Metamathematics for Systems Design

Comprehensive Transfer of Formal Methods Techniques to Cyber-Physical Systems

Ichiro Hasuo¹

Received: 25 May 2017 / Accepted: 29 June 2017 / Published online: 15 July 2017
© The Author(s) 2017. This article is an open access publication

Abstract This position paper describes the context, the goal, the strategy and the tactics of the ERATO MMSD project (2016–2022). The project aims at enhanced quality assurance measures for industry products like cars. In doing so, we follow a recent trend and exploit *formal methods*, a body of mathematical techniques originally developed for computer systems. However, there are fundamental gaps in application of formal methods to industry products: additional concerns in industry products such as continuous dynamics of physical components and quantitative measures such as probability, time, and cost make problems fundamentally different from those about software. Formal methods that accommodate these concerns is an active research area, which shows that it is a hard problem. There are several successful theoretical developments in this direction. They typically combine one individual technique with one specific concern, such as hybrid automata that extend automata with continuous dynamics. Our project aims to contribute to this hard problem in a unique way. In our project we will take a unique *metamathematical* strategy to bridging the gaps: instead of creating one technique for each concern, we want to find a meta-level theory that describes how to develop such techniques for many potential concerns in general. Through this strategy, together with our emphasis on real-world applications in industry, we expect a new prototype of applied mathematics will emerge. In this prototype, abstraction and genericity—characteristics of modern mathematics that are not often associated with application—are turned into crucial advantages in applications.

✉ Ichiro Hasuo
i.hasuo@acm.org

¹ National Institute of Informatics, Hitotsubashi 2-1-2, Tokyo 101-8430, Japan

Keywords Formal method · Cyber-physical system · Verification · Synthesis · Logic · Automaton · Category theory · Metamathematics · Control theory · Software engineering · Optimization · Machine learning · Hybrid dynamics · Quantitative reasoning

Introduction

Modern-day manufacturing undergoes one of the biggest changes ever, with computers tightly integrated in industry products. The number of silicon chips in industry products is dramatically increasing, thanks to those chips industry products nowadays have acquired not only enhanced performance and efficiency but also new functionalities that had been unforeseen. The most prominent examples of such phenomena are in the automotive industry, where engine control units (ECUs) control ignition timing in precision for enhanced fuel efficiency, and we have seen the rise of *autonomous driving*, a paradigm that had been almost unimaginable a decade ago. The term *cyber-physical systems* (CPS) refers to those systems that combine physical components with digital control by computers; modern industry products such as cars are thus representative of CPS.

Computers in industry products pose one of the greatest challenges on manufacturing, too. *Discrete dynamics* governed by computers exhibits a kind of complexity that is essentially different from continuous dynamics: lack of continuity is lack of uniformity, which makes it hard for us to grasp simple governing principles. As a consequence it is as hard as ever to reason about industry products, e.g. for their safety guarantee. For example, nowadays most cars are equipped with electronic throttle control (drive-by-wire); such systems not behaving in an expected manner can lead to severe consequences including loss of human lives. In the modern world, industry products play increasingly important roles and, therefore, their safety and correct behavior is a pressing issue.

It is in this context that the author was awarded an ERATO grant from Japan Science and Technology Agency (JST), leading to the commencement of *ERATO HASUO Metamathematics for Systems Design Project* (ERATO MMSD), October 2016–March 2022. In this project that will eventually employ more than ten researchers, we will use *formal methods*—mathematical techniques for quality assurance originally developed for software—for quality assurance of CPS like modern-day industry products.

This goal of extending formal methods from (homogeneous) software systems to (heterogeneous) cyber-physical systems is in fact a standard one: it has been pursued by many researchers in a number of organized research efforts. However, in the extension of formal methods techniques T by additional concerns e relevant to CPS, the construction of the extended techniques T^e has conventionally been done in an individual and one-by-one manner. As an example of such individual extended techniques T^e , *hybrid automata* take finite automata as their basis T and add the additional concern e of continuous dynamics.

What makes our ERATO MMSD project stand out is its unique research strategy that we call *metatheoretical transfer*: we formalize the process of extension $T +$

$e \rightarrow T^e$ itself as a theory that is parametrized over an arbitrary T and e . This way we obtain a *meta-level* theory on how an *object-level* theory T is extended by an additional concern e . Here the word *theory* refers to a body of definitions and theorems; the object-level theory T thus consists of definitions and theorems about the formal methods technique in question. Such a metatheory on the extension $T + e \rightarrow T^e$ can be thought of as a general recipe that works for various T s and e s in a uniform manner; it yields many concrete extensions T^e as instances. Moreover, via this identification of mathematical essences that goes deeper than superficial application of known algorithms, we would be able to apply a known technique from formal methods to a problem that is not the original goal of the techniques.

We have so far obtained a couple of examples of such metatheoretical transfer. In the *nonstandard transfer* methodology, for example, the original technique T can be in principle an arbitrary formal method technique as long as it is formalized in the language of first-order logic for real numbers. We fix the additional concern e to be continuous dynamics. Our methodology consists of adding a new constant dt to T , and constructing a denotational model of T that accommodates *infinitesimals* relying on Robinson's *nonstandard analysis*. This way we obtain the extended technique T^e , which is almost the same as the original discrete technique T (except for the addition of dt) but applies to continuous/hybrid dynamics. Moreover, the correctness of the technique T^e is guaranteed by the nonstandard denotational model. We have obtained a few concrete techniques by following the methodology. There we take the following techniques as a base technique T : the Floyd–Hoare program logic; a stream-processing language and a refinement type system for it; and abstract interpretation by Cousot–Cousot.

In our project we aim to let this distinctively (meta-)theoretical approach drive concrete applications in real-world problems in industry. Doing so requires careful planning: we need to match concrete industry problems with our theoretical techniques. There are many “no-go” type results (such as undecidability of reachability in hybrid automata) that suggest that conventional formal method goals such as verification are unlikely to be feasible for CPS. We are thus forced to aim at more *light-weight goals* like testing and monitoring; techniques towards these light-weight goals can still make huge difference in reality, given the big roles played by CPS in society. The flexibility we have from the metatheoretical approach is especially useful here: it allows us to adapt ideas from formal methods research to those light-weight methods; and those ideas strengthen the methods significantly, thus contributing to quality assurance measures for CPS with rigorous mathematical bases.

The current position paper lays out the goals, the strategy and the tactics of the ERATO MMSD project. The organization of the paper is as follows. In Sect. 2 we describe the project's backgrounds and contexts in further detail, and identify the challenges that the project shall address. In Sect. 3 our metatheoretical transfer strategy is elaborated on. We shall do so through revisiting the academic field of *metamathematics* that we draw inspiration from, and illustrating two instances of metatheoretical transfer. These examples are nonstandard transfer from discrete to continuous/hybrid, and coalgebraic unfolding. In Sect. 4 we describe our tactics with which we align the strategy with the practical goal of helping design processes of industry products. Here we describe some concrete research topics: they span

from category theory to logic, automata, control theory, software engineering, machine learning, artificial intelligence, mathematical optimization, and many other research fields. Our tactics are also concerned with how our results are put to real-world use. Our philosophy here is to be incremental (finding small portions of design processes in which we can help) rather than monolithic (like requiring comprehensive formal specification to start with). We also describe how we seek to contribute to forming new software-centric design processes of CPS—this is in collaboration with *Autonomoose*, an autonomous vehicle project at the University of Waterloo. In Sect. 5 we discuss the project organization that mirrors those strategy and tactics; in Sect. 6 we conclude.

Backgrounds, Contexts and Challenges

Computers in Industry Products and the Challenges Imposed Thereby

Most modern industry products incorporate computers as their vital components. Digital control realized by those computers has brought drastic changes in the landscape of manufacturing.

On the one hand, digital control has enabled radically enhanced capabilities of industry products. Examples include engine control units (ECUs) that achieve a level of fuel efficiency that would otherwise be impossible, and autonomous driving, a game-changing paradigm in automotive industry that had been almost unimaginable a decade ago.

On the other hand, greater complexity is inherent in digital dynamics governed by computers. A computer program is a recipe for dynamics in the form of a number of lines of code. In such a program it is hard to find the kind of uniformity and continuity that one would often find in physical dynamics: a simple governing equation is unlikely to exist and a small mutation of a program can have drastic effects. This complexity of digital control, and the complexity of modern industry products under digital control, poses a big challenge in the design of safe, reliable and efficient industry products. The challenge is being taken even more seriously now that computer-controlled artifacts play increasingly important roles in society: their failure can result in huge economic and social damage, and even loss of human lives.

This complexity challenge in manufacturing can be discerned in various concrete forms. Let us take the *V-model* (Fig. 1) as an example: it is one of the representative

Fig. 1 The V-Model

schemes of product development processes, originally devised for software development processes but nowadays widely applied also to industry products [28]. Here one would start with devising a high-level design (top-left), then refine the high-level design in a step-by-step manner, implement the design (bottom), and validate the implementation against those requirements in each design step (on the right). Validation is typically by extensive testing.

The following is a (far from comprehensive) list of concrete challenges that one is likely to encounter.

1. *Cost of hardware testing/simulation* In designing complex industry products following the V-model, one big challenge is in each validation step. A product (or a component of it) has so many digital operation modes and the behaviors in different modes are so different from each other. This makes extensive testing that covers all the different operation modes prohibitively expensive. In case of manufacturing (unlike software industry), the fact that an implementation is given by hardware is an additional major burden: testing (or *simulation* as it is commonly called) with a hardware prototype is time consuming and expensive.
2. *Correctness of designs and requirements* Extensive validation efforts (the dashed horizontal lines in Fig. 1) would be in vain if the designs and requirements (on the left in the figure) are faulty. Those designs and requirements sometimes fail to comply with laws and regulations, they can also fail to adequately address customers' needs and they can simply have inconsistencies within. It is in fact widely recognized in the industry that a majority of faults originate in the phase of deciding designs and requirements. Given the increasing complexity of industry products, together with increasingly diverse demand from society, it is a big challenge to come up with a "correct" set of desirable properties such as safety, reliability, efficiency, accountability and so on. Even if we succeed in doing so, coming up with a design that satisfies all those desirable properties is an even bigger challenge.
3. *Management of designs and requirements* In Fig. 1 and other schemes for design processes, design is conducted not in one shot but in an incremental manner, starting with a rough design and refining it to detailed ones in a step-by-step manner. Communicating designs and requirements along this workflow, from one step to another, is another major challenge. In many design processes a design is communicated in the form of a document written in a natural language. Such documents can be thousands of pages long; they are, therefore, hard to understand, communicate, get right, process and manage. A similar difficulty emerges in derivational development, where one aims at deriving a new design from an existing one.
4. *Optimization of complex systems* Modern industry products are complex systems whose performance—such as torque, fuel efficiency and rattle noise of cars—depends on a number of diverse parameters. Tuning those parameters for optimal performance is often a highly nonconvex problem and hence is extremely challenging. In industry products such as cyber-physical systems, coexistence of discrete and continuous parameters poses another big burden.

Formal Methods for Computer Systems

Many of the challenges that we have just described—arising from the complexity of digital control—are already present in *computer systems*, that is, systems that solely consist of digital components. In this context a body of techniques called *formal methods* has been introduced and developed.

Formal methods is a broad term but it generally refers to mathematically based techniques for getting systems right. Their emphasis on symbolic formalisms brings both *rigor* and possibility of *automation* with the help of computers that operate on symbolic representations. One can also see formal methods as means for reasoning formally about correctness and efficiency of systems, where *logic*, as a branch (or a foundation) of mathematics, naturally plays an important role.

