Formal Proof of the Validity of a Formula generated by the Formula Compiler

Samuel A. Alexander*
Department of Mathematics, the Ohio State University

March 7, 2011

Abstract

The Formula Compiler [1] is a program which generates mathematical formulas for computable functions. In this paper we formally prove the validity of one such formula, namely, a formula for the Fibonacci numbers. The details are highly tedious and the paper is provided merely for verification purposes.

1 The Formula

The Fibonacci sequence $F : \mathbb{N} \to \mathbb{N}$ is defined by the recurrence relation: $F(0) = F(1) = 1$, $F(n) = F(n-1)+F(n-2)$ for $n > 1$. Very simple and computationally useful non-self-referential formulas have long been known for $F(n)$. By contrast, the following formula is useless and is, basically, a crime against all that is right and holy in this world. It is only of philosophical interest, and the reason it is of philosophical interest is because it was mechanically generated with no human intervention. The formula is:

$$F(n) = r\left(1 + \sum_{\alpha=1}^{\infty} \delta\left(0, \sum_{\beta=1}^{\alpha} v(\beta, n)\right)\right)$$

*Email: alexander@math.ohio-state.edu

where:

$$
\begin{aligned}
v(\beta,n) &= c(\beta)(1-\delta(0,s(\beta)))\delta\left(s(\beta),\sum_{\gamma=1}^{s(\beta)}t(\beta,\gamma,n)(1-\delta(0,g(g(\beta,\gamma),2)))\right)\\
r(\psi) &= \delta(g(g(\psi,s(\psi)),1),40)(1)\\
&\quad+\delta(g(g(\psi,s(\psi)),1),80)((g(g(g(\psi,s(\psi)),2),2))+(g(g(g(\psi,s(\psi)),2),3)))\\
s(\lambda) &= \sum_{\mu=1}^{\lambda}(1-\delta(0,g(\lambda,\mu)))\\
t(\beta,\gamma,n) &= \delta(\gamma,1)t_1(g(\beta,\gamma),n,0,0,0)+\sum_{\epsilon=1}^{\beta}\sum_{\zeta=1}^{\beta}(1-\delta(\gamma,1))\delta(\epsilon,g(\beta,\gamma-1))\delta(\zeta,g(\beta,\gamma))m(\epsilon,\zeta)\\
c(\beta) &= \sum_{\mu=0}^{\beta}\delta(\mu,g(g(\beta,s(\beta)),1))(\delta(\mu,40)+\delta(\mu,80))\delta(0,g(g(\beta,s(\beta)),3))\\
t_1(\gamma,n,\tau_1,\tau_2,\tau_3) &= \delta(30,g(\gamma,1))\delta(\tau_1,g(\gamma,3))\delta(\tau_2,g(\gamma,4))\delta(\tau_3,g(\gamma,5))\delta(n,g(g(\gamma,2),1))\\
&\quad\cdot\delta(0,g(g(\gamma,2),2))\delta(0,g(g(\gamma,2),3))\\
m(\epsilon,\zeta) &= \ell(30,\epsilon,\zeta)+\ell(40,\epsilon,\zeta)+\ell(50,\epsilon,\zeta)+\ell(60,\epsilon,\zeta)+\ell(70,\epsilon,\zeta)+\ell(80,\epsilon,\zeta)\\
\ell(\alpha,\epsilon,\zeta) &= \delta(g(\epsilon,1),\alpha)m_\alpha(\epsilon,\zeta)\\
