15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Logical Theory & Completeness

André Platzer

Carnegie Mellon University
Lecture 24

1. Introduction

This course has studied a number of logics, first-order logic FOL in Lecture 2, differential dynamic logic $\mathsf{d}\mathcal{L}$ [Pla08, Pla10b, Pla12c, Pla12b] in Lecture 3 and Lecture 4 and following, differential temporal dynamic logic dTL [Pla07, Pla10b, Chapter 4] in Lecture 16 and 17, as well as differential game logic $\mathsf{dG}\mathcal{L}$ [Pla13] since Lecture 22. There are other logics for cyber-physical systems that have not been included in this course, but share similar principles for further dynamical aspects. Such logics include quantified differential dynamic logic $\mathsf{Qd}\mathcal{L}$ for distributed hybrid systems [Pla10c, Pla12a], which are systems that are simultaneously distributed systems and hybrid systems, as well as stochastic differential dynamic logic $\mathsf{Sd}\mathcal{L}$ for stochastic hybrid systems [Pla11], which simultaneously involve stochastic dynamics and hybrid dynamics. Logics play a stellar role not just in cyber-physical systems, but also many other contexts. Other important logics include propositional logic, restrictions of first-order logic to certain theories, such as first-order logic of real arithmetic [Tar51], and higher-order logic [And02]. But there are numerous other important and successful logics for many purposes, both general and specific.

In this lecture, we take a step back and study some common important concepts in understandings logics as the objects of study themselves. This study will necessarily be hopelessly incomplete for lack of time. But it should give you a flavor of important principles and concepts in logic that we have not already run across explicitly in earlier lectures of this course, even if many have been foreshadowed. We will also have the opportunity to apply these more general concepts to cyber-physical systems.

These lecture notes are based on [Sch12, Pla10b, Pla08, Pla12c, Pla12b, Pla10d, Pla14]. The most important learning goals of this lecture are:

Modeling and Control: This lecture culminates in a deep and surprising intimate relationship between discrete and continuous dynamics. This relationship provides a deeper understanding than ever before of the core principles behind cyber-physical systems, because it makes it possible to completely align and proof-theoretically equate discrete dynamics, continuous dynamics, and joint hybrid dynamics. This understanding has an effect on the significance of identifying the relevant dynamical aspects of cyber-physical systems by emphasizing the source of their simplicity: the individual parts of CPS models are easier than understanding the whole at once.

Computational Thinking: This lecture showcases exemplary computational thinking in action by exploiting proof-theoretical relationships to draw conclusions about cyber-physical systems and their analysis. It also practices the logical trinity consisting of the relationship of syntax, semantics, and axiomatics in more detail for the case of first-order logic. Finally, the lecture identifies logical parallels and differences of general first-order logic, interpreted first-order logic of real arithmetic, and differential dynamic logic.

CPS Skills: This lecture has only a minor impact on CPS skills in the form of shining a light of surprising intimacy on the relationship between discrete and continuous dynamics of CPS, a phenomenon that reemphasizes the possible liberties with the characterization of their dynamical aspects. By the complete alignment, one person's discrete dynamics may be another person's continuous dynamics and vice versa. This confirms the significant role that the design of proper models that are adequate and suitable for analysis plays in CPS verification and validation.

2. Soundness

The most important parts of a logic $\mathcal{L}$ are the following. The logic $\mathcal{L}$ defines what the *syntactically well-formed formulas* are. Every well-formed formula carries meaning, which the *semantics of formulas* in $\mathcal{L}$ defines. The semantics defines a relation $\vDash$ between sets of formulas and formulas, in which $\Phi \vDash \phi$ holds iff ϕ is a semantic consequence of the set of formulas Φ, i.e. ϕ is true (usually written $\nu \models \phi$) in every interpretation ν for which all formulas $\psi \in \Phi$ are true. The most important case for our purposes is the case $\Phi = \emptyset$ of validity, in which case $\vDash \phi$ holds iff ϕ is valid, i.e. true ($\nu \models \phi$) in all interpretations ν of $\mathcal{L}$. An interpretation ν in which ϕ is true (i.e. $\nu \models \phi$) is also called a *model* of ϕ.

For the case of first-order logic FOL, Lecture 2 defined both their syntax and semantics. The syntax and semantics of differential dynamic logic $\mathsf{d}\mathcal{L}$ has been defined in Lecture 3 and Lecture 4.

The syntax of a logic $\mathcal{L}$ defines what we can write down that carries meaning. The semantics of a logic $\mathcal{L}$ then defines what the meaning of the syntactic formulas is. The semantics, in particular, defines which formulas express true facts about the world, either in a particular interpretation ν or about the world in general (for valid formulas,

which are true regardless of the interpretation). Yet, the semantics is usually highly ineffective, so that it cannot be used directly to find out whether a formula is valid. Just think of formulas in differential dynamic logic that express safety properties of hybrid systems. It would not get us very far if we were to try to establish the truth of such a formula by literally computing the semantics (which includes executing the hybrid system) in every initial state, of which there are uncountably infinitely many.

Instead, logics come with *proof calculi* that can be used to establish validity of logical formulas in the logic $\mathcal{L}$. Those proof calculi comprised *axioms* (Lecture 5) and *proof rules* (Lecture 6 and others), which can be combined to prove or derive logical formulas of the logic $\mathcal{L}$. The proof calculus of the logic $\mathcal{L}$ defines a relation $\vdash$ between sets of formulas and formulas, in which $\Phi \vdash \phi$ holds iff ϕ is provable from the set of formulas Φ. That is, there is a proof of ϕ in the proof calculus of $\mathcal{L}$ that uses only assumptions from Φ. The most important case for our purposes is again $\Phi = \emptyset$, in which case $\vdash \phi$ holds iff ϕ is provable in the proof calculus of $\mathcal{L}$, i.e. there is a proof of ϕ.

Of course, only some formulas of $\mathcal{L}$ are provable, not all of them. The formula $p \wedge \neg p$ should not be provable in any proper logic, because it is inconsistently *false* and, thus, cannot possibly be valid.

We could have written down any arbitrary axiom, or we could have accidentally had a typo in the axioms. So a crucial question we have to ask (and have asked every time we introduced an axiom in other lectures of this course) is whether the axioms and proof rules are sound. In a nutshell, a proof calculus is sound if all provable formulas are valid.

