# Finding Linear-Recurrent Solutions to Hofstadter-Like Recurrences Using Symbolic Computation

Nathan Fox*

**Abstract**

The Hofstadter $Q$-sequence, with its simple definition, has defied all attempts at analyzing its behavior. Defined by a simple nested recurrence and an initial condition, the sequence looks approximately linear, though with a lot of noise. But, nobody even knows whether the sequence is infinite. In the years since Hofstadter published his sequence, various people have found variants with predictable behavior. Oftentimes, the resulting sequence looks complicated but provably grows linearly. Other times, the sequences are eventually linear recurrent. Proofs describing the behaviors of both types of sequence are inductive. In the first case, the inductive hypotheses are fairly ad-hoc, but the proofs in the second case are highly automatable. This suggests that a search for more sequences like these may be fruitful. In this paper, we develop a step-by-step symbolic algorithm to search for these sequences. Using this algorithm, we determine that such sequences come in infinite families that are themselves plentiful. In fact, there are hundreds of easy to describe families based on the Hofstadter $Q$-recurrence alone.

## 1 Introduction

The Hofstadter $Q$-sequence, which is defined by the recurrence $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$ and the initial conditions $Q(1) = Q(2) = 1$, was first introduced by Douglas Hofstadter in the 1960s [7]. The first fifteen terms increase monotonically, but thereafter the sequence rapidly devolves into chaos. On the macro level, $Q(n)$ seems to oscillate around $\frac{n}{2}$, but nobody has been able to prove anything more than the statement that if

$$\lim_{n \to \infty} \frac{Q(n)}{n}$$

exists, it must equal one half [6]. Furthermore, nobody has been able to prove that $Q(n)$ even *exists* for all $n$. If $Q(n - 1) \geq n$ for some $n$, then $Q(n)$ would be defined in terms of $Q(k)$ for some $k \leq 0$. But, $Q$ is not defined on nonpositive inputs, so $Q(n)$ would fail to exist. All subsequent terms would also fail to exist, so the sequence would be finite in

---

*Department of Mathematics, Rutgers University, Piscataway, New Jersey, fox@math.rutgers.edu

this scenario. If a sequence is finite because of this sort of happenstance, we say that the sequence *dies*. We do not know whether the Hofstadter $Q$-sequence dies, but we do know that it exists for at least $10^{10}$ terms [12, A005185].

More recently, Conolly examined the recurrence $C(n) = C(n - C(n - 1)) + C(n - 1 - C(n - 2))$ with $C(1) = C(2) = 1$ as the initial condition [2]. This setup looks quite similar to the $Q$-recurrence; the only difference is the $-1$ in the second term. But, the resulting sequence grows monotonically and satisfies

$$\lim_{n \to \infty} \frac{C(n)}{n} = \frac{1}{2}.$$

In particular, $C(n) - C(n-1)$ is always either 0 or 1, a property commonly known as *slow*. Similar sequences have been studied by other authors [14]. Most of the sequences they have analyzed use recurrences with shifts similar to the one in Conolly's recurrence, though they also analyze the Hofstadter $V$-Sequence, given by $V(n) = V(n - V(n - 1)) + V(n - V(n - 4))$ with $V(1) = V(2) = V(3) = V(4) = 1$, which is also slow [1]. Another example of a slow sequence is the Hofstadter-Conway \$10000 Sequence, given by $A(n) = A(A(n - 1)) + A(n - A(n - 1))$ with $A(1) = A(2) = 1$. Conway famously offered a \$10000 prize for a sufficient analysis of the behavior of this sequence, which was claimed by Colin Mallows a few years later [11].

There are two methods commonly used to prove that Hofstadter-like sequences are slow. In some cases, there are combinatorial interpretations for slow sequences involving counting leaves in nested tree structures [9]. The sequence counting the leaves is obviously slow; the main difficulty comes in showing that the nested recurrence also describes the same structure. For some slow sequences, though, there is no known combinatorial interpretation. The other proofs of slowness usually go by induction with complicated inductive hypotheses. For a sequence $(a_n)$, one would love to work with just the inductive hypothesis that $a_m - a_{m-1} \in \{0, 1\}$ for all $m < n$, but this is never enough. Instead, additional inductive hypotheses are required to handle certain cases. These extra hypotheses strongly depend on the sequence in question. While the "shifted" sequences have similar proofs to each other, the proof for the Hofstadter $V$-sequence uses a different sort of inductive hypothesis. One would like to try to automate these proofs, but this would require some method of determining the appropriate inductive hypotheses.

While the study of slow sequence has been fruitful [3,8,9,14], other predictable Hofstadter-like sequences have been found. Whereas slow sequences often result from varying the recurrence while retaining the initial condition of the $Q$-sequence, sequences from this other class often result from varying the initial condition while retaining the original $Q$-recurrence. The first such example comes from Golomb [6], who replaces the initial conditions of the $Q$-sequence with the new values $Q(1) = 3$, $Q(2) = 2$, $Q(3) = 1$. (We will use the shorthand $[3, 2, 1]$ to describe this initial condition, and we will use similar notation going forward.) The first few terms of the resulting sequence are $3, 2, 1, 3, 5, 4, 3, 8, 7, 3, 11, 10, \ldots$, and the pattern continues forever, giving the linear-recurrent (in fact, quasilinear) sequence defined

by

$$\begin{cases} Q(3k) = 3k - 2 \\ Q(3k + 1) = 3 \\ Q(3k + 2) = 3k + 2. \end{cases}$$

In our current setup, the only linear recurrent solutions we can possibly obtain to the Hofstadter $Q$-recurrence would be quasilinear, since we require $Q(n) \leq n$ for all $n$. For our purposes, this condition is overly restrictive. To allow ourselves more flexibility, we will instead say that $Q(n) = 0$ whenver $n \leq 0$. The original question of the existence of the Hofstadter $Q$-sequence can still be asked: "Is $Q(n) \leq n$ for all $n$?" Note that it is still possible for a sequence to die in this new setting. For example, in the Hofstadter $Q$-sequence under this convention, if it happens that *both* $Q(n-1) \geq n$ and $Q(n-2) \geq n$, then we obtain $Q(n) = 0$. As a result, $Q(n+1)$ would be defined in terms of itself, and hence it would be undetermined.

This more general setting allows us to find linear-recurrent solutions to the Hofstadter $Q$-recurrence that were not possible before. One of these was found by Ruskey, who used the initial condition $[3, 6, 5, 3, 6, 8]$ to embed the Fibonacci numbers in the Hofstadter $Q$-recurrence [13]. This sequence takes values

$$\begin{cases} Q(3k) = F_{k+4} \\ Q(3k + 1) = 3 \\ Q(3k + 2) = 6. \end{cases}$$

The solutions of Golomb and Ruskey both satisfy linear recurrences ($a_n = 2a_{n-3} - a_{n-6}$ in the case of Golomb; $a_n = 2a_{n-3} - a_{n-9}$ in the case of Ruskey). We would like to develop a general method for finding more solutions like these. One might hope for a method that, given a linear recurrence and a Hofstadter-like recurrence, determines whether there is a sequence that eventually satisfies both of them. Unfortunately, this is quite a lofty goal. Rather, we exploit a deeper structure of these sequences and generalize that. Golomb's sequence is a quasipolynomial with period 3. Ruskey's sequence, while not a quasipolynomial, is also structured as an interleaving of three simpler sequences. We will describe an automatic way of searching for such interleaved solutions, where the following items are allowed to vary:

- The recurrence under consideration

- The number of interleaved sequences

- The growth rate of each of the interleaved sequences.

The methods of this paper apply to *linear* nested recurrences. These are recurrences of the form

$$Q(n) = P(n) + \sum_{i=1}^{d} \alpha_i Q(E_i),$$

3

where $P(n)$ is an explicit expression in $n$, $k$ is a nonnegative integer, each $\alpha_i$ is an integer, and each $E_i$ is an expression of the same form as the generic formula for $Q(n)$ (thereby allowing for arbitrarily many nesting levels). The methods will not apply completely in all cases; certain positivity conditions will need to be satisfied. Most commonly, we will have $P(n) = 0$, each $\alpha_i$ positive, and each $E_i$ of the form $n - \beta_i - Q(n - \gamma_i)$ for some nonnegative integer $\beta_i$ and positive integer $\gamma_i$. (This is the case for the Hofstadter $Q$-recurrence, where $P(n) = 0$, $k = 2$, $\alpha_1 = \alpha_2 = 1$, $E_1 = n - Q(n - 1)$, and $E_2 = n - Q(n - 2)$.) We will call recurrences satisfying this condition about the $E_i$'s *basic*, and our methods will always apply fully to basic recurrences.