m_{30}(\epsilon,\zeta) &= L_D\left(\zeta,50-10\delta\left(0,\sum_{\theta=0}^{g(g(\epsilon,2),1)}\delta(\theta,2)\right)\right)L_C(\epsilon,\zeta)\\
m_{40}(\epsilon,\zeta) &= (1-\delta(0,g(\epsilon,3)))(1-\delta(0,g(g(\epsilon,3),2)))(1-\delta(0,g(\epsilon,4)))L_D(\zeta,g(\epsilon,5))L_E(g(\epsilon,3),\zeta)\\
&\quad\cdot\Delta(g(\epsilon,3),\zeta,g(\epsilon,4),1)\\
m_{50}(\epsilon,\zeta) &= L_D(\zeta,60)L_C(\epsilon,\zeta)\\
m_{60}(\epsilon,\zeta) &= t_1(\zeta,|(g(g(\epsilon,2),1))-(1)|,\epsilon,2,70)\\
m_{70}(\epsilon,\zeta) &= t_1(\zeta,|(g(g(\epsilon,2),1))-(2)|,\epsilon,3,80)\\
m_{80}(\epsilon,\zeta) &= (1-\delta(0,g(\epsilon,3)))(1-\delta(0,g(g(\epsilon,3),2)))(1-\delta(0,g(\epsilon,4)))L_D(\zeta,g(\epsilon,5))L_E(g(\epsilon,3),\zeta)\\
&\quad\cdot\Delta(g(\epsilon,3),\zeta,g(\epsilon,4),(g(g(\epsilon,2),2))+(g(g(\epsilon,2),3)))\\
L_D(\zeta,\mathbf{x}) &= \delta(\mathbf{x},g(\zeta,1))\\
L_C(\epsilon,\zeta) &= L_E(\epsilon,\zeta)\delta(g(\epsilon,2),g(\zeta,2))\\
L_E(\epsilon,\zeta) &= \delta(g(\epsilon,3),g(\zeta,3))\delta(g(\epsilon,4),g(\zeta,4))\delta(g(\epsilon,5),g(\zeta,5))\\
\Delta(\epsilon,\zeta,\mathbf{x},\mathbf{y}) &= \delta(g(g(\zeta,2),\mathbf{x}),\mathbf{y})\delta\left(0,\sum_{\mathbf{i}=1}^{\epsilon+\zeta}(1-\delta(g(g(\epsilon,2),\mathbf{i}),g(g(\zeta,2),\mathbf{i}))(1-\delta(\mathbf{i},\mathbf{x})))\right)\\
g(\mathbf{x},\mathbf{y}) &= \phi(\mathbf{x},\pi(\mathbf{y}+1))\\
\phi(\mathbf{x},\mathbf{y}) &= (1-\delta(\mathbf{y},0))(1-\delta(\mathbf{x},0))\sum_{\mathbf{i}=1}^{\mathbf{x}}\sum_{\mathbf{j}=1}^{\mathbf{x}}\delta(\mathbf{j}\mathbf{y}^{\mathbf{i}},\mathbf{x})\\
\pi(\mathbf{x}) &= \sum_{\mathbf{i}=1}^{2^{\mathbf{x}}}\mathbf{i}\sigma(\mathbf{i})\delta\left(\mathbf{x},\sum_{\mathbf{j}=1}^{\mathbf{i}}\sigma(\mathbf{j})\right)\\
\sigma(\mathbf{x}) &= \delta\left(2,\sum_{\mathbf{i}=1}^{\mathbf{x}}\sum_{\mathbf{j}=1}^{\mathbf{x}}\delta(\mathbf{i}\mathbf{j},\mathbf{x})\right)
\end{aligned}
$$

and $\delta(\mathbf{x},\mathbf{y}) = 0^{(\mathbf{x}-\mathbf{y})^2}$.

2 The Proof: Preliminary Part

My aim is to prove that this formula is a formula for the Fibonacci sequence.

Claim 1. $\delta(\mathbf{x},\mathbf{y}) = \begin{cases} 1, \text{ if } \mathbf{x}=\mathbf{y} \\ 0, \text{ if } \mathbf{x}\neq\mathbf{y}. \end{cases}$

This follows from the fact $0^0 = 1$ while $0^n = 0$ for any $n > 0$. Readers who would prefer not to use this fact (which might feel rather like cheating) are invited to replace the definition of δ using any of a number of formulas listed for sequence A000007 at the OEIS [3].

