ISSN 2186-7437

NII Shonan Meeting Report

No. 2018-16

Towards industrial application of advanced formal methods for cyber-physical system engineering

Fuyuki Ishikawa
Alexander Romanovsky
Thierry Lecomte

November 5–8, 2018

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Towards industrial application of advanced formal methods for cyber-physical system engineering

Organizers:
Fuyuki Ishikawa (National Institute of Informatics, Japan)
Alexander Romanovsky (Newcastle University, UK)
Thierry Lecomte (ClearSy, France)

November 5–8, 2018

Software systems plays an increasingly critical role in the society, as well as in the physical world, as being further investigated in the emerging paradigms such as Cyber-Physical Systems (CPS), Internet of Things (IoT), and Smart City/Office/Home. The demand for dependability is very strong in such systems, while the difficulties in its assurance is increasing more than ever, for example, due to the unprecedented complexity of the systems and their environments. Adequate theoretical foundations – specifically based on mathematics – are instrumental for tackling the difficulties when they are elaborated into systematic methods with engineering disciplines, i.e., formal methods.

At the same time, there have been serious discussions on how formal methods should be or can be introduced and leveraged in the industry [1]. Myths are often cited, representing typical kinds of "misunderstanding" such as the need for the user to possess extreme mathematical skills [2-4]. Such perceptions, true or false, can create a sense of doubt, distance, or even antipathy against formal methods. Even if engineers acknowledge the value of formal methods, there are obstacles in their effective and efficient applications. Typical obstacles include the initial costs of investment and education, the difficulties in tailoring a method for the target projects, and difficulties in convincing the stakeholders (including, the managers) in terms of various aspects such as cost-efficiency. It is worth noting that the discussion of these issues is not limited to a specific world of formal methods – which often provides useful exemplification of very general issues, including: what are knowledge and skills necessary in software engineering, how technology from the academia can be perceived and (un)accepted by the industry, and so on.

Today we have further drivers that motivate an active discussion of these issues. First, the increasing difficulties of the emergent systems are making it necessary to rethink the existing approaches to dependability assurance, including formal methods. Second, the recent technical advances are changing feasibility and usages of formal methods, e.g., enhancement of SAT/SMT solvers, as well as the infrastructure (GPU, clouds, etc.). Finally, most importantly, the industry has recently been achieving clear successes in the advanced use of formal methods. ClearSy showed the feasibility of correct-by-construction approach (obtaining the already-verified code) in their worldwide business [5]. Amazon showed how formal methods spread from one team to various teams

[6]. Facebook embedded a formal verification tool into its agile culture [7]. Sony FeliCa established use of formal specification as readable documents [8]. Similar examples can be found by ARM, RATP, Siemens, etc.

To address the issues above we organized a Shonan Meeting to discuss the issues around the application of advanced formal methods in industry. We aimed at promoting further discussions on the classical but essential issues, while timely responding to the rapid changes around software-intensive systems including the movement towards CPS. This meeting invited leading researchers and practitioners who are actively working in the industry or in the academia-industry collaboration.

The meeting scheduled two kinds of sessions as well as a special session described below. One was presentations and targeted discussions, where each participant presented ideas and positions that then kicks off various directions of discussions. The other sessions consisted of intensive follow-up discussions involving mixed groups of participants. The topics were discovered at the meeting.

As many Japanese companies are interested in leveraging formal methods, we also had a special session involving participants from these companies -not only the listed invitees for attendance through the week but more invitees who could only attend half or one day without stay.

We also had two events at NII after the Shonan Meeting.

- Symposium on Dependability and Safety of Autonomous Systems (`http://research.nii.ac.jp/dsas/`)
- Event-B Day 2018 in Tokyo (`http://research.nii.ac.jp/eventb2018/`)

[1] D. Bjørner et al. : 40 Years of Formal Methods – Some Obstacles and Some Possibilities?, FM 2014. Singapore.
[2] A. Hall : Seven Myths of Formal Methods, IEEE Software, Vol.7 No.5, 1990
[3] J. P. Bower et al. : Seven More Myths of Formal Methods, IEEE Software, Vol.12 No.4, 1995
[4] L. Maccherone et al. : Software Mythbusters Explore Formal Methods, IEEE Software, Vol.26 Nov.6, 2009
[5] T. Lecomte, Industrial use of Formal Methods for Safe and Secure Systems, Invited Talk. ISSRE 2016. Ottawa. Canada
[6] C. Newcombe et al. : How Amazon Web Services Uses Formal Methods, Communications of the ACM, Vol.58 No.4, 2015
[7] C. Calcagno et al. : Moving Fast with Software Verification, NFM 2014
[8] T. Kurita et al. : Practices for Formal Models as Documents: Evolution of VDM Application to "Mobile FeliCa" IC Chip Firmware, FM 2015

1 Overview of Talks

Practically Formal Methods & Tools for CPS

Alan Wassyng, McMaster University, Canada

Mathematics is one of the most powerful tools we have at our disposal. However, when we use mathematics we can still be wrong; we can still be ambiguous; and we can still be incomplete. Even worse, when the mathematics is correct, unambiguous and complete, it could still be misunderstood. So, while mathematics is useful, assuming that its use solves all our problems is wrong. What should we be promoting? Practical: Our methods and tools have to be usable on real world problems. They have to be usable by the people solving those real world problems. Formal: Powerful methods (and tools) based on sound mathematics, that should be more capable and more efficient than ad hoc solutions. PracticalLY (the real focus of the talk): There are numerous situations where we know we cannot be formal, or completely formal. It is important to realize that "pretending" we can solve the problem completely formally can suggest what we need to try to achieve, even if we cannot achieve complete formality. This talk presented some examples of how building methods and tools based on the direction we could take if everything was described completely formally can help in the quest to assure safety. The talk showed a comparison between GSN and a new approach we are developing for assurance cases. It highlighted the fact that GSN-like assurance is, ad hoc in the way that the case is argued. Accepted practice and patterns are predominantly important. The new approach is based on how we may want to assure the safety of systems if the development process and the system itself were formally modelled. These assurance steps can then guide our practical assurance methods and tools. The steps are based on what we would want to do if everything was completely formal. It is not ad hoc, and holds the promise of being more consistent, repeatable and thorough.

An early foundation for the new assurance method was presented recently at MODELS 2018. Zinovy Diskin, Tom Maibaum, Alan Wassyng, Stephen Wynn-Williams, and Mark Lawford. 2018. "Assurance via model transformations and their hierarchical refinement." In Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '18). ACM, New York, NY, USA, 426-436. DOI: `https://doi.org/10.1145/3239372.3239413`

Safecap Signalling Verification: how we talk to signalling engineers

Alexei Iliasov, Newcastle University, United Kingdom

Effective signalling is essential to the safe and efficient operation of a railway network. It enables trains to travel at high speeds, run close together, and serve multiple destinations. Whether by mechanical semaphores, colour lights or electronic messages, signalling only allows trains to move when it is safe for them to do so. Signalling locks moveable infrastructure, such as the points that form railway junctions, before trains travel over it. Furthermore, signalling often actively prevents trains travelling further or faster than it is safe and sometimes even drives the trains. At the heart of any signalling system there are one or more *interlockings*. These devices constrain authorisation of train movements as well as movements of the infrastructure to prevent unsafe situations arising.

Safecap platform is a framework for the design and analysis of railway signalling. It combines a number of tools such as a schema editor, a number of simulators of varying fidelity and, finally and most prominently, tools for formal verification of control tables and signalling data. Safecap has its own verification stack developed to replace abortive efforts to employ Event-B/Rodin in specification of safety principles and their subsequent verification against tables and data.

Safecap team has been recently involved in industry funded projects on verification of real-world railway interlocking data and software. This has brought a number of new challenges: scaling up the verification back end, making the tool user friendly enough to be applied by a non-expert, dealing with many inconsistencies of input data, catering for verification constraints with many exceptions and side conditions.

In order to make the tool applicable to an industrial data set we had to develop a custom verification back end with own solver and symbolic prover. These work in a conjunction with several external tools that provide an extra assurance. The built in solver and prover are much faster though and can be configured with custom rules to assist in the proofs of specific correctness criteria. They are also capable of reporting the state of a failed proof on the way that facilitates witness identification.

To make the tool usable in an industrial setting, Safecap hides any formal method specifics and reports verification errors in the terms of problem domain. This is a non trivial exercise as verification in general and identification of causes of a failed condition are two distinct issues. We approach this by separating identification of useful witnesses from verification conjecture and associate witness solving with a mapping into the problem domain and error reporting templates.

The reality of a large scale application of formal methods is that one have to reconcile formal, semi-formal and informal descriptions within same project. It is unfeasible to create a completely formal definition of the whole railway domain: the cost is prohibitive and it is likely there is no single formalism expressive enough to capture the whole range of concerns. A large scale model can only be created incrementally and it would be best if it is useful in even in a partial state of development. Much effort went into enabling incremental formalisation and especially expressly identifying the limitations and boundaries of an existing formal model as to avoid false reassurance stemming from inadequate formalisation. As a technical solution we apply mutation testing to identify parts of input data not sufficiently covered by verification constraints.