Theorem 1 (Soundness [Pla08, Pla10b, Pla12b])**.** *The proof calculus of differential dynamic logic is* sound*, i.e.* $\vdash \subseteq \vDash$*, which means that* $\vdash \phi$ *implies* $\vDash \phi$ *for all* $\mathsf{d}\mathcal{L}$ *formulas* ϕ*. That is, all provable* $\mathsf{d}\mathcal{L}$ *formulas are valid.*

The significance of soundness is that, whatever formula we derive by using the $\mathsf{d}\mathcal{L}$ proof rules and axioms, we can rest assured that it is valid, i.e. true in all states. In particular, it does not matter how big and complicated the formula might be, we know that it is valid as long as we have a proof for it. About the axioms, we can easily convince ourselves using a soundness proof why they are valid, and then conclude that all provable formulas are also valid, because they follow from sound axioms by sound proof rules.

Note 2 (Necessity of soundness)**.** *Soundness is a must for otherwise we could not trust our own proofs.*

3. Soundness Challenge for CPS

What good would it do to analyze safety of a CPS using a technique that is as faulty as the original CPS? If an unsound analysis technique says that a CPS is correct, we are,

fundamentally, not much better off than without any analysis, because all we can conclude is that we did not find problems, not that there are none.[1] After all, an unsound analysis technique could say "correct", which might turn out to be a lie because the correctness statement itself was not valid.

Note 3 (Challenge of soundness)**.** *In a domain that is as challenging as cyber-physical systems and hybrid systems, it is surprisingly easy for analysis techniques to become unsound due to subtle flaws. Necessary conditions for soundness and the numerical decidability frontier have been identified in the literature [PC07, Col07]. The crux of the matter is that hybrid systems are subject to a numerical analogue of the halting problem of Turing machines [PC07].*

There is a shockingly large number of approaches that, for subtle reasons, are subject to the unsoundness resulting from non-observance of the conditions identified in [PC07, Col07]. Consequently, such approaches need some of the additional assumptions identified in [PC07, Col07] to have a chance to become sound.

4. First-Order Logic

Even though this course primarily studied extensions of first-order logic by dynamic modalities for hybrid systems instead of pure first-order logic, the sequent proof rules of propositional logic and quantifiers (instantiation and Skolemization) give a suitable proof calculus for first-order logic. And this suitability of the proof calculus for first-order logic is a much stronger statement than soundness.

Soundness is the question whether all provable formulas are valid and is a minimal requirement for proper logics. Completeness studies the converse question whether all valid formulas are provable.

The first-order logic proof calculus can be shown to be both sound and complete, which is a result that originates from Gödel's PhD thesis [Göd30], albeit in a different form.

[1] Notwithstanding of the fact that unsound analysis techniques can still be very useful in practice, especially if they identify problems in system designs. Yet, we should exercise great care in concluding anything from unsound techniques that have not found a problem. As has been aptly phrased by Dijkstra [Dij70]: "Program testing can be used to show the presence of bugs, but never to show their absence!"

Theorem 2 (Soundness & completeness of first-order logic)**.** *First-order logic is* sound*, i.e.* $\vdash \subseteq \models$*, which means that* $\vdash \phi$ *implies* $\models \phi$ *for all first-order formulas* ϕ *(all provable formulas are valid). First-order logic is* complete*, i.e.* $\models \subseteq \vdash$*, which means that* $\models \phi$ *implies* $\vdash \phi$ *for all first-order formulas* ϕ *(all valid formulas are provable). In particular, the provability relation* $\vdash$ *and the validity relation* $\models$ *coincide for first-order logic:* $\vdash = \models$*. The same holds in the presence of a set of assumptions* Γ*, i.e.* $\Gamma \vdash \phi$ *iff* $\Gamma \models \phi$*, that is, a first-order formula* ϕ *is provable from a set of first-order assumptions* Γ *in first-order logic if and only if* ϕ *is a consequence of* Γ*, i.e. entailed by* Γ*, i.e. true in all models of* Γ*.*

This lecture will not set out for a direct proof of this result, because the techniques used for those proofs are interesting but would lead us too far astray. An indirect justification for what makes first-order logic so special that Theorem 2 can hold will be discussed later.

The following central result about compactness of first-order logic is of similar importance. Compactness is involved in most proofs of Theorem 2, but, once Theorem 2 has been proved, also easily follows from Theorem 2. Compactness means that if a formula A is a consequence of a set of formulas Γ, then it already is a consequence of finitely many formulas.

Theorem 3 (Compactness of first-order logic)**.** *First-order logic is* compact*, i.e.*

$$\Gamma \models A \iff E \models A \text{ for some finite } E \subseteq \Gamma \tag{1}$$

Proof. By Theorem 2, $\vdash = \models$. By completeness, the semantic compactness theorem (1) is equivalent to the syntactic compactness theorem:

$$\Gamma \vdash A \iff E \vdash A \text{ for some finite } E \subseteq \Gamma \tag{2}$$

Condition (2) is obvious, because provability implies that there is a proof, which can, by definition, only use finitely many assumptions $E \subseteq \Gamma$. □

Compactness is equivalent to the finiteness property, which, for that reason, is usually simply referred to as compactness. The finiteness property says that a set of formulas Γ has a model if and only if all its finite subsets of formulas have a model.

