Descriptive Control Theory: A Proposal

Sicun Gao

Abstract

Logic is playing an increasingly important role in the engineering of real-time, hybrid, and cyber-physical systems, but mostly in the form of posterior verification and high-level analysis. The core methodology in the design of real-world systems consists mainly of control theory and numerical analysis, and has remained mostly free of logic and formal approaches. As a result, besides facing extreme difficulty in guaranteeing the reliability of these systems, engineers are also missing out on the computational power of logic-based methods that has greatly advanced in the past decades. To change this situation, we need a logical and computational foundation for control theory. The name "descriptive control theory" emphasizes the overarching theme of using logic to express, analyze, and solve problems in control theory. If the program is successfully carried out, logical approaches will significantly extend existing engineering methods towards a unified methodology for handling nonlinear and hybrid systems, and bring design automation and reliability to an unprecedented level in the broad field of engineering.

1 Introduction

To engineer a system is, ideally, to prove a theorem. While such a statement can often be made exact for software systems in the sense of Curry-Howard, it seems hardly applicable to the engineering of *real-world systems*: systems that physically engage us in safety-critical ways, from airplanes to nuclear plants to cardiac pacemakers. Admittedly, the study of formal methods for real-time, hybrid, and cyber-physical systems has given logic an increasingly important role in real-world system engineering, but mostly in the form of posterior verification or very high-level analysis. The core methodology in the design of these systems consists mainly of control theory and numerical analysis, and has remained mostly free of logic and formal approaches. An obvious consequence is the well-known difficulty in guaranteeing the reliability of these systems. An equally fundamental problem, which appears less often recognized, is that engineers are missing out on the computational power of logic-based methods that has greatly advanced in the past decades. I believe that logic and formal approaches, backed by their underlying computational engines, have the power of bringing design automation and reliability to an unprecedented level in the broad field of engineering. Ultimately, the process of engineering should automatically produce logical derivations of correctness of the constructed systems, and in this process, logical decision procedures should play a part as indispensable as calculus, linear algebra, and optimization. When such a unification of methodologies happens, the systems that we are able to build will be orders of magnitude more complex and reliable, and the physical world and the computational world will quickly converge.

Towards such a goal, we need to develop a logical and computational foundation for control theory. I use the name "descriptive control theory" to emphasize the overarching theme of using

logic to express, analyze, and solve problems in control theory. The program is to study control theory problems through their logical encodings, and use such encodings as a portal for bringing in suitable computational engines to solve the problems. Besides providing a formal foundation for existing methods in control theory, an important theme is to show the strength of logical decision procedures for generalizing these methods from dealing with polynomial-time solvable problems to NP-hard problems, and as a result, towards handling nonlinear and hybrid dynamical systems.

Tarski's surprising result on the decidability of real arithmetic initiated the use of logic methods for solving problems in Euclidean spaces. Through quantifier elimination, the difficult tasks of geometric theorem proving become mechanizable, which significantly influenced the theory in practical fields such as robotic manipulation. The high computational complexity of the logical problems is still a bottleneck for delivering practical solutions. A fundamental problem of applying Taski's results to general control problems is the limitation of expressiveness of real arithmetic. Difficulty becomes apparent when we use first-order logic to reason about nontrivial dynamical systems: the first-order theory of real arithmetic with trigonometric functions is already highly undecidable. Consider the simple question of whether a one-dimensional continuous process, governed by some differential equation $\dot{x} = f(x,t)$, can start from $x = 0$ and reach $x = 1$ in finite time. A direct first-order encoding would require a formula such as the following:

$$\exists x_0 \exists t \exists x_t \left(x_0 = 0 \ \wedge \ x_t = x_0 + \int_0^t f(x(s), s) \mathrm{d}s \ \wedge \ x_t = 1 \right).$$

Given that solutions of differential equations are rarely polynomials, and most likely not analytically solvable, we can not directly reason with such logic formulas, and a logical approach seems hardly useful.

We have developed a framework to bypass this core difficulty. Noting that the reasoning of continuous systems naturally involves numerical errors, we realize that

$$\exists x_0 \exists t \exists x_t \left(|x_0| \leq \delta_1 \ \wedge \ |x_t - (x_0 + \int_0^t f(x(s), s) \mathrm{d}s)| \leq \delta_2 \ \wedge \ |x_t - 1| \leq \delta_3 \right)$$

is what really matters in practice, for some choices of numerical error bounds $\delta_1, \delta_2, \delta_3 \in \mathbb{Q}^+$. Formalizing this observation, we have developed the theory of *delta-decisions* over the reals [8, 7]. In this new theory we relax the standard decision problem to ask whether a sentence is true or its "delta-strengthening" is false. Namely, we allow one-sided, delta-bounded numerical errors in logical decisions. With this change, the decision problem for logic formulas with arbitrary numerically computable functions, in the exact sense as developed in computable analysis, becomes decidable in bounded domains. The complexity of the delta-decision problems are also comparable to their discrete counterparts. These results stand in sharp contrast to the negative results for the standard decision problems, and provide the basis for developing a logical approach of control theory. We give a brief technical review of the theory of delta-decisions in Section 2.1.

