Electronic Communications of the EASST
Volume 42 (2011)

Proceedings of the 4th International Workshop on Multi-Paradigm Modeling (MPM 2010)

Rule-Based Integration of Domain-Specific Modelling Languages

Benjamin Braatz, Christoph Brandt

12 pages

Guest Editors: Vasco Amaral, Hans Vangheluwe, Cécile Hardebolle, Lazlo Lengyel
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Rule-Based Integration of Domain-Specific Modelling Languages

Benjamin Braatz*, **Christoph Brandt**

SECAN-Lab, Université du Luxembourg, http://wiki.uni.lu/secan-lab/
benjamin.braatz@uni.lu, christoph.brandt@uni.lu

Abstract: Domain-specific modelling languages (DSMLs) can increase the acceptance of (semi-)formal modelling techniques. They allow all stakeholders in an application domain to participate in the modelling process using notations that are close to their understanding of the domain. When several groups of stakeholders are concerned with a certain aspect of the modelled system, the question arises how different DSMLs can be integrated with respect to this aspect. In this paper, we propose rule-based transformations as a means to integrate heterogeneous DSMLs overlapping on dedicated aspects. We illustrate the approach by a running example of a small visual DSML for IT landscapes and a textual DSML for firewall configurations.

Keywords: Domain-specific modelling language, language integration, Resource Description Framework, algebraic graph transformation

1 Introduction

Our motivation for the work presented in this paper is to obtain a framework for the definition and management of families of domain-specific modelling languages (DSMLs). We use the term DSMLs to denote small, flexible, visual and textual languages that are tailored to the needs of their users in a certain application domain. Among other features, this framework should allow families of integrated DSMLs, which means that several DSMLs—each created for a specific task or a specific group of users—are integrated on their common overlapping aspects.

In Section 2, we present a running example to illustrate this requirement. This running example comprises a visual DSML for IT landscapes and a textual DSML for firewall configurations.

We use the Resource Decription Framework (RDF), defined in [KC04], to represent the abstract syntax of DSMLs. This representation is introduced in Section 3. Since RDF is used as the fundamental data structure for the Semantic Web, it is well-suited for the distributed management of models in large organisations. With this choice, we expect to reduce the effort that is required for the creation of large integrated models from the knowledge of local users, while also allowing decentralised workflows.

In Section 4, we introduce algebraic graph transformations on RDF graphs. Algebraic graph transformations for RDF were proposed and developed in [BB08, Bra09]. They are used in our

* This author is supported by the National Research Fund, Luxembourg, and cofunded under the Marie Curie Actions of the European Commission (FP7-COFUND).

approach, an overview of which is given in [BB10a], to provide a single, uniform formalism for all kinds of modifications on models—editing, migration and integration.

In Section 5, we present the integration of the languages from the running example. The different tasks during the integration of two models are facilitated by dedicated transformation rules for the manual, automatic and semi-automatic integration steps.

Finally, in Section 6 we discuss related work and give some concluding remarks in Section 7.

2 Running Example

In this section, we present a running example for the integration of a graphical and a textual language. The graphical language is a DSML for IT infrastructures. The language is rather small and simple, but a similar language for the real world would in principle work on the same level of abstraction, tailored to the needs of the users and resembling the informal languages modellers use today. The textual language is a language for the configuration of firewalls, where we restrict ourselves to the specification of which ports are allowed in which direction.

In Figure 1, we show an example of the concrete syntax of these two languages and their integration. The landscape model on the right side shows some local area networks (LANs), the Internet as a wide area network (WAN) and the connections between them, where most of them are protected by firewalls. For the firewalls, we also specify which protocols are allowed in which direction. The local network LAN 1 is connected to the Internet through a demilitarised zone (DMZ). HTTP queries are allowed from the Internet to the DMZ and vice versa and from LAN 1 through a proxy in the DMZ to the Internet. Moreover, LAN 1 can reach a backup network via SSH. The firewall configurations on the left give a textual representation of the allowed protocols.

