The Iterated Romance of Syntax

Capturing the Compositional Dynamics of Polynomials In Syntax

Salvador Guzman

April 16, 2023

Abstract

One of the greatest intellectual crimes to beset us in the 20th-century was the premature death of the formalist program. The millennia old dream of *solving math* was never realized as our efforts fell short of our ambition. David Hilbert, with all his grace, his effulgent brilliance, his professional magnanimity, and his unflinching dedication, was left with only disgruntled disappointment. The formalist program was a noble effort, held aloft by the unyielding conviction and charisma of the foremost mathematicians of the age. This forlorn vignette is rendered at least somewhat emotionally digestible by the developments that followed. The dream of **syntax is all** became partially realized by the contributions of Haskell Curry, Alonzo Church, Stephen Kleene, Moses Schönfinkel and others. With their syntactic approach to mathematics, we capture the compositional beauty of infinity in our humble symbol. And thus, we compile the syntactic face of God.

Contents

1 The Tragedy of Formalism

The death of the formalism program was a necessary tragedy for the advancement of mathematics. Gone were the ensconced notions were the naive intuitions of mathematical computation. Rather, it was the lack of such a computational foundation for mathematics that needed to be addressed. It was around this time that mathematicians began to rigorously define what it means for a statement to be provable. Without a computational foundation, the notion of provability rested on a tentative and ambiguous folklore. For this heroic death, we inherited a proliferation of computable logical systems that could be used to encode mathematical ideas and theorems. The program's death died for our crass naivite and ensured the salvation of mathematics for the modern age. It is here where we demark the beginning of computation.

1.1 Poetry of Computation

The poetry of computation is the poetry of the formalist program reborn. Whether it be lambda calculus or combinatory logic, the accentuation of syntax is what deliberately marks this change in tradition. What Frege set in motion is that the story of math and logic is in part the story of syntax. And it is with syntax that we seek to capture the semantics of mathematics. Whether we behold the history of mathematical symbols establishing what we associate with the mathematical arts, or modern compilers that bleed out the captured semantics of a language to a hardware equivalent, syntax is the poetry of computation. What the formalist program lacked in ultimate success, it made up for in intuition and beauty.

1.2 A Nightmare Embraced

Of course, the program is not without its thorns. To assume that the program must be a feasible one is to posit that mathematics is devoid of content beyond its superficial and meaningless syntax. The inevitable progression of this species of thought is that mathematics is a mere syntactic game. Any semantic model we attach to this game must submit to the reality of being mere fiction. And if this holds, then our notions of mathematics fall into the void of nihilism and thus embrace the nightmare of extinction. This nightmare becomes a wonderful dream only when coupled with a more human focused approach to mathematics, such as intuitionism. I won't elaborate on what this latter program entails suffice to say that it dulls the sting of the void. Once we compliment our syntax program with a phenomenological narcotic, we can commandeer the development of mathematics where the syntax demands and inject our semantic creature comforts as an afterthought. After all, mathematics is a inductive, deductive and analytical art; there is no room for a bleeding heart.

1.3 A Gentle Promenade Down The Road to Perdition

Let us follow this dreamy reverie down the whirlpool of damnation. Retaining the insight that our formalist program may not condemn us to the void by a serendipitous pairing to some phenomenological strand of thought. I am not concerned what flavor or brand of

special pleading is provisioned. Consider the program plead. Now, envision if you might that the program were not doomed to the ash bin of history. In this alternative hypothetical universe, the program succeeds. We have successfully reduced all of mathematics up to that point to pure symbolic function of syntax. Semantics thus submits readily to capture by the mystical symbol. The program is a success. An element of this universe is in substance some ready made mathematical object that computes algebra and reduces it to theorems of our liking. Our appetite for conclusive proofs and pure deductive reasoning is sated. That is the dream. Well, if I have not belabored enough (forgive me!), I modestly posit that claims of the death of this program are of mild exaggeration. The program is not dead, it is merely gone dormant while our syntactic tools sharpened.

2 Building A Sea-fairing Vessel out of a Corpse

Dear reader, may you forgive me for my crass verbosity. I fear it is the only way my sinister little thoughts can find their way to a wider audience. Consider it a concession well spent. I do indeed hold that the purview of the formalist institution continues full steam, unencumbered by the romance of larger narratives. Again, I outlined in cursory detail the different syntactic program that explored what is possible with theory. As maligned as syntax is as a foundation for mathematics, it has not heeded the call to abolish itself out of existence. Progress, once tasted, is not easily relinquished. What has been thwarted by historical account conclusively is the naive version of the program. By necessity, any research that follows in the aftermath must deviate in substance from the initial thrust of the program. The impetus of critique imparts a degree of resiliency to subsequent efforts. And it is these efforts that carry the banner of formalism, notwithstanding the more judicious features.

