Self-graphing equations

July 4, 2013

Can you find an xy-equation that, when graphed, writes itself on the plane? This idea became internet-famous when a Wikipedia article on *Tupper's self-referential formula* went viral in 2012. Under scrutiny, the question has two flaws: it is meaningless (it depends on fonts) and it is trivial. To see the latter, let $F\colon \mathbb{R}^2 \to \{0,1\}$ be the function such that the graph $F(x,y) = 1$ is given by Figure 1. The equation $F(x,y) = 1$ is self-graphing by construction. (Objection: this equation is not in closed form! But what does that mean? *Closed form* is not a formally defined, or universally agreed upon, notion [2].) In Section 1 we fix these two flaws by formalizing the problem. In Sections 2 and 3 we use mathematical logic to prove existence of self-graphing equations in a conceptual way (no messy details required). The proof resembles the abstract proof that there are self-printing computer programs (or *quines*).

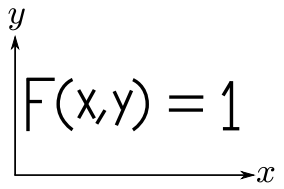

Figure 1: The graph used to define $F\colon \mathbb{R}^2 \to \{0,1\}$. It consists of thirteen line segments—two vertical, five horizontal and six diagonal—and two Bézier curves, arranged in a particular layout.

1 Formalization

> "What was a compelling proof in 1810 may well not be now; what is a fine closed form in 2010 may have been anathema a century ago."—J. Borwein and R. Crandall [2]

Definition 1. If $A \neq \emptyset$ is a finite alphabet, let $A^{<\mathbb{N}}$ be the set of finite strings from A. By a *notion of equations* we mean a finite alphabet A together with a *graphing function* $\mathrm{Gr}\colon A^{<\mathbb{N}} \to \mathscr{P}(\mathbb{R}^2)$ assigning to every $\sigma \in A^{<\mathbb{N}}$ a subset $\mathrm{Gr}(\sigma)$ of $\mathbb{R}^2$ called the *graph* of σ.

If A contains symbols x, y, ^{2}, $+$, $=$, and 1, $\mathrm{Gr}(x^2 + y^2 = 1)$ might (or might not) be the unit circle in $\mathbb{R}^2$. $\mathrm{Gr}(y = +)$ might (or might not) be the empty set, or even an error message written on the plane.

Definition 2. By a *glyphed notion of equations* we mean a triple $(A, \mathrm{Gr}, \mathrm{Gl})$ where (A, Gr) is a notion of equations and $\mathrm{Gl}\colon A \to \mathscr{P}([0,1]^2)$ is an injective function assigning to each $a \in A$ a distinct $\mathrm{Gl}(a) \subseteq [0,1]^2$ called the *glyph* of a.

If A contains a symbol X, $\mathrm{Gl}(X)$ might (or might not) be the union of the line segment from $(0,0)$ to $(1,1)$ and the line segment from $(0,1)$ to $(1,0)$.

Definition 3. If $X \subseteq \mathbb{R}^2$ and $n \in \mathbb{R}$, let $X_{\to n} = \{(x+n, y) : (x,y) \in X\}$ be the result of translating X to the right by n units. Let $(A, \mathrm{Gr}, \mathrm{Gl})$ be a glyphed notion of equations. For each finite string $\sigma = \sigma_0 \dots \sigma_n$ in $A^{<\mathbb{N}}$, let $\mathrm{Gl}(\sigma) \in \mathscr{P}(\mathbb{R} \times [0,1])$ be the union

$$\mathrm{Gl}(\sigma_0)_{\to 0} \cup \dots \cup \mathrm{Gl}(\sigma_n)_{\to n}.$$

Thus $\mathrm{Gl}(\sigma)$ is the result of writing σ out in the xy-plane, using the glyphs of A and writing from left to right starting at the origin.

Definition 4. Given a glyphed notion $(A, \mathrm{Gr}, \mathrm{Gl})$ of equations, a *self-graphing equation* is a string $\sigma \in A^{<\mathbb{N}}$ such that $\mathrm{Gr}(\sigma) = \mathrm{Gl}(\sigma)$.

