Regular Real Analysis

Swarat Chaudhuri
Rice University
swarat@rice.edu

Sriram Sankaranarayanan
University of Colorado, Boulder
srirams@colorado.edu

Moshe Y. Vardi
Rice University
vardi@cs.rice.edu

***Abstract*—We initiate the study of *regular real analysis*, or the analysis of real functions that can be encoded by automata on infinite words. It is known that ω-automata can be used to represent relations between real vectors, reals being represented in exact precision as infinite streams. The *regular functions* studied here constitute the functional subset of such relations.**

We show that some classic questions in function analysis can become elegantly computable in the context of regular real analysis. Specifically, we present an automata-theoretic technique for reasoning about limit behaviors of regular functions, and obtain, using this method, a decision procedure to verify the continuity of a regular function. Several other decision procedures for regular functions—for finding roots, fixpoints, minima, etc.—are also presented. At the same time, we show that the class of regular functions is quite rich, and includes functions that are highly challenging to encode using traditional symbolic notation.

I. Introduction

Real analysis is the branch of mathematics dealing with real numbers and functions over these. In particular, real analysis studies analytic properties of real functions and sequences, including convergence and limits of sequences of real numbers, the calculus of the real numbers, continuity, smoothness and related properties of real-valued functions. Calculus is a branch of real analysis that focuses, in essence, on the computational aspects of real analysis. The notation underlying calculus has evolved primarily to facilitate symbolic computation by hand. Therefore, the notation based on polynomials, ratios, radicals, exponentials, logarithms, and the like is intimately familiar to us.

While the standard notation of calculus continues to serve well for numerous applications to the natural sciences, there are at least two reasons why alternative representations of real functions are of interest. First, the traditional notation finds it difficult to express many natural functions. Consider for example the *Cantor function*, which: (A) takes an input $x \in [0, 1]$ expressed in base 3; (B) if x contains a 1, then replaces every digit after the first 1 by a 0; (C) replaces all 2s with 1s; and (D) interprets the result as a binary number, and returns this number (Cf. Example 1 and Fig. 2). Aside from being among the most well-known examples of fractal functions, this function is a frequently-cited example of a real function that is uniformly continuous but not absolutely continuous. At the same time, it is difficult to express this function in traditional analytical notation; neither is it easy to use the traditional calculus to infer its nontrivial analytical properties (e.g., continuity).

Second, even for the functions that the traditional notation can express, *automated* mathematical reasoning can pose a challenge. Consider the triangle waveform (Fig. 1), a fundamental class of waves in electrical engineering. Analytical representations of the characteristic function $\Delta(x)$ of a triangular wave with period 2 and varying between -1 and 1 include

$$\begin{aligned}\Delta(x) &= \frac{2}{\pi}\ \sin^{-1}[\sin(\pi x)] \\ &= 1 - 4|1/2 - \mathrm{frac}(x/2 + 1/4)|\end{aligned}$$

where $\mathrm{frac}(x)$ is the fractional part of x. However, we do not know of any natural, efficient procedure for deciding properties such as the continuity of expessions involving $\sin^{-1}(\cdot), \sin(\cdot)$ or $\mathrm{frac}(\cdot)$.

In this paper, we study an alternative, finite representation of functions of type $\mathbb{R}^k \to \mathbb{R}^l$ as *automata on infinite words*. Functions expressible this way will be called *regular real functions*.

It is well known that automata on infinite words can represent *regular relations* over real numbers, encoded as infinite streams of digits. Such representations have been used, for example, to develop decision procedures for real addition [1] and ω-automatic structures [2]. Our idea is to represent real regular functions as regular functional relations on the reals.

This new notation is quite conducive to efficient automated analysis. Indeed, the study of such automated analysis algorithms, rather than the expressiveness of the automata-based notation, forms the primary focus of this paper. Our central technical contribution is an automata-

This research was supported by NSF grants CCF-1156059 (CAREER), CCF-1162076, CNS-0953941 (CAREER), CNS-1049862, and CCF-1139011; NSF Expeditions in Computing project 1139011; BSF grant 9800096; and a gift from Intel.

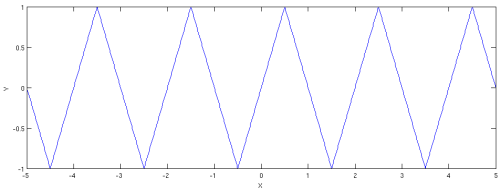


Figure 1. The triangle wave function.

theoretic technique for reasoning about limit behaviors of regular functions, which we apply in a PTIME ($O(n^4)$) decision procedure for verifying that a regular function is *continuous*. The same procedure can verify in PTIME that a regular function enjoys K-Lipschitz-continuity, a strong form of uniform continuity that implies differentiability almost everywhere. A corollary is that a continuous regular function is also Lipschitz-continuous. With minor modifications, the procedure can approximate, up to a constant factor, the optimal K for which the function is K-Lipschitz.

We also give an $O(n^2)$-time procedure for deciding the continuity of regular functions that can be represented by deterministic automata. Finally, we present procedures for linear rational arithmetic over functions, inverting a function, finding a function's roots and fixed points, and globally optimizing a function.

At the same time, we show our automata notation allows the encoding of functions that are extremely challenging to describe in a symbolic notation. For example, the Cantor function is now represented as a simple automaton that operates on inputs encoded in base 3 and generates outputs encoded in base 2. Other functions in this category include functions defined on fractals such as the Cantor set, and functions that mask bits in their real-valued inputs. The representation also allows the simple encoding of some functions like the triangle waveform that are expressible in the traditional symbolic notation, but not in a way that is amenable to automated symbolic reasoning.

The paper is organized as follows. In Sec. II, we define the class of regular real functions. Sec. IV, our main technical section, presents our decision procedures. We conclude with some discussion in Sec. V.

Related Work: Muller studied real functions computable online by a finite automaton [3], and Konecný studied real functions computable by finite transducers [4]. Rutten's study [5] of power series from an automata-theoretic perspective explored the expressive power of automata representing analytic objects. Also, the idea of representing exact reals by streams has been pursued in depth [6], [7]. In contrast, expressiveness is not the real concern of this work. Our study, instead, can be viewed as being part of *computable analysis*, which is concerned with the parts of analysis that can be carried out in a computable manner [8]. Our focus is on representation of functions as finite automata and the algorithmic consequences of such a representation.

Our focus here is also different than that of *automatic structures* [9], [2], [10]. In automatic structures the focus is on relational structures where the domain and relations can be represented by finite automata, as the first-order theory of such structures is decidable. Here we focus on functions that can be represented by finite automata, and our interest is in developing algorithms for their analytic properties.

Recently, some of us have studied algorithmic analysis of real functions encoded as *programs* [11], [12], [13], [14]. The procedures studied in those papers are sound but incomplete. In contrast, here we study analysis of functions represented by finite automata, and obtain efficient decision procedures for problems that are obviously undecidable in a Turing-complete notation.

Topological notions like continuity have been previously considered [15], [16] for automata representing functions from words to words. In particular, Carton et al. [16] study the continuity of functions over words represented by synchronized rational relations. Our work can be seen to study the implications of interpreting the input and output words of such functions as reals, with distances between them given by the Euclidean metric.