Corollary 4 (Finiteness)**.** *First-order logic satisfies the* finiteness property*, i.e.*

$$\Gamma \text{ has a model} \iff \text{all finite } E \subseteq \Gamma \text{ have a model} \tag{3}$$

Proof. Compactness (Theorem 3) implies the finiteness property. The key observation is that Γ has no model iff $\Gamma \models \mathit{false}$, because if Γ has no model, then false holds in all models of Γ of which there are none. Conversely, the only chance for false to hold in all models of Γ is if there are no such models, since false never holds. By Theorem 3,

$$\Gamma \models \mathit{false} \iff \exists \text{finite } E \subseteq \Gamma \quad E \models \mathit{false}$$

Hence,

Γ has a model $\iff \Gamma \not\models \mathit{false} \iff \forall$ finite $E \subseteq \Gamma \;\; E \not\models \mathit{false} \iff$ all finite $E \subseteq \Gamma$ have a model

It is worth noting that, conversely, the finiteness property implies compactness.

$$\begin{aligned}
\Gamma \models A &\iff \Gamma \cup \{\neg A\} \text{ has no model} \\
&\iff \text{some finite } E \subseteq \Gamma \cup \{\neg A\} \text{ has no model} && \text{by finiteness} \\
&\iff E \models A \text{ for some finite } E \subseteq \Gamma
\end{aligned}$$

The last equivalence uses that we might as well include $\neg A$ in E, because if E has no model then neither does $E \cup \{\neg A\}$. □

5. Löwenheim-Skolem-Herbrand Theory in a Nutshell

A beautiful and, in the long term, quite impactful theory due to Leopold Löwenheim [Löw15], Thoralf Skolem [Sko20], and Jacques Herbrand [Her30], gave rise to the first automated theorem prover. Granted, it was quite a theoretical procedure at first, but it was the first one and ultimately had practical offspring in the form of instance-based methods. Herbrand's procedure reduces first-order logic validity to a sequence of validity questions in propositional logic, each of which are perfectly decidable by truth-tables (or more practically by SAT solvers). We focus on the parts of the Löwenheim-Skolem-Herbrand theory that are most important for the subsequent development, which, incidentally, are its syntactic aspects. The Löwenheim-Skolem-Herbrand theory is developed in more depth in Appendix A, including the nontrivial semantic justifications for the syntactic transformations.

The first ingredient is Herbrand's theorem on what are now called Herbrand-disjunctions, i.e. validity-preserving instantiations of existential quantifiers as a disjunction of finitely many terms. Observe the relationship to quantifier elimination in real arithmetic, which will be discussed in more detail later.

Theorem 5 (Herbrand's theorem: Herbrand disjunctions [Her30])**.** *For a quantifier-free formula $\phi(x)$ of a free variable x without equality*

$\exists x\, \phi(x)$ *valid* $\iff \phi(t_1) \vee \cdots \vee \phi(t_n)$ *valid for some* $n \in \mathbb{N}$ *and ground terms* $t_1, \ldots, t_n$

Observe that the formula $\phi(t_1) \vee \cdots \vee \phi(t_n)$ is quantifier-free if $\phi(x)$ was quantifier-free. This reduces the validity of existential formulas of first-order logic to the validity of formulas in propositional logic. Not every first-order formula is of the form required in Theorem 5 but can be converted into that form without altering validity by Herbrandization, a dual of Skolemization:

Lemma 6 (Herbrandization)**.** *With each first-order logic formula ψ, a formula*

$$\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$$

with quantifier-free $\phi(x_1, \ldots, x_n)$ can be associated effectively that is valid if and only if ψ is. The formula $\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$ uses additional function symbols that do not occur in ψ.

Example 7. The formula $\forall x \, \exists y \, p(x, y)$ is valid iff the formula $\exists y \, p(a, y)$ is valid, where a is a new function symbol of arity zero. Obviously, if $\exists y \, p(a, y)$ is valid, it needs to be true no matter what the interpretation of a is, which implies that $\forall x \, \exists y \, p(x, y)$ is valid.

Likewise, $\forall x \, \exists y \, \forall z \, p(x, y, z)$ is valid iff $\exists y \, p(a, y, b(y))$ is, where a is a new function symbol of arity 0 and b a new function symbol of arity 1, i.e. expecting one argument. This time, the term $b(y)$ ensures that the original formula is valid for any value of z, no matter what the particular value of y was, because b is a function hat could have an arbitrary interpretation.

Even if semidecidability was proved differently by Gödel first [Göd29], Herbrand's subsequent result Theorem 5 enables a straightforward and constructive proof of the semidecidability of the validity problem of first-order logic:

Theorem 8 (Semidecidability of first-order logic [Göd29])**.** *Validity in first-order logic is semidecidable, i.e. there is an algorithm that, given any formula ϕ of first-order logic, correctly reports whether ϕ is valid or not and that also terminates for all ϕ that indeed valid. The algorithm will not generally terminate when ϕ is not valid.*

Proof. The semidecision procedure for validity of first-order logic formulas ψ proceeds as follows:

1. Herbrandize ψ to obtain a formula $\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$ by Lemma 6, which preserves validity.
2. Enumerate all $m \in \mathbb{N}$ and all ground terms t_i^j $(1 \leq j \leq n, 1 \leq i \leq m)$, over the new signature.
   a) If the *propositional* formula

$$\phi(t_1^1, \ldots, t_1^n) \vee \cdots \vee \phi(t_m^1, \ldots, t_m^n)$$

   is valid, then so is $\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$ and, hence, ψ is valid.

By Theorem 5 and Lemma 6, the procedure terminates for all valid first-order formulas (yet fails to terminate for other formulas). □

The procedure in this proof will always succeed but is rather silly, because it enumerates all terms for instantiation rather blindly. Nevertheless, refinements of this

idea lead to very successful automated theorem proving techniques for first-order logic known as *instance-based methods* [BT10], which restrict the instantiation to instantiation-on-demand in various ways to make the procedure more goal-directed. There are also many successful automatic theorem proving procedures for first-order logic that are based on different principles, including tableaux and resolution [Fit96].

6. Back to CPS

First-order logic is beautiful, elegant, expressive, and simple. Unfortunately, however, it is not expressive enough for hybrid systems [Pla10b, Pla12b, Pla13]. As soon as we come back to studying hybrid systems, the situation gets more difficult. And that is not by accident, but, instead, a fundamental property of first-order logic and of hybrid systems. Per Lindström characterized first-order logic in indirect ways to show which properties stronger logics could still possess and which ones they cannot [Lin69]. Hybrid systems themselves are also known not to be semidecidable, which shows that the semidecidable first-order logic cannot understand the full picture of hybrid systems.

Given that differential dynamic logic talks about properties of hybrid systems, and Turing machines are a special case of hybrid systems that just neglects the mention of differential equations, undecidability is not surprising. We show a very simple standalone proof of incompleteness by adapting a proof for programs, e.g., [Pla10d].

