. The approach we propose in this paper makes use of non-semantics-preserving model transformations. To permit formal reasoning about these transformations, we precisely define the notion of model transformation and then discuss how transformations can be classified according to their semantics-preservation properties.

Model transformation has been studied in the database community[5] in contexts such as database normalization, optimization, and reverse engineering [3]. Recent software engineering scholarship has investigated standardized languages such as QVT [2] for specifying transformations between models of a system. While discussing open issues in both the QVT standard and in the broader model transformation community, Stevens [11, 12] defines a useful framework for formal reasoning about model transformations. A very simple model of a unidirectional transformation is a function $t\colon \mathcal{M} \to \mathcal{N}$ where $\mathcal{M}$ and $\mathcal{N}$ are metamodels[6] [11]. Stevens ultimately proposes a more complex notation for model transformation due to the need to specify bidirectional transformations between persistent models. The target models in our approach are transient: they exist only to find an instance of the original model, and then are discarded. Thus, the simpler formalism serves our purpose well. In fact, because our approach only considers transformations between ORM models, we restrict our definition further:

Definition 2.1. *Let $\mathcal{ORM}$ be the set of all well-formed ORM models. Then a* model transformation *is a function $t\colon \mathcal{ORM} \to \mathcal{ORM}$.*

In Stevens' framework, two models are considered *consistent* if a *consistency relation* R holds between them [12]. That is, $R(M, N)$ implies that models M and N are consistent. A transformation t must not produce a target model inconsistent with the source model; therefore, a transformation t is *correct* with respect to a consistency relation R iff $t(M) = N$ implies $R(M, N)$[7]. While a transformation is a function, a consistency relation need not be: the same source model can be considered consistent with more than one target model. Thus, more than one correct transformation function can exist with respect to a given consistency relation. The definition of consistency depends on the application. For the purpose of finding an instance of a source ORM model from a target ORM$^-$ model, we require a notion of consistency that takes into account the relationship between the instance sets of each pair of consistent models. In the next two sections, we define three consistency relations, each based upon different relationships between the instance sets of consistent models, which are useful in the context of test data generation from ORM models.

3. TRANSFORMATION TYPES

3.1 Semantics-Preserving

Ideally, a transformed model will have an instance if and only if the original model has an instance. A transformation provides this guarantee if every instance of the source model maps to a unique instance of the target model and vice versa. We call such transformations *semantics-preserving.* A semantics-preserving transformation is defined in terms of the semantics-preserving consistency relation R_{sem}:

Definition 3.1. *For all models $M, N \in \mathcal{ORM}$, the consistency relation $R_{sem}(M, N)$ holds if and only if there exists a bijection $r\colon \mathcal{I}_M \to \mathcal{I}_N$. A transformation $t\colon \mathcal{ORM} \to \mathcal{ORM}$ is* semantics-preserving *if and only if for all models $M, N \in \mathcal{ORM}$, $t(M) = N$ implies $R_{sem}(M, N)$.*

Under a semantics-preserving transformation, an instance of the original model can always be recovered from an instance of the target model (and vice versa) via the bijective mapping. Recall that our objective is to find an instance for the original model, and so the transformed models are transient; having found an instance of the transformed model, we simply need to ensure that there exists some mapping back to the original. This property is guaranteed by Definition 3.1.

3.2 Strengthening

While a semantics-preserving transformation from an ORM model to an ORM$^-$ model can permit efficient instantiation of the source model, the NP-hardness of the ORM satisfiability problem[8] suggests that such transformations do not exist for all source ORM models. Nevertheless, a semantics-preserving transformation may not be necessary in every case: our objective is simply to find one instance of the original ORM model rather than every possible instance. Thus, we might be able to modify the model in a non-semantics-preserving fashion, as long as we ensure that there is some means to map instances of the target model back to instances of the original model. The existence of an injection from the target instance set $\mathcal{I}_N$ to the source instance set $\mathcal{I}_M$ would be a sufficient condition for this purpose. The target model

[3]An implicit cardinality constraint "$\# > 0$" covers each object type

[4]ORM$^-$ supports only internal, non-overlapping forms of uniqueness and frequency constraints. Explicit disjunctive mandatory constraints are not permitted.