Typical goals of formal methods are *verification*, *synthesis* and *specification*.

- *Verification* Verification means to give a mathematical proof to correctness.¹ The idea comes naturally when one deals with software systems. There, programs and their behaviors are mathematical objects with rigorous *definitions*; therefore, it makes sense to state their correctness as *theorems* and to prove them much like in mathematics. More specifically, a verification procedure would take a *system model* $\mathcal{M}$ and a desired property φ of $\mathcal{M}$ —called a *specification*—as input. It would return either a proof for $\mathcal{M} \models \varphi$ (“ $\mathcal{M}$ satisfies φ ”) or a counterexample that shows $\mathcal{M} \not\models \varphi$. Two major approaches to verification are *model checking* by exhaustive search in finite state spaces and (automatic or interactive) *theorem proving*. An example of the latter is in Fig. 2.
- *Synthesis* Synthesis has been another major goal in software science. It aims at generating a suitable parameter set of a given system—or its scheduler, its controller or the system itself—that satisfies a given desirable property (i.e. specification). A fully automatic example (by graph algorithms) is in Fig. 3.
- *Specification* Specification, i.e. expressing a desirable property in a formal and symbolic way that is mathematically rigorous and operable by a computer, is vital in the above goals. It is an interesting topic on its own too, e.g. for requirement management towards consistency among different stages of development.

Formal methods have a potential of bringing a number of breakthroughs. In verification, for example, one could prove “the system $\mathcal{M}$ satisfies φ under arbitrary input i ”, in a computer-assisted way (sometimes fully automatically). Dealing with infinitely many potential input values and giving a mathematical proof as a certificate, verification provides a level of guarantee that is out of the reach of testing or simulation (i.e. trying finitely many test cases and obtaining empirical assurance).

Formal methods for computer systems have a long history. Some foundational observations can be found in early papers like [14, 27, 47]. For some time, formal

¹ This is the case in software science. In other fields and contexts the term *verification* is sometimes identified with *validation* and includes empirical quality assurance by testing. In this paper we shall clearly distinguish between the two terms.

Fig. 2 Program verification by program logic. Given a program as a system model $\mathcal{M}$ and a specification φ , one aims at constructing a proof for $\mathcal{M} \models \varphi$ that is given in the form of a symbolic proof tree in program logic (such as the Floyd–Hoare logic)

Fig. 3 Controller synthesis by automata. Given a finite-state automaton $\mathcal{M}$ as a system model, and a specification φ , we synthesize input to $\mathcal{M}$ that steers $\mathcal{M}$ desirably so that it realizes φ . This is by the following steps: 1) combining $\mathcal{M}$ and φ into a new automaton $\mathcal{M}_\varphi$ and 2) conducting suitable search in $\mathcal{M}_\varphi$. The latter is often a well-known graph-theoretic problem and we can exploit efficient algorithms

methods had been considered by industry practitioners “leisure activities” of researchers in academia, with most of their techniques not quite scaling up to the complexity of real-world software. The trend changed around the turn of the twenty-first century, however. Thanks to increasingly efficient algorithms together with faster computing hardware and identification of application domains where one can make differences, there have been a number of tools that have been successfully applied to real-world systems. Examples include various theorem provers and SAT/SMT solvers employed in verification of the design of microprocessors (see, e.g. recent [89]); sophisticated model checkers such as Uppaal [11], PRISM [61] and Spin [48]; the tool Astrée employed in verification of flight control codes of Airbus aircrafts [77]; the SLAM tool for verifying Windows device drivers [10]; a separation logic-based verification engine integrated in Facebook’s software development cycle [16]; and many more.

Formal Methods for Cyber-Physical Systems

It is, therefore, natural that application of formal methods is pursued beyond computer systems. Two main fields in which this is happening are industry products and biological systems; these two fields are combined to form a broad discipline of *cyber-physical systems* (CPS). Extensive research efforts have been made there in the last decade or so, with different emphases that reflect the *heterogeneous nature* of cyber-physical systems. For example, CPS Week—a major event in the CPS

research community—consists of several conferences each of which covers a major scope in the CPS research. These major scopes include *hybrid dynamics* that combine digital and discrete dynamics with physical and continuous dynamics, *real-time properties* and so on.

The formal methods community today is heading to cyber-physical systems, taking up the challenge of heterogeneity therein, trying to embrace new features such as *continuous dynamics* and *quantitative performance measures*. In the United States, partly thanks to the initiatives organized by NSF and other government agencies, rapid integration is going on between the formal methods community and the *control engineering* community—a discipline of long history that conventionally studies continuous dynamics. In Europe, formal methods techniques have been vigorously turned quantitative, so that they are sensitive to quantitative performance measures such as probability, elapsed time and energy. This is in contrast with the conventional view of formal methods where algorithms would give plain Boolean yes/no answers.

This trend of formal methods towards heterogeneity is hard to miss: in recent editions of major conferences in formal methods (like CAV, POPL and TACAS), typically one of a few keynote talks is given by a researcher pursuing heterogeneity in formal methods. Often this speaker is from the control engineering community, a “continuous counterpart” of formal methods.

The CPS community recognizes the potential role of formal methods, too, of bringing the same breakthroughs as they did to computer systems through specification, verification and synthesis. For example, application of formal methods is recommended in the international standard ISO 26262 for automotive functional safety.

Let us elaborate on typical challenges we encounter when formal methods try to embrace heterogeneity in cyber-physical systems.

Continuous and Hybrid Dynamics

The dynamics of computer systems are governed by clocks; consequently, their mathematical modeling—on which conventional formal methods techniques are based—has the *discrete* notion of time. This is in stark contrast with physical dynamics, such as position, velocity, and yaw/pitch/roll angles of a car, that are typically modeled with ordinary differential equations (ODEs) based on the *continuous* notion of time. Most cyber-physical systems combine discrete and continuous dynamics because of its digital and physical components. This combination is called *hybrid dynamics* and has been a major scope in the CPS research.

To extend formal methods techniques so that their scope encompasses hybrid dynamics, a natural idea is to incorporate ODEs in an (originally discrete) framework. This idea is indeed adopted in many successful extended frameworks. In *hybrid automata* [6] (see Fig. 4) one augments automata (a standard formalism for algorithmic analysis of discrete dynamics) with ODEs inside each state. Here the transitions between states represent discrete dynamics, while the ODEs in a state represent continuous dynamics within the discrete mode. Another well-known

Fig. 4 A hybrid automaton, with ODEs (for continuous dynamics) in states (for discrete modes)

example that follows the same idea is *differential dynamic logic* [72], where dynamic logic (a variant of the Floyd–Hoare logic, cf. Fig. 2) is extended with ODEs.

These extended frameworks have produced remarkable successes in modeling and analysis of hybrid dynamics. Theoretical studies of them, at the same time, have revealed fundamental gaps of complexity between hybrid dynamics and purely discrete ones. For example, it is known that the reachability problem in hybrid automata is undecidable [6]; this “no-go” theorem implies that most automata-based verification/synthesis algorithms for discrete systems (cf. Fig. 3) do not extend to hybrid systems.

- Those discrete algorithms achieve scalability via reducing a verification/synthesis problem to reachability, which is in turn efficiently determined by various graph algorithms.
- However, in the case of hybrid dynamics one cannot rely on this workflow because of the aforementioned undecidability result.

More generally, presence of continuous dynamics makes many problems significantly more difficult—so much that we should not expect the same push button-style algorithms (like automata-based ones) would solve those problems.

Quantitative Performance Measures

Conventional verification problems ask for a yes/no answer to the question if a system model $\mathcal{M}$ satisfies a specification φ . This *qualitative* view on systems, specifications and correctness is not necessarily suitable for cyber-physical systems.

For example, a CPS often involves *probabilistic* behaviors—they can come from physical components that operate in a stochastic manner (e.g. due to mechanical wear), human behaviors that we cannot precisely predict, randomized algorithms therein e.g. for fairness or security, OEM components whose behaviors are not totally known, and so on. For CPS *real-time properties* are also important: for a specification that asks a certain desirable event to eventually occur, one would be interested in how soon the event occurs, or if the occurrence meets a certain deadline. Various notions of *cost* like energy are critical in many CPS scenarios, too. For example, in synthesis, one would be interested not only in any controller that steers a system $\mathcal{M}$ so that a specification φ is satisfied—one’s true interest would be in the optimal controller that achieves φ at the smallest possible cost.

These *quantitative* performance measures (as opposed to qualitative, Boolean and yes/no ones) have been energetically integrated in various formal methods techniques, with a number of theoretical successes and practical tools. This is

probably because of the general tendency that a shift from qualitative to quantitative does not necessarily add significant complexity—which is in sharp contrast with the situation with continuous and hybrid dynamics (Sect. 2.3.1). For example, for Markov chains,² reachability probabilities can be efficiently computed via linear programming (LP). This fact opens up possibilities of various efficient verification/synthesis algorithms, as shown in [9]. For *weighted automata*, where transitions are labeled with weights (or costs) taken from a certain *semiring*, the situation is similar. Many problems reduce to solving linear inequalities in the semiring and often it can be done rather efficiently [25]. For *timed automata* [7], too, several variants of *zone construction* are known to yield finite-state abstraction of timed automata (see, e.g. [45]).

Our Unique Strategy: *Metatheoretical Transfer*

So far we have argued the following:

- manufacturing in modern days faces a big challenge as a result of computers in products;
- formal methods have potential to help the situation; and
- application of formal methods to industry products (as CPS) calls for significant research efforts, to cope with heterogeneity like continuous/hybrid dynamics and quantitative performance measures.

In the ERATO MMSD project that the current position paper is about, our mission is to explore formal methods' potential to help dealing with problems in design processes of real-world industry products.

There have already been quite a few large-scale organized efforts towards formal methods applied to CPS, and these programs have successfully brought notable research outcomes. So it is natural to ask the following question: what makes our endeavor special and interesting? Is it not just a latecomer after many similar projects? What unique scientific values can we expect from our project? What about practical and industrial values? In this section we shall describe our answers to these questions, by laying out a scientific strategy that we believe makes our project stand out. We shall call the strategy *metatheoretical transfer*.

Metamathematics

Our strategy of metatheoretical transfer draws its inspiration from metamathematics, a branch of mathematics that studies mathematical activities themselves as its subjects.³ That is to say:

² *Markov chains* are automata in which a next state is chosen stochastically.

³ The word *metamathematics* can have different meanings in different contexts. A common one bears a strong historical flavor, specifically referring to Hilbert's program and its position in mathematical philosophy. Another usage is as a synonym for mathematical logic. In this paper we shall interpret the word in its literal and plain meaning, namely as mathematical studies of mathematical activities.

- in mathematical physics one would model and study natural phenomena in mathematical languages (such as that of differential equations);
- in metamathematics, similarly, one studies mathematicians' activities (like setting axioms and writing proofs) in rigorous mathematical terms. *Mathematical logic* is the typical language used here.

To get a feeling see Fig. 5, it is meant to be an excerpt from a textbook in metamathematics (or mathematical logic). We notice that terms such as “definition”, “theorem” and “proof” occur multiple times in the figure, and that they designate two different notions that lie at two different levels. The object-level notions—such as proofs that are trees consisting of symbols—designate the activities of mathematicians as our study subject; and the meta-level notions—such as proofs written in a natural language—designate the activities of us metamathematicians who observe mathematicians. Metamathematics have brought about some striking breakthroughs, theoretically and practically—one of them is computers themselves, that arose from the effort by Turing, von Neumann and others to formalize mathematicians' capabilities.

Metamathematical Transfer

Our research strategy of metamathematical transfer is then explained as follows.