In Section 2, we will introduce a formalism that encapsulates the notion of an interleaving of simple sequences. Then, in Section 3, we will describe our algorithm that finds these special solutions. Finally, in Sections 4 and 5, we will describe some notable sequences found using the methods in this paper, and we will discuss some future extensions.

A Maple package implementing all of the algorithms in this paper, as well as some related procedures, can be found at `http://math.rutgers.edu/~nhf12/nicehof.txt`. Generally speaking, the procedures in this package offer more general versions of the algorithms described in this paper. For example, the code allows the user to modify the "default" initial conditions (for when $n \leq 0$).

# 2    Positive Recurrence Systems

In the case of Ruskey's solution, each of the three interleaved sequences can be described by a homogeneous linear recurrence with nonnegative coefficients:

$$\begin{cases} a_k = a_{k-1} & a_0 = 3 \\ b_k = b_{k-1} & b_0 = 6 \\ c_k = c_{k-1} + c_{k-2} & c_1 = 5, c_2 = 8. \end{cases}$$

(Note that these recurrences are not unique.) Golomb's solution cannot be expressed in this way, but each of its interleaved sequences can be described by a nonhomogeneous linear recurrence with nonnegative coefficients:

$$\begin{cases} a_k = 3 + a_{k-1} & a_0 = 3 \\ b_k = 3 \\ c_k = 3 + c_{k-1} & c_1 = 1. \end{cases}$$

In both of these cases, we have a system of three nonhomogneous linear recurrences where all coefficients are nonnegative. Here, none of the recurrences refer to each other in their definitions, but theoretically this should be possible. This leads to the following generalization:

**Definition 1.** *A* positive recurrence system *is a system of $m$ nonhomogeneous linear recurrences of the form*

$$
\begin{cases}
a_k^{(1)} = P_1(k) + \sum_{\ell=1}^{d} \sum_{j=1}^{m} \alpha_{1,\ell,j} a_{k-\ell}^{(j)} \\
a_k^{(2)} = P_2(k) + \sum_{\ell=1}^{d} \sum_{j=1}^{m} \alpha_{2,\ell,j} a_{k-\ell}^{(j)} \\
\vdots \\
a_k^{(m)} = P_m(k) + \sum_{\ell=1}^{d} \sum_{j=1}^{m} \alpha_{m,\ell,j} a_{k-\ell}^{(j)}
\end{cases}
$$

*satsifying the following conditions:*

- *$d$ is a nonnegative integer.*

- *$P_1$ through $P_m$ are eventually nonnegative integer-valued polynomials.*

- *Each $\alpha_{i,\ell,j}$ is a nonnegative integer.*

Note that, for convenience, we may sometimes have a recurrence system where $a_k^{(i)}$ refers to $a_k^{(j)}$ for some $j < i$. This is permissible, as we can just replace $a_k^{(j)}$ with its right-hand side in order to conform to Definition 1.

The solutions to Hofstadter-like recurrences that we seek will eventually be interleavings of sequences that together satisfy a positive recurrence system. What follows is a formalization of this notion:

**Definition 2.** *Let $m$ be a positive integer. The sequence $(a_k)_{k \geq 1}$ is* positive-recurrent *with period $m$ if there exist an integer $K$ such that the sequences $\left\{ (a_{ms+r})_{s \geq K} : 0 \leq r < m \right\}$ satisfy a positive recurrence system.*

Observe that eventually nonnegative polynomials are trivially positive-recurrent with period 1, as we can take $d = 0$. Also, any sequence satisfying a homogeneous linear recurrence with nonnegative coefficients is positive-recurrent with period 1, as we can take $P_1$ identically 0.

Generally speaking, any positive-recurrent sequence is eventually linear recurrent, as are each of the component sequences. This is true because a positive recurrence system can be converted into a linear system of equations for the generating functions of the component sequences. Each generating function is therefore a rational function.

We will be concerned with determining the rate of growth of each sequence in a solution to a positive recurrence system. In order for things to be well-defined and easy to analyze, we will need the following technical definition.

**Definition 3.** *An* initial condition *of length $N$ to a positive recurrence system is called* eventually positive *if the following conditions hold:*

- *If $k \geq N$, then $P_r(k) \geq 0$ for all $r$ and all $k$.*

- *For all $0 \leq i \leq d$, $a_{N-i}^{(r)} > 0$ for all $r$.*

Any long enough positive initial condition is eventually positive. But, this definition allows for some nonpositive values early in the initial condition, so long as those values are never used in calculating recursively defined terms. Furthermore, we require all of the polynomials to be nonnegative when calculating recursive terms. This will be useful in our analysis, though it is not strictly necessary. (A much more complicated, weaker condition would suffice, and, in that case, we would not even need all of the polynomials to be eventually nonnegative.)

In the case where we have a solution to a positive recurrence system given by an eventually positive initial condition, the following algorithm determines the order of growth of each component sequence.

1. Define a weighted directed graph $G$ as follows:

    - The vertices of $G$ are the integers $\{1, \ldots, m\}$.
    - There is an arc from $i$ to $j$ if and only if, for some $\ell$, $\alpha_{i,\ell,j} > 0$.
    - The weight of the arc from $i$ to $j$ is

    $$\sum_{\ell=1}^{d} \alpha_{i,\ell,j}.$$

2. Initialize variables $d_1, d_2, \ldots, d_m$ so that $d_i$ equals the degree of $P_i$.

3. Let $W$ denote the set of vertices $v$ in $G$ satisfying one of the following:

    - $v$ is in a directed circuit with at least one arc having weight greater than 1.
    - $v$ is in more than one directed circuit that does not contain $v$ as an intermediate vertex.

    For each $v \in W$, set $d_v$ to $\infty$ and delete any outgoing arc from $v$ in $G$ that is part of a cycle. Call the resulting graph $G'$. (We can actually delete *all* outgoing arcs from $v$, but the form we have stated here will be more useful when we prove this algorithm's correctness.)

4. Define the following relation $\sim$ on $\{1, 2, \ldots, m\}$:

    $i \sim j$ if and only if $(i = j)$ or ($i$ and $j$ are in a cycle together in $G'$).

    It is easy to check that $\sim$ is an equivalence relation. Each equivalence class is either a single vertex or a cycle. (Every vertex in a cycle is in exactly one cycle, or else we would have removed all of its cycle-making outgoing arcs in step 3, and all such vertices are equivalent to each other.)

5. Define a directed graph $H$ as follows:

- The vertices of $H$ are the equivalence classes of $\{1, 2, \ldots, m\}$ under $\sim$.

- There is an arc from class $I$ to class $J$ if and only if there is an arc in $G'$ from some $i \in I$ to some $j \in J$.

If $H$ contains a directed cycle $I_1, I_2, \ldots, I_q$, then for each $1 \leq h < q$, there is an arc in $G'$ from some $i_h^{out} \in I_h$ to some $i_{h+1}^{in} \in I_{h+1}$. Also, there is an arc in $G'$ from some $i_q^{out} \in I_q$ to some $i_1^{in} \in I_1$. Furthermore, by the definition of $\sim$, for each $h$, there is a (possibly trivial) directed path from $i_h^{in}$ to $i_h^{out}$ within $I_h$. Concatenating all of these arcs together gives a cycle in $G'$ that includes elements of multiple equivalence classes, which contradicts the definition of $G'$.

So, we can conclude that $H$ contains no directed cycles.

6. For each vertex $I$ of $H$, initialize a variable $d_I = \max_{i \in I} d_i$.

7. Topologically sort the vertices of $H$. Consider the vertices $I$ from last to first:

   - If $I$ is a cycle in $G'$ (including a single vertex with a loop), set $d_I$ to $d_I + 1$.
   - For all $J$ with an arc from $J$ to $I$, set $d_J = \max(d_J, d_I)$.

At the end of this process, we have values $d_I$ for each equivalence class $I$. In general, for an integer $r$, let $\bar{r}$ denote its equivalence class under $\sim$. We now make the following claim:

**Claim 1.** *Suppose we have a positive recurrence system with $m$ component sequences, along with an eventually positive initial condition. Let $1 \leq r \leq m$ be an integer. If $d_{\bar{r}} < \infty$, then $a_k^{(r)} = \Theta\left(k^{d_{\bar{r}}}\right)$. If $d_{\bar{r}} = \infty$, then $a_k^{(r)}$ grows exponentially.*

The proof of this claim is uninteresting, so it has been relegated to Appendix A. The general strategy is to inductively substitute expressions around cycles in order to obtain recurrences for sequences $a_k^{(r)}$ that refer only to earlier versions of themselves. Then, we use the general theory of linear recurrences to make claims about their asymptotics.

