A Discrete/Continuous Constraint Net for Consistency Checking of Requirements Models

Axel Ratzke, Christoph Grimm
TU Kaiserslautern, Chair of Cyber-Physical Systems
[ratzke|grimm]@cs.uni-kl.de

***Abstract*—Requirements in complex systems can quickly introduce unforeseen inconsistencies or contradictions. This work shows the use of discrete/continuous constraint propagation to analyze the consistency of requirements models. The implementation checks representations using the SysMLv2 metamodel that are generated by parsers for a SysMLv2 textual subset or the proprietary language SysMD.**

***Index Terms*—configuration, constraint propagation, automotive, symbolic AI**

I. Introduction

Information that is semantically integrating information across the supply chain is a trend that will be dominating systems engineering processes and frameworks by the year 2035 [5]. This work in progress deals with constraint satisfaction- and propagation (CS) techniques for consistency checking in requirements models that are exchanged in the supply chain. The constraint propagation is intended to support the creation of consistent requirements models by showing contradictions in models, supporting the specification consistent value-ranges and units for parameters, assisting in the selection of suitable variants. Particular focus is on the consistency checking of SysMLv2/KerML models with CS.

A. Progress beyond state of the art

Several approaches use CS in the context of technical systems. A backtrack-free configuration approach for a domain-specific language Configit-PM is explored in [3]. COMPASS [1] gives a model based approach based on SysML and CML by refinement of (dynamic) state machine behavior.

This work aims at mapping a subset of SysMLv2's KerML representation to a CS problem. Compared with [3], [1], we target KerML representations. Such models combine discrete and continuous dependencies by a variety of modeling artifacts including effect chains, direct dependencies of values, SI unit support, and inheritance. The contribution of the work lies in closing the gap between KerML models and methods for CS, and finding CS methods that satisfy the needs for interactive work, i.e. response times, introspection, reasonable feedback.

This paper is structured as follows: Section II gives an overview of the overall framework. Section III demonstrates small examples for discrete and continuous models that are translated to KerML models. An outlook to future work is given in Section IV.

This work was funded by the German BMBF within the project GENIAL!

II. The AGILA/SysMD framework

The overall framework in which we deploy the constraint propagation consists of two parts. The frontend, SysMD Notebook, leans towards the popular Jupyter Notebooks. It allows a user to mix natural-language documentation and documents in the Markdown (MD) format with requirements models in a subset of SysMLv2 textual or its own proprietary language called SysMD. The backend, AGILA, holds a repository of models. It implements services including version management, user- and rights management, and the constraint network (CN) for consistency checking. The frontend and the backend are coupled via a service layer and/or via a REST API as shown by Figure 1.

Fig. 1. High-level architecture of AGILA/SysMD.

The SysMD compiler or, likewise the SysMLv2 [6] subset compiler, translates models into objects from the KerML metamodel [4]; an overview of the classes is shown by Figure 2.

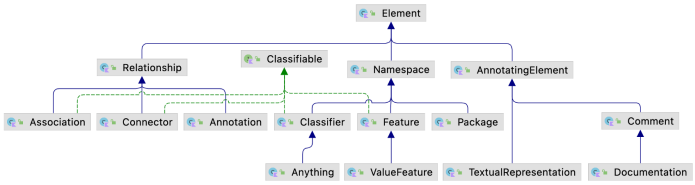

Fig. 2. SysMD/AGILA Implementation of the SysMLv2 metamodel.

In the backend, models are represented by objects of the classes Documentation (Markdown; is rendered by the frontend) and TextualRepresentation (language can be selected to be SysMLv2 or SysMD). After compiling such an object, the compiler generates interwoven trees of metamodel objects of Relationship, Classifier rsp. Interface Classifiable, or Feature (or, subclasses of these classes): The tree of Classifiable objects models specializations with the object Anything as root. A second tree models the ownership (owns/ownedBy) relationship and a root-namespace Global.

III. Constraint propagation in SysMD

In a requirements model, we model dependencies among different elements by arithmetic and Boolean expressions. In SysMD they are of the form:

```
Name: [All|One] Type [(Constraint)] [Unit] [= Expr]
```

where *Name* is a name of a variable, *Type* a defined type; currently we support Boolean, Integer, Real, String. *Constraint* specifies upper and/or lower bounds for the expected value. *Unit* is a SI or derived unit. Expr(ession) is a Boolean or Arithmetic or mixed Boolean/Arithmetic expression on other variables, multiplicities, or queries on the design structure. *All* specifies that the expression must be satisfied for all values of the Type with the given Constraint. *One* specifies that there must exist at least one value that satisfies the expression; it is the default.

