On the seeds and the great-grandchildren of a numerical semigroup

Maria Bras-Amorós*

*Universitat Rovira i Virgili, Av. Països Catalans 26, 43007, Tarragona, Catalonia, Spain.
E-mail adress: maria.bras@urv.cat

August 11, 2023

Abstract

We present a revisit of the seeds algorithm to explore the semigroup tree. First, an equivalent definition of seed is presented, which seems easier to manage. Second, we determine the seeds of semigroups with at most three left elements. And third, we find the great-grandchildren of any numerical semigroup in terms of its seeds.

The RGD algorithm is the fastest known algorithm at the moment. But if one compares the originary seeds algorithm with the RGD algorithm, one observes that the seeds algorithm uses more elaborated mathematical tools while the RGD algorithm uses data structures that are better adapted to the final C implementations. For genera up to around one half of the maximum size of native integers, the newly defined seeds algorithm performs significantly better than the RGD algorithm. For future compilators allowing larger native sized integers this may constitute a powerful tool to explore the semigroup tree up to genera never explored before. The new seeds algorithm uses bitwise integer operations, the knowledge of the seeds of semigroups with at most three left elements and of the great-grandchildren of any numerical semigroup, apart from techniques such as parallelization and depth first search as wisely introduced in this context by Fromentin and Hivert.

The algorithm has been used to prove that there are no Eliahou semigroups of genus 66, hence proving the Wilf conjecture for genus up to 66. We also found three Eliahou semigroups of genus 67. One of these semigroups is neither of Eliahou-Fromentin type, nor of Delgado's type. However, it is a member of a new family suggested by Shalom Eliahou.

MSC (2010): Primary 06F05, 20M14; Secondary 05A99, 68W30

1 Introduction

A *numerical semigroup* is a cofinite submonoid of $\mathbb{N}$. The elements in its complement in $\mathbb{N}$ are denoted the *gaps* of the semigroup and the *genus* of the semigroup is the number of its gaps. There is a finite number of semigroups of each given genus. A major problem in the theory of numerical semigroups is the exhaustive listing of all numerical semigroups for increasing values of the genus. See [15] for a general reference on numerical semigroups.

The *primitive elements* (or minimal generators) of a numerical semigroup are those elements of the semigroup that can not be obtained as a sum of two smaller semigroup elements. If we take

away a primitive element from a numerical semigroup we obtain another semigroup with genus increased by one. The elements of the semigroup that are smaller than some gap are denoted the *left elements* of the semigroup. The primitive elements that are larger than the largest gap are called *right generators*. We can organize all numerical semigroups in an infinite tree rooted at $\mathbb{N}$ and such that the children of a node are the semigroups obtained taking away one by one its right generators.

The tree of semigroups was first extensively explored in [1], in this case using brute force to find right generators. In [2] it was noticed that the search of right generators of a child can be restricted to the right generators of its parent and to one further element. Indeed, suppose that λ_1 is the smallest nonzero non-gap (i.e. the *multiplicity*) of a numerical semigroup Λ. If the gaps of Λ are not all in a row (i.e. if Λ is not *ordinary*), when we take away one right generator λ of Λ, the right generators of the new semigroup $\Lambda \setminus \{\lambda\}$ all belong to the set of right generators of Λ except the element $\lambda+\lambda_1$, should it be a primitive element of the new semigroup. If $\lambda+\lambda_1$ is a primitive element of the child, then λ is called a *strong* generator of Λ. Otherwise it is called a *weak* generator of Λ. Fromentin and Hivert presented in [14, 13] a very efficient algorithm using parallel computation and depth first search, representing semigroups by their decomposition numbers. Suppose that the numerical semigroup Λ is $\{\lambda_0 = 0 < \lambda_1 < \dots\}$. The sequence of decomposition numbers of Λ can be defined by $d_i = \left\lceil \frac{\#\{\lambda_j \in \Lambda : \lambda_i - \lambda_j \in \Lambda\}}{2} \right\rceil$. See its relationship with the ν sequence in [15, Exercise 2.16] and references therein. In [5] an algorithm was presented using the notion of *seed*. An element λ_s larger than the largest gap is a p-th order seed of Λ if $\lambda_s + \lambda_p \neq \lambda_i + \lambda_j$ for all $p < i \leqslant j < s$. Now, right generators are order-0 seeds and they are strong generators if and only if they are order-1 seeds. From another perspective, in [6, 4] a very simple but very efficient algorithm was presented, using the idea of right-generators descendant of a numerical semigroup. The *right-generators descendant* of a numerical semigroup Λ is the semigroup obtained by taking away from Λ *all* its right generators. The so-called RGD algorithm is at the moment the most efficient algorithm.

We present a revisit of the seeds algorithm to explore the semigroup tree. First, an equivalent definition of seed is presented, which seems easier to manage. Second, we determine the seeds of semigroups with at most three left elements. And third, we find the great-grandchildren of any numerical semigroup in terms of its seeds. For genera around one half of the maximum size of native integers, the newly defined seeds algorithm performs significantly better than the RGD algorithm. For future compilators allowing larger native sized integers this may constitute a powerful tool to explore the semigroup tree up to genera never explored before. The new seeds algorithm uses bitwise integer operations apart from techniques such as parallelization and depth first search as wisely introduced in this context by Fromentin and Hivert.

Using 128-bit integers, as is the current bit length limitation in C, one can explore semigroups up to genus 64, search for Eliahou semigroups and check the Wilf conjecture for genus up to 66 and count semigroups of genus up to 69. In an eventual $2b$-bit integer limitation scenario, our algorithm will be able to efficiently explore semigroups up to genus b, search for Eliahou semigroups and check the Wilf conjecture for genus up to $b + 2$ and count semigroups of genus up to $b + 5$. In particular, the algorithm has been used to prove that there are no Eliahou semigroups of genus 66, hence proving the Wilf conjecture for genus up to 66.

In Section 2 we revisit the definition of the gap bitstream and the seed bitstream of a numerical semigroup, this time using an alternative approach to the definition of seed with respect to the one in [5]. In Section 3 we determine these bitstreams for semigroups of small rank. In Section 4 we

revisit how the seed bistream of a semigroup can be computed from the seed bitstrem of its parent from the new perspective. In Section 5 we characterize new generators, new strong generators and newly strong generators from an analysis of the seed bitsream. In Section 6 we explain how the bitstreams can be used to determine the great-grandchildren of a numerical semigroup. All the previous results can be used to build a revisited version of the seeds algorithm and in Section 7 we present empiric performance results. In Section 8 we recall the Wilf conjecture, and Eliahou and Delgado's related results and use our algorithm to check the Wilf conjecture for genus 66 and to find three new Eliahou semigroups of genus 67, one is of Eliahou-Fromentin type, another one is of Delgado's type and the third one is neither of Eliahou-Fromentin type, nor of Delgado's type. However, we will show that it is a member of a new family suggested by Shalom Eliahou. In Section 9 we analyze the limits of our algorithm in terms of the integer bit size bound of the compilator in which the algorithm is implemented. In particular it is analyzed the limit of our experimental results in a C compilator with a bound of 128 bits per integer.

2 The bitstream of gaps and the bitstream of seeds of a numerical semigroup

If a numerical semigroup Λ is $\{\lambda_0 = 0 < \lambda_1 < \dots\}$, define its *multiplicity* as $m(\Lambda) = \lambda_1$, its Frobenius number, $F(\Lambda)$, as its largest gap and its *conductor*, $c(\Lambda)$, as its largest gap plus one. Let $L(\Lambda)$ be the set of left elements of Λ. A numerical semigroup is called *ordinary* if $c(\Lambda) = m(\Lambda) = \lambda_1$ and *pseudo-ordinary* if $c(\Lambda) = \lambda_2$. We say that the *rank* of Λ is

$$k(\Lambda) := c(\Lambda) - g(\Lambda).$$

This way, $\lambda_{k(\Lambda)} = c(\Lambda)$ and so the rank of Λ coincides with the cardinal of $L(\Lambda)$. Now, the unique semigroup of rank 0 is $\mathbb{N}$, the semigroups of rank 1 are the non-trivial ordinary semigroups, and the semigroups of rank 2 are pseudo-ordinary semigroups. Define the *jump function* as

$$\begin{array}{rccl} u: & \mathbb{N} & \longrightarrow & \mathbb{N} \\ & j & \mapsto & u_j := \lambda_{j+1} - \lambda_j \end{array}$$

Observe that $u_j = 1$ for all $j \geqslant k(\Lambda)$, and that $\sum_{j=0}^{k(\Lambda)-1} u_j = c(\Lambda)$. Furthermore, the multiplicity coincides with u_0 and $u_j \leqslant u_0$ for all $j \in \mathbb{N}$.

A bitstream is a finite sequence $a = a_0 \dots a_\ell$ where a_i is either 0 or 1 for every i. For our purposes, we can indistinctly use a for $a_0 \dots a_\ell$ and for any bitstream of the form $a_0 \dots a_\ell \underbrace{0 \dots 0}_{t}$ for any positive integer t. The *weight* of a bitstream is the number of its nonzero elements. The weight of a bitstream a will be denoted $w(a)$. The weight of the substream $a_i \dots a_j$ of a will be denoted $w_i^j(a)$. If $n \in \mathbb{N}$ we denote $(n)_b$ its binary representation. That is, $(n)_b = a_0 \dots a_\ell$ where $a_0, \dots, a_\ell$ are the unique values in $\{0, 1\}$ such that $n = a_0 + 2a_1 + 2^2 a_2 + \dots + 2^\ell a_\ell$. For instance,

$$(2^i)_b = \underbrace{0 \dots 0}_{i} 1, \qquad (2^i - 1)_b = \underbrace{1 \dots 1}_{i}.$$

On the contray, for a bitstream $a = a_0, \dots, a_\ell$ we denote $\mathrm{int}(a)$ the integer $a_0 + 2a_1 + 2^2 a_2 + \dots + 2^\ell a_\ell$.

Suppose a semigroup Λ has conductor c. From now on, whenever the context is clear, we will omit Λ in $c(\Lambda), k(\Lambda) \dots$. We encode its gaps as the bitstream

$$G(\Lambda) = g_0 \dots g_{c-1}$$

with $g_i = 0$ if $i+1 \in \Lambda$ and $g_i = 1$ otherwise. Notice that

$$G(\Lambda) = \frac{1}{2}\left(2^{c(\Lambda)} - 1 - \sum_{j=0}^{k(\Lambda)-1} 2^{\lambda_j}\right)_b .$$

Recall that for $p < k(\Lambda)$, an element λ_s larger than or equal to c is an order-p *seed* of Λ if $\lambda_s + \lambda_p \neq \lambda_i + \lambda_j$ for all $p < i \leqslant j < s$. One can check that any order-p seed is at most $c + u_p - 1$. In particular, the number of order-p seeds is at most u_p ([5, Lemma 1.2.]).

In [5], the *table of seeds* of a semigroup Λ was defined as the binary table whose rows are indexed in $0 \leqslant i \leqslant k(\Lambda) - 1$, the ith row has u_i entries, and the jth entry in the ith row (with $0 \leqslant j < u_i$) is 1 if $c + j$ is an order-i seed of Λ, and 0 otherwise.

Example 1. *The table of seeds of the semigroup* $\Lambda = \{0, 3, 6, 8, 9, 10, \dots\}$*, with conductor* $c = 8$*, is*

1	0	1
1	0	1
1	1	

We can also encode the seeds of Λ as the bitstream

$$S(\Lambda) = S_0 \dots S_{c-1}$$

with $S_i = 1$ if $c + i - \lambda_j$ is a j-th order seed of Λ where j is the unique non-negative integer such that $\lambda_j \leqslant i < \lambda_{j+1}$. In fact, the seed bitstream is nothing else but the concatenation of the rows of the table of seeds.

Example 2. *The same semigroup* $\Lambda = \{0, 3, 6, 8, 9, 10, \dots\}$ *has* $G(\Lambda) = 11011010$ *and* $S(\Lambda) = 10110111$*.*

In Figure 1 one can see the sequences G and S for the lowest genus semigroups organized in the semigroup tree. To make it easier to read, we represented each 1 in the sequences with a dark circle with its position written inside, and each 0 in the sequence with a light gray circle with its position also written inside.