Formal correctness criteria are not easy to write. One needs to have a good grasp of the problem domain and understanding how all the relevant system artefacts are represented in inputs to verification. Whereas traditional testing works by defining negative scenarios that unfold in time, static verification employed in Safecap requires all encompassing statements of what is correct. Such statements are necessarily quite complex and we have found that the standard logic is not assisting in breaking down such statements into smaller pieces. We have thus switched to so call non-monotonic logic where, unlike standard logic, extra hypothesis may invalidate a conjecture. This is useful to account for foreseen but yet unformalised exceptions to verification constraints.

Security Issues for Smart Factories (Industry 4.0)

Carolyn Talcott, SRI International, USA
Vivek Nigam, fortiss GmbH, Germany

According to Wikipedia, "Industry 4.0" is a name given to the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of things, cloud computing and cognitive computing. Industry 4.0 is commonly referred to as the fourth industrial revolution.

4Diac is a tool used by our colleages at fortiss for design of Industry 4.0 industrial automation/control applications based on the IEC 61499 standard. Applications are composed of function blocks that interact via event and data channels. Function blocks are mapped onto available devices which are grouped in subsystems that form distributed networks.

Security is an increasing concern as applications become more modular, networked, open, and incorporate off-the-shelf IoT devices. Security failures can cause safety failures.

In this talk we consider how formal methods can help to produce more secure designs and make the design-deploy-test cycle more efficient. Our approach is to develop formal executable models (using the rewriting logic system Maude). With such models one can execute sample scenarios (formal simulation/testing), as well as carrying out reachability analysis (simple model-checking). Resource consumption and timing properties can be modeled as well basic functionality. The basic model can be composed with attack models to evaluate counter measures. Symbolic analysis can be used to manage state space exploration.

In the talk we illustrate these possibilities with a simple running example called 'pickNPlace' where an arm moves on a track picking up an object at one end, and putting it down at the other end.

Security is not free. The benefit of benefit of counter measures (increased security) versus cost (for example of encryption) in latency. The choice amongst design alternatives will likely make a difference in the tradeoff balance. Thus it is important to consider security at design time. The proposed approach based on formal executable models can help automatically enumerate possibilities and evalute tradeoffs by determining the cost of securing different designs.

There are many issues to consider to realize the potential of the formal modeling approach. What are practical attack models? Dolev-Yao is too heavy weight. Since we are interested in cyber-physical systems, physical attacks must also be considered. Are there defense mechanisms that can be realized by suitable network designs? What can be done at the switch/firewall/port level? When is crypto necessary? How weak can it be? From a theoretical perspective, smart factories open up a whole new class of decision problems and complexity measures to study.

AdaHorn: A Tool for Verification of Ada Programs

Christian Herrera, Tewodros A. Beyene, Vivek Nigam, fortiss GmbH, Germany

Ada is a programming language widely used by the avionics, space, military, railways, among other communities of systems developers because its features, e.g. extremely strong typing, explicit concurrency, built-in language support for design-by-contract and non-determinism, allow developers to build robust and dependable safety critical systems. Two prominent tools that support the development of such systems are GNATProve and Polyspace. Those tools perform static analysis by abstract interpretation on Ada programs for verifying runtime errors, e.g. arrays out-of-bounds, arithmetic overflows, division by zero, etc. Abstract interpretation is a technique that uses overapproximations, e.g. on the set of possible program execution traces, or on the domains of the variables occurring in verified programs. It is well known that tools based on this technique often yield false positives, i.e. results wrongly indicating that errors in programs occur, and false negatives, i.e. results wrongly indicating that programs do not have errors. Due to the undecidability of the problem of proving any non-trivial property about a computer program, for tools using abstract interpretation it is impossible to entirely avoid both false positives and false negatives for all possible input programs.

Model checking is a technique that can provide conclusive results with respect to the conditions that hold or do not hold in programs. In this work we aim to complement the support available for the development of those systems by proposing AdaHorn, a model checker for verifying Ada programs with respect to correctness properties. AdaHorn translates Ada programs together with their related correctness properties into a set of Constrained Horn Clauses (CHC) which are solved by well-known solvers such as Eldarica and Z3.

Similar to the tools SeaHorn and JayHorn that respectively verify C and Java programs, the design of AdaHorn consists of three main modules: (1) Front-End Module which enables translating Ada programs into an intermediate representation, namely an XML serialization of an Abstract Syntax Tree, that allows program constructs, e.g. data types, procedures, functions, loops, etc., to be further translated into a suitable intermediate logic language, namely CHC, in order to express system's behaviour, (2) CHC Generator which performs the translation of Ada programs together with their related correctness properties into CHC and, (3) Back-End Module which uses the mentioned solvers to solve generated CHC. All three modules are easy to extend, and specially the Front-End Module and the Back-End Module allow an easy replacement of their underlying tools.

AdaHorn currently supports a small but useful subset of program constructs, e.g. integer, floating-point and boolean data types, loops, assertions, conditionals, arrays, procedures and functions. With this subset of constructs we are able to propose a set of Ada programs inspired by C programs from the competition SV-COMP 2017, and use them to compare the effectiveness of AdaHorn and GNATProve. Our experiments show that AdaHorn outperforms GNATProve by yielding correct results in more cases.

The contribution of this work is twofold, namely (1) we propose AdaHorn which to the best of our knowledge is the first CHC-based model checker for Ada programs, and which yields correct results in more cases than GNATProve. In our experiments GNATProve outputs false positives and false negatives. AdaHorn does not output false negatives, but does output false positives in very concrete cases, namely when using floating-point data types, these cases are related to the supported numerical precision of the solvers (Eldarica and Z3) used by AdaHorn. That precision differs from the one assumed by the compiler of Ada programs and, (2) we propose a useful set of Ada programs inspired by C programs from the competition SV-COMP 2017. These Ada programs can pave the way for extending the SV-COMP competition for Ada verification tools.

FASTEN: A Formal Specification Environment Targeting Practicing Engineers

Daniel Ratiu, Siemens, Germany

Todays software intensive systems have high dependability requirements since their malfunctioning might cause high costs. When classified as safety critical, the vast majority of systems functions are at lower criticality level (i.e. IEC61508 SIL 1 or SIL 2) and thereby existing standards do not mandate the use of formal methods for their development. Developers use agile processes, are part of distributed teams of engineers, work under budget constraints and high pressure due to short time to market requirements. Furthermore, the development is usually done by domain experts without background in formal methods. In this context formal methods are regarded as highly exotic tools and practitioners shy away in front of their (perceived) complexity.

In this presentation we introduce FASTEN, an open source formal specification environment targeted towards practicing engineers. FASTEN features a stack of domain specific languages (DSLs) built on top of SMV, the input language of the NuSMV model checker. The DSLs capture at language level typical usages (e.g. patterns, idioms) of the SMV language or higher level abstractions developed by the formal methods community. The DSLs of FASTEN support different notations like textual, tabular or diagrammatic and specification approaches like function tables, state-machines or contracts-based design. FASTEN features also a set of DSLs to describe verification conditions for components and bridge the gap between classical testing and formal verification. FASTEN promotes a multi-paradigm modelling approach and allows the engineers to use the most appropriate abstractions and notation for the task at hand. Whenever appropriate DSLs can be used, the models can be written in a more compact fashion and are more explicit and thereby simple to write and understand by users. If no adequate abstraction is available for the specific modeling problem then plain SMV can be used. From models we generate textual files containing the corresponding SMV code and these files are inputs for the NuSMV model checker. Verification results are read from the output of NuSMV and are lifted in FASTEN; the counterexample trace provided by NuSMV when a property fails can be replayed in FASTEN and the values corresponding to variables are displayed in the editor.

The set of DSLs provided by FASTEN is modular, extensible and forms an open stack. From an architectural point of view, FASTEN aims to be a framework to experiment with developing language abstractions amenable for formal methods for domain specific problems. FASTEN is developed using the JetBrains' Meta-Programming System (MPS) language workbench. MPS allows the development of modular and extensible languages which can be separately deployed and enabled by the user for the task at hand. All in all, MPS supports the definition of comprehensive domain specific modeling environments with advanced support for editing the models, generation of code and integration with external tools.

FASTEN home: `https://sites.google.com/site/fastenroot/`

Safety of autonomous systems: multi-layered perspective

Elena Troubitsyna, KTH, Sweden

Autonomous systems are capable of delivering their services in a highly independent way, i.e., without human supervision. Currently the autonomous systems, such as self-driving cars, drones etc. are getting ready for real-life deployment. Typically, autonomous systems rely on machine learning, e.g., to realise computer vision, and hence, the system behaviour continuously evolves. This makes the exhaustive pre-deployment verification of autonomous systems unfeasible. Therefore, to achieve safety of autonomous systems, we should rely on a combination of pre-deployment and run-time safety-monitoring mechanisms.

I discuss a novel multi-layered architecture to ensuring safety of autonomous systems. The main idea is to guarantee safety of autonomous systems by combining "safety-maximising" mission planning "strategic" safety, proactive runtime safety maintenance "tactical" safety and emergency response aiming at mitigating or removing hazard "active" safety. Each type of safety mechanisms is implemented at the corresponding architectural layer.

