Technical Report No. NJU-SEG- 2008-IC-001

An MDE Based Approach for Bridging Formal Models

Tian Zhang, Frederic Jouault, Jean Bezivin, Jianhua Zhao


Most of the papers available from this document appear in print, and the corresponding copyright is held by the publisher. While the papers can be used for personal use, redistribution or reprinting for commercial purposes is prohibited.
A MDE Based Approach for Bridging Formal Models

Tian Zhang†§‡, Frédéric Jouault‡, Jean Bézivin‡, Jianhua Zhao†§

†State Key Laboratory for Novel Software Technology
‡Department of Computer Science & Technology
Nanjing University, 210093 Nanjing, China
ztluck@seg.nju.edu.cn, zhaojh@nju.edu.cn
‡ATLAS team, INRIA & LINA
Nantes University, 44300 Nantes, France
{frederic.jouault, Jean.Bezivin}@univ-nantes.fr

Abstract

Different formal methods have presented plenty of formal models for system specification and proof. Hence the problem of bridging these formal models rises. MDE is a new paradigm in software engineering, which implements software by (meta-)modeling and model transforming. In this paper, we provide a MDE based approach for bridging heterogeneous formal models: Firstly, the heterogeneous formal models are introduced into MDE as domain specific languages by metamodeling. Then, transformation rules are built for semantics mapping. At last, model-text syntax rules are developed, so as to map models to programs. Our approach could be applied on formal models in both graphical style and grammatical style. A case study of bridging MARTE to LOTOS is also illustrated showing the validity and practicability of our approach.

1. Introduction

In computer science and software engineering, formal methods are mathematically-based techniques for the specification, development and verification of software and hardware systems. Although the use of mathematical logic is a unifying theme across the discipline of formal methods, there is no single best “formal method”. Different application domains are best served by different formal methods and models. Hence the problem of bridging different formal models rises.

MDE (Model-driven engineering) [1] is a new paradigm in software engineering, which implements software by (meta-)modeling and model transforming. The AMMA platform is one of representative open source tools supporting MDE.

In this paper, we provide a MDE based approach for bridging the heterogeneous formal models. The different formal models are viewed as the different DSLs (Domain Specification Languages) [2] so as to be introduced into MDE by metamodeling. Then, transformation rules are presented to bridge the heterogeneous formal models in the MDE context. At last, the syntactic transformation rules are developed which map the formal models from the Modelware domain to the programs in the Grammarware domain. Therefore, the bridges between the heterogeneous formal models are constructed.

Firstly, the formal models are introduced into the AMMA platform by KM3 (Kernel Metametamodel) [3] metamodeling. Secondly, the ATL (ATLAS Transformation Language) [4] transformation rules are developed based on the built metamodels of the formal models. These rules define the semantical mapping between the formal models. Through the execution of the ATL rules on the ATL virtual machine, the instantial models consistent with the source metamodel are transformed into those consistent with the target metamodel. At last, the model-to-text rules are developed using TCS (Textual Concrete Syntax) [5], so that the generated models could be mapped to the textual programs.

A case study of bridging MARTE (Modeling and Analysis of Real Time and Embedded systems) [6] to timed-LOTOS [8] is presented showing the validity and practicability of our approach.

This paper is organized as follows. Section 2 introduces the AMMA platform. Section 3 explains the MDE based approach of bridging formal models. Section 4 and 5 present the related work and draw the conclusion.
2. The AMMA platform

Our work is mainly based on KM3, ATL and TCS, hence we focus on them to give the introduction of the AMMA platform.

The main purpose of KM3 is to define a root for metamodel hierarchy of the AMMA platform, so that all languages, including ATL, TCS, etc, on it could be defined by KM3.

Another important purpose of KM3 is to define the metamodels for DSLs, which is so-called metamodeling. In this paper, the method of introducing formal models into MDE is based on KM3 metamodeling.

ATL is the core of the AMMA platform, on which the other parts are implemented. Being designed as an answer to the QVT RFP, ATL is a hybrid of declarative and imperative constructs.

3. MDE based approach for bridging formal models

Though the MDE based solution of bridging heterogeneous formal models is very different from the other methods, it faces the same sub-problems: semantical mapping and syntactical transformation. This section explains, at the level of semantics and syntax respectively, that how to construct the bridge based on the AMMA platform.

