Experimentally motivated transformations for intermodel links between conceptual models

Zubeida C. Khan[1,2], C. Maria Keet[1], Pablo R. Fillottrani[3,4], Karina Cenci[3]

[1] Department of Computer Science, University of Cape Town, South Africa, mkeet@cs.uct.ac.za
[2] Council for Scientific and Industrial Research, South Africa, zkhan@csir.co.za
[3] Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, {prf,kmc}@cs.uns.edu.ar
[4] Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina

Abstract. Complex system development and information integration at the conceptual layer raises the requirement to be able to declare intermodel assertions between entities in models that may, or may not, be represented in the same modelling language. This is compounded by the fact that semantically equivalent notions may have been represented with a different element, such as an attribute or class. We first investigate such occurrences in six ICOM projects and 40 models with 33 schema matchings. While equivalence and subsumption are in the overwhelming majority, this extends mainly to different types of attributes, and therewith requiring non-1:1 mappings. We present a solution that bridges these semantic gaps. To facilitate implementation, the mappings and transformations are declared in ATL. This avails of a common, and logic-based, metamodel to aid verification of the links. This is currently being implemented as proof-of-concept in the ICOM tool.

1 Introduction

Complex system development requires one to develop models before implementation. Such models may be too large to deal with at once, so that a modular approach is taken to conceptual model development, and they may represented in different modelling languages. This requires a CASE tool, or at least a modelling tool, that can manage modules and assertions of links between entities in the different modules. There are only few tools that can do this, such as ICOM [7] and Pounamu [20], which are at the proof-of-concept level and they allow only, at most, equivalence and subsumption among classes and among relationships, but not among attributes or roles, let alone have a way to handle, say, that an entity is an attribute in one model and a class in another.

In addition, modelling choices are made during the data analysis stage, such as choosing to make an attribute a simple one, a multivalued one, a composite one, or a class, with the canonical example being Address, and whether Marriage should be represented as a class or a relationship. Different choices are made in different projects for their own reasons that may have seemed good choices

at the time. However, such differences do resurface during system integration. While some transformation rules seem intuitively trivial, because they may be so in the general abstract sense, the conceptual and syntax aspects are tricky in the details and they are not readily specified formally and available, not even for models represented in the same language, let alone across modelling languages.

While it is theoretically possible to generate a huge set of links and transformations, practically, only a subset of them are needed and yet others are logically and ontologically not feasible. This leads to the following questions:

1. Taking 'projects' (sets of interlinked models) from one of such tools that allows class and relationship intermodel assertions of equivalence and subsumption: (a) Which type of intermodel assertions are actually used? (b) How often are they used, compared to each other and compared to the models' sizes? (c) Which module scenario was used? (e.g., for system integration, for managing cognitive overload). Or: What is the primary reason behind intermodel assertions, if possible to ascertain?
2. In an integration scenario, if not constrained by the limitations of the tool regarding the implemented types of intermodel assertions, then which links would be used or needed, including any possible between-type intermodel assertions?

To answer these questions, we conducted an experimental evaluation with six ICOM projects, and added 9 more integration scenarios to it using publicly available models in different languages on the same universe of discourse, involving 40 models overall in 33 schema matchings. We describe and analyse several types of intermodel links and for those that relate different types of entities—either in the same conceptual data modelling language or across languages—we present a structure that bridges the semantic gap. While this could have been formalised in a logic, we valued applicability and therefore used the well-known Model-Driven Engineering's ATL-style notation [12] for the transformation specifications, and use a common metamodel [14] to mediate between models represented in different languages.

In the remainder of the paper we first report on the experimental assessment of intermodel links in Section 2. The specification of the transformations is presented in Section 3. We compare it with related work and justify the approach taken in Section 4, discuss in Section 5, and conclude in Section 6.

2 Experimental assessment of intermodel assertions

The purpose of the experimental evaluation is to analyse existing conceptual data models on intermodel assertions among them.

2.1 Materials and methods

The assessment has been designed in two complementary experiments:

1. Analyse existing intermodel assertions: (i) Collect model sets from ICOM projects; (ii) For each project in the set: (a) analyse its contents and intermodel assertions, by measuring most frequently linked concept type, most

frequently linked relationship type, and number of intermodel links; (b) Analyse the project to determine whether it is an integration project or a module project; (iv) Repeat steps 2-3 for each model in each set.