Equipped with an expressive logic and decidability results, we can encode a wide range of control problems and study them from logical and computational perspectives. In particular, the following three directions form the main themes of descriptive control theory:

- **Descriptive Complexity.** The goal is to formalize the problems in control theory and to understand their computational complexity. We use the descriptive complexity approach: define a suitable logical language to express the control problems, such that their "practical"

computational complexity can be easily derived through the descriptions. Here, the measure of "practical" complexity is defined through delta-decisions of the logic formulas, which we will give more details below.

- **Logical Foundation.** The goal is to seek a logical foundation for existing theorems and methods in control theory. A natural approach is to follow the program of reverse mathematics to characterize control theory in suitable subsystems of second-order arithmetic, and for decidable theorems, decide their proof complexity. Such a foundation would also reveal computational content in these theorems. Moreover, the proofs should be formalized in an interactive theorem prover, which can be the basis for certifying practical control designs.

- **Computational Engine.** The goal is to develop practical decision procedures for the logic formulas involved, which can serve as general algorithms for solving the control problems. As logical decision procedures usually target at hard problems (NP-hard and beyond), they may significantly extend existing methods in control theory, which mostly rely on polynomial-time algorithms. Moreover, the decision procedures should always produce proofs that can be validated through interactive theorem provers, and thus guarantee correctness-by-construction.

In what follows, I will first cover the background in Section 2, and then discuss these three components in detail in Section 3, 4, 5. For each topic, I will outline two specific goals. The goals will combine theoretical investigations and practical implementations 6.

2 Background

2.1 Delta-Decidability over the Reals

In this section, we review our theory of delta-decisions over the reals [8]. The theory allows us to consider first-order formulas over the reals with arbitrary *Type 2 computable functions*, a notion that has been well-developed in computable analysis [18, 14]. We will introduce the notion of Type 2 computability first, and then give the main results about delta-decisions.

Following computable analysis, we can encode any real number as an infinite sequence of rational numbers. For each real number x, a *name* of x is any function $\phi : \mathbb{N} \to \mathbb{D}$ that binary-converges to x, namely,

$$\forall n \in \mathbb{N}, |\phi(n) - x| \leq 2^{-n}.$$

We can then compute a real function $f : \mathbb{R} \to \mathbb{R}$ if there is an oracle Turing machine M that, given the name of any argument $x \in \mathbb{R}$ of the function, computes the name of its value $f(x)$ up to an arbitrary digit in the following way:

Definition 2.1 (Type 2 Computable Functions)**.** A real function $f : \mathbb{R} \to \mathbb{R}$ is *Type 2 computable*, if there is a function-oracle Turing machine M such that for every $x \in \mathbb{R}$ and every name ϕ of x, given any $i \in \mathbb{N}$, the machine uses ϕ as an oracle, and n as the input, and computes a rational number $M^{\phi}(i) \in \mathbb{Q}$, such that $|M^{\phi}(i) - f(x)| \leq 2^{-i}$. In other words, M computes a function ψ that binary-converges to $f(x)$ as its representation.

A most important property of computable functions is that they must have a computable modulus of continuity. Most common continuous real functions are Type 2 computable, including: polynomials with computable coefficients, exponential, trigonometric, square root, and logarithm

functions, absolute value, and solution functions of Lipschitz-continuous ordinary differential equations. The complexity of Type 2 computable functions is also well studied. Intuitively, a real function $f : [0,1] \to \mathbb{R}$ is (uniformly) P-computable (PSPACE-computable), if it is computable by an oracle Turing machine M_f that halts in polynomial-time (polynomial-space) for every $i \in \mathbb{N}$ and every $\vec{x} \in \mathrm{dom}(f)$. We denote this class of functions as $\mathsf{P}_{\mathsf{C}[0,1]}$. The definitions of other classes such as $\mathsf{NP}_{\mathsf{C}[0,1]}, \mathsf{PSPACE}_{\mathsf{C}[0,1]}$ are similar. Omitting the formal details, we point out that most common real functions reside in $\mathsf{P}_{\mathsf{C}[0,1]}$: absolute value, polynomials, binary max and min, exp, and sin are all in $\mathsf{P}_{\mathsf{C}[0,1]}$. Moreover, it has been shown that solutions of Lipschitz-continuous differential equations are $\mathsf{PSPACE}_{\mathsf{C}[0,1]}$-complete [13, 14].