A. Continuous constraint propagation

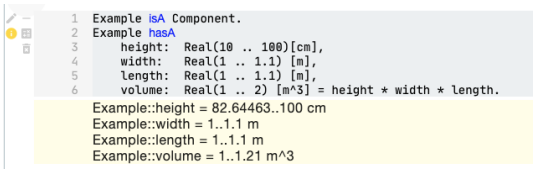

Fig. 3. Example continuous constraint propagation in SysMD Notebook

An example propagation of continuous constraints can be seen in Figure 3. The compiler uses the KerML classes "Classifier" and "Feature" with value and expression. For the CN we use the method of approximating and restricting the possible solution space. For the approximation and restricting the solution space, we use a combination of Interval Arithmetic, AA, ROBDD, and linear programming; for details see [2], [8]. Furthermore, consistency of units and its conversion is checked.

B. Discrete constraint propagation

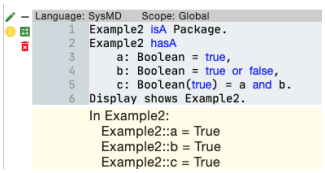

Fig. 4. Example for discrete constraint propagation in SysMD Notebook

A discrete example can be seen in Figure 4; again, a Classifier and 3 Feature classes with Expressions are used. The discrete part of the CN uses the method of domain reduction, where values that lead to inconsistent properties are removed (which means the constraints imposed on the domain are restricted). And a combination of different network consistency techniques is used subsequently to propagate the changes across the properties.

C. Mixed constraint propagation, Inheritance

Interactions between the discrete and continuous domain are modeled by comparison operations and the ITE-function that can select one of two Real/Integer parameters. The continuous and the discrete domain then take alternating turns, where according to the domain, solution space and variable domain respectively are restricted and reduced respectively until a fixed point is reached. The new state of one part of the network is then used as input for the other part of the network and possible changes are propagated by network consistency techniques. The procedure is then repeated until again a fixed point is reached. During this process new properties are generated based on the results of the net that refine the model, without the loss of valid solutions. Currently, the interaction between the two parts of the constraint net at the two modelled interaction points is the focus of this work in progress. Further interactions between the parts such as Boolean logic expressions over reals and integer (range) domains are researched.

Inheritance is handled in a separate step before the constraint net starts. It in particular requires cloning inherited feature values that are written by the constraint nets.

IV. Results and Future Work

The current implementation allows us to check consistency of KerML models that use inheritance (Classifier), Feature Models (Feature) with multiplicities, and simple queries of the backend. As of now, the runtime of the CN for models with some hundred variables is negligible. In future work, we plan to compare runtime with results for more complex models from e.g. [7]. Furthermore, we still need improvements regarding the integration of the discrete and continuous parts.

While the current work relies on analysis of decision diagrams, more domain specific constraint solving techniques for their respective domain and their interaction will need to be researched regarding its usability for interactive work. This in particular includes

References

[1] J. Bryans, R. Payne, J. Holt, and S. Perry. Semi-formal and formal interface specification for system of systems architecture. In *2013 IEEE International Systems Conference (SysCon)*, pages 612–619, United States, 2013. IEEE. 2013 IEEE International Systems Conference : SysCon 2013 ; Conference date: 15-04-2013 Through 18-04-2013.

[2] Christoph Grimm and Michael Rathmair. Dealing with uncertainties in analog/mixed-signal systems: Invited. In *DAC 2017, Austin, TX, USA*, pages 35:1–35:6, 2017.

[3] Tarik Hadzic, Sathiamoorthy Subbarayan, Rune M Jensen, Henrik R Andersen, Jesper Møller, and Henrik Hulgaard. Fast backtrack-free product configuration using a precompiled solution space representation. In *Int. Conf. on Product Configuration Systems*, pages 131–138. 2004.

[4] OMG. Kernel Modeling Language (KerML), Release 05/2022.

[5] OMG. Systems Engineering Vision 2035, Release 06/2022.

[6] OMG. Systems Modeling Language (SysMLv2), Release 05/2022.

[7] Chico Sundermann, Thomas Thüm, and Ina Schaefer. Evaluating sat solvers on industrial feature models. In *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*, VAMOS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[8] Carna Zivkovic, Christoph Grimm, Markus Olbrich, Oliver Scharf, and Erich Barke. Hierarchical verification of AMS systems with affine arithmetic decision diagrams. *IEEE TCAD*, 38(10):1785–1798, October 2019.