## 3   The Algorithm

What follows is a description of a generic run of our algorithm for finding positive-recurrent solutions to nested recurrences. We will walk through the steps of searching for period-$m$ solutions to a linear nested recurrence $Q(n)$. A dominant term in the complexity of our algorithm is that it is exponential in $m$, so the first thing we do is fix $m$. We will follow the steps of the algorithm by applying it to a running example: searching for period-4 solutions to the recurrence

$$R(n) = R(n - R(n-1)) + R(n - R(n-2)) + R(n - R(n-3)).$$

We choose this particular example because it will illustrate many facets of the algorithm without becoming too unwieldy.

## 3.1 Fixing the Behavior of the Subsequences

A positive-recurrent solution to $Q(n)$ with period $m$ has the form

$$\begin{cases} Q(mk) = a_k^{(0)} \\ Q(mk+1) = a_k^{(1)} \\ Q(mk+2) = a_k^{(2)} \\ \quad\vdots \\ Q(mk+(m-1)) = a_k^{(m-1)} \end{cases}$$

for some sequences $\left(a_k^{(0)}\right)$ through $\left(a_k^{(m-1)}\right)$. (For convenience, we index the interleaved sequences from zero in this context.) For convenience, we define the following growth properties that these component sequences may have:

**Definition 4.**

- We say $\left(a_k^{(r)}\right)$ is constant if, for sufficiently large $k$, $a_k^{(r)} = A$ for some constant $A$.

- We say $\left(a_k^{(r)}\right)$ is linear if, for sufficiently large $k$, $a_k^{(r)} = Ak + B$ for some constants $A$ and $B$.

- We say $\left(a_k^{(r)}\right)$ is superlinear if $a_k^{(r)} = \omega(k)$.

- We say that $\left(a_k^{(r)}\right)$ is standard linear if $a_k^{(r)} = mk + B$ for some constant $B$.

- We say that $\left(a_k^{(r)}\right)$ is steep linear if $a_k^{(r)} = Ak + B$ for some constants $A$ and $B$ with $A > m$.

- We say that $\left(a_k^{(r)}\right)$ is steep if $\left(a_k^{(r)}\right)$ is either steep linear or superlinear.

To start the algorithm, we need to decide, for each of the $m$ component sequences, are we looking for a solution where that subsequence is *constant*, *standard linear*, or *steep*? (We do not concern ourselves with component sequences intermediate in growth between constant and standard linear, as they seem to be uncommon in this context and can be harder to analyze. But, the Maple package can sometimes handle these, as long as the user explicitly asks it to.) To keep track of our choices, the algorithm stores variables $A_0, A_1, \ldots, A_{m-1}$. We set $A_r = 0$ if we decide that $\left(a_k^{(r)}\right)$ is to be constant, $A_r = m$ if $\left(a_k^{(r)}\right)$ is standard linear, and $A_r = \infty$ if $\left(a_k^{(r)}\right)$ is steep. Observe that if $\left(a_k^{(r)}\right)$ is not steep, then $\left(a_k^{(r)}\right) = A_r k + B_r$ for some constant $B_r$. In general, to perform an exhaustive search for positive-recurrent solutions, we iterate through the $3^m$ possible overall behaviors.

8

In our running example, we seek a solution of the form

$$\begin{cases} R(4k) = a_k^{(0)} \\ R(4k+1) = a_k^{(1)} \\ R(4k+2) = a_k^{(2)} \\ R(4k+3) = a_k^{(3)}. \end{cases}$$

Going forward, we will assume we are treating the case $A_0 = \infty$, $A_1 = 4$, $A_2 = 0$, and $A_3 = 0$. That is, we are seeking a solution with $\left(a_k^{(0)}\right)$ steep, $\left(a_k^{(1)}\right)$ standard linear, and the other two sequences constant.

## 3.2    Unpacking the Recurrence

Now that we have fixed the forms of the $\left(a_k^{(r)}\right)$, we can use these forms inductively to "unpack" the recurrent form for each $Q(mk+r)$. (The base case of the induction is covered by the fact that we allow an arbitrarily long initial condition.) We start from the innermost calls to $Q$ (i.e., calls without another $Q$ inside them) and work our way outward, eliminating (nearly) all of the the $Q$'s with $k$'s inside them. For an expression of the form $Q(mk+r)$, we replace it by the expression $A_r k + B_r$, where $A_r$ is the variable tracking the growth of $\left(a_k^{(r)}\right)$ and $B_r$ is a symbol. Each step after rewriting the inner calls involves looking at an expression of the form $Q(c)$, $Q(-\infty k-c)$, or $Q(mk-c)$ for some, possibly symbolic, constant $c$. Expressions of the first type are constant; expressions of the second type are zero (by our convention that evaluation at nonpositive indices gives zero). In both of these cases, we can immediately move outward. Expressions of the third type equal $a_{k-t}^{(r)}$ for $r = (-c) \bmod m$ and $t = \left\lceil \frac{c}{m} \right\rceil$. This all depends on the congruence class of $c \bmod m$, since

$$\left\lceil \frac{c}{m} \right\rceil = \begin{cases} \frac{c}{m} & c \equiv 0 \,(\mathrm{mod}\, m) \\ \frac{c+m-(c \bmod m)}{m} & \text{otherwise.} \end{cases}$$

So, at this step, we iterate through all possible congruence classes of these constants.

Once we do this, we replace $Q(mk-c)$ by $A_r(k-t)+B_r$ ( $r = (-c) \bmod m$ and $t = \left\lceil \frac{c}{m} \right\rceil$) for symbolic constant $B_r$. In the case the $\left(a_k^{(r)}\right)$ is steep, this replacement will ensure that the coefficient on $k$ in $mk - Q(mk - c)$ will be $-\infty$. If $\left(a_k^{(r)}\right)$ is not steep, this simply substitutes the formula for $\left(a_k^{(r)}\right)$ into the recurrence. In the case that we are considering an outermost $Q$ and $\left(a_k^{(r)}\right)$ is steep, we do not replace the call to $\left(a_k^{(r)}\right)$; we will leave it unevaluated.

If $Q(n)$ is basic, then the congruence choices we will have to make in this step are easily predetermined.

9

**Proposition 1.** *If $Q(n)$ is basic, then the c values appearing in expressions $Q(mk - c)$ that appear at the outer level when unpacking the recurrence are precisely determined by fixing the values of the $B_r$'s for which $\left(a_k^{(r)}\right)$ is constant.*

*Proof.* If the recurrence for $Q(n)$ is basic, it is of the form

$$Q(n) = P(n) + \sum_{i=1}^{d} \alpha_i Q(n - \beta_i - Q(n - \gamma_i)).$$

Substituting $n = mk + r$, we obtain

$$Q(mk + r) = P(mk + r) + \sum_{i=1}^{d} \alpha_i Q(mk + r - \beta_i - Q(mk + r - \gamma_i)).$$

The innermost expressions are the expressions $Q(mk + r - \gamma_i)$. For notational ease, let $r_i = (r - \gamma_i) \bmod m$. There are three cases to consider:

$Q(mk + r - \gamma_i)$ **is constant:** In this case, $Q(mk + r - \gamma_i) = B_{r_i}$. We then see the expression $Q(mk - \beta_i - B_{r_i})$ at the outer level. Since $\beta_i$ is a predetermined constant, the constant $\beta_i + B_{r_i}$ that appears here is completely determined by fixing $B_{r_i}$. Furthermore, $\left(a_k^{(r_i)}\right)$ is constant; it eventually equals $B_{r_i}$.

$Q(mk + r - \gamma_i)$ **is standard linear:** In this case, $Q(mk + r - \gamma_i) = mk + B_{r_i}$. This leads to the expression $Q(-\beta_i - B_{r_i})$ at the outer level, which is a constant.

$Q(mk + r - \gamma_i)$ **is steep:** In this case, the expression that appears at the outer level in this term is automatically zero.

In summary, only one case results in an outer expression of the form $Q(mk - c)$, and the value of $c$ in that case is completely determined by the value of $B_r$ for an $r$ with $\left(a_k^{(r)}\right)$ constant, as required. $\qquad \square$

The net result of this step is that we can now read off a recurrence system for the component sequences. If the recurrence was basic, we claim that this will be a positive recurrence system. First, all coefficients in the homogeneous part of the system will be nonnegative, because the coefficients on the recursive calls in a basic recurrence are all positive. Also, all constant component sequences will be positive, since we cannot have a solution with infinitely many nonpositive entries (or else we would not be able to explicitly calculate terms). Beyond this, nonhomogeneous parts are built out of recursive calls, which will inductively result in eventually nonnegative polynomials.