2. Simulated system integration scenarios:
   i. Collect model sets (online-sourced) in several subject domains with each at least two models in either UML, EER, or ORM.
   ii. For each model set, link the models to the other one(s) in the set—2 at a time—by using intermodel assertions manually, unconstrained by whether a tool would support such links. Concerning the links, the following decisions have been taken: 1) there are equivalence and subsumption links; 2) those that have a 1:1 mapping regarding the metamodel [14] are counted as 'full' links; 3) there are entities that are very similar (e.g., an attribute with or without a data type, different constraints), which are 'half' links; 4) those links that require some transformation (e.g., class to relationship) are 'trans' links; 5) concerning class hierarchies, there also may be 'implied' links.
   iii. Analyse the collected intermodel assertions by measuring most frequently linked concept type, most frequently linked relationship type, number of intermodel links, number of transformations.
   iv. Repeat steps 2-3 for each model pair in each set.

The materials consisted of six ICOM projects each with intermodel assertions, covering domains about telecommunications, college, governance, and taxation created by the students at UNS. The second model set covers nine 'projects', each containing three models in either UML, EER, or ORM, and each covering a different domain (bank, car insurance, flights, hospital, hotel, library, movie, sales, and university systems).

2.2 Results and discussion

The models and the analyses are available at `http://www.meteck.org/SAAR.html` and the results are summarised and discussed in this section.

Five of the six ICOM projects contain links between two models, and one contains links between three models. There are a total of 25 links, with an average of 4.17 links per project. There are 194 entities in the set of projects of which thus 25*2=50 entities (25%) are linked. The links are mainly equivalence and subsumption, with one being a disjointness link. 14 object types and 11 relationships were involved, with the breakdown as included in Table 1. Four of these projects were created for integration purposes, and the remaining two were created to manage cognitive overload, by splitting up large models into separate subject domain modules. An example of the latter is the project about a telecommunication data warehouse and a model with customer call information.

In order to uncover information about conceptual data model modules, we classify these projects according to the framework for ontology modularity [15]. This module classification is used to determine use-cases for creating modules, techniques that are used to create the modules, and properties that the modules exhibit. The projects on cognitive overload correspond to the subject domain

Table 1. Total links by type for the ICOM projects and for the simulated integration scenarios. OT = Object Type; VT = Value Type; att. = attribute; id. = identifer.

Link type	**Subdivision**	**Comments**
Links that can be declared in ICOM (projects/scenarios)		
Equivalence (6/106)	Among OTs (4/72)	Probably fewer logically
	Among attributes (0/26)	
	Among relationships (2/8)	
Subsumption (18/27)	Among OTs (9/16)	
	Among relationships (9/11)	Due to cardinality constraint differences
Disjointness (1/1)	Among OTs (1/1)	
New link types (scenarios only)		
'Half' Links (64)	'Missing datatype' between ER and UML (56)	
	Relationship constraint mismatch (4)	Neither subsumption nor equivalence
	Composite attribute 'leaves' (2)	
	Attribute constraint mismatch (2)	Both UML attribute, different cardinality
Implied Subsumption (12)		Excluding the hospital models, where it was too confusing to do manually
Transformation Links (48)	Attribute - Identifier (18)	UML attr. vs ER/ORM id.
	Attribute - VT (13)	
	Attribute - OT (5)	
	Weak OT - OT (4)	
	Composite - Attribute (3)	
	Relationship - Aggregate (2)	
	Relationship - OT (1)	
	Associative OT - OT (1)	
	OT - Nested OT (1)	

modules of the framework where the conceptual model is subdivided according to the subject domains; the projects on system integration correspond the high-level abstraction modules of the framework; further details are shown in Table 2.

For the projects created for integration, the most frequently linked relationship type is split equally between equivalence and subsumption, and the most frequently linked entity is split equally between object type and relationship for the four projects. For the projects created for managing cognitive overload, the most frequently linked relationship type is subsumption, and the most frequently linked entity is split equally between object type and relationship.

Now we consider the simulated integration scenarios. An example of manually aligned models is shown in Fig. 1, where the solid lines link entities of the same type (e.g., the object types er:Airplane and uml:Aircraft), the long-dashes dashed lines link semantically very similar entities (e.g., a full attribute,

Table 2. Classifying the ICOM projects using the framework for modularity.

	Use-case	**Type**	**Technique**	**Property**
Cognitive overload projects	Maintenance	Subject domain	*A priori*	Pre-assigned no. of modules
	Validation			
	Collaboration			Overlapping
	Reuse			
Integration projects	Comprehension	High-level abstraction	Manual	Source model
				Proper subset
				(Depth) Abstraction

as in uml:Airport.name and an attribute without data type, er:Airport.name), and the short-dashes dashed line requires some transformation, such as between er:Airplaine.Type (an attribute) and uml:Aircraft_Type (a class) and between er:Air-port.Code (an identifier, without data type) and uml:Airport.ID (a plain attribute, with data type). In these projects, there are 9.5 links in each 2-model integration scenario (a total of 257), with the model size alike depicted in Fig. 1.

