Methods for Semi-automatic Construction of Information Models Transformations

Sergey Stupnikov, Leonid Kalinichenko

Institute of Informatics Problems, Russian Academy of Science
ssa@ipi.ac.ru, leonidk@synth.ipi.ac.ru

Abstract. The diversity of existing information models is high. The process of manual construction of model transformation is labor-intensive. Therefore the problem of automation of information models transformation construction remains to be of high importance. The paper proposes two methods for semi-automatic construction of model transformations. The first method is aimed at construction of forward transformations on the basis of established correspondences between elements of the models. The second method is aimed at construction of backward transformations of models on the basis of forward transformations. The methods for semi-automatic construction of model transformations are parts of more general work — development of a Unifying Information Models Constructor (*Model Unifier* in short). The main practical task of the Unifier is a mapping of models of heterogeneous information resources into the canonical information model during the integration of heterogeneous resources.

1 Introduction

Current period of IT development is characterized by a substantial growth of the number of information models. New models appear and existing models evolve, such as data models (e.g., ODMG 2000, SQL 2003, UML, XML stack), workflow models, process service composition languages, semantic models, etc. A need for interoperable use (integration) of information resources represented in heterogeneous models in various applications, as well as their reuse and composition implementing new interoperable information systems [1] become very urgent. Generally resources are heterogeneous (represented in various models). So the problems of resource integration require to unify these models in the frame of some information model called *canonical*. This semantic unification of various models specifications in the canonical model requires special methods and tools based on semantics of models and allowing to reason provably about models.

The methods considered in this paper[1] are a part of more general work — creation of Unifying Information Models Constructor (*Model Unifier* in short) [2].

[1] This research has been done under the support of the RFBR (project 08-07-00157- a) and the Program for fundamental research of the Presidium of RAS, section 4 (project *Methods and tools for a subject mediator middleware providing for problem solving over set of heterogeneous distributed information resources*).

Main objective of the Unifier is to provably reduce a set of heterogeneous resource information models to the canonical unifying representation for the integration of information resources in subject mediators [1]. A process of information model *unification* includes construction of a transformation of the model into the canonical one and a verification of this transformation. The transformation is correct if the resource model *refines*[2] the canonical one [2].

Development of the Unifier prototype includes research of techniques for construction of information models transformations. In [2] the results of application of declarative metacompilation languages SDF (Syntax Definition Formalism) and ASF (Algebraic Specification Formalism) for the formal definition of model syntax and model transformation were reported. The ASF and SDF languages are supported by specific tools [3]. This paper presents some results of applying in the Unifier of another kind of model transformation languages which combine declarative and imperative facilities. Such languages are developed in frame of Model-Driven Architecture (MDA). The MDA approach is supported by the OMG MOF standard. QVT (Query-View-Transformation) [4] — model transformation standard proposed by OMG is an example of such languages.

Analysis in [2] showed that the model unification requires new methods for semi-automatic model transformation construction applying MDA technologies (in particular, QVT-like languages). The work presented in this paper is aimed at development of the required methods. The process of manual construction of the model transformation is labor-intensive. Therefore the problem of automation of information models transformations construction remains to be of high importance. In this work two aspects of automation of transformation construction are considered:

- construction of forward transformations on the basis of established correspondences between elements of the models;
- construction of backward transformations of models on the basis of forward transformations.

Correspondences between elements of models can be established in different ways. For instance, ontological approach [5] or schema matching can be used. In this paper a way of establishing these correspondences is not concretized. Instead a method for construction of transformations on the basis of somehow established correspondences between elements of the models is considered. Note that the information contained in a set of correspondences may be not sufficient for construction of a complete transformation. In this case an expert should complete the transformation according to the semantics of models using automatically generated transformation as a pattern.

The aim of the second method considered in this paper is to construct backward transformations of models on the basis of forward transformations. Consider two models S and T. Suppose a transformation $S2T$ of S into T is already constructed (applying the mentioned method for construction of transformations

[2] It is said that specification A *refines* specification D, if it is possible to use A instead of D so that the user of D does not notice this substitution.

on the basis of correspondences between elements of the models or manually). Let us call this transformation *forward*, *S* being a source model, *T* being a *target* model. The problem is to semi-automatically construct the transformation *T2S* of the model *T* into the model *S*. This transformation is called backward and is to be constructed on the basis of *S2T* transformation specification. The method for construction of backward transformations as well as the method for construction of transformations on the basis of correspondences between elements of the models allows to automatize the construction of transformations only partially. Facilities of model transformation languages differ in complexity and semantics. Some facilities allow to produce automatically the respective constructions of backward transformation. For other facilities a respective construction has to be produced manually on the basis of semantics of models.

In this work the ATL (ATLAS Transformation Language) [6] is used as a model transformation language. Type system of the ATL is very close to the type system of the OCL. For metametamodel (M3-layer, according to MOF architecture) the Ecore [8] is considered in this work.