Lemma 3. *Given an integer* λ *with* $c \leqslant \lambda < 2c$*, let* $j \in \{0, \dots, k-1\}$ *be such that* $\lambda_j \leqslant \lambda - c < \lambda_{j+1}$*. Then,*

(a) $\lambda - \lambda_j$ *is an order-*j *seed of* Λ *if and only if* λ *is not the sum of two elements in* $L(\Lambda)$*.*

(b) $\lambda - \lambda_j$ *is an order-*j *seed of* Λ *if and only if* λ *is not the sum of two elements in* $\{\lambda_{j+1}, \dots, \lambda_{k-1}\}$*.*

Proof. Note that $\lambda = \lambda_s$ with $s = \lambda - c + k(\Lambda)$. The non-gap $\lambda - \lambda_j$ is *not* an order-j seed of Λ if and only if $\lambda = (\lambda - \lambda_j) + \lambda_j$ is the sum of two elements in $\{\lambda_{j+1}, \dots, \lambda_{s-1}\}$. As λ can not be the

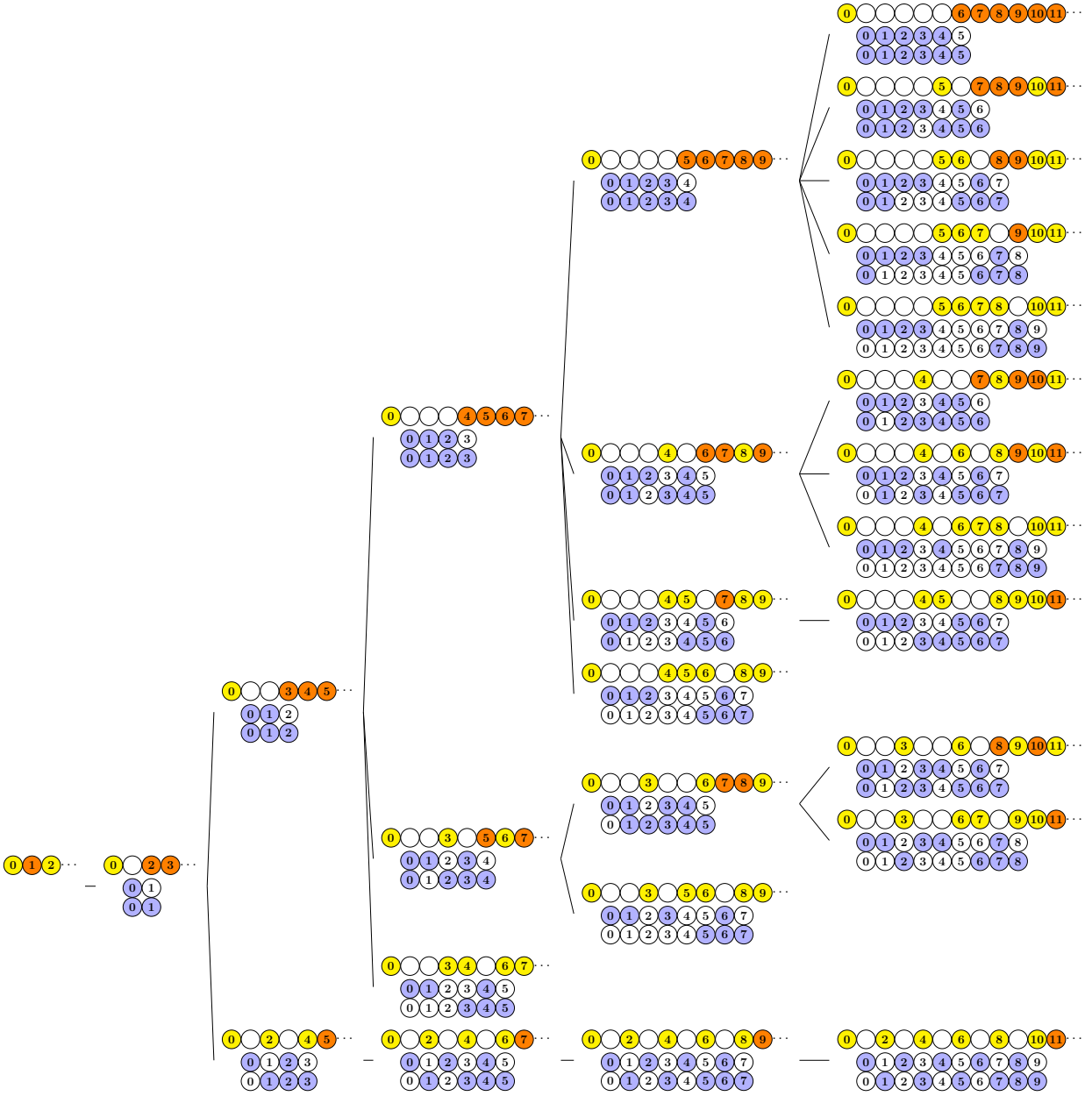

Figure 1: Lowest depth nodes of the semigroup tree. Each node is represented by its gap and its seed sequences.

sum $\lambda_u + \lambda_v$ with $\lambda_u > 0$ and $v \geqslant s$, we obtain that $\lambda - \lambda_j$ is *not* an order-j seed of Λ if and only if λ is the sum of two elements in $\{\lambda_{j+1}, \lambda_{j+2}, \dots\}$. In this case

$$\lambda = \lambda_{j+1+\ell_1} + \lambda_{j+1+\ell_2}$$

for some non-negative integers ℓ_1 and ℓ_2. Each of the two summands can now be bounded as follows.

- $\lambda_{j+1+\ell_1} < c$ since $\lambda_{j+1+\ell_1} = \lambda - \lambda_{j+1+\ell_2} \leqslant \lambda - \lambda_{j+1} < c$.
- $\lambda_{j+1+\ell_2} < c$ since $\lambda_{j+1+\ell_2} = \lambda - \lambda_{j+1+\ell_1} \leqslant \lambda - \lambda_{j+1} < c$.

□

Next corollary gives an alternative definition of the seed bitstream, and hence, an alternative definition of the notion of seed.

Corollary 4. *Let $\Sigma(\Lambda)$ be the bitstream $\Sigma_0\Sigma_1 \dots \Sigma_{2c-1}$, with $\Sigma_i = 0$ if i belongs to $L(\Lambda) + L(\Lambda)$ and $\Sigma_i = 1$ otherwise, then*

$$G_0G_1 \dots G_{c-2} = \Sigma_1 \dots \Sigma_{c-1}, \qquad G_{c-1} = 0$$

$$S_0S_1 \dots S_{c-1} = \Sigma_c\Sigma_{c+1} \dots \Sigma_{2c-1},$$

Hence,

$$G = ((\text{int}(\Sigma) \bmod 2^c)//2)_b\,,$$
$$S = (\text{int}(\Sigma)//2^c)_b\,,$$

where $a//b$ and $a \mod b$ denote the quotient and the remainder of the division of a by b, respectively.

Example 5. *Following the same example $\Lambda = \{0, 3, 6, 8, 9, 10, \dots\}$, we have $L(\Lambda) = \{0, 3, 6\}$ and $L(\Lambda) + L(\Lambda) = \{0, 3, 6, 9, 12\}$, that is $\Sigma = 0110110110110111$.*

$$\overbrace{0\ \underbrace{1\,1\,0\,1\,1\,0\,1}_{G(\Lambda)_{0,\dots,(c-2)}}\ \underbrace{1\,0\,1\,1\,0\,1\,1\,1}_{S(\Lambda)}}^{\Sigma(\Lambda)}$$

As a consequence of this alternative definition of $S(\Lambda)$ together with the fact that $\max(L(\Lambda) + L(\Lambda)) = 2\lambda_{k-1} = 2c - 2u_{k-1}$ we derive that $\Sigma_{2\lambda_{k-1}} = 0$ and $\Sigma_{2\lambda_{k-1}+1} = \Sigma_{2\lambda_{k-1}+2} = \dots = \Sigma_{2c-1} = 1$. That is,

$$S = S_0 \dots S_{2\lambda_{k-1}-1} \quad 0\ \underbrace{1 \cdots 1}_{2u_{k-1}-1}$$

Corollary 6. *The elements $c, c+1, \dots, c+u_{k-1}-1$ are order-$(k-1)$ seeds of Λ.*

3 The gap and the seed bitstreams of semigroups with small rank

Next lemma states the gap bitstream and the seed bitstream for semigroups of rank k at most 3. We will call $u := u_1$ and $v := u_2$.

Lemma 7. *The gap and the seed bitsreams of semigroups of rank k between 1 and 3 are given by the following formulas.*

- *For semigroups of rank $k = 1$,*

$$\begin{aligned} G(\{0, m, \dots\}) &= (2^{m-1} - 1)_b \\ S(\{0, m, \dots\}) &= (2^m - 1)_b \end{aligned}$$

- *For semigroups of rank $k = 2$ (in which case $1 < u \leqslant m$),*

$$\begin{aligned} G(\{0, m, m+u, \dots\}) &= (2^{m+u-1} - 1 - 2^{m-1})_b \\ S(\{0, m, m+u, \dots\}) &= (2^{m+u} - 1 - 2^{m-u})_b \end{aligned}$$

- *For semigroups of rank $k = 3$,*
  - *If $1 \leqslant u < m$, then $2 \leqslant v \leqslant m - u$ and*

$$\begin{aligned} G(\{0, m, m+u, m+u+v, \dots\}) &= (2^{m+u+v-1} - 1 - 2^{m-1} - 2^{m+u-1})_b \\ S(\{0, m, m+u, m+u+v, \dots\}) &= (2^{m+u+v} - 1 - 2^{m-u-v} - 2^{m-v} - 2^{m+u-v})_b \end{aligned}$$

  - *If $u = m$, then $2 \leqslant v \leqslant m$ and*

$$\begin{aligned} G(\{0, m, m+u, m+u+v, \dots\}) &= (2^{m+u+v-1} - 1 - 2^{m-1} - 2^{m+u-1})_b \\ S(\{0, m, m+u, m+u+v, \dots\}) &= (2^{m+u+v} - 1 - 2^{m-v} - 2^{m+u-v})_b \end{aligned}$$

Proof.

- For the rank-1 case $L(\Lambda) + L(\Lambda) = \{0\}$ and so,

$$\Sigma = (2^{2c} - 2)_b = 0\underbrace{1 \dots 1}_{2c-1}.$$

  Then the result follows from Corollary 4.

- For the rank-2 case, $L(\Lambda) + L(\Lambda) = \{0, m, 2m\}$ and so,

$$\begin{aligned} \Sigma &= (2^{2c} - 2 - 2^m - 2^{2m})_b \\ &= (2^c(2^c - 2^{2m-c} - 1) + (2^c - 2^m - 2))_b. \end{aligned}$$

  Hence, by Corollary 4,

$$\begin{aligned} G &= (2^{c-1} - 2^{m-1} - 1)_b \\ &= (2^{m+u-1} - 2^{m-1} - 1)_b, \end{aligned}$$

  and

$$\begin{aligned} S &= (2^c - 2^{2m-c} - 1)_b \\ &= (2^{m+u} - 2^{m-u} - 1)_b. \end{aligned}$$

- For the rank-3 case suppose $\Lambda = \{0, m, m+u, m+u+v, \dots\}$ with $v > 1$. Then the conductor of Λ is $c = m + u + v$.

