Revisiting the Complexity of Stability of Continuous and Hybrid Systems

Sicun Gao   **Soonho Kong**   **Edmund M. Clarke**

July 16, 2014
CMU-CS-14-112

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We develop a general framework for obtaining upper bounds on the "practical" computational complexity of stability problems, for a wide range of nonlinear continuous and hybrid systems. To do so, we describe stability properties of dynamical systems in first-order theories over the real numbers, and reduce stability problems to the δ-decision problems of their descriptions. The framework allows us to give a precise characterization of the complexity of different notions of stability for nonlinear continuous and hybrid systems. We prove that bounded versions of the δ-stability problems are generally decidable, and give upper bounds on their complexity. The unbounded versions are generally undecidable, for which we measure their degrees of unsolvability.

1 Introduction

Stability of dynamical systems is a central topic in control theory. The computational nature of stability properties has been a topic of much recent investigation [3, 5, 6, 1, 13, 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].

Note that the existing approaches measure complexity of stability through *symbolic* manipulation of their descriptions. While doing so is suitable for establishing hardness results (for subclasses of the systems), it is at the core different from the practice in control theory, which is mostly based on *numerical* computations over real numbers. We argue that more general results for complexity of stability need to take into account of how real numbers and real functions are computed, as studied in computable analysis [14, 12, 8]. However, while complexity for real functions is best measured in the model of Type 2 Turing machines [12], stability problems are still standard decision problems that should be measured in the standard complexity classes. Moreover, it is important to distinguish the difficulty with manipulating real numbers from the intrinsic complexity of control problems. For instance, if a real number x is represented numerically (as an infinite Cauchy sequence of rationals), determining whether "$x = 0$" is already undecidable [14]. Using such hardness in measuring complexity of practical control problems would be misleading, because in practice, the problems are always solved up to some nonzero error bound. That is, $|x| < \delta$ for a sufficiently small δ is what we need in practice, rather than the theoretically undecidable equality testing. The computational nature of the problem is very different with such a relaxation.

We will show that δ-decisions over the real numbers [10, 9] provides a suitable framework for measuring the intrinsic complexity of control problems to address the issues discussed above. Within this framework, we can study the following version of stability problems. Given a dynamical system and an arbitrarily small $\delta \in \mathbb{Q}^+$, we ask for one of the following answers:

- The system is stable.

- The system is unstable under numerical perturbations bounded by δ.

We call this the δ-stability problem. With this definition, we are able to give precise upper bounds for the "practical complexity" of stability problems for a wide range of continuous and hybrid systems. We are able to prove results of the following type:

- Bounded Lyapunov δ-stability resides in the complexity class $(\Pi_3^P)^C$, where C is the complexity of continuous functions in the system. (Π_3^P denotes the complexity class in the polynomial hierarchy).

- Bounded asymptotic δ-stability resides in the complexity class $(\Sigma_4^P)^C$.

- Unbounded Lyapunov δ-stability is undecidable, whose degree of undecidability is Π_1^0. Unbounded asymptotic δ-stability is undecidable, whose degree of undecidability is in Σ_2^0.

1

- Lyapunov methods reduce problems into lower complexity classes such as $(\Sigma_2)^{\mathsf{C}}$.

We believe these results are the first general characterization of the complexity of stability. Moreover, the importance of the results is not just theoretical. The past decade has seen great advancement in decision procedures (SAT, QBF, and SMT solvers) that can handle many large instances of NP-hard problems. The complexity analysis shows the possibility of developing generic algorithmic approaches to control problems of nonlinear and hybrid systems.

In all, the main contributions of the paper are as follows:

- We define a framework for measuring the "practical complexity" of stability problems for a wide range of nonlinear continuous and hybrid systems. To do so, we describe stability properties of systems as first-order formulas over the real numbers, and reduce stability problems to the δ-decision problems of these formulas.

- The framework allows us to obtain a precise characterization of the complexity of different notions of stability that has not been discovered previously. We prove that bounded version of the stability problems are generally decidable, and give upper bounds on their complexity. The unbounded versions are generally undecidable, for which we measure their degrees of unsolvability.

The paper is organized as follows. In Section II, we review definitions of complexity classes and some main results from computable analysis. In Section III, we review the theory of δ-decisions over the reals and introduce the logic language that can encode a wide range of dynamical systems and properties. In Section IV, we study the complexity of stability of continuous systems. In Section V, we study the same questions for hybrid systems. We conclude in Section VI and suggest future directions.

2 Preliminaries

2.1 Oracle Machines, Polynomial and Arithmetic Hierarchies

We review the basic definitions for complexity hierarchies.

A *(set-) oracle Turing machine M* extends an ordinary Turing machine with a special read/write tape called the *oracle tape*, and three special states q_{query}, q_{yes}, q_{no}. To execute M, we specify an oracle language $O \subseteq \{0,1\}^*$ in addition to the input x. Whenever M enters the state q_{query}, it queries the oracle O with the string s on the oracle tape. If $s \in O$, then M enters the state q_{yes}, otherwise it enters q_{no}. Regardless of the choice of O, a membership query to O counts only as a single computation step. A *function-oracle Turing machine* is defined similarly except that when the machine enters the query state the oracle (given by a function $f : \{0,1\}^* \to \{0,1\}^*$) will erase the string s on the query tape and write down $f(s)$. Note that such a machine must take $|f(s)|$ steps to read the output from the query tape. We write $M^O(x)$ (resp. $M^f(x)$) to denote the output of M on input x with oracle O (resp. f).