While designing autonomous systems, typically we should trade off safety with other system characteristics, such as performance, quality of service, etc. For instance, let us consider a swarm of drones. To accomplish the required mission, the drones should have a high probability of surviving throughout the entire mission, i.e., they should not prematurely deplete their batteries. This requires minimisation of energy consuming functions, e.g., travel distance, communication etc. However, such a minimisation might result in planning a mission that has a high risk of drone collision. In this context,strategic safety can be implemented by an optimisation algorithm that uses safety, i.e., the risk of collisions, and the travel distance as the parameters of the optimisation. Strategic safety relies on high performance optimisation algorithms and hence can be implemented in cloud.

When the swarm starts to move, the drones can deviate from the predefined routes due to unforeseen environmental factors, e.g., wind, or internal problems, e.g., transient hardware failures of drones. The positions of the drones should be continuously monitored and their routes recalculated to maintain efficiency and safe distance between the drones. Such activities are called "tactical" safety. Tactical safety can be implemented at the edge.

Finally, an unexpected obstacle might appear in the flight zone of the swarm. To avoid a collision, the "active" safety – an emergency stopping or safe maneuver should be executed. The active safety requires fast response and should be implemented locally, at the drone level.

Such a multi-layered distributed implementation of the safety mechanisms requires a complex coordination scheme that cater to variety of situations: drone and communication failures, changes in the external environment such as unexpected appearance of obstacles etc. To design the proposed multi-layered safety architecture, we rely on formal modelling in Event-B. We aim at deriving and verifying the communication protocol ensuring that safe decisions can be made at each layer at abstraction.

Toward Developing Machine Learning Based Systems with Formal Methods (A Preliminary Work)

Hironobu Kuruma, Hitachi, Ltd., Japan

Machine learning is expected to be an effective way to develop components that handle uncertainties of the real world. However, the developed components are in general difficult to interpret and/or predict their behaviors for humans. Our question is that can we use machine learning based components (AI modules) in safety critical systems? Correctness by construction is a promising approach to develop reliable software. Unfortunately, C by C cannot be applied to develop systems containing AI modules since neural networks are not constructed through step by step refinements and verifications. We are studying a software development process in which development process of AI modules is embedded into C by C process. Our approach is to separate AI module from logical modules at architecture design phase and assure the behavior of the AI module by testing. The abstract specification defined at the architecture design phase provides the AI module a partial specification. Assurance that the AI module conforms to the partial specification is given by use-case testing.

Let us consider an illustrative example: door lock system of a car. The system locks the door according to the velocity of the car. The velocity is controlled by a speed controller which is a machine learning based module and consequently its behavior is unpredictable. The door lock system is composed of a sensor, a controller, and an actuator. The sensor measures the velocity of the car periodically and the controller instruct the actuator "lock" when sensed velocity is greater than or equal to auto_lock_speed. The actuator completes its lock action with a delay. For example, if the door is not locked, the controller instruct actuator to "lock" when the sensed velocity exceeds the auto_lock_speed and the actuator starts lock action, which takes time to complete. If the sensed velocity falls below the auto_lock_speed before the completion of the lock action, the controller cancels "lock" instruction and the actuator cancels the lock action. And when sensed velocity exceeds the auto_lock_speed again, the controller instructs "lock" and the actuator starts lock action. After a delay of the actuator's action, the door is locked. A safety requirement is that the door should always be locked when sensed velocity is higher than a fixed value, even if the velocity changes unpredictably while the actuator is acting.

To model the door lock system in Event-B, we make an assumption on the speed control module: the maximum value of velocity difference within 1 sensing period is dv, i.e. if the current sensor value is v, the next sensor value v' is max(0, v-dv) ¡= v' ¡= v + dv. With this assumption, we modeled the door lock system in Event-B and verified that when sensed velocity v is greater than or equal to auto_lock_speed + dv + a_delay * dv then the door is always locked. Where, a_delay is the time interval between the beginning and the completion of actuator's locking action measured by the sensor's sensing period. In use-case testing, we test the speed control module with respect to the assumption. Assuming that the speed control module recognizes traffic signs and decides car speed, test input data are sequences of traffic signs arranged in accordance with use-case scenarios. Since the assumption is a partial specification of the speed control module, it defines only a range of expected output value of the speed control module. The use-case testing is performed by using the partial specification as a pseudo oracle, i.e. the speed control module passes the use-case testing if its output value is always in the ranges of expected output value for each input value.

We are studying stepwise refinement approach to develop software containing machine learning based component. In our study, the function of AI module is identified and represented non-deterministically in an abstract specification. The rest part of the abstract specification is refined and verified into logical software modules. Non-deterministic specification of the AI module provides specifications of logical modules with an assumption on the AI module for verification. It is also used as a pseudo test oracle in the conformance testing of the AI module.

References [1] H. Kuruma and S. Nakajima: Modeling and behavior analysis case-study of a time dependent system in Event-B, IPSJ Journal, 57(8), pp.1690 - 1702, 2016 (in Japanese). [2] H. Kuruma, T. Myojin, N. Sato, Y. Nakagawa and H. Ogawa: Formal Verification and Testing of Software Including Components Constructed Using Machine Learning, Proc. IPSJ/SIGSE Winter Workshop 2018, pp.6 - 7, 2018 (in Japanese).

Reuse of Proofs for Verifying Declarative Cloud Automation

Hiroyuki Yoshida, Japan Systems Co.Ltd, Japan

The correctness of cloud automation becomes a crucial problem of the society. We propose a reuse method of proofs for verifying desired properties of declarative specifications of cloud automation.

Theorem Proving can verify models of arbitrary many number of states and is suitable for proving absence of counter examples. However, it requires many human efforts to interactively supply many lemmas and write many lines of codes. Based on our experiences to apply theorem proving to verification of cloud automation, we found that many of lemmas and their proof codes are similar to each other; reuse of proofs is expected to reduce human efforts of proof development.

The proposed method resembles inheritance of Object Orientation, that is, lemmas proved in abstract modules can be used in inherited concrete modules similarly as methods defined in abstract classes can be used in the subclasses.

Out method is constructed on CafeOBJ system developed by Dr. Futatsugi's group. It is an executable formal specification language system to be used as a specification verification system. While CafeOBJ provides many advanced functionalities, one of them, say Template Modules, enables to define general entities, lemmas, and proofs and to reuse them by instantiating and renaming.

The method provides (1) a general way to formalize different kinds of cloud automation specifications, (2) a procedure for how to verify leads-to and invariant properties of the specifications, (3) model templates and a generic predicate library which are defined in a higher level of abstraction and can be reused as problem specific models and predicates, and (4) many lemmas for the generic predicates which have been already proved in the higher level and are automatically reused in the course of verification.

We applied it to a non-trivial problem, that is, to specify and verify the behavior models of the standard specification language, OASIS TOSCA, of cloud automation It shows that all of 37 sorts in the model are simply defined by reusing the templates and that two thirds of predicates and one third of lemmas are automatically reused.

Two-Layered Falsification of Hybrid Systems guided by Monte Carlo Tree Search

Ichiro Hasuo, National Institute of Informatics, Japan

Based on the paper of the same title, presented at EMSOFT 2018, coauthored with Zhenya Zhang, Gidon Ernst, Sean Sedwards and Paolo Arcaini.

Few real-world hybrid systems are amenable to formal verification, due to their complexity and black box components. Optimization-based falsification—a methodology of search-based testing that employs stochastic optimization—is attracting attention as an alternative quality assurance method. Inspired by the recent works that advocate coverage and exploration in falsification, we introduce a two-layered optimization framework that uses Monte Carlo tree search (MCTS), a popular machine learning technique with solid mathematical and empirical foundations. MCTS is used in the upper layer of our framework; it guides the lower layer of local hill-climbing optimization, thus balancing exploration and exploitation in a disciplined manner.

Using this technical work as an example, the take-home message of the talk is the role of technical ideas developed in formal methods in the quality assurance of cyber-physical systems (CPS). It is known that formal verification—in the sense of giving rigorous proofs for correctness specifications—is hard for CPS due to their complexity and lack of white-box models. This makes us naturally turn to testing as an alternative quality assurance measure. However, in many cases, the technical core of formal verification techniques lies in how to translate a verification problem into some optimization problem (such as SAT, SMT, graph reachability, LP, SDP and so on) that is then solved by a dedicated solver. We can draw an analogous picture in testing, where a testing problem (such as finding an error or optimizing coverage) is translated into some optimization problem. In both pictures, the translation needs to unfold the intricate structure that lies in the original (verification or testing) problem, and it is where we need the power of logic, automata and semantics. Therefore, many techniques and ideas developed in the former picture should be transferable to the latter picture. In the specific instance of the presented technical work, one can discern formal methods ideas in the following aspects: 1) the use of quantitative robustness semantics that turns a logical specification into a quantitative reward; and 2) the definition of the transition structure in which MCTS search is conducted. The latter definition crucially uses the time-causal structure of the falsification problem in question.

Coordination of Deduction and Search in Specification Verification of Transition Systems with CafeOBJ

Kokichi Futatsugi, NII/AIST/JAIST, Japan

CafeOBJ is a state-of-the-art algebraic specification language and is a member of the OBJ language family which also includes Maude. CafeOBJ is an executable specification language system which can be used as a specification verification system.