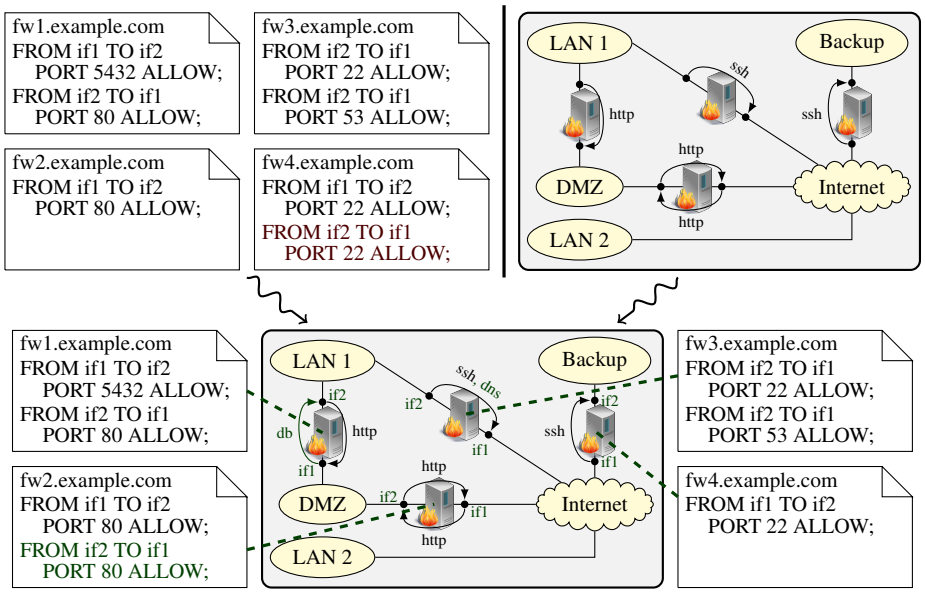

Figure 1: IT landscape and firewall configurations—language integration

During the integration, we first have to identify which textual firewall configuration corresponds to which firewall in the landscape diagram and which interface name to which connection. Then, we can search for inconsistencies between the diagram and the configurations. In our example, we find that the allowance of HTTP traffic from the DMZ to the Internet has to be added in the configuration of Firewall fw2 and the database access from the DMZ to LAN 1 and the DNS access from LAN 1 to the Internet were forgotten in the diagram. Moreover, the configuration of firewall fw4 allowed outbound SSH traffic from the backup network to the Internet which is not neccessary according to the diagram and, therefore, removed.

3 RDF Graphs: Abstract Syntax of DSMLs

The Resource Description Framework (RDF), defined in [KC04], provides the fundamental, generic data structure for the Semantic Web. It is used to state facts about resources that are identified by Uniform Resource Identifiers (URIs). The facts are given by subject–predicate–object triples, where the predicate is given by a URI and the object may also be a literal value. A set of facts is an RDF graph, where the subjects and objects are the nodes and the facts the edges of the graph, labelled with the corresponding predicate (which may also appear as a node). The idea is that everyone can publish such graphs to assert certain facts and these graphs can easily be joined to collect information from heterogeneous sources.

In Figure 2, we show how such a graph is used to represent part of an IT landscape model. The local net LAN 1 and the Internet are represented by URIs “mod:LAN1” and “mod:INet”, respectively, where “mod:” is a suitable namespace, e. g., “http://models.example.com/”. The names of the networks, which are shown as inscriptions in the concrete visual representation, are given by literal values in the abstract syntax. The predicate “rdf:type” (abbreviated as “a”) is used to connect nodes with their types. The types as well as the predicates are defined in another namespace “itml:”, which, e. g., might be “http://schema.example.com/”. Technically, it would be possible to use the same namespace for both, the language elements and the model instances, but separation of namespaces eases the distributed handling of schemas and (multiple) models by different groups of users. The connection between LAN 1 and the Internet is represented by a blank node “1”. Blank nodes do not have a global identity and, hence, other graphs cannot state additional facts about entities identified by blank nodes.