In most of the existing attempts towards heterogeneity and quantity, an individual technique in formal methods is extended, accounting for a single new concern at a time. See Table 1 for examples.

In contrast:

- we *parametrize* both a theory T , and a new concern e (time, probability, continuous dynamics, etc.), and
- establish a theory, from a meta-level point of view, on how one can incorporate e in the (object-level) theory T and obtain a new extended (object-level) theory T^e .

Here a theory (either on the object-level or the meta-level) refers to a body of definitions, lemmas and theorems.

A schematic overview is in Fig. 6. We first analyze the individual examples of extension of formal methods theories/techniques T_i with additional concerns e_i . By identification of some “mathematical essence”—that is, arguments and structures that are common to multiple of those individual extensions—we formulate a *metatheory* for such extensions $T + e \rightarrow T^e$. Concretely the metatheory consists of a general construction $T + e \rightarrow T^e$ that takes an original theory T and an additional concern e as input and yields the extended theory T^e as output. It is also desired that the metatheory ensures some correctness properties of the resulting theory T^e , such as soundness of the verification techniques in T^e . The general construction $T + e \rightarrow T^e$ would apply to various input T , e that is subject to certain conditions, yielding many concrete extensions $T_j + e_j \rightarrow T_j[e_j]$ for new input T_j, e_j .

Definition. A <i>proof</i> is a finite tree subject to the derivation rules:		
$\dots \quad \frac{\Gamma \Rightarrow \Delta, A \quad B, \Pi \Rightarrow \Sigma}{A \supset B, \Gamma, \Pi \Rightarrow \Delta, \Sigma} (\supset\text{-L}) \quad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \supset B} (\supset\text{-R}) \quad \dots$		
$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma} (\text{CUT})$		
A <i>theorem</i> is a formula A with a <i>proof</i> $\frac{\vdots}{\Rightarrow A}$.		
Theorem (cut elimination).		
Any <i>theorem</i> A has a <i>proof</i> $\frac{\vdots}{\Rightarrow A}$ in which the (CUT) rule is never used.		
Proof of the Theorem. Let Π be a <i>proof</i> of a <i>theorem</i> A . We turn Π into a cut-free <i>proof</i> Π' by induction. . .		

Fig. 5 Metamathematics. This is meant as an excerpt from a textbook in mathematical logic that covers Gentzen’s *sequent calculus* (see, e.g. [15]). Distinguish *object-level* notions (that are symbolic, i.e. syntactic constructs) from *meta-level* notions

Table 1 Existing approaches to heterogeneity in formal methods. Heterogeneity is achieved in a one-by-one manner, each time demanding substantial theoretical work

Original theory T (discrete)	+newconcerne $\Rightarrow$	Heterogeneous/quantitative theory T^e
Automata	+probability $\Rightarrow$	Various probabilistic automata such as Markov chains (Sect. 2.3.2) (see, e.g. [9])
Automata	+ODEs $\Rightarrow$	Hybrid automata [6]
Program logic	+ODEs $\Rightarrow$	Differential dynamic logic [72]
Safety by invariants	+ODEs $\Rightarrow$	Differential invariants [72]
(Bi)simulation	+continuity $\Rightarrow$	Approximate (bi)simulation [33]

The resulting theory/technique $T_j[e_j]$ may look different from any known theory/technique; however, brought by the general construction $T + e \rightarrow T^e$, the extension $T_j[e_j]$ shares some mathematical essence with the known extensions $T_i[e_i]$ that inspired the general construction. We may wonder at this stage if we could not have directly come up with the new extension $T_j[e_j]$ without going through the general construction $T + e \rightarrow T^e$. This is indeed possible, but the general construction often serves as a valuable guide. Another advantage is that the “correctness” of $T_j[e_j]$ (such as soundness of a verification technique therein) is most of the time automatically guaranteed, since it is already established at the abstract level of $T + e \rightarrow T^e$.

This metatheoretical approach to heterogenization of formal methods techniques is made possible by the mathematical languages of *category theory* and *logic*—quintessences of modern mathematics’ aspiration for structure, rigor, genericity and

Fig. 6 Metamathematical transfer, where we aim at a general construction (a meta-level theory) that takes an original theory T and an additional concern e as input, and yields the extended theory T^e as output

abstraction. These languages allow us to “define” T and e as mathematical entities, and to mathematically “construct” the extension T^e . This construction $T + e \rightarrow T^e$ (or $(T, e) \mapsto T^e$ to be more precise) is nothing but a (meta-level) theory of extending a (object-level) theory T with a new concern e , and the relationship between the last two “theories” is precisely like in metamathematics (Fig. 5).

A general construction $T + e \rightarrow T^e$ has practical benefits: it allows us to adapt existing formal methods techniques to heterogeneous application domains more quickly and comprehensively. This adaptability is all the more important now that the environments in which computers operate are rapidly diversifying. We will see many new concerns e , and a general recipe $T + e \rightarrow T^e$ makes us prepared for them.

Below in Sects. 3.3–3.4 we shall exhibit two examples of such uniform extension from T , e to T^e .

Metamathematical Transfer, Example I: Nonstandard Transfer from Discrete to Continuous/Hybrid

In our papers [44, 56, 78, 79] we introduced and elaborated a scheme called *nonstandard transfer* (NST in Table 2). It takes an arbitrary discrete formal technique T , and extends it to a technique T^{dt} for continuous and hybrid dynamics, a new concern that is significant in CPS. This extension is syntactically easy: one simply adds to (the logically formalized theory of) T a constant dt that designates an *infinitesimal* (infinitely small) value.

However, ensuring that the theory T^{dt} is consistent and meaningful is nontrivial. The use of infinitesimals is intuitive not only in our framework but also in formulating various notions in calculus—this is indeed the case in Leibniz’s formulation. Unfortunately, the existence of infinitesimals in the set $\mathbb{R}$ of reals

Table 2 A general extension methodology of *nonstandard transfer*, and its instances

	Original theory T	$\xRightarrow{+e}$	Extended theory T^e
NST (Sect. 3.3), <i>nonstandard transfer</i> [44, 56, 78, 79]	Arbitrary discrete FM technique T , formalized in first-order logic	$\xRightarrow{+dt}$	T^{dt} , an extension of T to continuous and hybrid dynamics
NST ₁ , an instance of NST [44, 78]	Imperative language and Floyd– Hoare logic	$\xRightarrow{+dt}$	While loops with infinitesimals, and their verification by inductive invariants
NST ₂ , an instance of NST [79]	Stream-processing language, Kahn-style denotational semantics [54] and program logic	$\xRightarrow{+dt}$	Language for processing continuous-time signals, its semantics and logic for verification
NST ₃ , an instance of NST [56]	Imperative language and reachability analysis by <i>abstract interpretation</i> [20]	$\xRightarrow{+dt}$	Reachability analysis of hybrid dynamics by abstract interpretation

causes contradiction. Assume a positive real δ is infinitesimal, that is, smaller than any positive real; then $\delta/2$ is even smaller than δ . This is why modern calculus employs arguments by ε and δ , and avoids infinitesimals.

It was Robinson's *nonstandard analysis* [73] that gave rigorous meaning to infinitesimals. This theory builds on formal logic and in particular results of *model theory* such as ultrafilters and ultraproducts, it extends the set $\mathbb{R}$ of reals to the set ${}^*\mathbb{R}$ of *hyperreals* that additionally contains infinitesimals, infinities, and so on. In Robinson's nonstandard analysis a positive infinitesimal is a hyperreal that is smaller than any *standard* real. Construction of hyperreals hinges on an ultrafilter, whose existence is usually shown via the axiom of choice.

This way nonstandard analysis offers a consistent theory in which reals and infinitesimals coexist. Even better, Robinson's nonstandard analysis offers the powerful *transfer principle* that states the following: a formula φ is true for reals ($\mathbb{R} \models \varphi$) if and only if it is for hyperreals (${}^*\mathbb{R} \models \varphi$). This means that a theorem obtained by usual deductive reasoning (for $\mathbb{R}$) is also a theorem in the extended setting of hyperreals (that include infinitesimals). In the original theory of Robinson, the formula φ in the transfer principle is taken from the first-order language of real arithmetic; our series of work [44, 56, 78, 79] can be understood as an attempt to accommodate program logics in nonstandard analysis.

In [44, 56, 78, 79] we have pursued uniform “hybridization” of discrete deductive verification frameworks (such as the Floyd–Hoare logic [27, 47] and abstract interpretation [20]) so that they also apply to continuous and hybrid dynamics. In these works we add an infinitesimal constant dt to a programming language so that it serves as a *modeling* language for continuous/hybrid dynamics. See Fig. 7, where the value of t continuously increases towards 1. We remark that the loop is executed infinitely many times—if it terminates within N iterations then this implies dt is not smaller than $1/N$, a contradiction. Therefore, the language **While** ^{dt} —an imperative language with while loops augmented with dt —is a

```

 $t := 0 ;$ 
 $\text{while } t \leq 1 \text{ do}$ 
 $t := t + dt$ 

```

Fig. 7 Continuous dynamics as a program. Here t increases continuously to 1

Table 3 A general extension methodology from automata to coalgebras

	Original theory T	$\xrightarrow{+e}$	Extended theory T^e
From automata to coalgebra, pursued e.g. in [39, 41, 43, 83, 85–87]	Theory of deterministic automata (that are F -coalgebras with $F = 2 \times (-)^{\Sigma}$)	$\xrightarrow{F=F_e}$	Theory of F -coalgebras as a general theory of state- based dynamics

modeling language rather than a programming language. Using nonstandard analysis, however, we can give rigorous denotational semantics for such programs (specifically the loop is iterated $1 + 1/dt$ times and the final value of t is $1 + dt$). Moreover, much like the transfer principle, we show that the same Floyd–Hoare logic works as well as in the discrete setting (cf. Fig. 2). The logic allows us to conclude, for example, that the value of t never exceeds 1.001 for the dynamics in Fig. 7. This theoretical framework of **While**^{dt} and the Floyd–Hoare logic is developed in [78], and in [44] we present an automatic theorem prover based on the theoretical framework.

The same workflow of adding dt to a programming language and using the same deductive verification technique (which is sound thanks to the transfer principle) has been pursued for other discrete frameworks, as we summarize in Table 2. This way we have established our nonstandard transfer workflow, a general extension scheme $T + e \rightarrow T^e$ where e is fixed to be continuous/hybrid dynamics (encoded by the infinitesimal constant dt) and T is arbitrary, as long as T is formalized in some first-order language. The resulting extension $T^e = T^{dt}$ looks quite different from other attempts of accommodating continuous/hybrid dynamics (see Sect. 2.3.1): ODEs do not occur in it, and a discrete deductive framework can be used *literally as it is*.

Metamathematical Transfer, Example II: from Automata to Coalgebras

In another example of general extension schemes $T + e \rightarrow T^e$, *automata*—a well-studied model of computation—are generalized to *F-coalgebras* for various choices of a *signature functor* F , see Table 3. Specifically we fix T to be the body of automata-theoretic techniques and leave e as arbitrary. Some e s allow encoding as a parameter $F = F_e$, and the resulting theory of F_e -coalgebras gives us “the theory of automata extended by e .”

The mathematical language that allows us to do so is that of *category theory*, an abstract formalism that is originally from algebraic topology but is nowadays used in various places in mathematics [8, 63, 66]. Category theory is centered around *arrows* instead of equations; see Table 4 for how notions in system theory are categorically formulated as coalgebras.

