Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

# Formal Validation and Model Generation for Domain-Specific Languages by Logic Solvers

Thesis booklet

**Oszkár Semeráth**

Thesis supervisor:
**Prof. Dániel Varró**

Budapest, 2019

# 1 Introduction

## 1.1 Domain-specific modeling languages

My thesis is motivated by the challenges in the development of complex, safety-critical systems such as automotive, avionics or cyber-physical systems, which is characterized by a long development time and strict safety standards (like DO-178C or DO-330). *Model-Based System Engineering (MBSE)* is a widely used technique in those application domains [WHR14], which facilitates the use of models in different phases of design and on various levels of abstraction. Furthermore, MBSE promotes the use of *Domain-Specific (Modeling) Languages (DSLs)* to precisely capture the main features of a target domain, thus enabling the engineers to model their solutions with the concepts of the problem domain. Additionally, advanced *modeling environments* can automate several development steps, with a particular emphasis on verification and validation (V&V). Thus they can significantly improve the overall productivity of the development and quality of the product.

A complex industrial modeling environment supports the development of models by continuously evaluating consistency constraints to ensure that the models satisfy the design rules of the domain. There is already efficient support for automatically validating constraints and design rules over large model instances of the DSL using tools like Eclipse OCL [Wil12; KPP09] or Viatra [Ber+11; Ber+10]. Modeling environments often incorporate the automated generation of various artifacts such as source code, task specific views, or documentation by using code generation or model transformation techniques. Additionally, the environment may incorporate mathematical analysis techniques to verify the correctness of the underlying design (e.g. by model checking).

Industrial modeling environments like Capella (by Thales), Artop, Yakindu (by Itemis), MagicDraw (by NoMagic) or Papyrus UML (by CEA) are frequently built on top of open source DSL frameworks such as Xtext , or Sirius  built on top of model management frameworks such as Eclipse Modeling Framework to significantly simplify productivity of tool development by automating the production of rich editor features (e.g. syntax highlighting, auto-completion) to enhance modeling for domain experts.

## 1.2 Towards the validation of modeling environments

However, the design of modeling environments for complex domain-specific languages is still a challenging task. As such tools are frequently used in critical systems design to detect conceptual flaws of the system model early in the development process to decrease verification and validation (V&V) costs, those tools should be validated with the same level of scrutiny as the underlying system as part of a software tool qualification process in order to provide trust in their output. Therefore, the need for software tool qualification (e.g. defined in safety-related standards DO-330) raises several challenges for building trusted DSL tools for a specific domain.

### 1.2.1 Architecture of a modeling environment

Architecturally, a modeling environment is composed of multiple components that need to be validated. First, the language specification of a DSL is based on a *Metamodel (MM)*, which defines the main concepts and relations of the language. The metamodel is extended by additional *Well-Formedness (WF) constraints* to restrict the range of valid models. Moreover, industrial models are frequently extended by *Derived Features (DF)*, which are calculated model elements offering shortcuts to access or navigate models. All together MM, WF and DF specify the graph-based data structure of instance models of the DSL (so-called *abstract syntax*). A modeling environment defines one or more concrete textual or graphical representations for a model (*concrete syntax*), and supports the editing, saving (serializing) and loading (parsing) of the models by providing advanced editors.

*View models* are a key concept in advanced modeling environments to provide viewpoint-specific focus (e.g., power flow or communication architecture of a system) to engineers by deriving another model (or diagram, table, etc.) which highlights relevant aspects of the system to help detect conceptual flaws. Typically, multiple views are defined for a given model (referred as source model), which are refreshed automatically upon changes in the source model. The derivation and maintenance of views have been extensively studied for a long time in database theory over relational knowledge bases, while it has recently become a popular research topic in MBSE [Deb+14; Gho+15; MC13].

Additionally, a modeling environment incorporates several procedures that use valid models as input, e.g. *model transformations* and *code generators*. Like any piece of software, those components are not free from flaws.

### 1.2.2 Language level validation

In case of complex, standardized industrial domains (like ARINC 653 for avionics or AUTOSAR in automotive), the sheer complexity of the language specification and the models is a major challenge in itself. (1) First, there are hundreds of well-formedness constraints and design rules defined by those standards, and due to the lack of validation, there is no guarantee for their consistency (one constraint may contradict another) or subsumability (one constraint may accidentally cover another). (2) The specification of derived features can also be inconsistent (e.g. DF specification may contradict another design rule), ambiguous or incomplete (DF specification implies more or fewer values than expected). In summary, the consistency and unambiguity of a DSL specification have to be ensured.

The definition of such large DSLs is a very challenging task not only due to their size and complexity, but also as we need to precisely understand the interactions between the additional design rules. Declaring a large number of DFs is also challenging with respect to the safety-specific WF constraints.

In general, mathematical precise *validation of DSL specifications* has been attempted by only a few approaches so far [JLB11; JS06], and even these approaches lack a systematic validation process. Therefore, I have identified language-level validation as the first research question of my thesis.

**Research Question 1: Language Validation.** How to validate the consistency, completeness and unambiguity of DSL specifications?

### 1.2.3 Testing modeling environments

Even if the DSL specification is sound, the implementation of the modeling tool may contain errors. The need for software tool qualification raises several challenges to build trusted DSL tools for a specific domain which typically uses systematic testing methods for the modeling environment itself. Such testing of modeling environments is significantly enhanced by the *automated generation of valid (or intentionally faulty) models* as test inputs [Mou+09] for DSL modeling tools, model transformations or code generators [Bro+06].

