Science of Computer Programming 126 (2016) 3–30

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Heterogeneous verification in the context of model driven engineering

Daniel Calegari [a,*], Till Mossakowski [b], Nora Szasz [c]

[a] Facultad de Ingeniería, Universidad de la República, 11300 Montevideo, Uruguay
[b] Otto-von-Guericke University Magdeburg, 39106 Magdeburg, Germany
[c] Facultad de Ingeniería, Universidad ORT Uruguay, 11100 Montevideo, Uruguay

ARTICLE INFO

Article history:
Received 1 May 2015
Received in revised form 17 February 2016
Accepted 19 February 2016
Available online 26 February 2016

Keywords:
Verification
Formal methods
Model-Driven Engineering
Theory of Institutions
Heterogeneous Tool Set

ABSTRACT

In some cases it may be useful to represent a problem in many logical domains, since they provide different perspectives for addressing formal verification. However, the maintenance of multiple representations in separate domains can be expensive if there is neither automated assistance nor a clear formal relation between these domains. We have addressed this problem in the context of Model-Driven Engineering (MDE). We defined solid foundations of a theoretical environment for formal verification using heterogeneous verification approaches. The environment is based on the Theory of Institutions which provides a sound basis for representing MDE elements and a way for specifying translations from these elements to other domains used for verification. In this paper we present how this environment can be supported in practice within the Heterogeneous Tool Set (Hets). Hets supports heterogeneous specifications and provides capabilities for monitoring the overall correctness of a heterogeneous proof. We first extend the theoretical environment with the inclusion of an institution for the Object Constraint Language (OCL), and then we define semantic-preserving translations from the OCL-constrained MDE elements to a core language of Hets. With this we can verify basic properties of our specification, and then use the existent connections between logical domains within Hets for broadening the spectrum of domains in which complementary verification properties can be addressed.

1. Introduction

Software development costs may be reduced by adopting a model-centric approach in which different views of the system to be constructed are provided by models. Models are abstractions of the system to be built (or some aspects of it) and also allow us to deal with its intrinsic complexity in a simplified manner. Moreover, the use of automated mechanisms (model transformations) in which models are transformed from higher abstraction levels until an executable system is built, may improve efficiency on the whole process. The Model-Driven Engineering (MDE, [1]) paradigm is based on these practices, also encompassing other engineering efforts, such as maintenance and reverse engineering.

Every model (from now on SW-model to avoid conflicts) *conforms* to a metamodel which introduces the syntax and semantics of certain kind of SW-models. Sometimes if there are conditions that cannot be captured by the structural rules of this language, another constraints language must be used to specify them. These considerations imply defining conformance

* Corresponding author. Tel.: +598 27114244x1125; fax: +598 27110469.
E-mail addresses: dcalegar@fing.edu.uy (D. Calegari), mossakow@iws.cs.uni-magdeburg.de (T. Mossakowski), szasz@ort.edu.uy (N. Szasz).

http://dx.doi.org/10.1016/j.scico.2016.02.003
0167-6423/

in terms of *structural* and *semantical* (or non-structural) conformance [2]. An SW-model is structurally conformant with respect to a metamodel if it is well-typed with respect to the metamodel and it also satisfies the multiplicity constraints. Semantical conformance requires, in addition to structural conformance, that the SW-model satisfies the invariants specified using the supplementary constraints language. A model transformation (or just transformation from now on) basically takes as input an SW-model conforming to certain metamodel and produces as output another SW-model conforming to its metamodel (possibly the same one).

Formal verification is often aided by a separation of duties between software developers. In general terms, MDE experts define models and transformations, while formal experts conduct the verification process, often aided by some (semi)automatic generation process which translates the MDE elements to their formal representation into the domain used for verification purposes. Nevertheless, there are multiple properties that can be verified, as well as a plethora of verification approaches with different objectives, formalisms and supporting tools [3]. Moreover, in some cases it may be useful to have a representation of MDE elements in different logical domains.

In [4,5] we proposed a theoretical environment for the formal verification of different MDE aspects using heterogeneous verification approaches [6]. The environment, based on the theory of Institutions [7], proposes a generic representation of the syntax and semantics of the MDE elements by means of institutions. Although the environment can be potentially formalized for any transformation approach and language, our proposal is aligned with the MetaObject Facility (MOF, [8]), i.e. a standard language for metamodeling, and the Query/View/Transformation Relations (QVT-Relations, [9]), i.e. a relational language which defines transformation rules as mathematical relations between source and target elements. We defined institutions CSMOF for the structural conformance relation between SW-models and MOF-based metamodels, and QVTR for QVT-Relations model transformation. The theory allows the definition of semantic-preserving translations (co-morphisms) between semantic domains defined as institutions and re-use their entailment systems for developing proofs. The definition of comorphisms not only provides a way to choose the formalism in which formal experts are more skilled to address a formal proof, but also allows supplementing the former specification of MDE elements with additional properties using the target logic. To the extent that there are many logics connected through comorphisms, the capabilities of the environment increase.

The aim of this paper is to present how the environment can be supported in practice using the Heterogenous Tool Set (HETS, [6,10]), which is meant to support heterogeneous multi-logic specifications. It also provides proof management capabilities for monitoring the overall correctness of a heterogeneous specification whereas different parts of it are verified using (possibly) different semantic domains.

Semantical conformance is not addressed by the original CSMOF institution. Thus, as a proof of concepts of the whole environment, we first define an institution for a small subset of the Object Constraint Language (OCL, [11]), which is used as a constraint navigational expression language with MOF and QVT-Relations languages. The OCL and CSMOF institutions are intended for metamodeling only and not for the definition of software systems. To do the later, pre and post condition constraints must be introduced in the OCL institution as well as the concept of methods in CSMOF. We then define how MDE elements can be integrated into HETS by defining semantic-preserving translations to the Common Algebraic Specification Language (CASL, [12]), which is a core language of HETS. The existent connections between CASL and other formalisms broadens the spectrum of formal domains in which verification can be addressed. We also detail the implementation of a prototype which allows us to specify MDE elements, supplement them with multi-logic properties, and perform a heterogeneous verification.

This paper is a substantially extended and thoroughly revised version of [13], which is a continuation of the formal foundations introduced in [5]. Additional material includes:

- the definition of an institution for a subset of OCL to be used within our environment;
- a detailed formalization of the semantic-preserving translations to CASL, together with a running example;
- a deeper discussion about the scope of verification within the environment and of related work.

The remainder of the paper is structured as follows. In Section 2 we present the basic notions of the Theory of Institutions which will be useful for the theoretical understanding of the following sections. For a more detailed introduction to the topic refer to [7,14]. In Section 3 we briefly introduce the institution CSMOF for the structural conformance relation between SW-models and MOF-based metamodels, as defined in [5], and, based on it, we define an institution for OCL to be used within our environment. In Section 4 we briefly introduce the institution QVTR for QVT-Relations model transformation, as defined in [5] together with the use of the OCL institution. In Section 5 and Section 6 we present the semantic-preserving translation from the MDE elements into CASL. In Section 7 we give details about the implementation of our environment within HETS and discuss the capabilities of our environment. Finally, in Section 8 we discuss related work and in Section 9 we present some conclusions and future work.

2. Institutions and their (co)morphisms

The notion of *Institution* [7] relies on Category Theory [15] for formalizing the notion of "logical system". It consists of vocabularies (called signatures) for constructing sentences in a logical system. A model (also called interpretation) provides semantics by assigning interpretations to the elements in the signature. A change of interpretation is done by the notion of