Table 4 Notions around system dynamics formulated in categorical terms

	categorically
system	$\begin{array}{c} FX \\ c \uparrow \\ X \end{array}$
behavior-preserving map	$\begin{array}{ccc} FX & \xrightarrow{Ff} & FY \\ c \uparrow & & \uparrow d \\ X & \xrightarrow{f} & Y \end{array}$
behavior	$\begin{array}{ccc} FX & \xrightarrow{F\text{beh}(c)} & FZ \\ c \uparrow & & \uparrow \text{final} \\ X & \xrightarrow{\text{beh}(c)} & Y \end{array}$ <p>where $\text{beh}(c)$ is the arrow induced by finality</p>

We shall briefly introduce the basics of coalgebraic modeling now, leaving further details to [50, 74]. Let $\mathbb{C}$ be a category and $F : \mathbb{C} \rightarrow \mathbb{C}$ be a functor. An F -coalgebra is a pair $\left(X, X \xrightarrow[c]{F} X\right)$ of an object X of $\mathbb{C}$ and an arrow $c : X \rightarrow FX$ in $\mathbb{C}$. For an example we can fix the base category $\mathbb{C}$ to be $\mathbb{C} = \mathbf{Sets}$, the category in which objects are sets and arrows are functions between them. We can further fix the signature functor F to be $F = 2 \times (_)^\Sigma$ where $2 = \{\text{tt}, \text{ff}\}$ is the set of Boolean truth values and Σ is a fixed alphabet. Then an F -coalgebra is nothing but a pair (X, c) of a set X and a function $c : X \rightarrow 2 \times X^\Sigma$. This is a coalgebraic modeling of deterministic automata (whose state spaces are not necessarily finite): X is the set of states and for each state $x \in X$, the value $c(x) = (b, f)$ tells us if the state x is accepting or not (whether $b \in 2$ is tt or ff), and its successors $(f(a) \in X)$ is the a -successor for each character $a \in \Sigma$.

F -coalgebras, by changing the parameter F , instantiate to various systems. This enables many results for (conventional) automata—they are identified with F -coalgebras for the specific choice $F = 2 \times (_)^\Sigma$, as we have seen—to be transferred smoothly to a variety of systems, leading to uniform verification methods that work for systems that are quantitative, probabilistic, timed, etc. alike. One among the notable successes is a uniform categorical definition of *bisimulation* by spans of coalgebra homomorphisms, see [50] for a comprehensive treatment.

This coalgebraic methodology has been actively pursued since mid-90s [2, 50, 51, 74]. In this context our unique standing is in the emphasis on *automatic* verification algorithms. For example, in [84, 86] we introduced *matrix simulations*, a probabilistic notion we obtained via a coalgebraic generalization of conventional simulations from [64]. It is demonstrated in [84, 86] that they allow effective algorithmic search by linear programming. Through this work and others [85, 86] we are now convinced of the following *coalgebraic unfolding* scheme (Fig. 8):

Fig. 8 Coalgebraic unfolding, using [86] as an example. Target problems (SAT, LP, etc.) are different, but the “unfolding” part is coalgebraically unified. Here our view on the decision problem SAT is that it is an optimization problem on the discrete domain $\{0, 1\}$

- For many formal methods problems, the key is to *unfold* complex structures and reduce them to (structurally) simpler, well-known problems such as satisfiability (SAT), linear programming (LP) and semidefinite programming (SDP, a well-known class of nonlinear convex optimization problems). For those well-known problems we can utilize known efficient algorithms.
- Different original problems have different target problems (Boolean to SAT, probabilistic to LP, etc.).
- However, the unfolding part exhibits structures of the same mathematical essence, and can, therefore, be unified, using coalgebraic abstraction (Fig. 8, the bottom row).

Our Tactics

We have described our strategy from a theoretical point of view. Another important point of our project is, on the practical side, to make real difference in manufacturing, at present and in the future. In this section we thus describe our tactics with which we align our theoretical research with industrial activities of designing high-quality products.

Technical Research Topics in Formal Methods Towards Heterogeneity

We shall first discuss some research topics towards heterogeneity in formal methods techniques. These topics naturally follow from our previous research activities and from our metamathematical transfer strategy (Sect. 3). Theoretical results in these topics are also expected to play pivotal roles in achieving the project’s goal.

Coalgebraic Model Checking that Unifies Qualitative and Quantitative

In the conventional theory of (discrete and qualitative) model checking, the following automata-theoretic workflow is widely employed. See, e.g. [88] and also Fig. 3.

- A target system is expressed by a finite-state automaton $\mathcal{M}$.
- A specification is expressed by a formula φ in *temporal logic* like LTL, CTL*, the modal μ -calculus, etc.
- To check if $\mathcal{M}$ satisfies φ (denoted by $\mathcal{M} \models \varphi$) we take the following steps.
 - One first translates the negated formula $\neg\varphi$ into an automaton $\mathcal{A}_{\neg\varphi}$. The translation is such that an execution trace σ satisfies $\neg\varphi$ if and only if σ is accepted by the automaton $\mathcal{A}_{\neg\varphi}$.
 - By suitable product construction one obtains $\mathcal{M} \otimes \mathcal{A}_{\neg\varphi}$, an automaton that accepts those execution traces which can arise from $\mathcal{M}$ and at the same time is accepted by $\mathcal{A}_{\neg\varphi}$.
 - Now one conducts the *emptiness check* for $\mathcal{M} \otimes \mathcal{A}_{\neg\varphi}$. If it succeeds (i.e. no traces are accepted by $\mathcal{M} \otimes \mathcal{A}_{\neg\varphi}$), we conclude that there are no execution traces of $\mathcal{M}$ that satisfy $\neg\varphi$, that is, $\mathcal{M} \models \varphi$.

Notable in the above workflow are the following facts. First, it heavily relies on automata—finite-state representation of dynamics—as one can exploit many known algorithms for their analysis and many known constructions for operating on them (e.g. Boolean operations). Second, to accommodate reactive and nonterminating behaviors and *fixed-point* specifications on them (such as safety, liveness, recurrence, persistence), the automata in the workflow must operate on infinite words (as opposed to finite words, e.g. in [49]). They consequently use seemingly complicated acceptance conditions like the Büchi and parity ones (see e.g. [37, 88, 92]). Lastly, in this verification workflow and in others (e.g. for synthesis, Fig. 3) the original problem is eventually reduced to emptiness check, a problem that allows an efficient algorithmic solution.

Automata-theoretic techniques in formal methods (such as the above example) have been energetically extended to *quantitative* settings—especially probabilistic ones. This is as we discussed in Sect. 2.3.2. In the probabilistic version of the above automata-theoretic workflow the greatest difference is that emptiness check, to which the original problem is reduced to, is replaced by calculation of acceptance probabilities. The latter is further reduced to calculation of *reachability probabilities* via notions like bottom strongly connected components, and reachability probabilities are typically solved by linear programming (LP), for which many efficient solvers are available (see, e.g. [9]).

From this trend in formal methods and also from our metamathematical transfer strategy (especially its instance of coalgebraic unfolding, see Sect. 3.4), it arises as a natural research question to *extract essence of automata-theoretic model checking* that underlies both the qualitative and quantitative settings. We aim at formulating such common structures in categorical (especially coalgebraic) terms.

In this direction we have obtained some promising preliminary observations. In [43] we presented the notion of *lattice-theoretic progress measure*, a general notion of witness for nested and alternating fixed-point specifications, and we applied it to coalgebraic model checking. The notion generalizes *invariants* for greatest fixed-point specifications (like safety) and *ranking functions* for least fixed-

point specifications (like liveness and termination). It also generalizes Jurdzinski’s combinatorial notion of progress measure [53] for algorithmic solution of parity games; based on general complete lattices, our generalized notion can also be employed in quantitative settings. In [19] we present preliminary results towards coalgebraic linear-time model checking that encompasses quantitative settings.

Quantitative Semantics for Enhanced Expressivity

Another research direction on quantitative logics that we wish to pursue is quantitative semantics for *enhanced expressivity*.

One example of such semantics is temporal logics *with discounting* [4, 5, 70], where events in the farther future have the smaller significance. For example, for the LTL specification Fp (“ p eventually becomes true”) we have the following difference:

- in the usual Boolean semantics the formula is true no matter how long it takes before p is true for the first time;
- in contrast, in the quantitative discounted semantics of [5], the formula’s truth value is $1/2^i$ in case p is true for the first time at time i .

One can argue that the latter semantics is more expressive and potentially more faithful to a designer’s intention.

Another example of such quantitative semantics is *robustness semantics* of temporal logics for continuous-time signals [3, 24, 26]. There the truth value of a formula φ is a real number that designates how robustly the formula is satisfied. This theoretical idea has opened up the field of *falsification by optimization*, a scalable methodology that attracts attention in quality assurance of CPS. See Fig. 10 and the discussion later in Sect. 4.2.1). Specifically, the original work [26] addressed *space/vertical robustness* (e.g. the bigger the value of x , the more robustly the formula $x > 0$ is satisfied); the work [24] introduced *time/horizontal robustness* (e.g. a deadline specification is more robustly satisfied if the desired event occurs way in advance). The work [3] proposes to combine space and time robustness by suitable integration.

We wish to further pursue the study of quantitative semantics of specification logics. We have to say the role of metatheories in this direction is less clear than in coalgebraic model checking—perhaps the flexibility in the choice of truth value domain in coalgebraic logics can be exploited (see e.g. [40, 43, 46]). In any case specification logics and their expressivity will be central in our work towards quality assurance of CPS.

Simulation and Bisimulation Notions

The notions of *bisimulation* and *bisimilarity* were originally introduced as a definition of equivalence between concurrent processes [68, 71]; they require two processes to be able to mimic each other’s actions and, moreover, to be able to continue doing so. Their one-sided variations are the notions of *simulation* and *similarity*, in which one process is required to mimic the other, but not conversely.

These notions of (bi)simulation have been extensively studied and used over years. A notable use of them is as a (sound but not necessarily complete) *proof method*. Consider nondeterministic finite automata (NFA), for example; there are many different possible definitions of their “behavior”, as organized in the so-called *linear time–branching time spectrum* [35]. A standard definition (*language equivalence*) is given by accepted languages; another standard one (*bisimilarity*), given by bisimulation, is strictly finer than language equivalence since it also takes internal branching structure into account.

Here one possible way to look at the situation is that one can use bisimulation as a “proof method” for language equivalence. A bisimulation is a binary relation between states subject to suitable “mimicking” conditions, therefore finding a bisimulation witnesses language inclusion. It is sound (since bisimilarity is coarser than language equivalence), although it is not complete (since there are processes that are language equivalent but not bisimilar). Bisimulation as a proof method is often more computationally tractable, too, since its mimicking conditions are formulated in a *local*, one-step manner. This is in contrast to language equivalence, a *global* notion that deals with unboundedly long words and thus arbitrarily many steps.

Bisimilarity as a proof method for language equivalence may not sound too appealing since language equivalence for NFA is decidable.⁴ The situation is different when one moves to quantitative settings. For a probabilistic variant of NFA, *language inclusion* (i.e. the one-way variant of language equivalence) is known to be undecidable [12]; therefore, one needs to rely on incomplete proof methods, among which is *simulation* (i.e. one-way bisimulation). It is our contribution in [86] to exploit our coalgebraic characterization of simulation [39] and formulate quantitative simulation in terms of matrices. The latter notion of quantitative simulation allows efficient search by linear programming, see Fig. 8.