Testing of critical systems frequently uses coverage metrics like MC/DC [Hay+01] to assess the thoroughness of a test suite and safety standards prescribe designated minimal test coverage with respect them (e.g. 100% MC/DC coverage for components with SIL5-level criticality). However, existing coverage metrics are dominantly developed for imperative source code, and they fail when applied on testing of modeling environments, which are dominantly data-driven applications working with complex graph structures (e.g. 100% MC/DC coverage can be achieved with a low number of trivial examples while it still provides very little assurance in reality). Therefore, coverage metrics are needed to objectively measure the quality of a test suite in such applications.

**Research Question 2: Tool Testing.** How to provide a test suite of instance models for a complex DSL with designated coverage?

### 1.2.4 Backward view synchronization

When several views are derived from a source model in a complex modeling environment, such view models are dominantly read-only representations derived by a unidirectional transformation from a source model, and those views cannot be changed directly. When a view model needs to be changed, the engineer is forced to edit and manually check the source model until the modified model corresponds to the expected view model. Additionally, the effects of a source change need to be observed in all other view models to avoid unintentional changes and to prevent the violation of structural well-formedness (WF) constraints. My thesis focuses on the back propagation of view changes.

My goal was to add backward change propagation support to view models in the modeling environment, so upon a change in the view model, a set of candidates for corresponding source model changes could be created.

**Research Question 3: View Synchronization.** How to propagate changes of a view model back to the source model to restore consistency?

### 1.2.5 Graph model generation

All the three questions 1-3 lead to the challenge of model generation as an underlying technique: test suite generation requires the generation of different instance models as test inputs, view synchronization uses model generation to create source model change candidates, and language validation requires the non-existence of counterexample models with undesired properties (like incompleteness and ambiguity) [JS06]. This leads to an unifying main research question of the thesis:

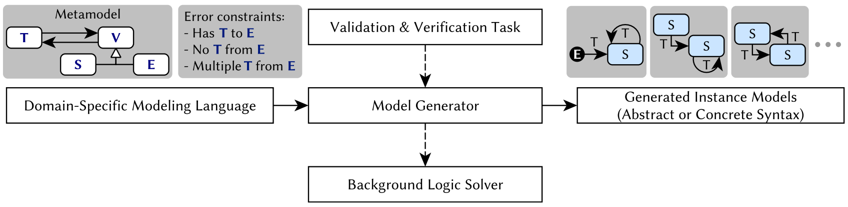
Metamodel
T
V
S
E
Error constraints:
- Has T to E
- No T from E
- Multiple T from E
Validation & Verification Task
Domain-Specific Modeling Language
Model Generator
Generated Instance Models (Abstract or Concrete Syntax)
Background Logic Solver
T
S
E

Figure 1. Setup of automated model generation

**Research Question 4: Graph-Model Generation.** How to automatically generate valid instance models for DSLs?

Figure 1 illustrates the use of model generation for various challenges of DSL environments as proposed in the current thesis. The setup is based on a *model generator* that reads the definition of the target *Domain-Specific Modeling Language* (typically defined by a metamodel and some well-formedness constraints) to carry out designated *validation and verification tasks* (e.g. expected language property, view model change or required test coverage) by producing certain *instance models* (or proves that no such models can be constructed).

Automated graph model generation is also a prerequisite of several other research lines. Object-oriented data structures can be represented as graphs of objects and pointers, and such, test generation [Mil+07; MK01] requires graph generation with different structures to discover bugs. Similarly, static analysis and verification techniques like [Ren04; RSW04] are using graph consistency checking techniques to ensure that certain invalid structures cannot emerge. Auto-generated graphs may also help the testing and benchmarking in other domains like graph databases [Bag+17; Szá+17; Szá19], since obtaining real graphs from business use cases is often difficult to due to the protection of intellectual property rights.

## 2 Challenges in model generation

As both language level validation, testing and view synchronization (**RQ1-3**) are based on generation of well-formed models (**RQ4**) and they imply similar inherent challenges. Figure 2 illustrates the connections between those research questions and challenges.

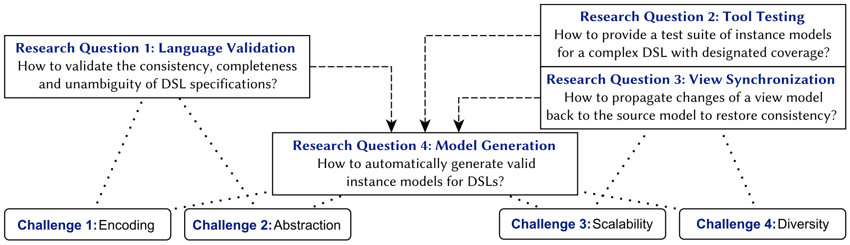
Research Question 1: Language Validation
How to validate the consistency, completeness and unambiguity of DSL specifications?
Research Question 2: Tool Testing
How to provide a test suite of instance models for a complex DSL with designated coverage?
Research Question 3: View Synchronization
How to propagate changes of a view model back to the source model to restore consistency?
Research Question 4: Model Generation
How to automatically generate valid instance models for DSLs?
Challenge 1: Encoding
Challenge 2: Abstraction
Challenge 3: Scalability
Challenge 4: Diversity

Figure 2. Relation between challenges and research questions

The main difficulty lies in the complexity of a DSL specification: each generated model needs to satisfy a set of *complex*, *global*, *interacting structural constraints* (a typical characteristic for DSLs). This necessitates the use of advanced *logic reasoning* technique during generation with a *background logic solver*, and a mapping of the DSL specification to (first order) logic and back. Although existing logic solving tools and algorithms (like SMT [DMB08] or SAT-solvers [Jac02]) are getting more and more powerful, their application imposes several challenges. In the following, I highlight four main challenge groups for using logic solvers in (graph) model generation.