Claim 2. $\sigma(\mathbf{x}) = \begin{cases} 1, \text{ if } \mathbf{x} \text{ is prime} \\ 0, \text{ otherwise.} \end{cases}$

By Claim 1 it follows that $\sigma(\mathbf{x}) = 1$ if and only if there are exactly two pairs $(\mathbf{i},\mathbf{j})$ with $1 \leq \mathbf{i},\mathbf{j} \leq \mathbf{x}$ such that $\mathbf{ij} = \mathbf{x}$, and $\sigma(\mathbf{x}) = 0$ otherwise. This establishes Claim 2.

Claim 3. $\pi(\mathbf{x}) = \begin{cases} 0, \text{ if } \mathbf{x}=0, \\ \text{the } \mathbf{x}\text{th prime number, otherwise.} \end{cases}$

This formula for the primes was first published in [2]. First note that

$$\delta\left(\mathbf{x}, \sum_{\mathbf{j}=1}^{\mathbf{i}} \sigma(\mathbf{j})\right)$$

equals 1 if there are exactly $\mathbf{x}$ primes between 1 and $\mathbf{i}$ inclusive, and 0 otherwise. Thus

$$\sigma(\mathbf{i})\delta\left(\mathbf{x}, \sum_{\mathbf{j}=1}^{\mathbf{i}} \sigma(\mathbf{j})\right)$$

equals 1 if and only if $\mathbf{i}$ is the $\mathbf{x}$th prime. Thus, every summand in the definition of $\pi(\mathbf{x})$ is zero, except possibly for the $\mathbf{i}$th summand where $\mathbf{i}$ is the $\mathbf{x}$th prime. If $\mathbf{x} > 0$ then the $\mathbf{x}$th prime is between 1 and $2^{\mathbf{x}}$ inclusive by elementary number theory, and the claim follows.

Claim 4. $\phi(\mathbf{x},\mathbf{y}) = \begin{cases} 0, \text{ if } \mathbf{x}=0 \text{ or } \mathbf{y}=0, \\ \text{the highest } n \text{ such that } \mathbf{y}^n|\mathbf{x}, \text{ otherwise.} \end{cases}$

By Claim 1,

$$\sum_{\mathbf{i}=1}^{\mathbf{x}} \sum_{\mathbf{j}=1}^{\mathbf{x}} \delta(\mathbf{j}\mathbf{y}^{\mathbf{i}}, \mathbf{x})$$

equals the number of pairs $(\mathbf{i},\mathbf{j})$ with $1 \leq \mathbf{i},\mathbf{j} \leq \mathbf{x}$ such that $\mathbf{j}\mathbf{y}^{\mathbf{i}} = \mathbf{x}$. If $\mathbf{x},\mathbf{y} > 0$ then the number of these pairs is equal to the maximum power of $\mathbf{y}$ dividing $\mathbf{x}$, by elementary number theory. The claim follows.

Claim 5. If $\mathbf{x} > 0$ then $g(\mathbf{x}, \mathbf{y})$ is equal to the exponent on p in the prime decomposition of $\mathbf{x}$, where p is the $\mathbf{y}$th odd prime. If $\mathbf{x} = 0$ then $g(\mathbf{x}, \mathbf{y}) = 0$.

Immediate by Claims 4 and 3.
For reasons that will become clear later, pronounce $g(\mathbf{x}, \mathbf{y})$ as "get $\mathbf{x}$ $\mathbf{y}$".
This is as far as we can go without introducing some new machinery.

3 Machinery

Definition 1. By a *computation stage* I mean a positive multiple of 5. If ϵ is a computation stage, with canonical prime decomposition P, then the *line number* of ϵ is the exponent on 3 in P; the *memory* of ϵ is the exponent on 5 in P; the *callstack* of ϵ is the exponent on 7 in P; the *calling variable* of ϵ is the exponent on 11 in P; and the *return line* of ϵ is the exponent on 13 in P.

By Claim 5, it follows that if ϵ is a computation stage, then its line number is $g(\epsilon, 1)$, its memory is $g(\epsilon, 2)$ (which is nonzero since $5|\epsilon$), its callstack is $g(\epsilon, 3)$, its calling variable is $g(\epsilon, 4)$, and its return line is $g(\epsilon, 5)$.