The polynomial hierarchy PH is a hierarchy of complexity classes that is defined through oracle computation. The base case are the well-known complexity classes P and NP. The classes in the hierarchy are recursively defined in the standard way:

$$\Sigma_0^P = \Pi_0^P = P, \Sigma_{k+1}^P(A) = NP^{\Sigma_k^P(A)}, \Pi_{k+1}^P(A) = coNP^{\Sigma_k^P(A)}$$

It is well-known that $PH \subseteq PSPACE$. If $P \neq NP$, then each class in the hierarchy contains harder problems than the previous ones. For undecidable problems, there exists an analogous arithmetic hierarchy. The base case is Σ_1^0, which is the class of the halting problem. The other classes in the arithmetic hierarchy $\Pi_0^1, \Sigma_2^0, ...$ alternate in a similar way. The detailed definitions of polynomial and arithmetic hierarchy can be found in standard textbooks on recursion theory and computational complexity such as [4].

2.2 Type 2 Computable Functions

Given a finite alphabet Σ, let Σ^* denote the set of finite strings and Σ^ω the set of infinite strings generated by Σ. For any $s_1, s_2 \in \Sigma^*$, $\langle s_1, s_2 \rangle$ denotes their concatenation. An integer $i \in \mathbb{Z}$ used as a string over $\{0, 1\}$ has its conventional binary representation. The set of *dyadic rational numbers* is $\mathbb{D} = \{m/2^n : m \in \mathbb{Z}, n \in \mathbb{N}\}$.

Computations over Infinite Strings   Standard computability theory studies operations over finite strings and does not consider real-valued functions. Real numbers can be encoded as infinite strings, and a theory of computability of real functions can be developed with oracle machines that perform operations using function-oracles encoding real numbers. This is the approach developed in Computable Analysis, a.k.a., Type 2 Computability. We will briefly review definitions and results of importance to us. Details can be found in the standard references [14, 12, 7].

Definition 2.1 (Names). *A name of $a \in \mathbb{R}$ is defined as a function $\gamma_a : \mathbb{N} \to \mathbb{D}$ satisfying*

$$\forall i \in \mathbb{N}, |\gamma_a(i) - a| < 2^{-i}.$$

For $\vec{a} \in \mathbb{R}^n$, $\gamma_{\vec{a}}(i) = \langle \gamma_{a_1}(i), ..., \gamma_{a_n}(i) \rangle$.

Thus the name of a real number is a sequence of dyadic rational numbers converging to it. For $\vec{a} \in \mathbb{R}^n$, we write $\Gamma(\vec{a}) = \{\gamma : \gamma \text{ is a name of } \vec{a}\}$. Noting that names are discrete functions, we can define

Definition 2.2 (Computable Reals). *A real number $a \in \mathbb{R}$ is computable if it has a name γ_a that is a computable function.*

A real function f is computable if there is a function-oracle Turing machine that can take any argument x of f as a function oracle, and output the value of $f(x)$ up to an arbitrary precision.

3

Definition 2.3 (Computable Functions). *We say a real function $f : \subseteq \mathbb{R}^n \to \mathbb{R}$ is Type 2 computable if there exists a function-oracle Turing machine $\mathcal{M}_f$, outputting dyadic rationals, such that for any $\vec{x} \in \text{dom}(f)$, any name $\gamma_{\vec{x}}$ for $\vec{x}$, and any $i \in \mathbb{N}$, the output of $M_f^{\gamma_{\vec{x}}(i)}$ satisfies that*

$$|M_f^{\gamma_{\vec{x}}}(i) - f(\vec{x})| < 2^{-i},$$

which means that it approximates $f(\vec{x})$ up to 2^{-i}.

In the definition, i specifies the desired error bound on the output of M_f with respect to $f(\vec{x})$. For any $\vec{x} \in \text{dom}(f)$, M_f has access to an oracle encoding the name $\gamma_{\vec{x}}$ of $\vec{x}$, and output a 2^{-i}-approximation of $f(\vec{x})$. In other words, the sequence

$$M_f^{\gamma_{\vec{x}}}(1), M_f^{\gamma_{\vec{x}}}(2), \dots$$

is a name of $f(\vec{x})$. Intuitively, f is computable if an arbitrarily good approximation of $f(\vec{x})$ can be obtained using any good enough approximation to any $\vec{x} \in \text{dom}(f)$.

Proposition 2.4 ([14]). *The following real functions are computable: addition, multiplication, absolute value,* min*,* max*,* exp*,* sin *and solutions of Lipschitz-continuous ordinary differential equations. Compositions of computable functions are computable.*

A key property of the above notion of computability is that computable functions over reals must be continuous. In fact, over any compact set $D \subseteq \mathbb{R}^n$, computable functions are uniform continuous with a *computable modulus of continuity*. Intuitively, if a function has a computable uniform modulus of continuity, then fixing any desired error bound 2^{-i} on the output, we can compute a *global* precision $2^{-m_f(i)}$ on the inputs from D such that using any $2^{-m_f(i)}$-approximation of any $\vec{x} \in D$, $f(\vec{x})$ can be computed within the error bound.