2.1 The Naive Intuition of the Initial Program

Why did the program fail? Did it succumb to the bleeding that sparring with intuitionism? Probably not as both programs are rather dead in their initial manifestation. I hold the notion that the intuition behind each of these efforts complimented each other pleasantly and should not have been competitors for our hearts and minds. But such is life when a post-mortem autopsy yields two corpses instead one. It is an incident of history I feel that David Hilbert triumphed and that he dueled the other school in the first place. The syntactic approach of Hilbert was a more pragmatic one. It focused on the material conditions in which mathematics was practiced and developed. Syntax was the conduit for his program because syntax was what conveyed the semantic information of mathematics. To Hilbert, analysis of the syntax, and thus the grammar of mathematics, was the key to understanding the exposition of a computable basis. It is not to say that Hilbert believed mathematics to be the mindless churning of symbols and syntax was the heart of mathematics. But in the pliant concession of this profile of the art do we extract the possible essence of a feasible program. In the rotten carcass of its demise, do we find brilliant gems of insight that can be salvaged for future use.

2.2 Rescinding The Full-throated Ambition of Formalism

Doomed ambition need not be a death sentence and even carrion can provide a cozy abode if for maggots alone. What precluded the longevity of the enterprise is the pallid condition of research into syntax and formal languages. Contemporaries would have to wait several decades until the illustrious formal studies of grammars and languages produced more solid foundations. Hilbert himself must have known this to some degree. His questions had a rather informal air to them and left as an open question whether new mathematical machinery would be needed to satisfy his demands. Given how much research was put into syntax, it would be a daunting task if he knew how much literature it would spawn. To wit: lambda calculus, combinatory logic, type theory, category theory and so on. These are a few syntax-related or algebraic-focused approaches to studying mathematics. The list is not exhaustive and the future is sure to spawn more. Thus, it is here that we find our departure of his program, respectfully pay homage to this effort, offer eulogies and then move on to greener pastures.

2.3 Seduction of Intuitionism

While ultimately vindicated by the promulgation of later research, intuitionism was dealt quite a fatal blow. Hilbert's words spoken against intuitionism proved potent vigour against it. His champion of "more results", not less, was the rhetoric of the day. I hope in our age, we see the flaws in such reasoning. Intuitionism hoped to bring mathematics to us and create a more constructive substrate to leverage for the communicating of mathematical reasoning. I admit to prejudice hearing it from Hilbert himself and saw myself as opposed out of principal such aspirational goals. But merit of extinguishing the practice of mathematicians, as practiced for millennia, was at the time not compelling. Strange it is today, to this humble observer, that the argument of ipso facto posteriority would rule the day in discussion of advancing technology. Do we say today that new programming languages should not be fashioned to not eclipsed the interest of previous languages and programmers? Outrages! But such was the time and such was curling of the monkey paw dealing intuitionism a dishonorable discharge from the ranks of mathematics.

3 Ambitions Laid Bare

So what is it that I seek? Thus far we have arrived at two dead ends and no light to elucidate our dark path. Have we fully exhausted our resolve? Of course not. What we are now left is the raison d'etre of this document. With everything I have said in heart and also with the motivations contained in the initial undertakings, I now present my own ambitions for the future of mathematics. I want a syntactic basis for mathematics. I want this to reduce the development of this field as a matter of logical consequence extrapolated from the choice of syntax. I also believe wholeheartedly the intuitionism utopia of a constructive and human-digestible vantage point for tackling analysis. It is in the spiritual synthesis in these things that some modicum of genius is found. But I will leave that enterprise for some future date. Suffice for now is my own modest ambitions.

3.1 Princess in the Tower of Algebra

It is my strong conviction that algebra as it is understood in mathematics is substitute for other concepts. Algebra as a principal captures structure, symmetry and dynamics of a system of objects. It is readily amenable to analysis by syntax since formalism is indeed syntax itself. It is very tempting to transgress the boundary of this observation and claim that algebra is only syntax. I will leave that transgression for some other day. For now, let's assume that algebra and algebras are pliant things and syntax is no censurious approach to understanding them. What I am wrestling with, dear reader, is generalizing the notion of algebra to a more general notion of structure permitting computation. We have an algebra of our beloved objects and can with exceeding ease compute properties with these systems. What if we had a algebra for everything? What if we had an algebra for generating other algebras? That licentious thought must sadly be put to rest for now. But I hope the temptation of such a beauty will stay with you and haunt your languid summer days.

3.2 Outline of my Ambitions

My train of thought lead me to extrapolate from syntactic principals to more general laws governing the dynamics of algebraic objects. I am effervescent with excitement to share what I have found! My personal research had found purchase in implementing syntactic methods for polynomials. Predilection proper of mine lead me to pursue a novel approach to computing the closed form of iterated polynomials. Knowing the particular polynomial function from the start yielding the closed form. In outline and part, here follows my general line of reasoning.