In general, in order to guarantee the correctness of our algorithm, we need the nested recurrence to be such that we obtain a positive recurrence system in this step. We stated earlier that the eventual nonnegativity condition on the nohnomogeneous parts is overly restrictive for what we want to do with positive recurrence systems, so it makes sense to

continue to run the rest of this algorithm even without a positive recurrence system. The algorithm may still succeed, but we can no longer guarantee that it will succeed.

In our running example, we are concerned with the congruence classes of $B_2$ and $B_3$ mod 4. There are eight cases to check. Going forward, we will assume we are looking at the case where $B_2 \equiv 0 \,(\mathrm{mod}\,4)$ and $B_3 \equiv 3 \,(\mathrm{mod}\,4)$. Under these assumptions, here is how the recurrence unpacks:

$$
\begin{aligned}
R(4k) &= R(4k - R(4k-1)) + R(4k - R(4k-2)) + R(4k - R(4k-3)) \\
&= R\left(4k - a_{k-1}^{(3)}\right) + R\left(4k - a_{k-1}^{(2)}\right) + R\left(4k - a_{k-1}^{(1)}\right) \\
&= R(4k - 0\,(k-1) - B_3) + R(4k - 0\,(k-1) - B_2) + R(4k - 4\,(k-1) - B_1) \\
&= R(4k - B_3) + R(4k - B_2) + R(4 - B_1) \\
&= a_{k-\frac{B_3+1}{4}}^{(1)} + a_{k-\frac{B_2}{4}}^{(0)} + R(4 - B_1) \\
&= 4\left(k - \frac{B_3+1}{4}\right) + B_1 + a_{k-\frac{B_2}{4}}^{(0)} + R(4 - B_1) \\
&= 4k - B_3 - 1 + B_1 + a_{k-\frac{B_2}{4}}^{(0)} + R(4 - B_1).
\end{aligned}
$$

$$
\begin{aligned}
R(4k+1) &= R(4k+1 - R(4k)) + R(4k+1 - R(4k-1)) + R(4k+1 - R(4k-2)) \\
&= R\left(4k+1 - a_k^{(0)}\right) + R\left(4k+1 - a_{k-1}^{(3)}\right) + R\left(4k+1 - a_{k-1}^{(2)}\right) \\
&= R(4k+1 - \infty k - B_0) + R(4k+1 - 0\,(k-1) - B_3) + R(4k+1 - 0\,(k-1) - B_2) \\
&= R(-\infty k + 1 - B_0) + R(4k+1 - B_3) + R(4k+1 - B_2) \\
&= 0 + a_{k-\frac{B_3+1}{4}}^{(2)} + a_{k-\frac{B_2}{4}}^{(1)} \\
&= 0\left(k - \frac{B_3+1}{4}\right) + B_2 + 4\left(k - \frac{B_2}{4}\right) + B_1 \\
&= B_2 + 4k - B_2 + B_1 \\
&= 4k + B_1.
\end{aligned}
$$

$$
\begin{aligned}
R(4k+2) &= R(4k+2 - R(4k+1)) + R(4k+2 - R(4k)) + R(4k+2 - R(4k-1)) \\
&= R\left(4k+2 - a_k^{(1)}\right) + R\left(4k+2 - a_k^{(0)}\right) + R\left(4k+2 - a_{k-1}^{(3)}\right) \\
&= R(4k+2 - 4k - B_1) + R(4k+2 - \infty k - B_0) + R(4k+2 - 0\,(k-1) - B_3) \\
&= R(2 - B_1) + R(-\infty k + 2 - B_0) + R(4k+2 - B_3) \\
&= R(2 - B_1) + 0 + a_{k-\frac{B_3+1}{4}}^{(3)} \\
&= R(2 - B_1) + 0\left(k - \frac{B_3+1}{4}\right) + B_3 \\
&= R(2 - B_1) + B_3.
\end{aligned}
$$

11

$$
\begin{aligned}
R(4k+3) &= R(4k+3-R(4k+2)) + R(4k+3-R(4k+1)) + R(4k+3-R(4k)) \\
&= R\left(4k+3-a_k^{(2)}\right) + R\left(4k+3-a_k^{(1)}\right) + R\left(4k+3-a_k^{(0)}\right) \\
&= R(4k+3-0k-B_2) + R(4k+3-4k-B_1) + R(4k+3-\infty k-B_0) \\
&= R(4k+3-B_2) + R(3-B_1) + R(-\infty k+3-B_0) \\
&= a^{(3)}_{k-\frac{B_2}{4}} + R(3-B_1) + 0 \\
&= 0\left(k-\frac{B_2}{4}\right) + B_3 + R(3-B_1) \\
&= B_3 + R(3-B_1).
\end{aligned}
$$

## 3.3   Checking for Structural Consistency

The previous step has given us a positive recurrence system that is eventually satisfied by the $\left(a_k^{(r)}\right)$'s. But, this system may not be consistent with the choices we have made. First, we check the following things:

- If $\left(a_k^{(r)}\right)$ is constant, our expression for $Q(mk+r)$ should consist only of constants.

- If $\left(a_k^{(r)}\right)$ is standard linear, our expression for $Q(mk+r)$ should be of the form $mk+c$ for some (possibly complicated) constant $c$.

- If $\left(a_k^{(r)}\right)$ is steep, our expression for $Q(mk+r)$ should have at least one of the following:

  ⋆ A term $dk$ with $d > m$.
  ⋆ A reference to some steep $\left(a_k^{(r')}\right)$.

If any of these is violated for any $r$, there is no solution with the given $A$ values and congruence conditions.

   If all of the above conditions are satisfied, we must determine the nature of each steep $\left(a_k^{(r)}\right)$. In particular, we need to determine if each one is steep linear or superlinear. As a bonus, we will be able to determine the degree of $\left(a_k^{(r)}\right)$ if it is a polynomial. Since the expressions we have for the $\left(a_k^{(r)}\right)$'s form a positive recurrence system, we can use the algorithm from Section 2 to accomplish precisely this task, provided we will start with an eventually positive initial condition. (We will construct this initial condition later.) In running this algorithm, we may find that we actually do not have a solution, as the third case above includes expressions like $a_k^{(r)} = a_{k-1}^{(r)}$, which do not result in steep sequences.

In our example, we verify successfully that $\left(a_k^{(1)}\right)$ is standard linear (the expression we obtained for $R(4k + 1)$ is $4k + B_1$) and that $\left(a_k^{(2)}\right)$ and $\left(a_k^{(3)}\right)$ are constant (expressions $R(2 - B_1) + B_3$ and $B_3 + R(3 - B_1)$ respectively). We now run our algorithm on the positive recurrence system we obtained:

$$
\begin{cases}
a_k^{(0)} = 4k - B_3 - 1 + B_1 + a_{k - \frac{B_2}{4}}^{(0)} + R(4 - B_1) \\
a_k^{(1)} = 4k + B_1 \\
a_k^{(2)} = R(2 - B_1) + B_3 \\
a_k^{(3)} = B_3 + R(3 - B_1).
\end{cases}
$$

The graph $G$ consists of four vertices. Vertex 0 has a loop with weight 1; the other three vertices are isolated. We initialize $d_0 = 1$, $d_1 = 1$, $d_2 = 0$, and $d_3 = 0$. Step 3 doesn't affect any of the vertices, so $G' = G$. Similarly, $\sim$ has no nontrivial relations, so $H \cong G$ via the isomorphism $i \leftrightarrow \{i\}$. When we process vertex $\{0\}$ in $H$, we set $d_{\{0\}} = 2$, and this is the only change made in Step 7. So, we obtain that $\left(a_k^{(0)}\right)$ is quadratic.

## 3.4   Building a Constraint Satisfaction Problem

If our parameters produce a solution, we now know precisely what the structure of that solution must be. At this point, we need to see if a solution can actually be realized.

In order to have a solution, we must check the following:

- If $\left(a_k^{(r)}\right)$ is constant, we must have $B_r > 0$. Otherwise, our solution would have infinitely many nonpositive values. This is not allowed, as we would then not be able to explicitly calculate terms of the sequence.

- If $\left(a_k^{(r)}\right)$ is constant, $B_r$ must equal our expression for $Q(mk + r)$.