First, the complete specification of the target DSL needs to be automatically translated into a formal language in order to represent model generation as a mathematical reasoning problem. The language elements in a DSL specification are representable by sets and relations, and first order logic with transitive closure may formalize most constraints that are needed in practice.

**Model Generation Challenge 1: Encoding.** How to encode DSL specifications as a logic problem where solutions of the logic problem represent valid models?

Language level validation (**RQ1**) gives particularly emphasis on the encoding to ensure that all language elements are correctly and completely mapped to logic. Similarly, a model generator (**RQ4**) need to support the logic problem created from a DSL.

Language level reasoning required for language validation (**RQ1**) is a challenging task as complete analysis has to cover an infinite range of possible design models. Moreover, a DSL specification uses a more expressive specification than first order logic, which is already undecidable. This necessitates the use of sophisticated abstractions and bounded analysis techniques, which sacrifice soundness or completeness, but are still able to carry out the target verification task by proving a stronger assumption, or giving counter-example to weaker one. Similarly, abstraction and approximation are key concepts in logic solvers used for model generation (**RQ4**).

**Model Generation Challenge 2: Abstraction.** How to provide efficient abstractions for language-level analysis of a DSL?

Even if an abstraction or encoding is sound, it may imply scalability challenges for model generation in practical application scenarios. As the metamodel of an industrial DSL may contain hundreds of types and relations, any meaningful instance model can easily be of similar size. Unfortunately, model generation cannot currently be achieved by a single direct call to the underlying (SAT/SMT) solver [JLB11][C10] due to scalability issues. Our finding was that existing model generation approaches based upon mappings to underlying SMT/SAT solvers could only scale to very small problems due to the fact that the mapping introduces far too many Boolean variables to encode potential graph nodes and edges that immediately explode the search space of the solver. Moreover, when rewriting (language-level) well-formedness constraints to their equivalent Boolean formula over a particular model, the size of this formula is grows rapidly with the size of the model. As such, evaluating a very large formula already creates major challenges for solvers. Indeed, we found no published results in related literature using logic solvers for graph model generation purposes that were able to derive graphs with over 150 nodes. Therefore, existing solver-based generators using currently available logic solvers fail to derive non-trivial and useful models for complex DSLs (**RQ4**). This hinders the use of model generators in practical tool testing and view synchronization scenarios (**RQ2**-**3**).

**Model Generation Challenge 3: Scalability.** How to derive large instance models for complex DSLs?

Finally, existing logic solvers tend to retrieve simple, unrealistic models consisting of unconnected islands, many isolated nodes and highly symmetric (copy-paste) fragments, which is problematic in a real testing scenario (**RQ4**). In fact, test suites created by existing solvers often contain highly similar (or even isomorphic) model sequences, which violates the common best practice of testing, i.e. to use a diverse set of test inputs that cover various equivalence classes. Furthermore, there are no widely used coverage metrics for measuring the diversity of graph-based models. Even preventing isomorphic solution in a model generator is challenging, as it may necessitate computationally expensive graph isomorphism checks.

**Model Generation Challenge 4: Diversity.** How to measure model diversity and generate a diverse set of models?

Conversely, in bidirectional synchronization (**RQ3**), one of the main goal is to avoid unnecessary changes into prevent information loss.

## 3 Research method

My research method has been aligned with the best practices of software engineering research. First, all **Research Questions 1-4** and **Challenges 1-4** have been motivated by generalizing practical issues gained in the context of industrial case studies. Moreover, the feasibility and usefulness of conceptual contributions have been demonstrated by developing prototype implementations (grouped into the open source VIATRA Solver framework [C8]). I integrated my contributions into general-purpose industrial modeling technologies (like EMF [Emf] and VIATRA [Ber+11; Ber+10]). Finally, I evaluated the feasibility and scalability of my approach using the prototype implementation on several (scalability) benchmarks derived from existing modeling environments and models of case studies. In my thesis I use three case studies to illustrate the challenges and my solutions on language validation, test generation, and view synchronization.

**Avionics Architecture.** An avionic DSL validation case study is taken from Trans-IMA [Hor+14] project (and used to answer **Research Question 1**). Trans-IMA aims at defining a model-driven approach for the synthesis of integrated Matlab Simulink models amenable to simulating the software and hardware architecture of an airplane.

**Yakindu Statecharts.** In my thesis, I used Yakindu Statecharts Tools [Yak] as an industrial case study for test generation (used in **Research Questions 2** and **4**). Yakindu Statecharts Tools is an integrated modeling environment developed by Itemis AG for the specification and development of reactive, event-driven systems based on the concept of statecharts captured in combined graphical and textual syntax.

**Remote Healthcare System.** My change propagation technique (**Research Questions 3**) is illustrated in a case study of a remote health care system developed in the Concert ARTEMIS project [Con]. The Concerto ARTEMIS research project proposed a reference multi-domain architectural framework for complex, highly concurrent, and multi-core critical systems. As a subproject, it developed a multi-view, hierarchical cross-domain design environment for heterogeneous platform architectures.

## 4 Summary of the Research Results

### 4.1 Contribution Overview

The results of my thesis are organized around the model generation setup, and illustrated in Figure 3). **Contributions** are highlighted in blue areas, solid lines denote transformations, dashed lines are dependencies, and dotted lines connect the **Contributions** with **Research Questions** and **Challenges**.