II. Regular real functions

In this section, we define ω-automata that recognize functions between real vector spaces. This definition depends on a representation of reals as infinite words; now we define this representation.

Modeling reals by ω-words: Let $\beta \geq 2$ be an integer-valued *base*. Let the *digits* under this base be $d_0, \ldots, d_{\beta-1}$, in increasing order, and let $Dig_\beta = \{d_0, \ldots, d_{\beta-1}\}$. Let $\mathsf{value}(d_j) : j$ for $j \in [0, \beta-1]$. Let positions in an ω-word be numbered $0, 1, \ldots$, and for any ω-word w, let $w(i)$ denote the symbol in the i-th position of w. Also, for $x \in \mathbb{R}$, let $Int_{x,\beta}$ and $Frac_{x,\beta}$ be the unique ω-words over Dig_β such that

$$\begin{aligned} |x| &= \sum_{i=0}^{\infty} \beta^i \mathsf{value}(Int_{x,\beta}(i)) \\ &\quad + \sum_{i=1}^{\infty} \beta^{-i} \mathsf{value}(Frac_{x,\beta}(i-1)) \\ Frac_{x,\beta} &\notin (Dig_\beta)^*(d_{\beta-1})^\omega \\ Int_{x,\beta} &\in (Dig_\beta)^*\ d_0^\omega \end{aligned}$$

Thus, $Int_{x,\beta}(i)$ and $Frac_{x,\beta}(i)$ are respectively the i-th least significant digit in the base-β representation of the integer part of x, and the i-th most significant digit in the base-β representation of the fractional part of x.

Now, let $\Sigma_\beta = Dig_\beta \times Dig_\beta$. Under base β, we represent a real x by a word $\rho_{x,\beta} = sgn \cdot w$, where:

1) sgn denotes the *sign symbol*. It equals the symbol $+$ if $x \geq 0$, and $-$ otherwise.
2) w is the unique ω-word over Σ_β such that for all $i > 0$, $\rho_{x,\beta}(i) = (Int_{x,\beta}(i), Frac_{x,\beta}(i))$.

The set of all words $\rho_{x,\beta}$ is denoted by R_β.

To define vectors, we need some more notation. Let $k > 0$ be a constant, integral *dimension*, let $\Sigma_\beta^k = \prod_{i=1}^k \Sigma_\beta$, and for $\sigma = (x_1, \ldots, x_k) \in \Sigma_\beta^k \cup \{+,-\}^k$, define $Proj_j(\sigma) = x_j$ for all j. For $\sigma_1 \in \Sigma_\beta^{k_1} \cup \{+,-\}^{k_1}$, $\sigma_2 \in \Sigma_\beta^{k_2} \cup \{+,-\}^{k_2}$, let

$$\langle\!\langle \sigma_1, \sigma_2 \rangle\!\rangle = (Proj_1(\sigma_1), \ldots, Proj_{k_1}(\sigma_1), \\ Proj_1(\sigma_2), \ldots, Proj_{k_2}(\sigma_2)).$$

For words $w_1 \in (\Sigma_\beta^{k_1} \cup \{+,-\}^{k_1})^\omega$ and $w_2 \in (\Sigma_\beta^{k_2} \cup \{+,-\}^{k_2})^\omega$, let $\langle\!\langle w_1, w_2 \rangle\!\rangle$ be the word w such that for all i, $w(i) = \langle\!\langle w_1(i), w_2(i) \rangle\!\rangle$. We abbreviate $\langle\!\langle w_1, \langle\!\langle w_2, \ldots, w_k \rangle\!\rangle \ldots \rangle\!\rangle$ by $\langle\!\langle w_1, \ldots, w_k \rangle\!\rangle$.

Now, a vector $\mathbf{x} \in \mathbb{R}^k$ is represented by a word $\rho_{\mathbf{x},\beta} = \langle\!\langle w_1, \ldots, w_k \rangle\!\rangle$ such that $w_i = \rho_{\mathbf{x}(i),\beta}$. The set of all words $\rho_{\mathbf{x},\beta}$, for some $\mathbf{x}$, is denoted by R_β^k. For $w \in \mathsf{R}_\beta^k$, we let $[\![w]\!]$ be $\mathbf{x} \in \mathbb{R}^k$ such that $\rho_{\mathbf{x},\beta} = w$.

Modeling functions: We define regular functions as restrictions of *synchronized rational relations* [17], or word relations that are accepted by finite automata operating on a product alphabet. The automata chosen for our definition are nondeterministic, and accept via the Büchi condition. Also, we allow the component words of our relations to be over different bases.

Recall that a *Büchi automaton* over a finite alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \longrightarrow, G)$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\longrightarrow \subseteq Q \times \Sigma \times Q$ is an *edge relation*, and $G \subseteq Q$ is a set of *repeating states*. We write $q_1 \xrightarrow{a} q_2$ if $(q_1, a, q_2) \in \longrightarrow$, and let the *size* of $\mathcal{A}$ be $(|Q| + | \longrightarrow |)$. A *run* of $\mathcal{A}$ on $w \in \Sigma^\omega$ is a word $\eta \in Q^\omega$ such that: (1) $\eta(0) = q_0$; and (2) for all i, $q_i \xrightarrow{w(i)} q_{i+1}$. The run η is *accepting* of some state in G appears infinitely often in η. The *language* $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of words on which $\mathcal{A}$ has an accepting run.

Also, let us lift the map $Proj_j$ as follows. Let $w \in (\Sigma_\beta^m \cup \{+,-\}^m)^\omega$, and let $w_1, \ldots, w_m$ be such that $w(i) = (w_1(i), \ldots, w_m(i))$ for all $i \geq 0$. For $1 \leq j \leq j' \leq m$, we define $Proj_{[j,j']}(w) = \langle\!\langle w_j, \ldots, w_{j'} \rangle\!\rangle$.

Definition 1 (Function automata, regular real functions)**.** Let β and γ, both positive integers greater than 1, respectively be the *input base* and the *output base*. Let $k, l > 0$ be integral *dimensions*. A *function automaton* of type $\mathbb{R}^k \to \mathbb{R}^l$ and over bases (β, γ) is a nondeterministic Büchi automaton over $(\Sigma_\beta^k \times \Sigma_\gamma^l) \cup \{+,-\}^{k+l}$ such that the following conditions hold:

(1) for all *inputs* $w \in \mathsf{R}_\beta^k$, there is at most one *output* $w' \in \mathsf{R}_\gamma^l$ such that $\langle\!\langle w, w' \rangle\!\rangle \in \mathcal{L}(\mathcal{A})$; and
(2) for all inputs $w \in \mathsf{R}_\beta^k$, there exists an output $w' \in \mathsf{R}_\gamma^l$ such that $\langle\!\langle w, w' \rangle\!\rangle \in \mathcal{L}(\mathcal{A})$.

