Computability and Stability for Hybrid Algorithms

Nachum Dershowitz[1] and Zvi Retchkiman Königsberg[2]

[1] School of Computer Science, Tel Aviv University
Ramat Aviv, Israel
nachum.dershowitz@cs.tau.ac.il
[2] Instituto Politécnico Nacional, CIC
Ciudad de Mexico, Mexico
mzvi@cic.ipn.mx

Abstract. Church's Thesis for discrete algorithms motivates an analogous thesis for dealing with analog algorithms. Specifically, the notions of analog algorithm and dynamical system are postulated to be equivalent. Stability for hybrid algorithms is addressed by considering Lyapunov energy functions for analog algorithms with continuous and discontinuous states.

Key words: Analog algorithms, Dynamical systems, Hybrid systems, Church's Thesis, Stability, Lyapunov functions.

1 Introduction

Gurevich [3] has shown that any algorithm that satisfies three intuitive postulates can be step-by-step emulated by an abstract state machine (ASM). Adding a postulate of effectivity, Dershowitz and Gurevich [2] proceeded to prove that all notions of effective algorithms for discrete-time models of computation (e.g. Turing machines, Minsky counter machines, Post machines, random access machines) are covered by their formalization. Bournez, Dershowitz and Néron [1] then extended that axiomatization to supply a generic notion of analog algorithm and prove completeness results. Their postulates, defining analog algorithms, are in the same spirit of those given for discrete algorithms. These notions are reviewed and adapted in the next two sections.

Our study of stability considers Lyapunov energy functions for algorithms with continuous and discontinuous states. It extends preliminary work for purely dynamical systems [5] to handle hybrid systems with both discrete and analog transitions. This is the subject of Section 4.

The agents of an artificial swarm system are often hybrid by nature. Stability is a crucial property for such swarm agents.

2 Computability of Discrete Algorithms

The basic characteristic of a computable function, as formalized in [3,2], is that there must exist a finite description of an algorithm describing how to compute the function.

According to this view, a function is computable if: (a) given an input from its domain, it can give the corresponding output by following a procedure (program) that is formed by a finite number of exact unambiguous instructions – possibly relying on unbounded storage space; (b) it returns such output (halts) in a finite number of steps; and (c) if given an input that is not in its domain, it either never halts or it gets "stuck" and fails.

Gurevich [3] proposed a generic model of computation that incorporates these properties in what constitutes a "formal" algorithm, and which is outlined next.

Postulate I (Discrete system). An algorithm is a state-transition system, consisting of a set (or proper class) of states, a subset of which are initial states, and a partial transition function on states that determines the next-state relation. States with no next state are *terminal*.

Postulate II (Abstract state). States are first-order structures with equality, all sharing the same fixed, finite vocabulary, including the scalar (nullary function) true. States and initial states are closed under isomorphism. Transitions preserve the base set (domain), and transitions and isomorphisms commute. The interpretations given by a state x to the function symbols f in the vocabulary of the structure are denoted by $[\![f]\!]_x$, and extended in the usual way to (ground) terms.

Definition 1 (Locations and updates). *If f is a j-ary function symbol in the state vocabulary and $\bar{a}$ is a j-tuple of elements of the base set of a state x, then their combination $f(\bar{a})$ is called a* location. *We denote by $[\![f(\bar{a})]\!]_x$ its interpretation $[\![f]\!]_x(\bar{a})$ in x. When x and y are structures over the same base set and vocabulary, $y \backslash x$ is the set of* updates $\{f(\bar{a}) \mapsto [\![f(\bar{a})]\!]_y : [\![f(\bar{a})]\!]_y \neq [\![f(\bar{a})]\!]_x\}$.

Postulate III (Bounded exploration). There exists some finite set of ground terms over the vocabulary of the states, such that states that agree on the values of these terms also agree on all next-step state changes.

An abstract state machine, or ASM, is a state-transition system in which algebraic states (without predicate symbols) store the values of functions of the current state. Transitions are programmed using a convenient language based on guarded commands for updating individual states. ASMs captures the notion that each step of an algorithm performs a bounded amount of work, whatever domain it operates over, so are central to the succeeding development.

Definition 2 (ASM). *An* abstract state machine (ASM) *is given by a set of algebraic states (without predicate symbols) sharing a vocabulary and closed under isomorphism, a subset of initial states also closed under isomorphism, and a program P, composed of:*

- *assignments $s := u$, for terms s and u over the vocabulary of the states;*
- *conditionals* **if** q **then** P *or* **if** q **then** P **else** R*, where q is a conjunction of equalities and inequalities between terms and P and R are programs; and*
- *parallel composition* **par** $P_1, \dots P_n$ **rap***, for programs $P_1, \dots, P_n$.*

A program P defines a set of updates $\bigtriangleup_P(x)$ for each state x, according to the standard semantics of these programming constructs, each update being of the form $f(\bar{a}) \mapsto b$, for values $\bar{a}, b$ in the base set of x.