Theorem 9 (Incompactness)**.** *Differential dynamic logic is not compact.*

Proof. It is easy to see that there is a set of formulas that has no model even though all finite subsets have a model, consider:

$$\{\langle x' = 1\rangle x > y\} \cup \{\neg(x + n > y) \ : \ n \in \mathbb{N}\}$$

The same happens with

$$\{\langle (x := x + 1)^* \rangle x > y\} \cup \{\neg(x + n > y) \ : \ n \in \mathbb{N}\}$$ □

Hence, differential dynamic logic does not have the finiteness property, which is equivalent to compactness (Corollary 4).

Since soundness and completeness imply compactness (see proof of Theorem 3), incompactness implies incompleteness[2], because $\mathsf{d}\mathcal{L}$ is sound. An explicit proof is as follows:

Theorem 10 (Incompleteness [Pla08])**.** *Differential dynamic logic has no effective sound and complete calculus.*

[2]Strictly speaking, incompleteness only follows for effective calculi. *Relative* soundness and completeness can still be proved for $\mathsf{d}\mathcal{L}$ [Pla08, Pla10b, Pla12b], which gives very insightful characterizations of the challenges and complexities of hybrid systems.

Proof. Suppose there was an effective sound and complete calculus for dL. Consider a set Γ of formulas that has no model in which all finite subsets have a model, which exists by Theorem 9. Then $\Gamma \vDash (0 > 1)$ is valid, thus provable by completeness. But since the proof is effective, it can only use finitely many assumptions $E \subset \Gamma$. Thus $E \vDash (0 > 1)$ by soundness. But then the finite set E has no model, which is a contradiction. □

Having said these negative (but necessary) results about differential dynamic logic (and, by classical arguments, any other approach for hybrid systems), let's return to the surprisingly amazing positive properties that differential dynamic logic possesses.

For one thing, the basis of differential dynamic logic is the first-order logic of real arithmetic, not arbitrary first-order logic. This enables a particularly pleasant form of Herbrand disjunctions (Theorem 5) resulting from quantifier elimination in real arithmetic (recall Lecture 18 and Lecture 19).

Definition 11 (Quantifier elimination)**.** A first-order theory admits *quantifier elimination* if, with each formula ϕ, a quantifier-free formula $\mathrm{QE}(\phi)$ can be associated effectively that is equivalent, i.e. $\phi \leftrightarrow \mathrm{QE}(\phi)$ is valid (in that theory).

Theorem 12 (Tarski [Tar51])**.** *The first-order logic of real arithmetic admits quantifier elimination and is, thus, decidable.*

Also recall from Lecture 18 and Lecture 19 that the quantifier-free formula $\mathrm{QE}(\phi)$ is constructed by substitution or virtual substitution from ϕ, with some side constraints on the parameter relations.

Note 14. *Virtual substitution in* $\mathrm{FOL}_{\mathbb{R}}$ *essentially leads to an equivalence of the form*

$$\exists x\, F \leftrightarrow \bigvee_{t \in T} A_t \wedge F_x^t \tag{4}$$

for a suitable finite *set T of extended terms that depends on the formula F and that gets substituted into F virtually, i.e. in a way that results in standard real arithmetic terms, not extended terms.*

The quantifier-elimination instantiations (4) are more useful than Theorem 5, because the required terms T for instantiation can be computed effectively and the equivalence holds whether or not the original formula was valid. This makes first-order logic of real arithmetic decidable, while general first-order logic is only semidecidable, because the proof of Theorem 8 still involves search over the appropriate Herbrand terms t_i to use, which were constructed effectively for $\mathrm{FOL}_{\mathbb{R}}$. This pleasant basis makes it possible to use the proof calculus of differential dynamic logic to synthesize constraints on the parameters to make an intended conjecture valid [Pla10b]. Aside from the fact that real

numbers are the appropriate basis for understanding real positions and velocities and the like in cyber-physical systems.

7. The Miracle of Hybrid Systems

The dℒ calculus is *sound* [Pla08, Pla12b], that is, every formula that is provable using the dℒ axioms and proof rules is valid, i.e., true in all states. That is, for all dℒ formulas ϕ:

$$\vdash \phi \text{ implies } \vDash \phi \tag{5}$$

Soundness should be *sine qua non* for formal verification, but, for fundamental reasons [PC07, Col07], is so complex for hybrid systems that it is sometimes inadvertently forsaken. In logic, we ensure soundness just by checking locally once for each axiom and proof rule. Thus, no matter how complicated a proof, the proven dℒ formula is valid, because it is a (complicated) consequence of lots simple valid proof steps.

More intriguingly, however, our logical setting also enables us to ask the converse: is the dℒ proof calculus *complete*, i.e., can it prove all that is true? That is, does the converse of (5) hold? Theorem 10 as well as a simple corollary to Gödel's incompleteness theorem show that already the fragments for discrete dynamical systems and for continuous dynamical systems are incomplete [Pla08].

In logic, the suitability of an axiomatization can still be established by showing completeness relative to a fragment [Coo78, HMP77]. This *relative completeness*, in which we assume we were able to prove valid formulas in a fragment and prove that we can then prove all others, also tells us how subproblems are related computationally. It tells us whether one subproblem dominates the others. Standard relative completeness [Coo78, HMP77], however, which works relative to the data logic, is inadequate for hybrid systems, whose complexity comes from the dynamics, not the data logic, first-order real arithmetic, which is perfectly decidable first-order real arithmetic [Tar51].

From Hybrid to Continuous. Using the proof calculus of dℒ, the problem of proving properties of hybrid systems reduces completely to proving properties of elementary continuous systems [Pla08].

Theorem 13 (Continuous relative completeness of dℒ [Pla08, Pla12b])**.** *The* dℒ *calculus is a* sound and complete axiomatization *of hybrid systems relative to differential equations, i.e., every valid* dℒ *formula can be derived from elementary properties of differential equations.*

In particular, if we want to prove properties of hybrid systems, all we need to do is to prove properties of continuous systems, because the dℒ calculus completely handles all other steps in the proofs that deal with discrete or hybrid systems. Of course, one has to be able to handle continuous systems in order to understand hybrid systems, because continuous systems are a special case of hybrid systems. But it turns out that

this is actually all that one needs in order to verify hybrid systems, because the dℒ proof calculus completely axiomatizes all the rest of hybrid systems.