The aggregates of the types of entities involved in the intermodel assertions follow from the data included in Table 1. Attributes are the ones involved most, with 119 in the 'source' and 108 in the 'target'. However, they are also the ones that occur most—by a large margin—in UML Class Diagrams and EER diagrams [13], and, as can be seen also from Fig. 1, once a class can be linked, there typically are also one or more attributes that can be linked.

As summarised and illustrated above, we identify two main kinds of links: those that relate elements that are homogeneous in the unifying metamodel [14], and those transformations that relate heterogeneous metamodel elements. The former are further classified into traditional equivalence, subsumption and disjointness links between compatible elements, which preserve the semantics of each individual model. These links cover more that half of the identified links (see Table 1), and relate compatible homogeneous entities of the original, possibly heterogeneous, models in the metamodel mappings; e.g., UML Class and EER and ORM Entity type are the same. They are homogeneous in the metamodel because they are instances of the same type, i.e., object type, attribute, or relationship, and they are compatible because they exhibit coherent properties, e.g., both attributes are ids, or the subsumed relationship has a more specific cardinality constraint than the containing one. Both original models maintain their respective semantics without changes. 'Half links' are 24.8% of the total number of links, relating homogeneous entities that do not exhibit compatible properties or constraints. The types of the mismatch in these constraints are described in Table 1. These 'half' links may be represented by equivalence axioms, but one or both original models then would have to be updated with new constraints. Any supporting tool, ideally with the aid of an automated reasoner, will have to notify the conceptual modeller of these updates in order to decide on their relevance for each model. Finally, the least common type of homogeneous

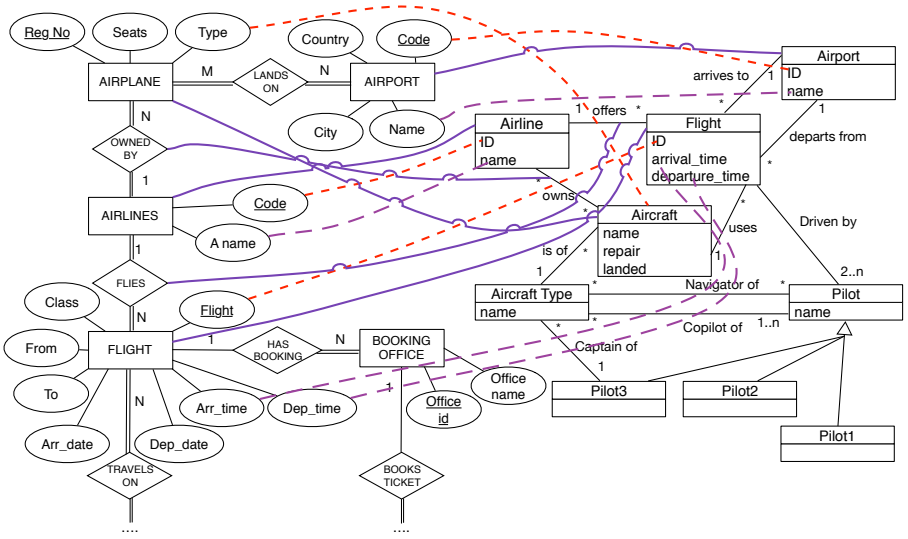

Fig. 1. The intermodel assertions between the EER and UML Flights models. Solid curvy line: links entities of the same type; long-dashes dashed line: links entities that are semantically very similar; short-dashes dashed line: requires some transformation.

links are already implied subsumption by the models semantics. These links do not need any new axiom and can be handled by an automated reasoner. The links representing transformations are analysed in next section.

3 Entity transformations

Many options exist to specify transformations both at the level of overall architecture, and for each component in the architecture, which logics, implementation languages, and technologies. Generally, for intermodel assertions, there are two input models with some intermodel assertions, a (formalised) metamodel that the entities in the models are mapped into, the transformation rules, and then the final check that the output of the rules indeed matches with the other model. An orchestration to execute and verify the intermodel assertions is depicted in Fig. 2, where we focus on an architecture for checking the links. There is already a mapping from each type of entity into the metamodel and back in the form of a table [14] and basic rules [6], so that, instead of defining very many transformation rules between individual languages in a mesh structure, one simply can classify a model element into the metamodel, especially when the metamodel drives the modelling environment. This is therefore not further elaborated on in the model mappings.

