Liet. matem. rink, **47**, *spec. nr.*, 2007, 208–214

Correctness of the transformation: transformation of ontology axioms in formal rules*

Olegas VASILECAS, Diana BŪGAITĖ (VGTU)
e-mail: olegas@fm.vgtu.lt, diana@isl.vgtu.lt

Abstract. In this paper, authors emphasise on the correctness of the transformation. Therefore, types of correctness are analysed here. As a context of transformation, the transformation of PAL constraints in SQL rules is chosen for more details. Formal rules for the transformation of PAL constraints in SQL rules are presented in this paper also. And finally, the analysis of the proposed transformation is checked.

Keywords: correctness of the transformation, ontology axioms, formal rules.

1. Introduction

In recent years, model-based software development processes have evolved. In this area, models are used for the generation of other models, for code generation, analysis, and simulation as well. The possibility to perform correct and automatic model transformation during reasonable short time is vital in the development of software applications. The Object Management Group (OMG) [1] is accomplishing this goal through the introduction of the Model Driven Architecture (MDA) with supporting detailed specifications.

In our research, we apply transformation to develop a rule model from the ontology (for more details see [2]). The knowledge-based information systems development using the domain ontology is the hot topic nowadays, since the semantic content expressed by ontology can be transformed in information systems artefacts, thereby reducing the costs of conceptual modelling [3]. In this context, researches are targeted on transformation of ontology in conceptual data model because they have some common aspects, i.e., both include concepts, relationships between concepts and rules (in ontology – axioms). However, a rule model, which is an important and integral part of each conceptual data model, is often neglected.

In this paper, authors emphasise on the correctness of the transformation. Therefore, the objective of this paper is to investigate is the proposed transformation correct.

2. Related work and background

According to [4], the correctness of model transformation is an important issue. This includes syntactical correctness, functional behaviour, and semantical correctness [4],

*The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)" Reg. No. B-07042.

[5]. It is hard to establish a single notion of correctness for model transformations. The most elementary requirements of a model transformation are syntactic.

- The minimal requirement is to assure **syntactic correctness** – is to guarantee that the generated model is a syntactically well-formed instance of the target model.
- An additional requirement is to assure **syntactic completeness** – is to completely cover the source model by transformation rules, i.e., to prove that there exists a corresponding element in the target model for each construct in the source model.

According to [6] to ensure the syntactical correctness of the output of a transformation it would be necessary to define a separate transformation language for every pair of source and target language. Such a transformation language would consist of the syntax definitions of the source and the target language and some language elements needed to be able to define the mapping. The synthesis of such a transformation language is always the same process and could therefore be automated.

Syntactic correctness and completeness was attacked in [7] by planner algorithms, and in [8] by graph transformation.

However, in order to assure a higher quality of model transformations, at least the following *semantic requirements* should also be addressed [5].

- **Termination:** The first thing we must also guarantee is that a model transformation will terminate. See also [9].
- **Uniqueness (Confluence, functionality):** As non-determinism is frequently used in the specification of model transformations (as in the case of graph transformation based approaches) we must also guarantee that the transformation yields a unique result. This is a language independent criterion. See also [9].
- **Semantic correctness (Dynamic consistency):** In theory, a straightforward correctness criterion would require to prove the semantic equivalence of source and target models. However, as model transformations may also define a *projection* from the source to the target (with deliberate loss of information), semantic equivalence between models cannot always be proved. Instead we define *correctness properties* (which are typically transformation specific) *that should be preserved by the transformation*.

According to [10] the topic about correctness is assigned to the category of formal methods. Owing to the formal foundation, the correctness of model transformations can be checked [4].

As stated in [11], transformation rules are correct in the sense that the target model is equivalent with the original (the proof of theorems can be find in [11]). The correctness of transformation of models can be checked by comparing the probability masses of target and source models. If they are equal, then transformation of models is correct.

A model transformation for translating a model from a source language in a target language can be defined by a set of compound rules [12]. Each such compound rule $r\colon\ (r_s, r_t)$ consists of two individual rules: The source transformation rule $r_s\colon\ L_S ::= R_S$ describes the transformation of the source model (with L_S representing the left hand side and R_{S} representing the right hand side), the target transformation rule $r_t\colon\ L_T ::= R_T$ specifies the transformation of the target model. Typically, source transformation rules will be identical transformations leaving the source model unchanged, represented as $r_s\colon\ L_S$ only.

Source and target rules are coupled by the ability to use shared variables. Such variables are denoted by <*variable*> [12]. Authors of [12] also briefly describe how a compound rule is applied, assuming that $L_S = R_S$ and that $X = \{x_1, .., x_n\}$ is the set of variables of L_S.