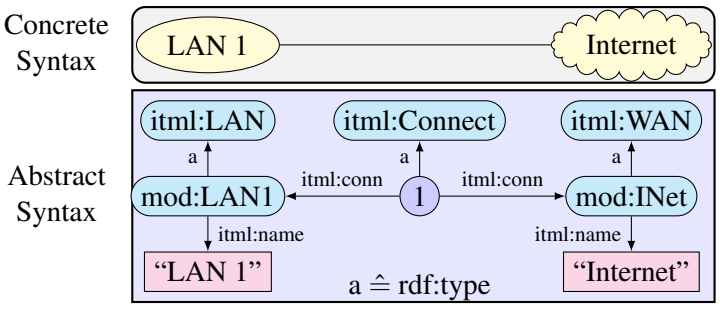

Figure 2: RDF graph representing the abstract syntax of a DSML model

In Figure 3, we show how the additional notation for protocols in the IT landscape language and the textual firewall configuration language are represented in RDF. In the landscape DSML, the arrows around the firewalls are represented by blank nodes with type “itml:Flow”, “itml:src” and “itml:trg” predicates indicating the direction of the arrow and “itml:prot” predicates specifying which protocols are allowed in this direction. The firewall configuration language uses blank nodes of type “fwcl:Rule” for each line in a firewall configuration, where the direction is given by “fwcl:from” and “fwcl:to” predicates and the corresponding port by a “fwcl:port” predicate.

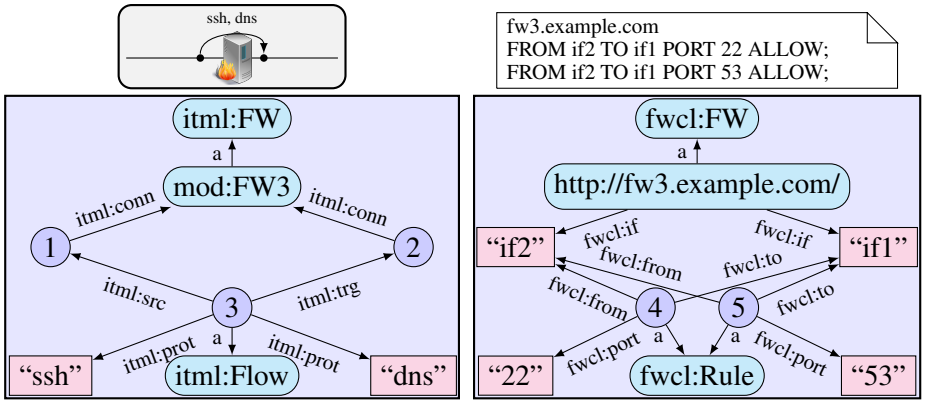

Figure 3: RDF representation of firewall configurations

We choose to base our framework on RDF graphs rather than classical graphs, typed, labelled or attributed graphs for several reasons. They are designed to work well in distributed environments due to the use of URIs for identifying nodes and predicates. With literal values as just another type of nodes they provide a lean formalisation for graphs with all kinds of attributations, where other approaches need much heavier formal machinery. Last but not least, we want to be able to interface with the rest of the Semantic Web in the long run.

Regarding the semantics of the used languages, we restrict ourselves to an implicit semantics in this paper. In [Usc03], the fact that not all applications using Semantic Web data structures necessarily also are “semantic” is discussed. And if they do, there are several levels of semantics—from informal up to machine-processable. As far as we know, there is not much work on formalising the kinds of semantics that are needed for heterogeneous modelling languages in the context of the Semantic Web–from operational semantics for behavioural techniques to denotational semantics for data type specifications. It is an interesting line of further research to develop ways to represent the semantics of DSMLs in a way that is compatible with the idea of the Semantic Web, i. e., interoperable over distributed, heterogeneous systems and ideally also machine-processable.

Compared to the definition of languages by meta modelling, e. g., using the Meta Object Facility (MOF), defined in [Obj06], the approach chosen here is very light-weight. RDF graphs are supposed to be defined in a distributed manner on the Semantic Web with the help of heterogeneous and extendable vocabularies. In contrast to that, MOF enforces that every model strictly conforms to a meta model, which, therefore, has to anticipate all needs of the users. Thus, the features of RDF ideally reflect the flexibility requirements of DSMLs.

In [Obj09], proposals for a mapping from MOF to RDF are requested. Such a mapping would allow the representation of MOF meta models and corresponding models as RDF graphs and, hence, facilitate the application of the methods of our approach to MOF meta models and models.

4 Algebraic Graph Transformations for RDF

We use rule-based, algebraic graph transformations to describe all kinds of changes on RDF graphs in a declarative way. A comprehensive overview over the theory of algebraic graph transformation can be found in [EEPT06]. The adaption of this theory to RDF was proposed in [BB08] and continued in [Bra09].