My first contribution group covers a *model generation framework*, addressing **Research Question 4**. The framework takes the formal description of a domain-specific language and a task task encoded as a logic problem, solves the problem with a background solver, and creates solutions representing instance models. The framework (and the contribution group) features an highly efficient novel *Graph Solver* algorithm, and integrates other popular background logic solvers (Z3 SMT solver [DMB08] and Alloy [Jac02; TJ07] with two underlying SAT solvers [LBP10; ES03]).

Next, my second contribution group proposes an encoding (**Challenge 1**) of domain-specific modeling languages (which covers EMF meta- and instance models, and Viatra queries) as logic problems. It supports abstraction techniques (**Challenge 2**) to over- and under-approximate constraints to a decidable fragment of logic [PMB08]. It formulates several DSL *validation tasks* (**Research Question 1**) based on the encoding of the DSL elements, which can be checked with the underlying model generator framework.

My third contribution group features iterative model generation techniques. It proposes the generation of models in multiple steps, reusing the previous solution(s) in the construction of the next. For test input generation (**Research Question 2**), it proposes an *incremental* and *diverse* model generation techniques. In incremental model generation, each model is generated as a sequence of models increasing in size where each generation step adds new elements to the existing models, thus improving scalability (**Challenge 3**). For diverse model generation (**Research Question 3**), the thesis proposes a shape-based [Ren04; RD06] distance metric to measure the diversity of models, and controls the model generation by ensuring a minimal distance between each model (facing **Challenge 4**). For view model synchronization (**Research Question 3**), the thesis proposes a backward change propagation technique that reuses the previous state of the source model (as a submodel) when it generates source model candidates to restore consistency.

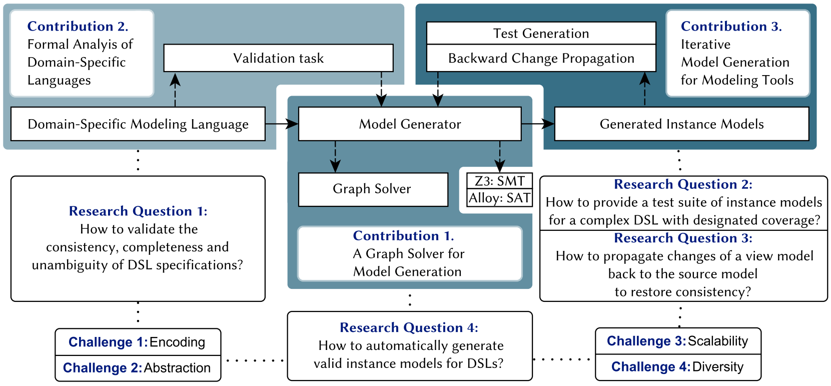
Contribution 2.
Formal Analyis of Domain-Specific Languages
Validation task
Test Generation
Backward Change Propagation
Contribution 3.
Iterative Model Generation for Modeling Tools
Domain-Specific Modeling Language
Model Generator
Generated Instance Models
Graph Solver
Z3: SMT
Alloy: SAT
Contribution 1.
A Graph Solver for Model Generation
Research Question 1:
How to validate the consistency, completeness and unambiguity of DSL specifications?
Research Question 2:
How to provide a test suite of instance models for a complex DSL with designated coverage?
Research Question 3:
How to propagate changes of a view model back to the source model to restore consistency?
Research Question 4:
How to automatically generate valid instance models for DSLs?
Challenge 1: Encoding
Challenge 2: Abstraction
Challenge 3: Scalability
Challenge 4: Diversity

Figure 3. Contribution overview

## 4.2 A graph solver for model generation

My first group of contributions deals with the background logic solvers that required for model generators to address **Research Question 4**.

**Contribution group 1.** I proposed a novel logic solver that operates directly on graph models. I integrated existing SAT [Jac02; TJ07; LBP10; ES03] and SMT [DMB08] solvers to the framework.

**1.1.** *Partial Modeling with 3-Valued Logic.*
I introduced a partial modeling formalism that uses 3-valued logic with interpreted equivalence and existence symbols. I showed that the new formalism is able to represent other popular partial modeling techniques (MAVO and TVLA) [C5][C7].
**1.2.** *Evaluation of Graph Patterns on Partial Models.*
I presented a graph predicate rewriting technique to enable the evaluation of under- and over-approximation of predicates directly over the representation of a partial model [C5] by graph query engines. I applied the technique to derive additional validation constraints to highlight unrepairable inconsistencies in modeling environments. [J1].
**1.3.** *Graph Solver approach.*
I created an efficient logic solver that generates consistent graph models for domain-specific languages. The approach uses (i) partial (graph) models as states and (ii) partial model refinement decision and unit propagation rules. The generation is is controlled by (iii) a design-space exploration engine which (iv) continuously evaluates the approximations of well-formedness constraints on partial solutions [C6].
**1.4.** *Implementation: Viatra Solver framework.*
I developed an open-source implementation of the graph solver [Vs], which also integrates existing logic solvers like Z3 or Alloy. The tool is available both as an integrated modeling tool and as a standalone application [C8].
**1.5.** *Experimental Evaluation.*
I carried out experimental scalability evaluation of the graph solver in three case studies of industrial DSLs.

**Added value.** The proposed approach is implemented in a framework [Vs][C8] that is able to solve model generation problems with three underlying solvers, including the novel graph generation technique which operates directly on graph models. The proposed partial modeling technique introduced a sophisticated encoding technique, which enabled the efficient evaluation of constraints on abstract, partial solutions (**Ch1: Encoding** and **Ch2: Abstraction**). In [C6] we showed that the graph solver is able to synthesize consistent graph models with over 500-6000 objects with similar consistency guarantees as other solvers (thus significantly outperforming them in **Ch3: Scalability**). The scalability of our solver is 1-2 orders of magnitude better than existing mapping based approaches using Alloy [TJ07] with state-of-the-art SAT-solver in the background (which scaled only up to models with 80 objects). In [C11] we showed that the generated models are also more diverse (**Ch4: Diversity**).