We can now consider the first-order language $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$ over the real numbers, which allows the use of arbitrary Type 2 computable functions. This language is rich enough for expressing a wide range of continuous and hybrid systems and their properties. We write $\mathcal{F}$ to denote an arbitrary collection of symbols representing Type 2 computable functions over $\mathbb{R}^n$ for various n. We always assume that $\mathcal{F}$ contains at least the constant 0, unary negation, addition, and the absolute value. (Constants are seen as constant functions.) Let $\mathcal{L}_\mathcal{F}$ be the signature $\langle \mathcal{F}, > \rangle$. $\mathcal{L}_\mathcal{F}$-formulas are always evaluated in the standard way over the corresponding structure $\mathbb{R}_\mathcal{F} = \langle \mathbb{R}, \mathcal{F}, > \rangle$. It is not hard to see that we only need to use atomic formulas of the form $t(x_1, ..., x_n) > 0$ or $t(x_1, ..., x_n) \geq 0$, where $t(x_1, ..., x_n)$ are built up from functions in $\mathcal{F}$. We can give an explicit definition of $\mathcal{L}_\mathcal{F}$-formulas as follows.

Definition 2.2 ($\mathcal{L}_\mathcal{F}$-Formulas)**.** Let $\mathcal{F}$ be a collection of Type 2 functions, which contains at least 0, unary negation -, addition +, and absolute value $|\cdot|$. We define:

$$t := x \mid f(t(x_1, ..., x_n)), \text{ where } f \in \mathcal{F}, \text{ possibly constant;}$$
$$\varphi := t > 0 \mid t \geq 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \forall x \varphi.$$

We use the notation of *bounded quantifiers*, defined as $\exists^{[u,v]} x.\varphi =_{df} \exists x.(u \leq x \wedge x \leq v \wedge \varphi)$ and $\forall^{[u,v]} x.\varphi =_{df} \forall x.((u \leq x \wedge x \leq v) \to \varphi)$. We say a sentence is bounded if it only involves bounded quantifiers.

On this normal form of $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas, we can then define the δ-variant of them as follows.

Definition 2.3 (δ-Variants)**.** Let $\delta \in \mathbb{Q}^+ \cup \{0\}$, and φ a bounded $\mathcal{L}_\mathcal{F}$-sentence of the form

$$\varphi : \ Q_1^{I_1} x_1 \cdots Q_n^{I_n} x_n . \psi[t_i(\vec{x}, \vec{y}) > 0; t_j(\vec{x}, \vec{y}) \geq 0],$$

where $i \in \{1, ...k\}$ and $j \in \{k+1, ..., j\}$. The *δ-strengthening* $\varphi^{+\delta}$ of φ is defined to be the result of replacing each atomic formula $t_i > 0$ by $t_i > \delta$ and each atomic formula $t_j \geq 0$ by $t_j \geq \delta$, that is,

$$\varphi^{+\delta} : \ Q_1^{I_1} x_1 \cdots Q_n^{I_n} x_n . \psi[t_i(\vec{x}, \vec{y}) > \delta; t_j(\vec{x}, \vec{y}) \geq \delta],$$

where $i \in \{1, ...k\}$ and $j \in \{k+1, ..., j\}$. Similarly, the *δ-weakening* $\varphi^{-\delta}$ of φ is defined to be the result of replacing each atomic formula $t_i > 0$ by $t_i > -\delta$ and each atomic formula $t_j \geq 0$ by $t_j \geq -\delta$, that is,

$$\varphi^{-\delta} : \ Q_1^{I_1} x_1 \cdots Q_n^{I_n} x_n . \psi[t_i(\vec{x}, \vec{y}) > -\delta; t_j(\vec{x}, \vec{y}) \geq -\delta].$$

Note that in the definition, the bounds on the quantifiers are not changed. Our main theorem is the following.

Theorem 2.1 (δ-Decidability)**.** There is an algorithm which, given any bounded $\mathcal{L}_\mathcal{F}$-sentence φ and $\delta \in \mathbb{Q}^+$, correctly returns one of the following two answers:

- "True": φ is true.
- "δ-False": $\varphi^{+\delta}$ is false.

Equivalently, there is an algorithm which, given any bounded φ and $\delta \in \mathbb{Q}^+$, correctly returns one of the following two answers:

- "δ-True": $\varphi^{-\delta}$ is true.
- "False": φ is false.

Note that the two cases can overlap. If φ is true and $\varphi^{+\delta}$ is false, then the algorithm is allowed to return either one. The proof idea is that for any formula φ, the strictification of φ is equivalent to the formula $\alpha(\varphi) > 0$. Whether this holds cannot, in general, be determined algorithmically, But given a small δ, we *can* make a choice between the overlapping alternatives $\alpha(\varphi) > 0$ and $\alpha(\varphi) < \delta$, and this is enough to solve the relaxed decision problem. The dual versions of the theorem are useful in different contexts. The weakening is used for confirming that a certain property holds. The strengthening is used for synthesizing parameters. The differences will be discussed in later sections. We also have complexity results such as follows.