Complexity of real functions is usually defined over compact domains. Without loss of generality, we consider functions over $[0,1]$. 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 $\vec{x} \in \text{dom}(f)$. The formal definition is as follows:

Definition 2.5 ([12]). *A real function $f : [0,1]^n \to \mathbb{R}$ is in $\mathsf{P}_{\mathsf{C}[0,1]}$ (resp. $\mathsf{PSPACE}_{\mathsf{C}[0,1]}$) iff there exists a representation (m_f, θ_f) of f such that*

- *m_f is a polynomial function, and*

- *for any $d \in (\mathbb{D} \cap [0,1])^n$, $e \in \mathbb{D}$, and $i \in \mathbb{N}$, $\theta_f(d,i)$ is computable in time (resp. space) $O((len(d)+i)^k)$ for some constant k.*

Proposition 2.6. *The following real functions all reside in Type 2 complexity class $\mathsf{P}_{\mathsf{C}[0,1]}$: absolute value, polynomials, binary* max *and* min*,* exp*,* sin*, and their bounded compositions.*

It is shown that solutions of Lipschitz-continuous differential equations are computable in $\mathsf{PSPACE}_{\mathsf{C}[0,1]}$. In fact, it is shown to be PSPACE-complete in the following sense.

Proposition 2.7 ([11]). *Let $g : [0,1] \times \mathbb{R} \to \mathbb{R}$ be polynomial-time computable and consider the initial value problem $\frac{df(t)}{dt} = g(t, f(t))$ for $f(0) = 0$ and $t \in [0,1]$. Then computing the solution $f : [0,1] \to \mathbb{R}$ is in PSPACE. Moreover, there exists g such that computing f is PSPACE-complete.*

3 $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-Formulas and δ-Decidability

3.1 $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-Formulas

We will use a logical language over the real numbers that allows arbitrary *computable real functions* [14]. We write $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$ to represent this language. Intuitively, a real function is computable if it can be numerically simulated up to an arbitrary precision. For the purpose of this paper, it suffices to know that almost all the functions that are needed in describing hybrid systems are Type 2 computable, such as polynomials, exponentiation, logarithm, trigonometric functions, and solution functions of Lipschitz-continuous ordinary differential equations.

More formally, $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}} = \langle \mathscr{F}, > \rangle$ represents the first-order signature over the reals with the set $\mathscr{F}$ of computable real functions, which contains all the functions mentioned above. Note that constants are included as 0-ary functions. $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formulas are evaluated in the standard way over the structure $\mathbb{R}_{\mathscr{F}} = \langle \mathbb{R}, \mathscr{F}^{\mathbb{R}}, >^{\mathbb{R}} \rangle$. It is not hard to see that we can put any $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formula in a normal form, such that its atomic formulas are of the form $t(x_1,...,x_n) > 0$ or $t(x_1,...,x_n) \geq 0$, with $t(x_1,...,x_n)$ composed of functions in $\mathscr{F}$. To avoid extra preprocessing of formulas, we can explicitly define $\mathscr{L}_{\mathscr{F}}$-formulas as follows.

Definition 3.1 ($\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-Formulas). *Let $\mathscr{F}$ be a collection of computable real functions. We define:*

$$t := x \mid f(t), \text{ where } f \in \mathscr{F} \text{ (constants are 0-ary functions)}$$
$$\varphi := t > 0 \mid t \geq 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x_i \varphi \mid \forall x_i \varphi.$$

In this setting $\neg \varphi$ is regarded as an inductively defined operation which replaces atomic formulas $t > 0$ with $-t \geq 0$, atomic formulas $t \geq 0$ with $-t > 0$, switches $\wedge$ and $\vee$, and switches $\forall$ and $\exists$.

Definition 3.2 (Bounded $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-Sentences). *We define the bounded quantifiers $\exists^{[u,v]}$ and $\forall^{[u,v]}$ as*

$$\exists^{[u,v]} x. \varphi \quad =_{df} \quad \exists x.(u \leq x \wedge x \leq v \wedge \varphi)$$
$$\forall^{[u,v]} x. \varphi \quad =_{df} \quad \forall x.((u \leq x \wedge x \leq v) \rightarrow \varphi)$$

where u and v denote $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$ terms, whose variables only contain free variables in φ excluding x. A bounded $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-sentence is $Q_1^{[u_1,v_1]} x_1 \cdots Q_n^{[u_n,v_n]} x_n \ \psi(x_1,...,x_n)$, where $Q_i^{[u_i,v_i]}$ are bounded quantifiers, and $\psi(x_1,...,x_n)$ is quantifier-free.

3.2 δ-Perturbations and δ-Decidability

Definition 3.3 (δ-Variants). *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$, and φ an $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formula*

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

where $i \in \{1,...k\}$ and $j \in \{k+1,...,m\}$. The δ-weakening φ^δ of φ is defined as the result of replacing each atom $t_i > 0$ by $t_i > -\delta$ and $t_j \geq 0$ by $t_j \geq -\delta$:

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

5

It is easy to see that the perturbed formula is implied by the original formula.