**Additional applications and related contributions.** Our research line on model generators is preceded by the development of several performance benchmarks for modeling and model management tools [VSV05; Ber+08] culminating in major open benchmarks like the Train Benchmark [Szá+17][C18] and the CPS Benchmark [Cps].

Design-space exploration (DSE) is a related technique for creating model candidates which are optimal with respect to given objective functions using a set of transformation rules. Viatra-DSE is a rule-based design space exploration technique for (graph) models [HHV15; Abd+14], which is the contribution of Ákos Horváth, Ábel Hegedüs and András Szabolcs Nagy.

In [C7], we extended 3-valued partial models [C5] as a background theory for representing inconsistent and ambiguous views models. The original theoretical background is my contribution, and the view modeling approach is a separate contribution from Kristóf Marussy.

## 4.3 Language-level validation for DSLs

The second group of contributions focuses on the language level validation of domain-specific languages to address **Research Question 1**.

**Contribution group 2.** I proposed formal analysis techniques for the language-level validation of domain-specific languages by mapping them to formal logic specifications.

**2.1.** *Mapping of graph patterns to effectively propositional logic.*
I introduced a technique to transform WF and DF rules captured by graph patterns to a decidable fragment of FOL [PMB08] by using over- and underapproximation techniques [C9].
**2.2.** *Identification of context dependent DSL validation criteria.*
I defined completeness and unambiguity properties of derived features, subsumption and equivalence relations of well-formedness constraints, derived features and instance models. These properties can be checked on the full DSL, or on a specific fragment of it [J2].
**2.3.** *Uniform validation of DSL specification.*
I introduced a technique that uniformly translates DLS elements to FOL

to analyze the consistency of the whole DSL specification, which includes metamodels, instance models, well-formedness (OCL or graph pattern) and derived features. I proposed a validation process for the DSL which systematically checks the language properties, and highlights inconsistencies by deriving representative counterexamples which violate the target properties [C9][J2].
**2.4.** *Application: Validation of an avionics DSL.*
I carried out the validation of a functional architecture modeling language of avionics systems developed in Trans-IMA project [Hor+14].

**Added value.** The main added value of the approach is to cover rich DSL constructs such as derived features and well-formedness constraints captured in declarative languages such as graph patterns and OCL invariants (**Ch1: Encoding**). While other approaches use bounded verification or simply ignore unsupported features, I proposed approximations to transform language features into a decidable fragment of first-order logic (called effectively propositional logic [PMB08; GM09]), and to handle language features which cannot be represented in FOL. Therefore, the correctness of a DSL can be proved using our method, while others only can detect errors (**Ch2: Abstraction**).

When an output model is derived as a witness or counterexample, this model is back-annotated to the DSL tool itself so that language engineers could observe the source of the problem in their custom language without the need for theorem proving skills.

**Related contributions.** The case study for language level validation was originally developed in the Trans-IMA project [Hor+14] where Ákos Horváth was the technical lead. The mapping technique of OCL constraint to first order logic [J2] is developed by Ágnes Barta.

Iterative model generation for modeling tools

In my third group of contributions, I developed automated test generation (to address **Research Question 2**) and view synchronization (to address **Research Question 3**) techniques using underlying model generators.

**Contribution group 3.** I proposed iterative techniques for generating a diverse set of input models with increasing model size and source model candidates using the output of logic solvers.

**3.1.** *Iterative and Incremental Model Generation.*
I elaborated a decomposition technique for instance models in order to specify a partial solutions for model generation using partial modeling, and metamodel pruning. I proposed an iterative workflow to incrementally generate instance models of increasing size [C10].
**3.2.** *Diversity and Distance Metrics.*
I proposed a distance metric based on graph shapes [Ren04; RD06] for measuring the diversity of a single model and set of models. I showed correlation between diversity and mutation score in using such models for mutation testing. I proposed an iterative technique for the generation of a diverse set of models using this distance metric [C11][J3].
**3.3.** *Incremental Synthesis for Bidirectional Transformations of View Models.*
I transformed query-based view specification into logic formulae to automatically synthesize possible source model changes consistent to a view model change [C12][C13][C7].
**3.4.** *Applications of Results.*
I demonstrated the applicability of the view model maintenance approach in the context of healthcare models developed in the Concerto ARTEMIS project [Con]. I successfully derived a diverse set of statechart models for the industrial Yakindu Statechart Modeling tool [Yak].

**Added value.** Incremental generation simultaneously improves the quality and size of models created by logic solvers. First, significantly larger model instances can be generated with the same solvers using iterative model generation technique (**Ch3: Scalability**). Furthermore, the diversity metrics based on neighborhood shapes [RD06] generalizes existing metrics (such as metamodel coverage and graph isomorphism used in many research papers). Moreover, our model generation technique derives a structurally diverse set of models by calculating the shape of the previous solutions and avoiding those shapes in the next generated model (**Ch4: Diversity**). Finally, incremental model generation also enables to deduce valid source candidates in case of multiple view models with favorable scalability (**Ch3: Scalability**).

**Additional applications and related contributions.** In R3-COP Artemis project [R3c], we carried out test environment generation for autonomous laser guided forklift robots. In collaboration with Ágnes Barta, Zoltán Szatmári and István Majzik we developed a test track generation technique that combines two solvers (Alloy [TJ07] with Sat4j [LBP10] and Z3 [MB08] as backend solvers) in multiple iterative steps to create complex test rooms [Sza16; HMM13].