Theorem 2.2. Let $\mathcal{F}$ be a class of computable functions. Let S be a class of $\mathcal{L}_{\mathcal{F}}$-sentences, such that for any φ in S, the terms in $\varphi_{[0,1]}$ are computable in complexity class C where $\mathsf{P}_{\mathsf{C}[0,1]} \subseteq \mathsf{C} \subseteq \mathsf{PSPACE}_{\mathsf{C}[0,1]}$. Then, for any $\delta \in \mathbb{Q}^+$, the δ-decision problem for bounded Σ_n-sentences in S is in $(\mathsf{\Sigma_n^P})^{\mathsf{C}}$.

As corollaries, we have the following completeness results for signatures of interest.

Corollary 2.1. Let $\mathcal{F}$ be a set of P-computable functions (which, for instance, includes exp and sin). The δ-decision problem bounded Σ_n-sentences in $\mathcal{L}_{\mathcal{F}}$ is $\mathsf{\Sigma_n^P}$-complete.

Corollary 2.2. Suppose $\mathcal{F}$ consists of Lipschitz-continuous ODEs over compact domains. The δ-decision problem for bounded $\mathcal{L}_{\mathcal{F}}$-sentences is PSPACE-complete.

Note that these complexity results stand in high contrast to the standard undecidability of the problems. In fact, they bring the hope of solving these formulas with practical decision procedures, as they are not beyond the complexity capacity of SAT and SMT solvers. Based on the theory, we have implemented a practical solver dReal that solves a wide range of nonlinear formulas, which will be discussed in Section 2.3.

2.2 Control Theory

Control theory studies methods of regulating dynamical systems to achieve desired goals. Some detailed examples and results of control theory will accompany the discussion when needed. The main topics in control theory are:

- Analyze the properties of a dynamical system under external controls, such as their stability, controllability, and observability.
- Design controllers that can regulate a dynamical system to satisfy certain specifications, such as stabilizing a system.

A complete theory for linear dynamical systems has been well developed. The behavior of a linear system, such as stability and controllability, can be completely understood through properties of the matrix that determines its dynamics. However, beyond linear systems, existing methods have significant difficulty in dealing with nonlinear and hybrid systems. The goal of developing a logic-based methodology is to tackle the difficulty of existing methods. The logical approaches naturally bring in the power of computational engines that can extend existing methods from linear to nonlinear and hybrid systems. An important goal is to ensure that it is a strict extension: existing methods in control theory will be formalized and considered as algorithms for specific subclasses of the problems. The ultimate goal is to develop a new methodology that combines automation and certification. When the solving process is mechanized, checking the correctness of the mechanism becomes a simpler way of ensuring correctness of the results.

2.3 Computational Framework

The theory of delta-decidability not only provides a new perspective to look at decision problems over the reals, but also guides the development of concrete algorithms for solving the delta-decision problems. We say an algorithm is *delta-complete*, if it always terminates with correct delta-decisions. We have formally analyzed constraint solving algorithms and formulated formal conditions under which they are delta-complete. By combining such algorithms with logical decision procedures, we obtain decision procedures that can exploit the most out of both symbolic and numerical algorithms.

Based on the theory, we have developed a practical solver dReal [9, 10] [1] that solves Σ_1-formulas in nonlinear theories over the reals in the framework of delta-complete decision procedures. Since 2012, dReal has been a very active project that combines the effort many collaborators [2] and successfully applied to many practical problems [9, 10, 12, 10]. An important feature of the dReal tool is that it produces logical proofs to certify its answers. Such proofs can be used to produce certification of the correctness of concrete control designs, which can be easily checked in an interactive theorem prover.

3 Descriptive Complexity

The first goal is to develop a logical description of problems in control theory. This step is the basis for both a logical foundation and a computational treatment of the problems. An immediate benefit of the encoding follows from the theory of delta-decisions over the reals: upper bounds on the complexity of these problems can be easily derived through their encoding in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$. For control design problems, goal is usually to find a function that satisfies certain goals. These problems are most naturally expressed with second-order quantification, which is the motivation for investigating second-order theories written within $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$.

I will first explain the approach of using first-order delta-decisions to encode control problems and deriving their complexity bounds, and then discuss the possibility of extending to a second-order formalism.

Related Work. Computational properties of most control problems are not clearly understood. The computational complexity of stability properties has been a topic of much recent investiga-

[1] http://dreal.cs.cmu.edu
[2] http://github.com/dreal

tion [3, 4, 5, 1, 16, 2]. A focus of existing work is to establish various hardness results, i.e., lower bounds on complexity. It is shown that stability of simple systems is hard or impossible to solve algorithmically. Such results are proved by reducing combinatorial problems over graphs or matrices to stability problems, which can be analyzed with techniques of standard complexity theory. A limitation is that reduction techniques are usually not suitable for establishing upper bounds on complexity, and indeed most questions about upper bounds are open [1].

3.1 Descriptive Complexity in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$

The majority of control problems ask for real functions that satisfy certain properties.