We choose algebraic graph transformation, since it allows to treat all kinds of transformations from language definition by grammars via model migration to language integration with a single, uniform formalism, which can be fully implemented. Being a formal technique, it also allows to reason formally about the effects of transformations to show, e. g., that certain derived transformation rules respect a given graph grammar or that transformations are independent of each other and can, hence, be swapped without affecting the final result of the complete transformation sequence.

Transformations are defined by transformation rules. An example of such a rule is shown in Figure 4. This rule removes a connection between two networks, represented by variables x and y and adds a new connection from network x to network z, where z is not allowed to be of type itml:WAN and it is not allowed to introduce a self connection, i. e., to assign x and z to the same net.

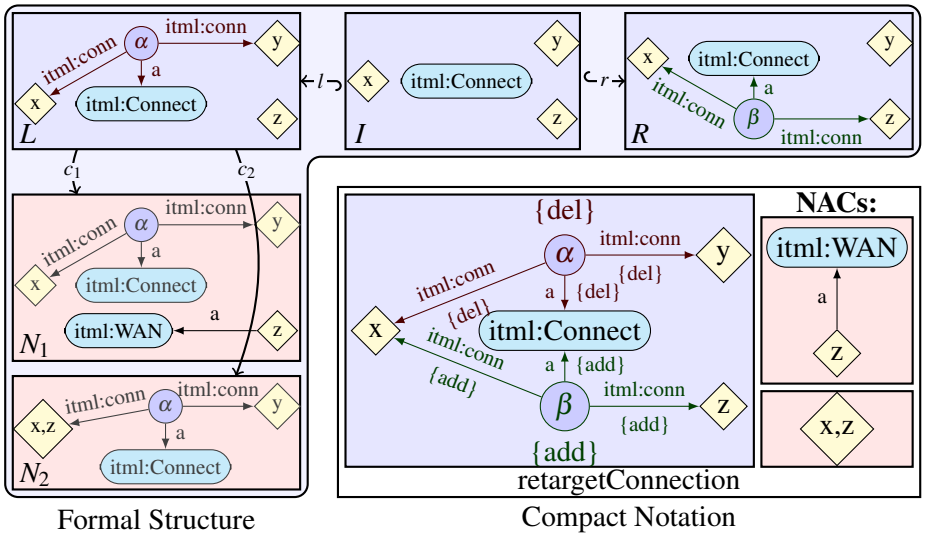

Figure 4: Example transformation rule retargetConnection

The rule consists of several RDF patterns, which are graphs with additional variables, and RDF pattern homomorphisms, which are structure preserving maps connecting the patterns, where the homomorphisms l and r are injective, i. e., one-to-one (visualised by the hook at the tail of the

arrows). The difference between the left-hand side L and the interface I are the elements of the pattern that are supposed to be deleted and the difference between I and the right-hand side R are the elements that are supposed to be added by the rule. Moreover, there is a set of negative application conditions (NACs), which are extensions of L and specify situations in which the rule is not applicable, where the NACs are in an implicit conjunction, i. e., all have to be satisfied and none of the situations is allowed. In the lower right, we show a compact notation for the rule, where irrelevant parts of the NACs are omitted and L, I and R are shown in a single diagram with the additional elements of L marked by "{del}" and the additional elements of R by "{add}".

In Figure 5, we show the application of the transformation rule from Figure 4 to an example graph. The application is determined by a match homomorphism m from the left-hand side L to the graph G. The application of the rule is then computed by two category theoretical constructions, namely a minimal pushout complement (MPOC) and a pushout (PO). (See [AHS09] for an introduction to category theory.) Intuitively, a pushout is a disjoint union over a common interface and used to add the additional blank nodes and facts of R to the context graph D. Conversely, a minimal pushout complement is used to derive the context graph D by deleting the additional blank nodes and facts of L from G.