The forward transformation approach [Deb+14; Gho+15] used in the proposed bidirectional synchronization is developed by Csaba Debreceni and Zoltán Ujhelyi. Publications [C12][C13] introducing backward change propagation are shared contributions with Csaba Debreceni. My contribution was the mapping of view models to logic using a selected part of the model that needs to change. The impact analysis responsible for selecting the changing part is the contribution of Csaba Debreceni. [C7] presents another view modeling technique using 3-valued partial model as background framework.

## 4.4 Future work

Our long-term research goal is to *unify and develop automated graph model generation techniques*, which are currently used in several independent research areas. To unify those research lines in [B14] we outlined the properties of an ideal model generator family which is abbreviated as CoRe-DiSc:

**Consistent**: A model generator is consistent, if it derives well-formed models (soundness), and able to derive all well-formed models for a given scope (completeness). In this classification, the main focus of this thesis was consistent model generation [C6][J2].

**Realistic**: A model generator is realistic if it is able to create models that are close to real ones with respect to some metrics. While several graph metrics have been proposed [Ber+13; BNL14; NL15; Izs+13], the characterization of realistic models is a major challenge [Szá+16].

**Diverse**: A model generator is diverse if it is able to guarantee given a difference between models with respect to a designated distance metrics. As secondary objective of my work, I proposed neighborhood-based distance metrics in [C11].

**Scalable**: A model generator is scalable if it is able to create large models in proportional time. Existing benchmarks are typically [Szá+17; Cps] scalable.

Currently, existing model generation approaches developed in different research areas usually support one (or rarely at most two) of these properties. The grand challenge of CoRe-DiSc is to develop an automated model generator which simultaneously satisfies multiple (ideally, all four) properties.

## Publications

| | |
|---|---|
| Number of publications: | 17 |
| Number of peer-reviewed journal papers (written in English): | 2 |
| Number of articles in journals indexed by WoS or Scopus: | 2 |
| Number of publications (in English) with at least 50% contribution: | 5 |
| Number of peer-reviewed publications: | 17 |
| Number of independent citations (26 March 2019): | 53 |

## Publications linked to the theses

| | Journal papers | International conference and workshop papers | Book chapters | Extended abstracts |
|---|---|---|---|---|
| **Thesis 1** | [J1] | [C4]·[C5]·[C6]·[C7]·[C8] | [B14] | [A15] |
| **Thesis 2** | [J2] | [C4]·[C9] | — | — |
| **Thesis 3** | [J3]* | [C4]·[C10]·[C11]·[C12]·[C13] | [B14] | — |

* These publications are currently under revision.
This classification follows the faculty's Ph.D. publication score system.

### Journal Papers

[J1] **O. Semeráth** and D. Varró. Evaluating Well-Formedness Constraints on Incomplete Models. *Acta Cybernetica* 23(2), 2017, pp. 687–713

[J2] **O. Semeráth**, Á. Barta, Á. Horváth, Z. Szatmári, D. Varró. Formal validation of domain-specific languages with derived features and well-formedness constraints. *Software and System Modeling* 16(2), 2017, pp. 357–392

[J3] **O. Semeráth**, R. Farkas, G. Bergmann, D. Varró. Diversity of Graph Models and Graph Generators in Mutation Testing. *International Journal on Software Tools for Technology Transfer (STTT)*, 2019
▹ *Under revision*

### International Conference and Workshop Papers

[C4] **O. Semeráth**. Formal Validation and Model Synthesis for Domain-specific Languages by Logic Solvers. In: *Proceedings of the ACM Student*

*Research Competition at MODELS 2016 co-located with the 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016), St. Malo, France, October 3-4, 2016.* 2016

[C5] **O. Semeráth** and D. Varró. Graph Constraint Evaluation over Partial Models by Constraint Rewriting. In: *Theory and Practice of Model Transformation - 10th International Conference, ICMT 2017, Held as Part of STAF 2017, Marburg, Germany, July 17-18, 2017, Proceedings*, pp. 138–154. 2017

[C6] **O. Semeráth**, A. S. Nagy, and D. Varró. A Graph Solver for the Automated Generation of Consistent Domain-Specific Models. In: *40th International Conference on Software Engineering (ICSE 2018)*, Gothenburg, Sweden: ACM, 2018

[C7] K. Marussy, **O. Semeráth**, and D. Varró. Incremental View Model Synchronization using Partial Models. In: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, Copenhagen, Denmark, October 14-19, 2018*, pp. 323–333. 2018

[C8] **O. Semeráth**, A. A. Babikian, S. Pilarski, D. Varró. VIATRA Solver: A Framework for the Automated Generation of Consistent Domain-Specific Models. In: *Proceedings of the 41th International Conference on Software Engineering: Demonstrations*, 2019

[C9] **O. Semeráth**, Á. Horváth, and D. Varró. Validation of Derived Features and Well-Formedness Constraints in DSLs - by Mapping Graph Queries to an SMT-Solver. In: *Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings*, pp. 538–554. 2013
▹ *IEEE/ACM Best Paper Award*

[C10] **O. Semeráth**, A. Vörös, and D. Varró. Iterative and Incremental Model Generation by Logic Solvers. In: *Fundamental Approaches to Software Engineering - 19th International Conference, FASE 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, pp. 87–103. 2016

[C11] **O. Semeráth** and D. Varró. Iterative Generation of Diverse Models for Testing Specifications of DSL Tools. In: *21st International Conference on Fundamental Approaches to Software Engineering (FASE)*, Thessaloniki, Greece: Springer, 2018