Use of (bi)simulation has been advocated in the context of CPS and more specifically in hybrid systems, too. In [33] the notion of *approximate bisimulation*—in which behavior discrepancy is tolerated up to a prescribed threshold ε —is introduced, and it has been extensively studied and applied since. One of the main applications is discrete abstraction of continuous dynamics: via an approximate bisimulation one establishes that a certain discrete finite-state system is an adequate abstraction of an original continuous dynamics. The finite-state abstraction can then be used for various purposes such as verification and controller synthesis (see, e.g. [34]).

In our ERATO MMSD project we will pursue both theoretical studies and practical applications of various (bi)simulation notions. For one, in the theoretical direction, notions for hybrid dynamics like approximate (bi)simulation have not yet been subject to categorical/coalgebraic studies⁵ that would help us organizing different (bi)simulation notions and also formulating relationship to specification logics. For another, in the practical direction, we believe the role of (bi)simulation

⁴ The problem is, however, PSPACE-complete, and bisimulation can actually be used as a faster but incomplete method to establish language equivalence (see, e.g. [13]).

⁵ A notable exception is [38], but their categorical approach is based on open maps [52], not on coalgebras.

$$\frac{\{A\} c_1 \{C\} \quad \{C\} c_2 \{B\}}{\{A\} c_1; c_2 \{B\}} \text{ (SEQ)} \quad \frac{A \supset A' \quad \{A'\} c \{B'\} \quad B' \supset B}{\{A\} c \{B\}} \text{ (CONSEQ)}$$

Fig. 9 Two rules from the Floyd–Hoare logic

as an incomplete but tractable proof method is even more important for CPS and hybrid dynamics. There many problems are far more difficult than in conventional discrete settings. For hybrid automata even the basic problem of reachability is undecidable [6]; this means the automata-theoretic model checking workflow in Sect. 4.1.1 is unlikely to work. Therefore, we will be relying more often on incomplete proof methods by various (bi)simulation notions.⁶

Compositionality

One of the principles advocated in formal methods is *compositionality*: it is desirable if the analysis of a large system can be conducted in a bottom-up way, based on the analysis of its constituent parts. Such a divide-and-conquer approach not only makes the whole analysis task tractable, but also allows modular analysis where one can reuse part of the analysis when the system is partially updated. The modularity is particularly important in our project, because we aim at assisting *existing design processes* where many components are black-box (see Sect. 4.2 later).

Let us look at a simple example. In the Floyd–Hoare logic one derives *Hoare triples* $\{A\} c \{B\}$: it means “after any successful execution of the program c under the precondition A , the postcondition B is guaranteed”. In Fig. 9 we find two rules of our current interest. The (Seq) rule is about the sequential composition $c_1; c_2$ of two programs c_1 and c_2 : if c_1 guarantees C assuming A and c_2 guarantees B assuming C , then $c_1; c_2$ guarantees B assuming A . The (Seq) rule is thus an exemplar of compositional reasoning. The (Conseq) rule says that one can strengthen preconditions and weaken postconditions.

For example, the following is known (see, e.g. [93]): in the setting of the usual imperative programming language and pre- and postconditions expressed in first-order logic, the *weakest precondition* $wp\llbracket c, B \rrbracket$ for a given program c and a postcondition B can be expressed in first-order logic. The weakest precondition $wp\llbracket c, B \rrbracket$ is defined to be such a formula that (1) the Hoare triple $\{wp\llbracket c, B \rrbracket\} c \{B\}$ is valid; (2) it is the weakest among such, that is, $\{A\} c \{B\}$ implies that the formula $A \supset wp\llbracket c, B \rrbracket$ is valid.

Thus, one may wonder if it is ever needed to invoke the (Seq) rule in Fig. 9 for C other than $wp\llbracket c_2, B \rrbracket$. We need the second assumption $\{C\} c_2 \{B\}$ to be valid, which means $C \supset wp\llbracket c_2, B \rrbracket$ is valid. Replacing C with a weaker assertion $wp\llbracket c_2, B \rrbracket$, there is a better chance of establishing $\{A\} c_1 \{wp\llbracket c, B \rrbracket\}$ than the original second assumption $\{A\} c_1 \{C\}$.⁷

⁶ Recall that our view on (bi)similarity is that it is a sufficient (but not necessary) condition for linear-time equivalence/inclusion.

⁷ In reality it is indeed necessary to come up with and use simple C that is different from $wp\llbracket c, B \rrbracket$. Following the general recipe (see, e.g. [93, Chapter 7]), the first-order formula for $wp\llbracket c, B \rrbracket$ is monstrous. It is very hard to use such complicated formulas in Hoare logic proofs, such as in deciding validity of first-order formulas $A \supset A'$ and $B' \supset B$ in the (Conseq) rule.

In the case of hybrid dynamics there is little hope of symbolically expressing weakest preconditions. For example, assume that the program c contains continuous dynamics governed by an ODE. Then to symbolically express the weakest precondition $wp[c, B]$ in general, we would need a closed-form solution for the ODE, which is not available most of the time. This means, in the application of the (Seq) rule, we need careful “negotiation” between the verification task of $\{A\} c_1 \{ _ \}$ and that of $\{ _ \} c_2 \{B\}$, so that we come up with a good “contract” C . The choice of C should balance the difficulty of establishing the two assumptions.

In the study of CPS and hybrid systems, pursuit of compositionality via a certain form of “contract” is nowadays widespread [29, 57, 90, 94]. One use of contracts typical of CPS is for separating safety from efficiency: the system architecture is hierarchical with the higher, symbolic level addressing safety and the lower, numeric level addressing efficiency; the higher level determines high-level policies such as discrete actions (as in *maneuver automata* [29]) or so-called *safety envelopes* (as in [90]); and the lower level synthesizes a controller that complies with the high-level policies and at the same time aims at efficiency. One possible view is that the high-level policies are contracts through which low-level controllers collectively achieve safety of the whole system.

The study of such hierarchical system architectures from the computer science viewpoint is a topic of our interest. Mathematically speaking, system composition is *algebraic*—the field of *process algebra* in theoretical computer science [1] is devoted to algebraic composition of state-based dynamics. The relationship to modal logics as specification languages is well studied, too. Some of these results there have seen their categorical generalization [42, 58, 81], where system dynamics and their algebraic composition together form a categorical notion of *bialgebra*, and modal logics are nicely accommodated in the categorical picture via *Stone-like dualities*. We shall start with adapting these existing results to CPS examples, also using our results in the topics we discussed in Sects. 4.1.1–4.1.3.

Later in the project we shall also study *security* of CPS. Security properties occupy special status in formal methods since they are notoriously non-compositional. We shall base ourselves on recent results on compositional security, including [22].

Collaboration and Integration with Control Theory and Robotics

The existing efforts towards extension of formal methods (FM) to CPS have been centered around their integration with *control theory* (CT) and *control engineering*—as we described in Sect. 2.3. There have been a number of techniques from the two fields that have proved to have the same mathematical essence, and similarly, a number of techniques exported from one field to the other. The following examples are known: the correspondence between *ranking functions* (FM) and *Lyapunov functions* (CT) (see, e.g. [17]); that between *invariants* (FM) and *barrier certificates* (CT); notions of *robustness* exported from CT to FM (see, e.g. [26]); and use of *(bi)simulation* (as witness for behavioral inclusion/equivalence) and *modal logics* (as specification languages) exported from FM to CT (see, e.g. [67]). Another

notable trend is use of numeric optimization algorithms in FM: it has been one of the basic tools in CT (see, e.g. [80]); nowadays a lot of problems in FM are solved by reducing to continuous optimization problems, too.

We will naturally follow the same strategy of close integration between formal methods and control theory/control engineering. Here our strategy of metamathematical transfer (Sect. 3) has potential to be advantageous. Very often, identification of commonalities between FM and CT techniques is done in the level of intuition; this means that, in the actual transfer of a theory from one field to the other, one needs to build a new theory from scratch, contemplating definitions and statements so that the new theory is consistent and useful. Having the existing theory in the other field as a guide is a big advantage, but the actual task of transfer is far from mechanical or trivial. This is in contrast with our metatheoretical methodology in which the (object-level) theory is formalized as a mathematical entity. Once we establish a (meta-level) theory for transfer, the task of transfer becomes mechanical, and the “correctness” of the resulting theory is trivially guaranteed. We have seen example of such transfer in Sects. 3.3–3.4.

Identification of Suitable Target Problems: We Start Small, Think Big

After going through the technical research topics in Sect. 4.1, we are now convinced of the following point: with CPS, we should not expect that many familiar problems (such as verification and synthesis) be solved as efficiently as with the conventional discrete settings. Once we have continuous dynamics in a system the following happen: (1) automata-theoretic techniques hardly work because reachability in hybrid automata is undecidable; (2) deductive techniques would typically call for closed-form solutions of ODEs, which are a rarity. In the shift from qualitative to quantitative (probability, time, etc.) linear programming (LP) works as an efficient counterpart of graph algorithms in the discrete and qualitative setting; this looks like a fortunate exception.

Another challenge that we face in heterogenization of formal methods techniques is *scarcity of formal specification*. This has been already identified as a major challenge for software systems, and various techniques for aiding formal specification have been in the software engineering community (see, e.g. [59]). Unfortunately the situation seems to be far worse in CPS. Formulating requirements and specifications for CPS calls for a lot of domain expertise in their physical components such as car engines; reflecting such domain expertise in a logical specification formalism is much harder a task than formalizing software designers’ intentions. In addition, in view of the complexity of modern industry products and the size of their design processes, it seems impossible to have a comprehensive formal specification for some product that would underlie its design process that is totally formalized.

Our ERATO MMSD project emphasizes application of its research results in industry—we wish to make real difference in real-world applications, which in turn would stimulate further theoretical work too. This forces us to take a pragmatic approach to formal methods applied to CPS. We do not take an abrupt approach in which we would insist every step in CPS design be formal, starting from a

comprehensive formal specification. Instead we start small: we first identify small portions of existing design processes of industry products in which ideas from formal methods can be helpful. The resulting practical benefits might look small, say the cost of design is suppressed by one percent. This seemingly marginal cost cut can add up to a big whole, however, if one imagines how many cars are currently built in a year. Moreover, accumulation of small “success stories” will give further momentum to formal methods applied to CPS, and hopefully lead to more systematic efforts towards formal specification in CPS design.

In our pragmatic approach to application of formal methods to CPS, there are two challenges as we discussed in earlier paragraphs, namely infeasibility of traditional formal methods goals (like verification and synthesis), and scarcity of formal specification. We shall cope with them by the following two principles: *testing rather than verification* (described below) and *compositionality* (Sect. 4.1.4).

Testing rather than Verification

Testing is currently a principal quality assurance means for software, and there have been a lot of work towards enhanced efficiency and confidence (including [65]). For CPS testing is a principal means, too. Since formal verification (i.e. mathematical proofs for correctness under every possible input) is hard for CPS as we discussed, we naturally turn to improving testing for CPS. At the same time, testing for CPS seems to be an area that is still under development, when compared to testing for software that is a mature area with a big body of literature and efficient tools.

Another advantage of testing rather than verification, at the current early stage of deployment of formal methods in CPS design, is that a bug is more “useful” than a correctness proof. To appreciate a correctness proof calls for backgrounds in mathematics and logic, as well as understanding of all the assumptions that the proof is based on. For this reason it is often hard to convince real-world engineers of the benefits of formal verification. In contrast, a bug of a system speaks for itself—it is something which a system designer can immediately work on.

We believe that study of formal methods can bring many benefits to testing too: even from a technique for verification (not for testing), we can extract ideas that can be used to make testing more efficient. For example, in [91] we successfully employed a few constructions for timed automata—originally devised for verification—for the purpose of monitoring real-time behaviors (monitoring can be understood for an oracle for testing). Our orientation towards abstraction (Sect. 3) helps, too: we always aim to identify mathematical essence in various formal methods technique; such essence can then be used for other purposes (like testing), independent of its original goal.