Disregarding canonicity, one can arbitrarily choose a glyphed notion of equations and go to work. Say that σ is *weakly self-graphing* if $\mathrm{Gr}(\sigma)$ is a translation of $\mathrm{Gl}(\sigma)$ after rescaling some symbols. With enough arbitrary choices, constructing weakly self-graphing equations is doable. Jakub Trávník found [7] such an equation (or rather, an inequality) in three evenings. Tupper claims [9] such an inequality with arithmetic, floors, moduli, absolute values, and ≈1800 decimal digits (see Figure 2). Would 500 digits suffice? Are absolute values necessary? (Incidentally, it was a simpler inequality of Tupper's, from [8] or [5], that went viral. Despite the name *Tupper's self-referential formula*, it is not self-referential. Its graph includes all possible $j \times 17$ bitmaps, and the formula is merely one. Tupper's actual self-referentials [9] are less known.)

Figure 2: An inequality (and its graph, rearranged) due to Jeff Tupper (used with his kind permission).

2 A Constructive Existence Proof

For self-containment, we will informally describe some computability theoretical notions. For rigorous details see [1], [6], or any computability theory textbook. Casually, a *Turing machine* is a finite set of states and rules according to which an automaton reads and writes values (from some alphabet) on an infinitely long tape. Mike Davey recently filmed a physical model of Turing machines [3]. Such machines can do any computation that can be described by an effective, unambiguous algorithm. This computational power is independent of arbitrary choices such as the particular alphabet printed on the tape. In the following definition we exploit that alphabetic freedom.

Definition 5. (See Figure 3) Let A be a finite nonempty alphabet, $\_ \notin A$ a *blank symbol*. An *A-Turing machine* is a Turing machine over alphabet $A \cup \{\_\}$. We fix an effective ordering of the A-Turing machines and write T_n^A for the nth one. For each n, φ_n^A denotes the following function. The domain of φ_n^A is $\{\sigma \in A^{<\mathbb{N}} : T_n^A \text{ halts on input } \sigma\}$. For $\sigma \in \mathrm{domain}(\varphi_n^A)$, $\varphi_n^A(\sigma) \in A^{<\mathbb{N}}$ is the output of T_n^A on input σ. A function $f : \subseteq A^{<\mathbb{N}} \to A^{<\mathbb{N}}$ (i.e., $f : X \to A^{<\mathbb{N}}$ for some $X \subseteq A^{<\mathbb{N}}$) is *computable* if $f = \varphi_n^A$ for some n (we call n a *code* for f), and f is *total computable* if in addition $\mathrm{domain}(f) = A^{<\mathbb{N}}$. The notions of computability and total computability extend to functions $f : \subseteq \mathbb{N} \to A^{<\mathbb{N}}$ and $f : \subseteq \mathbb{N} \to \mathbb{N}$ by identifying $\mathbb{N}$ with $A^{<\mathbb{N}}$ via some canonical bijection.

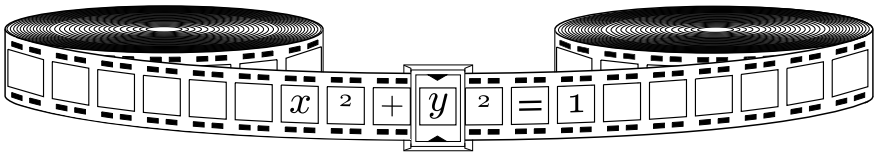

Figure 3: An A-Turing machine in action.

We will show the following condition is sufficient for existence of self-graphing equations.

Definition 6. A glyphed notion $\mathscr{A} = (A, \mathrm{Gr}, \mathrm{Gl})$ of equations is *Turing complete* if there is a computable (i.e. given by an effective non-ambiguous algorithm) function $f : \mathbb{N} \to A^{<\mathbb{N}}$ such that $\forall n \in \mathbb{N}$, if T_n^A halts on input $\emptyset$ then $\mathrm{Gr}(f(n)) = \mathrm{Gl}(\varphi_n^A(\emptyset))$. (If T_n^A runs forever on input $\emptyset$, the output of $f(n)$ is unimportant.)

Thus, $\mathscr{A}$ is Turing complete if there is an algorithm that takes (the index of) an A-Turing machine T_n^A and produces an equation $f(n)$ whose graph is the output of T_n^A if T_n^A halts when run on empty input (if not, $f(n)$ can be anything, but must be defined). Hereafter, when we say that a machine *halts*, we mean it halts on empty input.

There do exist Turing complete glyphed notions of equations not too far from our intuition of Cartesian equations. In Section 3 we sketch an argument that such a notion is provided by basic arithmetic, decimal digits, absolute values, and infinite series or products.

Theorem 1. For any Turing complete glyphed notion $\mathscr{A} = (A, \mathrm{Gr}, \mathrm{Gl})$, there is a self-graphing equation.