Proposition 3.4 ((see [10])). *For any φ, we have $\varphi \rightarrow \varphi^\delta$.*

In [10, 9], we have proved that the following δ-decision problem is decidable, which is the basis of our framework.

Theorem 3.5 (δ-Decidability [10]). *Let $\delta \in \mathbb{Q}^+$ be arbitrary. There is an algorithm which, given any bounded $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-sentence φ, correctly returns one of the following two answers:*

- δ-True*: φ^δ is true.*

- False*: φ is false.*

When the two cases overlap, either answer is correct.

Theorem 3.6 (Complexity [10]). *Let S be a class of $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-sentences, such that for any φ in S, the terms in φ are in Type 2 complexity class C. Then, for any $\delta \in \mathbb{Q}^+$, the δ-decision problem for bounded Σ_n-sentences in S is in $(\Sigma_n^{\mathsf{P}})^{\mathsf{C}}$.*

4 Stability of Continuous Systems

4.1 $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-Representations

Consider an n-dimensional autonomous ODE system

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t)) \tag{1}$$

where f is Lipschitz-continuous and $x(0) \in \mathbb{R}^n$. We define the $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-representation of the system to be a logical formula that describes the all points on the trajectory of the dynamical system.

Definition 4.1. *We say the system (1) is $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-represented by an $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-formula $\mathsf{flow}(x_0, x_t, t)$, if for any $x(t) \in \mathbb{R}$, $x(t)$ is on the trajectory of the system iff the $\mathsf{flow}(x_0, x_t, t)$ is true.*

From Picard-Lindelöf iteration, we know that the $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-representation for continuous systems has an explicit form:

Proposition 4.2. *The dynamical system in (1) has a trajectory that passes through $a \in \mathbb{R}$ iff the following $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-formula is true:*

$$\mathsf{flow}(x_0, x_t, t) =_{df} \left(x_t = \int_0^t f(x(s))\mathrm{d}s + x_0 \right)$$

Proposition 4.3. *A continuous system has a $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-representation, when f is a Type 2 computable function.*

Since f can be any numerically computable function, this definition covers almost all dynamical systems of interest. We can now speak of the dynamical system (1) and its $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation flow$(x_0, x_t, t)$ interchangeably.

The δ-perturbation on a system is defined through δ-perturbations on its $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation.

Definition 4.4. *The δ-perturbation of a system that is $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-represented by* flow$(x_0, x_t, t)$ *is the system represented by* flow$^\delta(x_0, x_t, t)$.

To be clear, the flow formula has an explicit definition:

Proposition 4.5. *The δ-perturbation of the system (1) is represented by*

$$\text{flow}^\delta =_{df} |x_t - (\int_0^t f(x(s))ds + x_0)| < \delta.$$

Note that the δ-perturbed system is always an overapproximation of the original system:

Proposition 4.6. *We have* $[\![\text{flow}]\!] \subseteq [\![\text{flow}^\delta]\!]$.

4.2 Complexity of Lyapunov Stability

We first study stability in the sense of Lyapunov, which we can write stable i.s.L. 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. The $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of stability in the sense of Lyapunov is naturally the following formula.

Definition 4.7 (L_stable)**.** *We encode conditions for Lyapunov stability with the formula* L_stable *as follows.*

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

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

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

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

It is not hard to see that the formula encodes the definition of stability in the sense of Lyapunov.

Proposition 4.8. *The origin is a stable equilibrium point iff* L_stable *is true.*

We can now define the δ-stability problem using the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation.

Definition 4.9 (δ-Stability i.s.L.)**.** *The δ-stability problem i.s.L. asks for one of the following answers:*

7

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

Proof. The $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formula L_stable is a σ_3 formula. By Definition 5.11, the δ-stability problem is equivalent to the δ-decision problem of the formula L_stable. Following Theorem 3.6, we have that the complexity of the δ-decision problem for the bounded form of L_stable is in $(\Pi_3^P)^C$. Consequently, the bounded δ-stability problem i.s.L. resides in $(\Pi_3^P)^C$. □

Following the complexity for Lipschitz-continuous ODEs, we have an upper bound for the complexity of a wide range of systems.

Corollary 4.11. *Suppose that in the system (1), f is a Type 2 polynomial-time computable function. Then the bounded δ-stability problem i.s.L. is in* PSPACE.

Proof. The $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation flow can be evaluated in PSAPCE. Since $(\Pi_3^P)^{\text{PSPACE}} \subseteq \text{PSPACE}$, we know that the problem resides in PSPACE. □

We have mentioned that most of common functions and their compositions are polynomial-time computable: polynomials, trigonometric functions, exponential functions, etc. Consequently, for most nonlinear continuous systems of practical interest, the stability problem is in PSPACE.

The unbounded case involves testing the bounded formula for longer and longer time durations. Thus, it is still undecidable. We can obtain the degree of undecidability of the unbounded case from the logical encoding.