The *(analytical) semantics* of $\mathcal{A}$ is the function $[\![\mathcal{A}]\!] : \mathbb{R}^k \to \mathbb{R}^l$ such that for all $\mathbf{x} \in \mathbb{R}^k, \mathbf{y} \in \mathbb{R}^l$,

$$[\![\mathcal{A}]\!](\mathbf{x}) = \mathbf{y} \text{ iff } \langle\!\langle \rho_{\mathbf{x},\beta}, \rho_{\mathbf{y},\gamma} \rangle\!\rangle \in \mathcal{L}(\mathcal{A}).$$

A function $f : \mathbb{R}^k \to \mathbb{R}^l$ is *regular* under input base β and output base γ iff it is the analytical semantics of a function automaton as above.

Our definition of regular functions is naturally generalized to one where different components of the input and output vectors of a function are coded in different bases. All the results of this paper hold even under this generalization. However, to keep our notation simple, we continue working with Definition 1.

We will also find useful a definition of regular sets of real vectors. Let us fix a base β. We let an automaton over (length-k) real vectors be a Büchi automaton $\mathcal{V}$ such that $\mathcal{L}(\mathcal{V}) \subseteq \mathsf{R}_\beta^k$. The *semantics* of $\mathcal{V}$ is the set $[\![\mathcal{V}]\!]$ such that $[\![\mathcal{V}]\!] = \{\mathbf{x} : \rho_{\mathbf{x}} \in \mathcal{L}(\mathcal{V})\}$. A set of real vectors is regular under base β if it equals $[\![\mathcal{V}]\!]$, for some automaton $\mathcal{V}$ as above.

Recognizing sets and functions: The *recognition problem* for regular functions (similarly, regular sets of real vectors) is to determine, given an arbitrary Büchi automaton $\mathcal{A}$, whether $\mathcal{A}$ is a function automaton (similarly, automaton over real vectors). We have:

Theorem 1. *The recognition problem for regular sets of vectors can be solved in $O(n^2)$ time. The recognition problem for regular functions is in PSPACE.*

Proof: (Sketch) We sketch a proof of the second statement. Given an automaton $\mathcal{A}_{in}$, we construct an automaton $\mathcal{A}$ that only runs on words of the form $\langle\!\langle w_1, w_2 \rangle\!\rangle$ where $w_1 \in \mathsf{R}_\beta^k, w_2 \in \mathsf{R}_\gamma^l$, and accepts such a word iff $\mathcal{A}_{in}$ does. This is achieved by intersecting $\mathcal{A}_{in}$ with k copies of an automaton that recognizes valid words in R_β that represent valid real number encodings and l copies of R_γ that represent valid words in base γ. To check if condition (1) in Definition 1 holds, we construct a product $\mathcal{A}'$ that operates over the alphabet

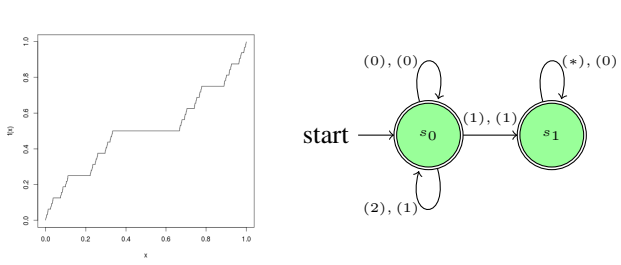


Figure 2. (**Top**) A plot of the cantor function on the $[0, 1)$ interval. (**Bottom**) A regular function representation with input base 3 and output base 2. The integer part since the inputs are in the $[0, 1)$ interval. Each edge is labeled by a pair $(i), (j)$ where i is the base 3 input bit and j is the base 2 output bit. $*$ denotes a *don't-care* digit (0, 1 or 2)

Σ^{k+2l} and satisfies

$$\mathcal{L}(\mathcal{A}') = \{\langle\langle w_1, w_2, w_3\rangle\rangle : \langle\langle w_1, w_2\rangle\rangle \in \mathcal{L}(\mathcal{A}), \langle\langle w_1, w_3\rangle\rangle \in \mathcal{L}(\mathcal{A}), w_2 \neq w_3\}.$$

Condition (1) in Def. 1 holds iff the language of this automaton is empty, which can be checked in quadratic time. To check condition (2), we construct an automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \{w : \exists w'.\langle\langle w, w'\rangle\rangle \in \mathcal{L}(\mathcal{A})\}$, then check that $\mathcal{A}'$ contains the language R^k_β of all valid words encoding real valued $k-$vectors. ■

III. Examples

In this section, we present some regular functions, showcasing some functions that can be represented succinctly by automata, but are challenging to describe in traditional symbolic notation.

Example 1 (Cantor function)**.** Consider the Cantor function $CF(x)$, which is a popular example of a fractal function and a function that is uniformly but not absolutely continuous, and is defined as follows:

1) Express x in base 3.
2) If x contains a 1, replace every digit after the first 1 by 0.
3) Replace all 2s with 1s.
4) Interpret the result as a binary number. The result $CF(x)$ is this number.

A plot of $CF(x)$ is provided in Fig. 2. It is easy to express $CF(x)$ by a function automaton over input base 3 and output base 2—the automaton is shown in Fig. 2 as well.

Example 2 (Modified Cantor Set)**.** Consider the Cantor set, defined informally as the limit of the following recursive subdivision process:

1) Start with an interval $[n, n + 1)$ where $n \in \mathbb{Z}$.

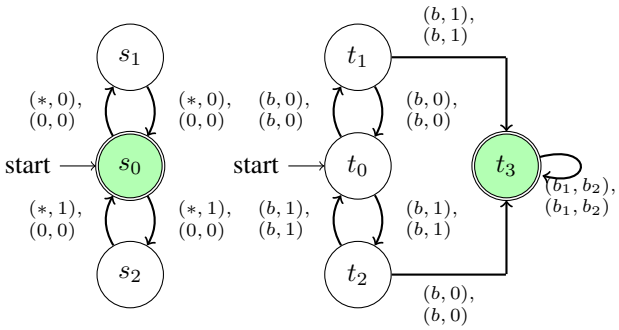


Figure 3. Automaton representing a function f_C which is non-zero outside the modified Cantor set. Note that that we have two start states s_0 and t_0. Each transition is labeled as $(i, r), (i', r')$ where (i, r) represents the integer and fractional parts of the input; and (i', r') represents integer/fractional parts of the output. Here b denotes a 0 or 1 bit that has the same value in the input and output. We omit the transitions reading the sign of the input.

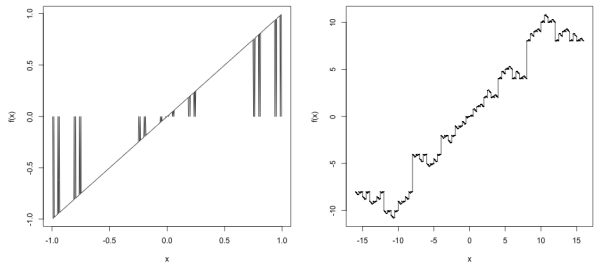


Figure 4. (**Left**) Function f_C on the modified Cantor set and (**Right**) A bitmasking function applying $x_i \wedge \overline{x_{i-1}}$ on successive bits of the binary expansion.