  If $u < m$ then $m + u < 2m$ and since $2m \in \Lambda$, we have $2m \geqslant c$ and so $v \leqslant m - u$. Then $L(\Lambda) + L(\Lambda) = \{0, m, m + u, 2m, 2m + u, 2m + 2u\}$ with $m + u < c \leqslant 2m$. So,

$$\begin{aligned} \Sigma &= (2^{2c} - 2 - 2^m - 2^{m+u} - 2^{2m} - 2^{2m+u} - 2^{2m+2u})_b \\ &= (2^c(2^c - 2^{2m+2u-c} - 2^{2m+u-c} - 2^{2m-c} - 1) + (2^c - 2^{m+u} - 2^m - 2))_b \end{aligned}$$

  and, by Corollary 4,

$$\begin{aligned} G &= (2^{c-1} - 2^{m+u-1} - 2^{m-1} - 1)_b \\ &= (2^{m+u+v-1} - 2^{m+u-1} - 2^{m-1} - 1)_b \end{aligned}$$

  and

$$\begin{aligned} S &= (2^c - 2^{2m+2u-c} - 2^{2m+u-c} - 2^{2m-c} - 1)_b \\ &= (2^{m+u+v} - 2^{m+u-v} - 2^{m-v} - 2^{m-u-v} - 1)_b. \end{aligned}$$

  Otherwise, $u = m$ and $L(\Lambda) + L(\Lambda) = \{0, m, 2m, 3m, 4m\}$ and so,

$$\begin{aligned} \Sigma &= (2^{2c} - 2 - 2^m - 2^{2m} - 2^{3m} - 2^{4m})_b \\ &= (2^c(2^c - 2^{4m-c} - 2^{3m-c} - 1) + (2^c - 2^{2m} - 2^m - 2))_b. \end{aligned}$$

  Hence, by Corollary 4,

$$\begin{aligned} G &= (2^{c-1} - 2^{2m-1} - 2^{m-1} - 1)_b \\ &= (2^{m+u+v-1} - 2^{m+u-1} - 2^{m-1} - 1)_b, \end{aligned}$$

  and

$$\begin{aligned} S &= (2^c - 2^{4m-c} - 2^{3m-c} - 1)_b \\ &= (2^{m+u+v} - 2^{m+u-v} - 2^{m-v} - 1)_b. \end{aligned}$$

□

4 Updating the bitstream of seeds from parents to children

Suppose that Λ is a semigroup of rank k and conductor c and let $\tilde{\Lambda} = \Lambda \setminus \{\lambda_s\}$ with $s \geqslant k$. We will denote $\tilde{\Sigma} = \Sigma(\tilde{\Lambda})$, $\tilde{G} = G(\tilde{\Lambda})$, $\tilde{S} = S(\tilde{\Lambda})$, $\tilde{c} = c(\tilde{\Lambda}) = \lambda_s + 1$, $\tilde{k}$ the rank of $\tilde{\Lambda}$, $\tilde{u}_j$ the jth jump of $\tilde{\Lambda}$ and $\tilde{\lambda}_j$ the jth element of $\tilde{\Lambda}$. Let us denote $L = L(\Lambda)$, $\tilde{L} = L(\tilde{\Lambda})$, and I_s the interval $[c, \lambda_s - 1]$, which may be empty. We have $\tilde{L} = L \cup I_s$. Hence,

$$\tilde{L} + \tilde{L} = (L + L) \cup (L + I_s) \cup (I_s + I_s)$$

with $L + I_s = \{\lambda_i + j : 1 \leqslant i < k \text{ and } c \leqslant j < \lambda_s\}$ and $I_s + I_s = \{\ell : 2c \leqslant \ell \leqslant 2\lambda_s - 2\}$. In particular,

$$\tilde{\Sigma}_\ell = \begin{cases} 0 & \text{if } \ell < 2c \text{ and } \ell \in L + I_s \\ \Sigma_\ell & \text{if } \ell < 2c \text{ and } \ell \notin L + I_s \\ 0 & \text{if } 2c \leqslant \ell \leqslant 2\lambda_s - 2 \\ 1 & \text{if } 2\lambda_s - 1 \leqslant \ell \leqslant 2\lambda_s + 1 \end{cases}$$

And so,

$$\begin{aligned} \tilde{S}_\ell &= \tilde{\Sigma}_{\ell+\lambda_s+1} \\ &= \begin{cases} 0 & \text{if } \ell + \lambda_s + 1 < 2c \text{ and } \ell + \lambda_s + 1 \in L + I_s \\ \Sigma_{\ell+\lambda_s+1} = S_{\ell+\lambda_s-c+1} & \text{if } \ell + \lambda_s + 1 < 2c \text{ and } \ell + \lambda_s + 1 \notin L + I_s \\ 0 & \text{if } 2c \leqslant \ell + \lambda_s + 1 \leqslant 2\lambda_s - 2 \\ 1 & \text{if } 2\lambda_s - 1 \leqslant \ell + \lambda_s + 1 \leqslant 2\lambda_s + 1 \end{cases} \end{aligned} \tag{1}$$

From this formula we can derive Algorithm 1 to obtain $\tilde{G}$ and $\tilde{S}$ from G, S, c, k, s. An intermediate bitstream variable $auxS$ is introduced which can be defined so that $\tilde{S}_\ell = auxS_{\ell+\lambda_s-c+1}$ except for the case $2\lambda_s - 1 \leqslant \ell + \lambda_s + 1 \leqslant 2\lambda_s + 1$. That is

$$auxS_\iota = \begin{cases} S_\iota & \text{if } \iota < c \text{ and } \iota + c \notin L + I_s \\ 0 & \text{otherwise.} \end{cases}$$

It can be easily computed since

$$auxS_\iota = \begin{cases} 0 & \text{if } \iota + c = \lambda_j + \tilde{i} \text{ for some } \lambda_j \in L(\Lambda) \text{ and some } \tilde{i} \text{ with } c \leqslant \tilde{i} \leqslant c + s - k - 1 \\ S_\iota & \text{otherwise,} \end{cases}$$

or, equivalently,

$$auxS_\iota = \begin{cases} 0 & \text{if } \iota = \lambda_j + i \text{ for some } \lambda_j \in \Lambda \text{ and some } \tilde{i} \text{ with } 0 \leqslant i \leqslant s - k - 1 \\ S_\iota & \text{otherwise.} \end{cases}$$

This is what is done in the for loop of the algorithm.

This algorithm to update G and S is the keystone of the seeds algorithm. In fact, it may seem more natural to compute $\tilde{\Sigma}$ from Σ, but the restriction on the length of integers in implementations makes it more useful to split Σ into G and S and $\tilde{\Sigma}$ into $\tilde{G}$ and $\tilde{S}$. We assume that integers are represented from the least significant bit to the most significant bit. For instance, $(1)_b = 100000\ldots$. If $a = a_0 \ldots a_\ell$ we call $a \gg t$ the bitstream $\underbrace{0 \ldots 0}_{t} a_0 \ldots a_\ell$ and $a \ll t$ the bitstream $a_t \ldots a_\ell$. Notice that $(n)_b \gg t = (2^t n)_b$ and $(n)_b \ll t$ is $(n//2^t)_b$, which is equivalent to discarding the first t positions. If $a = a_0 \ldots a_\ell$ and $a' = a'_0 \ldots a'_{\ell'}$, then the operations $a \wedge a'$ and $a \vee a'$ are defined as usual,

$$a \wedge a' = r_0 \ldots r_{\max\{\ell,\ell'\}}, \text{ where } r_i = \begin{cases} 1 & \text{if } a_i = 1 \text{ and } a'_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$a \vee a' = r_0 \ldots r_{\max\{\ell,\ell'\}}, \text{ where } r_i = \begin{cases} 1 & \text{if } a_i = 1 \text{ or } a'_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

Now we will analyze the seeds of $\tilde{\Lambda}$ depending on whether their order is an *old order*, i.e. at most $k-1$ or a *new order*, i.e., at least k. The result that we will obtain at the end of this analysis is Theorem 8. This theorem was already proved in [5] from the perspective of the originary definition of seed introduced there.

Algorithm 1 Obtain $\tilde{G}$ and $\tilde{S}$ from G, S, c, k, s.

Input: $G = G(\Lambda)$, $S = S(\Lambda)$, $c = c(\Lambda)$, $k = k(\Lambda)$, s

$A = G$
$auxS = S$
for i **from** 0 **to** $s - k - 1$ **do**
  $A = A \gg 1$
  $auxS = auxS \wedge A$
end for
$\tilde{G} = G \vee (1 \gg (c + s - 1))$
$\tilde{S} = (auxS \ll (s - k + 1)) \vee (7 \gg (c + s - 2))$

Old-order seeds

Suppose $p < k - 1$ and let $\rho < \tilde{u}_p = u_p$. We want to analyze whether $\lambda_s + 1 + \rho$ is an order-p seed of $\tilde{\Lambda}$, that is, we want to determine whether $\tilde{S}_\ell = 1$, for $\ell = \lambda_p + \rho$. Since $p < k - 1$, we have $\ell + \lambda_s + 1 \leqslant \lambda_p + u_p + \lambda_s = \lambda_{p+1} + \lambda_s \leqslant \lambda_{k-1} + \lambda_s \leqslant c - 2 + \lambda_s \leqslant 2\lambda_s - 2$. Hence, we are in one of the first three hypotheses of equation (1).

First case. Suppose $0 \leqslant \rho < u_p - (\lambda_s - c + 1)$. In this case $\ell + \lambda_s + 1 < \lambda_{p+1} + c$. On one hand this implies that $\ell + \lambda_s + 1 < 2c$ and so, we are in one of the first two hypothesis of equation (1). On the other hand the same inequality implies that

$$\lambda_p + (\lambda_s + 1) \leqslant \ell + \lambda_s + 1 < \lambda_{p+1} + c$$

and so $\ell + \lambda_s + 1 \notin L + I_s$. Hence, $\tilde{S}_\ell = 1$ if and only if $\Sigma_{\ell+\lambda_s+1} = 1$, which is equivalent to $S_{\ell+\lambda_s-c+1} = 1$. In this case we say we have an *old-order recycled seed* since this order-p seed of $\tilde{\Lambda}$ is originated by an order-p seed of Λ. Indeed, $\lambda_p \leqslant \ell + \lambda_s - c + 1 < \lambda_{p+1}$. Notice that if $\lambda_t = \lambda_s + 1 + \rho$ is an old-order recycled seed then $\lambda_t < \tilde{c} + u_p - 1$.

Second case. Suppose $u_p - (\lambda_s - c + 1) \leqslant \rho \leqslant u_p - 2$. In this case $\ell + \lambda_s + 1$ lies between $\lambda_{p+1} + c$ and $\lambda_{p+1} + \lambda_s - 1$. Hence, $\ell + \lambda_s + 1 \in L + I_s$ and so $\tilde{S}_\ell = 0$.

Third case. Suppose $\rho = u_p - 1$. In this case $\ell + \lambda_s + 1 = \lambda_{p+1} + \lambda_s$.

- If $u_{p+1} \leqslant \lambda_s - c$ then we need to distinguish whether $p = k - 2$ or $p < k - 2$.
  - If $p = k - 2$, then $\ell + \lambda_s + 1 = \lambda_{k-1} + \lambda_s \geqslant \lambda_{k-1} + u_{k-1} + c = 2c$ while $\ell + \lambda_s + 1 = \lambda_{k-1} + \lambda_s \leqslant 2\lambda_s - 2$, hence we are in the third hypothesis of equation (1) and, hence, $\tilde{S}_\ell = 0$.
  - If $p < k - 2$, then $\ell + \lambda_s + 1 = \lambda_{p+2} + (\lambda_s - u_{p+1})$ with $c \leqslant (\lambda_s - u_{p+1}) \leqslant \lambda_s - 1$, hence $\ell + \lambda_s + 1 \in L + I_s$ and we are in the first hypothesis of equation (1). Then $\tilde{S}_\ell = 0$.
- If $u_{p+1} > \lambda_s - c$ then $\lambda_{p+1} + \lambda_s < \lambda_{p+2} + c$, which implies that $\ell + \lambda_s + 1 < 2c$ and so we are in the first or second hypothesis of equation (1). If $p = k - 2$ then $\ell + \lambda_s + 1 = \lambda_{k-1} + \lambda_s \notin L + I_s$. If $p < k - 2$ then by the previous inequality $\ell + \lambda_s + 1 = \lambda_{p+1} + \lambda_s < \lambda_{p+2} + c$ which implies, again, $\ell + \lambda_s + 1 \notin L + I_s$. Hence, $\tilde{S}_\ell = 1$ if and only if $\Sigma_{\ell+\lambda_s+1} = 1$, which is equivalent to $S_{\ell+\lambda_s-c+1} = 1$. In this case we say we have an *old-order new seed* since this order-p seed of $\tilde{\Lambda}$ is originated by an order-$(p + 1)$ seed of Λ. Indeed, $\lambda_{p+1} \leqslant \ell + \lambda_s - c + 1 < \lambda_{p+2}$.

Suppose now $p = k-1$ and let $\rho < \tilde{u}_{k-1}$. Notice that $\tilde{u}_{k-1} = u_{k-1}$ if and only $\lambda_s \neq c$ and $\tilde{u}_{k-1} = u_{k-1} + 1$ if $\lambda_s = c$.

First case. Suppose $0 \leqslant \rho < u_{k-1} - (\lambda_s - c + 1)$. In this case $\ell + \lambda_s + 1 < \lambda_k + c$. On one hand this implies that $\ell + \lambda_s + 1 < 2c$ and on the other hand, since $\ell + \lambda_s + 1 \geqslant \lambda_{k-1} + \lambda_s + 1$, we have $\ell + \lambda_s + 1 \notin L + I_s$. Hence, $\tilde{S}_\ell = 1$ if and only if $\Sigma_{\ell+\lambda_s+1} = 1$, which is equivalent to $S_{\ell+\lambda_s-c+1} = 1$. In this case we have again an *old-order recycled seed*.