A CafeOBJ's specification is a description of models, and its verification is to show that the models has some desirable properties by deducing the properties from the specification. Specification verification in this sense is theorem proving where a set of axioms is the specification and theorems are the desirable properties.

The theorem proving capability of CafeOBJ has been significantly enhanced with some achievements including (1) formalization of induction and casesplit based on constructor-based specification calculus and (2) specification/verification of transition systems with conditional transition rules.

A proof score is a description in CafeOBJ of a proof. A specification in CafeOBJ is a set of equations, and CafeOBJ implements equational inference through reduction (i.e. rewriting with the equations by using them as left to right rewrite/reduction rules). For a CafeOBJ's specification (i.e. a specification module) M and a property p on M, expression M $\models$ p means that each model of M satisfies p. If CafeOBJ returns 'true' for a reduction 'reduce in M : p .' (i.e. p reduces true at module M) then M $\models$ p is proved. A reduction that returns true is called effective. A proof score for M $\models$ p is a description of a set of effective reductions (including descriptions of modules that are necessary to execute the reductions) such that the success of the all reductions implies M $\models$ p.

Proof scores can naturally describe several kinds of inductions including structural induction, and quite a few cases are developed until now. Well-founded induction seems to subsume all of inductions used and could have a potential to enhance effectiveness and efficiency of proof scores.

One of the most interesting features of CafeOBJ is smooth coordination of deduction and search. Built-in search predicates provide effective ways to do verifications/simulations by search for cases of small size. PTG (Proof Tree Generation) and RC/RPC (Refinement of Cases through Refinement of Proof Constants) facilities provide effective ways to do deduction (proof tree generation) with systematic search induced by refinement of state configurations (through refinement of proof constants).

Application of Formal Methods to Railway Signalling Software

Laurent Voisin, Systerel, France

Formal methods bring the rigour of mathematics to the analysis, development and verification of systems and software. They provide the means to demonstrate that a design fulfills its specifications and are particularly helpful for the development of critical software-based systems. They also lead to eliciting all the hypotheses (notably on the environment in which the software is to operate) that are needed to prove the software safe.

My presentation reflected back on 20 years of experience of practicing formal methods for railway signalling software. Some of the available formal techniques such as Software B and Event-B allow to design correct-by-construction software and systems. These are a priori techniques. There are also a posteriori techniques such as formal data validation and SAT-based model-checking that can be easier to deploy in an existing development process.

A crucial point is that whatever the technique used, it must be backed by tools such as Atelier B, Rodin, Ovado or Systerel Smart Solver. A formal method without any appropriate tooling does not bring much: one just cannot trust hand-written proofs for software, the domain is too complex and therefore machine-checked proofs are necessary.

The lessons we have learned are that formal techniques can indeed be applied successfully in industry and can be both efficient and effective. They allow to reach a very high level of quality for safety critical systems. The global cost (for critical systems) is not higher than with usual techniques (such as thorough testing) but the quality of the resulting artefact, including its documentation, is higher.

Some criteria for success are to pick the right technique for the problem at hand, to define and follow a consistent and dedicated method for applying the formal technique, and finally to know when to stop formalising (diminishing returns). One also needs to have an expert team at hand to help, as formal techniques are currently much less mainstream than classical programming (no internet search, for instance).

The A720-A10 Interchange Case Study in SysML/KAOS

Marc Frappier, Université de Sherbrooke, Canada
Régine Laleau, Université Paris-Est Créteil, France

Nowadays, the usefulness of the formal verification and validation of system specifications is well established, at least for critical systems. However, one of the main obstacles to their adoption lies in obtaining the formal specification of the system, and, in the case of refinement-based formal methods such as B System or Event-B, in obtaining the most abstract specification that heads the development of the system. There is a significant gap between the unstructured informal requirements that are defined at the inception of a project and the formal system specification to build.

The SysML/KAOS requirements engineering method has been proposed to overcome this difficulty. It includes a goal modeling language to model requirements from stakeholders needs. This language covers functional goals, non-functional goals and obstacle (hazard) analysis that creates new goals to mitigate hazards. Translation rules from a goal model to a B System specification have already been defined. They allow to obtain a skeleton of the system specification. To complete it, a language has been defined to express the domain model associated to the goal model. The domain model describes the main entities of the system and their associated knowledge. Its translation gives the structural part of the B System specification.

In this presentation, we report on a case study performed at the City of Montréal where we have applied our method to model (after the fact) the redesign of an interchange between two main highways in the city center. One of the highway is underground, and has a ramp to connect to the other highway, which has been replaced by a street. Traffic lights at the first intersection of this street may induce a traffic queue within the ramp, which may ultimately progress towards the highway. The ramp being a curved tunnel, drivers have a limited line of vision. Hence, there is a potential for collisions between cars exiting from the highway and the cars stopped within the ramp. Sensors have been installed on the ramp from the street intersection to highway connection. These sensors are used to measure traffic within the ramp and display warning messages on message panels located upstream on the highway to warn drivers to reduce their speed to avoid collision with cars stalled in the ramp. The sensors are also used to control the traffic lights on the street to give priority to the ramp traffic when congestion occurs.

We have produced a goal model of this system, modelling both the physical environment and the software, in order to analyse safety properties. Our client, the City of Montréal, favourably received our model, appreciated the clarity of the goal model, and the ability to analyse its completeness. They intend to conduct another pilot project with our method on another system that must be developed in the coming months. We are in the process of finalising the formal models and proving the essential safety properties using Atelier B and Rodin, two CASE tools that support Event-B and the System B notation. We are also using the editor that we have developed for the goal model using OpenFlexo, and another tool that we have developed for the domain model. The models are translated into Event-B and System B using Jetbrains. The formal models are of an interesting size, and we are looking at which parts are truly essential in order to prove safety the necessary safety properties.

Incremental Assurance of Automotive Systems

Mark Lawford, McMaster University, Canada

This talk presents research projects underway at the McMaster Centre for Softawre Certification (McSCert) to improve the safety, security and dependability of advanced automotive software systems, by developing methods and tools for performing effective and practical incremental safety assurance in the automotive industry. The state of practice in safety assurance in the automotive industry can be made more rigorous and efficient by using safety engineering knowledge, together with model management techniques, to structure and incorporate assurance cases in the software development lifecycle. As systems evolve and product lines of systems are created, incremental assurance of systems becomes an imperative. To remain competitive and deliver ever-safer vehicles at a purchase cost the public is willing to pay, automotive companies must develop methods that will enable the use of the assurance developed for the previous instance of the system. Since safety is a global property of a system, incremental assurance must cope with the complexity of emergent behaviours that introduce safety concerns in a previously safe system. Assurance and compliance activities can be made more effective and less costly by combining sophisticated model management techniques with existing and new techniques for building and certifying safe, secure and dependable systems.

The de facto functional safety standard for electronic and software units in automobiles is ISO 26262 [ISO11]. All automotive Original Equipment Manufacturers (OEMs) as well as the Tier 1 automotive suppliers are (voluntarily) striving to comply with ISO 26262, or, at the very least, be consistent with ISO 26262. This provides a convenient and important focus for our research. Specifically, we are developing theoretically sound methods and prototype tools for: (i)

Safe reuse of vehicle design and assurance case artifacts under evolution scenarios, that include incremental design changes (e.g., modifying/adding features and composing systems); (ii) Checking and maintaining consistency between automotive software development and safety processes, and safety standards, specifically ISO 26262; (iii) Safety related impact analysis to detect when undesirable emergent behaviour might necessitate rework of existing assurance artifacts; (iv) Developing assurance cases for vehicle product lines.

The research project underway at McSCert with our collaborators is attempting to:

- Providing an effective model-based analysis of automotive software-related safety processes. This is crucial because all automotive manufacturers have been moving to a model-driven development environment, typically using MathWorks's Matlab/Simulink
- Providing methods and prototype tools that will support incremental safety assurance and safety assurance of product lines
- Providing methods and prototype tools that will reduce the time and cost associated with software compliance activities, especially with respect to standards such as ISO 26262.

As an example of a tool to perform impact analysis on software models, we present of Boundary Diagram Tool for change impact analysis of large Simulink designs of embedded systems. In our previous work, we developed the Reach/Coreach Tool for model slicing within a single Simulink model [PPL15,PPL17]. The Boundary Diagram Tool extends the Reach/Coreach Tool to trace the impact of model changes through multiple models comprising an embedded system, including network interfaces. The change impact analysis results are represented using various diagrams motivated by industrial needs. Different techniques are employed to improve understanding of impact analyses of large industrial systems. The tool is currently being integrated into the software development process of an automotive OEM (Original Equipment Manufacturer). It supports the following activities of the process: change request analysis, design change evaluation, implementation, verification and integration. The tool also aids impact analyses required for compliance with functional safety standards.

Like most systems, automotive software systems evolve due to many reasons including adding, removing or modifying features, fixing bugs, or improving system quality. In this context, safety cases, used to demonstrate that a system satisfies predefined safety requirements, often dictated by a standard such as ISO 26262, need to co-evolve. A necessary step is performing an impact assessment to identify how changes in the system affect the safety case. In [KSC17] we show how exploiting knowledge about system changes and ISO 26262 to perform impact assessment of GSN safety cases with the goal of reducing unnecessary revision work required by a safety engineer.