For transformation rules, we consider principally those that are across languages, that require type conversions, which are not covered by regular mappings

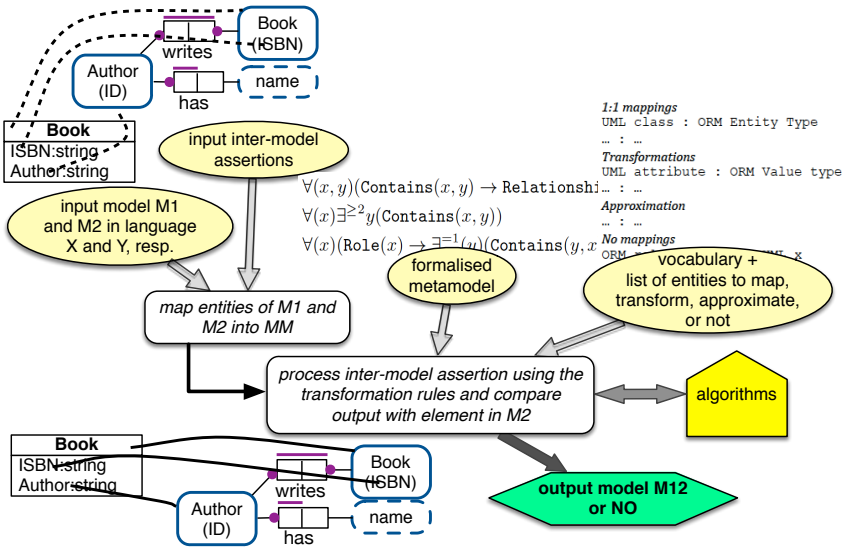

Fig. 2. General approach for validating intermodel assertions (based on [6]).

(like in [6]), those that can appear between entities that occur most often in conceptual models. The latter is based on the experimental evaluation reported in [13] of 101 UML class diagrams, ER/EER, ORM/ORM2 models. They are mainly Object type, binary Relationship, Attribute (with the attribute-value type conversion as specified in [6]), Single identifier, Mandatory constraint, Object type cardinality, and Subsumption between object types. The rules assume that the models, called `Model1` and `Model2`, are syntactically correct. This means that, e.g., in the attribute to object type rule, an attribute indeed is a binary relationship between an object type C and a data type D. The main design objective of these rules is that we only allow equivalence, disjointness, and subsumption axioms between homogeneous metamodel entities. To do so, we introduce a third model, called `Intermodel`, that keeps all intermediate metamodel entities that are necessary to implement the transformation rule, which is a proper fragment of the complete metamodel. This implies that the original link has no direction, without source and target models. It is just a link relating entities in two models, and we have to cope with this difference.

Prioritising the rules that are useful based on the most used entities, we describe the rules for Attribute $\leftrightarrow$ Object type, Attribute $\leftrightarrow$ Single identifier, Object type $\leftrightarrow$ Relationship, and Weak Object Type $\leftrightarrow$ Object Type, whereas the rule for Attribute $\leftrightarrow$ Value type has been presented already in [6]. The here omitted transformations can also be described as ATL rules.

In the Attribute $\leftrightarrow$ Object Type rule, an attribute $A \mapsto C \times \mathtt{D}$ becomes an object type A' with a new stub attribute $a \mapsto A' \times \mathtt{D}$ and has a relationship R

```
rule Att<-->OT {
 from
     a : Model1.MM!Attribtue (a.range(dt)),
     o : Model2.MM!ObjectType
 to
     newO : InterModel.MM!ObjectType ( newO.hasAttribute <-- a1 ),
     a1 : InterModel.MM!Attribute ( a1.domain <-- newO,
              a1.range<--dt, a1.of <-- co ),
     e : InterModel.MM!EqualityConstraint( e.declaredOn(a),
               e.declaredOn(a1)),
     co : InterModel.MM!CardO ( co.cardinalityConstraint <-- cc,
             co.attribute <-- a1, co.objectType <-- newO ),
     sid : InterModel.MM!SingleIdentification ( sid.declaredOn <-- a1,
             sid.identifies <-- newO, sid.mandatory <-- mc),
     m : IntereModel.MM!Mandatory ( m.declaredOn <-- a1.contains ),
     cc : InterModel.MM!CardinalityConstraint (cc.maximumCardinality <-- 1,
             cc.minimumCardinality <-- 1),
     s : InterModel.MM!Subsumption ( s1.super <-- newO, s1.sub <-- o )
}
```

Fig. 3. Attribute ↔ Object Type transformation specified as an ATL rule.

