Reading part: Design-Space Exploration with Alloy

Ing. Ken Vanherpen

Abstract

In the growing world of MDE many tools are offered to describe a (part of a) system, constrain it, and check some properties about it. One of those tools is Alloy, which will be used during this project to explore the design space containing CBDs. This paper will summarize some basic concepts which are fundamental for the project.

Keywords:
MDE, Alloy, DSL, DSE, FORMULA

1. Introduction

The goal of this project is to model Causal Block Diagrams (CBDs) in the Alloy-language (i.e. model a DSL), generate some valid models by using Alloy, where after the models need to be transformed to python models in order to carry out some performance metrics (i.e. DSE). If the optimal model has been located, the reverse path can be followed so as to select the ideal Alloy-model. For this project there are two MDE-topics which are particularly relevant, namely 'Domain Specific Language (DSL)' and 'Design Space Exploration (DSE)'. Both of these topics will be discussed in this paper, which summarizes some fundamentals which are relevant for the project. Section 2 will discuss the used tool Alloy. Section 3 will dicuss some frameworks which will become handy. In section 4 some papers are described where Alloy was used to automatic complete partial models. Also, a comparison with some other tools is made. Section 5 will conclude this paper.

2. Alloy

Alloy [MIT, 2013] is a declarative constraint language for describing structures and a tool (Alloy Analyzer) for exploring them. One of the key benefits

of Alloy is the use of quantifiers such as *one*, *lone*, *all*, *some* to specify constraints, called 'facts'. More over, constraints are expressed as first-order relational logic statements, making Alloy very suitable for analysis and synthesis purposes (i.e. DSE). Once an Alloy model is modeled, the Alloy Analyzer compiles it into a propositional Conjunctive-Normal Form (CNF) which are solved using an appropriate satisfiability (SAT) solver. The Alloy Analyzer will try to find structures which satisfy the constraints, and return them in the form of models which are called 'counterexamples'.

In this paper it will become clear that Alloy has both proponents and opponents. The SAT-solver will appear to be the bottleneck, especially compared to the more effective Satisfiability Modulo Theory (SMT) solver which solves expressions based on a more expressive language.

3. Useful frameworks

The first two papers which are cited in this section, describe a framework which may be useful throughout this project.

In [Sen and Vangheluwe, 2006] a first framework is represented which facilitates rapid model design and transformation by using Graph Grammar rules. A multi-domain problem is first meta-modeled using Unified Modeling Language Class Diagram (UML CD), and further constrained using OCL or Python. The presented framework consists out of three meta-models:

1. MM of the Real World.
2. MM of the Idealized Physical World.
3. MM of Bond Graph.

Each of those meta-models specifies a formalism which is used to create a model. By using Graph Grammar rules, the paper represents how a model expressed in one formalism can be transformed into a model expressed in another formalism. Ultimately, models are represented using an object-oriented textual formalism called Modelica. Those models can be simulated using a so called Differential-Algebraic Equation (DAE) solver. A Trajectory formalism expresses the simulation results of a multi-domain problem.

A second framework is presented in [Saxena and Karsai, 2010], where a Generic Design Space Exploration (GDSE) framework is used to solve DSE problems from different domains. The GDSE framework consists out of four steps:

1. Design a Domain Specific Modeling Language (DSML). In fact, a meta-model is modeled by a domain expert capturing entities in the domain and their relationships. The meta-model can then be used to solve multiple DSE problems within the modeled domain.
2. Extend the DSML (eDSML). The meta-model of the first step is extended to capture specific elements (such as constraints) related to a specific DSE problem.
3. Create a domain specific design space model. The eDSML is then used to create a design space, which can be employed by domain-engineers to create instances.
4. Perform DSE. The design space model, which is created in the previous step, is translated to a specific solver.

To constraint the meta-model (step 2), a Constraint Specification Language (CSL) was developed in order to overcome the limitations of some other languages:

- ILP is limited by using only linear constraints.
- SAT uses only boolean constraints.
- OCL is sufficiently expressive, but the used expressions can become verbose and hard to read.

CSL is closely related to OCL, although there are two main differences:

1. CSL supports dual-context expressions, where two contexts are referenced by keywords.
2. Expressions in CSL are algebraic relationships between attributes of the classes.