In this direction of testing we shall discuss three concrete research topics. One is *monitoring* of real-time behaviors that we already mentioned. In monitoring, given a log of events and a specification, one seeks all the segments of the log that matches the specification. This may sound like much simpler a task than verification and synthesis—the latter deal with all the possible input while monitoring deals with a single log. However, efficient monitoring of a number of big logs in embedded applications is a nontrivial matter. Monitoring is an important building block of

Fig. 10 Robustness semantics and falsification by hill-climbing

testing frameworks, too, since we need to judge if an execution of a system is erroneous or not. Recent works on real-time monitoring include [55, 82, 91].

Another concrete topic in testing is *search-based testing*, and more specifically *falsification by optimization*. The work [26] pioneers this topic, in which the quantitative robustness semantics (Sect. 4.1.2) allows us to find an error input (a “bug”) by hill-climb style stochastic optimization (Fig. 10). Also with the recent surge in advancement of techniques of machine learning and stochastic optimization, falsification by optimization attracts much attention as a scalable method for CPS quality assurance. We shall continue working on this topic, exploiting our expertise in logic (like in [3]) and also collaborating with machine learning and optimization techniques. A work related in essence is [31] for satisfiability check of assertions that involve floating-point operations.

Yet another concrete topic related to testing is *statistical model checking*. It is similar to testing—one samples some input values and observes if the system operates correctly—but in statistical model checking a bigger emphasis is on estimation of likelihood of errors and its statistical confidence. There one would rely on statistical arguments like Chernoff-like bounds and Bayesian inference (see, e.g. [62]).

Collaboration with Machine/Human Intelligence

One rough way of looking at various problems in quality assurance of systems is that they are search problems: in testing one searches for bugs, in verification one searches for proofs and in synthesis one searches for a suitable parameter or controller. Other search problems that we have discussed include the following: search for (bi)simulations as witnesses for behavior inclusion/equivalence (Sect. 4.1.3) and search for a mediating condition (“contract”) in compositional reasoning (Sect. 4.1.4).

Automata-theoretic approaches (sketched in Sect. 4.1.1) can be understood as reduction of various problems to the reachability problem, i.e. searching for a path in a graph to a certain set of states. Being one of the structurally simplest search problems, the reachability problem allows for efficient algorithmic solutions by exhaustive search. Nevertheless, sometimes the state space is too large for exhaustive search already in discrete settings where we deal with software systems. This is the case when a state of the system in question is determined by a number of

variables—the number of states is exponential in the number of variables. This is the phenomenon called *the curse of dimensionality*.

Continuous dynamics in CPS poses an obvious challenge: a state space is typically (uncountably) infinite, thus exhaustive search is no longer an option.

All these call for efficient search methods in a space that is too big for exhaustive search. We identify this as one of the key challenges that affect the efficiency of various techniques that we shall be developing. In our project we will in particular try exploiting the following machine/human intelligence: (1) *numeric* and *statistical* methods from mathematical optimization and machine learning and (2) *human expertise*, such as that of engineers who have worked on car engines for years. These two shall altogether be called *machine/human intelligence*.

Numeric and Statistical Methods

Numeric methods from mathematical optimization (such as the interior point method for convex optimization) and statistical methods from machine learning (such as simulated annealing, Monte Carlo sampling, Gaussian process learning, and so on) are widely known countermeasures for the curse of dimensionality. Their use for formal methods for CPS has been pursued, too: falsification by optimization (Sect. 4.2.1) employs statistical optimization, as we already saw. Reduction of problems to convex optimization is increasingly popular too [17, 21, 76]. This reflects a common workflow in control engineering, where various control problems are reduced to mixed integer optimization problems (see, e.g. [80]).

In our project we will aim at systematic integration of these methods in formal methods techniques. There are two issues here: *numeric errors* and *statistical confidence*.

Numeric Errors Numeric errors are inherent in numeric and stochastic methods. In a typical workflow we reduce the original problem P to an optimization problem P' , and the latter is solved, say, by the interior point method. Its numeric solution x for P' is subject to numeric errors, and it can be the case that the numeric solution x for P' is not a valid solution for P . In many scenarios P is a symbolic problem (like verification) and such error is not tolerated.

Even if x is a valid solution of P , there is another question of the “quality” of a solution. Assume P is a problem whose solution is a polynomial inequality, and that both of the inequalities $1.0013x^2 + 2.208 \times 10^{-9}xy > 0$ and $x^2 > 0$ are solutions. One can say that the latter solution $x^2 > 0$ is more desirable—in terms of human understanding and also for the purpose of symbolic reasoning that uses a solution of P . However, relying on numeric solvers, what we get looks more often like the former complicated solution. This challenge—which our colleague Takamasa Okudono calls the problem of *interpretability*, borrowing the term from machine learning—is identified in [75] and has been raised several times ever since, but without a systematic solution to the best of our knowledge. We shall strive for a systematic solution that combines (1) checking validity of numeric solutions and (2) perturbing numeric solutions, if necessary, for validity and better interpretability.

Statistical confidence This is a problem inherent in heuristic methods like stochastic sampling: if a solution is found then it *is* a solution; if a solution is not found, however, this does not guarantee absence of solutions. One thing we can do here is to polish up the method so that it will find solutions more often; the other thing is statistical inference that tells, based on the observed absence of solutions, the confidence with which we can conclude that no solutions exist. See [23] for an example of works in this direction. We shall combine results and observations in statistical model checking (Sect. 4.2.1) to systematically study confidence bounds for statistical methods in formal methods for CPS.

Human Expertise

Numeric and statistical methods can be understood to put suitable bias on otherwise exhaustive and blind search. The same role can be played by human specialists, exploiting their domain expertise. This makes even more sense in the CPS context, where systems are currently designed by experts with ample experience and deep knowledge (although they are stored and communicated informally). We shall, therefore, pursue formal methods frameworks that have *human-in-the-loop*—not only in the sense that their target systems encompass humans (drivers and pedestrians in autonomous driving, for example), but also in the sense that for analyses such as testing, specification, verification and synthesis, the frameworks interact with human specialists and exploit their expertise. In doing so, a key is user interface (UI); see [69] for an example where spreadsheets—a formalism familiar to wide audience including CPS engineers—are used as a modeling formalism.

Formal Methods “for” Intelligence

So far we have discussed how we will be using “intelligent bias” in search problems in formal methods; its source can be numeric and statistical algorithms, or human specialists. Here we discuss the other direction, that is, how formal methods can be used to help intelligence, especially *machine learning* (ML) and *artificial intelligence* (AI).

In the recent rapid advancement of the autonomous driving technology, a notable trend is to let ML/AI learn how to drive, using, e.g. deep neural networks. An obvious challenge here is safety guarantee of such learned driving algorithms: learning algorithms like neural networks are often black-boxes, in which case there is no telling what a car does in safety-critical corner cases.

In our ERATO MMSD project we will pursue formal methods techniques that address correctness guarantees of ML/AI algorithms. One direction of doing so is to analyze the ML/AI algorithms themselves, for which the theory of *probabilistic programming languages* (see, e.g. [36]) and their categorical abstraction (see, e.g. [40, 46]) can be useful. Another possible direction is to regard ML/AI algorithms as black-box components of a whole system, and to try to guarantee that the whole system is safe no matter how those black-box components behave. This hierarchical view on systems is much like what we discussed in Sect. 4.1.4.

Putting to Real-World Use

Here we describe how we put our research results to real-world use.

Incremental Support of CPS Design Processes

As we described in Sect. 4.2 our goal is to make real differences, if small, in real-world CPS design. They will provide useful feedback that will drive further theoretical developments; given the scale of the market of CPS, even a fractional improvement can yield a big difference in total; and accumulating small success stories will pave the way to more comprehensive and systematic attempts to integrate formal methods in the design processes of industry products.

For such an *incremental* approach to formal methods in CPS design, it is important to work with actual practitioners closely, so that the collaboration leads to identification of actual issues in which formal methods can be of help. In software engineering there are a lot of mature techniques and tools; therefore, for many industrial issues there already exist tools that readily address those issues. This is not the case with CPS, however; conventional problems in formal methods are much harder with CPS (Sect. 4.2) and, therefore, it is unlikely that an efficient tool is available to an obvious problem in industry. Consequently, the issues for which we can be of help will be small and subtle; to identify them we theoreticians need to listen carefully to practitioners.

We believe our orientation to abstraction and genericity is an advantage in doing so. To focus on mathematical essence in a formal method technique is to understand how and why the technique works; such understanding of the working mechanism would then allow us to apply the technique to a seemingly different but essentially similar problem.

Supporting Software-Centric CPS Design Processes

In the above incremental formal methods in support of existing CPS design processes, we observe the existing design processes and find small portions of them in which formal methods can be of help. Besides that, in our ERATO MMSD project we aim to contribute to CPS design processes *to come*, too.

Recent industry trends have turned many industry products into commodities—these trends are notable, e.g. with smartphones and television sets. In this process design processes become more software centric. It is natural to expect the same to happen to many other industry products, too. An example is cars, whose functionalities are dramatically changing due to the advancement of autonomous driving. As to such safety-critical applications as cars, their software-centric design processes should require a high level of safety guarantee—one that is only realizable with formal methods.

To explore the shape of such new CPS design processes, and the roles of formal methods in it, our project collaborates closely with the *Autonomoose* project⁸ at the

⁸ <http://www.autonomoose.net/>.

University of Waterloo. Unlike incremental help of already established CPS design processes (Sect. 4.4.1), with the Autonomoose project we collaborate in an early stage where an autonomous driving system is being built from scratch. This will allow us to try more exploratory techniques.

The ERATO MMSD Project

The strategy of metatheoretical transfer (Sect. 3) will be implemented in the ERATO MMSD Project through the tactics described in Sect. 4. The organizational structure of the project—consisting of four research groups—reflects the strategy and tactics.

- Group 0, called *Metatheoretical Integration Group* and led by Shin-ya Katsumata, aims to formulate the extension processes $T + e \rightarrow T^e$ themselves in rigorous mathematical terms (Sect. 3). Relevant technical fields are mathematical logic and category theory as languages for describing mathematical theories. A remote site of Group 0 is at RIMS, Kyoto University that is led by Masahito “Hassei” Hasegawa.
- Group 1, called *Heterogeneous Formal Methods Group* and led by the author, aims to conduct the extension $T + e \rightarrow T^e$ for a variety of T s and e s. The extension will be guided by real-world applications as well as by our metatheories (i.e. general extension schemes); conversely, developments in Group 1 will also provide concrete extensions $T + e \rightarrow T^e$ that will then inspire their metatheoretical unification conducted by Group 0. Collaboration with control theory—a discipline that is complementary to computer science in the study of CPS—is a key aspect of the activities of Group 1. A remote site of Group 1 is at Osaka University that is led by Toshimitsu Ushio.
- Group 2, called *Formal Methods in Industry Group* and led by Krzysztof Czarnecki, is based at the University of Waterloo, where a number of strategic initiatives are taking place towards collaborations with automotive industry (such as WatCAR and Autonomoose). The group’s mission is to pursue effectiveness of the theories and techniques developed in the project (especially by Groups 1 and 3). It applies the techniques to real-world applications and at the same time identifies key theoretical challenges that are then fed back to the other groups. Out of the two directions of real-world applications in Sect. 4.4, Group 2 is more focused on software-centric CPS design processes (Sect. 4.4.2), while the other direction (incremental support of existing CPS design processes, Sect. 4.4.1) is pursued by all the groups altogether.
- Group 3, called *Formal Methods and Intelligence Group* and led by Fuyuki Ishikawa, pursues collaboration between formal methods and machine/human intelligence (Sect. 4.3). The group’s goal is similar to that of Group 1—namely concrete extensions $T + e \rightarrow T^e$ —but Group 3 approaches it principally via the fields such as *software engineering*, ML/AI and user interface (while Group 1 is via software science and control theory).