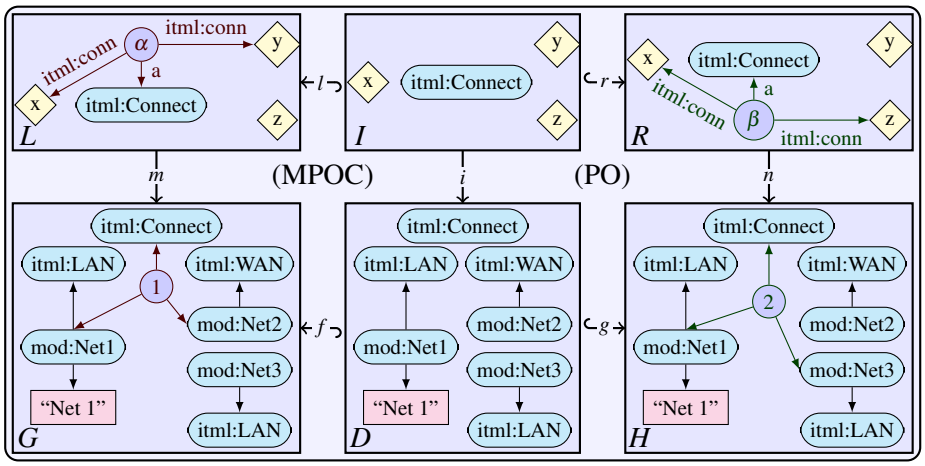

Figure 5: Application of transformation rule retargetConnection

The use of rule-based graph transformations for modifying RDF graphs has several advantages. An implementation of a transformation engine may be reused for a multitude of purposes, where the search for matches and the transformation only have to be implemented once and for all. Modifications are specified on an adequate level of abstraction and automatically respect a given grammar if they are composed from it, which is also the main advantage of using grammars for language definition instead of meta modelling. Moreover, a single, uniform formalism can be used for language definition, syntax-directed editing and complex modifications. In the following section, we show how algebraic graph transformation rules can be used to specify the integration of several DSMLs on overlapping aspects.

5 Integration of Languages

Since, the landscape language uses the names of protocols and the configuration language uses port numbers, we need a mapping between them, before we can start our integration effort. This mapping is given in Figure 6, where blank nodes with corresponding int:prot and int:port predicates are used to represent this relation.

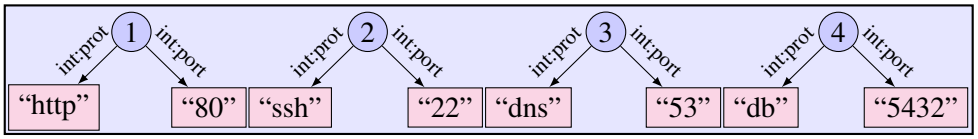

Figure 6: Mapping between protocols and ports

For integrating several DSMLs, we define some sets of RDF graph transformation rules.

First, the rules in Figure 7 are used to *manually* establish the connection between the firewalls in the landscape model and the corresponding configuration language snippets and the connections in the landscape and the corresponding interfaces in the configurations. This has to be done manually, since there is not enough information in the models to deduce these correspondences automatically. It is a constructive choice.

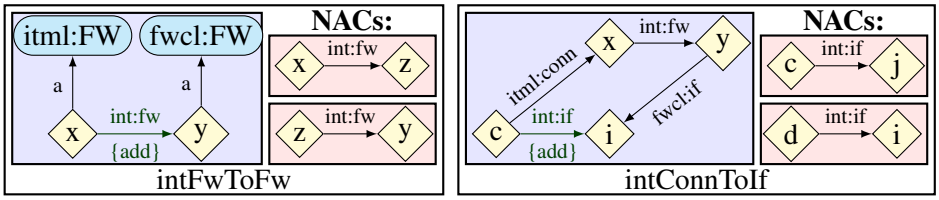

Figure 7: Manual integration rules

The rule in Figure 8 can be used to *automatically* add lines to the firewall configurations, where there is an additional protocol that is allowed according to the landscape model. This rule can be applied as long as possible, since we decide to consider the landscape model as superior in these situations.

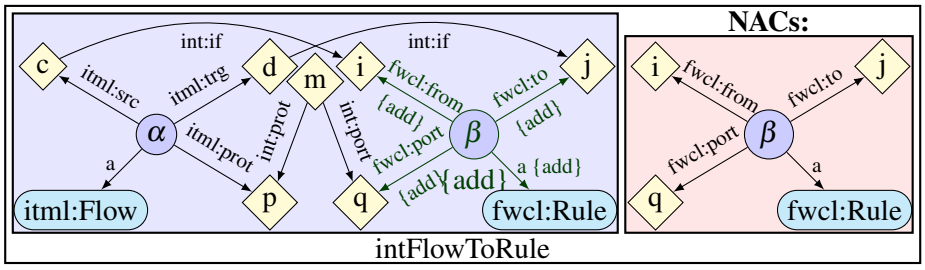

Figure 8: Automatic integration rule

If there is a line in the firewall configurations that has no corresponding protocol in the landscape model then the rules in Figure 9 are used. These rules are supposed to be applied *semi-automatically*. The matches can be found automatically, but in each case two of the rules are applicable and the user has to decide which should be applied. Either the protocol is added to the landscape model by one of the first two rules—intRuleToExFlow if there is already another protocol in the same direction, intRuleToNonexFlow if the arrow also needs to be created—or the line is deleted from the firewall configuration by the third rule intDelRule.