Definition 2. By a *computation* I mean a positive integer. If β is a computation then the *number of stages* of β is the number of distinct odd primes which divide β. For any $n > 1$, the *nth stage of β* is the exponent of p in the prime decomposition of β, where p is the nth odd prime.

Definition 3. If ϵ is a computation stage and $n > 0$, then the *value of the nth variable in ϵ* is the exponent on p in the prime decomposition of the memory of ϵ, where p is the nth odd prime.

This definition makes sense: since $5|\epsilon$, the memory of ϵ is positive, so has a prime decomposition.

By Claim 5, if ϵ is a computation stage and $n > 0$ then the value of the nth variable in ϵ is $g(g(\epsilon, 2), n)$.

4 The Proof: Main Part

Claim 6. Suppose ϵ, ζ are computation stages and $\mathbf{x} > 0$, $\mathbf{y} \geq 0$. Let P and Q be the prime decompositions of the memories of ϵ and ζ, respectively. Then $\Delta(\epsilon, \zeta, \mathbf{x}, \mathbf{y}) = 1$ if Q is identical to P except for the exponents on 2 and on the $\mathbf{x}$th odd prime, and the exponent on the $\mathbf{x}$th odd prime in Q is $\mathbf{y}$. Otherwise, $\Delta(\epsilon, \zeta, \mathbf{x}, \mathbf{y}) = 0$.

First note that for any $\mathbf{i}$, if p is the $\mathbf{i}$th odd prime, then

$$(1 - \delta(g(g(\epsilon, 2), \mathbf{i}), g(g(\zeta, 2), \mathbf{i}))(1 - \delta(\mathbf{i}, x)))$$

is equal to 1 iff the p-exponent in P differs from the p-exponent in Q *and* $\mathbf{i} \neq \mathbf{x}$. It is 0 otherwise. This follows from previous claims and remarks. Second, note

that if $\mathbf{i} > \epsilon + \zeta$, and p is the $\mathbf{i}$th odd prime, then the p-exponent in both P and Q is 0, since a positive p-exponent k would entail either $\epsilon \geq 5^{p^k} > 5^{(\epsilon+\zeta)^k} > \epsilon$ or $\zeta \geq 5^{p^k} > 5^{(\epsilon+\zeta)^k} > \zeta$. Combining these facts, it follows

$$\sum_{\mathbf{i}=1}^{\epsilon+\zeta}(1 - \delta(g(g(\epsilon, 2), \mathbf{i}), g(g(\zeta, 2), \mathbf{i}))(1 - \delta(\mathbf{i}, \mathbf{x})))$$

equals the number of odd primes, excepting the $\mathbf{x}$th one, where exponents differ in P and Q. From this, the claim follows.

Claim 7. If ϵ, ζ are computation stages and $\mathbf{x} > 0$, $\mathbf{y} \geq 0$, then $\Delta(\epsilon, \zeta, \mathbf{x}, \mathbf{y}) = 1$ iff the memories of ϵ and ζ are equal except that the value of the $\mathbf{x}$th variable in ζ is $\mathbf{y}$. Otherwise, $\Delta(\epsilon, \zeta, \mathbf{x}, \mathbf{y}) = 0$.

This is simply a rephrasing of Claim 6 using Definition 3.

Claim 8. Let ϵ, ζ be computation stages. Then $L_E(\epsilon, \zeta) = 1$ if ϵ and ζ have the same callstack, calling variable, and return line. Otherwise, $L_E(\epsilon, \zeta) = 0$.

Immediate.

Claim 9. Let ϵ, ζ be computation stages. Then $L_C(\epsilon, \zeta) = 1$ if ϵ and ζ have the same memory, callstack, calling variable, and return line. Otherwise, $L_C(\epsilon, \zeta) = 0$.

Immediate.

Claim 10. Let ζ be a computation stage and let $\mathbf{x} \geq 0$. Then $L_D(\zeta, \mathbf{x}) = 1$ if ζ has line number $\mathbf{x}$. Otherwise $L_D(\zeta, \mathbf{x}) = 0$.