- If $\left(a_k^{(r)}\right)$ is standard linear, $B_r$ must equal the constant term in our expression for $Q(mk + r)$.

- If $\left(a_k^{(r)}\right)$ is steep linear, we may need a steepness constraint. This constraint is somewhat more complicated; we describe it below.

- Any constant that we have forced to have a certain congruence mod $m$ must actually have that congruence.

- For any two constants of the form $Q(c)$ and $Q(d)$ that appear, if $c = d$ then $Q(c)$ must equal $Q(d)$.

- If constant $Q(c)$ appears, then if $c \leq 0$ we must have $Q(c) = 0$.

The last two of these restrictions gives a set of conditional constraints to check; the rest of the constraints are unconditional constraints.

As mentioned above, constraining steep linear $\left(a_k^{(r)}\right)$'s to actually be steep requires a more complicated constraint. This stems from the fact that steep linear $\left(a_k^{(r)}\right)$'s can arise in three different ways. The steep linear $\left(a_k^{(r)}\right)$'s are a subset of the linear $\left(a_k^{(r)}\right)$'s. In terms of the positive recurrence system algorithm, the linear $\left(a_k^{(r)}\right)$'s are the ones whose vertices are labeled 1 when the algorithm terminates. The following are all the ways this could happen, in terms of the graphs $G'$ and $H$ in that algorithm.

1. The vertex $r$ could be labeled 1 because the expression for $Q(mk + r)$ is a degree-1 polynomial.

2. The vertex $r$ could be labeled 1 because it is not in a cycle in $G'$ and, when it came time to assign $\bar{r}$ its final label, the largest label in $H$ it pointed to was a 1.

3. The vertex $r$ could be labeled 1 because it is in a cycle in $G'$ and, when it came time to assign $\bar{r}$ its final label, the largest label in $H$ it pointed to was a 0 (or it pointed to no other vertices in $H$).

In Case 1, $\left(a_k^{(r)}\right)$ is steep linear if and only if the leading coefficient of that polynomial is greater than $m$. We already checked this in our structural consistency check, so if $\left(a_k^{(r)}\right)$ is linear because of Case 1, we need no steepness constraint. In Case 2, we have that $r$ is pointing to something else linear. But, our unpacking step would have removed all references to standard linear $\left(a_k^{(r')}\right)$'s. So, in Case 2, every $\left(a_k^{(r')}\right)$ that $\left(a_k^{(r)}\right)$ still refers to must be steep linear. This immediately forces $\left(a_k^{(r)}\right)$ itself to be steep linear without imposing any extra constraints.

This leaves only Case 3. In this case, $r$ is in a directed cycle in $G'$, say $r = r_0, r_1, r_2, \ldots, r_{t-1}, r_t = r$. Each of the corresponding sequences has an expression of the form

$$a_k^{(r_i)} = c_i + a_{k-e_i}^{(r_{i+1})}.$$

Repeated substitution yields the formula

$$a_k^{(r)} = \sum_{i=0}^{t-1} c_i + a_{k-\sum_{i=0}^{t-1} e_i}^{(r)}$$

for some constants $e_i$. We require that $\left(a_k^{(r)}\right)$ be steep. This will be accomplished if we have that

$$\sum_{i=0}^{t-1} c_i > m \sum_{i=0}^{t-1} e_i.$$

So, this is the steepness constraint we add in Case 3. In particular, we arrive at the same constraint for all functions in a given equivalence class.

In our example, we obtain the following constraints:

- $\left(a_k^{(0)}\right)$ is superlinear, so there are no constraints associated to it.

- $\left(a_k^{(1)}\right)$ is standard linear, and we have $R(4k+1) = 4k+B_1$. This gives us the constraint $B_1 = B_1$. (This constraint is tautological, but this is okay.)

- $\left(a_k^{(2)}\right)$ is constant, and we have $R(4k+2) = R(2-B_1)+B_3$. This gives us the following constraints:

  ⋆ $B_2 > 0$
  ⋆ $B_2 = R(2 - B_1) + B_3$.

- $\left(a_k^{(3)}\right)$ is constant, and we have $R(4k+3) = B_3+R(3-B_1)$. This gives us the following constraints:

  ⋆ $B_3 > 0$
  ⋆ $B_3 = B_3 + R(3 - B_1)$.

- Our congruence constraints are

  ⋆ $B_2 \equiv 0 \,(\mathrm{mod}\,4)$
  ⋆ $B_3 \equiv 3 \,(\mathrm{mod}\,4)$.

- Our conditional constraints are

  ⋆ If $2 - B_1 = 3 - B_1$, then $R(2 - B_1)$ must equal $R(3 - B_1)$.
  ⋆ If $2 - B_1 \le 0$, then $R(2 - B_1) = 0$.
  ⋆ If $3 - B_1 \le 0$, then $R(3 - B_1) = 0$.

## 3.5 Solving the Constraint Satisfaction Problem

The recurrence solution we are seeking should exist if and only if this constraint system is satisfiable. The system of unconditional constraints is almost an integer program. The following modifications can turn it into an integer program:

- Since all variables are integers, strict inequalities of the form $x > y$ can be made loose by replacing them by the equivalent inequality $x \ge y + 1$.

- Congruence constraints can be converted to equality constraints via the introduction of auxiliary variables. Namely, $x \equiv y \,(\mathrm{mod}\,m)$ is the same constraint as $x = Km + y$, where $K$ is a new auxiliary variable.

Furthermore, the conditional constraints can be incorporated into the integer program. For each constraint of the form $(c = d) \Rightarrow (e = f)$, we consider three cases. (If one fails, we try the next one.)

- Add the constraints $c = d$ and $e = f$.

- Add the constraint $c <= d - 1$.

- Add the constraint $c >= d + 1$.

And, for each constraint of the form $(c \leq d) \Rightarrow (e = f)$, we consider two cases.

- Add the constraints $c \leq d$ and $e = f$.

- Add the constraint $c >= d + 1$.

Maple has a built-in procedure for satisfying *linear* integer programs. Since we are only considering linear nested recurrences, the program we obtain is, in fact, linear. Integer linear programming is an NP-hard problem [10]. But, experimentally, the instances that arise in this context seem not to be very hard. Heuristically guessing $B_r$ values that are far apart seems to make satisfying the constraints be a quick process. In particular, the constraints we obtain are typically satisfiable unless there is some obvious reason why they should not be.

The following assignments satisfy our example constraint system:

- $B_1 = 0$

- $B_2 = 4$

- $B_3 = 3$

- $R(2) = 1 \, (= R(2 - B_1))$

- $R(3) = 0 \, (= R(3 - B_1))$.

This means we have the following eventual solution:

$$
\begin{cases}
R(4k) = 4k - 3 - 1 + 0 + a_{k-1}^{(0)} + R(4) = R(4k - 4) + 4k + R(4) - 4 \\
R(4k + 1) = 4k \\
R(4k + 2) = 4 \\
R(4k + 3) = 3.
\end{cases}
$$

Note that these constraints have other satisfying values, and each other satisfaction leads to another eventual solution to the recurrence.

## 3.6   Constructing an Initial Condition

We now have an eventual solution to our recurrence. But, it is quite likely that there will need to be some initial values to make the solution exist. These initial value requirements can arise from a number of sources:

- An expression of the form $Q(n - c)$ in the recurrence probably requires that the initial condition have length at least $c$. (The only case where it would not require this is if the solution we found has $Q(n - c) = 0$ for $n \leq c$.)

- There are sometimes equality constraints involving $Q(c)$ for some constant $c$. Unless the value required of $Q(c)$ is what it must be anyway in the eventual solution, the initial condition must have length at least $c$.

- In constructing our solution, we used the fact that steep sequences $\left(a_k^{(r)}\right)$ are eventually greater than $k + c$ for any constant $c$. But, for fixed $c$ they can be smaller for small $k$. The initial condition must be long enough to allow all the steep sequences to become sufficiently large.

The question now is, when does the initial condition end? We try to find as generic an initial condition as possible. This means that we try to minimize the length of the initial condition while including as many free parameters as we can. As a result, the initial condition we find may include symbols, and some of these symbols may have (obviously satisfiable) constraints placed on them. (To find an explicit initial condition, replace each symbol by a value satisfying any constraints on it.) Here is an outline of a procedure for finding an initial condition. This is a more streamlined version of the procedure in the Maple package. That version is more complicated, as it tries to use various tricks to shorten the initial condition.

1. Look for the largest $c_0$ such that $Q(c_0)$ appears in a constraint from the constraint satisfaction problem. Initialize the list $L$ to the symbolic list $[Q(1), Q(2), \ldots, Q(c_0)]$.