This project organization allows researchers with different backgrounds from formal methods to control theory, software engineering, logic and category theory, to closely collaborate with each other. This helps Group 0's efforts to identify theoretical essence that is common to different academic fields. This heterogeneous project organization also helps to provide multiple views on concrete industry problems so that we can come up with a bigger variety of solutions to them.

Conclusions and Future Perspectives

We have described the ERATO MMSD Project: its contexts (complexity of industry products under computer control) and the challenges it aims to address (formal methods applied to CPS), our strategy of *metatheoretical transfer*, and how we implement the strategy so that our distinctively theoretical (or metatheoretical) approach also brings real differences to real-world problems. We emphasize that our approaches are incremental: our results will be put to use not only in verification but also in testing; in doing so, our theoretical approach is beneficial since it allows transfer of ideas to seemingly different problems. Exploitation of *intelligence*—such as ML/AI and human expertise—is featured, too.

In industry application, we start small, identifying small portions of existing design processes in which our techniques can be of help. We hope that by the end of the project (March 2022) we will have seen several such “success stories”—they will give further momentum to formal methods applied to CPS, hopefully leading to formal methods in CPS design as industry practice.

From an academic viewpoint, the project is a unique venue where the integration of software science and control theory—two major disciplines for analysis of dynamics—is pursued on the solid basis of logical and categorical metatheories. We believe our attempt is special, too, when seen as an instance of *applied mathematics*. Modern mathematics with its orientation towards abstraction and genericity is not very often associated with concrete applications. In our project, through our use of logic and category theory, we wish to make a case that abstraction is power in application: by abstraction we obtain a theory that generalizes a known, more specific one; and this general theory makes us be better prepared against challenges that are yet to come. Such power of abstraction is all the more important in the modern world where technological, industrial and social situations around us are changing so rapidly.

Acknowledgements Thanks are due to the following people: the core members of the ERATO MMSD project (Krzysztof Czarnecki, Masahito Hasegawa, Fuyuki Ishikawa, Shin-ya Katsumata, Kohei Suenaga, Toshimitsu Ushio) for intensive discussions on the project goals, strategies and tactics; the author's collaborators and students (including Takumi Akazaki, Kenta Cho, Yi Chou, Gidon Ernst, Noemie Fong, Soichiro Fujii, Masaki Hara, Wataru Hino, Naohiko Hoshino, Bart Jacobs, Toshiki Kataoka, Kengo Kido, Hiroki Kobayashi, Yoshihiro Kumazawa, Satoshi Kura, Tetsuri Moriya, Koko Muroya, Shota Nakagawa, Hiroshi Ogawa, Ibuki Okamoto, Takamasa Okudono, Yuichiro Oyabu, Sasinee Pruekprasert, Sean Sedwards, Hiroyoshi Sekine, Eugenia Sironi, Shunsuke Shimizu, Ryo Tanaka, Natsuki Urabe, Masaki Waga, and Akira Yoshimizu) for discussions and joint works that have inspired the project; Kazuyuki Aihara, Masami Hagiya, Takahiro Homma, Shinichi Honiden, Jun-ichi Imura, Shinpei Kato, Naoki Kobayashi, Hisashi Miyashita, Daisuke Sakamoto, and Hideyuki Tokuda for their support and helpful

discussions; and the anonymous reviewer for extensive and useful comments. This work is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), Japan Science and Technology Agency.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Aceto, L., Fokkink, W., Verhoef, C.: Structural operational semantics. In Bergstra, J., Ponse, A., Smolka, S. (eds.) *Handbook of process algebra*, pp. 197–292. Elsevier (2001)
- Adámek, J., Koubek, V.: On the greatest fixed point of a set functor. *Theor. Comp. Sci.* **150**, 57–75 (1995)
- Akazaki, T., Hasuo, I.: Time robustness in MTL and expressivity in hybrid system falsification. In Kroening and Pasareanu [60], pp. 356–374
- de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) *Automata, Languages and Programming*, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings, vol. 2719 of *Lect. Notes Comp. Sci.*, pp. 1022–1037. Springer (2003)
- Almagor, S., Boker U., Kupferman, O.: Discounting in LTL. In: Ábrahám, E., Havelund, K. (eds) *Tools and Algorithms for the Construction and Analysis of Systems—20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5–13, 2014. Proceedings*, vol. 8413 of *Lecture Notes in Computer Science*, pp. 424–439. Springer (2014)
- Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. *Theor. Comp. Sci.* **138**(1), 3–34 (1995)
- Alur, R., Dill, D.L.: A theory of timed automata. *Theor. Comput. Sci.* **126**(2), 183–235 (1994)
- Awodey, S.: *Category theory*. Oxford Logic Guides. Oxford Univ Press, Oxford (2006)
- Baier, C., Katoen, J.P.: *Principles of model checking*. The MIT Press (2008)
- Ball, T., Cook, B., Levin, V., Rajamani, S.K.: SLAM and static driver verifier: technology transfer of formal methods inside microsoft. In: Boiten, E.A., Derrick, J., Smith, G. (eds) *Integrated Formal Methods*, 4th International Conference, IFM 2004, Canterbury, UK, April 4–7, 2004, Proceedings, vol. 2999 of *Lecture Notes in Computer Science*, pp. 1–20. Springer (2004)
- Behrmann, G., David, A., Larsen, K.G.: A tutorial on uppaal. In: Bernardo, M., Corradini, F. (eds.) *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13–18, 2004, Revised Lectures*, vol. 3185 of *Lecture Notes in Computer Science*, pp. 200–236. Springer (2004)
- Blondel, V.D., Canterini, V.: Undecidable problems for probabilistic automata of fixed dimension. *Theory Comput. Syst.* **36**(3), 231–245 (2003)
- Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In Giacobazzi and Cousot [32], pp. 457–468
- Büchi, J.: On a decision method in restricted second order arithmetic. In: *Proc. International Congress on Logic, Method, and Philosophy of Science*, 1960, pp. 1–12. Stanford University Press (1962)
- Buss, S.R.: An introduction to proof theory. In: Buss, S.R. (ed) *Handbook of proof theory*, pp. 1–78. Elsevier (1998)
- Calcagno, C., Distefano, D., Dubreil, J., Gabi, D., Hooimeijer, P., Luca, M., O’Hearn, P.W., Papakonstantinou, I., Purbrick, J., Rodriguez, D.: Moving fast with software verification. In: Havelund, K., Holzmann, G.J., Joshi, R. (eds) *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27–29, 2015, Proceedings*, vol. 9058 of *Lecture Notes in Computer Science*, pp. 3–11. Springer (2015)

17. Chakarov, A., Voronin, Y., Sankaranarayanan, S.: Deductive proofs of almost sure persistence and recurrence properties. In Chechik and Raskin [18], pp. 260–279
18. Chechik, M., Raskin, J. (eds) Tools and Algorithms for the Construction and Analysis of Systems—22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2–8, 2016, Proceedings, vol. 9636 of Lecture Notes in Computer Science. Springer (2016)
19. Cirstea, C., Shimizu, S., Hasuo, I.: Parity automata for quantitative linear time logics. In: Proc. 7th Conference on Algebra and Coalgebra in Computer Science (CALCO 2017). To appear (2017)
20. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Graham, R.M., Harrison, M.A., Sethi, R. (eds.) Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977, pp. 238–252. ACM (1977)
21. Dai, L., Xia, B., Zhan, N.: Generating non-linear interpolants by semidefinite programming. In: Sharygina, N., Veith, H. (eds) Computer Aided Verification, 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13–19, 2013. Proceedings of Lecture Notes in Computer Science, vol. 8044, pp. 364–380. Springer (2013)
22. Datta, A., Franklin, J., Garg, D., Jia, L., Kaynar, D.K.: On adversary models and compositional security. *IEEE Secur Priv* **9**(3), 26–32 (2011)
23. Diwakaran, R.D., Sankaranarayanan, S., Trivedi, A.: Analyzing neighborhoods of falsifying traces in cyber-physical systems. In S. Martínez, E. Tovar, C. Gill and B. Sinopoli, editors, Proceedings of the 8th International Conference on Cyber-Physical Systems, ICCPS 2017, Pittsburgh, Pennsylvania, USA, April 18–20, 2017, pp. 109–119. ACM (2017)
24. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds) Formal Modeling and Analysis of Timed Systems—8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8–10, 2010. Proceedings of Lecture Notes in Computer Science, vol. 6246, pp. 92–106. Springer (2010)
25. Droste, M., Kuich, W., Vogler, H.: Handbook of Weighted Automata, 1st edn. Springer Publishing Company, Incorporated (2009)
26. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.* **410**(42), 4262–4291 (2009)
27. Floyd, R.W.: Assigning meanings to programs. In: Schwartz, J. (ed) Mathematical Aspects of Computer Science of Proceedings of Symposium on Applied Mathematics, vol. 19, pp. 19–32 (1967)
28. Forsberg, K., Mooz, H.: The relationship of system engineering to the project cycle. In: Proceedings of the National Council for Systems Engineering First Annual Conference, pp. 57–61 (1991)
29. Frazzoli, E.: Robust hybrid control of autonomous vehicle motion planning. PhD thesis, Massachusetts Institute of Technology (2001)
30. Frehse, G., Mitra, S. (eds.) Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017, Pittsburgh, PA, USA, April 18–20, 2017. ACM (2017)
31. Fu, Z., Su, Z.: XSat: a fast floating-point satisfiability solver. In S. Chaudhuri and A. Farzan, editors, Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17–23, 2016, Proceedings, Part II, of Lecture Notes in Computer Science, vol. 9780, pp. 187–209. Springer (2016)
32. Giacobazzi, R., Cousot, R. (eds) The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013. ACM (2013)
33. Girard, A., Pappas, G.J.: Approximation metrics for discrete and continuous systems. *IEEE Trans. Automatic Control* **52**(5), 782–798 (2007)
34. Girard, A., Pappas, G.J.: Approximate bisimulation: a bridge between computer science and control theory. *Eur. J. Control* **17**(5–6), 568–578 (2011)
35. van Glabbeek, R.J.: The linear time–branching time spectrum I; the semantics of concrete, sequential processes. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, chap. 1, pp. 3–99. Elsevier (2001)
36. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In J.D. Herbsleb and M.B. Dwyer, editors, Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31–June 7, 2014, pp. 167–181. ACM (2014)
37. Grädel, E., Thomas, W., Wilke, T.: *Automata, Logics, and Infinite Games: A Guide to Current Research*, vol. 2500 of Lecture Notes in Computer Science. Springer (2002)
38. Haghverdi, E., Tabuada, P., Pappas, G.J.: Bisimulation relations for dynamical, control, and hybrid systems. *Theor. Comput. Sci.* **342**(2–3), 229–261 (2005)