The methods considered in this paper can be adapted for construction of transformations applying model transformation languages similar to ATL (for instance, QVT or its dialects). Also the choice of a specific M3 metamodel is not essential. Other metamodels such as MOF or KM3 can also be used. The methods proposed can be also used independently of the Unifier for the non- verifiable construction of forward and backward transformations of information models in the MDA context.

The paper is structured as follows. In section 2 the method for construction of transformations on the basis of established correspondences between elements of the models is considered. The method for construction of backward transformations of models on the basis of forward transformations is considered in section 3. In section 4 the related works are considered. In conclusion the contribution of the work is summarized.

2 Construction of Transformations on the Basis of Established Correspondences Between Elements of the Models

In the paper the transformations between the OWL (Web Ontology Language, [10]) and the SYNTHESIS [9] language (considered as a canonical information model for subject mediators [1]) are presented as examples. OWL is treated as a source model and SYNTHESIS — as a target model. Only fragments of the languages are considered.

Consider an example of input data for the semi-automatic construction of transformation of OWL into SYNTHESIS (i.e. the list of correspondences between models elements). OWL and SYNTHESIS were represented as M2 models conforming to Ecore metamodel. On figure 1 some elements of the models are shown (to make models more readable they are represented as UML class diagrams). Elements of the models are classes (for instance, *OWLOntology* — ele-

ment of type *EClass* in Ecore), attributes (for instance, *URIReference.name* — element of type $EAttribute$ in Ecore), associations (for instance, *Graph.ontology* — element of type $EReference$ in Ecore).

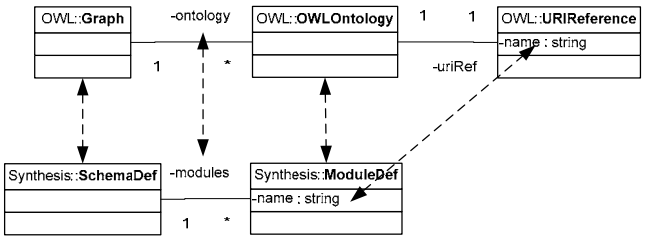

Fig. 1. Correspondences between OWL and the SYNTHESIS models elements

A kernel element of OWL specification is graph (*Graph*) consisting of ontologies (*OWLOntology*). A kernel element of SYNTHESIS specification is schema (*SchemaDef*) consisting of modules (ModuleDef).

The correspondences between elements are denoted by dotted lines. The correspondences can be established between classes (*Graph* corresponds to *SchemaDef*), between associations (*OWLOntology.ontology* corresponds to *SchemaDef.modules*), between attributes (*OWLOntology.uriRef.name* corresponds to *ModuleDef.name*). An element of a model can correspond to several elements of the other model.

Generally the set of correspondences is a subset R of a cartesian product $E_S \times E_T$ Here E_S is a set of elements of the source model (OWL), E_T is a set of elements of the target model (SYNTHESIS).

The aim of the method proposed in this section is to construct automatically the most accurate as possible transformation of a source model into a target model on the basis of a set R of correspondences between elements of the models.

A transformation is specified by means of ATL *modules*. A module is composed of a header section (defining the name of the transformation module and the names of the variables corresponding to the source and target models) and a set of rules that defines the way target models are generated from the source ones. The *matched rules* constitute the core of a transformation. They allow to specify: (1) for which kinds of source elements target elements must be generated, (2) the way the generated target elements have to be initialized.

The method is based on a set of *generation rules* for the automatic construction of a transformation module from a given source model S into a given target model T:

– a generation rule for construction of the header of the transformation;
– generation rules for construction of skeletons of ATL matched rules constituting the transformation (skeleton consists of a name and source and target elements of a rule);
– generation rules for construction of initializations of target elements in transformation rules.

An element of the source model can correspond to several elements of the target model and vice versa. For different cases different generation rules of rule skeleton construction and target element initialization construction are provided.

To save space the general form of the generation rules is omitted. The generation rules are only briefly illustrated by the example of transformation of OWL into SYNTHESIS. The header of the transformation looks as follows:

```
module OWL2Synthesis;
create OUT: Synthesis from IN: OWL;
```

Transformation rules generated on the basis of element correspondences shown on figure 1 look as follows:

```
rule Graph2SchemaDef{
  from g: OWL!OWLGraph
  to s: Synthesis!SchemaDef( modules <- g.ontology )
}
rule OWLOntology2ModuleDef{
  from o: OWL!OWLOntology
  to m: Synthesis!ModuleDef( name <- o.uriRef.name )
}
```

A correspondence between classes (for instance, $Graph \sim SchemaDef$) leads to a construction of a rule (*Graph2Schema*). A correspondence between attributes (associations) leads to an initialization of a target attribute (association). For instance, a correspondence between *Graph.ontology* and *SchemaDef.modules* associations leads to the initialization $modules <- g.ontology$. The initialization means that for every instance of a collection *g.ontology* (they are instances of *OWLOntology*) the respective instance of a collection modules is produced. This is done with the help of the rule *OWLOntology2ModuleDef* which transforms *OWLOntology* instances into *ModuleDef* instances.