Immediate.

Definition 4. Let S_{80} be the string: "return variable 2 plus variable 3". If ϵ and ζ are computation stages, say that the pair (ϵ, ζ) *follows* S_{80} if the following hold:

1. The callstack of ϵ is a computation stage.
2. The calling variable of ϵ is nonzero.
3. The line number of ζ equals the return line of ϵ.
4. The callstack, calling variable, and return line of ζ equal those of the callstack of ϵ.
5. The memory of ζ equals the memory of the callstack of ϵ except that the the value of the nth variable in ζ is the value of the 2nd variable in ϵ plus the value of the 3rd variable in ϵ, where n is the calling variable of ϵ.

Claim 11. Let ϵ, ζ be computation stages. If (ϵ, ζ) follows S_{80} then $m_{80}(\epsilon, \zeta) = 1$. Otherwise, $m_{80}(\epsilon, \zeta) = 0$.

Immediate.

Definition 5. Let S_{70} be the string: "let variable 3 be this function evaluated at the difference between 2 and variable 1". If ϵ and ζ are computation stages, say that the pair (ϵ, ζ) *follows* S_{70} if the following hold:

1. The line number of ζ is 30.
2. The callstack of ζ is ϵ.
3. The calling variable of ζ is variable 3.
4. The return line of ζ is 80.
5. The value of variable 1 in ζ is the abolute value of: the value of variable 1 in ϵ, minus 2.
6. The value of variables 2 and 3 in ζ are 0.

Claim 12. Let ϵ, ζ be computation stages. If (ϵ, ζ) follows S_{70} then $m_{70}(\epsilon, \zeta) = 1$. Otherwise, $m_{70}(\epsilon, \zeta) = 0$.

Immediate by the definition of t_1.

A similar definition and proof go for m_{60}: $m_{60}(\epsilon, \zeta) = 1$ iff (ϵ, ζ) follows the string "let variable 2 be this function evaluated at the difference between 1 and variable 1".

Definition 6. Let S_{50} be the string "do nothing and go to line 60". Say that a pair (ϵ, ζ) of computation stages follows S_{50} if ϵ and ζ have the same memory, callstack, calling variable, and return line, and the 3-exponent of ζ is 60.

Claim 13. (ϵ, ζ) follows S_{50} iff $m_{50}(\epsilon, \zeta) = 1$, and otherwise $m_{50}(\epsilon, \zeta) = 0$.

Again, immediate by Claims 10 and 9.

Definition 7. Let S_{40} be the string "return 1". Say that a pair (ϵ, ζ) of computation stages *follows* S_{40} if the following hold:

1. The callback of ϵ is a computation stage.
2. The calling variable of ϵ is nonzero.
3. The line number of ζ equals the return line of ϵ.
4. The callstack, calling variable, and return line of ζ equal those of the callstack of ϵ.
5. The memory of ζ equals the memory of the callstack of ϵ except that the value of the nth variable in ζ is 1, where n is the calling variable of ϵ.

Claim 14. (ϵ, ζ) follows S_{40} iff $m_{40}(\epsilon, \zeta) = 1$, and otherwise $m_{40}(\epsilon, \zeta) = 0$.

Immediate.

Definition 8. Let S_{30} be the string "if variable 1 is less than 2 then goto line 40, otherwise goto line 50". Say that a pair (ϵ, ζ) of computation stages *follows* S_{30} if:

1. ζ and ϵ have the same memory, callstack, calling variable, and return line.
2. If the value of variable 1 in ϵ is < 2, then the line number of ζ is 40.
3. If the value of variable 1 in ϵ is ≥ 2, then the line number of ζ is 50.

Claim 15. (ϵ, ζ) follows S_{30} iff $m_{30}(\epsilon, \zeta) = 1$. Otherwise, $m_{30}(\epsilon, \zeta) = 0$.