Gurevich [3] goes on to prove the following important result.

Theorem 3 (Representation). *For every process satisfying Postulates I–III, there is an abstract state machine (ASM) in the same vocabulary, with the same sets of states and initial states, that emulates it step-by-step, state-for-state.*

To capture the notion of effectiveness, one additional postulate regarding initial states is needed.

Postulate IV (Arithmetical state). Up to isomorphism, all initial states have the natural numbers as their base set, all share the same operations and constants – save input values, and there is exactly one initial state for each possible input. Their operations are all basic arithmetic ($+$, $-$, $\times$, $\div$, $<$), or can be programmed by ASMs using only basic arithmetic, or else are completely undefined.

Employing this last postulate, arithmetical ASMs may be defined [2]. With all this information, the Church Thesis is proved.

Theorem 4 (Church's Thesis). *A numeric function is partial recursive if and only if it is computed by a state-transition system satisfying Postulates I–IV. The input is contained in the initial state of a computation and the output in its terminal state.*

Remark 5. We have restricted this presentation to algorithms that work over the natural numbers. However, it is possible to extend it to other possible domains (strings, lists, graphs, etc.) by introducing an encoding notion and the concept of arithmetized algorithm, as done in [2]. No matter what other effective model of computation is chosen, its power of computation will not be increased beyond that given by partial recursive functions. Theorem 3 plays a fundamental role in the proof. See [2] for more details. As a corollary Turing's Thesis is obtained.

3 Effectiveness of Hybrid Algorithms

We are interested next in extrapolating from the above discussion to analog algorithms along the lines suggested by Bournez, Dershowitz, and Néron [1].

Postulate Ia (Dynamical system). A *hybrid* algorithm is a *dynamical system* (T, X, A, φ_t) consisting of a time set T (a monoid with an addition operator $+$ and neutral 0), a metric state space X (with metric d), initial states $A \subseteq X$, and a family of evolution operators $\varphi_t : X \rightharpoonup X$, parameterized by $t \in T$ (but not necessarily defined for all $t \in T$) and satisfying the following two properties: $\varphi_{t+s} = \varphi_t \circ \varphi_s$ and φ_0 is the identity function.

Remark 6. In our definition of dynamical system, it is allowed to have, in general, more than one evolution operator.

Dynamical systems are classified based on the properties of T, X, and φ. The time set T, is it continuous or discrete? Is the state space X finite or infinite? Continuous or discrete? Finite-dimensional or infinite-dimensional? Regarding the evolution map φ_t: is it deterministic or stochastic, autonomous or time-dependent, invertible or not, etc.?

When $T = \mathbb{R} = (-\infty, \infty)$, we speak of a *continuous-time* dynamical system, and when $T = \mathbb{N} = \{0, 1, 2, \cdots\}$ we speak of a *discrete-time* dynamical system. We will consider T equipped with the absolute value as a normed space $(T, |\cdot|)$.

A dynamical system is generally defined by one or more differential or difference equations.

Remark 7. When dealing with continuous dynamical systems determined by ordinary differential equations on $\mathbb{R}^n$, the euclidean metric d is

$$d(x, y) = |x - y| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \quad \forall x, y \in \mathbb{R}^n.$$

For discrete dynamical systems determined by difference equations, X equipped with this euclidean metric defines a metric space.

Definition 8 (Computable system). *A dynamical system is said to be* computable *if its family of evolution operators (also called its* trajectories*) are obtained as solutions of its mathematical model.*

Postulate IIa (Abstract state). A hybrid algorithm is an abstract transition system satisfying Postulate II.

Definition 9 (Generator). *An* infinitesimal generator *is a function that maps states to updates, and which respects isomorphisms.*

Definition 10 (Semantics). *A* semantics ψ *over a class C of sets S is a partial function mapping initial evolutions (non-point evolutions starting at $t = 0$) over some $S \in C$ to an element of S. The infinitesimal generator associated with a semantics ψ maps the state space X, for $x \in X$ such that $\psi([\![f(\bar{a})]\!]_{\varphi_t(x)})$ is defined for all locations $f(\bar{a})$, to the set of updates $\Delta_\psi(x) = \{f(\bar{a}) \mapsto \psi([\![f(\bar{a})]\!]_{\varphi_t(x)}) : f$ in vocabulary of x and $\bar{a}$ in base set of $x\}$.*

Remark 11. When $T = \mathbb{R}$, an example of semantics over the class of sets S containing T is the derivative ψ_{der}, when it exists. When $T = \mathbb{N}$, an example of semantics over the class of all sets would be the function $\psi_{\mathbb{N}}$ mapping f to $\psi_{\mathbb{N}}(f_n) = f_{n+1}, n \in \mathbb{N}$.