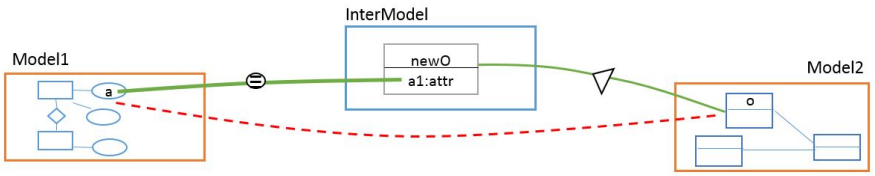

Fig. 4. Graphical rendering of the rule Attribute↔Object Type output; see text for details.

to an object type C. The rule in ATL-style notation is depicted in Fig. 3, and the intuition of the rule is depicted in Fig. 4. It introduces a new object type (`newO`) in `InterModel`, which is a "proxy" element for the object type and it is identified by the new attribute `a1` which is equivalent to the original attribute `a` from `Model1`. The specification of `a1` includes its domain, range, and the fact that it is an identifier for `newO`. The latter is characterised by a SingleIdentification constraint (a mandatory and a 1:1 cardinality constraint between the attribute and the object type). A subsumption between `newO` and the original object type `o` in `Model2` closes the connection between the original two elements.

Any automated reasoning results on each of the (formalised) original models —obtained by, e.g., a Description Logic-based reasoner—do not change by this transformation rule. Considering the three models together, we have a one-to-one correspondence between attribute values and object type instances. In case the connected object type exhibits additional constraints that are not consistent with the identification constraint in its attribute, an automated reasoner would detect the inconsistency of the conjoining model, and the tool would suggest the user to remove it or change the constraints.

The Attribute ↔ Single identifier rule is shown in Fig. 5. There is a 'silent' data type (placeholder), so only the equality between the attributes and the identification constraint for the non-key attribute has to be added to `InterModel`.

```
rule Att<-->ID {
  from
     a1 : Model1.MM!Attribtue ( a1.domain(o1) ) ,
     a2 : Model2.MM!Attribute ( a2.range(dt), a2.domain(o2) ),
     ic2 : Model2.MM!IdentificationConstraint ( ic.declaredOn(a2),
              ic.identifies(o2) ),
  to
     e : InterModel.MM!EqualityConstraint( e.declaredOn(a1),
               e.declaredOn(a1)),
     ic : InterModel.MM!IdentificationConstraint (
              ic.declaredOn(a1), ic.identifies(o1) ),
     co : InterModel.MM!CardO ( co.cardinalityConstraint <-- cc,
              co.attribute <-- a1, co.objectType <-- o1 ),
     cc : InterModel.MM!CardinalityConstraint (
              cc.maximumCardinality <-- 1,
              cc.minimumCardinality <-- 1 ),
     m : InterModel.MM!Mandatory ( m.declaredOn <-- a1.contains,
             a1.contains.plays <-- dt )
}
```

Fig. 5. Attribute ↔ Id transformation specified as an ATL rule.

```
rule OT<-->Rel {
  from
     o : Model1.MM!ObjectType,
     r : Model2.MM!Relationship
  to
    newRel : InterModel.MM!Relationship ( newRel.contains(newRol) ),
    newRol : InterModel.MM!Role ( newRol.linkedTo(rp) ),
    cc : InterModel.MM!CardinalityConstraint (
             cc.maximumCardinality <-- 1,
             cc.minimumCardinality <-- 1, cc.of(rp) ),
    rp : InterModel.MM!RolePlaying (rp.plays(o) ),
    s : InterModel.MM!Subsumption (
          s.super(newR), s.sub(r) )
}
```

Fig. 6. Object Type ↔ Relationship transformation specified as an ATL rule.

A mandatory and 1:1 cardinality constraints must also be included. Reasoning services on the conjoining three models would result in an equivalence axiom between the containing object types. In case constraints attached to both attributes are not consistent, the tool would suggest either to remove the rule or to modify the constraints.

Regarding the Object type ↔ Relationship transformation, the rule introduces in `InterModel` a new Relationship `newR` and a new Role `newRol` that holds a unique 1:1 role attached to the object type (see Fig. 6). The original relationship is subsumed by this new relationship. Subtle issues relating participating constraints for the original object types and relationships may arise when reasoning is applied to the conjoining three models, and several changes may be suggested by the tool. For example, new cardinality constraints, subsumption, or equivalences may appear, in addition to inconsistencies between them. This shows the necessity in the tool for both graphical editing and reasoning services.