3.1. Semantical mapping

From the viewpoint of MDE, the different formal models are the different DSLs. Hence the semantical mapping relations could be defined as the model transformation rules on the corresponding metamodel.

The metamodel of formal model is built using KM3 metamodeling which is the base for the bridge construction. The steps of building the metamodel could be described as: firstly, the formal model is analyzed so as to abstract its constructs as well as the corresponding structure; secondly, the metamodel is defined using the KM3 notations.

Basing on the metamodels of the formal models, the ATL transformation rules could be constructed, which define the semantical mapping from the source formal model to the target one. Every single ATL rule supports the unidirectional mapping. Therefore, the bidirectional mapping could be obtained by constructing ATL rules on the both directions. The semantical mapping through ATL could be illustrated in Figure 2.

3.2. Syntactical transformation

Presently, the notation of formal models is mainly of graphical and textual styles. For instance, UML, Petri Net, MSC are expressed in the graphical style, and CCS/CSP, LOTOS are expressed in the textual style.

To transform graphical notations, which is supposed to be the transformation within Modelware, we use the UML, the defacto industrial standard, as the intermedia format. Using the APIs provided by the AMMA platform, the models could be serialized into XMI format. So the graphical models could cooperate with each other. This method is
broadly used in industry, hence we just reuse it in our bridge construction directly.

To transform notations between Modelware and Grammarware is placed much emphasis, as it plays the dominant part in the syntactical transformation of our approach. In fact, it is the base for transforming textual notations in Grammarware. Figure 3 illustrates the TCS based syntactical transformation between Modelware and Grammarware.

**Figure 3. TCS syntactical transformation.**

Assume that there are two formal models $DSL_1$ and $DSL_2$ in Figure 3 to be syntactically transformed. As for $DSL_1$, $Injector1$ could transform its programs to models, and $Extractor1$ could transform its models to programs. It is little complicated to transform textual programs from $DSL_1$ to $DSL_2$. To achieve it, the $DSL_1$ models are transformed to $DSL_2$ models firstly. Then their models are transformed using $Extractor1$ and $Extractor2$ respectively to the programs. Hence, the textual programs of $DSL_1$ and $DSL_2$ are syntactically transformed.

**4. A case study of bridge construction**

This section presents a case study of the bridge construction from MARTE to timed-LOTOS following the approach demonstrated in Section 3.

**4.1. The metamodels of MARTE & LOTOS**

MARTE is a newly released UML profile for specifying the RTES (Real Time and Embedded Systems). In this paper we focus on the RTEMoCC package that provides the core concepts for RTES design to construct the bridge. Figure 4 illustrates the simplified metamodel of RTEMoCC.

![Figure 4. Simplified RTEMoCC metamodel.](image)

LOTOS is a specification language that has incorporated CCS and CSP for specifying the concurrency behavior and ACT-ONE for specifying abstract data type. In 1989, LOTOS was released as the international standard (ISO 8807). Timed-LOTOS [8] is the enhancement LOTOS with the timed features. Figure 5 illustrates the simplified metamodel of LOTOS.

![Figure 5. Simplified LOTOS metamodel.](image)

Table 1 lists a partial rules that transform MARTE models into LOTOS models (the details are omitted for the reason of space limitation).

The excerpt of LOTOS textual concrete syntax is shown in Listing 1. The key word `PrimitiveTemplate` defines the template which specifies the lexer token corresponding to a given metamodel `DataType`, identified by its name. While the `template` key word defines the template specifying how classes are represented.