Second case. Suppose $u_{k-1} - (\lambda_s - c + 1) \leqslant \rho < \tilde{u}_{k-1}$. In this case $\ell + \lambda_s + 1$ is at least $\lambda_k + c = 2c$. Hence, $\tilde{S}_\ell = 1$ if and only if $\ell + \lambda_s + 1 \geqslant 2\lambda_s - 1$, that is, if and only if $\rho \geqslant \lambda_s - \lambda_{k-1} - 2$, and so,

$$\lambda_s - \lambda_{k-1} - 2 \leqslant \rho \leqslant \tilde{u}_{k-1} - 1.$$

This may occur only if $\tilde{u}_{k-1} - 1 \geqslant \lambda_s - \lambda_{k-1} - 2$, which is equivalent to $\lambda_s \leqslant \tilde{\lambda}_k + 1$ and which in turn is only possible if

- $\lambda_s = c + 1$ and $\tilde{\lambda}_k = \lambda_k = c$, in which case $\tilde{u}_{k-1} = u_{k-1}$ and $u_{k-1} - 1 \leqslant \rho \leqslant u_{k-1} - 1$.
- $\lambda_s = c$ and $\tilde{\lambda}_k = \lambda_k + 1$, in which case $\tilde{u}_{k-1} = u_{k-1} + 1$ and $u_{k-1} - 1 \leqslant \rho \leqslant u_{k-1}$.

So, there are only old-order new-seeds $\lambda_s + 1 + \rho$ of order $k-1$ if and only if either $\lambda_s = c+1$ and $\rho = u_{k-1} - 1$ or $\lambda_s = c$ and $\rho = u_{k-1} - 1$ or $\rho = u_{k-1}$.

New-order seeds

If $\lambda_s = c$ then all the possible seed orders of $\tilde{\Lambda}$ are already seed orders of Λ and so, there are no new-order seeds.

Thus, we can assume that $\lambda_s \neq c$. Then $\tilde{u}_{s-1} = 2$ and $\tilde{u}_i = 1$ for any other new seed order i.

For $p = s-1$, since $\rho < \tilde{u}_p = \tilde{u}_{s-1} = 2$ we only need to consider the cases $\rho = 0$ and $\rho = 1$. But either for $\rho = 0$ and for $\rho = 1$ the non-gaps $\lambda_s + 1 + \rho$ correspond to order-$(s-1)$ seeds. Indeed the corresponding ℓ is $\ell = \lambda_{s-1} + \rho = \lambda_s + \rho - 1$ and hence, $\ell + \lambda_s + 1 = 2\lambda_s + \rho$, which in both cases lies between $2\lambda_s - 1$ and $2\lambda_s + 1$. There are no more order-$(s-1)$ seeds since $\tilde{u}_{s-1} = 2$.

For $p = s-2$, if p is indeed a new seed order, then $\tilde{u}_{s-2} = 1$. Then ρ can only be 0 Now, $\ell + \lambda_s + 1 = \lambda_{s-2} + \lambda_s + 1 = 2\lambda_s - 1$ and by equation (1), $\tilde{S}_\ell = 1$ and, so, $\lambda_s + 1$ is the unique order-$(s-2)$ seed.

For $k \leqslant p < s-2$, necessarily $\tilde{u}_p = 1$ and, so, the unique possible seed would be $\rho = 0$. In this case $\ell = \lambda_p$ and $\ell + \lambda_s + 1 = \lambda_p + \lambda_s + 1 < 2\lambda_s - 1$. Hence, by equation (1), $\tilde{S}_\ell = 0$ and, so, $\lambda_s + 1 + \rho$ is not an order-$(s-2)$ seed.

The previous results can be summarized in the next theorem.

Theorem 8. *Suppose that Λ is a semigroup of rank k and conductor c and let $\tilde{\Lambda} = \Lambda \setminus \{\lambda_s\}$ with $s \geqslant k$. The seeds of $\tilde{\Lambda}$ of order p are exactly the ones in the next list.*

1. *If $p < k$ (**Old-order seeds**):*
   - **Old-order recycled seeds:**
     - *Any order-p seed λ_t of Λ with $t > s$ (necessarily with $\lambda_t - c(\Lambda \setminus \{\lambda_s\}) < u_p - 1$)*
   - **Old-order new seeds:**
     - $\lambda_t = \lambda_s + u_p$ *if $p < k-1$ and λ_s is an order-$(p+1)$ seed of Λ*

- $\lambda_t = \lambda_s + u_p$ *and* $\lambda_t = \lambda_s + u_p + 1$ *if* $p = k-1$ *and* $\lambda_s = c$
- $\lambda_t = \lambda_s + u_p$ *if* $p = k-1$ *and* $\lambda_s = c+1$

2. *If* $p \geqslant k$ *(***New-order seeds***)*

   - $\lambda_s + 1$ *if* $p = s-2$
   - $\lambda_s + 1$ *and* $\lambda_s + 2$ *if* $p = s-1$

5 New generators, new strong generators and newly strong generators

Let Λ be a numerical semigroup with $c(\Lambda) \geqslant \lambda_1$. A *new generator* of Λ is a generator of Λ that is not a generator of $\Lambda \cup F(\Lambda)$. A *new strong generator* of Λ is a strong generator of Λ that is not a generator of $\Lambda \cup F(\Lambda)$. A *newly strong generator* of Λ is a generator of Λ and also a generator of $\Lambda \cup F(\Lambda)$, that is weak in $\Lambda \cup F(\Lambda)$ and strong in Λ.

Recall that order-i seeds are elements between $c(\Lambda)$ and $c(\Lambda)+u_i(\Lambda)-1$ and they are represented by 1's in the corresponding positions of $S(\Lambda)$ between positions λ_i and $\lambda_i + u_i(\Lambda) - 1 = \lambda_{i+1} - 1$. Recall also that right generators of Λ are exactly the order-0 seeds of Λ and that a right generator is strong if and only if it is an order-1 seed. In particular, right generators lie between $c(\Lambda)$ and $c(\Lambda) + m(\Lambda) - 1$ while strong generators lie between $c(\Lambda)$ and $c(\Lambda) + u(\Lambda) - 1$, where $u(\Lambda) = u_1(\Lambda)$.

For ordinary semigroups (i.e. semigroups of rank $k = 1$), the notion of strong generator is not defined. But we can define the *ordinarily strong generators* of an ordinary semigroup as those generators than when are taken away, the obtained semigroup has new generators. One can check that the ordinary semigroup Λ_m of multiplicity m has exactly two ordinarily strong generators: m and $m+1$. The semigroup $\Lambda_m \setminus \{m\}$ has two new generators, $2m$ and $2m+1$; the semigroup $\Lambda_m \setminus \{m+1\}$ has one new generator, $2m+1$.

Let Λ be a numerical semigroup with $S = S(\Lambda)$, $c = c(\Lambda)$, $m = m(\Lambda)$, $u = u(\Lambda) = u_(\Lambda)$, and $v = v(\Lambda) = u_2(\Lambda)$.

(1) Existence of right generators. The element $\mu \geqslant c$ is a right generator of Λ if and only if $S_{\mu-c} = 1$. The number of right generators of Λ is $w_0^{m-1}(S)$. By Lemma 7, in the particular case $k(\Lambda) = 1$, we have $w_0^{m-1}(S) = m$ and in the particular case $k(\Lambda) = 2$, we have $w_0^{m-1}(S) = m-1$

(2) Existence of strong generators. If $k(\Lambda) = 1$ there are no strong generators by definition but there are two ordinarily strong generators. If $k(\Lambda) > 1$, the element $\mu \geqslant c$ is a strong generator of Λ if and only if $S_{\mu-c} = S_{\mu+m-c} = 1$. The number of strong generators of Λ is $w_0^{u-1}(S \wedge (S \ll m))$. In the particular case $k(\Lambda) = 2$, by Lemma 7, $w_0^{u-1}(S \wedge (S \ll m)) = \begin{cases} u-1 & \text{if } m-u < u \\ u & \text{otherwise.} \end{cases}$

(3) Existence of new generators. Suppose μ_1 is a right generator of Λ.

- Suppose $k(\Lambda) = 1$ then Λ is ordinary and, as discussed above, $\Lambda \setminus \{\mu_1\}$ has exactly two new generators if $\mu_1 = m$, one new generator if $\mu_1 = m+1$ and zero new generators otherwise.

- If $k(\Lambda) > 1$, by Theorem 8 (old-order new seeds, $p < k-1$) there is at most one new generator of $\Lambda \setminus \{\mu_1\}$, which must be $\mu_1 + m$. Indeed, $\mu_1 + m$ is a new generator if and only μ_1 is a strong generator of Λ and so, if and only if $S_{\mu_1 - c} = S_{\mu_1 + m - c} = 1$. This situation occurs $w_0^{u-1}(S \wedge (S \ll m))$ times.

(4) Existence of new strong generators. Suppose μ_1 is a right generator of Λ.

- Suppose $k(\Lambda) = 1$.
  - If $\mu_1 = c$ then $k(\Lambda \setminus \{\mu_1\}) = 1$ and $\Lambda \setminus \{\mu_1\}$ has no strong generators since in this case there are no order-1 seeds.
  - If $\mu_1 = c+1$, we just saw that there is one new generator of $\Lambda \setminus \{\mu_1\}$, which is $\mu_1 + m$. By Theorem 8 (new-order new seeds, $p = s - 1$) the new generator is strong if and only if $\mu_1 + m = \mu_1 + 1$ or $\mu_1 + m = \mu_1 + 2$. In the first case we have a contradiction with $k(\Lambda) = 1$. So, there exists one new strong generator if $m = 2$ for $\mu_1 = 3$, and zero new strong generators otherwise.
  - If $\mu_1 > c + 1$, we already saw that there are no new generators of $\Lambda \setminus \{\mu_1\}$.
- If $k(\Lambda) = 2$ we already saw that there is one new generator of $\Lambda \setminus \{\mu_1\}$, which is $\mu_2 = \mu_1 + m$, only if μ_1 is a strong generator of Λ. By Lemma 7 for the case of rank $k = 2$, this is equivalent to $\mu_1 \leqslant c + u - 1$. Now, μ_2 may be strong in $\Lambda \setminus \{\mu_1\}$ if and only if it is a seed of order 1 of $\Lambda \setminus \{\mu_1\}$. We know that order-1 seeds of $\Lambda \setminus \{\mu_1\}$ are at most $\mu_1 + u_1(\Lambda \setminus \{\mu_1\})$. In particular, we need $\mu_2 = \mu_1 + m \leqslant \mu_1 + u_1(\Lambda \setminus \{\mu_1\})$ and so $u_1(\Lambda \setminus \{\mu_1\}) \geqslant m$ which implies $u_1(\Lambda \setminus \{\mu_1\}) = m$. This in turn implies that either $u = m - 1$ and $\mu_1 = c$ or $u = m$ and $\mu_1 > c$. Now, since $\mu_2 = \mu_1 + m$, it can not be an (old-)order-1 recycled seed of $\Lambda \setminus \{\mu_1\}$. Then it needs to be an (old-)order-1 new seed of $\Lambda \setminus \{\mu_1\}$. Then, by Theorem 8 (for old-order new seeds with $p = k - 1$), μ_2 is a strong generator of $\Lambda \setminus \{\mu_1\}$ if and only if either
  - $u = m - 1$ and $\mu_1 = c$
  - $u = m$ and $\mu_1 = c + 1$.