1. Reducing the algebra of distinctly nuanced algebraic objects onto simple arithmetic is the *sine qua non* of the mathematics.

2. The lofty ambition of all esoteric fields should be model-reduction to arithmetic.

3. Whatever wanton discretion lead to the promulgation of arcane nomenclature and formalism must collapse to the imperative of simplicity.

4. Nothing is simpler than the immeasurable simplicity of numbers and nothing is more understood.

5. The goal of all mathematics should be the reduction of complicated objects to an algebra that is isomorphic to the natural numbers.

And with that prelude I outline my method for reduction of iterated polynomials.

3.3 Polynomials as a Testing Ground

As I have belabored to death at this point, my choice of polynomials as the testing ground for my syntactic approach is not particularly stimulating. I picked a well understood object to play with. Nothing beyond a superficial inclination on my part can justify this choice. While my approach can be said to yield tangible results in its application to polynomials, it

is left as an open question whether a just approach formulated for different class of objects will yield similar productivity. I believe it will but have no desire to prove it. That will take more quiet research on my part before I am ready to divulge any success in other areas.

3.4 Approach Proper

Let the following definition be the starting point of our discussion.

$$f(x) = ax^2 + bx + c, \quad f : \mathbb{R} \to \mathbb{R} \wedge a, b, c \in \mathbb{R} \tag{1}$$

Supposed we wanted to find the closed form of $f(x)$ for n iterations. That would be quite an ordeal. While easily expressible symbolically, $f(x, n)$ where $n \in \mathbb{N}$, the prospect of computing the closed form is of a degree more onerous. For that reason, we will start simpler monomials to represent the undertaking in a more accommodating context. Let the following three monomials guide our discourse.

$$f_1(x) = a_1, \quad f_1 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R} \tag{2}$$

$$f_2(x) = a_2 \cdot x, \quad f_2 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R} \tag{3}$$

$$f_3(x) = a_3 \cdot x^2, \quad f_3 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R} \tag{4}$$

Monomials, by their simple structure, can easily yield a closed form through modest effort. The following are the closed forms of the monomials.

$$\begin{aligned} f_1(x) = a_1, \quad f_1 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R} \\ f_1(x, n) = a_1 \end{aligned} \tag{5}$$

$$\begin{aligned} f_2(x) = a_2 x, \quad f_2 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R} \\ f_2(x, n) = a_2 \cdot f_2(x, n-1) \\ f_2(x, n) = {a_2}^n \cdot x \end{aligned} \tag{6}$$

$$\begin{aligned} f_3(x) = a_3 \cdot x^2, \quad f_3 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R} \\ f_3(x, n) = a_3 \cdot f_3(x, n-1) \\ f_3(x, n) = {a_3}^{2n-1} \cdot x^{2n} \end{aligned} \tag{7}$$

Beauty is concise. And if polynomials behaved accordingly, closed forms would be easier to compute. But alas, polynomials are not monomials. We can still extract general tendencies by perusing the pattern of the closed forms. The coefficients for example become parameters in the final form. They are fed through the iterated function and come out with just as

iterated as the variable x proper. This is so for amenable to our normal intuition and does not deviated from any of our expectations. What polynomials have in store for us however complicate the iterative story.

Scaffolding ontop of the simple monomial substrate is the vexing existence of addition. I have much theory and philosophy to dissert on the nature of addition, for example how addition is semantic equivalent to append by way of a list; for now, we keep in mind that addition is not a number. Expressed formally, addition is,

$$+ : \mathbb{P}^{\alpha} \times \mathbb{P}^{\beta} \to \mathbb{P}^{\gamma}, \quad \alpha, \beta, \gamma \in \mathbb{N} \wedge \gamma \in \{\alpha, \beta\} \tag{8}$$

In this context, we are concerned with addition as a binary operation on monomials that returns another monomial. The degree of the monomial is the largest of the two operands.

Naturally, we would seek to find some isomorphism of addition to numbers. Set more informally, we would like to treat addition as operated by iteration the way numbers are operated by 1-arity functions. There is a rich field of theory targeting this intuition. Operator theory seeks to understand the structure and dynamics of functions when applied to an function. The operand function is understood to be a member of some space of functions and operator function, called an operator, is understood to be the mechanism by which the operand is mapped to another space. That is all I will say about that now, suffice to add there is beauty in this approach.