This central result shows that we can prove properties of hybrid systems in the dℒ calculus exactly as good as properties of differential equations can be proved. One direction is obvious, because differential equations are part of hybrid systems, so we can only understand hybrid systems to the extent that we can reason about their differential equations. We have shown the other direction by proving that all true properties of hybrid systems can be reduced effectively to elementary properties of differential equations. Moreover, the dℒ proof calculus for hybrid systems can perform this reduction constructively and, vice versa, provides a provably perfect lifting of every approach for differential equations to hybrid systems.

Another important consequence of this result is that decomposition can be successful in taming the complexity of hybrid systems. The dℒ proof calculus is strictly compositional. All proof rules prove logical formulas or properties of HPs by reducing them to structurally simpler dℒ formulas. As soon as we understand that the hybrid systems complexity comes from a combination of several simpler aspects, we can, hence, tame the system complexity by reducing it to analyzing the dynamical effects of simpler parts. This decomposition principle is exactly how dℒ proofs can scale to interesting systems in practice. Theorem 13 gives the theoretical evidence why this principle works in general, not just in the case studies that have been considered so far. This is a good illustration of our principle of multi-dynamical systems and even a proof that the decompositions behind the multi-dynamical systems approach are successful. Note that, even though Theorem 13 proves (constructively) that every true property of hybrid systems can be proved in the dℒ calculus by decomposition from elementary properties of differential equations, it is still an interesting question which decompositions are most efficient.

From Hybrid to Discrete. In a certain sense, it may appear to be more complicated to handle continuous dynamics than discrete dynamics. If the continuous dynamics are not just subsuming discrete dynamics but if they were "inherently more", then one might wonder whether hybrid systems verification could be understood with a discrete dynamical system like a classical computer at all. Of course, such a naïve consideration would be quite insufficient, because, e.g., properties of objects in uncountable continuous spaces can very well follow from properties of finitary discrete objects. Finite dℒ proof objects, for example, already entail properties about uncountable continuous state spaces of systems.

Fortunately, all such worries about the insufficiency of discrete ways of understanding continuous phenomena can be settled once and for all by studying the proof-theoretical relationship between discrete and continuous dynamics. We have shown not only that the axiomatization of dℒ is complete relative to differential equations, but that it is also complete relative discrete systems [Pla12b].

Theorem 14 (Discrete relative completeness of dℒ [Pla12b]). *The* dℒ *calculus is a* sound and complete axiomatization *of hybrid systems relative to discrete systems, i.e., every valid* dℒ *formula can be derived from elementary properties of discrete systems.*

Thus, the dℒ calculus can also prove properties of hybrid systems exactly as good as properties of discrete systems can be proved. Again, the proof of Theorem 14 is constructive, entailing that there is a constructive way of reducing properties of hybrid systems to properties of discrete systems using the dℒ calculus. Furthermore, the dℒ calculus defines a decision procedure for dℒ sentences relative to an oracle for discrete systems [Pla12b]. Theorems 13 and 14 lead to a surprising result aligning discrete and continuous systems properties.

Theorem 15 (dℒ equi-expressibility [Pla12b]). *The logic* dℒ *is* expressible *in both its discrete and in its continuous fragment: for each* dℒ *formula ϕ there is a continuous formula $\phi^\flat$ that is equivalent, i.e., $\vDash \phi \leftrightarrow \phi^\flat$ and a discrete formula $\phi^\#$ that is equivalent, i.e., $\vDash \phi \leftrightarrow \phi^\#$. The converse holds trivially. Furthermore, the construction of $\phi^\flat$ and $\phi^\#$ is effective (and the equivalences are provable in the* dℒ *calculus).*

The proof of the surprising result Theorem 15 is constructive but rather nontrivial (some 20 pages). Consequently, all hybrid questions (and, thus, also all discrete questions) can be formulated constructively equivalently as purely continuous questions and all hybrid questions (also all continuous questions) can be formulated constructively equivalently as purely discrete questions. There is a constructive and provable reduction from either side to the other.

Note 18 (Complete logical alignment). *As a corollary to Theorems 13 and 14, we can proof-theoretically and constructively equate*

$$hybrid = continuous = discrete$$

by a complete logical alignment *in the sense that proving properties of either of those classes of dynamical systems is the same as proving properties of any other of those classes, because all properties of one system can be provably reduced in a complete, constructive, and equivalent way to any of the other system classes.*

Even though each kind of dynamics comes from fundamentally different principles, they all meet in terms of their proof problems being interreducible, even constructively; see Fig. 1. The proof problem of hybrid systems, the proof problem of continuous systems, and the proof problem of discrete systems are, thus, equivalent. Any proof technique for one of these classes of systems completely lifts to proof techniques for the other class of systems.

Since the proof problems interreduce constructively, every technique that is successful for one kind of dynamics lifts to the other kind of dynamics through the dℒ calculus

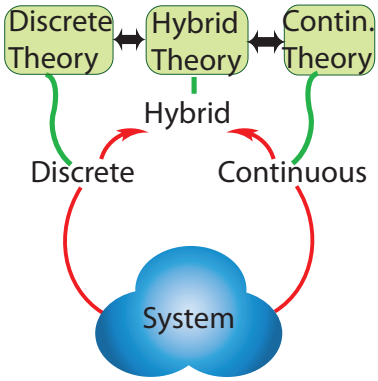

Figure 1: The proof theory of hybrid systems provides a complete proof-theoretical bridge aligning the theory of discrete systems and the theory of continuous systems

in a provably perfect way. Induction, for example, is the primary technique for proving properties of discrete systems. Hence, by Theorem 14, there is a corresponding induction technique for continuous systems and for hybrid systems. And, indeed, *differential invariants* [Pla10a, Pla12d] are such an induction technique for differential equations that has been used very successfully for verifying hybrid systems with more advanced differential equations [PC08, PC09a, PC09b, PQ09, Pla10b, MGP13]. In fact, differential invariants had already been introduced in 2008 [Pla10a] before Theorem 14 was proved [Pla12b], but Theorem 14 implies that a differential invariant induction technique has to exist. These results also show that there are sound ways of using discretization for differential equations [Pla12b] and that numerical integration schemes like, e.g., Euler's method or more elaborate methods can be used for hybrid systems verification, which is not at all clear a priori due to inherent numerical approximation errors, which may blur decisions either way [PC07].