Note that the mentioned correspondences between the elements of the models almost always contain only a part of the semantics of the required model transformation. For instance, elements may be in a correspondence but a transformation may depend on some complicated condition. The additional semantics should be added to an automatically constructed transformation manually. So the construction of the transformation consists of two sequential steps:

- automatic construction of a transformation on the basis of correspondences between elements of models (transformation constructed is called a pattern);
- manual modification of the pattern transformation according to the semantics that is not contained in the set of correspondences.

3 Construction of Backward Transformations of Models on the Basis of Forward Transformations

Consider a transformation t of a model S into a model T and a transformation r of T into S.

Definition 1. A transformation r is backward for a forward transformation t if for any model s conforming to S the following condition holds: $r(t(s)) = s$.

Definition 2. The transformation r is weakly backward for t if for any model s conforming to S the following condition holds: $t(r(t(s))) = t(s)$.

A backward transformation can be constructed only for a forward transformation t that transfers all the information contained in a source model. It is not the case for a weakly backward transformation: it is just satisfied with the information transferred by the forward transformation. Obviously a backward transformation is a weakly backward one. A backward transformation is possible if every initialization in every rule binds an attribute of a target with exactly one attribute of a source model. In some cases it is possible to weaken this condition.

A backward transformation is a proper transformation not requiring an expert intervention. A weakly backward transformation is a pattern transformation to be suggested to an expert for checking the transformation and manually modifying it if required.

Definition 3. A subset L of transformations represented in some model transformation language (for instance, ATL) is (weakly) reversible if an algorithm exists that for any $l \in L$ constructs a (weakly) backward transformation r for l.

Definition 4. A subset K of facilities (operations, functions, statements) of a model transformation language is (weakly) reversible if a set of transformations defined with the help of K is (weakly) reversible.

Probably any model transformation language (for instance, ATL) as a whole is weakly reversible. But weak reversibility makes sense only for a part of the language facilities.

The problem of automation of backward transformations construction is reduced to several subproblems:

- find a maximal (weakly) reversible subset K_R of a model transformation language facilities;
- develop an algorithm for construction of the backward transformations (when the forward ones are defined with the help of K_R). An algorithm may have a form of a semantic function r[] which converts a construct of a forward transformation into a construct of a backward transformation. Such conversion is called reverse;
- prove a correctness of the algorithm. Proof is done inductively over facilities of the language.

A set of generation rules for construction of backward transformations for a subset of ATL language has been developed. A proof of correctness is a future work.

The set of the generation rules developed includes reversing of the following ATL facilities:

– header of a transformation;
– helpers (ATL equivalents for Java methods);
– matched rules;
– local variables in rules;
– initializations of target elements in declarative and imperative parts of the rules;
– if-then statement and assignment statement in initializations;
– path expressions in source model elements;
– if-then and let OCL expressions;
– OCL primitive data types operations: logical negation; addition, subtraction, toString operation of numerical types; multiplication, division, sqrt, exp, log, toDegrees, toRadians of Real type; toInteger, toReal, toSequence, split, concat (in a special case) of String type;
– toSequence operation of Set, OrderedSet, Bag OCL types; toSet operation of OrderedSet, Bag types;
– append, prepend, insertAt operations of Sequence OCL type;
– any and select operations of collection OCL types.

For some facilities only a weak reversing is possible (conditional statement, any and select operations).

The objective of future work includes a study of the possibility to reverse such ATL facilites as called rules (imperative rules), iterative target pattern of a rule, the for statement and some others.

Due to the lack of space the generation rules of backward transformations construction are illustrated only by a small example. Consider a backward translator for a subset of forward translator of OWL into SYNTHESIS considered in section 2:

```
module Synthesis2OWL;
create OUT: OWL from IN: Synthesis;
rule SchemaDef2Graph{
 from s: Synthesis!SchemaDef
 to g: OWL!OWLGraph
 do{ g.ontology <- s.modules; }
 }
rule ModuleDef2OWLOntology{
 from m: Synthesis!ModuleDef
 to o: OWL!OWLOntology, u: OWL!URIReference
 do{ o.uriRef <- u; u.name <- m.name; }
}
```

The example illustrates an idea of reversing of path expressions. To reverse a path o.uriRef.name in the forward rule (OWLOntology2ModuleDef) it is required to add an element (u: OWL!URIReference) to a target pattern of the backward rule (ModuleDef2OWLOntology) and properly initialize it in an imperative do section of the rule.