Finally, the Weak Object Type ↔ Object Type transformation involves creating in `Intermodel` a new object type which inherits the identification constraints of the original object type in `Model2`, and making it a subsumee of the weak object type. The rule is shown in Fig. 7.

```
rule WOT<-->OT {
  from
     w : Model1.MM!WeakObjectType,
     o : Model2.MM!ObjectType
  to
    newO : InterModel.MM!ObjectType,
    s1 : InterModel.MM!Subsumption (
          s.super(o), s.sub(newO) ),
    s2 : InterModel.MM!Subsumption (
          s.super(newO), s.sub(w) )
}
```

Fig. 7. Weak Object Type ↔ Object Type transformation specified as an ATL rule.

All the rules described here, as well as those rules of the 'half' links type, involve possible updates in the original models after reasoning over the whole set of models that essentially form one logical theory in the background. Constraints from one model may propagate to the other through the proposed links. A conceptual model design tool that follows this approach will have to present the changes to the user together with supporting justifications and the designer would have to decide to accept the changes, or delete the links.

To conclude this section, it is important to remark that we have shown a way to specify common transformation links between heterogeneous entities in different conceptual models, in a first step without altering their meaning. The transformations are specified as ATL rules, showing its feasibility for representation in any other related formalism. These rules together with the policy in covering the rest of the links described at the beginning of this section, cover the most important links identified in our experiment.

4 Related work

The general problem addressed in this paper is not new, especially works on 1:1 transformations, but there are scant results on intermodel assertions across conceptual data modelling languages and pairing different types of element in a sound way. Atzeni et al. [1] has similarities to our approach, in the sense of using a "supermodel", but a scope of only transforming, say, an ER model into a UML Class diagram—rather than also intermodel assertions between them—and no 'type transformations'. It also covers fewer types of entities, and glosses over subtle issues such as ER's identifier and a UML attribute that ought to have had an {id}. Their follow-up paper provides an in-depth formal framework to handle rules with Datalog and reason over them [2], which may be useful, but not the rules either, i.e., not what exactly should be verified. The other, and more application-oriented, system is the Pounamu tool for visual modelling [20], which perhaps could be extended with the here presented transformation rules, provided the metamodel would be extended with the more recent language features (like UML's {id}) and the rules added. We also considered Eclipse's metamodel of the Eclipse Modeling Framework [`https://eclipse.org/modeling/emf/`], so as to recast our metamodel and the UML, EER, and ORM2 fragments as

Eclipse models, but it is not expressive enough to represent them, and therewith constrain the rules. For instance, the EMF metamodel does not deal with roles, relationships, and cardinalities, or constrain attributes to be declared only for classes and relationships, which, however, are necessary to be declared somewhere in order to enable validation of intermodel assertions.

Concerning the representation of the rules for the entity conversions, several proposals other than ATL exist to specify model transformations as a whole or of certain elements. From a rigorous logic-based viewpoint, Distributed Description Logics (DDL) might be an option, and a few types of conversions have been defined at an abstract level, covering concept↔role and attribute↔role using so-called "bridge rules" [9]. These two types of transformations do not cover the full range needed for intermodel assertions in conceptual models, nor do the DDL DLs have all the features of the main conceptual modelling languages. Module interaction with a logic-based approach has been investigated for the Semantic Web as well. OWL itself only supports whole-module imports [18], however, and applied ε-connections are used for 1:1 mappings only [4]. The Distributed Ontology Language (DOL, `http://ontoiop.org`) [17] may be useful, as it provides a language to integrate logic-based models that may be represented in different languages. DOL was accepted for standardisation by the OMG in March 2016 and is in the preliminary stages regarding the software infrastructure and conformance of logics suitable for conceptual data modelling languages.

Model Driven Engineering typically uses any of graph, rule, or imperative-based languages for model transformations, such as Triple Graph Grammars (TGG) [10], OMG's Query/View/Transformation (QVT) [19], and Eclipse's Atlas Transformation Language (ATL) [12] that is a modified version of OMG's OCL. QVT was designed principally for a UML-to-relational mapping, and is thus difficult to reuse for our setting. TGG seems exceedingly suitable, but either the underlying formalism will have to be integrated with the metamodel first, or, if the diagrammatic option is chosen, be manually redesigned for implementation in ICOM, thus requiring double work, and with its main implementation in Eclipse, then still faces thosee limitations as mentioned above. ATL is implementation-oriented and tailored to handling data types, with an intuitive notation and very similar to our implementation-independent rule-based notation of the metamodel-mediated rules-based approach [6]. Therefore, we used ATL-style notation in the type conversion rules. While there indeed is a general downside to ATL of having to know the metamodel (compared to the concrete syntax-based graph transformation and Attributed Graph Grammar) [11], we do know it and the rules have to be specified only once for system implementation, not by users of the intermodel assertions, hence that downside is not applicable. Purely implementation-oriented approaches, such as the type transformations for programs using lambda calculus and Haskell [16], are too narrowly focussed and therewith not easily adaptable to the generic conceptual modelling setting.