We need to justify the focus on the δ-perturbed version of the problems. The ability of reasoning with both δ-weakening and delta-strengthening gives us the ability of choosing the right perturbation to use.

- For the problems that are concerned with a positive property, such as stability, controllability, observability, the perturbation on the negative side strengthens the problems, and are in fact the problems that we would like to solve, rather than the precise ones.
- For problems that are concerned with finding a witness such that some property holds, such as an optimal controller, it is more important to solve the perturbation on the positive side. The interpretation would be that the synthesized plan

The study of stability properties of dynamical systems, for instance, can be fully described in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$. Here is an example of how this approach can be applied (more details in [11]).

Example 3.1. Following standard definition, a system is stable i.s.L. if given any ε, there exists δ such that for any initial value x_0 that is within δ from the origin, the system stays in ε-distance from the origin. Naturally, the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representation of stability in the sense of Lyapunov is encoded in the following way:

Definition 3.1 (L_stable)**.** We define the $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$-formula L_stable to be:

$$\forall^{[0,\infty)}\varepsilon\exists^{[0,\varepsilon]}\delta\forall^{[0,\infty)}t\forall x_0\forall x_t.\ (||x_0|| < \delta \wedge x_t = \int_0^t f(s)ds + x_0) \rightarrow ||x_t|| < \varepsilon.$$

The *bounded form* of L_stable is defined by bounding the quantifiers as:

$$\forall^{[0,e]}\varepsilon\exists^{[0,\varepsilon]}\delta\forall^{[0,T]}t\forall^{X}x_0\forall^{X}x_t.\ (||x_0|| < \delta \wedge x_t = \int_0^t f(s)ds + x_0) \rightarrow ||x_t|| < \varepsilon,$$

where $e, T \in \mathbb{R}^+$ and X is a compact set.

We can then define the δ-stability problem using the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representation:

Definition 3.2 (δ-Stability i.s.L.)**.** The δ-stability problem i.s.L. asks for one of the following answers:

- stable: The system is stable i.s.L. (L_stable is true).
- δ-unstable: Some δ-perturbation of L_stable is false.

We defined the *bounded* δ-stability problem by replacing L_stable with the bounded form of L_stable in the definition. Now, using the complexity of the formulas, we have the following complexity results for the bounded version of Lyapunov stability.

> **Theorem 3.1** (Complexity)**.** Suppose all terms in the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representation of a system are in Type 2 complexity class C. Then the bounded δ-stability problem i.s.L. resides in complexity class $(\Pi_3^\mathsf{P})^\mathsf{C}$.

This completes the example of obtaining complexity. ■

Note that in the example, the new problems are defined through delta-perturbations of their logical encoding. Thus the new definitions are dependent on the descriptive approach. The same methodology can be extended to a wide range of topics in control theory. The fields of optimal, robust, adaptive control design, which are themselves fields in control theory, provides plenty of opportunity for logical formalization.

3.2 Descriptions in $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$

For many control design problems, their direct encoding requires a second-order language that naturally extends $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$ with second-order quantifiers, which we call $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$. For instance, consider the problem of finding an optimal controller for a system $\dot{x} = f(x, u)$, with a cost function $g(x(t), u(t))$. Such a controller exists if the following $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$-formula is true:

$$\begin{aligned}&\exists U \forall U' \forall x_0 \forall x_t \forall x'_t \\ &\Bigg(\Big(x_t = x_0 + \int_0^t f(x(s), U(s))\mathrm{d}s \wedge\ x'_t = x_0 + \int_0^t f(x(s), U(s))\mathrm{d}s\ \wedge\ \phi(x_0, x_t)\ \wedge\ \phi(x_0, x'_t) \Big) \\ &\qquad\qquad \rightarrow \Big(\int_0^t g(x(s), U(s))\mathrm{d}s \leq \int_0^t g(x(s), U'(s))\mathrm{d}s \Big) \Bigg).\end{aligned}$$

U and U' denote control functions, and ϕ encodes some constraints on the initial and end states. The formula states that there exists a control function U such that any other control function U' that achieves the same goals would cost more than U, with respect to the cost function g.

We have conjectured in [8] that the second-order language $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$ is still delta-decidable under suitable interpretations of the second-order variables, since techniques for proving δ-decidability in the first-order case should apply to arbitrary compact metric spaces. We need to develop complexity analysis for $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$, which should systematically extend existing complexity results in computable analysis from real functions to functionals.

3.3 Main Goals in Descriptive Complexity

In sum, the two main goals under the theme of descriptive complexity are the following:

Goal 1. Express control theory in $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$ with a suitable interpretation (by choosing appropriate domains for the second-order variables), whose bounded δ-decision problem should be decidable.

Goal 2. Develop a descriptive complexity theory for $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$-formulas with respect to the δ-decisions and prove decidability results, possibility with preliminary complexity results.

The goals are important for the investigations in the other two components of the main theme. With the logical encodings, we can investigate the strength of theorems in control theory regarding these properties. At the same time, the encodings place the problems into different computational hierarchies, for which we can apply practical computational engines to solve.