The construction of a backward transformation consists of two sequential steps:

– automatic construction of a transformation on the basis of a forward transformation. Every weakly reversible construction of the forward transformation is reversed, other constructions are marked;
– manual modification of the automatically produced transformation. Reversible constructions do not need any modifications (due to the generation rules of constructions they are constructed properly). Weakly reversible constructions are checked and modified if required. Marked constructions are reversed manually.

4 Related Work

Existing approaches for automated model transformation construction include usually as their part some model matching methods. In [5] ontology-based model transformation (ontMT) is proposed. It integrates ontologies in modeling by utilizing MDA and Ontology technological space to automate the generation and evolution of model transformations. A source and a target models are binded with a reference ontology. Relations between concepts of the models are cal- culated and bidirectional transformation of models expressed in QVT relations language is generated.

The work described in [11] presents a semi-automatic approach for the development of transformations via weaving models of the model weaving approach. It describes an iterative procedure of weaving link generation, similarity calculation, and weaving link selection. The weaving model is translated by high-order transformations into transformation model (ATL metamodel).

In [12] a by-example approach is proposed for defining mappings on the M1 layer between concrete domain models which incorporates the notation of modeling languages and allows the generation of model transformation code based on the M2 layer. The information gathered in the mappings are used to generate two-way ATL transformations.

The approach proposed in this paper is more technically oriented. It concentrates on generation of a code of a model transformation and abstracts from model matching techniques. A set of generation rules is provided for the automatic construction of a model transformation on the basis of a specific class of correspondences between elements of the models. Some advanced facilities of ATL are used to construct transformations. A method for construction of a back- ward transformation allows to reverse a transformation. It is especially useful when a transformation is not initially bidirectional.

5 Conclusion

The problem of automation of information models transformations construction remains to be of high importance due to the number of various information models. This paper proposes two methods for semi-automatic construction of model transformations. The first method is aimed at construction of forward transformations on the basis of established correspondences between elements of

the models. The second method is aimed at construction of backward transformations of models on the basis of forward transformations. The ATL language (an analogue of the OMG QVT) is applied as a model transformation language.

The methods for semi-automatic construction of model transformations are parts of more general work — development of the Unifying Information Models Constructor. The main practical task of the Unifier is a mapping of the models of heterogeneous information resources into the canonical information model during the heterogeneous information resources integration.

The methods proposed can be adapted for construction of transformations applying model transformation languages similar to ATL (for instance, QVT). Also the choice of a specific M3 metamodel is not essential. The methods can be used independently of the Unifier for the non-verifiable construction of forward and backward transformations of information models in the MDA context.

References

1. Kalinichenko L. A., Briukhov D. O., Martynov D. O., Skvortsov N.A., Stupnikov S.A.: Mediation Framework for Enterprise Information System Infrastructures. Proc. of ICEIS, 246–251 (2007)
2. Kalinichenko L. A., Stupnikov S. A.: Constructing of Mappings of Heterogeneous Information Models into the Canonical Models of Integrated Information Systems. Proc. of ADBIS. – Pori: Tampere University of Technology, 106–122 (2008)
3. Van den Brand M. G. J. et al.: The ASF+SDF meta-environment: a component-based language development environment. Compiler Construction 2001 / Ed. by R. Wilhelm. – Springer, 365–370 (2001)
4. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification (2007), http://www.omg.org/cgi-bin/doc?ptc/2007-07-07
5. Roser S., Bauer B.: Automatic Generation and Evolution of Model Transformations Using Ontology Engineering Space. J. Data Semantics (JODS) 11:32-64 (2008)
6. ATL Project, http://www.eclipse.org/m2m/atl/
7. OMG/OCL Object Constraint Language (OCL) 2.0. OMG Final Adopted Specification. ptc/03-10-14 (2003)
8. Budinsky, F., Steinberg, D., Ellersick, R., Grose, T. Eclipse Modeling Framework, Chapter 5 Ecore Modeling Concepts. – Addison Wesley Professional, (2004)
9. Kalinichenko L. A., Stupnikov S. A., Martynov D. O. SYNTHESIS: a Language for Canonical Information Modeling and Mediator Definition for Problem Solving in Heterogeneous Information Resource Environments. – M.: IPI RAS. – 171 p. (2007), http://synthesis.ipi.ac.ru/synthesis/publications/07synthesis
10. OWL Web Ontology Language Reference. W3C Recommendation (2004), http://www.w3.org/TR/owl-ref/
11. Fabro, M.D.D., Valduriez, P.: Semi-automatic model integration using matching transformations and weaving models. In: 22nd ACM Symposium on Applied Computing, Model Transformation Track, (2007)
12. Wimmer M., Strommer M., Kargl H., Kramler G.: Towards Model Transformation Generation By-Example. HICSS (2007)