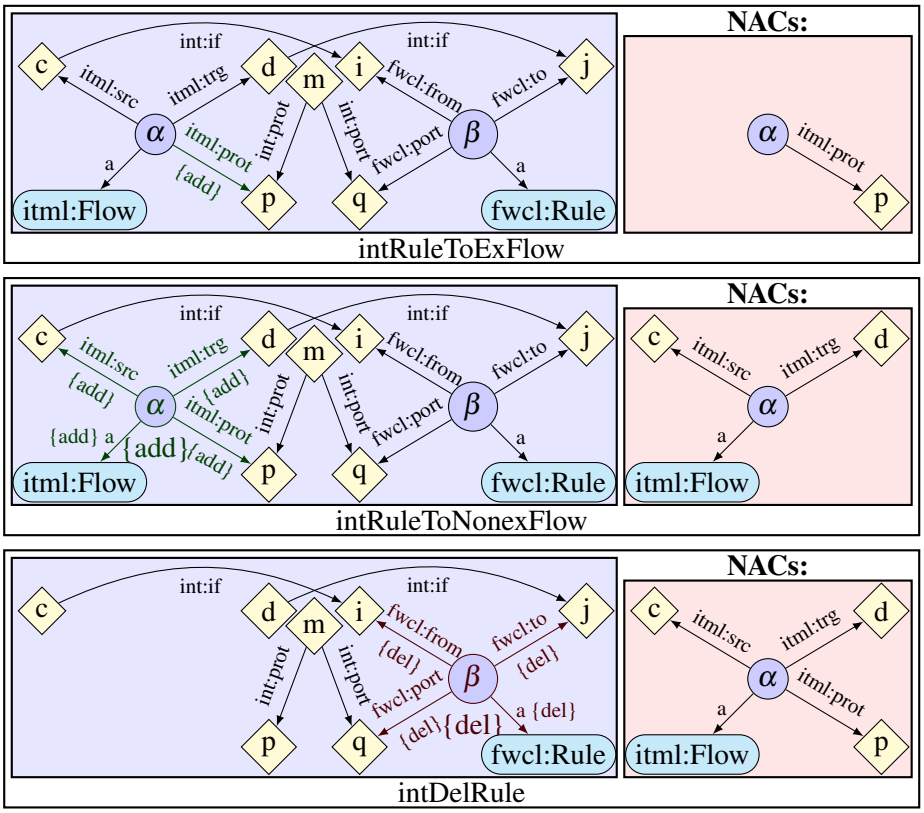

Figure 9: Semi-automatic integration rules

The distinction between manual, automatic and semi-automatic rules is not visible in the rules themselves, since they are all of the same kind as defined in Section 4. Rather, the distinction will be made in the metadata of the rules in a future implementation, where the difference lies in the treatment of the matches. For the manual rules, the user is required to provide the match for rule applications, where the implementation can guide her on the choice of matches that are legal w. r. t. the application conditions. For the automatic and semi-automatic rules, the implementation is supposed to apply them on all possible matches, where the difference is that there are several conflicting possibilities in the semi-automatic case, which have to be resolved by the user.

Table 1 shows a comparison of this rule-based model integration with the process of manually integrating models or documents. In any case, the search for inconsistencies is replaced by the

automatic match-finding for the integration rules. For the deterministic cases that lead to the automatic integration rules in Figure 8, the integration can be completely executed without user intervention, while for the non-deterministic cases that lead to the semi-automatic integration rules in Figure 9, the user still has to decide which of the two possibilities should be executed. When trying to manually integrate models, the much more error-prone task of constructing inconsistency eliminations is required, while these eliminations are given once and for all in the rule-based approach. Thus, the integration by rule-based graph transformation leads not only to less effort for the integration but also to quality gains by reducing missed inconsistencies and inappropriate eliminations.