1. An occurrence of the left side L_S of the source transformation rule is searched within the source model, such an occurrence is called source match.
2. Having found a source match, the variables are given concrete values, leading to a variable instantiation denoted X^I .
3. The left side L_T of the target transformation rule is instantiated with the values of the variables, denoted also by $L_T(X^I)$.
4. An occurrence of the instantiated left side of the target transformation rule is searched within the target model. Such an occurrence is called target match.
5. The right side R^T of the target transformation rule is instantiated with the values of the variables.
6. The occurrence of the instantiated left side is replaced with the instantiated right side R^T of the target transformation rule.

To ensure the syntactic correctness of a transformation unit consisting of compound rules, we have to deal with the following problems [12]:

- With regards to a rule, both the source part as well as the target part must be syntactically correct.
- The variables used in the target part must also occur in the source part.
- With regards to a transformation unit, non-terminals created must also be deleted by later rule applications because otherwise the target model might contain these non-terminals.
- It must be ensured that all rules of a transformation unit are reachable, i.e., there are derivations that can make use of the rule.

According [13] **functional behaviour** of the model transformation based on graph transformation [12], [14] means that there are local confluence and termination of graph transformation which implies that model transformation defines a function from a source to a target. The model transformation is locally confluent, if the model transformation source language is restricted by the syntax grammar.

3. The context of transformation

As was said in Introduction, in our research, we apply transformation to develop a rule model from the ontology (for more details see [2]). Therefore, the formal expression of ontology axioms (and ontology as a whole) and formal rules is presented in this chapter.

Protégé Axiom Language (PAL) [15] constraints and Structured Query Language (SQL) triggers were chosen for the detailed study of transformation.

From [15] it was determined that axioms (A_i^P) expressed using PAL can be expressed as follows:

$$A^P = < A_i^P | i = 1, \ldots, k > \quad \text{with} \qquad (1)$$

$A_i^P =< PAL\text{-}name_i, PAL\text{-}documentation_i, PAL\text{-}RANGE_i', PAL\text{-}statement_i >$, where:

- *PAL-name* holds a label for the constraint.
- *PAL-documentation* holds a natural language description of the constraint.
- $PAL\text{-}RANGE = \{PAL\text{-}range_0, PAL\text{-}range_1, \dots, PAL\text{-}range_k\}$ holds definitions of local and global variables that appear in the statement. It is a class or a set of classes, defining the main concepts of domain in ontology.
- *PAL-statement* holds the sentence of the constraint.

$$PAL\text{-}statement =< if\text{-}part, then\text{-}part >, \tag{2}$$

where:

- $if - part$ is a statement or a set of statements, denoted by symbol '=>', which holds possible conditions of the state,
- $then - part$ is a statement or a set of statements, which comes after *if-part* and holds possible state or sometimes action in the domain.

$If - part$ is optional. The analysis of possible axioms, which can be defined using EZPal Tab plug-in [16], shows that $PAL - statement$ matches a possible *state* of a domain. There is no *action* if a new state of the domain does not violate the conditions defined in the axioms. *Action* is '*forbid change*' if a new state of the domain violates the conditions defined in the axioms.

A *statement* holds on a certain number of variables, which each range over a particular set of values. Therefore, a constraint in PAL consists of a set of variable range definitions and a logical statement that must hold on those variables. The language of PAL is a limited predicate logic extension of Protégé-2000 that supports the definition of such ranges and statements. The syntax of PAL is a variant of the Knowledge Interchange Format (KIF) [17]: It supports KIF connectives but not all of KIF constants and predicates (e.g., the theory of arithmetic is much smaller).

Variables begin either with '?' to indicate a local variable or with '%' to indicate a global variable. Variables can be a class, which define the main concepts of domain in ontology, or a slot, which presents some properties of classes or their relationships with other classes.

SQL triggers (*SQL-RULE* = $\{sql\text{-}rule_0, sql\text{-}rule_1, \dots, sql\text{-}rule_{n4}\}$) can be expressed as follows:

$$\begin{aligned} sql\text{-}rule_i =&< sql\text{-}comment_i, sql\text{-}trigger\text{-}name_i, sql\text{-}table_i, sql\text{-}event_i, \\ &IF\text{-}SQL'_i, THEN\text{-}SQL'_i >, \end{aligned} \tag{3}$$

where $sql\text{-}comment$ holds documentation of a rule, $sql\text{-}trigger\text{-}name$ holds the name of a rule, $sql\text{-}table$ holds the name of a table or a view to which the rule is attached, $sql\text{-}event$ holds an event (*INSERT*, *DELETE, UPDATE* or *SELECT*), which triggers the rule, $if\text{-}sql$ is a SQL statement, which holds the condition of the rule, $then\text{-}sql$ is a SQL statement, which holds the action of the rule. For more about SQL statements see Microsoft Visual Studio Documentation.