Challenges with Hybrid Relations. Theorem 15 is a hybrid miracle. Naïve ways of relating discrete and continuous dynamical systems are bound to fail. It is, for example, not generally the case that a property F transfers from a continuous system to its Euler discretization, nor vice versa. That is, neither the following equivalence nor the left-to-right implication nor the right-to-left implication generally holds:

$$[x' = \theta]F \overset{?}{\leftrightarrow} [(x := x + h\theta)^*]F \tag{6}$$

This formula would relate a property F of a continuous dynamical system $x' = \theta$ to property F of its Euler discretization $(x := x + h\theta)^*$ with discretization step size $h > 0$ *if only it were true*. Unfortunately, as such, the formula is not generally valid. Fig. 2 illustrates a counterexample to formula (6) from prior work [Pla12b], to which we refer for further details. The error of the Euler discretization grows quickly compared to the true solution in Fig. 2. For example, $F \equiv (x^2 + y^2 = 1)$ is an invariant of the true

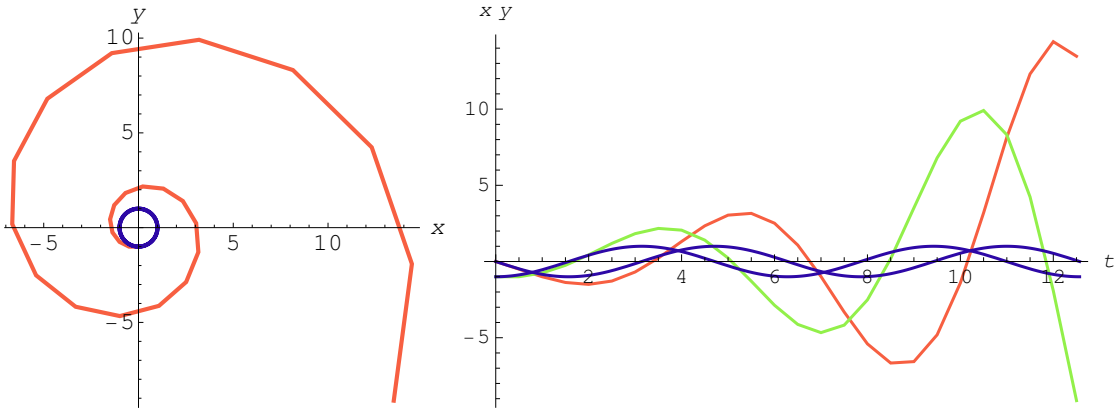

Figure 2: **(left)** Dark circle shows true solution, light line segments show Euler approximation for discretization step $h = \frac{1}{2}$ **(right)** Dark true bounded trigonometric solution and Euler approximation in lighter colors with increasing errors over time t

solution but not its approximation. On the bright side, the error can be smaller for *some* (not all) smaller discretization steps h and the error is quite reasonable for a certain period of time.

A. Löwenheim-Skolem-Herbrand-Theory in a Nutshell

The value of a logical formula is subject to interpretation in the semantics of the logic. The truth-value of a formula generally[3] depends on the interpretation of its symbols (such as its free variables or function symbols). In a certain sense maybe the most naïve interpretation of first-order logic interprets all terms as themselves. Such an interpretation I is called *Herbrand model*. It stubbornly interprets a term $f(g(a), h(b))$ in the logic as itself: $[\![f(g(a), h(b))]\!]_I = f(g(a), h(b))$. And likewise for all other ground terms.

That may sound like a surprising and stubborn interpretation. But, even more surprisingly, it is not at all an uninsightful one, at least for first-order logic. So insightful, that it even deserves a name: Herbrand models. Certainly, it is one of the many permitted interpretations.

[3] Except of the best formulas, which are valid, i.e. true in all interpretations. But we still first need to understand how those valid formulas are interpreted before we could call them valid.

Definition 16 (Herbrand Model)**.** An interpretation I is called *Herbrand model* if it has the free semantics for ground terms, i.e.:

1. The domain D is the ground terms (i.e. terms without variables) $\mathrm{Trm}^0(\Sigma)$ over Σ
2. $I(f) : D^n \to D; (t_1, \ldots, t_n) \mapsto f(t_1, \ldots, t_n)$ for each function symbol f of arity n

Let Γ be a set of closed universal formulas. $\mathrm{Trm}^0(\Sigma)(\Gamma)$ is the set of all ground term instances of the formulas in Γ, i.e. with (all possible) ground terms in $\mathrm{Trm}^0(\Sigma)$ instantiated for the variables of the universal quantifier prefix.

$$\mathrm{Trm}^0(\Sigma)(\Gamma) = \{\phi(t_1, t_2, \ldots, t_n) \;:\; (\forall x_1 \forall x_2 \ldots \forall x_n \, \phi(x_1, x_2, \ldots, x_n)) \in \Gamma \\ t_1, \ldots, t_n \in \mathrm{Trm}^0(\Sigma), \text{ for any } n \in \mathbb{N}\}$$

That is, for any $n \in \mathbb{N}$ and for any formula

$$\forall x_1 \forall x_2 \ldots \forall x_n \, \phi(x_1, x_2, \ldots, x_n)$$

in Γ and for any ground terms $t_1, \ldots, t_n \in \mathrm{Trm}^0(\Sigma)$, the set $\mathrm{Trm}^0(\Sigma)(\Gamma)$ contains the following ground instance of ϕ:

$$\phi(t_1, t_2, \ldots, t_n)$$

Theorem 17 (Herbrand [Her30])**.** *Let Γ be a (suitable) set of first-order formulas (i.e. closed* universal formulas *without equality and with signature Σ having at least one constant). Then*

Γ *has a model* $\iff$ Γ *has a Herbrand model*
$\iff$ *ground term instances* $\mathrm{Trm}^0(\Sigma)(\Gamma)$ *of* Γ *have a model*

Using the Herbrand theorem twice gives:
Γ has a model $\iff$ ground term instances $\mathrm{Trm}^0(\Sigma)(\Gamma)$ of Γ have a Herbrand model