Table 1: Comparison of rule-based and manual integration

	Rule-Based Integration	**Manual Integration**
Deterministic	1. Automatic Match-Finding	Find Inconsistencies
	2. Automatic Elimination by Rules	Eliminate Inconsistencies
Non-Deterministic	1. Automatic Match-Finding	Find Inconsistencies
	2. Alternatives Given by Rules	Construct Alternatives
	3. Decision on Alternatives	

This methodology can also be used to integrate models from other modelling approaches that have a translation to RDF. For example, once a MOF to RDF mapping, as requested in [Obj09] and already mentioned at the end of Section 3, is provided, models of MOF-based languages can be integrated with each other and with native languages in our framework. Moreover, RDF graph transformation rules can also be used to completely translate models between languages, which is, however, outside the scope of the present paper.

6 Related Work

In [AFR06], the authors evaluate several DSML tools, which were available at that time, w. r. t. a catalog of criteria. One of those criteria is the integration with other languages. The tools that fulfil this criterion achieve integration either by building on UML or by technological integration via the Eclipse platform. It remains unclear how gaps between similar but slightly different concepts and structures are bridged in these solutions. We believe that our approach of rule-based linking and modification of the models is the right answer to this problem.

In fact, most integrated modelling language families, e. g., the domain-specific ITML family, presented in [FHK+09], and the UML, specified in [Obj10], achieve language integration *a priori* by defining the common aspects on which the different languages overlap. In contrast to that, our approach allows an *a posteriori* integration of languages that were designed without knowledge of each other.

There are several approaches to language integration based on the Triple Graph Grammar (TGG) formalism. For example, TGGs are used on top of the meta modelling environment AToM^3 in [GL08], the MOSL language, presented in [AKR06], is the basis of the MOFLON meta modelling framework and uses TGGs in connection with MOF, OCL and story diagrams,

and language integration with TGGs is theoretically examined in [EEH08]. In TGGs, the connections between two languages are described by three graphs, one for each of the languages and a third one describing the connection by morphisms into the other two. This structure is simultaneously built up by a TGG, where translation rules in both directions and integration rules can be derived from the TGG. While our approach requires to design integration and possibly also translation rules manually, it also adds the flexibility to decide for the concrete case which is the adequate way to integrate a certain situation. For example, for the language integration of the present paper, it is a design decision that additional elements in the network diagram are transferred to the firewall configurations automatically, while additional rules in the firewall configurations require a user decision if they are to be added to the diagram or deleted from the configuration. Such an effect would be cumbersome to achieve with TGGs if it is at all possible. Moreover, the treatment of all models in the same graph structure—a giant global graph (cf. [BL07])—seems to be more appropriate for the Semantic Web and relieves us from managing multiple graphs.

In the context of the Semantic Web and knowledge management, the problem of mapping different ontologies is related to our present work. See [CSH06] for a survey on different approaches to ontology mapping. Our approach to integration is complimentary to the problem of ontology mapping, alignment and integration. While we take the relations between the heterogeneous DSMLs as given by a user of our framework and provide means for integrating instances of the DSMLs, ontology mapping approaches try to find similarities between concepts in heterogeneous ontologies (semi-)automatically, but usually do not provide means for complex structural changes in the instance integration process as they are provided by our approach.

7 Summary and Future Work

In this paper, we have shown how graph transformation rules can be used to integrate domain-specific models that overlap on certain aspects. We have used RDF graphs for the abstract syntax of domain-specific models. Algebraic graph transformation provides a single formalism that can be used not only for the manual, automatic and semi-automatic integration tasks shown in this paper but also for specifying domain-specific modelling languages by graph grammars and for providing complex modifications like refactorings.

The main contribution of this paper is to show how our combination of RDF and algebraic graph transformation can be used for DSMLs and especially to integrate heterogeneous DSML models. The main benefits are the relatively lean abstract syntax graphs in comparison to, e. g., MOF, the possibility to treat heterogeneous models in a single, distributed graph structure, which is achieved by using RDF, and the use of the single formalism of algebraic graph transformation for all kinds of tasks from language definition to integration. The limits are the need to bridge the technological gap if models from other approaches need to be imported or accessed and, up to now, the lack of an implementation of this framework.