[5]Under the name *schema transformation*

[6]Following Stevens, we identify a metamodel with the set of models it defines.

[7]Stevens defines other properties such as hippocraticness and undoability which are relevant when the target model is persistent.

[8]See [10] regarding the problem's NP-completeness

N could be more restrictive than the source model M, in the sense that the target model could admit fewer instances. For this reason, we call such transformations *strengthening*. A strengthening transformation is defined in terms of the strengthening consistency relation R_{str}:

DEFINITION 3.2. *For all models $M, N \in \mathcal{ORM}$, the consistency relation $R_{str}(M, N)$ holds if and only if there exists an injection $r\colon \mathcal{I}_N \to \mathcal{I}_M$. A transformation $t\colon \mathcal{ORM} \to \mathcal{ORM}$ is* strengthening *if and only if for all models $M, N \in \mathcal{ORM}$, $t(M) = N$ implies $R_{str}(M, N)$.*

Analysis of a model developed by LogicBlox for a commercial client suggests that strengthening transformations can be useful in a practical setting. The model contains a set of constraints that are outside of both ORM$^-$ and the extensions in [7]. However, replacing some of these constraints with more restrictive constraints produces an ORM$^-$ model that, while not semantically equivalent to the original, is nevertheless satisfiable. In fact, because we do not modify the object or fact types of the original model, the instances of the transformed model are also instances of the original model. That is, the instance set of the transformed model is a subset of the instance set of the original model. We call transformations with this property *constraint strengthening*. A constraint-strengthening transformation is defined in terms of the constraint-strengthening consistency relation R_{con}:

DEFINITION 3.3. *For all models $M, N \in \mathcal{ORM}$, the consistency relation $R_{con}(M, N)$ holds if and only if $\mathcal{I}_N \subseteq \mathcal{I}_M$. A transformation $t\colon \mathcal{ORM} \to \mathcal{ORM}$ is called* constraint-strengthening *if and only if for all models $M, N \in \mathcal{ORM}$, $t(M) = N$ implies $R_{con}(M, N)$.*

Corollary 3.4 directly follows from Definitions 3.2 and 3.3:

COROLLARY 3.4. *Every constraint-strengthening transformation is a strengthening transformation.*

For example, Figure 2 is the result of applying a constraint-strengthening transformation to Figure 1. Specifically, the transformation replaced the subset constraints with mandatory constraints on the superset roles. The mandatory constraint on role $R2$, which requires that all Patients enroll in some Study, implies a subset constraint from $R5$ to $R2$. The mandatory constraint on role $R1$ has similar semantics. Hence, any instance of the transformed model is still an instance of the original model. However, the semantics have not been preserved: there are instances of the original model that are not valid in the transformed model. This type of transformation, which we call a *Subset-Mandatory* transformation, is specified formally as follows:

DEFINITION 3.5. *A transformation $t\colon \mathcal{ORM} \to \mathcal{ORM}$ is a* Subset-Mandatory *transformation if for all models $M = (\mathcal{O}, \mathcal{F}, \mathcal{C})$ such that $t(M) \neq M$:*

1. *M contains a simple subset constraint C from some subset role $R_{i,j}$ in fact type F_i to some superset role $R_{k,l}$ in fact type F_k, with $i \neq k$*
2. *$Player(R_{i,j}) = Player(R_{k,l})$*
3. *$t(M) = (\mathcal{O}, \mathcal{F}, \mathcal{C}')$ where $\mathcal{C}'$ is $\mathcal{C}$ replacing subset constraint C with a mandatory constraint C' on role $R_{k,l}$*

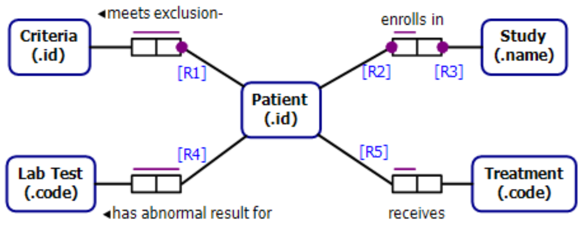

Figure 2: Constraint strengthening applied to Fig. 1

LEMMA 3.6. *Every Subset-Mandatory transformation t is constraint strengthening.*