39. Hasuo, I.: Generic forward and backward simulations. In: Baier, C., Hermanns, H. (eds.) *International Conference on Concurrency Theory (CONCUR 2006)*, vol. 4137 of *Lect. Notes Comp. Sci.*, pp. 406–420. Springer, Berlin (2006)
40. Hasuo, I.: Generic weakest precondition semantics from monads enriched with order. *Theor. Comput. Sci.* **604**, 2–29 (2015)
41. Hasuo, I., Jacobs, B., A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Comp. Sci.*, 3(4:11), 2007
42. Hasuo, I., Jacobs, B., Sokolova, A.: The microcosm principle and concurrency in coalgebra. In: *Foundations of Software Science and Computation Structures of Lect. Notes Comp. Sci.*, vol. 4962, pp. 246–260. Springer (2008)
43. Hasuo, I., Shimizu, S., Cirstea, C.: Lattice-theoretic progress measures and coalgebraic model checking. In R. Bodik and R. Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pp. 718–732. ACM (2016)
44. Hasuo, I., Suenaga, K.: Exercises in Nonstandard Static Analysis of hybrid systems. In: Madhusudan P., Seshia, S.A. (eds.) *CAV*, vol. 7358 of *Lect. Notes Comp. Sci.*, pp. 462–478. Springer (2012)
45. F. Herbretreau, B. Srivathsan and I. Walukiewicz. Efficient emptiness check for timed büchi automata. In T. Touili, B. Cook and P.B. Jackson, editors, *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, vol. 6174 of *Lecture Notes in Computer Science*, pp. 148–161. Springer, 2010
46. W. Hino, H. Kobayashi, I. Hasuo and B. Jacobs. Healthiness from duality. In M. Grohe, E. Koskinen and N. Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pp. 682–691. ACM, 2016
47. Hoare, C.A.R.: An axiomatic basis for computer programming. *Commun. ACM*, 12:576–580, 583 (1969)
48. Holzmann, G.J.: *The SPIN Model Checker, primer and reference manual*. Addison-Wesley (2004)
49. Hopcroft, J.E., Motwani, R., Ullman, J.D.: *Introduction to automata theory, languages, and computation*. Addison-Wesley, Boston, 3rd(edn.) (2006)
50. Jacobs, B.: *Introduction to Coalgebra: Towards Mathematics of States and Observation*, vol. 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press (2016)
51. Jacobs, B., Rutten, J.J.M.M.: An introduction to (co)algebra and (co)induction. In *Advanced Topics in Bisimulation and Coinduction*, no. 52 in *Cambridge Tracts in Theoretical Computer Science*, pp. 38–99. Cambridge Univ. Press (2011)
52. Joyal, A., Nielsen, M., Winskel, G.: Bisimulation from open maps. *Inf. & Comp.* **127**(2), 164–185 (1996)
53. Jurdzinski, M.: Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *STACS*, vol. 1770 of *Lecture Notes in Computer Science*, pp. 290–301. Springer (2000)
54. Kahn, G.: The semantics of simple language for parallel programming. In: *IFIP Congress*, pp. 471–475 (1974)
55. Kane, A.: *Runtime monitoring for safety-critical embedded systems*. PhD thesis, Carnegie Mellon University (2015)
56. Kido, K., Chaudhuri, S., Hasuo, I.: Abstract interpretation with infinitesimals - towards scalability in nonstandard static analysis. In B. Jobstmann and K.R.M. Leino, editors, *Verification, Model Checking, and Abstract Interpretation, 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17–19, 2016. Proceedings*, vol. 9583 of *Lecture Notes in Computer Science*, pp. 229–249. Springer (2016)
57. Kim, E.S., Arcak, M., Seshia, S.A.: A small gain theorem for parametric assume-guarantee contracts. In *Frehse and Mitra [30]*, pp. 207–216
58. Klin, B.: Bialgebraic methods and modal logic in structural operational semantics. *Inf. & Comp.* **207**(2), 237–257 (2009)
59. Kobayashi, T., Ishikawa, F., Honiden, S.: Refactoring refinement structure of Event-B machines. In: J.S. Fitzgerald, C.L. Heitmeyer, S. Gnesi and A. Philippou, editors, *FM 2016: Formal Methods - 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings*, vol. 9995 of *Lecture Notes in Computer Science*, pp. 444–459 (2016)
60. Kroening, D., Pasareanu, C.S.: *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II of Lecture Notes in Computer Science*. vol. 9207, Springer (2015)

61. M.Z. Kwiatkowska, G. Norman and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, vol. 6806 of *Lect. Notes Comp. Sci.*, pp. 585–591. Springer, 2011
62. A. Legay, S. Sedwards and L. Traonouez. Rare events for statistical model checking an overview. In K.G. Larsen, I. Potapov and J. Srba, editors, *Reachability Problems - 10th International Workshop, RP 2016, Aalborg, Denmark, September 19-21, 2016, Proceedings*, vol. 9899 of *Lecture Notes in Computer Science*, pp. 23–35. Springer, 2016
63. Leinster, T.: *Basic Category Theory*. Cambridge Univ. Press (2014)
64. Lynch, N., Vaandrager, F.: Forward and backward simulations. I. Untimed systems. *Inf. & Comp.* **121**(2), 214–233 (1995)
65. L. Ma, C. Artho, C. Zhang, H. Sato, J. Gmeiner and R. Ramler. GRT: program-analysis-guided random testing (T). In M.B. Cohen, L. Grunske and M. Whalen, editors, *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, pp. 212–223. IEEE Computer Society, 2015
66. S. Mac Lane. *Categories for the Working Mathematician*. Springer, Berlin, 2nd edn., 1998
67. R. Majumdar. Robots at the edge of the cloud. In Chechik and Raskin [18], pp. 3–13
68. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989
69. Miyashita, H., Tai, H., Amano, S.: Controlled modeling environment using flexibly-formatted spreadsheets. In: Jalote, P., Briand, L.C., van der Hoek, A. (eds.) *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31–June 07, 2014*, pp. 978–988. ACM (2014)
70. Nakagawa S., Hasuo, I.: Near-optimal scheduling for LTL with future discounting. In P. Ganty and M. Loret, editors, *Trustworthy Global Computing - 10th International Symposium, TGC 2015, Madrid, Spain, August 31 - September 1, 2015 Revised Selected Papers of Lecture Notes in Computer Science*, vol. 9533, pp. 112–130. Springer (2015)
71. Park, D.M.R.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) *Proceedings 5th GI Conference on Theoretical Computer Science*, vol. 104 of *Lect. Notes Comp. Sci.*, pp. 15–32. Springer, Berlin (1981)
72. Platzer, A.: *Logical analysis of hybrid systems—proving theorems for complex dynamics*. Springer (2010)
73. Robinson, A.: *Non-standard analysis*. Princeton Univ Press, Princeton (1966)
74. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. *Theor. Comp. Sci.* **249**, 3–80 (2000)
75. Schneider, K., Brandt, J. (eds.): *Verifying Nonlinear Real Formulas Via Sums of Squares*. Springer, Berlin (2007)
76. Shoukry, Y., Nuzzo, P., Sangiovanni-Vincentelli, A.L., Seshia, S.A., Pappas, G.J., Tabuada, P.: SMC: satisfiability modulo convex optimization. In Frehse and Mitra [30], pp. 19–28
77. Souyris, J., Delmas, D.: Experimental assessment of astrée on safety-critical avionics software. In F. Saglietti and N. Oster, editors, *Computer Safety, Reliability, and Security, 26th International Conference, SAFECOMP 2007, Nuremberg, Germany, September 18-21, 2007.*, of *Lecture Notes in Computer Science*, vol. 4680, pp. 479–490. Springer (2007)
78. Suenaga, K., Hasuo, I.: Programming with infinitesimals: a while-language for hybrid system modeling. In L. Aceto, M. Henzinger and J. Sgall, editors, *ICALP (2)*, of *Lect. Notes Comp. Sci.*, vol. 6756, pp. 392–403. Springer (2011)
79. Suenaga, K., Sekine, H., Hasuo, I.: Hyperstream processing systems: nonstandard modeling of continuous-time signals. In Giacobazzi and Cousot [32], pp. 417–430
80. Tedrake, R.: Convex and combinatorial optimization for dynamic robots in the real world. In Frehse and Mitra [30], p. 141
81. Turi, D., Plotkin, G.: Towards a mathematical operational semantics. In: *Logic in Computer Science*, pp. 280–291. IEEE, Computer Science Press (1997)
82. Ulus, D., Ferrère, T., Asarin, E., Maler, O.: Online timed pattern matching using derivatives. In Chechik and Raskin [18], pp. 736–751
83. Urabe, N., Hara, M., Hasuo, I.: Categorical liveness checking by corecursive algebras. In: *Proc. LICS 2017*. To appear (2017)
84. Urabe, N., Hasuo, I.: Generic forward and backward simulations III: quantitative simulations by matrices. In P. Baldan and D. Gorla, editors, *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2–5, 2014. Proceedings of Lecture Notes in Computer Science*, vol. 8704, pp. 451–466. Springer. Best paper award (2014)

85. Urabe, N., Hasuo, I.: Coalgebraic infinite traces and kleisli simulations. In: Moss, L.S., Sobocinski, P. (eds) 6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24–26, 2015, Nijmegen, The Netherlands, of LIPIcs, vol. 35, pp. 320–335. Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik (2015)
86. Urabe, N., Hasuo, I.: Quantitative simulations by matrices. *Inf. Comput.* **252**, 110–137 (2017)
87. Urabe, N., Shimizu, S., Hasuo, I.: Coalgebraic trace semantics for bueschi and parity automata. In: Desharnais, J., Jagadeesan, R. (eds) 27th International Conference on Concurrency Theory, CONCUR 2016, August 23–26, 2016, Québec City, Canada, of LIPIcs, vol. 59, pp. 24:1–24:15. Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik (2016)
88. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G.M. (eds) Banff Higher Order Workshop of Lecture Notes in Computer Science, vol. 1043, pp. 238–266. Springer (1995)
89. Vijayaraghavan, M., Chlipala, A., Dave, N.: Modular deductive verification of multiprocessor hardware designs. In Kroening and Pasareanu [60], pp. 109–127
90. Vitus, M.P., Zhang, W., Tomlin, C.J.: A hierarchical method for stochastic motion planning in uncertain environments. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2263–2268 (2012)
91. Waga, M., Akazaki, T., Hasuo, I.: A boyer-moore type algorithm for timed pattern matching. In M. Fränzle and N. Markey, editors, Formal Modeling and Analysis of Timed Systems, 14th International Conference, FORMATS 2016, Quebec, QC, Canada, August 24–26, 2016, Proceedings of Lecture Notes in Computer Science, vol. 9884, pp. 121–139. Springer (2016)
92. Wilke, T.: Alternating tree automata, parity games, and modal μ -calculus. *Bull. Belg. Math. Soc. Simon Stevin* **8**(2), 359–391 (2001)
93. G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993
94. Yamaguchi, T., Kaga, T., Donzé, A., Seshia, S.A.: Combining requirement mining, software model checking and simulation-based verification for industrial automotive systems. In: Piskac, R., Talupur, M. (eds.) 2016 Formal Methods in Computer-Aided Design, FMCAD 2016, Mountain View, CA, USA, October 3–6, 2016, pp. 201–204. IEEE (2016)

Ichiro Hasuo, Ph.D., is an Associate Professor at the National Institute of Informatics (NII), Tokyo, Japan. He is at the same time the Research Director of the JST ERATO “Metamathematics for Systems Design” Project. He received Ph.D. (cum laude) in Computer Science from Radboud University Nijmegen, The Netherlands, in 2008. Before joining NII, he held an Assistant Professor position at Research Institute for Mathematical Sciences (RIMS), Kyoto University, and Lecturer and Associate Professor positions at the Department of Computer Science, the University of Tokyo. His research interests include the following: mathematical (logical, algebraic, and categorical) structures in software science (esp. formal methods); abstraction and generalization of deductive and automata-theoretic verification techniques; and their application to cyberphysical systems.