2) For each interval $[\ell, u) \subseteq \mathbb{R}$ encountered, we recursively apply the procedure on intervals $[\ell, \ell + \frac{(u-\ell)}{4})$ and $[\ell + \frac{3(u-\ell)}{4}, u)$.

Let C be the set obtained in the limit, which we will call the modified Cantor set. The set C is represented by real numbers r such that the fractional parts of the binary expansion of r lies in the language $(00|11)^\omega$.

Consider the following piecewise function f_C:

$$f_C(x) = \begin{cases} x & \text{if } x \notin C \\ 0 & \text{otherwise} \end{cases}$$

Figure 3 shows an automaton with input base 2 and output base 2 that recognizes the function f_C. Note that the automaton has two start states, and uses nondeterminism to guess upfront whether the input is expected to belong to C or otherwise.

Example 3 (Triangle waveform)**.** Consider the triangle waveform $\Delta(x)$ (Fig. 1) with period 2 and varying between -1 and 1. One definition of this function is

$\Delta(x) = 1 - 4|1/2 - \mathrm{frac}(x/2 + 1/4)|$, where $\mathrm{frac}(x)$ is the fractional part of x.

This function is easily encoded using an automaton $\mathcal{A}$ with input and output bases 2. Note that there is a simple automaton encoding the absolute value function $|\cdot|$: it just reads the sign of the input x and switches it from $-$ to $+$ if needed. An automaton for $\mathrm{frac}(x)$ is also easy to construct. Now we construct $\mathcal{A}$ using procedures for composing and doing arithmetic over regular functions that we give in Sec. IV.

Example 4 (Bit Masking Functions)**.** Consider the function that operates on the binary expansion of the fractional part by setting the value of the current bit f'_i of the output to be $f_i \wedge \overline{f_{i-1}}$. Likewise, for the integer part, we have $b'_i \leftarrow b_i \wedge \overline{b_{i+1}}$. The boundary case for $i = 0$ for the fractional part is handled by setting $f'_0 \leftarrow f_0 \wedge \overline{b_0}$. Figure 4 plots this function over a range of inputs in $(-16, 16)$. Note the fractal (self-similar) nature of this function. In fact, such functions which are readily expressible by finite state machines defy an easy analytical closed form characterization in terms of familiar algebraic or transcendental functions.

IV. A CALCULUS OF REGULAR REAL FUNCTIONS

This section presents our "calculus": a set of decision procedures for reasoning about regular functions. The central contribution here is an automata-theoretic method for reasoning about limit behaviors of regular functions, which we apply in a decision procedure to verify the continuity of a regular function f (Sec. IV-B).

A. Elementary operations

We start by giving procedures for some elementary operations on regular functions. Our first procedure is for *composing* two regular functions:

Theorem 2 (Composition)**.** *Given function automata $\mathcal{A}_1$ and $\mathcal{A}_2$ over the same input/output bases, one can construct in $O(n^2)$ time a function automaton $\mathcal{A}$ such that $[\![A]\!] = [\![\mathcal{A}_1]\!] \circ [\![\mathcal{A}_2]\!]$.*

Proof: We first add an extra input to $\mathcal{A}_1$ to yield $\mathcal{A}'_1$ that operates over a triple $\langle\langle w_1, w, w_2\rangle\rangle$ by simply ignoring w_2 and accepting the input if $\langle\langle w_1, w\rangle\rangle$ is accepted by $\mathcal{A}_1$. Likewise, $\mathcal{A}'_2$ is obtained by adding an extra input so that $\langle\langle w_1, w, w_2\rangle\rangle$ is accepted by $\mathcal{A}'_2$ iff $\langle\langle w, w_2\rangle\rangle$ is accepted by $\mathcal{A}_2$. We intersect $\mathcal{A}'_1$ and $\mathcal{A}'_2$, and project the middle input w. This yields the required function composition. ∎

A variant of the above construction shows that:

Theorem 3 (Application)**.** *Let $\mathcal{A}$ be an automaton accepting $f : \mathbb{R}^k \to \mathbb{R}^l$, and $\mathcal{V}$ an automaton encoding a set S of length-k vectors such that the input base of $\mathcal{A}$ equals the base of $\mathcal{V}$. The set $f(S)$ is regular, and an automaton for it can be constructed in $O(n^2)$ time.*

Regular functions are closed under linear arithmetic operations of addition, subtraction and scaling by rationals. These results are interesting given that it is known that addition over rationals is *not* expressible using automata over finite words [18]. Recent work by Abu Zaid et al. demonstrates the surprising result that if for any chosen representation for real numbers addition is $\omega-$automatic then multiplication is not, and vice-versa [19]. It follows that regular functions are not closed under multiplication under the representation chosen in this paper.

Theorem 4. *Given function automata $\mathcal{A}_1$ and $\mathcal{A}_2$ over the same input/output bases, one can construct in $O(n^2)$ time function automata $\mathcal{A}_+$ and $\mathcal{A}_-$ such that $[\![\mathcal{A}_+]\!] = [\![\mathcal{A}_1]\!] + [\![\mathcal{A}_2]\!]$ and $[\![\mathcal{A}_-]\!] = [\![\mathcal{A}_1]\!] - [\![\mathcal{A}_2]\!]$.*

Theorem 5. *Given a function automaton $\mathcal{A}_1$ and a rational scalar x (encoded in the same base as the input base of $\mathcal{A}_1$), one can construct in $O(n^2)$ time a function automaton $\mathcal{A}$ such that $[\![A]\!] = x[\![\mathcal{A}_1]\!]$.*

B. Reasoning about limit behaviors: Continuity and Lipschitz-continuity

Classical analysis is fundamentally the study of limit behaviors of functions. Now we present the centerpiece of this paper: an automata-theoretic method for reasoning about limit behaviors of regular functions. We show our method in action by using it in a PTIME decision procedure for verifying *continuity*, the most fundamental of analytic properties. The same procedure can decide *Lipschitz-continuity*.

For simplicity, we restrict ourselves here to regular functions of type $(0,1) \to (0,1)$.[1] Such functions can be represented by automata that make sure that the integral parts of the input and output are always of the form $(d_0)^\omega$. In fact, we assume that these integral parts, as well as the initial sign bit, do not exist at all—i.e., a real is just represented by an infinite word over the set of digits Dig. A function is then represented by an automaton over the alphabet $Dig \times Dig$, where the first component represents the input bit and the second component represents the output.

Definition 2 (Continuity, Lipschitz)**.** A function $f : (0,1) \to (0,1)$ is *continuous* at $y \in (0,1)$ if $\lim_{x \to y} f(x) = f(y)$. By an alternate definition,

[1]This assumption is only for simpler exposition. Our results extend to general regular functions $f : \mathbb{R}^k \to \mathbb{R}^l$.

f is continuous if for any sequence $\{x_n\}_{n\in\mathbb{N}}$ of points in $(0,1)$, we have $\lim_{n\to\infty} x_n = y \implies \lim_{n\to\infty} f(x_n) = f(y)$. By yet another definition (the Weierstrass definition), f is continuous at y if

$$\forall \epsilon > 0 : \exists \delta > 0 : \forall x : \\ y - \delta < x < y + \delta \Rightarrow f(y) - \epsilon < f(x) < f(y) + \epsilon.$$

The function f is *K-Lipschitz* (or Lipschitz-continuous with Lipschitz constant K) if for all $x, y \in (0,1)$ such that $x \neq y$, $\frac{|f(x)-f(y)|}{|x-y|} < K$.