While the previous work deals with the explicit model of an assurance case, it does not consider the development and safety processes. In contrast to currently dominant approaches to building assurance cases, which are focused on goal structuring and/or logical inference, in [DMW18] we considering assurance as a model transformation (MT) enterprise: saying that a system possesses an assured property amounts to saying that a particular assurance view of the system comprising the assurance data, satisfies acceptance criteria posed as assurance constraints. While the MT realizing this view is very complex, we show that it can be decomposed into elementary MTs via a hierarchy of refinement steps corresponding to steps in the development and safety processes. The transformations at the bottom level are ordinary MTs that can be executed for data specifying the system, thus providing the assurance data to be checked against the assurance constraints. In this way, assurance amounts to traversing the hierarchy from the top to the bottom and assuring the correctness of each MT in the path. Our approach has a precise mathematical foundation (rooted in process algebra and category theory) - a necessity if we are to model precisely and then analyze our assurance cases.

References:

[DMW18] Z. Diskin, T. Maibaum, A. Wassyng, S. Wynn-Williams, M. Lawford. "Assurance via model transformations and their hierarchical refinement." In Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 426-436. ACM, 2018.

[ISO11] International Organization for Standardization (ISO), ISO 26262 Road Vehicles – Functional Safety, ISO Standard 2011.

[KSC17] Sahar Kokaly, Rick Salay, Marsha Chechik, Mark Lawford, Tom Maibaum: Safety Case Impact Assessment in Automotive Software Systems: An Improved Model-Based Approach. SafeComp 2017, LNCS 10488, Springer, 69-85, 2017.

[PPL17] V. Pantelic, S. Postma, M. Lawford, M. Jaskolka, B. Mackenzie, A. Korobkine, M. Bender, J. Ong, G. Marks, A. Wassyng, Software engineering practices and Simulink: Bridging the gap. International Journal on Software Tools for Technology Transfer, Springer, 1-23, 2017.

[PPL15] V. Pantelic, S. Postma, M. Lawford, A. Korobkine, B. Mackenzie, J. Ong, M. Bender, "A Toolset for Simulink: Improving Software Engineering Practices in Development with Simulink," Proceedings of 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2015), SCITEPRESS, 2015, 50-61.

Moore-Machine Filtering for Timed and Untimed Pattern Matching

Masaki Waga, Sokendai University, Japan

Monitoring is an important body of techniques in runtime verification of real-time, embedded and cyber-physical systems. Mathematically, the monitoring problem can be formalized as a pattern matching problem against a pattern automaton. Motivated by the needs in embedded applications—especially the limited channel capacity between a sensor unit and a processor that monitors—we pursue the idea of filtering as preprocessing for monitoring. Technically, for a given pattern automaton, we present a construction of a Moore machine that works as a filter. The construction is automata-theoretic, and we find the use of Moore machines particularly suited for embedded applications, not only because their sequential operation is relatively cheap but also because they are amenable to hardware acceleration by dedicated circuits. We prove soundness (i.e., absence of lost matches), too. We work in two settings: in the untimed one, a pattern is an NFA; in the timed one, a pattern is a timed automaton. The extension of our untimed construction to the timed setting is technically involved, but our experiments demonstrate its practical benefits.

As in the discussion after the talk, energy efficiency is also an essential requirement in embedded applications and one major future work toward a real application is to concern the energy consumption. Our filter reduces the energy consumption by reducing the size of the data transmission, but our filter itself requires additional energy. An evaluation of the total energy consumption is crucial for a real application.

ASM-based design and verification of self-adaptive systems

Paolo Arcaini, National Institute of Informatics, Japan

Self-Adaptation is recognized as an effective approach to deal with the increasing complexity, uncertainty, and dynamicity of modern software systems. Such systems operate in dynamic environments and deal with highly changing operational conditions: components can appear and disappear, may become temporarily or permanently unavailable, may change their behaviour, etc. A self-adaptive system is able to adapt autonomously to internal dynamics and changing conditions in the environment in order to achieve particular quality goals and to ensure the required functionality.

Feedback control loops that monitor and adapt managed parts of a software system are considered crucial for realizing self-adaptation in software systems. The MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) autonomic control loop is the most influential reference control model for self-adaptive systems. In case of large, complex and decentralized systems, multiple interacting MAPE loops are introduced. Some common design patterns of interactive MAPE components have been proposed in the literature [4]; however, a well-defined way to document them and to express the semantics of their interactions is still missing.

In this talk, I present the framework we proposed for formal modeling and analysing self-adaptive systems. We introduced a formalism, called self-adaptive Abstract State Machines, that exploits the concept of multi-agent Abstract State Machines (ASMs) to specify distributed and decentralized adaptation control in terms of MAPE-K control loops [3]. We support validation and verification techniques for discovering unexpected interfering MAPE-K loops, and for assuring correctness of MAPE components interaction when performing adaptation [2,3]; such activities are performed in the ASMETA framework (`http://asmeta.sourceforge.net/`).

When specifying a self-adaptive ASM, a designer must implement a communication mechanism among the different agents involved in the self-adaptation. Implementing this mechanism could be tedious and error-prone; we therefore proposed the MAPE Specification Language (MSL) (`https://github.com/fmselab/msl`) as a modeling front-end to define and instantiate common patterns of interacting MAPE components when architecting the adaptation logic of a self-adaptive system [1]. We also provided a semantic mapping (implemented by a model generator) to transform MSL descriptions of MAPE pattern instances into templates of self-adaptive ASMs; such templates contain placeholders that must be filled by the designer with the ASM rules defining the adaptation logic. Then, on the obtained model, the designer can apply all the validation and verification techniques of the ASMETA framework.

References

[1] P. Arcaini, R. Mirandola, E. Riccobene, and P. Scandurra. A DSL for MAPE patterns representation in self-adapting systems. In C. E. Cuesta, D. Garlan,

and J. Perez, editors, Software Architecture, pages 3-19, Cham, 2018. Springer International Publishing.
[2] P. Arcaini, E. Riccobene, and P. Scandurra. Modeling and Analyzing MAPE-K Feedback Loops for Self-adaptation. In Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self- Managing Systems, SEAMS 2015, Florence, Italy, May 18-19, 2015. ACM, 2015.
[3] P. Arcaini, E. Riccobene, and P. Scandurra. Formal design and verification of self-adaptive systems with decentralized control. ACM Trans. Auton. Adapt. Syst., 11(4):25:1-25:35, Jan. 2017.
[4] D. Weyns, B. R. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Goschka. Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers, chapter On Patterns for Decentralized Control in Self-Adaptive Systems, pages 76-107. Springer, Berlin, Heidelberg, 2013.

On a cyber-physical nature of modern railway signalling systems

Paulius Stankaitis, Newcastle University, United Kingdom

For years formal methods have been successfully applied to the railway domain however yet a considerably little work has been done in including a cyber-physical nature of railway for a safety reasoning. Established railway operation principles did not require that so formal methods mainly focused on a static railway data verification - control table verification. However modern signalling systems were developed to reduce an overdesign and hence increase the capacity of railway networks.

Railway operational principles have been rapidly moving towards a continuous agent communication and a more dynamic parameter (e.g. permitted speed profile) computation which are indeed two essential aspects of cyber-physical systems - communication and computation. Furthermore, some of modern railway signalling systems are heterogeneous, meaning, parts of the railway network are controlled by different signalling systems. To give an example Crossrail is a major ongoing railway project where mainline services will be integrated with a high performance urban railway system. This particular network will operate with three different signalling systems. In western and eastern branches of the network fixed block signalling systems will operate whereas the central area will be operated with a moving block principle. Novel signalling interfaces will be developed to ensure a smooth and safe rolling stock signalling transition.

To model and reason about safety of a modern signalling system we believe it is paramount to consider a cyber-physical nature of railway. In this presentation I have presented a brief evolution of railway signalling systems and discussed the direction of my research work which could allow reasoning about safety of these systems. In particularly, the research focuses on a refinement and proof based modelling of distributed-hybrid systems. The presentation also summarises challenges and weaknesses of the proposed method, for example: inadequate verification tools for hybrid models and a currently weak visualisation support (for railway engineers).

Towards State Space Reduction for B and Event-B

Philipp Koerner, University of Dusseldorf, Germany

State space reduction techniques have proven themselves to be very useful both to improve performance for model checking and for reasoning about state spaces using visualisations. In particular, partial order reduction (POR) performs very well for several low-level formalisms, e.g, petri-nets and Promela. Yet, for high-level formalisms like B and Event-B, only little work has been done on evaluating the impact of POR. Here, we discuss the current situation concerning state space reduction for two high-level formalisms, B and Event-B.

ProB is an animator and model checker for B, Event-B and several other formalisms. It offers symmetry reduction for deferred sets and, as part of his PhD thesis, Ivaylo Dobrikov implemented a partial order reduction algorithm. In theory and using examples, where the reduction is obvious, the reduction works as expected. For many real-life examples from industry, literature and academia, however, little to none reduction is achieved.