The following PAL constraints transformation in SQL rules is developed:

1. $A^P \Rightarrow SQL\text{-}RULE \Leftrightarrow \forall sql\text{-}rule_i \in SQL\text{-}RULE\ \exists a_i^P \in A^P \wedge (\forall a_i^P \in A^P\ \exists sql\text{-}rule_i \in SQL\text{-}RULE)$, where *SQL-RULE* is a set of formal rules in a form of SQL triggers.
   a. $\forall sql\text{-}comment_i \in sql\text{-}rule_i\ \exists PAL\text{-}documentation_i \in A_i^P \wedge (\forall PAL\text{-}documentation_i \in A_i^P \exists sql\text{-}comment_i \in sql\text{-}rule_i)$
   b. $\forall sql\text{-}trigger\text{-}name_i \in sql\text{-}rule_i\ \exists PAL\text{-}name_i \in A_i^P \wedge (\forall PAL\text{-}name_i \in A_i^P \exists sql\text{-}trigger\text{-}name_i \in sql\text{-}rule_i)$
   c. $\forall sql\text{-}table_i \in sql\text{-}rule_i\ \exists PAL\text{-}range_i \in A_i^P \wedge \neg(\forall PAL\text{-}range_i \in A_i^P \exists sql\text{-}table_i \in sql\text{-}rule_i)$
   d. $\forall if\text{-}sql_i \in sql\text{-}rule_i\ \exists PAL\text{-}statement_i \in A_i^P \wedge (\forall PAL\text{-}statement_i \in A_i^P \exists if\text{-}sql_i \in sql\text{-}rule_i)$
   e. $then\text{-}sql_i = \{$*commit transaction*, *rollback transaction*$\}$ with $\forall then\text{-}sql_i \in sql\text{-}rule_i$

4. Correctness of PAL constraints transformation in SQL rules

In this section authors check the correctness of PAL constraints transformation in SQL rules according to related work and background.

The goal of checking a rule is to ensure that there are no correctness errors in the source and no correctness errors in the target.

As we can state, from the related work presented in this paper, correctness is of three forms – syntactic, semantic and functional behaviour.

Syntactic correctness of transformation of PAL constraints in SQL rules.

Checking the PAL constraint is rather straightforward as it involves simply the decision of whether the left side conforms to the PAL syntax. The syntax correctness of PAL constraints was checked by executing those constraints. Since there was no syntactical errors during the execution, we can state, that our PAL constraints are syntactically correct. Since KIF syntax is strictly formal (see [17], PAL is also well-formed.

Checking SQL rules requires the checking if rules match SQL triggers. Since we use the structure of SQL triggers fore the transformation, we can state that the structure and syntax of SQL rules is correct and is the same as syntax and structure of SQL triggers.

All variables used in SQL rules must also be used in PAL constraints. If a variable is used in the SQL rule, but not declared, then a variable mismatch occurs. Such variable mismatches can easily be found by comparing the set of variables of SQL rules and PAL constraints.

The variable correctness criteria can be easily verified by a simple algorithm that computes the set of variables of the left and right-hand sides.

Semantic correctness of transformation of PAL constraints in SQL rules.

As stated in the section of related work and background, we should prove the semantic equivalence of PAL constraints and SQL rules. From the developed formulas 1, 2 and 3 we can state, that PAL constraints and SQL rules are semantically equivalent, since both includes *name*, *documentation* (*comment*), *table* (*range*), *if* and *then* parts with the same meaning.

Functional behaviour of transformation of PAL constraints in SQL rules. As stated in the section of related work and background, functional behaviour includes local confluence and termination of transformation, e.g., the existence of a unique result of the transformation for every valid input.

In our case transformation defines a function from PAL constraints to SQL rules, since variables used in PAL constraints are used in SQL rules. Transformation changes only the representation of those variables, the meaning leaves the same. Moreover, the source language is restricted by the syntax grammar. For each valid input (PAL constraint) a unique result (SQL rule) is produced (the results of transformation of PAL constraints in SQL rules are presented in [2].

5. Conclusions and future works

The analysis of the related works on model transformation shows that correctness of transformation is one of the main topics should be analysed in model transformation. It includes syntactic and semantic correctness and functional behaviour.