It is well-known that for any function f, if f is Lipschitz-continuous, then f is continuous everywhere.

Now note that a decision procedure for continuity of regular functions follows from the Weierstrass definition: because regular functions are closed under addition and Büchi automata permit elimination of quantifiers over words, we can construct an automaton accepting the set of points at which a function is continuous. However, due to quantifier alternation in the definition, the complexity of this procedure is EXPSPACE [20]. In contrast, the method we present is in PTIME ($O(n^4)$). It is completely different from the above naive approach in that it relies on the definition of continuity in terms of limit sequences.

Now we proceed to our decision procedure. First we give a fast, $O(n^2)$-time procedure for the case when the automaton encoding the function is deterministic. For notational simplicity, we restrict ourselves in the rest of this section to the case where the input and output bases are both 2. However, the results are easily extended to general input/output bases.

1) Continuity Analysis: Deterministic Automaton: Let us define *deterministic function automata* in the standard way. For a deterministic automaton $\mathcal{A}$, and reachable state s, let $\text{out}_s(w)$ represent the output real number obtained for the input string with fractional part w and integer part 0^ω, with *the automaton initialized to state s.*

Theorem 6. *The function represented by $\mathcal{A}$ is* continuous *if and only if for every reachable state s in $\mathcal{A}$, $out_s(01^\omega) = out_s(10^\omega)$.*

The reader may notice that the fractional part 01^ω in base 2 does not represent a valid real number. However, this fact is immaterial to our theorem. The fact that the automaton is deterministic means that it produces outputs for $\alpha_i : 01^i0^\omega$ for each $i \in \mathbb{N}$. We can invoke pumping lemma and the properties of deterministic Büchi languages to show that $\mathcal{A}$ must necessarily produce some output for the string 01^ω starting from any reachable state s. Continuity analysis concerns this output. We now provide a proof of the theorem above, starting with the forward direction.

Lemma 1. *If there exists a reachable state s of $\mathcal{A}$ such that for strings $\alpha : 01^\omega$ and $\gamma : 10^\omega$, $out_s(\alpha) \neq out_s(\gamma)$ then the function f is discontinuous.*

Proof: Consider the sequence of inputs $\alpha_i : 0(1)^i0^\omega$ for $i = 1, 2, \ldots, \infty$. We note that since $\mathcal{A}$ is deterministic, the output produced by $\mathcal{A}$ starting at state s when fed the string α_i will coincide with the output from α upto i or more places. Let the output from α and γ deviate after N digits. Due to determinism we conclude that the outputs for $\alpha_{N+1}, \alpha_{N+2}, \ldots$ deviate from the output for γ.

Now let π be some finite prefix that allows us to reach the state s starting from the initial state s_0. Consider the real value represented by $x : (\pi\gamma)$ and the sequence of real values represented by $x_i : (\pi\alpha_i)$. We note that $x_i \to x$ as $i \to \infty$. However, $|f(x_i) - f(x)| > 2^{-(1+N+|\pi|)}$. Therefore, f is discontinuous at x. ■

We now prove the other direction of the implication. We note that, in general, every discontinuity of f is witnessed by the input r and a sequence $x_1, x_2, \ldots$ such that $x_j \to r$ as $j \to \infty$ and $f(x_j) \not\to f(r)$. The following claim is quite useful in proving the reverse direction. Let w_r be the word in $(0+1)^\omega$ that represents r and z_j represent x_j.

Lemma 2. *Assume that w_r has infinitely many 0s and infinitely many 1s. For any sequence $x_j \to r$, let $K_j \geq 0$ be the length of the longest common subsequence between the ω-words z_j and w_r. It follows that $K_j \to \infty$ as $j \to \infty$.*

Lemma 3. *If f is discontinuous at r then r must be represented by a string of the form $\pi 0^\omega$.*

Proof: We split cases on w_r, the word representation of r. (case-1) w_r has infinitely many zeros and infinitely many ones. From Lemma 2 above, z_j and w_r coincide to arbitrarily many positions as $j \to \infty$. Therefore, the determinism of $\mathcal{A}$ guarantees that a discontinuity cannot occur at r since if the inputs coincide to K_j places, then so must the outputs.

(case-2) The only possible case is that w_r has finitely many 1s (finitely many zeros is not a possible representation of a real). Therefore r must be represented by a string of the form $\pi 0^\omega$. ■

Finally, we can prove the reverse direction.

Lemma 4. *If a function is discontinuous then there exists a reachable state s of $\mathcal{A}$ such that $out_s(10^\omega) \neq out_s(01^\omega)$.*

Theorem 6 extends to bases $\beta \geq 2$. For base 3, we require that at each state the outputs for the string $\alpha_1 : 10^\omega$ coincide with $\gamma_1 : 02^\omega$, and the output for $\alpha_2 : 20^\omega$ be equal to $\gamma_2 : 12^\omega$.

Example 5. Consider the automaton for the Cantor function from Example 1 (see Fig. 2). We verify continuity by comparing the output on states s_0 and s_1 for the strings α_j, γ_j for $j = 1, 2$. State s_1 has an output 0^ω regardless of the input. For s_0, $\text{out}_{s_0}(10^\omega) = 10^\omega$ and $\text{out}_{s_0}(01^\omega) = 01^\omega$. Both 10^ω and 01^ω are binary representations of $\frac{1}{2}$ (we ignore the fact that the latter input/output pair are disallowed in our formalism). Next, we observe that $\text{out}_{s_0}(20^\omega) = 10^\omega$ and $\text{out}_{s_0}(12^\omega) = 10^\omega$. Therefore, we verify that the Cantor function is continuous.

Theorem 7. *Given a deterministic regular function $\mathcal{A}$ the complexity of checking continuity is $O(|\mathcal{A}|^2)$.*

2) Continuity Analysis: Nondeterministic Automata: Now we consider continuity analysis for the more general case where the automaton encoding a regular function f is nondeterministic. In fact, what we give is an automata-based construction to search for a real number $x \in (0, 1)$ at which the function is *discontinuous*. In more detail, the construction simulates two sequences $\{x_i\}$ and $\{y_i\}$ that converge to x in the limit as $i \to \infty$, such that, $f(x_i)$ and $f(y_i)$ converge to different limits, or diverge. The main technique is to define an automaton $\mathcal{A}_R$ that accepts a four-tuple of reals $(x, y, f(x), f(y))$. The automaton's states are labelled by propositions to track positional differences between the binary expansions of x, y and $f(x), f(y)$ respectively. A discontinuity is found when the position of divergence between x, y can be pushed arbitrarily far away from the radix point whereas the divergence between $f(x), f(y)$ can be made to stay within some initial bound. Such a pattern is established by finding a *lasso* in $\mathcal{A}_R$.