Recently, we integrated ProB into LTSmin, a state-of-the-art language-independent model checker that often outperforms ProB even though it uses its next-state function. We re-used the analysis of ProB's POR in order to define the independence relationship between events that is given to LTSmin. Using LTSmin's algorithm for POR, which is more fine-grained, in general, better reduction is gained or it performs as well as ProB's algorithm. Yet, the overall reduction remains unsatisfactory.

At the moment, we do not have an explanation for our observations. Some theories are as follows:

- The analysis performed is not precise enough. That means, independence of events is not discovered consistently and overapproximation happens too often. This might be caused by a constraint solver that is too weak to handle the necessary constraints. A possible solution is to explore the impact of alternative solvers (e.g., via a translation to SMT) or somehow to improve ProB's current abilities.

- The modelling style that is prevalent in B and Event-B is a bad fit for POR. Maybe the models that we tested do not offer any possibilities for POR in the first place. Then, it would be interesting to analyse which constructs or pattern are the cause. - The high-level approach in general gives less possibilities for POR. For example, a set union is part of the language that can be executed in a single transition. In lower-level formalisms without language support for sets, it might be necessary to add each element individually into some representation, where the ordering does not matter and, thus, POR might be applied.

Additionally, we found an issue concerning the analysis step: the classical POR algorithm requires an action-deterministic transition system, i.e., a given action has only one successor state. In B and Event-B however, it is possible to have several successors using the same event by different instantiation of parameters. Thus, it might be reasonable to split an event into several events wherever possible, i.e., where only a finite set of parameters are possible.

In summary, the practical applications of state space reduction techniques in B and Event-B are very limited. Thus, we will try to analyse how we can improve POR when using ProB in the future.

Automatic extraction of modes and invariants from hybrid data-flow models

Rémi Delmas, Thomas Loquen, Pierre Roux, ONERA Centre Midi-Pyrénnées, Dept. of Information Processing and Modelling, Toulouse, France

Control engineers typically use hybrid data-flow formalisms such as Simulink and SCADE to design and model advanced embedded controllers. These formalisms allow to mix continuous data-flows whose behaviour is defined by switched ODE systems, with discrete data-flows, whose behaviour is defined by inductive definitions cadenced by the occurrence of zero crossing events over continuous or discrete data-flows. Data-flow formalisms are declarative formalisms with deterministic semantics and geared towards simulation and efficient sequential code generation, with language features facilitating encapsulation, composition and reuse.

On the other end of the spectrum, we find formalisms based on hybrid automata, an extension of explicit-state automata with continuous state variables, and in which each explicit state specifies a distinct ODE system, state invariant and exit transitions to other states. These formalisms have a non-deterministic semantics and were initially meant as a theoretical framework for studying the semantics of different classes of hybrid systems, (bounded) reachability problems and formal verification. As a result, most if not all formal analysis engines for hybrid systems are based on some dialect of hybrid automata.

The goal of this work in progress is to bridge the representation gap between hybrid automata and hybrid data-flow formalisms, in order to bring the benefits of symbolic reachability analysis and formal verification to the control engineers in the data-flow realm. This work is conducted in partnership between ONERA and Airbus Operations SAS, Dassault Aviation and LAAS-CNRS in the context of the IKKY-SEFA DGAC programme.

Taking inspiration from previous work by Bourke et al. [1], Schrammel et al. [2] and Vianna et al. [3] we propose a recursive translation algorithm from hybrid a data-flow language to extended algebraic decision diagrams (XADD). These diagrams are an extension of multi-terminal decision diagrams in which terminal nodes are elements of some real term algebra, and decisions nodes are labelled by relational operations over the same real term algebra. The compilation to XADDs allows to mechanically identify and cleanly separate the discrete switching structure (represented in the the decision diagram structure), the zero crossings conditions (represented by the conditions labelling the decision nodes), and pure ODE systems (represented by the real terms contained in terminal nodes) hidden inside the complex control flow structure of a hybrid data-flow model. The conjunction of conditions on the path from the root of the decision diagram to a terminal node (ODE system) yields the invariant condition associated with that terminal node.

Ongoing work aims at further flattening the resulting XADD to disjunctive normal form to obtain a hybrid automaton-like structure and connect to existing analysis tools for (bounded) reachability analysis and formal verfication. For cases in which the flattening is impossible (the DNF representation can be exponentially larger in the worst case), the transition relation of the hybrid system is extracted from the XADD as first order predicates in order to use SMT-ODE [4] solvers directly to perform bounded reachability analysis, deferring the combinatorial explosion.

References:

[1] T. Bourke and M. Pouzet. Zélus: a synchronous language with odes. In *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 113–118, 2013.
[2] P. Schrammel and B. Jeannet. From hybrid data-flow languages to hybrid automata: a complete translation. In *Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China, April 17-19, 2012*, pages 167–176, 2012.
[3] L. G. Vianna, S. Sanner, and L. N. de Barros. Bounded approximate symbolic dynamic programming for hybrid mdps. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*, 2013.
[4] S. Gao, S. Kong, and E. M. Clarke. dreal: An SMT solver for nonlinear theories over the reals. In *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, pages 208–214, 2013.

Designing Methodologies Enabling Automated Verification

Tetsuya Tohdo, Denso, Japan

This talk provides some experience of applying automatic test generation to product development in DENSO CORPORATION. Ideas and technologies based on Formal Methods will contribute to support the real development if we succeed in formalizing "engineering" by reflecting the real meaning of development tasks in the method. We have successful experiences, for example, in which a test coverage criteria for model-based development is defined and applied for generating test data used for the back to back unit testing. One of the major challenges is to apply similar engineering approach to the functional testing of the cyber-physical systems. The objectives of functional testing is to check if the system behavior is as intended and to check if any undesired behavior is not included, but the both objectives are hard to formalize because the requirement specifications are often incomplete and sometimes even inconsistent during the development phase. Our approach is to formalize a notion of "exhaustiveness" of system design and verification. As simulation (including prototyping) is a technique frequently used for identifying and verifying requirements, formalization of the exhaustiveness of such simulation activities will support to automate the verification work such as consistency checking and test generation. We have not yet reached satisfying concept, but based on the observation of system development, we are consider relative guidance such as criteria to generating corner cases as "exhaustiveness" that covers ambiguous part of requirements from intended part. This approach will be an extension of contract-based design in the context of using ambiguous abstraction of the requirements such as approximation with unknown accuracy and description by analogy.

Challenges of Formal Methods for Next Generation of Cars

Toshiaki Aoki, JAIST, Japan

Recently, the safety and reliability of automotive systems are becoming a big concern in society. Although vehicles have been controlled by simple mechanics in the past, many of electronic parts are embedded in them at present according to the progress of electronic control technology and its performance. These electronic parts can realize the complex control of the vehicles, and make it possible to provide high functionality to vehicles such as automatic speed controlling and emergency braking. However, unfortunately, electronic parts also introduce the problems of the reliability and safety of the vehicles because the automotive systems become more complicated and their scale larger. In addition, we are now going to realize autonomous driving in near future.

It is quite obvious that the autonomous driving makes the automotive systems much more complicated, causes the drastic increase of those scales and introduces new problems about the safety. Some major issues for them are described as follows. 1) Artificial intelligence and machine learning techniques are being used in the automotive systems. How do we deal with the safety for automotive systems which use those kinds of techniques? 2) Autonomous stuffs introduce various kinds of uncertainty. How do we facilitate them in the development and operation? 3) Model-Based Development(MBD) which uses MATLAB/Simulink and SCADE is popular for automotive systems. The model of MBD contains much of undecidability as well as becomes large scale. How do we deal with such undecidability and large scale for the verification? 4) Because large scale software requires high performance of the platform, many core platforms which accompany advanced scheduling enabling the high concurrency are used for the automotive systems. On the other hand, such platforms need to ensure real-time properties for important parts of the systems. Thus, the scheduling is different from that of ordinary concurrent systems. How we develop and verify such scheduling is challenging. 5) The automotive systems are becoming more and more open. In this case, their quality assurance such as following coding standards, safety analysis and safety cases, is becoming more and more important. Dealing with them in automotive system developments is needed. As shown in the above, there are important challenges for automotive systems of next generation cars and we are challenging them with formal methods.

Refactoring Refinement of Event-B Models

Tsutomu Kobayashi, Fuyuki Ishikawa, National Institute of Informatics, Japan

Event-B has been attracting strong attention because of its flexible refinement mechanism to deal with the complexity of contemporary software. There are various formal methods that support data refinement, which aims for deriving program construction from comprehensive specifications. Event-B supports not only data refinement but also superposition refinement, which enables developers to gradually add aspects of target systems to models. Therefore, Event-B is suitable for constructing complex system models.

Because of this refinement mechanism, developers have multiple possible ways of adding aspects of target systems (refinement strategies). To construct a model of a system Mxyz that has aspects x, y, z, a developer may construct a model Mz about aspect z first, then a model Myz by adding the aspect y before constructing Mxyz. Alternatively, a developer may construct a model Mxz about aspects x,z first and then construct Mxyz.