What I want to accomplish is even more basic than operators and function spaces. I proclaim that there is a tendency of tracking the evolution of a binary operation across iterations in the same way that there is a tacit pattern for numbers. Let's think about this with solid examples. Let the following be our starting points.

$$n_1 \in \mathbb{N} \tag{9}$$

$$\lambda_1 : \mathbb{N} \to \mathbb{N} \tag{10}$$

Given a simple equation like λ_1, application of this function n_1 can be symbolically expressed with $\lambda_1(n_1)$. Applying the inverse would like this: $\lambda_1^{-1}(\lambda_1(n_1))$. The inverse is the function that when applied to the result of λ_1 yields n_1. The inverse is λ_1^{-1}. The full syntax for applying the inverse is,

$$(\lambda_1^{-1} \circ \lambda_1) \quad n_1 \tag{11}$$

Beautiful. This yields our original operand, n_1. Supposed we wanted to analyze the type signature of the composed object. Nevermind that the composition of an inverse with the original would annihilate itself to the identity. Here it is.

$$(\lambda_1^{-1} \circ \lambda_1) : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \tag{12}$$

I brought up the notion of type because by this point, its mention seems more than proper. In building the type up from the ground, we must keep in mind that the inverse takes a function that already has its own type. What 12 demonstrates is that we can easily feed a function another function and retrieve a semantically coherent interpretation as a result. Perchance we can provide some similar resolution to the problem of addition.

The first thing we would do is curry the function. This is a common practice in functional programming to reduce a function of multiple arity to a series of functions of arity 1. The following is the curried form of addition.

$$+ : \mathbb{P}^{\alpha} \to \mathbb{P}^{\beta} \to \mathbb{P}^{\gamma}, \quad \alpha, \beta, \gamma \in \mathbb{N} \wedge \gamma \in \{\alpha, \beta\} \tag{13}$$

Again, let's take a moment to appreciate the beauty of this. There is an intrinsic aesthetic quality to symbolically manipulating functions and objects in general. However, we must part ways somewhat with this enthusiastic talk of beauty. While we the symbols we have accumulated do indeed express the definition properly, we are still far away from a concrete platform for rendering iterated forms. Do not confuse symbolic semantic capture with functional mechanism of achieving the computational result.

How can we leverage the curried form of addition to achieve our goal? Well, it helps think of mathematical expressions as a sequence of operations performed on some operand. Presented symbolically, we can express this as,

$$\phi : \Gamma_{\alpha} \to \Gamma_{\omega} \tag{14}$$

Think of *phi* as a full-fledged program that a programmer would write up to achieve some task. The set Γ_{α} captures the input to the program and the set Γ_{ω} captures the output. Specifying the individual steps as a sequence of operations, the program can be expressed as,

$$\phi = \phi_1 \circ \phi_2 \circ \phi_3 \circ \cdots \circ \phi_n \tag{15}$$

The type signature of the program is as follows,

$$\Gamma_{\alpha} = \Gamma_1 \tag{16}$$

$$\Gamma_{\omega} = \Gamma_2 \tag{17}$$

$$\phi_1 : \Gamma_1 \to \Gamma_2 \tag{18}$$

$$\phi_2 : \Gamma_2 \to \Gamma_3 \tag{19}$$

$$\phi_3 : \Gamma_3 \to \Gamma_4 \tag{20}$$

$$\phi_n : \Gamma_n \to \Gamma_{n+1} \tag{21}$$

$$\phi : \Gamma_1 \to \Gamma_2 \to \Gamma_3 \to \cdots \to \Gamma_{n-2} \to \Gamma_{n-1} \to \Gamma_n \tag{22}$$

$$\phi : \Gamma_{\alpha} \to \Gamma_2 \to \Gamma_3 \to \cdots \to \Gamma_{n-1} \to \Gamma_n \to \Gamma_{\omega} \tag{23}$$

The composition of functions is associative and computation can begin at any point and spread outwards.

Gorgeous. A function as a sequence of types is useful for breaking down an operation to the metaphorical sum of its parts. We want to be able to do this to polynomials. Not just in their structured operational form but in their iterative dynamic one as well. Given 7, we can express the iteration by following the type signature of the incremental iterations. Starting from 7,

$$f_3(x) = a_3 \cdot x^2, \quad f_3 : \mathbb{R} \to \mathbb{R} \wedge a \in \mathbb{R}$$

$$f_3(x, n) = a_3 \cdot f_3(x, n-1)$$

$$f_3(x, n) = {a_3}^{2n-1} \cdot x^{2n}$$

$$f_3(x, 0) = x \tag{24}$$

$$f_3(x, 1) = a_3 \cdot x^2 \tag{25}$$

$$f_3(x, 2) = a_3^3 \cdot x^4 \tag{26}$$

$$f_3(x, 3) = a_3^5 \cdot x^6 \tag{27}$$

Let's take do proper accounting of the degree of the polynomial across iterations.