A transformation of PAL constraints in SQL rules is defined in formal way to check the correctness of proposed transformation. A process of checking the correctness of PAL constraints transformation in SQL rules shows that this transformation is possible and correct. Still this checking should be extended for all ontology and strictly formalised.

References

1. J. Miller, J. Mukerji (Eds.), *MDA Guide Version 1.0.1*, OMG (2003).
2. D. Bugaite, O. Vasilecas, Ontology-based information systems development: the problem of automation of information processing rules, in: *Proc. ADVIS'2006*, E. Neuhold, T. Yakhno (Eds.), Springer, LNCS, **4243** (2006), pp. 187–196.
3. N. Guarino, Formal ontology and information systems, in: *Proc. of FOIS'98*, Trento, Italy, IOS Press (1998), pp. 3–15.
4. H. Ehrig, K. Ehrig, U. Prange, G. Taentzer, *Fundamentals of Algebraic Graph Transformation*, Springer (2006).
5. D. Varro, A. Pataricza, Automated formal verification of model transformations, in: *Proc. of Critical Systems Development in UML (CSDUML 2003) of the UML'03 Workshop*, J. Jurjens *et al.* (Eds.), TUM-I0323 (2003), pp. 63–78.
6. M. Feilkas, How to represent models, languages and transformations?, in: *Proc. of the 6th OOPSLA Workshop on Doamin-Specific Modelling (DSM'06)*, J. Gray *et al.* (Eds.), Jyväskylä University Printing House (2006), pp. 169–176.
7. D. Varro, G. Varro, A. Pataricza, Designing the automatic transformation of visual languages, *Science of Computer Programming*, **44**(2), 205–227 (2002).
8. J.H. Hausmann, R. Heckel, S. Sauer, Extended model relations with graphical consistency conditions, in: *UML 2002 Workshop on Consistency Problems in UML-based Software Development* (2002), pp. 61–74.
9. R. Heckel, J.M. Kuster, G. Taentzer, Confluence of typed attributed graph transformation systems, in: *Proc. of the First International Conference on Graph Transformation*, A. Corradini *et al.* (Eds.), Springer, *LNCS*, **2505**, 161–176 (2002).
10. Wikipedia, Correctness. Wikipedia. The Free Encyclopedia (2007), (June, 2007) `http://en.wikipedia.org/wiki/Correctness`.
11. G.W.M. Kuntz, *Symbolic Semantics and Verification of Stochastic Process Algebras*, Dissertation (2006), (June, 2007), `http://deposit.ddb.de/cgi-bin/dokserv?idn=97894139x`.

12. J.M. Kuster, Systematic validation of model transformations, in: *Proc. of 3rd UML Workshop in Software Model Engineering (WiSME 2004)*, M. Gogolla, P. Sammut, J. Whittle (Eds.), Lisbon, Portugal (2004), (June, 2007), `http://www.metamodel.com/wisme-2004/accept/4.pdf.`
13. H. Ehrig, K. Ehrig, Overview of Formal Concepts for Model Transformations based on Typed Attributed Graph Transformation, in: *Proc. of the First International Workshop on Graph and Model Transformation (GraMoT)*, G. Karsai, G. Taentzer (Eds.), Elsevier Science B.V., Berlin, Germany (2005), (March, 2007) `http://tfs.cs.tu-berlin.de/gramot/Gramot2005/FinalVersions/PDF/EhrigEhrig.pdf`
14. J.M. Kuster, Definition and validation of model transformations, *Software and Systems Modeling (SoSyM)*, **5**(3), Springer, 233–259 (2006).
15. W. Grosso, The Protégé Axiom Language and Toolset (”PAL”) (2002), (September, 2005) `http://protege.stanford.edu/plugins/paltabs/pal-documentation/`
16. J. Hou, EZPal Tab. Stanford University (2005) (March, 2006) `http://protege.stanford.edu/plugins/ezpal/`
17. M.R. Genesereth, R.E. Fikes, Knowledge Interchange Format. Version 3.0. (1992), (June, 2007) `http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps`

REZIUMĖ

S. Auhor. Transformacijos teisingumas: ontologijos aksiomų transformacija į formalias taisykles

Straipsnyje yra nagrinėjamas transformacijų teisingumas. Todėl susijusių darbų apžvalgoje autoriai analizuoja teisingumo rūšis ir kaip jos yra tikrinamos. Detaliam nagrinėjimui yra pasirenkama PAL ribojimų transformacija į SQL taisykles (PAL ribojimų transformacija į SQL taisykles yra formaliai aprašoma). Ir galiausiai, yra nagrinėjamas pasiūlytos transformacijos teisingumas.