We proposed a method of refinement refactoring for Event-B models. When an existing model and a new refinement strategy are given, the method constructs new models that are consistent with the original model by following the new strategy. The core of the refactoring method lies in the interpolation of refinement. When an abstract Event-B model MA, a concrete model MC refining MA, and a set of variables to be defined in the new model (interpolating criterion) V are provided, our interpolation method constructs a consistent intermediate model MB written with variables V such that MB refines MA and MC refines MB. Combined with merging of refinement (i.e., the reverse operation of interpolating), developers can merge multiple refinement step into a single large step and decompose it by following a new refinement strategy.

Interpolation of refinement is not straightforward. A naive approach towards interpolation of refinement is slicing, namely constructing the intermediate model as a collection of parts of original models that can be written with interpolating criterion V. However, slicing is not enough for constructing a consistent intermediate model, because slicing may drop predicates of the original model that are necessary for consistency. Thus, we need to derive predicates that are written with variables of V and sufficient for satisfying consistency conditions. We call such predicates complementary predicates.

To derive complementary predicates, we use Craig's interpolation theorem. An interpolant X of formula P ? Q is written with symbols that appear in both P and Q. For a formula of consistency of the original model Hyp ? Goal, our interpolation method converts the formula into an equivalent formula Hyp? ? Goal? such that the set of variables common in Hyp? and Goal? is a subset of V. We defined conversion rules for each consistency rule of Event-B. With this conversion, our method enables developers to systematically derive complementary predicates.

We implemented functionalities of slicing refinement, deriving complementary predicates, and merging refinement as a plug-in tool of IDE for Event-B (Rodin Platform). Our plug-in provides a list of variables of the original model with checkboxes. By selecting the interpolating criterion V using this plug-in interface, a user can obtain models for the new refinement strategy. The generated models are guaranteed to be consistent with the original model because our tool automatically derives a complementary predicate for every consistency condition necessary for new models. Interpolants of converted formulae are calculated by Z3 SMT solver. Thus, our tool realizes automatic refactoring of Event-B models.

Refinement refactoring has various benefits. Developers can use it as an automatic abstraction method to obtain a simple and consistent model of complicated models. By specifying the interpolating criterion V as a set of variables of reusable aspects, it can also be used to extract reusable aspects of the existing model as an abstract model. Thus, refinement refactoring strengthens advantages of formal specification and superposition refinement such as understandability and reusability of models. We also used it for design space exploration of Event-B models. In order to compare multiple refinement strategies for a model, we used our tool to generate multiple refactored versions of an existing model by following different refinement strategies. As a result, we found good practices for refinement strategy design such as introducing variables that are frequently used in invariants.

We conclude that refinement refactoring can be a promising method for engineering of rigorous models for complicated software systems.

A preview of the HiPEAC Vision 2019 report

Tullio Vardanega, University of Padua, Italy

The take-up of IoT and cyber-physical systems in a variety of application domains fosters the convergence of high-integrity embedded systems with high-performance computing. The former feed the edge and the fog part of the system, close to the physical world; the latter fuel the back-end part, deployed in the cloud; both tend to form an unprecedented continuum of computing that needs to be addressed as a high-integrity whole, before being broken down into specialized parts. The traits of the application domains deployed on those systems, of which automotive is a most remarkable instance, increasingly exhibit high-integrity needs. This trend makes that ambit a very significant arena for the industrial application of advanced formal methods, which is the topic of the present Shonan meeting.

HiPEAC (`https://www.hipeac.net/`) is a long-lived European network of researchers, supported by the European Union's Horizon 2020 program, focusing on high-performance and embedded architecture and compilation challenges. One of the tasks that HiPEAC is assigned every two years, is to produce a "Vision" report that surveys the trends and the challenges for business and technology in the domains of interest, notably IoT, CPS and HPC, and formulates recommendations for where the research and development focus of Europe should be placed in the next decade. The 2019 vision report will be presented at the prime yearly event of the HiPEAC network, the coming January, in Valencia, Spain. Over the last 14 months, I had the opportunity to engage myself in the group that undertook the work of creating that report. On the occasion of this Shonan meeting, I thought it would be interesting, for me and for the attending colleagues, to offer an excerpted preview of that report.

It is not for this short summary to replay all of the preview that I have given, but a few highlights are definitely worth of mention, because they overlapped with discussions that the attending group had throughout the meeting. Most notably, the vision report singles out some business trends that are distinctly influential on where research focus is being shifted: servitization, which transforms products into pay-as-you-go services, which are obtained from the cited continuum of computing; verticalization, which develops ad-hoc hardware architectures and the associated programming environments to respond to the needs of heterogeneous computing (championed by deep learning) that cannot be supported by traditional von-Neumann-type processors; human-in-the-loop systems, which require major advances in the degree of "artificial intelligence" feeding interaction, sensing, and possibly planning tasks at various points of the cited continuum of computing, but prevalently at the edge, which is most exposed to the human element. Designing, implementing, and verifying systems with those characteristics is a very big and hard challenge, which can hardly be performed with conventional techniques and technologies. Many of the discussions we had at this Shonan meeting provided useful insights on what advanced formal methods might do to help address the emerging challenges.

Lessons learnt from experiments in the development of CPS using refinement

Yamine Ait Ameur, Guillaume Dupont, Marc Pantel, Neeraj Kumar Singh, INPT-ENSEEIHT/IRIT, University of Toulouse, France

Due to their presence in many critical systems, the formal verification of cyber-Physical systems is a key issue in system engineering. As such systems integrate both continuous and discrete behaviours and contrary to classical discrete systems, the formal specification and verification of such systems require taking into account continuous features like differential equations to characterise plant behaviours and the necessary logic and proof system.

Several research work addressed the formal verification of hybrid systems (Alur et.al). Hybrid model checking, proof based approaches, program analysis, simulation have been proposed. Both approaches consider a hybrid system as the tight integration of discrete and concrete features defining models for controllers and for the plant to be controlled. However, addressing formal verification at the model level abstracts implementation details in particular floating point arithmetic. The verification of the defined models and the synthesis of controllers guarantees the system requirements independently of any implementation. Only implementation requirements like floating point computation will remain to be addressed once code is obtained from the verified models.

In this talk, we discuss a correct-by-construction approach based on refinement and proof based techniques. We propose to handle the integration of continuous and discrete behaviours in Event-B and the Rodin platform. This approach required the modelling of continuous features currently not available in the core Event-B (J.R. Abrial). For this purpose we, use the Theory plug-In developed for Rodin to define a theory for manipulated continuous functions and ODEs. Moreover, we discuss several issues related to modelling of centralised or distributed controllers in present of several plants.

2 Problem Statements

Below shows part of problem statements from industrial participants.

On our experience of industrial projects using formal approach

Daichi Mizuguchi, Atelier Corporation, Japan

Since 2013, Atelier has been offering services on the domains including:

- Functional safety:
  - ISO 26262, IEC 61508, ISO 25119, etc.
  - Hazard analysis, Safety concept, Safety design
- Cybersecurity :
  - SAE J3016, ISO/SAE(CD) 21434
  - Threat analysis, Cybersecurity concept

There we can take formal approach such as:

- Specification verification – Event B
- System architecture – B method
- Design verification – SPIN/Promela

Our experiences on applying formal methods based on our customer's requests include projects such as:

- "Specification analysis of automotive software component"
  - "Transmission control module" and "Sensor input module"
  - Event B and model checking
  - Discovered:
    - Unspecified conditions when to delay the shift position change after a shift change decision.
    - Several unknown conditions when sensor is considered to be ON instead of OFF (against the specification).
- "Transformation from semi-formal models to formal models"
  - Aim: To reduce effort to make formal models
  - Showed:
    - B model can be derived from "block definition diagrams" and "sequence diagrams".
    - Safety properties can be distributed and embedded in MACHINES as specification in the B model.
- "Test cases generation for Cybersecurity"
  - Model mutation on the event B model and model checking [1]
  - An on-going project.

However, we are not so successful in pushing formal approach into real development so far. The following points can be discussed to make formal approach more actively and aimfully used in the industry:

- Most developments are modification projects on the past projects:
  - where testing, reverse engineering, refactoring and such are given higher priority.

- It is hard to change the existing development process and tools.
- Division of work is not enough/thorough.
  - Sometimes, specification, design and testing are done by one person, in which case verification activities may not be taken seriously.
- When do you feel assured:
  - either your system is "well-tested" or "proved"?
  - Differences in "Assurance culture"? item Confidence level, confidence in tools.

References:
[1] A. Savary, M. Frappier and J.L. Lanet: "Detecting Vulnerabilities in Java-Card Bytecode Verifiers using Model-Based Testing", Integrated Formal Methods (IFM2013).