Our construction uses a widget automaton $\mathcal{A}_w$ for tracking positional differences between two input sequences in Σ^ω. This automaton recognizes a relation over inputs $x, y \in (0, 1)$. The states of $\mathcal{A}_w$ are partitioned into two sets S (standing for "Same") and D (standing for "Different").

$\mathcal{A}_w$ has the following key property: For each input (x, y) such that $2^{-(1+K)} \leq |x - y| < 2^{-K}$ for some $K \geq 0$, every resulting run remains in a S state for the first $K + 1$ steps and then transitions to a D state, remaining in a D state for the rest of the run.

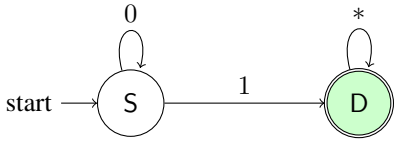
Formally, the language accepted by $\mathcal{A}_w$ is $L_w = \{(x, y) \mid x \neq y,\ x, y \in (0, 1)\}$. However, more important than the language accepted is the labeling function performed by the run of $\mathcal{A}_w$, which precisely pinpoints the first point of difference between the sequences representing (x, y). In this regard, we may also regard $\mathcal{A}_w$ as a finite state transducer outputting a label S or D upon encountering each bit of (x, y).

1) Consider inputs $x = 0\underline{1}0011101\ldots$ and $y = 0\underline{0}10011000\ldots$. We note that the first position of difference is at the second significant digit (implicitly, all digits are after the radix point). Therefore, we require each run of $\mathcal{A}_w$ to be of the form $\mathsf{S}^2\mathsf{D}^\omega$.
2) On the other hand, consider the input $x = 0010000^\omega$ and $y = 000111\ldots$. The significant difference here is actually at the sixth digit, even though the two inputs seem to have diverged at the third significant place. This subtlety arises for inputs that fit the pattern $(0{+}1)^*10^\omega$, wherein there is an infinite trail of zeros. Therefore, we expect each run of $\mathcal{A}_w$ upon input (x, y) to be of the form $\mathsf{S}^6\mathsf{D}^\omega$.

We now describe the construction of $\mathcal{A}_w$. For the sake of presentation, we do not show all the parts of the construction, appealing instead to well known closure properties of ω- automata.

We construct $\mathcal{A}_w$ by composing together the following components:

1) Let $\mathcal{A}_{|x-y|}$ represent the function automaton that computes $|x - y|$. This is obtained by composing the functions $f(x, y) = (x - y)$ and $g(z) = |z|$.
2) Let $\mathcal{A}_1$ represent the automaton with two states S, D, standing for *same* and *different*, respectively, that remains in the S state as long as the input is 0 and transitions to a D state as soon as a 1 is encountered:



The overall automaton $\mathcal{A}_w$ is obtained by composing the automaton for $\mathcal{A}_{|x-y|}$ with the automaton $\mathcal{A}_1$ to track the first position of difference, as explained above. The composition is intended to simulate the computation of $|x - y|$ by $\mathcal{A}_{|x-y|}$ upon encountering x, y. The result is fed to $\mathcal{A}_1$ operating on the output of $\mathcal{A}_{|x-y|}$. A standard product construction between $\mathcal{A}_{|x-y|}$ and $\mathcal{A}_1$ achieves the composition. The output of $\mathcal{A}_{|x-y|}$ is then projected away.

Each state of $\mathcal{A}_w$ is a pair consisting of a state of $\mathcal{A}_{|x-y|}$ and $\mathcal{A}_1$. A state of $\mathcal{A}_w$ is said to be a S state if the component corresponding to $\mathcal{A}_1$ of the state is S.

Likewise, a state of $\mathcal{A}_w$ is said to be a D state if the component of $\mathcal{A}_w$ is said to be D.

Lemma 5. *For any inputs x, y, where $2^{-K} > |x - y| \geq 2^{-(1+K)}$ then every accepting run of $\mathcal{A}_w$ remains in a S state for the first $K+1$ steps and in a D state for the remainder of the run.*

Proof: $\mathcal{A}_w$ works (conceptually) by first computing $\mathcal{A}_{|x-y|}$ and then feeding the output to $\mathcal{A}_1$. Since $|x - y|$ is a function, the output for a given x, y is unique. Likewise, note that $\mathcal{A}_1$ is deterministic. Therefore, for a given output $|x-y|$, the run induced on $\mathcal{A}_1$ is unique. Furthermore, if $|x-y| \in [2^{-K}, 2^{-(1+K)})$, then $|x-y|$ is of the form $0^K 1(0+1)^\omega$. Therefore, $\mathcal{A}_1$ has a run of the form $\mathsf{S}^{K+1}\mathsf{D}^\omega$. As a result, for the overall composition $\mathcal{A}_w$, any resulting run (note that $\mathcal{A}_{|x-y|}$ need not be deterministic) has a prefix of $K+1$ states labelled S and the remainder of the states labelled as D. ∎

The next step of the procedure constructs an automaton $\mathcal{A}_R$ representing a relation R between four real numbers (x, y, z, w) such that:

1) $z = f(x)$ and $w = f(y)$. This is obtained as an interaction between two copies of the automaton for f, one for each of the equalities. Extra tapes added to the first copy enforcing $z = f(x)$, to accommodate y, w, and likewise x, z tapes added to the second copy enforcing $w = f(y)$.
2) We take the product with $\mathcal{A}_w(x, y)$ and $\mathcal{A}_w(z, w)$ to mark the positional differences between the inputs x, y and the outputs $f(x), f(y)$.

We label a state in the product automaton $\mathcal{A}_R$ as S_I if the component corresponding to $\mathcal{A}_w(x, y)$ is a S state and D_I otherwise. Likewise, we label a state S_O if the component $\mathcal{A}_w(z, w)$ is part of a S state and D_O otherwise.

Informally, a state labelled S_I in $\mathcal{A}_R$ tells us that the inputs x, y have not diverged. Likewise a state labelled D_I tells us that the inputs have diverged sometime in the past. The same consideration applies to the S_O and D_O labels.

We assume that all states in $\mathcal{A}_R$ are reachable from the initial state and furthermore, every state in $\mathcal{A}_R$ can reach a repeating cycle. States that do not satisfy these criteria can be removed from $\mathcal{A}_R$ without modifying its language.

To decide if f is discontinuous, we search for a $(\mathsf{S}_I, \mathsf{D}_O)$ lasso in $\mathcal{A}_R$—i.e., a substructure of $\mathcal{A}_R$ with the following components (see Figure 5):

1) A path π_1 from some start state s_0 satisfying S_I to a state s_1 satisfying $(\mathsf{S}_I, \mathsf{D}_O)$. The states involved in the path π_1 are all S_I states.

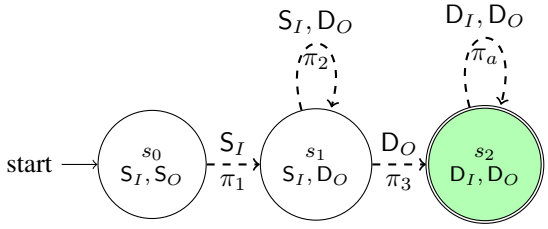


Figure 5. A $(\mathsf{S}_I, \mathsf{D}_O)$ lasso for finding discontinuity.