Theorem 4.12. *The unbounded Lyapunov δ-stability problem is in Π_1^0.*

Proof. We compute δ-decisions of the bounded form of the formula L_stable for increasingly larger time bound T. If for any T the formula is δ-false, then the system is δ-unstable. On the other hand, we will not be able to confirm that the system is stable as T approaches infinity. Thus, the problem is in Π_1^0 of the arithmetic hierarchy. □

4.3 Complexity of Asymptotic Stability

Following standard terminology, we say a system is asymptotically stable if it is Lyapunov stable, and there exists some bound on the perturbation in the initial state such that the system will converge to the origin eventually. We now study the complexity of this problem.

First, since asymptotic stability involves properties of the system at the limit, we need to be express that as an $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-formula, as follows.

Definition 4.13. *We define the following formula for* $\lim_{x \to \infty}(f(x), c)$

$$\lim_{x \to \infty}(f(x), c) =_{df} \forall^{[0,\infty)}\varepsilon \exists^{[0,\infty)}x \forall^{[x,\infty)}x' \left(|f(x) - c| < \varepsilon\right).$$

We can use the conventional notation $\lim_{x \to \infty} f(x) = c$. *Also, for convergence at a point* $a \in \mathbb{R}^+$, *we define*

$$\lim_{x \to a}(f(x), c) =_{df} \forall^{[0,\infty)}\varepsilon \exists^{[0,\infty)}\delta \forall^{[a-\delta,a+\delta]}x \left(|f(x) - c| < \varepsilon\right).$$

Note that here the quantification on ε *and* δ *can be easily bounded, since we do not need to consider* ε *and* δ *that are very large. Although further parameterization on the bounds are needed, for notational simplicity we simply treat this formula as a bounded* $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-*formula.*

Now, asymptotic stability is defined as:

Definition 4.14 (A_stable)**.** *We define* A_stable *to be the following* $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-*formula*

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

$$\wedge \exists^{[0,\infty)}\delta' \forall^{[0,\infty)}t \forall x_0 \forall x_t \left(\left(||x_0|| < \delta' \wedge x_t = \int_0^t f(s)ds + x_0\right) \to \lim_{t \to \infty} ||x_t|| = 0\right).$$

The bounded form of A_stable *is defined as:*

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

$$\wedge \exists^{[0,d]}\delta' \forall^{[0,T']}t \forall^X x_0 \forall^X x_t \left(\left(||x_0|| < \delta' \wedge x_t = \int_0^t f(s)ds + x_0\right) \to \lim_{t \to T'} ||x_t|| = 0\right)$$

where $e, T, T', d \in \mathbb{R}^+$ *and* X *is a compact set.*

Proposition 4.15. *The origin is asymptotically stable for a system iff the formula* A_stable *is true.*

We can now define the δ-stability problem using the $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-representation.

Definition 4.16 (Asymptotic δ-Stability)**.** *The* δ-*stability problem i.s.A. asks for one of the following answers:*

- stable*: The system is stable i.s.A. (*A_stable *is true).*

- δ-unstable*: Some* δ-*perturbation of* A_stable *is false.*

9

We defined the bounded δ-*stability problem by replacing* A_stable *with the bounded form of* A_stable *in the definition.*

We can now obtain complexity results for the problem.

Theorem 4.17. *Suppose all terms in the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of a system are in Type 2 complexity class* C. *Then bounded asymptotic δ-stability is in* $(\Sigma_4^P)^C$.

Proof. The complexity of the formula is higher than the one encoding Lyapunov stability, because of the quantification structure in the encoding of the limit. After rearranging the formula, we have

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

$$\wedge\,\exists^{[0,d]}\delta'\forall^{[0,T']}t\forall^X x_0\forall^X x_t\forall^{[0,e']}\varepsilon'\exists^{[0,d']}\delta''\forall^{[-\delta'',+\delta'']}t\left((||x_0||<\delta'\wedge x_t=\int_0^t f(s)ds+x_0)\to||x_t||<\varepsilon'\right)$$

This is a Σ_4-formula. Following Theorem 3.6 we know that the problem resides in $(\Sigma_4^P)^C$. □

The degree of undecidability for the unbounded version is, however, different from Lyapunov stability. This is because we need to find the bound of perturbation that ensures the convergence to the origin.

Corollary 4.18. *Unbounded asymptotic δ-stability is in* Σ_2^0.

Proof. In the formula A_stable, we need to incrementally search for a value for δ'. Each of the value corresponds to an unbounded search for the time bound, which is similar to the case of Lyapunov complexity. Thus, we need to solve unbounded $\exists\forall$ quantification, which means the unbounded problem is in Σ_2^0 of the arithmetic hierarchy. □

It is probably interesting to note that the problem $P\neq NP$ has the same degree of undecidability.

There is also the notion of "asymptotic stability in the large," which ensures that for any perturbation on $x(0)$, the system will stabilize. The quantification turns out to be slightly different:

Proposition 4.19 (Asymptotic Stability in the Large). *The origin is an asymptotically stable equilibrium point iff the following $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formula is true*

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

$$\wedge\forall^{[0,\infty)}\delta'\forall^{[0,\infty)}t\forall x_0\forall x_t\left((||x_0||<\delta'\wedge x_t=\int_0^t f(s)ds+x_0)\to\lim_{t\to\infty}||x_t||=0\right).$$

Computationally, this is in fact a simpler task than asymptotic stability. We state the following result without duplicating the proofs.