[C12] **O. Semeráth**, C. Debreceni, Ákos Horváth, D. Varró. Change Propagation of View Models by Logic Synthesis using SAT solvers. In: *Proceedings of the 5th International Workshop on Bidirectional Transformations, Bx 2016, co-located with The European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 8, 2016.* Pp. 40–44. 2016

[C13] **O. Semeráth**, C. Debreceni, Ákos Horváth, D. Varró. Incremental backward change propagation of view models by logic solvers. In: *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016*, pp. 306–316. 2016

### Book Chapters

[B14] D. Varró, **O. Semeráth**, G. Szárnyas, Á. Horváth. Towards the Automated Generation of Consistent, Diverse, Scalable and Realistic Graph Models. In: *Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig*, pp. 285–312. 2018

### Extended Abstracts

[A15] **O. Semeráth** and D. Varró. Validation of Well-formedness Constraints on Uncertain Models. In: *THE 10TH JUBILEE CONFERENCE OF PHD STUDENTS IN COMPUTER SCIENCE*, Szeged, Hungary, 2016

## Additional publications (not linked to theses)

### International Conference and Workshop Papers

[C16] G. Szárnyas, **O. Semeráth**, B. Izsó, C. Debreceni, Á. Hegedüs, Z. Ujhelyi, G. Bergmann. Movie Database Case: An EMF-IncQuery Solution. In: *Proceedings of the 7th Transformation Tool Contest part of the Software Technologies: Applications and Foundations (STAF 2014) federation of conferences, York, United Kingdom, July 25, 2014.* Pp. 103–115. 2014

[C17] Z. Micskei, R. Konnerth, B. Horváth, **O. Semeráth**, A. Vörös, D. Varró. On Open Source Tools for Behavioral Modeling and Analysis with fUML and Alf. In: *Proceedings of the 1st Workshop on Open Source Software for Model Driven Engineering co-located with ACM/IEEE 17th International Conference on Model Driven Engineering Languages & Systems, OSS4MDE@MoDELS 2014, Valencia, Spain, September 28, 2014.* Pp. 31–41. 2014

[C18] G. Szárnyas, **O. Semeráth**, I. Ráth, D. Varró. The TTC 2015 Train Benchmark Case for Incremental Model Validation. In: *Proceedings of the 8th Transformation Tool Contest, a part of the Software Technologies: Applications and Foundations (STAF 2015) federation of conferences, L'Aquila, Italy, July 24, 2015.* Pp. 129–141. 2015

## References

[Abd+14] H. Abdeen, D. Varró, H. A. Sahraoui, et al. Multi-objective optimization in rule-based design space exploration. In: *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*, pp. 289–300. 2014.

[Bag+17] G. Bagan, A. Bonifati, R. Ciucanu, et al. gMark: schema-driven generation of graphs and queries. *IEEE Transactions on Knowledge and Data Engineering* 29(4), 2017, pp. 856–869.

[Ber+08] G. Bergmann, Á. Horváth, I. Ráth, et al. A benchmark evaluation of incremental pattern matching in graph transformation. In: *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*, pp. 396–410. 2008.

[Ber+10] G. Bergmann, Á. Horváth, I. Ráth, et al. Incremental Evaluation of Model Queries over EMF Models. In: *MODELS'10*, LNCS, vol. 6395, Springer, 2010.

[Ber+11] G. Bergmann, Z. Ujhelyi, I. Ráth, et al. A graph query language for emf models. In: J. Cabot and E. Visser (eds.), *Fourth International Conference on Theory and Practice of Model Transformations*, LNCS, vol. 6707, pp. 167–182. Springer, 2011.

[Ber+13] M. Berlingerio et al. Multidimensional networks: foundations of structural analysis. *World Wide Web* 16(5-6), 2013, pp. 567–593.

[BNL14] F. Battiston, V. Nicosia, and V. Latora. Structural measures for multiplex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 89(3), 2014.

[Bro+06] E. Brottier, F. Fleurey, J. Steel, et al. Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool. In: *17th International Symposium on Software Reliability Engineering, 2006. ISSRE '06.* Pp. 85–94. 2006.

[Con] CONCERTO ARTEMIS project. `concerto-project.org/`.

[Cps] *CPS Benchmark*. `https://github.com/viatra/viatra-cps-benchmark`. The Eclipse Project. 2017.

[Deb+14] C. Debreceni, Á. Horváth, Á. Hegedüs, et al. Query-driven incremental synchronization of view models. In: *Proceedings of the 2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, p. 31. 2014.

[DMB08] L. De Moura and N. Bjørner. Z3: an efficient SMT solver. In: *Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems*, TACAS'08/ETAPS'08, pp. 337–340. Springer-Verlag, 2008.

[Emf] *Eclipse Modeling Framework*. `http://www.eclipse.org/emf`. The Eclipse Project.

[ES03] N. Eén and N. Sörensson. An extensible sat-solver. In: *International conference on theory and applications of satisfiability testing*, pp. 502–518. 2003.

[Gho+15] H. Gholizadeh, Z. Diskin, S. Kokaly, et al. Analysis of source-to-target model transformations in quest. In: *Proceedings of the 4th Workshop on the Analysis of Model Transformations*, pp. 46–55. 2015.

[GM09] Y. Ge and L. Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: A. Bouajjani and O. Maler (eds.), *Computer Aided Verification*, LNCS, vol. 5643, pp. 306–320. Springer Berlin Heidelberg, 2009.

[Hay+01] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, et al. A practical tutorial on modified condition/decision coverage, 2001.