Theorem 8 (Semidecidability of first-order logic [Göd29])**.** *Validity in first-order logic is semidecidable, i.e. there is an algorithm that, given any formula ϕ of first-order logic, correctly reports whether ϕ is valid or not and that also terminates for all ϕ that indeed valid. The algorithm will not generally terminate when ϕ is not valid.*

Proof. For suitable first-order formulas F (i.e. $\neg F$ satisfies the assumptions of Theorem 17), semidecidability follows from the following reductions:

F valid $\iff$ $\neg F$ unsatisfiable
$\iff$ $\mathrm{Trm}^0(\Sigma)(\neg F)$ have no model — by Theorem 17
$\iff$ some finite subset of $\mathrm{Trm}^0(\Sigma)(\neg F)$ has no Herbrand model — by Corollary 4

Thus, it remains to consider the assumptions in Theorem 17 whether first-order formulas that are not suitable can be turned into formulas that are suitable. First of all, Σ can be assumed without loss of generality to have at least one constant symbol for, otherwise, a constant can be added to Σ without changing validity of F. Furthermore, a formula F is valid iff its universal closure is, where the universal closure of a formula F is obtained by prefixing F with universal quantifiers $\forall x$ for each variable x that occurs free in F. Finally, existential quantifiers in first-order formula $\neg F$ can be removed without affecting satisfiability by Skolemization, which introduces new function symbols much like the quantifier proof rules from Lecture 6 did. □

Note 22 (Limitations of Herbrand models)**.** *Herbrand models are not the cure for everything in first-order logic, because they unwittingly forget about the intimate relationship of the term $2+5$ to the term $5+2$ and, for that matter, to the term $8-1$. All those terms ought to denote the same identical object, but end up denoting different ground terms in Herbrand models. In particular, a Herbrand model would not mind at all if a unary predicate p would hold of $2+5$ but not hold for $5+2$ even though both ought to denote the same object. Thus, Herbrand models are a little weak in arithmetic, but otherwise incredibly powerful.*

Herbrand's theorem has a second form with a close resemblance to the core arguments of quantifier elimination in first order logic of real arithmetic from Lecture 18 and Lecture 19.

Theorem 5 (Herbrand's theorem: Herbrand disjunctions [Her30])**.** *For a quantifier-free formula $\phi(x)$ of a free variable x without equality*

$\exists x\, \phi(x)$ *valid* $\iff$ $\phi(t_1) \vee \cdots \vee \phi(t_n)$ *valid for some* $n \in \mathbb{N}$ *and ground terms* $t_1, \ldots, t_n$

Proof. The proof follows directly from Theorem 17 and Corollary 4:

$\exists x\, \phi(x)$ valid
$\iff$ $\neg\exists x\, \phi(x)$ unsatisfiable
$\iff$ $\forall x\, \neg\phi(x)$ has no model
$\iff$ $\mathrm{Trm}^0(\Sigma)(\forall x\, \neg\phi(x))$ has no model — by Theorem 17
$\iff$ $\{\neg\phi(t) \,:\, t \text{ ground term}\}$ has no model — by definition
$\iff$ $\{\neg\phi(t_1), \ldots, \neg\phi(t_n)\}$ has no model for some $t_1, \ldots, t_n$ and some n — by Corollary 4
$\iff$ $\neg\phi(t_1) \wedge \cdots \wedge \neg\phi(t_n)$ has no model for some n and some $t_1, \ldots, t_n$
$\iff$ $\phi(t_1) \vee \cdots \vee \phi(t_n)$ valid for some n and some $t_1, \ldots, t_n$ □

Theorem 5 continues to hold for first-order formulas $\exists x_1 \ldots \exists x_n\, \phi(x_1, \ldots, x_n)$ with multiple existential quantifiers. More general forms of the Herbrand theorem hold for arbitrary first-order formulas that are not in the specific form assumed above [Her30].

These more general Herbrand theorems won't be necessary for us, because, for validity purposes, first-order formulas can be turned into the form $\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$ with quantifier-free $\phi(x_1, \ldots, x_n)$ by introducing new function symbols for the universal quantifiers using essentially the quantifier proof rules from Lecture 6:[4]

$$(\forall \mathrm{r}) \ \frac{\Gamma \vdash \phi(s(X_1, .., X_n)), \Delta}{\Gamma \vdash \forall x \, \phi(x), \Delta} \, {}_1 \qquad (\exists \mathrm{l}) \ \frac{\Gamma, \phi(s(X_1, .., X_n)) \vdash \Delta}{\Gamma, \exists x \, \phi(x) \vdash \Delta} \, {}_1$$

[1] s is a new (Skolem-Herbrand) function and $X_1, .., X_n$ are all (existential) free logical variables of $\forall x \, \phi(x)$.

The clou about quantifier rules ∀r,∃l is that they preserve validity. By soundness, if their premiss is valid then so is their conclusion. Yet, in the case of rules ∀r,∃l the converse actually holds as well. If their conclusion is valid then so is their premiss. For rule ∀r, for example, the conclusion says that $\phi(x)$ holds for all values of x in all interpretations where Γ holds and Δ does not. Consequently, in those interpretations, $\phi(s(X_1, .., X_n))$ holds whatever the interpretation of s is, because s is a fresh function symbol, which, thus, does not appear in Γ, Δ.

Lemma 18 (Herbrandization). *With each first-order logic formula ψ, a formula*

$$\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$$

with quantifier-free $\phi(x_1, \ldots, x_n)$ can be associated effectively that is valid if and only if ψ is. The formula $\exists x_1 \ldots \exists x_n \, \phi(x_1, \ldots, x_n)$ uses additional function symbols that do not occur in ψ.

The procedure in this proof will always succeed but it enumerates the ground terms for instantiation rather blindly, which can cause for quite a bit of waiting. Nevertheless, refinements of this idea lead to very successful automated theorem proving techniques for first-order logic known as *instance-based methods* [BT10], which restrict the instantiation to instantiation-on-demand in various ways to make the procedure more goal-directed. There are also many successful automatic theorem proving procedures for first-order logic that are based on different principles, including tableaux and resolution [Fit96].