Currently, we are implementing this graph transformation on top of an RDF triple store. This leads to a transformation engine, which can be used as a model repository for families of domain-specific modelling languages according to the concepts presented in this paper. A sketch of the architecture and protocol of this transformation engine can be found in [BB10b].

References

[AFR06] D. Amyot, H. Farah, J.-F. Roy. Evaluation of Development Tools for Domain-Specific Modeling Languages. In Gotzhein and Reed (eds.), *Proc. SAM 2006*. LNCS 4320, pp. 183–197. Springer, 2006.
doi:10.1007/11951148_12

[AHS09] J. Adámek, H. Herrlich, G. E. Strecker. *Abstract and Concrete Categories: The Joy of Cats*. Dover, 2009.
http://katmat.math.uni-bremen.de/acc/

[AKR06] C. Amelunxen, A. Königs, T. Rotschke. MOSL: Composing a Visual Language for a Metamodeling Framework. In *Proc. VL/HCC 2006*. Pp. 81–84. IEEE, 2006.
doi:10.1109/VLHCC.2006.33

[BB08] B. Braatz, C. Brandt. Graph Transformations for the Resource Description Framework. In Ermel et al. (eds.), *Proc. GT-VMT 2008*. Electronic Communications of the EASST 10. 2008.
http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/158

[BB10a] B. Braatz, C. Brandt. Domain-Specific Modelling Languages with Algebraic Graph Transformations on RDF. In *Proc. SLE 2010*. 2010.
http://planet-sl.org/sle-conference/index.php?option=com_content&task=view&id=162&Itemid=228

[BB10b] B. Braatz, C. Brandt. How to Modify on the Semantic Web? A Web Application Architecture for Algebraic Graph Transformations on RDF. In *Proc. SWIM 2010*. 2010.
http://mais.dia.uniroma3.it/SWIM2010/Accepted_Papers.html

[BL07] T. Berners-Lee. Giant Global Graph. Blog Post, Nov. 2007.
http://dig.csail.mit.edu/breadcrumbs/node/215

[Bra09] B. Braatz. *Formal Modelling and Application of Graph Transformations in the Resource Description Framework*. PhD thesis, Technische Universität Berlin, 2009.

[CSH06] N. Choi, I.-Y. Song, H. Han. A Survey on Ontology Mapping. *ACM SIGMOD Record* 35(3):34–41, Sept. 2006.
doi:10.1145/1168092.1168097

[EEH08] H. Ehrig, K. Ehrig, F. Hermann. From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. Forschungsbericht 2008-03, Fakultät IV, Technische Universität Berlin, 2008.
http://www.eecs.tu-berlin.de/fileadmin/f4/TechReports/2008/2008-03.pdf

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. Springer, 2006.
doi:10.1007/3-540-31188-2

[FHK+09] U. Frank, D. Heise, H. Kattenstroth, D. F. Ferguson, E. Hadar, M. G. Waschke. ITML: A Domain-Specific Modeling Language for Supporting Business Driven IT Management. In Rossi et al. (eds.), *Proc. DSM 2009*. HSE B-108. 2009.
http://www.dsmforum.org/events/DSM09/Papers/Heise.pdf

[GL08] E. Guerra, J. de Lara. Meta-Modelling and Graph Transformation for the Definition of Multi-View Visual Languages. In Ferri (ed.), *Visual Languages for Interactive Computing: Definitions and Formalizations*. Chapter IV, pp. 74–101. Information Science Reference, 2008.
http://astreo.ii.uam.es/~jlara/MultipleViews.pdf

[KC04] G. Klyne, J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium (W3C), Feb. 2004.
http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[Obj06] Object Management Group (OMG). Meta Object Facility (MOF) Core Specification. Jan. 2006.
http://www.omg.org/spec/MOF/2.0/

[Obj09] Object Management Group (OMG). Request for Proposal. MOF to RDF Structural Mapping in support of Linked Open Data. Dec. 2009.
http://www.omg.org/cgi-bin/doc?ad/2009-12-09

[Obj10] Object Management Group (OMG). OMG Unified Modeling Language (OMG UML). May 2010.
http://www.omg.org/spec/UML/2.3/

[Usc03] M. Uschold. Where are the Semantics in the Semantic Web? *AI Magazine* 24(3):25–36, Fall 2003.
http://www.aaai.org/ojs/index.php/aimagazine/article/view/1716