2) A $(\mathsf{S}_I, \mathsf{D}_O)$ cycle π_2 from s_1 onto itself.
3) A path π_3 from s_1 to a repeating state s_2 satisfying $(\mathsf{D}_I, \mathsf{D}_O)$ that lies in a cycle.
4) An accepting cycle π_a containing s_2.

The main result of our approach is based on the following theorem:

Theorem 8. *Function f is discontinuous if and only if, $\mathcal{A}_R$ has a $(\mathsf{S}_I, \mathsf{D}_O)$ lasso.*

Proof: We will show that if such a path exists, then f is indeed discontinuous. Next we will show that if f is discontinuous, then such a lasso can be found.

We recall that f is continuous iff for every sequence $\{x_i\}$ converging to x, the sequence $\{f(x_i)\}$ converges to $f(x)$. Our approach will demonstrate two different sequences converging to some x, wherein, the values of the functions converge to different values.

($\Leftarrow$) Assume that a $(\mathsf{S}_I, \mathsf{D}_O)$ lasso, as described in Figure 5, can be found in the automaton $\mathcal{A}_R$.

We consider sequences $\{x_i\}$, $\{y_i\}$ that follow the path $\pi(n) : \pi_1\pi_2^n\pi_3\pi_a^\omega$ for each $n > 0$. Let N_j represent the length of a path π_j for $j = 1, 2, 3$. Let $V(x, \pi)$ represent the value of x along a path segment π (assuming that the implicit radix point lies just before the path).

The value of x represented by $\pi(n)$ is

$$\begin{aligned} x_n &= V(x, \pi_1) + 2^{1-N_1}(1 - 2^{-nN_2-1})V(x, \pi_2) \\ &\quad + 2^{-N_1-nN_2}V(x, \pi_3) + 2^{1-N_1-nN_2-N_3}V(x, \pi_a)\,. \end{aligned}$$

Likewise, the value of y represented by $\pi(n)$ is

$$\begin{aligned} y_n &= V(y, \pi_1) + 2^{1-N_1}(1 - 2^{-nN_2-1})V(y, \pi_2) \\ &\quad + 2^{-N_1-nN_2}V(y, \pi_3) + 2^{1-N_1-nN_2-N_3}V(y, \pi_a)\,. \end{aligned}$$

Note that since π_1, π_2 are part of S_I, we have

$$V(x, \pi_1) = V(y, \pi_1), \quad V(x, \pi_2) = V(y, \pi_2)\,.$$

Consider the limits of the sequence x_n, y_n, as $n \to \infty$. Specifically,

$$\begin{aligned} \lim_{n\to\infty} x_n &= V(x, \pi_1) + 2^{1-N_1}V(x, \pi_2) \\ &= V(y, \pi_1) + 2^{1-N_1}V(y, \pi_2) \\ &= \lim_{n\to\infty} y_n \end{aligned}$$

Likewise, we note that $V(f(x), \pi_1) \neq V(f(y), \pi_2)$. Therefore, for all n,

$$|f(x_n) - f(y_n)| > 2^{-N_1}\,.$$

Therefore, we have

$$\left|(\lim_{n\to\infty} f(x_n)) - (\lim_{n\to\infty} f(y_n))\right| \;>\; 2^{-N_1}\,.$$

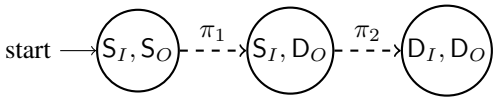
Therefore, f is discontinuous at $x = V(x, \pi_1) + 2^{1-N_1}V(x, \pi_2)$.

($\Rightarrow$) We will now prove that if $\mathcal{A}_R$ does not have a $(\mathsf{S}_I, \mathsf{D}_O)$ lasso, then the function f is Lipschitz continuous. I.e.,

$$(\exists\ K)\ (\forall\ x, y)(x \neq y)\ \Rightarrow\ \frac{|f(x) - f(y)|}{|x - y|} \leq K\,.$$

It is well-known that Lipschitz continuity implies continuity. Consider any two inputs x, y such that $x \neq y$. The tuple $(x, y, f(x), f(y))$ has an accepting run through $\mathcal{A}_R$. Each run starts in a $(\mathsf{S}_I, \mathsf{S}_O)$ state and ends up in a $(\mathsf{D}_I, \mathsf{D}_O)$ repeating cycle. We differentiate two types of accepting runs:

1) The run reaches a D_O state at the same step or after reaching a D_I state. We note that in this case, the output diverges at the same step or after the inputs do. Therefore, $|f(x) - f(y)| \leq |x - y|$.
2) The run reaches a $(\mathsf{S}_I, \mathsf{D}_O)$ state before reaching a $(\mathsf{D}_I, \mathsf{D}_O)$ state. The diagram below illustrates the prefix of such a run:



In this case, there are no cycles involving $(\mathsf{S}_I, \mathsf{D}_O)$ states, or else, we will have a $(\mathsf{S}_I, \mathsf{D}_O)$ lasso in $\mathcal{A}_R$. Therefore, the length of the run segment π_2 is bounded by N_s, the number of states satisfying $(\mathsf{S}_I, \mathsf{D}_O)$. As a result, in this case, we have $|f(x) - f(y)| < 2^{-|\pi_1|}$ while $|x - y| > 2^{-|\pi_1|-N_s}$. Therefore, $\frac{|f(x)-f(y)|}{|x-y|} < 2^{N_s} = 2^{O(N^2)}$.

■

Observe that the essential idea in the above proof—capturing a limit of the form $\lim_{x\to 0} g(x)$ using a cycle in an automaton that can be "pumped" arbitrarily many times—is not restricted to continuity. We can apply this idea to other properties that involve limits as well.

Theorem 9. *Given a function automaton $\mathcal{A}$, we can check if $[\![\mathcal{A}]\!]$ is continuous in $O(n^4)$ time.*

Proof: The automaton $\mathcal{A}_R$ is quadratic in the size of $\mathcal{A}$. We can determine if $\mathcal{A}_R$ has a (S_I, D_O) lasso in $O(|\mathcal{A}_R^2|)$ time. ■

Now we note that:

Theorem 10. *If a regular function f is continuous then it is K-Lipschitz with $K < 2^{O(n^2)}$, where n is the size of the automaton representing f. Further, in this case our decision procedure can compute at no extra cost a K such that $K \leq \gamma K_{\min}$, where $K_{\min}$ is the* minimum Lipschitz constant *for which f is Lipschitz-continuous, and γ is the output base.*

Proof: The proof of the first statement simply copies the ($\Rightarrow$) direction of the proof of Theorem 8.

We prove the rest for $\gamma = 2$. If f is continuous, then our procedure can compute using a graph search the maximum length N_s of the fragment π_2 in any run of $\mathcal{A}_R$ (π_2 and $\mathcal{A}_R$ are as before). Let the K in the theorem statement equal 2^{N_s}. Now observe that if $K_{\min} < K/2 = 2^{N_s-1}$, this maximum length is $(N_s - 1)$ rather than N_s, which is a contradiction. ■

We note that by Rademacher's theorem [21], a K-Lipschitz function $f : \mathbb{R}^k \to \mathbb{R}^l$ is differentiable almost everywhere (i.e., the set of all its nondifferentiable points has measure 0). It follows that a continuous regular function is differentiable almost everywhere.