Theorem 4.20. *Suppose all terms in the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of a system are in Type 2 complexity class* C. *Then bounded asymptotic δ-stability in the large is in* $(\Pi_3^P)^C$. *The unbounded case resides in* Π_1^0.

10

4.4 Complexity of Lyapunov Methods

We show that Lyapunov methods reduce the complexity of stability problems. We only discuss the first-order encodings of the problems, in which a Lyapunov function is considered with a template function with unspecified parameters.

Proposition 4.21. *Consider the dynamical system (1). Let $V(p,x)$ be a function, parameterized by p, 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 $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formula is a sufficient condition for stability in the sense of Lyapunov*

$$\exists p^D \forall^X x \left(V(p,0) = 0 \wedge (x \neq 0 \rightarrow V(p,x) > 0) \wedge \frac{\partial V(p,x)}{\partial x} f(x) \leq 0 \right)$$

- *The following is a sufficient condition for asymptotic stability:*

$$\exists p^D \forall^X x \left(V(p,0) = 0 \wedge (x \neq 0 \rightarrow V(p,x) > 0) \wedge \left(x = 0 \rightarrow \frac{\partial V(p,x)}{\partial x} f(x) = 0 \right) \wedge \left(x \neq 0 \rightarrow \frac{\partial V(p,x)}{\partial x} f(x) \right.$$

Definition 4.22 (δ-Complete Lyapunov Test). *Let $V(p,x)$ be a proposed template for Lyapunov function. The δ-complete Lyapunov test asks for one of the following answers:*

- Success*: There exists an assignment to p such that the Lyapunov function witness stability of the system.*

- δ-Fail*: The Lyapunov conditions fail under δ-perturbations for all possible parameterizations of $V(p,x)$ in the parameter space D.*

Theorem 4.23. *Suppose all terms in the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of the Lyapunov conditions are in Type 2 complexity class* C*. The complexity of bounded δ-complete Lyapunov methods is in $(\Sigma_2^P)^C$.*

It is clear that for the fully unbounded case (where both D and X are unbounded), undecidability comes from the search in larger and larger parameter and state space.

Corollary 4.24. *The unbounded δ-complete Lyapunov test for an unbounded system is in Σ_2^0.*

5 Stability of Hybrid Systems

An important benefit of using logic formulas for describing systems is that discrete changes can be naturally represented. Although the discrete components significantly complicates the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representations of the problems, they do not change the quantification structure of the encodings. Thus, we will see that the complexity upper bound of the continuous systems mostly carry over to the case of hybrid systems as well. On the other hand, it is indeed easier to show hardness results (lower-bound) using logical operations, and in this sense hybrid systems are intrinsically more complicated than continuous systems.

5.1 $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-Representations of Hybrid Systems

We first show that $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-formulas can concisely represent hybrid automata.

Definition 5.1. *A hybrid automaton in $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-representation is a tuple*

$$H = \langle X, Q, \{\mathsf{flow}_q(\vec{x}, \vec{y}, t) : q \in Q\}, \{\mathsf{inv}_q(\vec{x}) : q \in Q\}, \{\mathsf{jump}_{q \to q'}(\vec{x}, \vec{y}) : q, q' \in Q\}, \{\mathsf{init}_q(\vec{x}) : q \in Q\}\rangle$$

where $X \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$, $Q = \{q_1, ..., q_m\}$ is a finite set of modes, and the other components are finite sets of quantifier-free $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-formulas.

Notation 5.2. For any hybrid system H, we write $X(H)$, $\mathsf{flow}(H)$, etc. to denote its corresponding components.

Almost all hybrid systems studied in the existing literature can be defined by restricting the set of functions $\mathscr{F}$ in the signature. For instance,

Example 5.3 (Linear and Polynomial Hybrid Automata). Let $\mathscr{F}^{\mathrm{lin}} = \{+\} \cup \mathbb{Q}$ and $\mathscr{F}^{\mathrm{poly}} = \{\times\} \cup \mathscr{F}^{\mathrm{lin}}$. Rational numbers are considered as 0-ary functions. In existing literature, H is a *linear hybrid automaton* if it has an $\mathscr{L}_{\mathbb{R}_{\mathscr{F}^{\mathrm{lin}}}}$-representation, and a *polynomial hybrid automaton* if it has an $\mathscr{L}_{\mathbb{R}_{\mathscr{F}^{\mathrm{poly}}}}$-representation.

Example 5.4 (Nonlinear Bouncing Ball). The bouncing ball is a standard hybrid system model. Its nonlinear version (with air drag) can be $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-represented as follows:

- $X = \mathbb{R}^2$ and $Q = \{q_u, q_d\}$. We use q_u to represent bounce-back mode and q_d the falling mode.