PROOF. Consider any models M and N such that $t(M) = N$. If there exists an instance $I \in \mathcal{I}_N$, then I satisfies all constraints in $\mathcal{C}' = \mathcal{C} \setminus \{C\} \cup \{C'\}$. Because I satisfies C', $\pi_{R_{k,l}}(I(F_k)) = I(Player(R_{k,l}))$, which implies that $\pi_{R_{i,j}}(I(F_i)) \subseteq \pi_{R_{k,l}}(I(F_k))$. Therefore, I also satisfies C and is an instance of M. Hence, $\mathcal{I}_N \subseteq \mathcal{I}_M$. $\square$

This simple transformation achieves an important objective: it replaces a subset constraint, which is not part of the ORM$^-$ language, with a simple mandatory constraint, which is defined in ORM$^-$. Hence, after applying this transformation, we can use the ORM$^-$ algorithm to efficiently search for an instance. If we identify an instance, it is also an instance of the original model. As we will discuss in the next section, however, a strengthening transformation can produce an unsatisfiable target model from a satisfiable source model.

Given an arbitrary real-world ORM model, a single transformation is unlikely to produce an ORM$^-$ model. Thus, we need to be able to compose multiple transformations to produce an ORM$^-$ model from which to compute an instance of the source ORM model. The following corollary is useful in this regard:

COROLLARY 3.7. *The sets of semantics-preserving transformations, strengthening transformations, and constraint-strengthening transformations are each closed under composition.*

The proof follows easily from closure properties of bijective and injective functions and from the transitivity of the subset relation.

4. FUTURE RESEARCH

4.1 Triviality

Our objective is to find a weakly satisfiable target model (i.e., there must be at least one instance of the target model in which every object type is populated). Not every strengthening transformation can meet this objective: a strengthening transformation may be too restrictive and produce an unsatisfiable model. In that case, the instance set $\mathcal{I}_N$ of the target model N is empty. An empty instance set satisfies Definition 3.2 because the empty function is trivially injective. However, such a transformation is not useful for our purpose. We call a transformation whose target instance set is empty a *trivial* transformation.

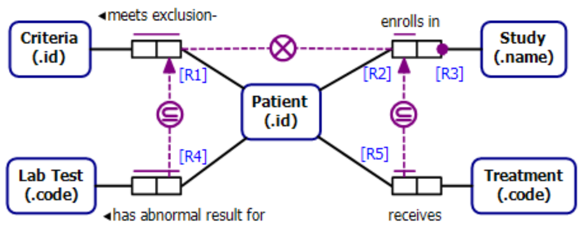

Figure 3: Alternate version of Fig. 1

Consider Figure 3, which is an alternate version of Figure 1 in which the modeler added an exclusion constraint between roles $R1$ and $R2$ to enforce the rule that no Patient who meets an exclusion Criteria may enroll in a Study. If we apply the Subset-Mandatory strengthening transformation to both subset constraints as we did in Figure 2, the result will be an unsatisfiable model. If mandatory constraints on roles $R1$ and $R2$ require that every Patient both meets some exclusion Criteria and enrolls in some Study, then the exclusion constraint must be violated. If we honor the exclusion constraint, then one of the mandatory constraints must be violated. Thus, there would be no way to populate any role played by Patient while satisfying all constraints in the model.

Research by Jarrar [6] to enumerate unsatisfiability patterns in ORM models provides a starting point for identifying patterns that guarantee triviality. For example, replacing the subset constraints in Figure 3 with mandatory constraints on $R1$ and $R2$ would produce the "Exclusion-Mandatory" pattern [6]. One goal of our research will be to identify additional anti-patterns that guarantee the image under a strengthening transformation is unsatisfiable. We can formalize such a pattern as a predicate that defines a set of ORM models. Given such a predicate, we could reason about whether a transformation is guaranteed to be trivial.