$$f_3 \in \mathbb{P}^2 \tag{28}$$

$$f_3(x, 1) = f_3 \tag{29}$$

$$f_3(x, 2) \in \mathbb{P}^4 \tag{30}$$

$$f_3(x, 3) \in \mathbb{P}^6 \tag{31}$$

Expressed as a sequence of operations, the iteration of monomial becomes,

$$f_3(x, n) \in \mathbb{P}^{2n} \tag{32}$$

Let's define a helper type signature to aid in writing down the sequence type signature.

$$\Delta : \mathbb{T} \times \mathbb{N} \to \mathbb{T}^n \tag{33}$$

Δ is a type function that accepts some type, from the set of types $\mathbb{T}$, and a natural number. I will use this to produce the type of degree of the polynomial.

$$f_3(f_3, n) : \Delta(\mathbb{P}, 2) \to \Delta(\mathbb{P}, 4) \to \cdots \to \Delta(\mathbb{P}, 2n-2) \to \Delta(\mathbb{P}, 2n) \tag{34}$$

Note that the type signature of the polynomial holds even for polynomials. While the example I used was for a simple $\mathbb{P}^2$ monomial, the same type holds for any polynomial. This is because the degree of the polynomial is purely a function of the number of iterations and the highest degree of the polynomial. So it seems we made some progress. We can now express the type signature for a polynomial as a sequence of operations.

Thus, given some arbitrary polynomial of degree n, we can express the type signature as,

$$f_n(f_n, m) : \Delta(\mathbb{P}, n) \to \Delta(\mathbb{P}, n^m) \tag{35}$$

Meditate on the symbolic elegance of such a statement. We have determined what the type of an iterated polynomial is by pure symbolic expression of its own degree and number of iterations. That is a beautiful thing to behold but we still do not have a way to express the actual computation still.

Now, with all this build up, I think we are ready to share the secret. Finally, now is the time to reveal the Copernican Revolution in Syntax. Much in the spirit of Gödel's numbering scheme, we can assign a number to each of our operations that appear in a polynomial. The operations in a polynomial are addition, multiplication, and exponent. Rather than symbolically numbering it based on where the symbols for each operation appears, we will assign a scheme ourselves. Quite frankly, we don't care about the operations syntactically. We know they are contained somewhere in the expression. They key is not so much the position but the relation of each of the operation to the one another.

The technique is to apply an ordering to the operations. The order should not be confused with order of operations. While it is related, I care more about reducing the overhead of representing and computing each operation. I would like to reduce all three operations to pure symbolic function of computing a more primitive composite. Let the following definition be our guiding light.

$$\mathbb{O} = \{+, \cdot, ^\circ\} \tag{36}$$

$$\pi(0) = + \tag{37}$$

$$\pi(1) = \cdot \tag{38}$$

$$\pi(2) = ^\cdot \tag{39}$$

$$\pi(0, 1) = + \tag{40}$$

$$\pi(0, 2) = + \circ + = \pi(0) \circ \pi(0) \tag{41}$$

$$\pi(1, 0) = + \tag{42}$$

$$\pi(1,1) = \cdot \tag{43}$$

$$\pi(1,2) = \cdot \circ + = \pi(1) \circ \pi(1) \tag{44}$$

$$\pi(1,n) = \pi_1(1) \circ \pi_2(1) \circ \cdots \circ \pi_{n-1}(1) \circ \pi_n(1) \tag{45}$$

This is all fine and good. I have sequenced the operations in a given order. This particular should not require much justification. The "smaller" operation of addition is at the bottom and the "larger" operation of exponentiation is the biggest one. All I did is number them starting 0 and incrementing by 1. The function π is a function that returns a function. Think of it has a convenient hash to store the operations in a given order. When we add n as second operand to π, we are adding the ability to iterate our operation.

Now here comes the real magic. Here it is.

$$\pi(i) = \mathbb{O}_{i=0}^{n_i} \pi(i-1) \tag{46}$$

This equation is my contribution. Let me explain the vague terms and then I will continue to explain the equation.

1. $\pi(i)$ is a function that returns a function.
2. i is a natural number used to index π.
3. $\mathbb{O}$ is my ad hoc way of composing some set of operations. In this example, I am using to compose a function with itself
4. n_i is the number of times to compose the function with itself. What n_i is not important. It is a natural number that determines how many times to compose the function with itself. And the number defines how many iterations are needed to "reach" the next operation in the sequence.
5. $\pi(i-1)$ is the function that is being composed with itself. Specifically, the previous function in the ordering is being iterated to yield the next operator in the sequence.

While it is rough in some ways, I think I have achieved something significant. Behold my gift to you. It is this equation and symbols.