First, note that $g(g(\epsilon, 2), 1)$ is the value of variable 1 in ϵ. Therefore,

$$\sum_{\theta=0}^{g(g(\epsilon,2),1)} \delta(\theta, 2)$$

equals the number of times 2 occurs between 0 and the value of variable 1 in ϵ, inclusive. Of course, 2 will occur zero times iff the value of variable 1 in ϵ is < 2. From here, the claim is clear (use Claims 10 and 9).

Claim 16. Let (ϵ, ζ) be any pair of computation stages. Then $m(\epsilon, \zeta) = 1$ if and only if the line number of ϵ is an element of $\{30, 40, 50, 60, 70, 80\}$ and (ϵ, ζ) follows S_α where α is the line number of ϵ. Otherwise, $m(\epsilon, \zeta) = 0$.

Follows immediately from the above claims.

Claim 17. Suppose β is a computation (recall Definition 2). Then $s(\beta)$ equals the number of stages of β.

Note that for any $1 \leq \mu \leq \beta$, $1 - \delta(0, g(\beta, \mu))$ equals 1 if and only if the μth odd prime divides β. If any odd prime divides β, then that odd prime is at most β itself. The claim follows.

Definition 9. A computation β is *gapless* if β has a positive number of stages and, whenever n is a positive integer at most the number of stages of β, then the nth stage of β is a computation stage (i.e., a positive multiple of 5).

Claim 18. Let β be a computation, $n \in \mathbb{N}$. If β is not gapless, then $v(\beta, n) = 0$ for any $n \in \mathbb{N}$.

Proof. If there are zero stages of β, then the factor $(1 - \delta(0, s(\beta)))$ of $v(\beta, n)$ kills it. Assume β has at least one stage.

If β is not gapless, there are two possibilities: either there is a $1 \leq \gamma$ such that the γth odd prime divides β but $g(\beta, \gamma)$ is not a computation stage (i.e., is not a positive multiple of 5), or there are $1 \leq \gamma < \gamma'$ such that the the γ'th odd prime divides β while the γth odd prime does not.

Assume the former is true: the γth odd prime divides β but $g(\beta, \gamma)$ is not a positive multiple of 5. Then $g(g(\beta, \gamma), 2) = 0$. Thus, the γth summand in the

sum defining $v(\beta, n)$ is zero. Every other summand is evidently 0 or 1, so the total sum cannot possibly equal $s(\beta)$. It follows $v(\beta, n) = 0$.

But suppose there are $1 \leq \gamma < \gamma'$ such that the γ'th odd prime divides β while the γth odd prime does not. I claim $t(\beta, \gamma, n) = 0$. If $\gamma = 1$ then $(1 - \delta(\gamma, 1)) = 0$, killing the double sum in the definition of t, so we need only show $t_1(g(\beta, \gamma), n, 0, 0, 0) = 0$. This is true because $\delta(30, g(g(\beta, \gamma), 1)) = 0$ is one of the factors in the product $t_1(g(\beta, \gamma), n, 0, 0, 0)$. But assume $\gamma > 1$. Then the first summand of t is killed by the $\delta(\gamma, 1)$ factor and it's enough to show the double sum vanishes. Every summand in the double sum is a product one of whose factors is $\delta(\zeta, g(\beta, \gamma))$. This factor can only be nonzero if $\zeta = g(\beta, \gamma)$. But $g(\beta, \gamma) = 0$ by choice of γ, and the dummy variable ζ ranges over $(1, \ldots, \beta)$. So every summand in the double sum in question is 0, so $t(\beta, \gamma, n) = 0$. Again, there are only $s(\beta)$ summands in the sum in the definition of $v(\beta, n)$, each of which is 1 or 0, and I've shown one of them is 0, so the whole sum cannot equal $s(\beta)$, so $v(\beta, n) = 0$. □

Definition 10. If β is a gapless computation, then the *final stage* of β is $g(\beta, s(\beta))$.

Notice that the final stage of a gapless computation is a computation stage.

Claim 19. Suppose β is a gapless computation. Then $c(\beta) = 1$ if the final stage of β has line number 40 or 80 and its callstack is 0. Otherwise, $c(\beta) = 0$.