2. The constraints involving expressions of the form $Q(c)$ constitute a linear system. Solve this system for as many constants $Q(c)$ as possible. (If the system is underdetermined, these solutions may be in terms of others of these constants.) Substitute these solution values into $L$ for their respective constants.

3. Denote by $\gamma$ the maximum number such that $Q(n - \gamma)$ appears in the recurrence. Generate the next $\max(m, \gamma)$ terms of the recurrence $Q$ with the initial condition $L$. If $Q$ is ever evaluated at a symbolic index, assume that this index is nonpositive and record the resulting constraint on the symbols. (If the sequence dies during this computation, go to step 6.)

4. If $\max(m, \gamma)$ terms were generated, look at the terms in the steep positions and determine which future terms directly depend on them. If the computations involving the steep terms used to compute the later terms all result in 0, then the terms in the

17

steep positions are sufficiently large. (If these terms are symbolic, this may result in inequality constraints on some constants.) Otherwise, the steep terms are not sufficiently large, so go to Step 6.

5. If the non-steep terms agree with their eventual values, then the initial condition is complete. In this case, return $L$ along with the constraints that were imposed on the symbols when generating the $m$ terms.

6. If not enough terms were generated, terms were not large enough, or some term did not agree with its eventual value, then extend $L$ by one term. In that position, put the value of the eventual solution. (If that term is a member of a steep sequence, keep that term symbolic.) Then, forget any constraints imposed on symbols in step 4 and return to Step 3.

This algorithm generates an initial condition for the eventual solution found in the previous steps.

For our example, the largest $c$ such that $R(c)$ appears in a constraint is $c = 3$. So, we initialize $L = [R(1), R(2), R(3)]$. We see that $R(2) = 1$ and $R(3) = 0$, so this changes $L$ to $[R(1), 1, 0]$. In this case $m = 4$, and $\gamma = 3$ since $R(n-3)$ appears in the recurrence. We now start extending $L$:

- We try to compute $R(4)$:

$$\begin{aligned} R(4) &= R(4 - R(3)) + R(4 - R(2)) + R(4 - R(1)) \\ &= R(4 - 0) + R(4 - 1) + R(4 - R(1)) \\ &= R(4) + R(3) + R(4 - R(1)). \end{aligned}$$

We see that $R(4)$ is defined in terms of $R(4)$, which is not allowed. So, the sequence dies immediately. We failed to generate 4 terms, so we extend $L$. The fourth term would be part of the quadratic (steep) sequence, so we keep $R(4)$ symbolic. We now have $L = [R(1), 1, 0, R(4)]$

- The next two times we try to compute a term of $R$, the sequence will die immediately. This is the case because the 0 that killed our sequence in the $R(4)$ case is referenced again when computing $R(5)$ and when computing $R(6)$. The predicted values for $R(5)$ and $R(6)$ are both 4. So, $L$ will be extended to $[R(1), 1, 0, R(4), 4]$ and then to $[R(1), 1, 0, R(4), 4, 4]$.

- We try to compute $R(7)$:

$$\begin{aligned} R(7) &= R(7 - R(6)) + R(7 - R(5)) + R(7 - R(4)) \\ &= R(7 - 4) + R(7 - 4) + R(7 - R(4)) \\ &= R(3) + R(3) + R(7 - R(4)) \\ &= 0 + 0 + R(7 - R(4)) \\ &= R(7 - R(4)). \end{aligned}$$

In order to evaluate $R(7 - R(4))$, we assume that $R(4) \geq 7$, so this term is zero. This gives $R(7) = 0$. Then, when we try to compute $R(8)$, we will fail. The predicted value for $R(7)$ is 3, so we extend $L$ to $[R(1), 1, 0, R(4), 4, 4, 3]$.

- We try to compute $R(8)$:

$$
\begin{aligned}
R(8) &= R(8 - R(7)) + R(8 - R(6)) + R(8 - R(5)) \\
&= R(8 - 3) + R(8 - 4) + R(8 - 4) \\
&= R(5) + R(4) + R(4) \\
&= 4 + 2R(4).
\end{aligned}
$$

This is a symbolic answer, but that is permitted. The term $R(8)$ would be in the quadratic sequence, so this agrees with the eventual solution provided $R(4)$ is sufficiently large. We now try to compute $R(9)$:

$$
\begin{aligned}
R(9) &= R(9 - R(8)) + R(9 - R(7)) + R(9 - R(6)) \\
&= R(9 - 4 - 2R(4)) + R(9 - 3) + R(9 - 4) \\
&= R(5 - 2R(4)) + R(6) + R(5) \\
&= R(5 - 2R(4)) + 4 + 4 \\
&= 8 + R(5 - 2R(4)).
\end{aligned}
$$

If $2R(4) \geq 5$, we obtain $R(9) = 8$, which is its value in the eventual solution. If we compute $R(10)$, we find that it equals its predicted value of 4 provided that $2R(4) \geq 6$. Then, if we compute $R(11)$, we find that it equals its predicted value of 3 provided that $2R(4) \geq 7$. Combining all of this, we find that, for all $R(1)$ and all $R(4) \geq 4$, $[R(1), 1, 0, R(4), 4, 4, 3]$ is an initial condition that yields the eventual solution we found in the previous step.

# 4    Some Findings

As our running example illustrates, it is possible to obtain quasi-quadratic solutions to Hofstadter-like recurrences. This begs the question as to whether higher degree quasipolynomials are possible. If they are, the algorithm will find them. It turns out that there are quasipolynomial solutions to the Hofstadter $Q$-recurrence of every positive degree [5]. In addition, there are solutions that include both quadratic and exponential subsequences, such as the sequence obtained from the Hofstadter $Q$-recurrence with $[9, 0, 0, 0, 7, 9, 9, 10, 4, 9, 9, 3]$ as the initial condition [12, A275153]. It is likely that a construction similar to the one for arbitrary degree quasipolynomials [5] will also lead to examples including higher degree polynomials along with exponentials. There are also solutions to the Hofstadter $Q$-recurrence with linear subsequences with slopes greater than 1, and such subsequences can be obtained by any of the three ways mentioned in Subsection 3.4:

- The length-45 initial condition

$$[0, 4, -40, -9, 8, -8, 7, 1, 5, 13, -24, -1, 8, 8, 8, 1, 5, 13, -8, 7, 8, 8, 23, 1, 5, 13, 8, 15,$$
$$8, 16, 31, 1, 5, 13, 24, 23, 8, 24, 39, 1, 5, 13, 40, 31, 8]$$

  leads to a period-8 solution with $Q(8n + 3) = 16n - 40$. This is the case because unpacking $Q(8n+3)$ involves adding two standard linear terms together [12, A275361].

- The length-16 initial condition

$$[-9, 2, 9, 2, 0, 7, 9, 10, 3, 0, 2, 9, 2, 9, 9, 9]$$

  leads to a period-9 solution where $Q(9n + 2)$ and $Q(9n + 8)$ both have slope 10. But, $Q(9n + 2)$ has slope 10 because unpacking it yields $Q(9n - 1)$ plus a constant [12, A275362].

- In the previous example, $Q(9n + 8)$ has slope 10 because unpacking it results in $10 + Q(9n - 1)$. This appears to be, by far, the most common way steep linear solutions arise.

Our algorithm was also used to examine what sorts of recurrences can be satisfied by the exponential subsequences. This led to the observation that any homogeneous linear recurrence with positive coefficients that sum to at least 2 can be realized [4].

We have used our algorithm to fully explore positive recurrent solution families to the Hofstadter $Q$-recurrence with small periods. Given a solution to the Hofstadter $Q$-recurrence, any shift of this solution is also a solution, since the recurrence only depends on the relative indices of the terms (and not the absolute indices). So, solution families found by the algorithm can be considered as-is or modulo this shifting operation. There are no period-1 positive recurrent solutions to the Hofstadter $Q$-recurrence, and there are two families (one modulo shifting) of period 2 solutions. These solutions consist of one constant sequence interleaved with one standard linear sequence. For example, the initial condition $[2, 2]$ gives rise to the sequence $2, 2, 4, 2, 6, 2, 8, 2, \ldots$ [12, A275365]. There are 12 period 3 solution families (4 modulo shifting). One of these families includes Golomb's solution, and another includes Ruskey's sequence. The other two families consist of eventually quasilinear solutions with two constant sequences and one standard linear sequence (and appear to be related to each other). One of these families includes sequence A264756 [12]. There are 12 period 4 families (5 modulo shifting), all of which are quasilinear with constant and standard linear sequences. Some of these families include the period 2 solutions, but each such family also include additional solutions that do not have period 2. There are 35 period 5 families (7 modulo shifting). Again, all of these are quasilinear. But, one of these families has a steep linear subsequence [12, A269328]. There is a lot more variety beginning at period 6. There are 294 solution families (86 modulo shifting) in this case. These solutions include quadratics as well as mixing of exponentials with steep linears. A file containing information on all of the solution families examined, modulo shifting, can be found at http://www.math.rutgers.edu/~nhf12/research/hof_small_periods.txt.