Exercises

Exercise 1. The arguments for incompleteness and incompactness of dL hardly depend on dL, but, rather, only on dL's ability to characterize natural numbers. Incompleteness

[4] The new function symbols are usually called Skolem functions and the process called Skolemization, because Thoralf Skolem introduced them in the first correct proof of the Skolem-Löwenheim theorem [Sko20]. Strictly speaking, however, Herbrand functions and Herbrandization are the more adequate names, because Jacques Herbrand introduced this dual notion for the first proof of the Herbrand theorem [Her30]. Skolemization and Herbrandization are duals. Skolemization preserves satisfiability while Herbrandization preserves validity.

and incompactness hold for other logics that characterize natural numbers due to a famous result of Gödel [Göd31]. Both the discrete and the continuous fragment of $\mathsf{d}\mathcal{L}$ can characterize the natural numbers [Pla08].

1. Show that the natural numbers can be characterized in the discrete fragment of $\mathsf{d}\mathcal{L}$, i.e. only using assignments and repetition.
2. Then go on to show that the natural numbers can also be characterized in the continuous fragment of $\mathsf{d}\mathcal{L}$, i.e. using only differential equations.
3. Conclude from this that both the discrete and the continuous fragment of $\mathsf{d}\mathcal{L}$ are not compact, nor is any other logic that can characterize the natural numbers.

References

[And02] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer, 2nd edition, 2002.

[BT10] Peter Baumgartner and Evgenij Thorstensen. Instance based methods - a brief overview. *KI*, 24(1):35–42, 2010.

[Col07] Pieter Collins. Optimal semicomputable approximations to reachable and invariant sets. *Theory Comput. Syst.*, 41(1):33–48, 2007. doi:10.1007/s00224-006-1338-3.

[Coo78] Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput.*, 7(1):70–90, 1978.

[Dij70] Edsger Wybe Dijkstra. Structured programming. In John Buxton and Brian Randell, editors, *Software Engineering Techniques. NATO Software Engineering Conference 1969*. NATO Scientific Committee, 1970.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, 2nd edition, 1996.

[Göd29] Kurt Gödel. *Über die Vollständigkeit des Logikkalküls*. PhD thesis, Universität Wien, 1929.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Mon. hefte Math. Phys.*, 37:349–360, 1930.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Mon. hefte Math. Phys.*, 38:173–198, 1931.

[Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la Société des Sciences et des Lettres de Varsovie, Class III, Sciences Mathématiques et Physiques*, 33:33–160, 1930.

[HMP77] David Harel, Albert R. Meyer, and Vaughan R. Pratt. Computability and completeness in logics of programs (preliminary report). In *STOC*, pages 261–268. ACM, 1977.

[LIC12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[Lin69] Per Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969. doi:10.1111/j.1755-2567.1969.tb00356.x.

[Löw15] Leopold Löwenheim. Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, 76:447–470, 1915.

[MGP13] Stefan Mitsch, Khalil Ghorbal, and André Platzer. On provably safe obstacle avoidance for autonomous robotic ground vehicles. In Paul Newman, Dieter Fox, and David Hsu, editors, *Robotics: Science and Systems*, 2013.

[PC07] André Platzer and Edmund M. Clarke. The image computation problem in hybrid systems model checking. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *HSCC*, volume 4416 of *LNCS*, pages 473–486. Springer, 2007. doi:10.1007/978-3-540-71493-4_37.

[PC08] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *LNCS*, pages 176–189. Springer, 2008. doi:10.1007/978-3-540-70545-1_17.

[PC09a] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Form. Methods Syst. Des.*, 35(1):98–120, 2009. Special issue for selected papers from CAV'08. doi:10.1007/s10703-009-0079-8.

[PC09b] André Platzer and Edmund M. Clarke. Formal verification of curved flight collision avoidance maneuvers: A case study. In Ana Cavalcanti and Dennis Dams, editors, *FM*, volume 5850 of *LNCS*, pages 547–562. Springer, 2009. doi:10.1007/978-3-642-05089-3_35.

[Pla07] André Platzer. A temporal dynamic logic for verifying hybrid system invariants. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 4514 of *LNCS*, pages 457–471. Springer, 2007. doi:10.1007/978-3-540-72734-7_32.

[Pla08] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.*, 41(2):143–189, 2008. doi:10.1007/s10817-008-9103-8.

[Pla10a] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010. doi:10.1093/logcom/exn070.

[Pla10b] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla10c] André Platzer. Quantified differential dynamic logic for distributed hybrid systems. In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *LNCS*, pages 469–483. Springer, 2010. doi:10.1007/978-3-642-15205-4_36.

[Pla10d] André Platzer. Theory of dynamic logic. Lecture Notes 15-816 Modal Logic, Carnegie Mellon University, 2010. URL: http://www.cs.cmu.edu/~fp/courses/15816-s10/lectures/25-DLtheo.pdf.

[Pla11] André Platzer. Stochastic differential dynamic logic for stochastic hybrid programs. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 431–445. Springer, 2011. doi:10.1007/978-3-642-22438-6_34.

[Pla12a] André Platzer. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. *Logical Methods in Computer Science*, 8(4):1–44, 2012. Special issue for selected papers from CSL'10. doi:10.2168/LMCS-8(4:17)2012.

[Pla12b] André Platzer. The complete proof theory of hybrid systems. In LICS [LIC12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12c] André Platzer. Logics of dynamical systems. In LICS [LIC12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pla12d] André Platzer. The structure of differential invariants and differential cut elimination. *Logical Methods in Computer Science*, 8(4):1–38, 2012. doi:10.2168/LMCS-8(4:16)2012.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

[Pla14] André Platzer. Analog and hybrid computation: Dynamical systems and programming languages. *Bulletin of the EATCS*, 114, 2014. URL: http://bulletin.eatcs.org/index.php/beatcs/article/viewFile/292/274.

[PQ09] André Platzer and Jan-David Quesel. European Train Control System: A case study in formal verification. In Karin Breitman and Ana Cavalcanti, editors, *ICFEM*, volume 5885 of *LNCS*, pages 246–265. Springer, 2009. doi:10.1007/978-3-642-10373-5_13.

[Sch12] Peter H. Schmitt. Formale Systeme. Vorlesungsskriptum Fakultät für Informatik , Universität Karlsruhe, 2012.

[Sko20] Thoralf Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theorem über dichte Mengen. *Videnskapsselskapet Skrifter, I. Matematisk-naturvidenskabelig Klasse*, 6:1–36, 1920.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.