Alternatively, we would like to identify conditions on a source model that guarantee a strengthening transformation is non-trivial if the source model is satisfiable. For example, consider a predicate Q defined as follows:

DEFINITION 4.1. *For all models $M \in \mathcal{ORM}$ that satisfy conditions (1) and (2) of Definition 3.5, define Q to contain those models M that additionally satisfy the following:*

1. *The fact type F_k containing the superset role $R_{k,l}$ is binary*
2. *A simple internal uniqueness constraint C_{uc} covers $R_{k,l}$*
3. *The only constraints covering $R_{k,l}$ are C_{uc} and the subset constraint C*
4. *The other role of F_k, $R_{k,m}$, is covered by a (possibly implied) cardinality constraint $\# > 0$ and no other constraints.*

This predicate is useful for our purposes: if it holds for a satisfiable model, then applying the Subset-Mandatory transformation to that model is guaranteed to produce a satisfiable target model.

LEMMA 4.2. *Let t be a Subset-Mandatory transformation. For all models $M \in \mathcal{ORM}$ such that $Q(M)$ holds, $\mathcal{I}_M \neq \emptyset \implies \mathcal{I}_{t(M)} \neq \emptyset$.*

PROOF. Consider any model $M \in \mathcal{ORM}$ such that $Q(M)$ and $\mathcal{I}_M \neq \emptyset$. Consider any instance $I \in \mathcal{I}_M$. Let $\mathcal{Z} = I(Player(R_{k,l})) \setminus \pi_{R_{k,l}}(I(F_k))$. If $\mathcal{Z} = \emptyset$, then $I \in \mathcal{I}_{t(M)}$. Otherwise, let o be any element of $\pi_{R_{k,m}}(I(F_k))$. Construct $I' = I$, except that $I'(F_k) = I(F_k) \cup (\mathcal{Z} \times \{o\})$. I' satisfies the constraints of parts (2) through (4) of Definition 4.1, so $I' \in \mathcal{I}_M$. Moreover, $\pi_{R_{k,l}}(I'(F_k)) = I'(Player(R_{k,l}))$, so $I' \in \mathcal{I}_{t(M)}$. Hence $\mathcal{I}_{t(M)} \neq \emptyset$. □

4.2 Traceability

The definitions of semantics-preserving and strengthening transformations require that a mapping exists from the target instance set to the source instance set. However, the definitions provide no clue as to how that mapping might be identified. Moreover, while the mapping $r\colon \mathcal{I}_N \to \mathcal{I}_M$ is injective for both semantics-preserving and strengthening transformations, we do not require that the transformation t itself is injective. That is, more than one source model might map to the same target model. Given a target model N, we need some means to determine the source model M from among the set of potential source models for which the consistency relation R holds. To address this issue, we need a means to associate traceability information with the target model. In other words, for each transformation, we wish to maintain a record of the correspondence between source and target model elements [8]. The exact form of this traceability information is an area for future research.

5. RELATED WORK

Test data generation is an active area of research. Similar work includes the TESTBLOX tool developed by Torlak to generate test data from models of multidimensional OLAP cubes [13]. Our work directly extends [10] and [7] through model transformations. Model transformations were formalized by the database community in the 1990s by authors such as Hainaut [3, 4]. Hainaut defines a schema transformation T declaratively in terms of a precondition P and a postcondition Q. Essentially, P and Q define subsets of the source and target metamodels, respectively, on which the transformation acts. In our notation, $P = \{M \in \mathcal{ORM} \mid t(M) \neq M\}$. In addition, a transformation includes a mapping t between instances. Hainaut's definition of a transformation at the instance level is similar to our definition of consistency relations based on the relationship between instance sets.

Part of our work relates to classifying transformations as semantics-preserving, strengthening, and constraint strengthening. Czarnecki [1] and Mens [8] each propose broader classification schemes for model transformations. They define a variety of dimensions along which transformations may be classified including implementation issues (e.g., rule set execution order), specification paradigms (e.g., relational, graph-based, imperative), and transformation characteristics (e.g., directionality).

Hainaut [3] describes semantics-decreasing and semantics-augmenting transformations. A semantics-decreasing transformation removes model elements; a semantics-augmenting transformation adds model elements. These notions differ from strengthening. For instance, the Subset-Mandatory transformation both adds and removes one model element, and so it cannot be clearly classified as semantics-augmenting or semantics-decreasing. Proper and Halpin's discussion of strengthening transformations shares more similarity with our work [9]. ORM models may be expressed as formulas in predicate logic. Thus, a model instance for Proper and Halpin is a valid interpretation of the logical formula of that model. They define a strengthening transformation as generating a target model whose set of valid interpretations is a subset of the source model's set of valid interpretations. While fundamentally the same as our concept of strengthening, Proper and Halpin apply strengthening transformations to database optimization rather than to test data generation. Finally, our work to identify triviality predicates is similar to work by Jarrar [6] to identify unsatisfiability patterns.