To prove Theorem 1 we will use two theorems from computability theory.

Informally, one thinks of algorithms as being made of very simple instructions like "Add 1 to x" or "Goto line 20". Glossed informally, computability theory's *Smn Theorem* says that it is safe for algorithms to include complex meta-instructions of the form "Let x be (a numerical encoding of) the following algorithm: ...". To avoid detour, we only formally state one special case. Loosely speaking, if we have an algorithm $\mathcal{A}_f$ for $f\colon \subseteq \mathbb{N} \to A^{<\mathbb{N}}$, then we can form the algorithm in Figure 4.

1. Take input n.
2. Let x be (a code of) the algorithm:
   1'. Take input m.
   2'. Ignore m. Run $\mathcal{A}_f$ on input n.
3. Output x.

Figure 4: Pseudocode for a special case of the *Smn* Theorem.

Lemma 2. (The *Smn* Theorem, special case) For any total computable $f\colon \mathbb{N} \to A^{<\mathbb{N}}$, there is a total computable $F\colon \mathbb{N} \to \mathbb{N}$ such that $\forall n$, $\varphi^A_{F(n)}(\emptyset) = f(n)$.

We will also use the *Recursion Theorem*, surely one of computability theory's most treasured jewels. Informally, it says it is safe for algorithms to include self-referential meta-instructions of the form "Let x be a numerical code for *this* algorithm."

Lemma 3. (The Recursion Theorem) For any total computable $F\colon \mathbb{N} \to \mathbb{N}$, there is some $n \in \mathbb{N}$ such that $\varphi^A_n = \varphi^A_{F(n)}$.

Note that the function φ^A_n given by Lemma 3 is not necessarily total. A fun informal way to see this is as follows. Using the informal interpretations of Lemmas 2 and 3, there is a function $\varphi^A_n\colon \subseteq A^{<\mathbb{N}} \to A^{<\mathbb{N}}$ (given by Lemma 3) with the following pseudocode: "Take input m. Ignore m. Let x be a code for *this* algorithm. Simulate T^A_x on empty input. If it halts with output y, then output the next A-string after y lexicographically." What is $\varphi^A_n(\emptyset)$? If it exists, it is the next string after $\varphi^A_n(\emptyset)$. Impossible.

Proof of Theorem 1. Since $\mathscr{A}$ is Turing complete, there is a total computable $f\colon \mathbb{N} \to A^{<\mathbb{N}}$ such that $\forall n \in \mathbb{N}$, if $\varphi^A_n(\emptyset)$ is defined then $\mathrm{Gr}(f(n)) = \mathrm{Gl}(\varphi^A_n(\emptyset))$.

By Lemma 2, let $F\colon \mathbb{N} \to \mathbb{N}$ be total computable such that $\forall n \in \mathbb{N}$, $\varphi^A_{F(n)}(\emptyset) = f(n)$. By Lemma 3, there is some $n \in \mathbb{N}$ such that $\varphi^A_n = \varphi^A_{F(n)}$. In particular,

$$\varphi^A_n(\emptyset) = \varphi^A_{F(n)}(\emptyset) = f(n) \text{ is defined. } (*)$$

Let $\sigma = f(n)$, then

$$\begin{aligned} \mathrm{Gr}(\sigma) &= \mathrm{Gr}(f(n)) \\ &= \mathrm{Gl}(\varphi^A_n(\emptyset)) && (\text{By } * \text{ and choice of } f) \\ &= \mathrm{Gl}(f(n)) && (\text{By } *) \\ &= \mathrm{Gl}(\sigma). \end{aligned}$$

By Definition 4, σ is a self-graphing equation. □

Theorem 1 is constructive up to applying the *Smn* Theorem and the Recursion Theorem. It turns out that these two computability theoretical theorems are themselves fully constructive, making Theorem 1 fully constructive as well. In principle, one could design an algorithm to take specifications of Turing complete glyphed notions of equations and mechanically generate self-graphing equations from them.

3 A Turing Complete Notion of Equations

Let $A = \{a_1, \ldots, a_N\}$ be a finite alphabet with symbols for arithmetic, decimal digits, absolute values, and infinite series or products. For $\sigma \in A^{<\mathbb{N}}$, if σ is a grammatically correct xy-equation, let $\mathrm{Gr}(\sigma)$ be its graph in the ordinary sense. For grammatically incorrect σ, let $\mathrm{Gr}(\sigma)$ be some nonempty subset (e.g. an error

message) of the *left* half-plane, this will ensure Theorem 1 produces a grammatically correct equation. The graph of

$$(x - |x|)^2 + (1 - x - |1 - x|)^2 = 0$$

is the line segment from $(0, 0)$ to $(1, 0)$; using this it is easy to find equations for line segments with rational endpoints. Such line segments can be combined to design glyphs (based on typography) such that $\forall a \in A$ there is an equation $L_a(x, y) \in A^{<\mathbb{N}}$ (not involving infinite series or products) with $\mathrm{Gr}(L_a(x, y)) = \mathrm{Gl}(a)$.