5 Discussion

Design decisions for each conceptual model are usually taken in the isolated context of the application for that model. When the time arrives to integrate it with other models, a gap between different representations must be bridged using intermodel links. Our exploration for their usage showed that a wide variety of links are needed, ranging from trivial equivalence to complex transformation between model elements. For the links used, and in order to answer questions 1a and 1b from Section 1, links between homogeneous entities are used most widely, of which equivalence axioms are the majority. They are followed by the 'half-links', subsumption, transformation, implied subsumption, and disjointness. From the point of view of tools, currently there is lack of support for all of these links The diversity of these links shown in Section 2 make it necessary for tools to improve the assistance in developing the correct balance between a coherent and close model integration and the preservation of each individual model semantics.

The experimental evaluation also showed that with actual projects, subsumptions are used most, while the integration simulation scenarios brought to the fore the links between attributes—not available in tools—with as close second object types, and there were many more equivalences in the integration scenarios than in the ICOM projects. This difference may be attributed to the low number of projects and, perhaps (not tested), the modeller. The experimental evaluation projects were created for either integration purposes or modules for managing cognitive overload. Both equivalence and subsumption are considered the most frequently linked relationship type for the integration projects while subsumption is the most frequently linked relationship type for the cognitive overload projects, thus, for the ICOM projects, there is no significant correlation between the type of project and the links in them.

The main issues with links between elements in the models revolved around attributes, with mismatches on datatype and cardinality. We had expected more Attribute $\leftrightarrow$ Object type and Relationship $\leftrightarrow$ Object type across-type links, as such decisions feature prominently in the modelling process. Why this is not the case is an aspect of further investigation. One could evaluate more models, though the number was substantial in the experiment, and perhaps retrieve real models from industry. That said, one faces a chicken-and-egg problem with the experimental approach in this case: if the feature is not available—such as the advanced intermodel links—then it will not be used so will be hard to find, and one would need a tool to check whether the links in the scenarios are correct, but the counting of the links needs to feed into the tool development so as to assert them.

Most of the rules can be easily incorporated in integration tools with subsumption, equivalence, and disjointness axioms between homogeneous elements. However, there is still space for complex rules that require more elaborate mechanisms in order to be supported. The ATL rules in Section 3 express these mechanisms in an implementation-oriented way.

Rule results are represented in a separate, intermediate model which holds all new elements. This scheme allows for both preserving each model semantics while

making feasible a closer integration. Close integration with reasoning services are necessary for tools in this case so as to go beyond the syntax and semantics of the modelling languages and also deduce useful information about the consistency of the linked model. ATL rules can be easily modified or extended in case the result patterns require different translations. In this light, connections with ontology patterns [5] are left as possible future work.

A general issue with model transformation is testing for correctness [3] to answer the question: will the metamodel + ATL rules do the right thing? While our metamodel is complex, it is formalized for easier processing where its constraints direct the checking of the intermodel assertions [6]. For the basic transformation rules, an implementation to compare transformation outputs with an oracle—deemed a problem in [3]—will not be an issue practically despite that the graph isomorphism problem is NP-complete, because the scope of an intermodel assertion is a small fragment of the model localized to the entities involved in that intermodel assertion, not the whole model. We are currently implementing the first step—models related to the metamodel—in the ICOM tool [8].

Regarding verification of the models, it is possible to use the metamodel to verify the models' syntax and send the portion of the models that fall within a suitable decidable fragment of first order logic to the automated reasoner to detect inconsistencies and other deductions, which is already possible in ICOM [7]. While not all language features can be formalised in a decidable language, most of those computationally thorny features (e.g., antisymmetry) are not used anyway [13], hence, this is a feasible solution.

Finally, while the details are becoming quite tedious, it will result in an easy interface that hides all the technicalities, syntax, and ontological issues, so that the modeller can focus on the universe of discourse.