Can we verify that a regular function is differentiable everywhere? Note that a witness to a function's non-differentiability is a pair of sequences that show that the limit $\lim_{h\to 0} D(x, h)$ where $D(x, h) = \frac{f(x+h)-f(x)}{h}$ does not exist. If D was a regular function, our approach for reasoning about limits would extend to this problem as well. The challenge, however, is that $D(x, h)$ contains a division, which cannot in general be encoded by finite automata.

C. Inversion, roots, fixpoints, optimization

An appealing feature of regular functions is that their roots can be found easily. Before we show why, observe that regular functions are easily invertible:

Theorem 11. *Given a function automaton $\mathcal{A}$ for a bijective regular function $f : \mathbb{R}^k \to \mathbb{R}^l$, we can construct a function automaton for f^{-1} in $O(n)$ time.*

Proof: We simply swap the inputs and outputs of $\mathcal{A}$—i.e., we replace each transition $q \xrightarrow{\langle\!\langle \sigma_1, \sigma_2 \rangle\!\rangle} q'$ in $\mathcal{A}$, where $\sigma_1 \in \Sigma^k, \sigma_2 \in \Sigma_l$, by the transition $q \xrightarrow{\langle\!\langle \sigma_2, \sigma_1 \rangle\!\rangle} q'$. As f is bijective, this modified automaton $\mathcal{A}'$ is a function automaton encoding $f^{-1} : \mathbb{R}^l \to \mathbb{R}^k$. ■

Note that the problem of *checking* whether a given function automaton represents a bijective function is in PSPACE. We simply use the above construction to generate $\mathcal{A}'$, then check that this automaton is a function automaton using the algorithm in Theorem 1.

Now consider the problem of computing the *roots* of a regular function f—i.e., the set of all solutions to the equation $f(\mathbf{x}) = \mathbf{0}$. We have:

Theorem 12. *The set R of roots of a regular function f is regular as well. Given an automaton $\mathcal{A}$ for f, an automaton for R can be constructed in $O(|\mathcal{A}|)$ time.*

Proof: Let $\mathcal{A}_0$ be the automaton that accepts $\langle\langle w, 0\rangle\rangle$ for arbitrary string w and output 0. The size of $\mathcal{A}_0$ is fixed given the dimensionality of the output for f. We intersect $\mathcal{A}$ with $\mathcal{A}_0$ and project the second (output) component away from the result. The resulting automaton accepts the set of roots of $\mathcal{A}$. ■

The above proof can be easily modified to show that:

Theorem 13. *The set F of fixpoints of a regular function f is regular as well. Given an automaton $\mathcal{A}$ for f, an automaton for F can be constructed in $O(|\mathcal{A}|)$ time.*

It is also easy to *globally optimize* a regular function. Given a regular function f, let us seek the set of *minima* (maxima) of f—i.e., inputs $\mathbf{x}$ for which $f(\mathbf{x})$ is minimized (maximized). We have:

Theorem 14. *If f is a regular function, then the set μ_f of minima of f is regular as well. Further, given an automaton for f, we can construct in exponential time an automaton for μ_f.*

V. Conclusion

In this paper, we have introduced *ω-automata* as a representation of real functions. We have given examples of nontrivial functions (e.g., the Cantor function) expressible this way, and presented a PTIME method for deciding the continuity of a regular function.

We leave open two questions. First, is it decidable to check whether a regular function is *differentiable*? For reasons given at the end of Sec. IV-B, this problem is challenging. Second, is there a simple characterization of the class of continuous or differentiable regular functions? Note that the Cantor function is an example of a nontrivial regular function that is also continuous.

References

[1] B. Boigelot, S. Jodogne, and P. Wolper, "An effective decision procedure for linear arithmetic over the integers and reals," *ACM Trans. Comput. Log.*, vol. 6, no. 3, pp. 614–633, 2005.

[2] A. Blumensath and E. Grädel, "Automatic structures," in *LICS*, 2000, pp. 51–62.

[3] J. Muller, "Some characterizations of functions computable in on-line arithmetic," *IEEE Trans. Computers*, vol. 43, no. 6, pp. 752–755, 1994.

[4] M. Konecný, "Real functions computable by finite automata using affine representations," *Theor. Comput. Sci.*, vol. 284, no. 2, pp. 373–396, 2002.

[5] J. Rutten, "Automata, power series, and coinduction: Taking input derivatives seriously," in *ICALP*, 1999, pp. 645–654.

[6] H.-J. Boehm, R. Cartwright, M. Riggle, and M. O'Donnell, "Exact real arithmetic: A case study in higher order programming," in *LISP and Functional Programming*, 1986, pp. 162–173.

[7] P. Potts, A. Edalat, and M. Escardó, "Semantics of exact real arithmetic," in *LICS*, 1997, pp. 248–257.

[8] K. Weihrauch, *Computable analysis: an introduction*. Springer, 2000.

[9] B. Khoussainov and A. Nerode, "Automatic presentations of structures," in *LCC*, 1994, pp. 367–392.

[10] B. Khoussainov, A. Nies, S. Rubin, and F. Stephan, "Automatic structures: Richness and limitations," *Logical Methods in Computer Science*, vol. 3, no. 2, 2007.

[11] S. Chaudhuri, S. Gulwani, and R. Lublinerman, "Continuity analysis of programs," in *POPL*, 2010, pp. 57–70.

[12] S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. NavidPour, "Proving programs robust," in *SIGSOFT FSE*, 2011, pp. 102–112.

[13] S. Chaudhuri, S. Gulwani, and R. Lublinerman, "Continuity and robustness of programs," *CACM*, vol. 55, no. 8, pp. 107–115, 2012.

[14] S. Chaudhuri and A. Solar-Lezama, "Smooth interpretation," in *PLDI*, 2010, pp. 279–291.

[15] C. Prieur, "How to decide continuity of rational functions on infinite words," *Theor. Comput. Sci.*, vol. 276, no. 1-2, pp. 445–447, 2002.

[16] O. Carton, O. Finkel, and P. Simonnet, "On the continuity set of an omega rational function," *ITA*, vol. 42, no. 1, pp. 183–196, 2008.

[17] C. Elgot and J. Mezei, "On relations defined by generalized finite automata," *IBM Journal of Research and Development*, vol. 9, no. 1, pp. 47–68, 1965.

[18] T. Tsankov, "The additive group of the rationals does not have an automatic presentation," *Journal of Symbolic Logic*, vol. 76, no. 4, pp. 1341–1351, 2011.

[19] F. Abu Zaid, E. Grädel, and Ł. Kaiser, "The Field of Reals is not omega-Automatic," in *STACS'12*, 2012.

[20] A. P. Sistla, M. Vardi, and P. Wolper, "The complementation problem for büchi automata with appplications to temporal logic," *Theor. Comput. Sci.*, vol. 49, pp. 217–237, 1987.

[21] E. DiBenedetto, *Real analysis*. Springer, 2002.