---

**Table 1. Transformation Rules Summary**

<table>
<thead>
<tr>
<th>Name</th>
<th>Transformation-Rule &amp; Helpers Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IntegerType</code></td>
<td>Match MARTE.Integer to LOTOS.IntegerType</td>
</tr>
<tr>
<td><code>InterfaceTemplate</code></td>
<td>Match MARTE.Interface to LOTOS.Interface</td>
</tr>
<tr>
<td><code>MethodTemplate</code></td>
<td>Match MARTE.Method to LOTOS.Specifications</td>
</tr>
<tr>
<td><code>String</code></td>
<td>Match UML2.String to LOTOS.Specifications</td>
</tr>
<tr>
<td><code>Duration</code></td>
<td>Match MARTE.Duration to LOTOS.Process</td>
</tr>
<tr>
<td><code>dateTime</code></td>
<td>Match MARTE.DateTime to LOTOS.IntegerType</td>
</tr>
<tr>
<td><code>ActionPrefix</code></td>
<td>Match MARTE.ActionPrefix to LOTOS.Process</td>
</tr>
<tr>
<td><code>Service</code></td>
<td>Match MARTE.Service to LOTOS.Gate</td>
</tr>
<tr>
<td><code>OperationDeclaration</code></td>
<td>Match MARTE.OperationDeclaration to LOTOS.Operation</td>
</tr>
<tr>
<td><code>Operation</code></td>
<td>Match MARTE.Operation to LOTOS.Gate</td>
</tr>
<tr>
<td><code>Parameter</code></td>
<td>Match MARTE.Parameter to LOTOS.Operation</td>
</tr>
<tr>
<td><code>TypeDefinition</code></td>
<td>Match MARTE.TypeDefinition to LOTOS.Gate</td>
</tr>
<tr>
<td><code>Behavior</code></td>
<td>Match MARTE.Behavior to LOTOS.Gate</td>
</tr>
<tr>
<td><code>Equation</code></td>
<td>Match MARTE.Equation to LOTOS.Gate</td>
</tr>
<tr>
<td><code>GateAction</code></td>
<td>Match MARTE.GateAction to LOTOS.Gate</td>
</tr>
<tr>
<td><code>Choice</code></td>
<td>Match MARTE.Choice to LOTOS.Gate</td>
</tr>
<tr>
<td><code>GuardExpression</code></td>
<td>Match MARTE.GuardExpression to LOTOS.Gate</td>
</tr>
<tr>
<td><code>Parameter</code></td>
<td>Match MARTE.Parameter to LOTOS.Gate</td>
</tr>
</tbody>
</table>

---
Listing 1. The excerpt of LOTOS.tcs

```java
syntax LOTOS(k = 0) {

    primitiveTemplate identifier for String default using NAME:
    value = "token";

    primitiveTemplate identifierOrKeyword for String using NAME orKeyword:
    value = "token";

    primitiveTemplate stringSymbol for String using STRING:
    value = "token";
    serializer = '\'' + %value%.toCString() + '\'';

    primitiveTemplate integerSymbol for Integer default using INT:
    value = Integer.valueOf(%token%);

    primitiveTemplate floatSymbol for Double default using FLOAT:
    value = Double.valueOf(%token%);

    template Specification main context:
    "specification" name:
    (isDefined(gates) ? "[" gates{separator = ","} "]") :
    (isExit ? "exit" : "noexit")
    [ types ] {nbNL = 2}
    "behaviour" 
    (isDefined(definition) ? definition)
    "endspec" ;
}
```

5. Related work

The work presented in this paper is related to bridging heterogeneous formal models. The traditional solution to this problem is to build the transformation for the particular formal models, such as those mentioned in [9] and [10], from SMV to PVS, from SMV to Spin, from Automata to Petri Net and from Interface Automata to I/O Automata, etc.

However, these ad-hoc bridges are hard to reuse. As a reflection of this situation, [9] presents a general transformation framework. One shortcoming of this framework is that the semantical mapping and syntactical transformation are not integrated.

Comparing to the traditional methods, the MDE based approach is superior in the following three points: firstly, there is a supporting standard. The framework standard MDA provides the specifications for different methodologies and tools in MDE. The AMMA platform is MDA compatible in that the models could be interchanged using XMI format. Therefore, the bridges constructed on the AMMA platform is of high reusability.

Secondly, the results of each step of the bridge construction could be reused respectively. The metamodels of formal models could be reused in different bridge constructions. The ATL rules could be inherited and extended. The TCS syntax on the metamodel is also reusable in the different bridges.

At last, two sub-problems of semantical mapping and syntactical transformation could be easily differentiated and solved respectively.

6. Conclusion

The problem of bridging heterogeneous formal models is a hot, but difficult, topic in the research field of formal methods. In this paper, we try to consider it in a different way, i.e., from the viewpoint of MDE. Based on the AMMA platform, our approach is practical and useful to both graphical and textual formal models. The main contribution of our work is as follows:

On the one hand, our approach has more general usability. Both graphical and textual formal models are covered by our approach. On the other hand, the bridge constructed in our approach is more reusable. The metamodels, ATL rules and TCS syntax could be reused in different bridge constructions.

Consequently, our approach is helpful to integrate the different formal methods in software development, so as to reuse their verification methods and supporting tools.

References


For Research Only