6 Conclusions

Intermodel assertions are typically more equivalence than subsumption assertions, and mainly among classes and among attributes. When the modeller has the flexibility, there are also links between different types of language features, such as attribute↔value type, attribute↔object type, and plain attribute ↔ composite attribute. To be able to handle such assertions in a modelling tool, we availed of the unifying metamodel and creatively used the ATL language in particular to declare rules for the intermodel assertions, thereby bridging this semantic gap. This is achieved by transforming the relevant fragment of the source models into a temporary ATL target model that is a proper fragment of the metamodel in order to check whether the assertion is acceptable. We are currently implementing a proof-of-concept of this approach by extending the ICOM tool. We also aim to work on a proof of correctness of transformation rules.

Acknowledgments This work is based in part upon research supported by the National Research Foundation of South Africa (Project UID90041) and the Argentinean Ministry of Science and Technology.

References

1. Atzeni, P., Cappellari, P., Torlone, R., Bernstein, P.A., Gianforme, G.: Model-independent schema translation. VLDB Journal 17(6), 1347–1370 (2008)
2. Atzeni, P., Gianforme, G., Cappellari, P.: Data model descriptions and translation signatures in a multi-model framework. AMAI 63, 1–29 (2012)
3. Baudry, B., Ghosh, S., Fleurey, F., France, R., Le Traon, Y., Mottu, J.M.: Barriers to systematic model transformation testing. Comm. ACM 53(6), 139–143 (2010)
4. Cuenca Grau, B., Parsia, B., Sirin, E.: Combining OWL ontologies using ε-connections. J. Web Sem. 4(1), 40–59 (2006)
5. Falbo, R.A., Guizzardi, G., Gangemi, A., Presutti, V.: Ontology patterns: clarifying concepts and terminology. In: Proc. of OSWP'13 (2013)
6. Fillottrani, P.R., Keet, C.M.: Conceptual model interoperability: a metamodel-driven approach. In: Proc. of RuleML'14. LNCS, vol. 8620, pp. 52–66. Springer (2014), aug 18-20, 2014, Prague, Czech Republic
7. Fillottrani, P.R., Franconi, E., Tessaris, S.: The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web J 3(3), 293–306 (2012)
8. Fillottrani, P.R., Keet, C.M.: A design for coordinated and logics-mediated conceptual modelling. In: Proc. of DL'16. p. (in print). CEUR-WS (2016), 22-25 April, 2016, Cape Town, South Africa
9. Ghidini, C., Serafini, L., Tessaris, S.: Complexity of reasoning with expressive ontology mappings. In: Proc. of FOIS'08. FAIA, vol. 183, pp. 151–163. IOS Press (2008)
10. Golas, U., Ehrig, H., Hermann, F.: Formal specification of model transformations by triple graph grammars with application conditions. Elect Comm of the EASST 39, 26 (2011)
11. Grønmo, R., Møller-Pedersen, B., Olsen, G.K.: Comparison of three model transformation languages. In: Proc. of ECMDA-FA'09. LNCS, vol. 5562, pp. 2–17. Springer (2009), 23-26 June, 2009, Enschede, The Netherlands
12. Jouault, F., Allilaire, F., Bzivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming 72(12), 31 – 39 (2008)
13. Keet, C.M., Fillottrani, P.R.: An analysis and characterisation of publicly available conceptual models. In: Proc. of ER'15. LNCS, vol. 9381, pp. 585–593. Springer (2015), 19-22 Oct, Stockholm, Sweden
14. Keet, C.M., Fillottrani, P.R.: An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2. Data & Knowl Eng 98, 30–53 (2015)
15. Khan, Z.C., Keet, C.M.: An empirically-based framework for ontology modularization. Applied Ontology 10(3-4), 171–195 (2015)
16. Leather, S., Jeuring, J., Lh, A., Schuur, B.: Type-changing rewriting and semantics-preserving transformation. Sci Comp Prog 112, 145–169 (2015)
17. Mossakowski, T., Kutz, O., Codescu, M., Lange, C.: The distributed ontology, modeling and specification language. In: Proc. of WoMo'13. CEUR-WS, vol. 1081 (2013), corunna, Spain, September 15, 2013
18. Motik, B., Patel-Schneider, P.F., Grau, B.C.: OWL 2 web ontology language: Direct semantics. W3C recommendation, W3C (27 October 2009 2009), `http://www.w3.org/TR/owl2-direct-semantics/`
19. Object Management Group: Meta Object Facility (MOF) 2.0 – Query/View/Transformation Specification. `http://www.omg.org/spec/QVT/1.2`
20. Zhu, N., Grundy, J., Hosking, J.: Pounamu: a metatool for multi-view visual language environment construction. In: Proc. of VLHCC'04 (2004), rome, 25-29 Sept. 2004