The factor $\delta(\mu, g(g(\beta, s(\beta)), 1))$ ensures the only nonzero summand in the definition of $c(\beta)$ is the summand where $\mu = g(g(\beta, s(\beta)), 1))$ is the line number of the final stage of β. The second factor is 1 if this line number is 40 or 80, 0 otherwise. The third factor is 1 if the callstack of the final stage $g(\beta, s(\beta))$ of β is 0, 0 otherwise. The claim follows.

Claim 20. Suppose β is a gapless computation and $n \in \mathbb{N}$. Let ϵ be the first stage of β and let ζ be the final stage of β. Then $v(\beta, n)$ is either 1 or 0, and it is 1 if and only if:

1. ϵ has line number 30, callstack 0, calling variable 0, and return line 0.
2. The value of variables 1, 2, and 3 in ϵ are n, 0, and 0, respectively.
3. ζ has line number 40 or 80 and callstack 0.

Assume $v(\beta, n) = 1$. Then $t(\beta, \gamma, n) = 1$ for every $1 \leq \gamma \leq s(\beta)$, lest the sum in the definition of $v(\beta, n)$ would be smaller than $s(\beta)$, making $v(\beta, n)$ vanish. In particular, $t(\beta, 1, n) = 1$. It follows $t_1(g(\beta, 1), n, 0, 0, 0) = 1$. By definition of t_1 it follows conditions 1 and 2 are met. Furthermore, since $v(\beta, n) = 1$, this forces $c(\beta) \neq 0$. The factor $\delta(\mu, g(g(\beta, s(\beta)), 1))$ in the definition of $c(\beta)$ ensures the only nonzero summand is the one when $\mu = g(g(\beta, s(\beta)), 1))$ is the line number of the final stage of β. The factor $\delta(\mu, 40) + \delta(\mu, 80)$, then, being nonzero, guarantees this line number is 40 or 80. The factor $\delta(0, g(g(\beta, s(\beta)), 3))$ ensures the final stage of β has callstack 0.

The converse is similar.

Claim 21. Suppose β is a gapless computation, $n \in \mathbb{N}$, and $v(\beta, n) = 1$. By the previous claim, the line number of the final stage is 40 or 80. If it is 40, then $r(\beta) = 1$. Otherwise, $r(\beta)$ equals the sum of the values of the 2nd and 3rd variables in the final stage of β.

Immediate (remember that $g(g(\beta, s(\beta)), 1)$ is the line number of the final stage of β).

Claim 22. Suppose β is a gapless computation, $n \in \mathbb{N}$. Then $v(\beta, n) = 1$ or $v(\beta, n) = 0$, and $v(\beta, n) = 1$ if and only if:

1. The first stage of β has line number 30, no callstack, calling variable or return line, its 1st variable has value n and 2nd and 3rd variables have value 0.
2. The final stage of β has line number 40 or 80 and no callstack.
3. For every pair (ϵ, ζ) of consecutive stages of β, ϵ has line number in $\{30, 40, 50, 60, 70, 80\}$ and (ϵ, ζ) follows S_α where α is the line number of ϵ.

The factor $c(\beta)$ of $v(\beta, n)$ ensures part 2. Aside from that, $v(\beta, n) = 1$ precisely if all $s(\beta)$ summands in the summation notation are 1. Each summand equals $t(\beta, \gamma, n)$ since β is gapless. $t(\beta, \gamma, n)$ is $t_1(g(\beta, 1), n, 0, 0, 0)$ if $\gamma = 1$, ensuring the first condition. Otherwise $t(\beta, \gamma, n)$ is a double sum whose only nonzero summand is when $\epsilon = g(\beta, \gamma - 1)$ and $\zeta = g(\beta, \gamma)$, and this summand is $m(\epsilon, \zeta)$, ensuring condition 3.

5 The Proof: Final Part

Claim 23. Let $n \in \mathbb{N}$. Suppose there is some gapless computation κ such that $v(\kappa, n) = 1$. Then

$$1 + \sum_{\alpha=1}^{\infty} \delta \left(0, \sum_{\beta=1}^{\alpha} v(\beta, n) \right)$$

is the minimum such gapless computation. If there is no such κ, then the sum in question diverges.