The idea behind it is that any operations that we are currently studying can be put in some ordering. The order can be any you desire but should primarily reflect some intuitive notion of the "size" and ranking of the operation. Ultimately, we are the arbiter of our symbols. We assign at our discretion and compute at our leisure. A comfortable arrangement can similarly be made with any class of algebraic objects. Now lets apply our little trick. To a simple polynomial of degree 1. Let us play with the following definition.

$$g(x) = 2 \cdot x + 3 \tag{47}$$

$$g(x) = \pi(0)(\pi(1)(2,x),3) \tag{48}$$

$$g(x,0) = x \tag{49}$$

$$g(x,1) = g(x) = \pi(0)(\pi(1)(2,x),3) \tag{50}$$

$$g(x,2) = g(g(x)) = \pi(0)(\pi(1)(2,x),3) \circ \pi(0)(\pi(1)(2,x),3) \tag{51}$$

Note that the general difficulty of this problem, the problem of finding a closed-form for iterating polynomials, comes from the onerous requirement of having to treat a polynomial at one point as a noun and operand and simultaneously as a verb and operator. This is the crux of the problem.

We need to make another adjustment to be able to overcome a hurdle. You can see in 51 we almost have a streamlined account of pure function composition. If it weren't those pesky numbers, we would have a pure sequence. Well, for the context of what we are trying to accomplish, we don't care what the particular non-function operand is. All we care is the particular ordering of the sequence. So let us make a small adjustment to our definition. Not only are we going to curry our functions again, we will abstract away wherever we see a number. I justification for this is that numbers are terminal nodes in our computation tree. We follow the thread of execution until we reach a number; otherwise, we thread into another function to compute the subtree. So the new definition is,

$$g(x,2) = (g \circ g)x = ((\pi(0) \circ \pi(1)) \circ (\pi(0) \circ \pi(1)))x \tag{52}$$

That is much simpler. We can see the general order of operations as they are applied in the final expression. The expression when computed should yield what the original polynomial would yield, iterated once. Closer and closer do the symbols encroach on a more elegant basis for computation on pure function composition alone.

What is currently thwarting our efforts for simplifying the symbols of composition is the gap that exists between the terms $\pi(0)$ and $\pi(1)$. If we could combine the two terms amongst themselves, we could achieve a modicum of simplification that may improve our chances of computing the final expression. However, we need to know what the dynamics is when we have the composition expression $\pi(0) \circ \pi(1)$. We need to know what the composition of two separate functions is. Can they be reduced? Or better yet, can they be reordered without affecting the final result? In other words, does the algebra for composition allow for reordering? Given that we have constructed and manipulated our symbols at our pleasure, perhaps we can discover some rule that would permit judicious reordering so that like terms can combine. Here is the possible algebraic rule that provides a equitable denouement to our dilemma.

$$\pi(0) \circ \pi(1) = \pi(0) \circ \mathbb{O}_{i=0}^{n_i} \pi(0) \tag{53}$$

$$((\pi(0) \circ \pi(1)) \circ (\pi(0) \circ \pi(1))) = \pi(0) \circ \mathbb{O}_{i=0}^{n_i} \pi(0) \circ \pi(0) \circ \mathbb{O}_{i=0}^{n_i} \pi(0) \tag{54}$$

$$m = 2 \cdot n_i + 2 \tag{55}$$

$$\pi_1(0) \circ \pi_2(0) \circ \cdots \circ \pi_{m-1}(0) \circ \pi_m(0) \tag{56}$$

Using subscript to properly index the function, they can be safely ignored. What we have done is the final composition of a polynomial with multiplication and addition only. Even though we have only referenced two different types of functions, the technique we used to reduce a function of $\pi(1)$ to $\pi(0)$ can be applied to any function $\pi(i), \quad i \in \mathbb{N} \wedge i > 0$. Given a function $\pi(i)$, we can reduce it to $\pi(i-1)$ by 46. Any function can be reduced to a function of $\pi(0)$.

That is quite a quaint development but I rather not have to convert all functions to $\pi(0)$ before I can compute the final expression. Much to unwieldy and for a algebraic technique, it leaves a lot to be desired. Instead, lets focus on what concession we can make to extract a proper reordering to move around immiscible terms.

Why should we want this? Well, naively our intuition is guiding us towards combining like terms so that the final operation is is sequenced by function. First we compute all the functions of $\pi(1)$, then all the functions of $\pi(0)$, and so on. Given our propensity to prefer an ordering that coincides with our intuition, it is rather fortunate that the gods of algebra posit a sort of rank ordering for execution. We compute the "larger" $\pi(i)$ first and then the "smaller" $\pi(i-1)$, and so on. If we had the larger $\pi(2)$, corresponding to exponentiation, we would strive as result of manipulating our algebra to compute it and then passing the result to the smaller $\pi(1)$. Naturally this is what we get.