4 Logical Foundation

The next task is to develop a logical foundation for main results in control theory. Mathematically, a formal foundation of control methods provides a basis for developing rigorous methodology. The goal is to develop a formal basis of control theory in suitable subsystems of second-order arithmetic, following the program of reverse mathematics. The benefit of focusing on weak theories, compared to ZFC or higher, is that we can analyze the proofs such that their computational content becomes clear. Computationally, formal proofs of main results in control theory can be used to certify correctness of concrete system designs.

Related Work. There are various approaches in providing logic-based approaches to dynamical systems and control theory. Most of the existing work focuses on expressive nonclassical languages that can encode complex behaviors of dynamical systems [15]. Our focus is to develop a formal foundation for the proofs of the main theorems, which has not been done before. Most of core of control theory only requires a classical first-order to second-order language. Once the study of the core problems is clear, more language constructs, such as temporal modalities, can be further introduced.

4.1 Proof Complexity and Reverse Mathematics

It is an interesting question to know the mathematical commitment that we have when proving facts about dynamical systems. Theorems in control theory typically give conditions about when a system satisfies certain control properties. There are two sides to control theory. One is the analysis side, and the other is the algebraic side. The proof complexity is lower on the algebraic side, which imposes stronger assumptions on the structure of the systems, for instance, linear systems represented as matrices. The analysis side requires stronger theories.

One presumably easy but still interesting first task is to study the reverse mathematics of the state-space control theory for linear dynamical systems. It is mostly clear that RCA_0 is enough to develop much of linear algebra [17]. Indeed, following the work in proof complexity [6], restricted forms of many facts from linear algebra, such as the Cayley-Hamilton theorem, have polynomial-time proofs. Most of the control theory for linear dynamical systems are stated in the language of matrices. For instance, consider the *Kalman test for controllability*. The Kalman test for controllability states that an n-dimensional linear system $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ is *controllable* iff the *Kalman matrix* $[B\ AB\ \cdots\ A^{n-1}B]$ is of rank n. Such theorems are quite straightforward consequences of the Cayley-Hamilton theorem. Their proof complexity should not be high.

The analysis aspect of control theory studies dynamical systems through properties of solutions of differential equations. For instance, consider the Lyapunov method for nonlinear stability as follows.

Example 4.1. Recall that the definition of stability in the sense of Lyapunov is given by the

following $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$-formula:

$$\varphi_S : \ \forall\varepsilon\exists^{[0,\varepsilon]}\delta\forall^{[0,\infty)}t\forall x_0\forall x_t.\ (||x_0|| < \delta \wedge x_t = \int_0^t f(s)ds + x_0) \to ||x_t|| < \varepsilon.$$

Let $V(p,x)$ be a function, parameterized by $p \in \mathbb{R}$, whose partial derivative $\partial V/\partial x$ is a Type 2 computable function. Let D be the parameter space for p and X be the state space of x. We then have the following $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula is a sufficient condition for stability in the sense of Lyapunov

$$\varphi_L : \ \exists p\forall x \left(V(p,x) \geq 0 \wedge V(p,0) = 0 \wedge \frac{\partial V(p,x)}{\partial x} f(x) \leq 0 \right)$$

The Lyapunov methods is based on the proof that $\varphi_L \to \varphi_S$ holds as a theorem over the reals, which can be analyzed in a suitable subsystem of second-order arithmetic. ■

Analysis methods cover most of the content in nonlinear control. The classical aspects of control theory based on frequency-domain analysis is worth investigating as well, which involves more complex analysis, and functional analysis. Many valuable results in this direction have been obtained in the work of Yokoyama [19]. For instance, Yokoyama showed that uniformly convergence of Fourier series for C^1-functions and L^2-convergence of Fourier series for continuous functions are equivalent to WKL_0 over RCA_0. In this perspective, it is interesting to understand whether modern control theory, while being more powerful, depends on weaker axioms than classical control theory. If that is the case, this would be a clear sign of improvement in the development of control theory.

4.2 Formal Proofs

An important part of establishing a logical foundation for control theory is the formalization of the proofs in an interactive theorem prover. The interactive proofs do not necessarily start from the weak subsystems, but the formalization in the previous section will certainly be valuable for the formalization of the proofs.

Candidate proof assistants include Coq, HOL, Isabel. Both the algebraic side and analysis side of control theory can find basic packages to start with. For instance, as a core theorem in matrix algebra, the Cayley-Hamilton theorem is proved in Coq and HOL. It is the basis of deriving other parts of the theory of linear systems.

Different consideration seems to favor different systems for developing the formalization. HOL has concise formulation, while Coq seems to be attractive to a wider community and would be a good choice for the future steps of the integration into proofs of correctness of control software. Another option is to work on top of the new theorem prover "Lean" (http://leanprover.net), a project recently started by Leonardo de Moura. The design principle is to have a framework that can use automatic solvers at the backend as engines in the proofs as much as possible. The benefit of working with Lean is that I can take an active role in developing the theorem prover itself which can help the development of a control theory library.