I claim that in the outtermost summation notation, the αth summand is 1 if there are no such κ between 1 and α inclusive, and 0 otherwise. First, assume there are no such κ between 1 and α inclusive. Then $\sum_{\beta=1}^{\alpha} v(\beta, n) = 0$, and by delta'ing it with 0, we get 1, as desired. Second, assume there is such a κ between 1 and α inclusive. Then $\sum_{\beta=1}^{\alpha} v(\beta, n) \geq 1$, and when we delta it with 0 we get 0.

The claim immediately follows. If there is such a κ (we can assume it minimal), then the first $\kappa - 1$ terms of the outtermost sum are 1, and all others are 0. If there is no such κ, then every summand of the outtermost sum is 1, and the series diverges.

Theorem 24. For any $n \in \mathbb{N}$, the nth Fibonacci number is

$$F(n) = r\left(1 + \sum_{\alpha=1}^{\infty} \delta\left(0, \sum_{\beta=1}^{\alpha} v(\beta, n)\right)\right)$$

Proof. Consider the following algorithm written in pseudocode.

- 00. Accept input n (call n the *1st variable*), goto line 05.
- 05. Initialize an empty stack, goto line 10.
- 10. Initialize $x = 0$ (call x the *2nd variable*), goto line 20.
- 20. Initialize $y = 0$ (call y the *3rd variable*), goto line 30.
- 30. If $n < 2$ then goto line 40, else goto line 50.
- 40. If the stack is empty, output 1. Otherwise, pull a record off the stack. The record will tell us a line number, a variable, and a computation stage. Change x, y, n to match the recorded computation stage, then put 1 in the recorded variable, go to the recorded line number, and discard the record.
- 50. "Endif" (go to line 60).
- 60. Create a record which lists x as a variable, 70 as a line number, and a computation stage which is a snapshot of the current state of memory. Put this record on the stack. Change n to $|n - 1|$, change x and y to 0, and go to line 30.
- 70. Create a record which lists y as a variable, 80 as a line number, and a computation stage which is a snapshot of the current state of memory. Put this record on the stack. Change n to $|n - 2|$, change x and y to 0, and go to line 30.
- 80. If the stack is empty, output $x + y$. Otherwise, pull a record off the stack. Change x, y, n to match the recorded computation stage, then put $x+y$ in the recorded variable, go to the recorded line number, and discard the record.

While we admit this is a candidate for Worst Fibonacci Algorithm Ever, it is nevertheless clear that it does compute the nth Fibonacci number. When we run it, we can skip straight to line 30, considering the previous lines done by default; evidently we will never need to go to a line before line 30 thereafter. By construction, running the algorithm traces out a gapless computation β with (by the claims in the previous section) $v(\beta, n) = 1$.

Thus for every n there is a gapless computation β_n with $v(\beta_n, n) = 1$. Conversely, by the previous section, any gapless computation $\beta_n{}'$ with $v(\beta_n{}', n) = 1$ faithfully follows the above algorithm. Thus any such $\beta_n{}'$ passes through stages whose memories agree with the stages of β_n, at least for the values of variables

1, 2, and 3. So for any such $\beta_n{}'$, in the final computation stage, the values of variables 2 and 3 are the same as the values of variables 2 and 3 in the final stage of β_n, i.e., their sum is the nth Fibonacci number. The theorem follows. □

References

[1] S.A. Alexander. The Formula Compiler. http://www.xamuel.com/formula.php

[2] S.A. Alexander. (2006). Formulas for computable and non-computable functions. *The Rose-Hulman Journal of Undergraduate Mathematics* **7** (2). http://www.rose-hulman.edu/mathjournal/archives/2006/vol7-n2/paper2/v7n2-2pd.pdf

[3] N.J.A. Sloane. The Online Encyclopedia of Integer Sequences (OEIS). http://www.oeis.org/A000007