Remark 12. From now on, we assume that some semantics ψ is fixed to deal with different types of dynamical systems, it could be ψ_{der}, but it could also be another one. However, it is assumed that the class of dynamical systems is restricted to those that guarantee the existence of the respective semantics and as a result its associated set of updates is well defined. Therefore, not all possible dynamical systems are allowed.

The following corresponds to the Bounded Exploration Postulate, but now for continuous transitions.

Postulate IIIa (Bounded exploration). For any hybrid algorithm, there exists a finite set T of variable free terms over the vocabulary of its states, such that $\triangle_\psi(x) = \triangle_\psi(y)$ for all states x and y that coincide for all terms in T.

Definition 13 (Hybrid system). *A* hybrid algorithm *is a ψASM that satisfies Postulates Ia–IIIa.*

In addition to the rules of ASM programs as given in Definition 2, we need dynamic rules.

Definition 14 (Dynamic ASM). *Programs may include statements Dynamic$(f(t_1, \cdots, t_j), t_0)$, where f is a symbol of arity j and $t_0, t_1, \ldots, t_j$ are ground terms. This rule imposes constraints $\psi(f(t_1, \ldots, t_j)) = t_0$, on the updates $\triangle_\psi(x)$.*

The following plays a fundamental role.

Theorem 15. *For every hybrid algorithm, there is a ψASM that has the identical set of updates for all states.*

The proposed model can adequately describe hybrid systems, made of alternating sequences of continuous evolution and discrete transitions.

Example 16 (Bouncing ball). Let us consider a simple model of a bouncing ball, a classic example of a hybrid dynamical system, whose mathematical model is given by the equations $x'' = -gm$, where g is the gravitational constant and $v = x'$ is the velocity, except that upon impact, each time $x = 0$, the velocity changes according to $v = -kv$, where k is the coefficient of impact. Every time the ball bounces, its speed is reduced by a factor k. Its evolution is described by its associated set of updates of the following program rules

```
if $x = 0$ then $v := -kv$
           else par $Dynamic(x, v), Dynamic(v, -gm)$ rap
```

with dynamics ψ_{der}.

Definition 17 (Program). *A* ψASM *comprises the following: an ASM program, a set S of first-order structures with equality over some finite vocabulary $\mathcal{V}$ closed under isomorphisms with a subset I of S closed under isomorphisms, and a well-defined update set of computations $\triangle_\psi$ associated with ψ.*

We are assuming for that for each dynamical system, the trajectories can be computed from the description of its dynamical system, as, for example, in the case of nonlinear differential equation, the Lipschitz conditions are satisfied, etc. In other words, not all dynamical systems are contemplated just those that guarantee their existence.

Definition 18 (Unambiguity). *A semantics ψ is* unambiguous *if for all sets S of first-order structures over some finite vocabulary $\mathcal{V}$ closed under isomorphisms, and for all subsets $S' \in S$ closed under isomorphisms, whenever there exists some φ and a ψASM, then φ is unique.*

Bournez, Dershowitz, and Néron finish their presentation giving their main result (analogous to Theorem 3).

Theorem 19. *Assuming ψ is unambiguous, for every process satisfying Postulates Ia–IIIa, there is an equivalent ψASM.*

Theorem 20 (Church's Thesis for hybrid algorithms). *A dynamical system is computable if and only if a ψASM computes it.*

Proof. If the dynamical system is computable (per Definition 8), there exists an algorithm that computes its trajectories from its mathematical model description and, therefore, the ψASM program will be able to emulate and compute these trajectories by a proper definition of its rules. For the other direction of the implication, given a ψASM that first interprets the fixed dynamical system and then computes its trajectories, we define a numerical procedure that mimics it and therefore computes the dynamical system's trajectories. In fact, its trajectories define an exact mathematical model of themselves. □

4 Stability of Hybrid Algorithms

We are ready now to consider the stability concept for hybrid algorithms in terms of Lyapunov energy functions. We deal with algorithms whose states are structures with metric space S, d as base set.

Definition 21 (Stability). *Consider a hybrid algorithm. We say that state x with $a \in S$ and time-indexed location $f_{t,t_0}(a)$, where t and t_0 belong to T, is* stable *if for all $t_0 \in T$ and for all $\varepsilon > 0$ there exists $\delta = \delta(t_0, \varepsilon) > 0$ such that if given $a' \in S$, with $d(a', a) < \delta \Rightarrow d([\![f_{t,t_0}(a')]\!]_x, [\![f_{t,t_0}(a)]\!]_x) < \varepsilon$ for all $t \in T$.*

Chaotic systems are unstable.