  So, there exists one new strong generator if $u = m - 1$ for $\mu_1 = c$ and there exists one new strong generator if $u = m$ for $\mu_1 = c + 1$. There exist zero new strong generators otherwise.
- Suppose $k(\Lambda) \geqslant 3$. A new generator μ_2 of $\Lambda \setminus \{\mu_1\}$ may only be strong if $u = m$. Indeed, first of all, $u_0(\Lambda) = u_0(\Lambda \setminus \{\mu_1\})$ and $u_1(\Lambda) = u_1(\Lambda \setminus \{\mu_1\})$. Second, in order for μ_2 to be a new generator, $\mu_2 = \mu_1 + m$. Third, for μ_2 to be strong in $\Lambda \setminus \{\mu_1\}$ it must be an order-1 seed of $\Lambda \setminus \{\mu_1\}$. Since the conductor of $\Lambda \setminus \{\mu_1\}$ is $\mu_1 + 1$, an order-1 seed of $\Lambda \setminus \{\mu_1\}$ is at most $\mu_1 + u$. Hence $\mu_1 + m \leqslant \mu_1 + u$ and this is only possible if $u = m$. Now, since $\mu_2 = \mu_1 + m$, it can not be an (old-)order-1 recycled seed of $\Lambda \setminus \{\mu_1\}$. Then it needs to be an (old-)order-1 new seed of $\Lambda \setminus \{\mu_1\}$. In this case, by Theorem 8 (old-order new seeds with $p < k - 1$), μ_1 needs to be an order-2 seed of Λ and so, $S_{\mu_1 - c + \lambda_2} = S_{\mu_1 - c + 2m} = 1$. Hence, there exists one new strong generator of $\Lambda \setminus \{\mu_1\}$ if $u = m$ and $S_{\mu_1 - c} = S_{\mu_1 - c + m} = S_{\mu_1 - c + 2m} = 1$. This situations occurs $w_0^{v-1}(S \wedge (S \ll u) \wedge (S \ll (m + u)))$ times whenever $u = m$. Otherwise there exist no new strong generators.

(5) Existence of newly strong generators. If μ_2 is a newly strong generator of $\Lambda\setminus\{\mu_1\}$, then, on one hand $c \leqslant \mu_1 < \mu_2 < c+m$ and so $\mu_2 - \mu_1 < m$ and, on the other hand μ_2 is an (old-)order-1 new seed of $\Lambda \setminus \{\mu_1\}$.

- If $k(\Lambda) = 1$, by Lemma 7 μ_1, μ_2 with $\mu_1 < \mu_2$ are right generators if and only if $m \leqslant \mu_1 < \mu_2 < 2m$. By Theorem 8 (new-order new seeds with $p = k$), μ_2 is a newly strong generator of $\Lambda \setminus \{\mu_1\}$ if and only if one of the following options holds.
  - $\mu_1 = \lambda_3 = m+2$ and $\mu_2 = \mu_1 + 1 = m+3$ and $m \geqslant 4$
  - $\mu_1 = \lambda_2 = m+1$ and $\mu_2 = \mu_1 + 1 = m+2$ and $m \geqslant 3$
  - $\mu_1 = \lambda_2 = m+1$ and $\mu_2 = \mu_1 + 2 = m+3$ and $m \geqslant 4$

  So, if $m \leqslant 2$ there exist zero newly strong generators for any μ_1; if $m = 3$ there exists one newly strong generator for $\mu_1 = m+1$ and zero newly strong generators otherwise; if $m \geqslant 4$ there exist two newly strong generators for $\mu_1 = m+1$, one newly strong generator for $\mu_1 = m+2$, and zero newly strong generators otherwise.

  We add that there exists one newly ordinarily strong generator for $\mu_1 = m$, which is $m+2$.
- If $k(\Lambda) = 2$, by Theorem 8 (old-order new seeds with $p = k-1$), one of the following options must hold.
  - $\mu_1 = c = m+u$ and $\mu_2 = \mu_1 + u = m+2u$ with μ_1, μ_2 right generators of Λ. That is, by Lemma 7, $\mu_1 - c \neq m-u$ and $\mu_2 - c \neq m-u$, i.e., $u \neq m$ and $m \neq 2u$. Conversely, one can check that if $u < m$ and $m \neq 2u$ then $m+2u$ is a newly strong generator of $\Lambda \setminus \{m+u\} = \{0, m, m+u+1, m+u+2, \dots\}$.
  - $\mu_1 = c = m+u$ and $\mu_2 = \mu_1 + u + 1 = m+2u+1$ with μ_1, μ_2 right generators of Λ. That is, by Lemma 7, $\mu_1 - c \neq m-u$ and $\mu_2 - c \neq m-u$, i.e., $u \neq m$ and $m \neq 2u+1$, which together with the condition $\mu_2 < c+m$ results in the conditions $u < m-1$ and $m \neq 2u+1$. Conversely, one can check that if $u < m-1$ and $m \neq 2u+1$ then $m+2u+1$ is a newly strong generator of $\Lambda\setminus\{m+u\} = \{0, m, m+u+1, m+u+2, \dots\}$.
  - $\mu_1 = c+1 = m+u+1$ and $\mu_2 = \mu_1 + u = m+2u+1$ with μ_1, μ_2 right generators of Λ. That is, by Lemma 7, $\mu_1 - c \neq m-u$ and $\mu_2 - c \neq m-u$, i.e., $u \neq m-1$ and $m \neq 2u+1$, which together with the condition $\mu_2 - \mu_1 < m$ results in the conditions $u < m-1$ and $m \neq 2u+1$. Conversely, one can check that if $u < m-1$ and $m \neq 2u+1$ then $m+2u+1$ is a newly strong generator of $\Lambda\setminus\{m+u+1\} = \{0, m, m+u, m+u+2, m+u+3, \dots\}$.

  So, if $u < m$ and $m \neq 2u$ there exists one newly strong generator for $\mu_1 = m+u$, while if $u < m-1$ and $m \neq 2u+1$ there exists one newly strong generator for $\mu_1 = m+u$ (different than the one just mentioned) and a newly strong generator for $\mu_1 = m+u+1$. There exist zero newly strong generators otherwise.
- If $k(\Lambda) \geqslant 3$, by Theorem 8 (old-order new seeds with $p < k-1$), $\mu_2 = \mu_1 + u$ and μ_1 needs to be an order-2 seed of Λ and so, $S_{\mu_1 - c + \lambda_2} = S_{\mu_1 - c + m + u} = 1$. Furthermore, since $\mu_2 - \mu_1 < m$ and $\mu_2 = \mu_1 + u$, it follows that $u < m$. Hence, there exists one newly strong generator of $\Lambda \setminus \{\mu_1\}$ if $u < m$ and $S_{\mu_1 - c} = S_{\mu_1 - c + u} = S_{\mu_1 - c + u + m} = 1$. This situation occurs $w_0^{v-1}(S \wedge (S \ll u) \wedge (S \ll (m+u)))$ times whenever $u < m$. Otherwise there exist no newly strong generators.

6 The great-grandchildren of a numerical semigroup

We explain now how to obtain the set of all great-grandchildren of a semigroup in the semigroup tree just from its seeds, and the parameters m, u, v.

Each particular case of the lemma is illustrated with an example at the appendix. In those examples we represent the tree of descendants of semigroups with different characteristics. Each semigroup in the tree is represented by its table of seeds.

Theorem 9. *Let $S = S(\Lambda)$, $m = m(\Lambda)$, $u = u(\Lambda)$, $v = v(\Lambda)$.*

The number n_c of children, the number n_{gc} of grandchildren, and the number n_{ggc} of great-grandchildren of Λ is, respectively,

$$n_c(\Lambda) = \begin{cases} 1 & \text{if } k(\Lambda)=0, \quad \textit{(see Fig. 2)} \\ m & \text{if } k(\Lambda)=1, \quad \textit{(see Fig. 2)} \\ m-1 & \text{if } k(\Lambda)=2, \quad \textit{(see Fig. 2)} \\ w_0^{m-1}(S) & \text{if } k(\Lambda)\geqslant 3, \quad \textit{(see Fig. 3-Fig. 7)} \end{cases}$$

$$n_{gc}(\Lambda) = \begin{cases} 2 & \text{if } k(\Lambda)=0, \quad \textit{(see Fig. 2)} \\ \binom{m}{2}+3 & \text{if } k(\Lambda)=1, \quad \textit{(see Fig. 3-Fig. 5)} \\ \binom{n_c(\Lambda)}{2}+w_0^{u-1}(S\wedge(S\ll m)) & \text{if } k(\Lambda)>1, \quad \textit{(see Fig. 6-Fig. 7)} \end{cases}$$

$$n_{ggc}(\Lambda) = \begin{cases} 4 & \text{if } k(\Lambda)=0, \quad \textit{(see Fig. 2)} \\ \binom{m}{3}+3m+1 & \text{if } k(\Lambda)=1, m=2,3, \quad \textit{(see Fig. 3 and Fig 4)} \\ \binom{m}{3}+3m+3 & \text{if } k(\Lambda)=1, m\geqslant 4, \quad \textit{(see Fig. 5)} \\ \binom{m-1}{3}+(u-\delta_a)(m-2)+\delta_b+2\delta_c+\delta_d & \text{if } k(\Lambda)=2, \quad \textit{(see Fig. 6)} \\ \binom{n_c(\Lambda)}{3}+w_0^{u-1}(S\wedge(S\ll m))(n_c(\Lambda)-1) & \\ \qquad +w_0^{v-1}(S\wedge(S\ll u)\wedge(S\ll(u+m))) & \text{if } k(\Lambda)\geqslant 3, \quad \textit{(see Fig. 7)} \end{cases}$$

where

$$\delta_a = \begin{cases} 1 & \textit{if } m<2u \\ 0 & \textit{otherwise} \end{cases}$$

$$\delta_b = \begin{cases} 1 & \textit{if } u=m \textit{ or } 2u\neq m \\ 0 & \textit{otherwise} \end{cases}$$

$$\delta_c = \begin{cases} 1 & \textit{if } u<m-1 \textit{ and } 2u+1\neq m \\ 0 & \textit{otherwise} \end{cases}$$

$$\delta_d = \begin{cases} 1 & \textit{if } u=m-1 \\ 0 & \textit{otherwise} \end{cases}$$

Proof. The unique child of $\mathbb{N}$ is $\mathbb{N}\setminus\{1\}$, while for a semigroup $\Lambda\neq\mathbb{N}$ the number of children is the number of its right generators. Then the formula for n_c follows from **(1)**.

Let us prove now the formula for $n_{gc}(\Lambda)$. The number of grandchildren of $\mathbb{N}$ is 2 since this is the number of semigroups of genus 2.

If $\Lambda\neq\mathbb{N}$, then a grandchild of Λ is a semigroup of the form $\Lambda\setminus\{\mu_1,\mu_2\}$ with $c\leqslant\mu_1<\mu_2$ and either

(a) μ_1, μ_2 are right generators of Λ,

(b) μ_1 is a right generators of Λ, and μ_2 is a new generator of $\Lambda \setminus \{\mu_1\}$.

Case (a) can be obtained in $\binom{n_c(\Lambda)}{2}$ different ways. Case (b), for $k(\Lambda) = 1$ can be obtained in three different ways, taking μ_1 as one of the two ordinarily strong generators and μ_2 as one of the new generators corresponding to μ_1 (**(3)**). Case (b), for $k(\Lambda) \geqslant 2$ can be done as many times as the number of strong generators of Λ, which is $w_0^{u-1}(S \wedge (S \ll m))$ (**(2)**).

Let us prove now the formula for n_{ggc}. If $\Lambda = \mathbb{N}$ then we know that the number of its great-grandchildren is 4. It can be just observed in Figure 1.

If $\Lambda \neq \mathbb{N}$, then a great-grandchild of Λ is a semigroup of the form $\Lambda \setminus \{\mu_1, \mu_2, \mu_3\}$ with $c \leqslant \mu_1 < \mu_2 < \mu_3$ and either

(a) μ_1, μ_2, μ_3 are right generators of Λ,

(b) μ_1, μ_2 are right generators of Λ, and μ_3 is a new generator of $\Lambda \setminus \{\mu_1\}$ or a new generator of $\Lambda \setminus \{\mu_2\}$,

(c) μ_1 is a right generator of Λ and μ_2, μ_3 are new generators of $\Lambda \setminus \{\mu_1\}$,

(d) μ_1, μ_2 are right generators of Λ, μ_3 is not a right generator of Λ, nor a right generator of $\Lambda \setminus \{\mu_1\}$, nor a right generator of $\Lambda \setminus \{\mu_2\}$, but it is a new generator of $\Lambda \setminus \{\mu_1, \mu_2\}$,

(e) μ_1 is a right generator of Λ, μ_2 is a new generator of $\Lambda \setminus \{\mu_1\}$ and μ_3 is a new generator of $\Lambda \setminus \{\mu_1, \mu_2\}$.

Now we are going to analyze each case separately.

- The situation (a) is counted by $\binom{n_c(\Lambda)}{3}$. In particular, for $k(\Lambda) = 1$ this equals $\binom{m}{3}$ and for $k(\Lambda) = 2$ this equals $\binom{m-1}{3}$.