With a suitable choice of the proof environment, the goal is to have a control theory library. An ideal plan is to produce a textbook-like document that develops the core part of control theory with all theorems and proofs formalized in the proof assistant of choice. Theorems in control theory that requires formalization includes stability, observability, controllability for linear systems as previously mentioned, Lyapunov theory for nonlinear systems, and various results in optimal, robust, and adaptive control.

Besides the mathematical value of such formalization, An important use of the formal proofs is computational: these proofs can serve as a basis for formal verification of practical control designs. An interesting question is how to connect proofs of the main theorems to automated proofs of the concrete designs.

4.3 Main Goals in Logical Foundation

Goal 3. Categorize theorems in control theory in suitable subsystems of second-order arithmetic, or find proof complexity for decidable theorems concerning linear systems.

Goal 4. Formalize the main theorems in control theory in an interactive theorem prover.

5 Computational Engine

The main benefit of using a logical approach to express and formalize control theory is that it is backed by the use of computational engines. The methodology becomes that we can express a problem formally, and then the problem is solved using decision procedures for the logic theory. In this way, decision procedures for the logic formulas become generic algorithms for the control problems.

It is important to make sure that existing methods in control theory can be used as partial algorithms for subclasses of these formulas. A challenge is to ensure that algorithms based on matrix operations and convex optimization, for instance, can all be suitably called by the logic solver. To make sure of this in a solver, this requires detailed analysis of the numerical algorithms so that delta-completeness can be ensured. The additional benefit, besides automation, is that these decision procedures should automatically produce witnesses or proofs for their answers, such that the correctness of the full control design can be certified with a formal proof. For sat answers, one can simply plug in the solutions and check the correctness (up to delta-bounded errors). For unsat answers, one has to produce a proof of refutations that have been found by the solver. This requires detailed analysis of numerical algorithms.

In the next steps, a main target is Σ_2-sentences in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$, and also to handle important classes in $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$, at least up to the point that the existing algorithms in control design can be used. This involves the use of methods from calculus of variations, dynamic programming, etc. An interesting point to note is that the problems here all become generalizations of optimization problems. Scalar optimization problems corresponds to $\exists\forall$ formulas with one single existentially quantified variable.

5.1 From Σ_1 to Σ_2 Problems

As mentioned in the background section, we have developed the framework of *δ-decision procedures* for solving Σ_1-sentences in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$ [7], and implemented a practical solver dReal based on the theory. Many standard control-theoretic problems, such as the validation of Lyapunov functions for non-linear systems can be straightforwardly encoded as Σ_1-sentences, for which dReal has been used in practical problems [12].

The important next step of research is to solve Σ_2-sentences, which will be able to encode a wide range of general control problems. Consider Lyapunov analysis of systems as an example.

Example 5.1. The search of Lyapunov functions can be done by fixing a template of Lyapunov functions and search for parameters. Let D be the parameter space for p and X be the state space

of x. We then have the following $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula is a sufficient condition for stability in the sense of Lyapunov

$$\varphi_L : \exists p \forall x \left(V(p,x) \geq 0 \wedge V(p,0) = 0 \wedge \frac{\partial V(p,x)}{\partial x} f(x) \leq 0 \right)$$

If p is a single parameter, the problem can be solved by binary search on p. In general, if p is a vector, we need procedures for solving Σ_2 problems. ■

General Σ_2 problems can be seen as a generalization of vector optimization problems. Decision procedures for such sentences can be developed through recursive calls to the algorithms for the existential sentences, and also exploit existing optimization algorithms:

- The first one is to use optimization solvers as oracles, and solve problems in the DPLL(T) way. This direction should extend all benefits of existing optimization algorithms. Ideally, this provides a framework that strictly generalizes existing practice in control theory. The key is to formalize the numerical procedures such that delta-completeness is achieved.
- The second one is the generic procedure of calling the solver itself recursively. One would need to first leave the existentially quantified variables as free, and solve the universally quantified constraints first. The results are then used for pruning the constraints on the existentially quantified variables.

A challenge is that when we solve for the innermost universally quantified constraints, we need over-approximation on the values of the negation of the formulas. This requires algorithms for computing under approximations for the real variables. Theoretically, this is not harder than solving based on over approximation, but practical algorithms are yet to be developed. On the other hand, we can formulate a notion of mixing delta strengthening and weakening for applications.

5.2 Towards $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$

The next step is to develop algorithms for solving second-order formulas in $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$. As discussed in Section 3.2, control design problems can usually be encoded as the search for functions that satisfy certain constraints. For instance, optimal control problems search for functions that minimize a functional that represents cost. Obviously, it is computationally difficult to have a generic algorithm. There is no existing work on computing second-order formulas over the reals. Still, there are three directions to approach the problem.