Definition 22 (Continuity). *Consider a hybrid algorithm. We say that state x with $a \in S$ and time-indexed location $f_t(a)$ is* continuous at $t \in T$ *if for all $\varepsilon > 0$ there exists $\delta = \delta(t) > 0$ and state y such that if given $t' \in T$, with $|t - t'| < \delta \Rightarrow d([\![f_t(a)]\!]_x, [\![f_{t'}(a)]\!]_y) < \varepsilon$.*

Definition 23 (Class $\mathcal{K}$). *A continuous function $\alpha : [0, \infty) \to [0, \infty)$ is said to belong to* class $\mathcal{K}$ *if it is strictly increasing and $\alpha(0) = 0$.*

Postulate E (Bounded energy). The Lyapunov energy function associated with a hybrid algorithm at its starting time point $t_0 \in T$ multiplied by some finite constant $c \geq 1$ bounds the whole Lyapunov energy function, transferred or transformed by the whole algorithm, as the Lyapunov energy function evolves in time.

Theorem 24. *Consider a hybrid algorithm with the possibility of discontinuous states at points $t_1, t_2, \ldots \in T$. Assume there exists a Lyapunov function $V : S \times T \to \mathbb{R}^+$ and two functions $\alpha, \beta \in \mathcal{K}$, such that*

$$\begin{aligned}\alpha(d([\![f_{t,t_0}(a')]\!]_x, [\![f_{t,t_0}(a)]\!]_x)) &\leq V([\![f_{t,t_0}(a')]\!]_x, t)\\ &\leq \beta(d([\![f_{t,t_0}(a')]\!]_x, [\![f_{t,t_0}(a)]\!]_x))\end{aligned}$$

for all $a, a' \in S$, $t, t_0 \in T$. Assume Postulate E and that $[\![f_{t_0,t_0}(a')]\!]_x = a'$ holds, then the hybrid algorithm is stable.

Proof. We want to show that there exists a $\delta = \delta(t_0, \varepsilon) > 0$ such that given a' with $d(a', a) < \delta \Rightarrow d([\![f_{t,t_0}(a')]\!]_x, [\![f_{t,t_0}(a)]\!]_x) < \varepsilon$ for all $t \in T$. We claim $\delta = \beta^{-1}(\alpha(\varepsilon)/c)$ does the job. Indeed, $d([\![f_{t,t_0}(a')]\!]_x, [\![f_{t,t_0}(a)]\!]_x) \leq \alpha^{-1}(V([\![f_{t,t_0}(a')]\!]_x, t)) \leq \alpha^{-1}(cV([\![f_{t_0,t_0}(a')]\!]_x, t_0)) = \alpha^{-1}(cV(a', t_0)) \leq \alpha^{-1}(c\beta(d(a', a))) < \varepsilon$, where Postulate E has been used in the second inequality and the equation $[\![f_{t_0,t_0}(a')]\!]_x = a'$ in the first. □

An example of a stable hybrid algorithm whose Lyapunov function satisfies the conditions imposed by Theorem 24 is the one provided in [4], which consists of a ball in a constant gravitational field bouncing inelastically on a flat vibrating table. It is interesting to see how the Lyapunov function, proposed in the cited paper, monotonically decreases as t increases. In other words, Postulate E holds with $c = 1$.

References

1. Bournez, O., Dershowitz, N., Néron, P.: An axiomatization of analog algorithms. In: Computability in Europe 2016: Pursuit of the Universal (CiE, Paris, France). Lecture Notes in Computer Science, Vol. 9709., Switzerland, Springer (2016) 215–224. Available at `http://nachum.org/papers/AxiomatizationAnalog.pdf`; full version at `https://arxiv.org/pdf/1604.04295v2.pdf`
2. Dershowitz, N., Gurevich, Y.: A natural axiomatization of computability and proof of Church's Thesis. Bulletin of Symbolic Logic **14** (2008) 299–350. Available at `http://nachum.org/papers/Church.pdf`

3. Gurevich, Y.: Sequential abstract state machines capture sequential algorithms. ACM Transactions on Computational Logic **1** (2000) 77–111. Available at `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.146.3017&rep=rep1&type=pdf`
4. Heimsch, T.F., Leine, R.I.: A novel Lyapunov-like method for the non-autonomous bouncing ball system. In: Proceedings of the 7th European Nonlinear Dynamics Conference (ENOC), Rome (2011)
5. Retchkiman, Z., Dershowitz, N.: The Church thesis, its proof, and the notion of stability and stabilization for analog algorithms. Communications in Applied Analysis **23** (2019)