$k(\Lambda) = 1$	$k(\Lambda) = 2$	$k(\Lambda) \geqslant 3$
$\binom{m}{3}$	$\binom{m-1}{3}$	$\binom{n_c(\Lambda)}{3}$

- Suppose now situation (b). For $k(\Lambda) = 1$, considering the two ordinarily strong generatrors, this may occur in the following cases:
  - $\mu_1 = m$, $\mu_2 = m + 1$, $\mu_3 = 2m$
  - $\mu_1 = m$, $\mu_2 = m + 1$, $\mu_3 = 2m + 1$
  - $\mu_1 = m$, $\mu_2 \in \{m + 2, \ldots, 2m - 1\}$, $\mu_3 = 2m$
  - $\mu_1 = m$, $\mu_2 \in \{m + 2, \ldots, 2m - 1\}$, $\mu_3 = 2m + 1$
  - $\mu_1 = m + 1$, $\mu_2 \in \{m + 2, \ldots, 2m - 1\}$, $\mu_3 = 2m + 1$

  So, this situation occurs $2 + 3(m - 2) = 3m - 4$ times. For $k(\Lambda) \geqslant 2$, the situation in (b) corresponds to the number of strong generators multiplied by the number of right generators minus one. By **(1)** and **(2)**, this equals $w_0^{u-1}(S \wedge (S \ll m))(n_c(\Lambda) - 1)$, which for the particular case $k(\Lambda) = 2$ is $(u - \delta_a)(m - 2)$.

$k(\Lambda)=1$	$k(\Lambda)=2$	$k(\Lambda)\geqslant 3$
$3m-4$	$(u-\delta_a)(m-2)$	$w_0^{u-1}(S\wedge(S\ll m))(n_c(\Lambda)-1)$

- Situation (c) only occurs once for $k(\Lambda)=1$, for $\mu_1=m,\mu_2=2m,\mu_3=2m+1$.

$k(\Lambda)=1$	$k(\Lambda)=2$	$k(\Lambda)\geqslant 3$
1		

- Situation (d) for $k(\Lambda)>1$ or for $k(\Lambda)=1$ with $\mu_1>m$, is equivalent to μ_1,μ_2 being right generators of Λ, μ_2 being a newly strong generator of $\Lambda\setminus\{\mu_1\}$ and μ_3 being the unique new generator of $\Lambda\setminus\{\mu_1,\mu_2\}$. According to **(5)** this occurs as many times as indicated in the table

$k(\Lambda)=1$	$k(\Lambda)=2$	$k(\Lambda)\geqslant 3$
0 if $m\leqslant 2$ 1 if $m=3$ 3 if $m\geqslant 4$	$\delta_{b'}+2\delta_c$	$(1-\delta_{b''})w_0^{v-1}(S\wedge(S\ll u)\wedge(S\ll(u+m)))$

where

$$\delta_{b'} = \begin{cases} 1 & \text{if } u<m \text{ and } 2u\neq m \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{b''} = \begin{cases} 1 & \text{if } u=m \\ 0 & \text{otherwise} \end{cases}$$

Situation (d) for $k(\Lambda)=1$ and $\mu_1=m$ may only occur for μ_2 equal to one of the ordinarily strong generators of $\Lambda\setminus\{m\}$, which are $m+1$ and $m+2$. For $m=2$, the cases in which this occurs are excatly when either $(\mu_1,\mu_2,\mu_3)=(2,3,6)$ or $(\mu_1,\mu_2,\mu_3)=(2,3,7)$. For $m>2$, the cases in which this occurs are excatly when either $(\mu_1,\mu_2,\mu_3)=(m,m+1,2m+2)$, $(\mu_1,\mu_2,\mu_3)=(m,m+1,2m+3)$, or $(\mu_1,\mu_2,\mu_3)=(m,m+2,2m+2)$. So, we need to add to the previous table these cases.

$k(\Lambda)=1$	$k(\Lambda)=2$	$k(\Lambda)\geqslant 3$
2 if $m=2$ 3 if $m\geqslant 3$		

- Situation (e) is equivalent to μ_1 being a right generator of Λ, μ_2 being a new strong generator of $\Lambda\setminus\{\mu_1\}$ and μ_3 being a new generator of $\Lambda\setminus\{\mu_1,\mu_2\}$. For $m=2$ this occurs exactly for $(\mu_1,\mu_2,\mu_3)=(2,4,7)$ and for $(\mu_1,\mu_2,\mu_3)=(3,5,7)$. Otherwise, according to **(4)** this occurs as many times as indicated in the table.

$k(\Lambda)=1$	$k(\Lambda)=2$	$k(\Lambda)\geqslant 3$
2 if $m=2$ 0 if $m\geqslant 3$	$\delta_{b''}+\delta_d$	$\delta_{b''}\cdot w_0^{v-1}(S\wedge(S\ll u)\wedge(S\ll(u+m)))$

The result follows from the fact that $\delta_b = \delta_{b'} + \delta_{b''}$.

□

Example 10. *For instance, the semigroup* $\Lambda = \{0, 3, 6, 8, 9, 10, \dots\}$ *(with* $S(\Lambda) = 10110111$*) satisfies*

- $n_c(\Lambda) = 2$*, since* $w_0^{m-1}(S) = w_0^2(S) = w(101) = 2$.
- $n_{gc}(\Lambda) = 3$*, since* $\binom{n_c(\Lambda)}{2} + w_0^{u-1}(S \wedge (S \ll m)) = 1 + w_0^3(101 \wedge 101) = 1 + 2 = 3$.
- $n_{ggc}(\Lambda) = 3$*, since* $m = u = 3$*,* $v = 2$*,* $w_0^2(S \wedge (S \ll m)) = w(101 \wedge 101) = 2$*, and* $w_0^1(S \wedge (S \ll u) \wedge (S \ll (u+m)) = w(10 \wedge 10 \wedge 11) = 1$*, the formula gives* $\binom{2}{3} + 2(2-1) + 1 = 3$.

7 The seeds algorithm revisited

The previous results can be used to define a revisited version of the seeds algorithm whose implementation in C++ can be seen in the appendix. The bit count required by Lemma 9 is performed using Brian Kernighan's algorithm. Parallelizing the implementation gives very good computation times. A parallel implementation of the algorithm can be found in

`https://github.com/mbrasamoros/seeds-algorithm`.

The performance of the algorithm is next compared to that of the Fromentin-Hivert algorithm [14, 13] and to that of the RGD algorithm [6, 4]. Using 8 workers in all algorithms we obtain the following table, where computation time is measured in seconds.

genus	40	42	44	46	48	50	52	54	56	58
FH	1	3	10	28	75	207	537	1410	3977	10620
RGD	2	4	9	23	62	158	403	1064	2892	7462
Seeds	1	2	6	15	41	107	284	758	1962	5257

Using 12 workers in all algorithms we obtain the next table.

genus	40	42	44	46	48	50	52	54	56	58	60
FH	1	2	7	19	53	145	372	978	2760	7398	21880
RGD	1	3	6	18	45	121	291	799	2101	5292	13785
Seeds	1	2	4	11	27	73	195	503	1329	3556	9459

And using 24 workers we obtain this other table.

genus	40	42	44	46	48	50	52	54	56	58	60
FH	1	2	6	17	45	126	332	876	2494	6600	19090
RGD	1	2	5	13	38	88	244	625	1551	4458	10382
Seeds	1	1	4	9	25	67	179	462	1226	3307	8586

8 Computation of Eliahou semigroups

Fix a numerical semigroup Λ. Let $k = \text{rank}(\Lambda)$, let p be the number of primitive elements of Λ, and let r the number of right generators of Λ. The Wilf conjecture states that $c(\Lambda) \leqslant kp$ [16]. Let $q = \left\lceil \frac{c(\Lambda)}{m(\Lambda)} \right\rceil$ and $\rho = qm(\Lambda) - c(\Lambda)$. We say that Λ is an Eliahou semigroup if the constant

$E(\Lambda) = k(p-r) - q(m-r) + \rho$ is negative. Shalom Eliahou proved that if the Wilf conjecture does not hold for a semigroup, then it must be an Eliahou semigroup [11]. Hence, verifying that the Eliahou semigroups up to a certain genus satisfy the Wilf conjecture proves the Wilf conjecture for all semigroups up to this genus. Eliahou semigroups are very rare. Let $\langle a,b,c\rangle\,|_\kappa$ be the minimum semigroup containing a,b,c and all integers larger than or equal to κ. Jean Fromentin found that the unique Eliahou semigroups of genus $g \leqslant 60$ are exactly

$$\begin{aligned}
\varepsilon_1 &= \langle 14,22,23\rangle\,|_{56},\\
\varepsilon_2 &= \langle 16,25,26\rangle\,|_{64},\\
\varepsilon_3 &= \langle 17,26,28\rangle\,|_{68},\\
\varepsilon_4 &= \langle 17,27,28\rangle\,|_{68},\\
\varepsilon_5 &= \langle 18,28,29\rangle\,|_{72},
\end{aligned}$$

which have genus $43, 51, 55, 55$ and 59, respectively. In [7] it was computed that the unique Eliahou semigroups with genus between 61 and 65 are exactly

$$\begin{aligned}
\varepsilon_6 &= \langle 19,29,31\rangle\,|_{76},\\
\varepsilon_7 &= \langle 19,30,31\rangle\,|_{76},
\end{aligned}$$

both of genus 63.

Shalom Eliahou and Jean Fromentin found in [12] numerical semigroups with arbitrarily small Eliahou constant. In particular, they proved [12, Proposition 3.1.] that for positive integers m,a,b with $(3m+1)/2 \leqslant a < b \leqslant (5m-1)/3$ and for which the elements $a, b, 2a, a+b, 2b, 3a, 2a+b, a+2b, 3b$ are all different modulo m, it holds that $EF(m,a,b) := \langle m,a,b\rangle\,|_{4m}$ has Eliahou constant equal to -1. Manuel Delgado constructed in [8], for each integer number, infinite families of numerical semigroups having Eliahou constant equal to that number. In particular, he constructed infinite families of semigroups with negative Eliahou constant. The semigroups constructed by Delgado are

$$D^{(i,j)}(p,\tau) = \langle m^{(i,j)}, g^{(i,j)}, g^{(i,j)}+1\rangle\,|_{c^{(i,j)}},$$

for p an even positive integer and τ, i, j non-negative integers, where

$$\begin{aligned}
m^{(i,j)} &= \frac{p^2}{4} + p(\frac{\tau}{2}+2) + 2 + j\frac{p}{2}\\
g^{(i,j)} &= \frac{p^2}{2} + p(\tau+\frac{7}{2}) - \tau + j(p-1) + im^{(i,j)}\\
c^{(i,j)} &= \frac{p^3}{4} + p^2(\frac{\tau}{2}+2) + 2p - \tau + j\frac{p^2}{2} + i\left(\frac{p}{2}+1\right)m^{(i,j)}
\end{aligned}$$

The Eliahou semigroups in these families found by Eliahou and Formentin and by Delgado still

satisfy the Wilf conjecture [12, 8]. It holds that

$$\begin{aligned}
\varepsilon_1 &= EF(14,22,23) = D^{(0,0)}(4,0)\\
\varepsilon_2 &= EF(16,25,26) = D^{(0,1)}(4,0)\\
\varepsilon_3 &= EF(17,26,28)\\
\varepsilon_4 &= EF(17,27,28)\\
\varepsilon_5 &= EF(18,28,29) = D^{(0,2)}(4,0)\\
\varepsilon_6 &= EF(19,29,31)\\
\varepsilon_7 &= EF(19,30,31)
\end{aligned}$$

Consequently, the Wilf conjecture holds for all semigroups of genus up to 65.

Now suppose that Λ is not ordinary and let Λ' be a child of Λ obtained by taking away $c+s$ with $s \geqslant 0$. Let $p', r', k', q', m', \rho'$ be the parameters involved in the computation of the Eliahou constant for the semigroup Λ'. If we define

$$\begin{aligned}
\delta_w &= \begin{cases} 1 & \text{if } c+s \text{ is a weak generator of } \Lambda \\ 0 & \text{otherwise} \end{cases}\\
\delta_\rho &= \begin{cases} 1 & \text{if } \rho \leqslant s \\ 0 & \text{otherwise,} \end{cases}
\end{aligned}$$

then $m' = m$, $p' = p - \delta_w$, $k' = k + s$, $q' = q + \delta_\rho$, and $\rho' = \rho - s - 1 + \delta_\rho m$. Let the *preceding sibling* Λ'_0 of Λ' be defined as follows. If $c+s$ is the smallest right generator of Λ, then $\Lambda'_0 = \Lambda$. Otherwise, let s_0 be such that $c+s_0$ is the largest right generator of Λ smaller than $c+s$. Then $\Lambda'_0 = \Lambda \setminus \{c+s_0\}$. Let now r'_0 be the number of right generators of the semigroup Λ'_0. Then the number of right generators of Λ' is $r' = r'_0 - \delta_w$. This enables us to use our exploration of semigroups up to genus g for finding all Eliahou semigroups of genus up to $g+1$. In fact, our algorithm only checks the Eliahou inequality for semigroups with $k \geqslant 3$. But as proved in [11, Section 7.2] this covers all possible counterexamples.