Arithmetic can be used to simulate propositional logic: $xy = 0$ if and only if $x = 0$ or $y = 0$; $x^2 + y^2 = 0$ if and only if $x = 0$ and $y = 0$; and (see [4]) $0^{x^2} = 0$ if and only if $x \neq 0$. Infinite series or products can simulate logical quantifiers ranging over $\mathbb{N}$, for example

$$\prod_{i=0}^{\infty} 0^{f(i)^2} = \begin{cases} 1 \text{ if } \forall x \in \mathbb{N},\ f(x) = 0, \\ 0 \text{ otherwise.} \end{cases}$$

To see $(A, \mathrm{Gr}, \mathrm{Gl})$ is Turing complete, we must convince ourselves there is an algorithm that, given $n \in \mathbb{N}$, produces an equation $f(n) \in A^{<\mathbb{N}}$ such that if T_n^A halts then $\mathrm{Gr}(f(n)) = \mathrm{Gl}(\varphi_n^A(\emptyset))$. We can say more: there is a fixed equation $f_0 \in (A \cup \{\mathbf{n}\})^{<\mathbb{N}}$ (where $\mathbf{n}$ is a special new symbol) such that if we define $f(n)$ to be the result of replacing $\mathbf{n}$ by (the decimal digits for) n in f_0, then f works. By the previous paragraph, it suffices to find a logical formula $\theta(x, y, \mathbf{n})$ in the language of arithmetic and absolute values, with all quantifiers ranging over $\mathbb{N}$, such that for all $x, y \in \mathbb{R}$ and $n \in \mathbb{N}$, $\theta(x, y, n)$ is true if and only if T_n^A halts and $(x, y) \in \mathrm{Gl}(\varphi_n^A(\emptyset))$. Such a formula is as follows (we use $\bigvee_{k=1}^{N} \psi(k)$ to abbreviate "$\psi(1)$ or ... or $\psi(N)$"):

$$\exists m \in \mathbb{N} \text{ such that } T_{\mathbf{n}}^A \text{ halts after} \leq m \text{ steps and } \exists \ell \in \mathbb{N} \text{ such that } \bigvee_{k=1}^{N} \Big(a_k \text{ is the } \ell\text{th symbol of } \varphi_{\mathbf{n}}^A(\emptyset), \text{ and } L_{a_k}(x - \ell, y) \text{ holds [i.e., } (x, y) \text{ lies in } \mathrm{Gl}(a_k)_{\to \ell}] \Big).$$

That the subclauses "$T_{\mathbf{n}}^A$ halts after $\leq m$ steps" and "a_k is the ℓth symbol of $\varphi_{\mathbf{n}}^A(\emptyset)$" can really be formalized with only arithmetic, absolute values, and quantifiers ranging over $\mathbb{N}$, follows from a course in first-order mathematical logic [1].

References

[1] S. Bilaniuk, *A Problem Course in Mathematical Logic*, self-published, 1991, available at `http://euclid.trentu.ca/math/sb/pcml/pcml-16.pdf`.

[2] J. Borwein, R. Crandall, Closed Forms: What They Are and Why We Care, *Notices Amer. Math. Soc.* **60** (2013) 50–65.

[3] M. Davey, A Turing Machine, Webpage published in March 2010 (accessed 30 June 2013), available at `http://aturingmachine.com/`

[4] D. Knuth, Two notes on notation, *Amer. Math. Monthly* **99** (1992) 403–422.

[5] Self-answering Problems, *Math Horizons* **13** (2006) 19.

[6] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing, 1997.

[7] J. Trávník, Self Referential Formula in Math. Webpage published 23 Jun 2011 (accessed 6 June 2013), available at `http://jtra.cz/stuff/essays/math-self-reference/index.html`

[8] J. Tupper, Reliable Two-Dimensional Graphing Methods for Mathematical Formulae with Two Free Variables, in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 2001, 77–86.

[9] J. Tupper, Index of /selfplot. Webpage (accessed 6 June 2013), available at `http://www.peda.com/selfplot/`