6. CONCLUSION

We have had initial success in applying constraint strengthening to instantiate a conceptual model developed by LogicBlox that contains constraints outside of ORM$^-$. We plan to extend the empirical validation of constraint strengthening as a test data generation approach through a study of additional commercial models provided by LogicBlox. During this study, we plan to identify additional strengthening transformations along with conditions under which they are non-trivial. In cases where a triviality predicate does not exist to predict the behavior of a strengthening transformation, we will investigate using the ORM$^-$ consistency checker as a subroutine and backtracking to attempt other transformations if the first transformation fails. We plan to further explore the composition of semantics-preserving and strengthening transformations to determine how traceability metadata can assist in recovering an original model from a sequence of transformations. We will also investigate how the order of transformations affects satisfiability.

In some sense it is intuitive that strengthening produces a model that is easier to instantiate than the original: strengthening essentially restricts the search space for an instance. At the same time, however, one may argue that a strengthened model is less useful for test data generation than the original, as certain database states will not be tested. We argue that two concerns justify strengthening transformations despite this limitation. First, it is clearly preferable to find some instance of a conceptual model, however restricted semantically, than no instance. Moreover, certain applications such as performance testing may benefit from generating as large a test database as possible—with the requirement that the test data satisfies all constraints—without concern for the semantic richness of the data. Thus, strengthening transformations may be useful in practice. As we continue our work on transformation-based test data generation, we will continue to explore these and other applications of strengthening transformations.

7. ACKNOWLEDGMENTS

We would like to thank Matt McGill for early discussions that led to some of these results and for valuable comments on an earlier version of this paper.

8. REFERENCES

[1] K. Czarnecki and S. Helsen. Classification of model transformation approaches. In *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 2003.

[2] Object Management Group. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.1*, January 2011. OMG Document: formal/2011-01-01.

[3] J.-L. Hainaut. Specification preservation in schema transformations—application to semantics and statistics. *Data Knowl. Eng.*, 19(2):99–134, 1996.

[4] J.-L. Hainaut. *Transformation of Knowledge, Information and data: Theory and Applications*, chapter Transformation-based Database Engineering, pages 1–26. IDEA Group, 2005.

[5] T. Halpin and T. Morgan. *Information Modeling and Relational Databases.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2008.

[6] M. Jarrar and S. Heymans. Unsatisfiability reasoning in ORM conceptual schemes. In *Current Trends in Database Technology – EDBT 2006*, volume 4254 of *Lecture Notes in Computer Science*, pages 517–534. Springer Berlin Heidelberg, 2006.

[7] M. J. McGill, L. K. Dillon, and R. E. K. Stirewalt. Scalable analysis of conceptual data models. In *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, ISSTA '11, pages 56–66, New York, NY, USA, 2011. ACM.

[8] T. Mens and P. Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152(0):125 – 142, 2006. Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005).

[9] H. A. Proper and T. A. Halpin. Conceptual schema optimisation—database optimisation before sliding down the waterfall. Technical Report 341, Department of Computer Science, University of Queensland, Brisbane, Australia, July 1995. Version of June 23, 2004 at 10:31.

[10] Y. Smaragdakis, C. Csallner, and R. Subramanian. Scalable satisfiability checking and test data generation from modeling diagrams. *Automated Software Engineering*, 16(1):73–99, 2009.

[11] P. Stevens. A landscape of bidirectional model transformations. In *Generative and Transformational Techniques in Software Engineering II*, volume 5235 of *Lecture Notes in Computer Science*, pages 408–424. Springer Berlin Heidelberg, 2008.

[12] P. Stevens. Bidirectional model transformations in QVT: semantic issues and open questions. *Software & Systems Modeling*, 9(1):7–20, 2010.

[13] E. Torlak. Scalable test data generation from multidimensional models. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pages 36:1–36:11, New York, NY, USA, 2012. ACM.