What is interesting in this paper is the translation of the design space model to different solvers (step 4) by using a two-stage transformation, first to a model in Intermediate Language (IRL) by using Graph Grammar and then to a solver-independent medium-level language called Minizinc. By using IRL all the visual and other details not related to DSE are captured as well as the design space and its constraints in a form closer to one required by solver languages. Moreover, IRL allows the use of symbolic constraint satisfaction tools that work at a higher abstraction level as compared to ILP and SAT. A DSE-problem is presented in this paper where the problem is modeled step

by step using the step plan. Unfortunately, no comparison is made between their results and the result when solving the problem using an ILP or SAT solver.

4. Use of Alloy

In [Sen, Baudry and Vangheluwe, 2008] a methodology is presented for automatic model completion by using Alloy. Three reason are given to indicate why model completion are more difficult compared to code completion:

1. Code completion techniques use the BNF grammar while models are specified by a meta-model and constraints on it.
2. Model completion must consider the entire model as constraints span the entire model.
3. Model completion must help to reduce the effort of a modeller by automatically satisfying all relevant constraints.

In this paper, a meta-model is built directly in AToM^3 model editor, whereby the constraints of the meta-model are defined using Alloy facts. The methodology presented consists out of four steps:

1. Design a DSML in AToM^3, consisting out of a meta-model constrained by Alloy facts and a concrete visual syntax. By constraining the meta-model using Alloy, which is a textual constraint language, the semantics have no side effect on the meta-model or its instances.
2. Transformation of the DSML to a model editor. In fact, creating a domain specific design space model.
3. User interactions which includes: drawing a partial model, editing some model completion parameters and click a button to generate complete model(s).
4. Model completion where the meta-model, Alloy model and Alloy facts are synthesized in order to solve a final Alloy model and return complete models as recommendations to the model editor.

In order to complete a partial model, first the meta-model is translated to an Alloy model. Hereby, special care has to be taken when transforming classes, multiplicity, containment and inheritance relations of a meta-model to Alloy signatures and facts. Second, the partial model is translated by synthesizing an Alloy predicate (a more relaxed form of facts) from a partial model.

The model completion process takes the partial model (drawn by the user) as an input, along with the meta-model, the Alloy facts, and a set of parameters which defines the scope of the complete models to be synthesized. The complete model recommendations are then presented in the domain-specific model editor in their concrete visual syntax.

The paper illustrates that the presented methodology is effective for small partial models, whereby the level of detail of the partial model is crucial in terms of presenting complete model recommendations.

A last paper represents a so called FORMULA framework as a tool towards generic automation for MDA [Jackson, Kang, Dahlwaid, Seifert and Santen, 2010], similar to the previous described paper. Very interesting about this paper is the comparison with other tools as for example Alloy. The main difference between FORMULA and Alloy is their underlying solver. FORMULA utilizes symbolic execution of logic programs and reduction to a Satisfiability Modulo Theories (SMT) solver for model synthesis. SMT solvers are characterized by powerful decision procedures for theories, including linear arithmetic, uninterpreted functions, term algebras and bit vectors. Alloy, more specific the Alloy Analyzer, on the other hand uses first-order relational logic specifications. These are encoded as a matrix by a back-end compiler, compiled into a CNF where after it is represented to a SAT-solver. The size of the matrix is exponential in the arity of the relation.

Experiments shown in this paper reveal that automatic completion using Alloy is not appropriate for industrial case studies, especially when the degrees of freedom in the synthesis problem is large. The problem manifest itself when encoding the matrix using the back-end compiler, which runs out of memory capacity, even before reaching the SAT-solver. This is due to the compilation into a CNF, whereby only bit-level reasoning can be applied.

5. Conclusions and future work

One can conclude that the use of Alloy is not appropriate for large cases, which is fortunately is not the case in scope if this project. The use of boolean constraints makes modeling in Alloy certainly not easier, so some expertise is necessary. It seems that the tutorial on the main web page [MIT, 2013] is a good starting point to get familiar with Alloy and its boolean constraints. However, it is clear that Alloy is characterized by a large learning curve.

Elaborating the actual project, as explained in the section 1, will be done using the frameworks represented in section 3. Specially the second framework will become handy.

References

Jackson, E., Kang, E., Dahlwaid, M., Seifert, D., Santen, T., 2010. Components, platforms and possibilities: Towards generic automation for mda.

MIT, 2013. Alloy. URL: `http://alloy.mit.edu/alloy/index.html`.

Saxena, T., Karsai, G., 2010. Mde-based approach for generalizeing design space exploration. MODELS 2010 I, 46–60.

Sen, S., Baudry, B., Vangheluwe, H., 2008. Towards domain-specific model editors with automatic model completion.

Sen, S., Vangheluwe, H., 2006. Multi-domain system modeling and control based on meta-modeling and graph rewriting.