With the new algorithm we found three further Eliahou semigroups of genus 67. They are

$$\begin{aligned}
\varepsilon_8 &= \langle 20,31,32\rangle\ |_{80}\\
\varepsilon_9 &= \langle 20,32,33\rangle\ |_{80}\\
\varepsilon_{10} &= \langle 19,26,27\rangle\ |_{90}
\end{aligned}$$

The three semigroups ε_8, ε_9, ε_{10}, have Eliahou constant equal to -1. The parameters involved in the computation of the Eliahou constant of the semigroups ε_8 and ε_9 coincide and they are

$$c = 80,\ m = 20,\ q = 4,\ \rho = 0,\ p = 13,\ r = 10,\ k = 13.$$

The parameters involved in the computation of the Eliahou constant of the semigroup ε_{10} are

$$c = 90,\ m = 19,\ q = 5,\ \rho = 5,\ p = 7,\ r = 4,\ k = 23.$$

The semigroups ε_8 and ε_9 are of the type described by Eliahou and Fromentin in [12]. Furthermore, the semigroup ε_8 is the Delgado semigroup $D^{(0,3)}(4,0)$. That is,

$$\begin{aligned} \varepsilon_8 &= EF(20,31,32) = D^{(0,3)}(4,0) \\ \varepsilon_9 &= EF(20,32,33) \end{aligned}$$

The semigroup ε_{10} is neither of Eliahou-Fromentin type nor of Delgado's type.

Shalom Eliahou noticed that, indeed, ε_{10} is a member of a new family of Eliahou semigroups, defined as

$$BEF_t = \langle 2t+1, 3t-1, 3t\rangle \mid_{10t},$$

for $t \geqslant 9$. All the semigroups in this family satisfy

$$c = 10t,\ m = 2t+1,\ q = 5,\ \rho = 5,$$

and, since

$$\begin{aligned} BEF_t = \{&0, 2t+1, 3t-1, 3t,\ 4t+2, 5t, 5t+1, 6t-2, 6t-1, 6t,\ 6t+3, \\ &7t+1, 7t+2,\ 8t-1, 8t, 8t+1,\ 8t+4,\ 9t-3, 9t-2, 9t-1, 9t,\ 9t+2, 9t+3,\ 10t, \dots\}, \end{aligned}$$

the rank is $k = 23$ whenever $t \geqslant 8$. In particular, their genus is $10t - 23$. The elements between the conductor c and $c + m - 1$ that are not generators are

$$10t, 10t{+}1, 10t{+}2,\ 10t{+}5,\ 11t{-}2, 11t{-}1, 11t, 11t{+}1,\ 11t{+}3, 11t{+}4,\ 12t{-}4, 12t{-}3, 12t{-}2, 12t{-}1, 12t.$$

For $t \geqslant 9$ this list contains exactly 15 elements, and so the number of right generators is $r = 2t + 1 - 15 = 2t - 14$. For $t = 8$, the list contains 14 elements and so $r = 2t - 13$. Hence, $E(BEF_8) = 4$ while $E(BEF_t) = -1$ for all $t \geqslant 9$. This shows that the family BEF_t for $t \geqslant 9$ is indeed a family of Eliahou semigroups and that the condition $t \geqslant 9$ is necessary. It is easy to check that the Wilf conjecture holds for these semigroups since $c = 10t$, $k = 23$, $p = 2t - 11$. One can check that

$$\varepsilon_{10} = BEF_9.$$

We can not ensure a priori that the three Eliahou semigroups $\varepsilon_8, \varepsilon_9, \varepsilon_{10}$ are *all* Eliahou semigroups of genus between 66 and 67 because we have the integer bit length limitation. We will analyze it in next section.

9 How far can we get using 128 bit integers?

Let n_g be the number of semigroups of genus g. Let E_g be the set of Eliahou counterexamples of genus smaller than or equal to g. As a consequence of the previous sections, if we obtain the parameters G and S of all semigroups of genus up to g, then we can also deduce n_{g+3} and E_{g+1}. The limitation of our algorithm comes from the fact that the variables G and S allocate values in positions 0 to $c-1$. If we use 128 bit integers, this means that we explore correctly the semigroups of conductor up to 128.

9.1 Exploring up to genus 64

Since $c \leqslant 2g$, if we use 128 bit integers, this guarantees that the algorithm will proceed correctly to explore the tree up to genus 64, thus guaranteeing the correct computation of n_{67} and E_{65}.

9.2 Exploring up to genus 65

If we use the algorithm to explore up to genus 65, we will missexplore semigroups of genus 65 and conductor 129 or 130, that is, we will be missexploring exactly the symmetric (i.e. rank equal to g) and pseudo-symmetric (i.e. rank equal to $g-1$) semigroups of genus 65. The error in these cases will be that we will not be able to write

- $S_{c-1} = 1$ when $c \geqslant 129$
- $S_{c-2} = 1$ and $G_{c-2} = 1$ when $c \geqslant 130$

But the parameters such as the genus, conductor, multiplicity, first jump and second jump, or the number of right generators will be correctly computed.

Computation of n_{68}

To analyze this case we need the next lemmas which are consequences of results one can find in [3].

Lemma 11. *[3, Lemma 3] Non-hyperelliptic symmetric semigroups have no children.*

Lemma 12. *Pseudo-symmetric semigroups have no grandchildren.*

Proof. From [3, Lemma 4] we deduce that there is a unique pseudo-semigroup of genus g with only one interval of non-gaps between 0 and the conductor. If $g \geqslant 5$ then this semigroup has only one child and no grandchildren. From [3, Lemma 5] we deduce that pseudo-semigroups with $\lambda_1 = 3$ have one child and no grandchildren. Finally, [3, Lemma 6] says that each pseudo-symmetric semigroup with $\lambda_1 \neq 3$ and with more than one interval of non-gaps between 0 and the conductor is a leaf in the semigroup tree. □

From these lemmas we deduce that the missexplored symmetric and pseudo-symmetric semigroups of genus 65 do not interfere in the computation of n_{68}. Indeed, these semigroups have at most one child and no grandchildren. Hence they do not contribute to n_{68}.

Computation of E_{66}

As a consequence of [3, Lemma 4], [3, Lemma 5], and [3, Lemma 6], we can state the following lemma.

Lemma 13. *The unique semigroups that are children of symmetric or pseudo-symmetric semigroups of a given genus $g \geqslant 5$ are:*

1. *The hyperelliptic semigroup of genus $g+1$*
2. *The semigroup $\{0, g, g+1, \ldots, 2g-3\} \cup [2g, \infty)$*
3. *The semigroup $\{0, 3, 6, \ldots, 3t, 3(t+1)-1, 3(t+1), 3(t+2)-1, 3(t+2), \ldots, 3(2t-1)-1, 3(2t-1)\} \cup \{3(2t-1)+2, 3(2t-1)+3\} \cup [3(2t-1)+5, \infty)$ for an adequate t*

4. *The semigroup* $\{0,3,6,\ldots,3t,3(t+1),3(t+1)+1,3(t+2),3(t+2)+1,\ldots,3(2t),3(2t)+1\}\cup\{3(2t)+3,3(2t)+4\}\cup[3(2t)+6,\infty)$ *for an adequate* t.

Now we can check that all semigroups in the previous lemma have non-negative Eliahou constant, thus proving that our algorithm using 128-bit integers guarantees the computation of E_{66}. Indeed,

1. The hyperelliptic semigroup of genus $g+1$ has parameters $m=2$, $p=2$, $k=g+1$, $r=1$, $q=g+1$, and $\rho=0$. Hence, the Eliahou constant is $(g+1)-(g+1)=0$.

2. The semigroup $\{0,g,g+1,\ldots,2g-3\}\cup[2g,\infty)$ has parameters $m=g$, $p=g-2$, $k=g-1$, $r=0$, $q=2$, and $\rho=0$. Hence, the Eliahou constant is $k(p-r)-q(m-r)+\rho=(g-1)(g-2)-2g=g^2-5g+2$, which is positive for $g\geqslant 5$.

3. The semigroup $\{0,3,6,\ldots,3t,3(t+1)-1,3(t+1),3(t+2)-1,3(t+2),\ldots,3(2t-1)-1,3(2t-1)\}\cup\{3(2t-1)+2,3(2t-1)+3\}\cup[3(2t-1)+5,\infty)$ for an adequate t has parameters $m=3$, $p=2$, $k=3t+1$, $r=0$, $q=2t+1$, and $\rho=1$. Hence, the Eliahou constant is $k(p-r)-q(m-r)+\rho=(3t+1)2-(2t+1)3+1=0$.

4. The semigroup $\{0,3,6,\ldots,3t,3(t+1),3(t+1)+1,3(t+2),3(t+2)+1,\ldots,3(2t),3(2t)+1\}\cup\{3(2t)+3,3(2t)+4\}\cup[3(2t)+6,\infty)$ for an adequate t. has parameters $m=3$, $p=2$, $k=3t+3$, $r=0$, $q=2t+2$, and $\rho=0$. Hence, the Eliahou constant is $k(p-r)-q(m-r)+\rho=(3t+3)2-(2t+2)3=0$.

These results allow us to conclude with the next lemmas.

Proposition 14. *The unique Eliahou semigroups of genus up to* 66 *are exactly* $\varepsilon_1,\varepsilon_2,\varepsilon_3,\varepsilon_4,\varepsilon_5,\varepsilon_6,\varepsilon_7$.

Corollary 15. *The Wilf conjecture holds for all semigroups of genus up to 66.*

9.3 Exploring up to genus 66

At this point we can say that we found all semigroups of E_{67} except the elements of E_{67} that have genus 67 and are children of semigroups of genus 66 of rank $g, g-1, g-2, g-3$, should they exist. The children of rank-g and rank-$(g-1)$ semigroups can be discarded from the results in the previous section. It remains as an open question to see whether there may be children of rank-$(g-2)$ and rank-$(g-3)$ semigroups with negative Eliahou constant.

Acknowledgment

The author would like to thank Julio Fernández-González for many helpful insights and comments and César Marín Rodríguez for his ideas and his contribution to the implementation of the algorithm. She would also like to thank Manuel Delgado and Shalom Eliahou for their comments and for noticing that two of the Eliahou semigroups of genus 67 are, in fact, Eliahou-Fromentin semigroups. Also, as mentioned above, Shalom Eliahou observed that a new family of semigroups could be defined with Eliahou constant equal to -1, containing the semigroup $\langle 19,26,27\rangle\,|_{90}$. The author would also like to thank Paul Shin for his careful reading of the manuscript and for his interesting observations as well as the referee that helped improving the proof of Lemma 3.

All the graphs have been drawn using the `drawsgtree` tool, which can be downloaded from `https://github.com/mbrasamoros/drawsgtree`.

The author was supported by the Spanish government under grant PID2021-124928NB-I00, and by the Catalan government under grant 2021 SGR 00115.

References

[1] M. Bras-Amorós. Fibonacci-like behavior of the number of numerical semigroups of a given genus. *Semigroup Forum*, 76(2):379–384, 2008.

[2] M. Bras-Amorós. Bounds on the number of numerical semigroups of a given genus. *J. Pure Appl. Algebra*, 213(6):997–1001, 2009.

[3] M. Bras-Amorós and S. Bulygin. Towards a better understanding of the semigroup tree. *Semigroup Forum*, 79(3):561–574, 2009.

[4] M. Bras-Amorós and J. Fernández-González. `https://github.com/mbrasamoros/RGD-algorithm`.

[5] M. Bras-Amorós and J. Fernández-González. Computation of numerical semigroups by means of seeds. *Math. Comp.*, 87(313):2539–2550, 2018.

[6] M. Bras-Amorós and J. Fernández-González. The right-generators descendant of a numerical semigroup. *Math. Comp.*, 89(324):2017–2030, 2020.