Let's say we had the following polynomial,

$$f(x) = 2 \cdot x^2 + 2 \cdot x + 1 \tag{57}$$

The previous paragraph should narrowed down our perspective to polynomials as computation. More Specifically, we can think of polynomial as rank-ordered computation of $\pi(i) \quad i \in 0, 1, 2$, and so on. The rank ordering This corresponds to computing all $\pi(2)$ regardless of where they appear in the expression. Of course, given that we are working with polynomials, we know quite clearly where all the $\pi(2)$ are. But the point of this exercise is that their order in situ the expression does not matter. It is their rank ordering that matters. While I lack further justification for why this should be the case, there is something seductive about executing the "larger". functions first.

Before we continue, let me justify the rank order computation. A rich field of study is the generalization of the function. While we can indeed maintain that a function $z(x), \quad x \in \mathbb{R}$ is a function of a single real variable, we not limited by our own pronouncement. What we decide is exclusively left to our own prerogative and what we can do is to our imagination. One can imagine the operand fed to function as some unprocessed, abstract, amorphous object. All that is needed for $z(x)$ to be valid for an algebraic object is that the operational semantics. And since we are the arbiters of all things definitions, we can easily construct a valid interpretations for our own purposes. Now assuming that $z(x)$ is a polynomial, we can create a valid interpretation for objects other than numbers. As mentioned before, there already is a field of study that has capitalized on this opportunity. Some canonical objects that have studied are matrices, vectors, tensors, functions, operators and so on. The amorphous argument is then fed to conveyor belt of operations. While this is a deviation

from our study, a satisfying symbolic sentence that more or less captures the essence of this intuition is the following.

$$z(x) = \alpha_0 \cdot \pi(0) + \alpha_1 \cdot \pi(1) + \alpha_2 \cdot \pi(2) \tag{58}$$

In a perfect world, all operators would be commutative and the forthwith declarations of involved operators would be followed with a simple linear algebraic expression that impresses the vector space of computation. Alas, we do not reside in the perfect paradise quite yet.

Back to our polynomial operator. Ignoring the specific number of invocations for each of the $\pi(i)$ functions, we can write the following expression to specify the ordering of the computation.

$$f(x) = \pi(0) \circ \pi(1) \circ \pi(2) \tag{59}$$

Expressed using our composition summation notation, we can write the following

$$f(x) = \mathbb{O}_{i=2}^{0} \pi(i) \tag{60}$$

This expression terminates our study of the symbolic algebraic expression. We must return to see how if we can derive further algebraic rules for computational primitives. Again, we want to combine like terms. Let's start with a simple polynomial.

$$f(x) = 2 \cdot x + 1 \tag{61}$$

What would the iterative composition of this function look like. The following outlines computing the iteration of $f(x)$ for 1 times.

$$f(x) \circ f(x) \tag{62}$$

$$f(x) \circ (2 \cdot x + 1) \tag{63}$$

$$(2 \cdot x + 1) \circ (2 \cdot x + 1) \tag{64}$$

$$2 \cdot (2 \cdot x + 1) + 1 \tag{65}$$

$$4 \cdot x + 3 \tag{66}$$

A function as both noun and verb is indeed the site contention. Thankfully, the simplicity of this polynomial allows us to derive a single iteration relative easy. Since there are no powers in this expression, any iteration is symbolically a function of addition and multiplication. What would 2 iterations produce?

$$f(x) \circ f(x) \circ f(x) \tag{67}$$

$$f(x) \circ (2 \cdot x + 1) \circ f(x) \tag{68}$$

$$(2 \cdot x + 1) \circ (2 \cdot x + 1) \circ f(x) \tag{69}$$

$$(4 \cdot x + 3) \circ f(x) \tag{70}$$

$$(4 \cdot x + 3) \circ (2 \cdot x + 1) \tag{71}$$

$$4 \cdot (2 \cdot x + 1) + 3 \tag{72}$$

$$8 \cdot x + 7 \tag{73}$$

While pursuing this example to its full logical conclusion is a bit tedious, it is demonstrative of a simple pattern pertaining to our choice of polynomial. The coefficient for the leading term for the nth iteration is 2^n. This makes sense since it leads the expansion of any iteration. Any operand inserted into the polynomial, including another function, will always be multiplied by 2 first. The next term, 1, is the constant term. In this polynomial, we have a simple number acting as the next leading degree of the polynomial. In other more complicated examples, we can imagine the another variable appearing and complicating the story somewhat. However, the logic I am describing still holds. Thus, the constant term for the nth iteration would be $2^n - 1$.