- $\mathsf{flow} = \{\mathsf{flow}_{q_u}(x_0, v_0, x_t, v_t, t), \mathsf{flow}_{q_d}(x_0, v_0, x_t, v_t, t)\}$. We use x to denote the height of the ball and v its velocity. Instead of using time derivatives, we can directly write the flows as integrals over time, using $\mathscr{L}_{\mathbb{R}_\mathscr{F}}$-formulas:

  - $\mathsf{flow}_{q_u}(x_0, v_0, x_t, v_t, t)$ defines the dynamics in the bounce-back phase:

    $$\left(x_t = x_0 + \int_0^t v(s)ds\right) \wedge \left(v_t = v_0 + \int_0^t g(1 - \beta v(s)^2)ds\right)$$

  - $\mathsf{flow}_{q_d}(x_0, v_0, x_t, v_t, t)$ defines the dynamics in the falling phase:

    $$\left(x_t = x_0 + \int_0^t v(s)ds\right) \wedge \left(v_t = v_0 + \int_0^t g(1 + \beta v(s)^2)ds\right)$$

  where β is a constant. Again, note that the integration terms define Type 2 computable functions.

- $\mathsf{jump} = \{\mathsf{jump}_{q_u \to q_d}(x, v, x', v'), \mathsf{jump}_{q_d \to q_u}(x, v, x', v')\}$ where

  - $\mathsf{jump}_{q_u \to q_d}(x, v, x', v')$ is $(v = 0 \wedge x' = x \wedge v' = v)$.

- $\text{jump}_{q_d \to q_u}(x,v,x',v')$ is $(x = 0 \wedge v' = \alpha v \wedge x' = x)$, for some constant α.

- $\text{init}_{q_d} : (x = 10 \wedge v = 0)$ and $\text{init}_{q_u} : \bot$.

- $\text{inv}_{q_d} : (x >= 0 \wedge v >= 0)$ and $\text{inv}_{q_u} : (x >= 0 \wedge v <= 0)$.

Trajectories of hybrid systems combine continuous flows and discrete jumps. This motivates the use of a hybrid time domain, with which we can keep track of both the discrete changes and the duration of each continuous flow. A hybrid time domain is a sequence of closed intervals on the real line, and a hybrid trajectory is a mapping from the time domain to the Euclidean space.

We now define δ-perturbations on hybrid automata directly through perturbations on the logic formulas in their $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representations. For any set S of $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formulas, we write S^δ to denote the set containing the δ-perturbations of all elements of S.

Definition 5.5 (δ-Weakening of Hybrid Automata)**.** *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$ be arbitrary. Suppose*

$$H = \langle X, Q, \text{flow}, \text{jump}, \text{inv}, \text{init} \rangle$$

is an $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of hybrid system H. The δ-weakening of H is

$$H^\delta = \langle X, Q, \text{flow}^\delta, \text{jump}^\delta, \text{inv}^\delta, \text{init}^\delta \rangle$$

which is obtained by weakening all formulas in the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representations of H.

Example 5.6. The δ-weakening of the bouncing ball automaton is obtained by weakening the formulas in its description. For instance, $\text{flow}_{q_u}^\delta(x_0, v_0, x_t, v_t, t)$ is

$$|x_t - (x_0 + \int_0^t v(s)ds)| \le \delta \wedge |v_t - (v_0 + \int_0^t g(1 - \beta v(s)^2)ds))| \le \delta$$

and $\text{jump}_{q_d \to q_u}^\delta(x,v,x',v')$ is

$$|x| \le \delta \wedge |v' - \alpha v| \le \delta \wedge |x' - x| \le \delta.$$

It is important to note that the notion of δ-perturbations is a purely syntactic one, defined on the description of hybrid systems. Following Proposition 3.4, it can be easily seen that the syntactic perturbations correspond to semantic over-approximation of H in the trajectory space.

5.2 Complexity of Stability

We now obtain complexity results for stability of hybrid systems. The main difference from the continuous systems is that the set of reachable states of a hybrid system requires a more complex encoding. However, we will see that they do not change the upper bound of the complexity, since the quantification structure does not change.

First, we need to define a set of auxiliary formulas that will be important for ensuring that a particular mode is picked at a certain step.

Definition 5.7. *Let* $Q = \{q_1, ..., q_m\}$ *be a set of modes. For any* $q \in Q$, *and* $i \in \mathbb{N}$, *use* b_q^i *to represent a Boolean variable. We now define*

$$\mathsf{enforce}_Q(q, i) = b_q^i \wedge \bigwedge_{p \in Q \setminus \{q\}} \neg b_p^i$$

$$\mathsf{enforce}_Q(q, q', i) = b_q^i \wedge \neg b_{q'}^{i+1} \wedge \bigwedge_{p \in Q \setminus \{q\}} \neg b_p^i \wedge \bigwedge_{p' \in Q \setminus \{q'\}} \neg b_{p'}^{i+1}$$