In addition, we found positive recurrent solutions to other recurrences, including the Conolly recurrence [12, A275363] and the Hofstadter-Conway recurrence [12, A052928]. This second case is notable because the solution has period 2 with both subsequences linear. This can happen because the Hofstadter-Conway recurrence is not basic. As a result, the congruence classes of the constant terms in the linear polynomials end up determining much of the behavior.

# 5    Future Work

This algorithm has various positivity requirements in order to ensure that it runs correctly. Nested recurrences are allowed to have negative terms; these would inherently violate our algorithm's conditions. It would be useful to have a version of this algorithm that can handle these cases. Also, we are not currently allowing implicit solutions. Allowing these would give a method of handling sequences with infinitely many nonpositive entries, which may or may not be helpful.

In addition, the algorithm only works for linear recurrences. But, it makes perfect sense to look for solutions to nonlinear nested recurrences that consist of simpler interleaved sequences. Most of the steps of this algorithm still work when the recurrences can be nonlinear. Whether that means that $Q$ terms are multiplied by each other or raised to powers, the only thing that changes significantly is the algorithm for determining the orders of growth of the subsequences. Nonlinearity introduces some extra complications. For example, the recurrence $C(n) = C(n - C(n-1))^2 - 2C(n - C(n-1)) + 2$ has constant solutions (constant 1 and constant 2), but it also has a solution

$$\begin{cases} C(2k) = 2^{2^{k-1}} + 1 \\ C(2k+1) = 2. \end{cases}$$

The distinction here comes from the fact that repeated squaring causes numbers to rapidly grow, unless the initial number was 0 or 1. Such concerns do not arise when dealing with only linear recurrences. Hence, it would be useful to have a version of the algorithm that can handle nonlinear recurrences. Of course, we would have to modify what we are looking for, since the example here includes a doubly exponential subsequence, which cannot possibly be a component of a positive-recurrent sequence.

# A    Appendix: Proof of Claim 1

In this section, we will prove Claim 1:

**Claim 1.** *Suppose we have a positive recurrence system with m component sequences, along with an eventually positive initial condition. Let $1 \le r \le m$ be an integer. If $d_{\bar{r}} < \infty$, then $a_k^{(r)} = \Theta\left(k^{d_{\bar{r}}}\right)$. If $d_{\bar{r}} = \infty$, then $a_k^{(r)}$ grows exponentially.*

*Proof.* Since each component sequence has a rational generating function, no component sequence can possibly grow faster than exponentially. So, proving only lower bounds when sequences should grow exponentially will suffice. (We take advantage of this simplification throughout this proof.)

For each vertex $r$ in a directed graph $F$, define the *potential* of $r$ (denoted $\phi_F(r)$) as the sum of the lengths of all directed cycles in $F$ containing $r$ (or 0 if $r$ is not in any cycles). Also, for each $r \in [m]$, define $J(r)$ as the set of immediate successors of $r$ in $G'$. We will prove Claim 1 by induction on $\phi_G(r)$. We will examine three cases (essentially a special case, a base case, and an inductive step).

$\phi_G(r) = 0$: In this case, $r$ is in no cycles in $G$, and $\bar{r} = \{r\}$. Also, every successor of $r$ in $G$ is in $J(r)$, and every coefficient $\alpha_{r,\ell,j}$ with $j \notin J(r)$ is zero. Inductively, suppose that any $j \in J(r)$ satisfies Claim 1. (The base case of $J(r) = \emptyset$ is implicitly included here.) This all gives

$$a_k^{(r)} = P_r(k) + \sum_{\ell=1}^{d} \sum_{j \in J(r)} \alpha_{r,\ell,j} a_{k-\ell}^{(j)}.$$

Let $D = \max\left\{d_{\bar{j}} : j \in J(r)\right\}$ (or $D = -\infty$ if this set is empty). If $D = \infty$, then, by induction, some sequence $a_k^{(j)}$ with $j \in J(r)$ grows exponentially. As a result, $a_k^{(r)}$ grows exponentially, and we have $d_{\bar{r}} = \infty$, as required. On the other hand, if $D < \infty$, then we have $a_k^{(r)} = \Theta\left(k^{\max(D,\deg(P_r))}\right)$. Since $d_{\bar{r}} = \max\left(D, \deg(P_r)\right)$ in this case, we have $a_k^{(r)} = \Theta\left(k^{d_{\bar{r}}}\right)$, as required.

$\phi_G(r) = 1$: In this case, the only cycle of $G$ that $r$ is in is a self-loop, and $\bar{r} = \{r\}$. Also, every non-$r$ successor of $r$ in $G$ is in $J(r)$, and every coefficient $\alpha_{r,\ell,j}$ with $j \notin \{r\} \cup J(r)$ is zero. Inductively, suppose that any $j \in J(r)$ satisfies Claim 1. (The base case of $J(r) = \emptyset$ is implicitly included here.) This all gives,

$$a_k^{(r)} = P_r(k) + \sum_{\ell=1}^{d} \alpha_{r,\ell,r} a_{k-\ell}^{(r)} + \sum_{\ell=1}^{d} \sum_{j \in J(r)} \alpha_{r,\ell,j} a_{k-\ell}^{(j)}.$$

Let $D = \max\left\{d_{\bar{j}} : j \in J(r)\right\}$ (or $D = -\infty$ if this set is empty). If $D = \infty$, then, by induction, some sequence $a_k^{(j)}$ with $j \in J(r)$ grows exponentially. As a result, $a_k^{(r)}$ grows exponentially, and we have $d_{\bar{r}} = \infty$, as required. On the other hand, if $D < \infty$, then we have

$$a_k^{(r)} = \sum_{\ell=1}^{d} \alpha_{r,\ell,r} a_{k-\ell}^{(r)} + \Theta\left(k^{\max(D,\deg(P_r))}\right).$$

This is a nonhomogeneous linear recurrence for $a_k^{(r)}$ with characteristic polynomial

$$P(x) = x^d - \sum_{\ell=1}^{d} \alpha_{r,\ell,r} x^{d-\ell}$$

22

If the weight on the self-loop on $r$ is 1, then

$$\sum_{\ell=1}^{d} \alpha_{r,\ell,r} = 1.$$

So, $p(1) = 0$, and we also note that this is a simple root, and the other nonzero roots of $p$ are all simple and roots of unity. By the theory of linear recurrences, this results in a solution where $a_k^{(r)} = \Theta\left(k^{1+\max(D,\deg(P_r))}\right)$. Since $d_{\bar{r}} = 1 + \max(D, \deg(P_r))$ in this case, we have $a_k^{(r)} = \Theta\left(k^{d_{\bar{r}}}\right)$, as required.

If the weight on the self-loop on $r$ is greater than 1, then

$$\sum_{\ell=1}^{d} \alpha_{r,\ell,r} > 1.$$

So, $p(1) < 0$, but $p(x) \to \infty$ as $x \to \infty$. So, $p$ has a real root that is greater than 1. By the theory of linear recurrences, this results in a solution where $a_k^{(r)}$ grows exponentially. And, $d_{\bar{r}} = \infty$ in this case, as required.

$\phi_G(r) > 1$: Here $r$ is in some cycle of length at least 2. Consider such a cycle $C$. Let $s$ denote the immediate successor of $r$, and let $t$ denote the immediate successor of $s$. We will construct a new positive recurrence system, and if we run our algorithm on this new system, we will obtain a graph in step 1. Call this graph $\tilde{G}$. The system and $\tilde{G}$ will have the following properties:

- The system will have $m - 1$ component sequences

$$b_k^{(1)}, b_k^{(2)}, \ldots, b_k^{(s-2)}, b_k^{(s-1)}, b_k^{(s+1)}, b_k^{(s+2)}, \ldots, b_k^{(m)},$$

  one corresponding to each sequence $a_k^{(i)}$ with $i \neq s$.
- For every $i \neq s$, if $b_k^{(i)}$ is given the same initial condition as $a_k^{(i)}$, then $b_k^{(i)} \leq a_k^{(i)}$.
- If $s$ is in a unique cycle in $G$, then $b_k^{(r)} = a_k^{(r)}$.
- We will have $\phi_{\tilde{G}}(r) < \phi_G(r)$.