Introducing Formal Methods into Industry

Hironobu Kuruma, Hitachi, Ltd., Japan

As a part-time lecturer, I am teaching B method and Event-B in the TOPSE education program at NII. The objective of B method course is to study software development process on a formal basis and 15 time-block (1 time-block is 1 hour and 30 minutes) lectures are delivered to about 10 students from industry every year. Event-B course is aiming at studying system modeling and analysis using Event-B. Around 5 students a year from industry take this course and 7 time-block lectures are delivered. I experienced several barriers for students from industry to learning B method and Event-B. The first barrier is mathematical symbols. Most students are not familiar with mathematics and they are puzzled by mathematical notations in formal specifications. The second one is interactive proof. Japanese industries wish for development methods that do not rely on skills of each engineer. So, the students are confused by the fact that the success of interactive proof depends on the personal skill. The third one is modeling. Engineers might say that they usually do not aware of invariants of their design. Designing in a formal specification language forces the students to change their way of design. The fourth one is refinement strategy. The use of refinement in system development is new to the students. They feel the refinement of abstract model in B and/or Event-B tricky before they get program codes. To promote formal methods in industry, I and my colleagues in industry and academia made several case studies on formal modeling and verification of industrial products that are developed by using traditional methods. Most case studies end successfully and we have pointed out a number of inconsistencies in specifications. However, the engineers who provided design information on their products are disappointed since they expect striking results such as finding defects that never be found by human reviews in program codes. They might say most inconsistencies we found in specifications are easy to fix for domain experts or already fixed in the coding process and/or the testing-debugging cycle. The subject is the tradeoff between the cost of training engineers and the risk of changing development process and the benefit of formal specification and verification. Since it is difficult to evaluate the training cost and process risk of traditional methods used in developing existing products, we should show the benefit of formal methods by applying them to new products under development in collaboration with engineers. I think what needed to introduce formal methods into industry are: First, to show the benefit of using formal methods. In my experience, engineers have questions such as formal methods improve the reliability of their products, formal methods improve the productivity of their development process, and formal methods can be introduced at reasonable costs. I believe that applying formal methods to products under development in collaboration with them provides answers to their questions. Second, to show how to use formal methods. Engineers have questions such as for what purpose formal methods could be used, who, for example system integrators or software designers, should use formal methods, and how homogeneity of products can be assured. Systematized modeling and verification know how or domain specific development tools may provide answers.

3 Summary of Discussions

3.1 Practices and Industry

The notes below summarize the group discussions about the following topics:

- Formal Methods for Practicing Engineers
- Industrial Needs Concerning Formal Methods
- Challenging and Potential Solutions to Use Formal Methods in Industry

The discussion started by a summary of existing approaches which are based on formal methods technologies. The participants agreed that there is a continuous spectrum of approaches which make use of formal technologies, each of these having different degrees of automation and adoption in the industry and in different business domains. At one end of the spectrum are comprehensive specification methods like, for example, B, Z or VDM. At the other end are formal technologies which are at the core of static analysis tools or tests generators.

A second direction for the discussion was about the main motivation of the industry to use different formal approaches. In general, the participants agreed that economic factors drive the use of formal methods – however, these factors manifest quite differently.

- For example, a classical example of use of formal methods in the industry is the development of safety critical systems where the use of formal approaches eases the certification and reduces the amount of testing.
- Another industrial field where formal approaches are deployed more and more is the development of security critical systems. The potential damage produced by successful attacks, and thereby the loss of money, motivate companies invest ressources to increase the level of security.
- A third vector for the adoption of formal techniques is to increase the productivity of forward engineering activities (e.g. through synthesis, automation of quality assurance) in the context of the even shorter time

to market and the pressure to deliver functionalities in a continuous (or even perpetual) manner. In these cases, the automation offered by formal approaches can substantially increase the development speed.

- Last but not least, the increased complexity to todays systems pose challenges on their integration. A classical example is the specification and integration of sub-systems developed by external suppliers. This poses big challenges and could make the case for the use of more rigorous specification of interfaces.

The third direction of discussion was to identify main causes which prevent the industry from using formal methods on a larger scale.

- The use of more advanced formal aproaches require changes in processes, organizations and practices. These changes come with a risk and thereby the management often does not take the decision to introduce them. For bigger changes to happen, many stakeholders (technical people, QA, managers, assessors, customers) should see a clear and undoubtful added value of the proposed method – this is often not the case.
- Usability remains one of the major concerns with formal methods – we need tool front ends both for the input and the output of the verification engines.
- In the context of safety critical systems, a broader deployment of formal verification requires that the used formal tools are certified. Certification of formal verification tools is a research topic which needs more investigations.
- Last but not least, full correctness is sometimes the wrong target for a component since many of the errors which can be identified by formal means have no practical relevance or are prohibitively expensive to fix.

3.2 Specification

Our discussion session was centered around the question "how do we obtain formal specifications in real-world application of formal methods?". The discussion group was formed by an industry practitioner, a few with ample experience of industrial collaboration, and a couple of theoreticians. The following are some points raised during our discussion.

Firstly, it was recognized that specifications play the role of interface among different parties during the design and development of actual systems. In the formal methods context, a prototypical example is that two parties, namely a customer (system designers) and a prover (formal-method engineers), interact and communicate using specifications as media. However, the organization of different parties is much more diverse than the simple customer-prover model. The following examples are raised in the discussion: 1) three teams for sensing, planning and control, respectively, talking to one another in the design of automated-driving cars; and 2) a more hierarchical, waterfall-style organization in avionics where specifications made by safety engineers are passed to a redundancy team, and then to software developers. It was pointed out that, for the development of real-world systems, it is often very important to decide the structure of the organization, such as “how many teams?”, “what is this team in charge of?”, and so on.

Another important point was different modes of interaction via specifications. It is agreed that specifications are almost never correct, and they have to be elicited through interaction among different parties. There are many ways in which this interaction can happen, and we identified a few prototypes: 1) requirements get refined by multiple queries from the prover to the customer; 2) a system prototype is produced for the communication purpose (i.e. sharing concepts and issues); and 3) the customer gives the prover an implementation as a “specification.”

As a logical consequence from the previous points, we agreed that specifications should be as intentional as possible. It is easier to fix a problem in a specification if it says not only what is wanted but also why it is wanted. This point was confirmed empirically, too, from the experience of industry collaboration of some of us.

After the discussion of the above points, we talked about some research directions. Notable among them was a formal meta-model for specification elicitation, and a tool support for the meta-model. We can aim at a framework that addresses the above issues, that is, different organization structures, interaction among different parties, intentions behind formal specifications, and importantly, incremental development of specifications over time. We suspect this direction is novel: existing works on requirement engineering address the incremental aspects but they are most of the time informal, on the one hand; and on the other hand, existing formal approaches to specifications (such as goal trees and KAOS) may not fully address the incremental aspects of specification elicitation.

3.3 Hybrid Systems

A lot of systems we use and design are hybrid systems - systems, where a continuous plant is driven by a discrete controller. In this discussion, participants shared opinions on formal methods applicability for modelling and verification of such systems. In particularly, we analysed a recently developed modelling framework based on a refinement and proof for developing hybrid system models with the Event-B specification language. The first conclusion was that a lot of assumptions and mathematical theorems used to validate hybrid systems, especially from control theory, are not explicitly stated (referenced) in the model. With the proposed approach these assumptions would be stated (or referenced) and would ensure the traceability requirement for a system certification. The second part focused on discussing system constraint discovery by analysing undischarged proof obligations of mathematically proving hybrid models. Lastly, we finished discussion with a general topic on the need/benefits of formal methods for hybrid systems, especially, in the context of non-existent adequate verification tools and significantly increased modelling effort. Although, no single conclusion was made, few technical challenges to be addressed were identified.

List of Participants

- Alan Wassyng, McMaster University, Canada
- Alexei Iliasov, Newcastle University, United Kingdom
- Carolyn Talcott, SRI International, USA
- Christian Herrera, fortiss GmbH, Germany
- Daichi Mizuguchi, Atelier Corporation, Japan
- Daniel Ratiu, Siemens, Germany
- Elena Troubitsyna, KTH, Sweden
- Hironobu Kuruma, Hitachi, Ltd., Japan
- Hiroyuki Yoshida, Japan Systems Co.Ltd, Japan
- Ichiro Hasuo, National Institute of Informatics, Japan
- Kokichi Futatsugi, NII/AIST/JAIST, Japan
- Krzysztof Czarnecki, University of Waterloo, Canada
- Laurent Voisin, Systerel, France
- Marc Frappier, University of Sherbrooke, Canada
- Mark Lawford, McMaster University, Canada
- Masaki Waga, Sokendai University, Japan
- Paolo Arcaini, National Institute of Informatics, Japan
- Paulius Stankaitis, Newcastle University, United Kingdom
- Philipp Körner, University of Dusseldorf, Germany
- Rémi Delmas, ONERA, France
- Tetsuya Tohdo, Denso, Japan
- Tom McCutcheon, Newcastle University, United Kingdom
- Toshiaki Aoki, JAIST, Japan
- Tsutomu Kobayashi, National Institute of Informatics, Japan
- Tullio Vardanega, University of Padova, Italy
- Yamine Ait Ameur, IRIT, France

Meeting Schedule

Check-in Day: November 4 (Sun)

- Welcome Banquet

Day1: November 5 (Mon)

- Opening
- Introduction
- Talks and Discussions

Day2: November 6 (Tue)

- Problem Statements from Industry
- Group Photo Shooting
- Group Discussions

Day3: November 7 (Wed)

- Talks and Discussions
- Excursion and Main Banquet

Day4: November 8 (Thu)

- Talks and Discussions
- Wrap up

Acknowledgements

The organisers are very much grateful to the participants for all their contributions and for the active engagement in the discussions and sharing their ideas. We are hoping the participants enjoyed the School as much as we had. We would like to thank the Shonan Meetings organisation for their help and support.