We omit the subscript Q *when the context is clear.*

Definition 5.8 (k-Step Reachable Set)**.** *Suppose H is invariant-free, and U a subset of its state space represented by* unsafe*. The $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-formula* $\mathsf{Reach}_{H,U}(k, M)$ *is defined as:*

$$\bigvee_{q \in Q} \left(\mathsf{init}_q(\vec{x}_0) \wedge \mathsf{flow}_q(\vec{x}_0, \vec{x}_0^t, t_0) \wedge \mathsf{enforce}(q, 0) \wedge \forall^{[0, t_0]} t \forall^X \vec{x} \left(\mathsf{flow}_q(\vec{x}_0, \vec{x}, t) \rightarrow \mathsf{inv}_q(\vec{x}) \right) \right)$$

$$\wedge \bigwedge_{i=0}^{k-1} \left(\bigvee_{q, q' \in Q} \left(\mathsf{jump}_{q \rightarrow q'}(\vec{x}_i^t, \vec{x}_{i+1}) \wedge \mathsf{flow}_{q'}(\vec{x}_{i+1}, \vec{x}_{i+1}^t, t_{i+1}) \wedge \forall^{[0, t_{i+1}]} t \forall^X \vec{x} \left(\mathsf{flow}_{q'}(\vec{x}_{i+1}, \vec{x}, t) \rightarrow \mathsf{inv}_{q'}(\vec{x}) \right) \right. \right.$$

$$\left. \left. \wedge \mathsf{enforce}(q, q', i) \wedge \mathsf{enforce}(q', i+1) \right) \right)$$

Proposition 5.9 (Hybrid Lyapunov Stability)**.** *The origin is a stable equilibrium point if*

$$\forall^{[0, \infty)} \varepsilon \exists^{[0, \varepsilon]} \delta \forall^{[0, \infty)} t \forall x_0 \forall x_t (\|x_0\| < \delta \wedge \mathsf{Reach}(x_0, x_t, t)) \rightarrow \|x_t\| < \varepsilon.$$

Proposition 5.10 (Asymptotic Stability)**.** *The origin is an asymptotically stable equilibrium point if*

$$\forall^{[0, \infty)} \varepsilon \exists^{[0, \varepsilon]} \delta \forall^{[0, \infty)} t \forall x_0 \forall x_t \left((\|x_0\| < \delta \wedge \mathsf{Reach}(x_0, x_t, t)) \rightarrow \|x_t\| < \varepsilon \right)$$

$$\wedge \exists^{[0, \infty)} \delta' \forall^{[0, \infty)} t \forall x_0 \forall x_t \left((\|x_0\| < \delta' \wedge \mathsf{Reach}(x_0, x_t, t)) \rightarrow \lim_{t \to \infty} \|x_t\| = 0 \right).$$

The definition is δ-stability is the same as in the continuous case.

Definition 5.11 (δ-Stability)**.** *The (Lyapunov or asymptotic) δ-stability problem of hybrid systems asks for one of the following answers:*

- stable*: The system is stable.*

- δ-unstable*: Some δ-perturbation of the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of stability is false.*

Theorem 5.12. *Suppose all terms in the $\mathscr{L}_{\mathbb{R}_{\mathscr{F}}}$-representation of stability are in Type 2 complexity class* C*. We have*

- *The bounded Lyapunov δ-stability problem of hybrid systems is in* $(\Pi_3^P)^C$*. The asymptotic δ-stability of hybrid systems is in complexity class* $(\Sigma_4^P)^C$*.*

- *The unbounded δ-stability problem of hybrid systems is in Π_1^0 and asymptotic δ-stability is in Σ_2^0.*

From these results, it may seem that hybrid systems are not harder than continuous systems, in terms of the upper bounds on complexity. However, the discrete components of hybrid systems make it much easier to reach a high lower bound on the complexity. For instance, it is easy to show that the complexity results are tight in the following sense:

Theorem 5.13. *Suppose the terms in describing the stability formulas are polynomial-time Type 2 computable. The bounded Lyapunov and asymptotic δ-stability of hybrid systems are both (Π_3^P)-complete.*

The reason is that logic formulas can be easily encoded as jumping conditions of hybrid systems. It is then straightforward to reduce complete problems in the complexity class to stability problems of hybrid systems. We omit the full proof here.

6 Conclusion and Future Work

We defined a framework for measuring the "practical complexity" of stability problems for a wide range of nonlinear continuous and hybrid systems. To do so, we describe stability properties of systems as first-order formulas over the real numbers, and reduce stability problems to the δ-decision problems of these formulas. The framework allows us to obtain a precise characterization of the complexity of different notions of stability that has not been discovered previously. We prove that bounded version of the stability problems are generally decidable, and give precise measure of the upper bound of their complexity. The unbounded versions are generally undecidable, for which we give precise measures of their degrees of undecidability.

We believe the results serve as a basis for developing computational methods towards nonlinear and hybrid control techniques. An immediate next step is to use these methods to study other problems such as controllability and observability of nonlinear systems. On the other hand, the logical descriptions of the problems can directly guide the development of practical decision procedures for the 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] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. 2009.

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

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

[7] V. Brattka, P. Hertling, and K. Weihrauch. A tutorial on computable analysis. In S. B. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms*, pages 425–491. Springer New York, 2008.

[8] P. J. Collins. Computable Analysis With Applications To Dynamic Systems. CWI Technical Report MAC-1002, CWI, May 2010. This research was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) Vidi grant 639.032.408.

[9] 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.

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

[11] A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. In *IEEE Conference on Computational Complexity*, pages 149–160. IEEE Computer Society, 2009.

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

[13] 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.

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