Hence, if we inductively assume that Claim 1 holds for any component sequence $r'$ of smaller potential in *any* positive recurrence system, this construction will complete the proof. (The properties are sufficient to show that $a_k^{(r)}$ grows at least as fast as it should, and the equality case handles the sub-exponential component sequences.)

We have

$$a_k^{(s)} = P_s(k) + \sum_{\ell=1}^{d} \sum_{j=1}^{m} \alpha_{s,\ell,j} a_{k-\ell}^{(j)} \geq P_s(k) + \sum_{\ell=1}^{d} \left( \alpha_{s,\ell,t} a_{k-\ell}^{(t)} + \sum_{j \in J(s)} \alpha_{s,\ell,j} a_{k-\ell}^{(j)} \right).$$

23

If $s$ is in a unique cycle in $G$, then every immediate successor of $s$ other than $t$ is in $J(s)$. This means that all dropped terms in going from the equality to the inequality were in fact zero, so we actually have equality here in this case.

We also have

$$a_k^{(r)} = P_r(k) + \sum_{\ell=1}^{d}\sum_{j=1}^{m} \alpha_{r,\ell,j} a_{k-\ell}^{(j)} = P_r(k) + \sum_{\ell=1}^{d}\left(\alpha_{r,\ell,s} a_{k-\ell}^{(s)} + \sum_{j\in[m]\setminus\{s\}} \alpha_{r,\ell,j} a_{k-\ell}^{(j)}\right).$$

Let $J' = [m] \setminus (\{t\} \cup J(s))$. Combining everything together yields

$$a_k^{(r)} \geq P_r(k) + \sum_{\ell_1=1}^{d}\left(\alpha_{r,\ell_1,s}\left(P_s(k-\ell_1) + \sum_{\ell_2=1}^{d}\left(\alpha_{s,\ell_2,t} a_{k-\ell_1-\ell_2}^{(t)} + \sum_{j\in J(s)} \alpha_{s,\ell_2,j} a_{k-\ell_1-\ell_2}^{(j)}\right)\right)\right.$$

$$\left. + \sum_{j\in[m]\setminus\{s\}} \alpha_{r,\ell_1,j} a_{k-\ell_1}^{(j)}\right)$$

$$= \left(P_r(k) + \sum_{\ell=1}^{d}\alpha_{r,\ell,s} P_s(k-\ell)\right) + \sum_{\ell_1=1}^{d}\left(\alpha_{r,\ell_1,t} a_{k-\ell_1}^{(t)} + \sum_{\ell_2=1}^{d}\alpha_{s,\ell_2,t} a_{k-\ell_1-\ell_2}^{(t)}\right)$$

$$+ \sum_{\ell_1=1}^{d}\sum_{j\in J(s)}\left(\alpha_{r,\ell_1,j} a_{k-\ell_1}^{(j)} + \sum_{\ell_2=1}^{d}\alpha_{s,\ell_2,j} a_{k-\ell_1-\ell_2}^{(j)}\right) + \sum_{\ell=1}^{d}\sum_{j\in J'}\alpha_{r,\ell,j} a_{k-\ell}^{(j)}.$$

Again, if $s$ is in a unique cycle in $G$, we actually have equality here.

Our new positive recurrence system is obtained from the current one as follows:

- The equation for $b_k^{(i)}$ for $i \notin \{r,s\}$ is

$$b_k^{(i)} = P_i(k) + \sum_{\ell=1}^{d}\sum_{j\in[m]\setminus\{s\}}\alpha_{i,\ell,j} b_{k-\ell}^{(j)}.$$

- We have

$$b_k^{(r)} = \left(P_r(k) + \sum_{\ell=1}^{d}\alpha_{r,\ell,s} P_s(k-\ell)\right) + \sum_{\ell_1=1}^{d}\left(\alpha_{r,\ell_1,t} b_{k-\ell_1}^{(t)} + \sum_{\ell_2=1}^{d}\alpha_{s,\ell_2,t} b_{k-\ell_1-\ell_2}^{(t)}\right)$$

$$+ \sum_{\ell_1=1}^{d}\sum_{j\in J(s)}\left(\alpha_{r,\ell_1,j} b_{k-\ell_1}^{(j)} + \sum_{\ell_2=1}^{d}\alpha_{s,\ell_2,j} b_{k-\ell_1-\ell_2}^{(j)}\right) + \sum_{\ell=1}^{d}\sum_{j\in J'}\alpha_{r,\ell,j} b_{k-\ell}^{(j)}.$$

In the definition of $b_k^{(r)}$, the first part is the nonhomogeneous part: a polynomial of degree $\max\left(\deg\left(P_r\right), \deg\left(P_s\right)\right)$. The second part describes the references to $\left(b_k^{(t)}\right)$.

The third part describes the references to $\left(b_k^{(j)}\right)$ where $j \in J(s)$. The final part describes all the other references. (Note that there are no references to the nonexistent sequence $\left(b_k^{(s)}\right)$.)

From this construction and our calculation, we notice that the first three desired properties of this new system definitely hold. We can also see how $\tilde{G}$ is obtained from $G$:

- First, delete all incoming arcs to $s$, other than the one from $r$. (This is accomplished by the first bullet point above.)

- Then, contract the arc $rs$. (This is accomplished by the second bullet point above.)

- If any multiple arcs were created in the previous step, replace them by a single arc whose weight is the sum of the weights of the arcs being replaced. (This would come up, for example, if $r$ has a self-loop and is also in a 2-cycle, with $s$ being the other vertex in that cycle. After contracting $rs$, $r$ would have two self-loops, which then need to be combined.)

From this description, it is clear that every cycle in $G$ containing $r$ corresponds to a cycle in $\tilde{G}$ containing $r$, and the latter cycle can be no longer than the original. The cycle corresponding to $C$ in $\tilde{G}$ is strictly shorter than $C$, since $s$ has been removed from it. So, $\phi_{\tilde{G}}(r) < \phi_G(r)$, as required.

$\square$

# Acknowledgements

# References

[1] B. Balamohan, A. Kuznetsov, and Stephen Tanny, *On the behavior of a variant of Hofstadter's Q-sequence*, J. Integer Sequences **10** (2007), 29.

[2] B.W. Conolly, *Meta-Fibonacci sequences, Chapter XII in S. Vajda, Fibonacci & Lucas Numbers, and the Golden Section*, Ellis Horwood Limited, 1989.

[3] Alejandro Erickson, Abraham Isgur, Bradley W. Jackson, Frank Ruskey, and Stephen M. Tanny, *Nested recurrence relations with Conolly-like solutions*, SIAM Journal on Discrete Mathematics **26** (2012), no. 1, 206–238.

[4] Nathan Fox, *Linear recurrent subsequences of generalized meta-Fibonacci sequences*, Journal of Difference Equations and Applications (2016), 1–8.

[5] _____, *Quasipolynomial Solutions to the Hofstadter Q-Recurrence*, arXiv preprint arXiv:1511.06484 (2015).

[6] S.W. Golomb, *Discrete Chaos: Sequences Satisfying "Strange" Recursions*, unpublished manuscript, 1991.

[7] Douglas Hofstadter, *Gödel, Escher, Bach: an Eternal Golden Braid*, Penguin Books, 1979.

[8] A. Isgur, R. Lech, S. Moore, S. Tanny, Y. Verberne, and Y. Zhang, *Constructing New Families of Nested Recursions with Slow Solutions*, SIAM Journal on Discrete Mathematics **30** (2016), no. 2, 1128–1147.

[9] Abraham Isgur, David Reiss, and Stephen Tanny, *Trees and meta-Fibonacci sequences*, The Electronic Journal of Combinatorics **16** (2009), no. R129, 1.

[10] Richard M. Karp, *Reducibility among combinatorial problems*, Complexity of computer computations, 1972, pp. 85–103.

[11] Colin L Mallows, *Conway's challenge sequence*, American Mathematical Monthly **98** (1991), no. 1, 5–20.

[12] *OEIS Foundation Inc.*, The On-Line Encyclopedia of Integer Sequences, 2016. http://oeis.org/.

[13] F. Ruskey, *Fibonacci Meets Hofstadter*, The Fibonacci Quarterly **49** (2011), no. 3, 227-230.

[14] Stephen M Tanny, *A well-behaved cousin of the Hofstadter sequence*, Discrete Mathematics **105** (1992), no. 1, 227–239.