[HHV15] Á. Hegedüs, Á. Horváth, and D. Varró. A model-driven framework for guided design space exploration. *Automated Software Engineering* 22(3), 2015, pp. 399–436.

[HMM13] G. Horányi, Z. Micskei, and I. Majzik. Scenario-based automated evaluation of test traces of autonomous systems. In: M. Roy (ed.), *Proceedings of ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems (DECS) at SAFECOMP'13*, pp. 181–192. 2013.

[Hor+14] Á. Horváth, Á. Hegedüs, M. Búr, et al. Hardware-software allocation specification of ima systems for early simulation. In: *Digital Avionics Systems Conference (DASC)*, IEEE, 2014.

[Izs+13] B. Izsó, Z. Szatmári, G. Bergmann, et al. Towards precise metrics for predicting graph query performance. In: *ASE*, pp. 421–431. 2013.

[Jac02] D. Jackson. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.* 11(2), 2002, pp. 256–290.

[JLB11] E. K. Jackson, T. Levendovszky, and D. Balasubramanian. Reasoning about metamodeling with formal specifications and automatic proofs. In: *Model Driven Engineering Languages and Systems*, pp. 653–667. Springer, 2011.

[JS06] E. K. Jackson and J. Sztipanovits. Towards a formal foundation for domain specific modeling languages. In: *Proceedings of the 6th ACM / IEEE Int. Conf. on Embedded Software*, EMSOFT '06, pp. 53–62. ACM, 2006.

[KPP09] D. S. Kolovos, R. F. Paige, and F. A. C. Polack. On the evolution of ocl for capturing structural constraints in modelling languages. In: *Rigorous Methods for Software Construction and Analysis*, pp. 204–218. 2009.

[LBP10] D. Le Berre and A. Parrain. The sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation* 7, 2010, pp. 59–64.

[MB08] L. de Moura and N. Bjørner. Z3: an efficient SMT solver. In: *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference (TACAS 2008)*, LNCS, vol. 4963, pp. 337–340. Springer, 2008.

[MC13] N. Macedo and A. Cunha. Implementing QVT-R bidirectional model transformations using Alloy. In: *Fundamental Approaches to Software Engineering*, pp. 297–311. 2013.

[Mil+07] A. Milicevic, S. Misailovic, D. Marinov, et al. Korat: A tool for generating structurally complex test inputs. In: *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pp. 771–774. 2007.

[MK01] D. Marinov and S. Khurshid. Testera: A novel framework for automated testing of java programs. In: *16th IEEE International Conference on Automated Software Engineering (ASE 2001), 26-29 November 2001, Coronado Island, San Diego, CA, USA*, p. 22. 2001.

[Mou+09] A. Mougenot, A. Darrasse, X. Blanc, et al. Uniform random generation of huge metamodel instances. In: *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications*, ECMDA-FA '09, pp. 130–145. Springer-Verlag, 2009.

[NL15] V. Nicosia and V. Latora. Measuring and modeling correlations in multiplex networks. *Phys. Rev. E* 92, 3 2015, p. 032805.

[PMB08] R. Piskac, L. de Moura, and N. Bjorner. Deciding Effectively Propositional Logic with Equality. Microsoft Research, MSR-TR-2008-181 Technical Report. 2008.

[R3c] R3-COP (Resilient Reasoning Robotic Co-operative Systems). ARTEMIS project n° 100233, `http://http://www.r3-cop.eu/`.

[RD06] A. Rensink and D. Distefano. Abstract graph transformation. *Electronic Notes in Theoretical Computer Science* 157(1), 2006, pp. 39–59.

[Ren04] A. Rensink. Canonical graph shapes. In: *Programming Languages and Systems, 13th European Symposium on Programming, ESOP 2004*, pp. 401–415. 2004.

[RSW04] T. W. Reps, M. Sagiv, and R. Wilhelm. Static program analysis via 3-valued logic. In: *International Conference on Computer Aided Verification*, pp. 15–30. 2004.

[Sza16] Z. Szatmári. Metamodel-based model generation and validation techniques with applications. PhD dissertation. Budapest University of Technology and Economics, 2016.

[Szá+16] G. Szárnyas, Z. Kővári, Á. Salánki, et al. Towards the characterization of realistic models: evaluation of multidisciplinary graph metrics. In: *MODELS*, 2016.

[Szá+17] G. Szárnyas, B. Izsó, I. Ráth, et al. The Train Benchmark: cross-technology performance evaluation of continuous model queries. *Softw. Syst. Model.* 2017.

[Szá19] G. Szárnyas. Query, Analysis, and Benchmarking Techniques for Evolving Property Graphs of Software Systems. PhD thesis. Budapest University of Technology and Economics, 2019.

[TJ07] E. Torlak and D. Jackson. Kodkod: a relational model finder. In: *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 632–647. Springer, 2007.

[Vs] *Viatra Solver*. 2018.

[VSV05] G. Varró, A. Schürr, and D. Varró. Benchmarking for graph transformation. In: *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2005), 21-24 September 2005, Dallas, TX, USA*, pp. 79–88. 2005.

[WHR14] J. Whittle, J. Hutchinson, and M. Rouncefield. The state of practice in model-driven engineering. *IEEE software* 31(3), 2014, pp. 79–85.

[Wil12] E. D. Willink. An extensible OCL virtual machine and code generator. In: *Proc. of the 12th Workshop on OCL and Textual Modelling*, pp. 13–18. ACM, 2012.

[Yak] *Yakindu*. `http://statecharts.org/`. Yakindu Statechart Tools. 2017.