One straightforward approach is to restrict the search with templates, which amounts to restricting the interpretation of the second-order variables, and reduce the problems to a series of first-order Σ_2 problems. This is in fact the commonly used strategy in control design. For instance, when designing a PID (Proportional-Integral-Derivative) controller, we would use the template

$$\mathrm{u}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

and search for the parameters $K_p, K_i, K_d \in \mathbb{R}$. Another example is in Lyapunov analysis, in which one typically search through polynomials of increasingly higher degree.

The second approach is to work with parametric equations of a function, which can be considered as a curve, and numerically compute the values of the parameters to traverse on the curve. This is indeed what we have done for formulas with ODEs, when dealing with reachability problems for

hybrid dynamical systems [10]. In this case, we compute the solution of the ODE is parameterized by the time variable, and produce a complete trace. The piecewise linear trace is a delta-approximation of the exact function. We need to formalize this approach and further investigate its usability and limits.

The third direction is to incorporate methods from infinite-dimensional optimization, such as calculus of variations and dynamic programming. The algorithms typically involve solving partial differential equations for solving such problems. Formalization of the existing algorithms would require a significant amount of work, which will be an important step towards the merging of symbolic and numerical algorithms.

5.3 Main Goals in Computational Engine

Goal 5. Develop practical algorithms for the δ-decision problem of $\exists\forall$-sentences in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$.

Goal 6. Develop a framework for solving δ-decision problems of $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$-sentences.

6 Summary

The six goals in the previous sections can be divided into theoretical and computational goals. The theoretical ones include:

1. Completely formalize first-order problems in control theory in $\mathcal{L}^1_{\mathbb{R}_\mathcal{F}}$, and characterize the complexity accordingly.
2. Develop a suitable second-order language $\mathcal{L}^2_{\mathbb{R}_\mathcal{F}}$ and prove decidability for control design problems that can be expressed accordingly.
3. Categorize theorems in control theory in suitable subsystems of second-order arithmetic, and find proof complexity for decidable theorems.

And the computational ones are:

1. Develop formal proofs of basic theorems in control theory in an interactive theorem prove.
2. Implement a practical solver for Σ_2 problems in control theory.
3. Develop a framework for solving second-order problems.

References

[1] A. A. Ahmadi. Algebraic relaxations and hardness results in polynomial optimization and Lyapunov analysis. PhD Thesis, Massachusetts Institute of Technology, 2011.

[2] A. A. Ahmadi, A. Majumdar, and R. Tedrake. Complexity of ten decision problems in continuous time dynamical systems. *CoRR*, abs/1210.7420, 2012.

[3] A. A. Ahmadi and P. A. Parrilo. Stability of polynomial differential equations: Complexity and converse lyapunov questions. *CoRR*, abs/1308.6833, 2013.

[4] V. D. Blondel and J. N. Tsitsiklis. Complexity of stability and controllability of elementary hybrid systems. *Automatica*, 35(3):479–489, 1999.

[5] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

[6] S. Cook and P. Nguyen. *Logical Foundations of Proof Complexity*. Springer, 2010.

[7] S. Gao, J. Avigad, and E. M. Clarke. Delta-complete decision procedures for satisfiability over the reals. In B. Gramlich, D. Miller, and U. Sattler, editors, *IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 286–300. Springer, 2012.

[8] S. Gao, J. Avigad, and E. M. Clarke. Delta-decidability over the reals. In *LICS*, pages 305–314, 2012.

[9] S. Gao, S. Kong, and E. M. Clarke. dreal: An smt solver for nonlinear theories over the reals. In M. P. Bonacina, editor, *CADE*, volume 7898 of *Lecture Notes in Computer Science*, pages 208–214. Springer, 2013.

[10] S. Gao, S. Kong, and E. M. Clarke. Satisfiability modulo odes. In *FMCAD*, pages 105–112. IEEE, 2013.

[11] S. Gao, S. Kong, and E. M. Clarke. Revisiting the complexity of stability of continuous and hybrid systems. *CoRR*, abs/1404.7169, 2014.

[12] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In M. Fränzle and J. Lygeros, editors, *HSCC*, pages 133–142. ACM, 2014.

[13] A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

[14] K.-I. Ko. *Complexity Theory of Real Functions*. BirkHauser, 1991.

[15] A. Platzer. Logics of dynamical systems. In *LICS*, pages 13–24, 2012.

[16] P. Prabhakar and M. Viswanathan. On the decidability of stability of hybrid systems. In C. Belta and F. Ivancic, editors, *HSCC*, pages 53–62. ACM, 2013.

[17] S. Simpson. *Subsystems of Second-Order Arithmetic*. Springer, 2005.

[18] K. Weihrauch. *Computable Analysis: An Introduction*. 2000.

[19] K. Yokoyama. Standard and non-standard analysis in second order arithmetic. Ph.D. Thesis, 2009.