What even this simple example demonstrates is that any algebraic object, in this case a polynomial, can act as a verb and a noun. Iteration of a polynomial renders out the interpretation of the object as necessary. Iteration is the verb operator acting on the noun operator, wherein the verb and the noun are the same object. It is telling that the degree does not change for our example across iterations. The nth iteration is give by,

$$f_1(x) \circ \cdots \circ f_n(x) = 2^n \cdot x + (2^n - 1) \tag{74}$$

Now let's transmogrify our polynomial into a more general version of a first degree polynomial. Let's take,

$$\theta(x) = \alpha \cdot x + \beta \tag{75}$$

Using our older notation of two operands to indicate iteration, we have,

$$\theta(x, n) = \alpha^n \cdot x + \sum_{i=0}^{n-1} \alpha^i \cdot \beta \tag{76}$$

$$\theta(x, n) = \alpha^n \cdot x + \frac{a^n - 1}{a - 1} \cdot \beta \tag{77}$$

Search your feelings; you will know it to be true. This closed-form contains much information, perhaps more than I know what to do with at the moment. What follows are some informal observations about the form. First, there are two terms. This is a consequence of the fact that we are using two operands and that none of the terms are exponentiation.

Exponents would change the degree of the. A singular monomial of one degree as an operator says to not change the degree. The coefficient does however become multiplied as a result of the iteration and thus achieves some degree of exponentiation but the variable is not affected.

There is a isomorphism for each term once iterated back to the original term. We can track what the term is symbolically as we iterated closer and closer to the end state. For the monomial of degree 1, the change comes from the coefficient being exponentiated by itself. In terms of a recipe for computation, function as a verb, there is no "instruction" in the polynomial to change the degree of the polynomial.

Before we any more, we should compare this to a higher order monomial. Let us study the following.

$$\epsilon(x) = \beta \cdot x^{\alpha} \tag{78}$$

The iterated closed-form of n iterations is given by,

$$\epsilon(x, n) = \beta^{\frac{a^n - 1}{a - 1}} \cdot x^{\alpha^n} \tag{79}$$

Hmm. Our suspicions seem to be justified. There is some semblance of equivalency that transcends the particular choice of polynomial. While the degree is affected for sure, the general relation in structure of application of operations is maintained. Let us manipulate symbolically the different polynomial to our stand π notation and see if we can tease out any similarity.

$$\theta(x) = \pi(0)(\pi(1)(x, \alpha), \beta) \tag{80}$$

$$\theta(x, n) = \pi(0)(\pi(1)(x, \alpha^n), \pi(1)(\beta, \frac{a^n - 1}{a - 1})) \tag{81}$$

And now the second.

$$\epsilon(x) = \pi(1)(\pi(2)(x, \alpha), \beta) \tag{82}$$

$$\epsilon(x, n) = \pi(1)(\pi(2)(x, \alpha^n), \pi(2)(\beta, \frac{a^n - 1}{a - 1})) \tag{83}$$

I chose to ignore some of the terms in my symbolic manipulation. I did this in order to highlight the similarities between the two closed-forms of the iterations. I also moved terms around where the commutativity of the operator allows. The similarities should be shouting out for analysis. Let's now put them together to better understand the relationship between the two.

$$\theta(x, n) = \pi(0)(\pi(1)(x, \alpha^n), \pi(1)(\beta, \frac{a^n - 1}{a - 1}))$$
$$\epsilon(x, n) = \pi(1)(\pi(2)(x, \alpha^n), \pi(2)(\beta, \frac{a^n - 1}{a - 1}))$$

We even have the same terms being operated. The pattern seems to be "upgrading" all of our operations. The key insight is that both θ and ϵ repreesnt the same "type" of operation sequence. The only difference is that ϵ is an "upgraded" version of θ. And it is upgraded by 1. This means whatever operations we find in θ, we add one to the index to get the next rank operation. The is the genius of the syntactic approach. It is that nothing but ordering separates the operations that we see in a polynomial.

While I will not go into further detail here, I believe this to be fertile ground for future research. In solving the problem of iterated functions using my syntax, I have solved it for all thusly situated operations. The general form of the closed-form solution given the two cases I have presented here is as follows.

$$\zeta(x, n) = \pi(i)(\pi(i+1)(x, \alpha^n), \pi(i+1)(\beta, \frac{a^n - 1}{a - 1})) \tag{84}$$

I believe their to be general forms for polynomials of larger degree and of more complexity. What result yields is that iteration and discovering the closed-form for such functions is a matter of symbolic and syntactic analysis. There is no need to resort reasoning about the semantics of what the operations capture. All that is needed is the particular relation of each operation to every other operation and subsequent application thereof.

And thus do I conclude my presentation. I hope you have enjoyed it as much as I have.

4 Suggested Points of Departure for Future Research

For any brave souls wishing to further this line of research, first and foremost, God help you. Secondly, the following are some concerns that may be addressed in content by future endeavors.

1. Is this a viable approach to the problem of iterated functions?
2. Is there any counter example which demonstrates that this approach does not work for polynomials?
3. Is there a way to generalize this approach to other types of functions?
4. Is there a way to generalize this approach to other types of algebraic objects?