[7] M. Bras-Amorós and C. Marín Rodríguez. New Eliahou semigroups and verification of the Wilf conjecture for genus up to 65. In Torra and Narukawa, editors, *Modeling Decisions for Artificial Intelligence*, volume 12898 of *Lect. Notes Comput. Sci.*, pages 17–27. Springer, 2021.

[8] M. Delgado. On a question of Eliahou and a conjecture of Wilf. *Math.Z.*, 288(1-2):595–627, 2018.

[9] M. Delgado. Trimming the numerical semigroups tree to probe Wilf's conjecture to higher genus, 2019.

[10] M. Delgado. *Conjecture of Wilf: a survey*, volume 40 of *Springer INdAM Series*, pages 39–62. Springer International Publishing, 2020.

[11] S. Eliahou. Wilf's conjecture and Macaulay's theorem. *J. Eur. Math. Soc. (JEMS)*, 20(9):2105–2129, 2018.

[12] S. Eliahou and J. Fromentin. Near-misses in Wilf's conjecture. *Semigroup Forum*, 98(2):285–298, 2019.

[13] J. Fromentin and F. Hivert. `https://github.com/hivert/NumericMonoid`.

[14] J. Fromentin and F. Hivert. Exploring the tree of numerical semigroups. *Math. Comp.*, 85(301):2553–2568, 2016.

[15] J. C. Rosales and P. A. García-Sánchez. *Numerical semigroups*, volume 20 of *Developments in Mathematics*. Springer, New York, 2009.

[16] H. S. Wilf. A circle-of-lights algorithm for the "money-changing problem". *Amer. Math. Monthly*, 85(7):562–565, 1978.

A Illustrative examples

The examples in this appendix are included in order to illustrate all cases considered in Lemma 9.

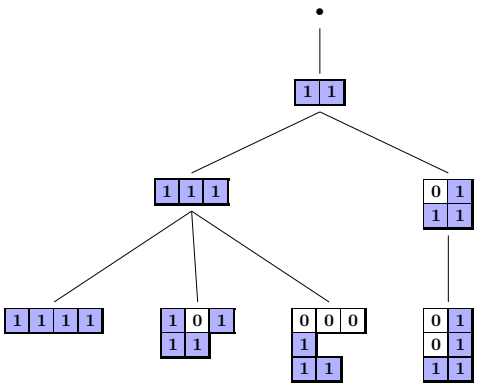

Figure 2: Seed tables of the children, grandchildren, and great grandchildren of $\{0, 1, \dots\}$

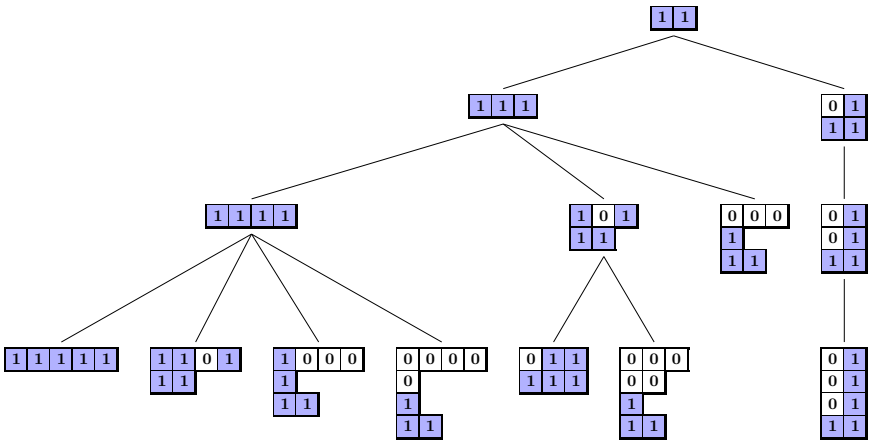

Figure 3: Seed tables of the children, grandchildren, and great grandchildren of $\{0, 2, \dots\}$

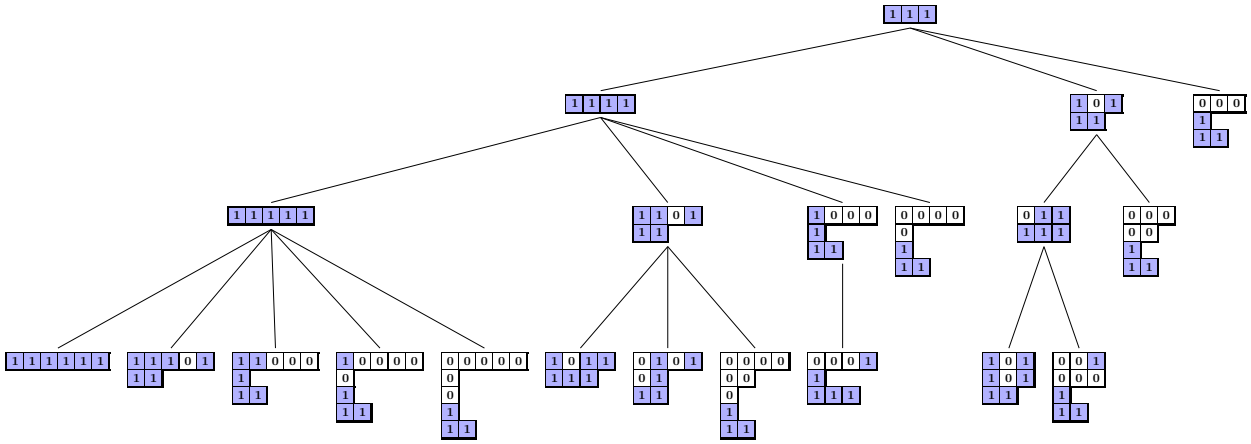

Figure 4: Seed tables of the children, grandchildren, and great grandchildren of $\{0, 3, \dots\}$

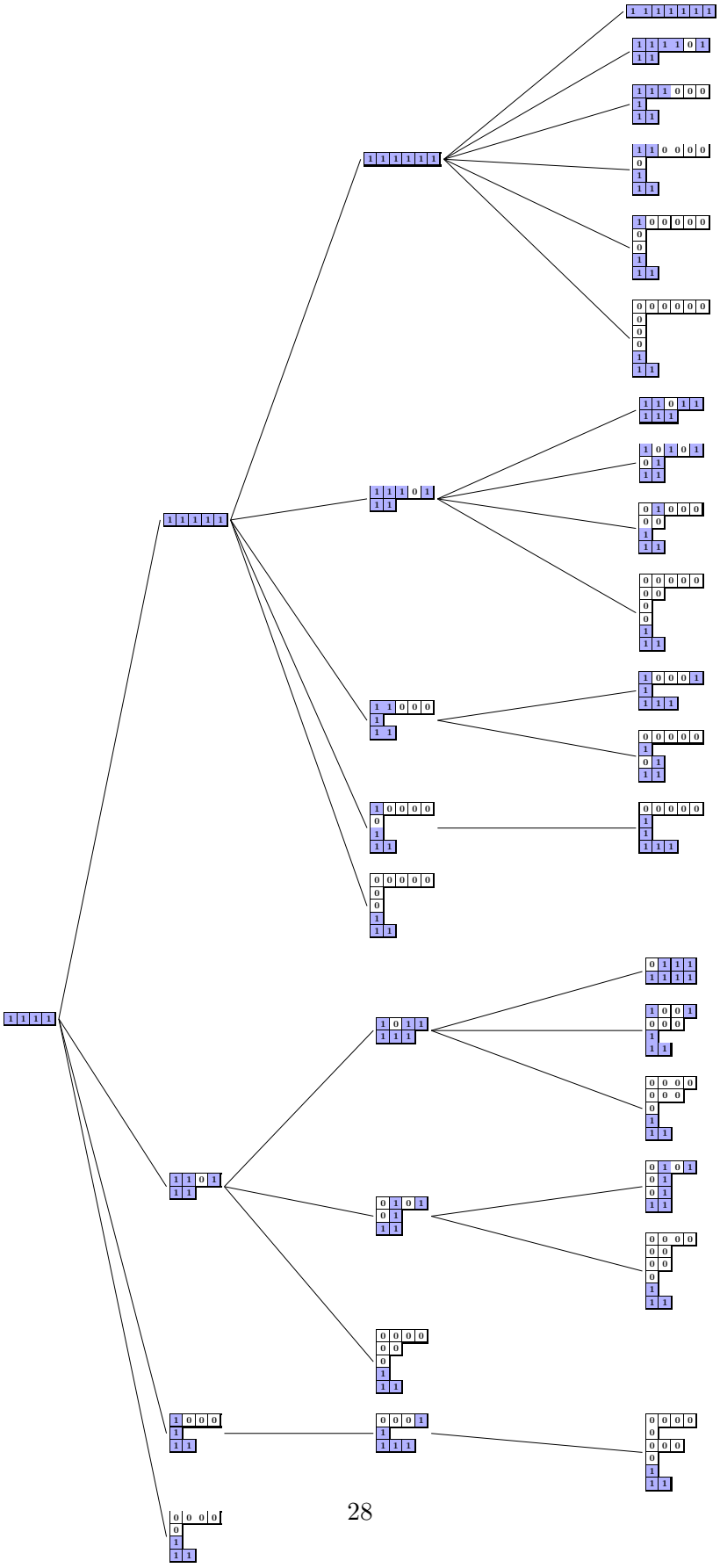

Figure 5: Seed tables of the children, grandchildren, and great grandchildren of $\{0, 4, \dots\}$

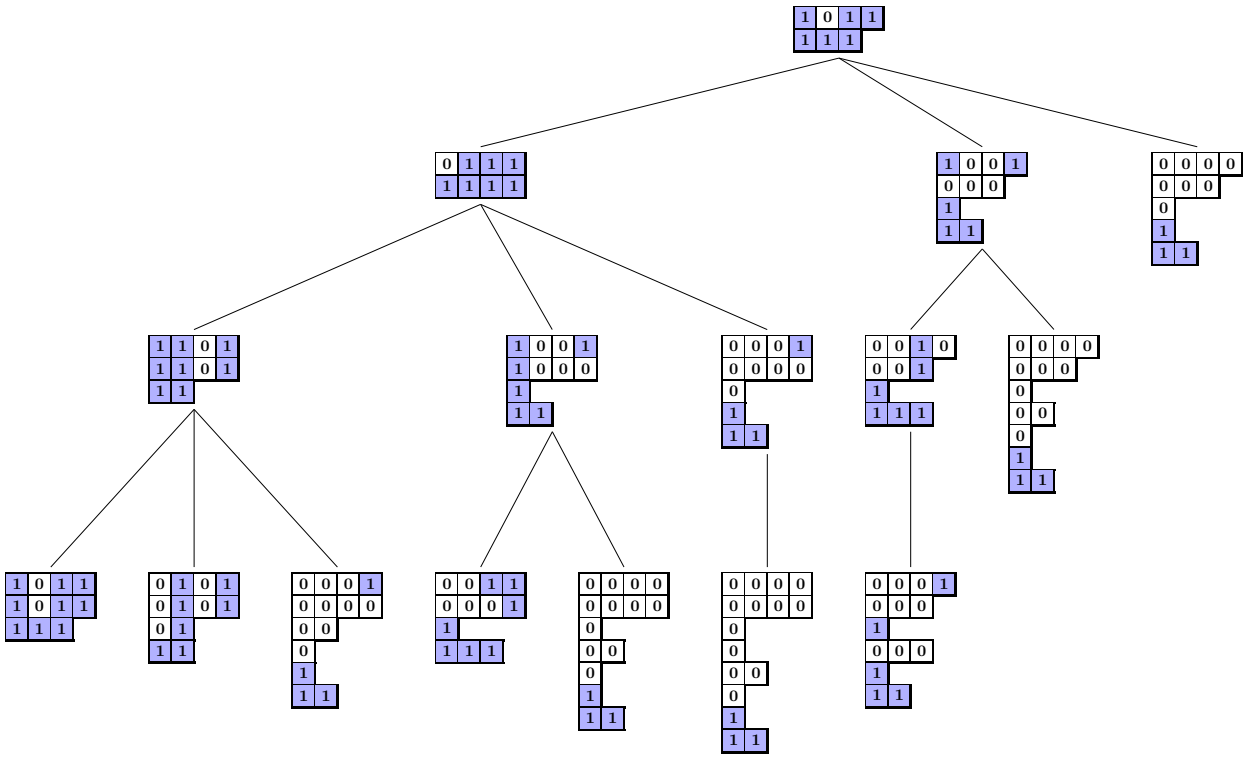

Figure 6: Seed tables of the children, grandchildren, and great grandchildren of $\{0, 4, 7, \dots\}$

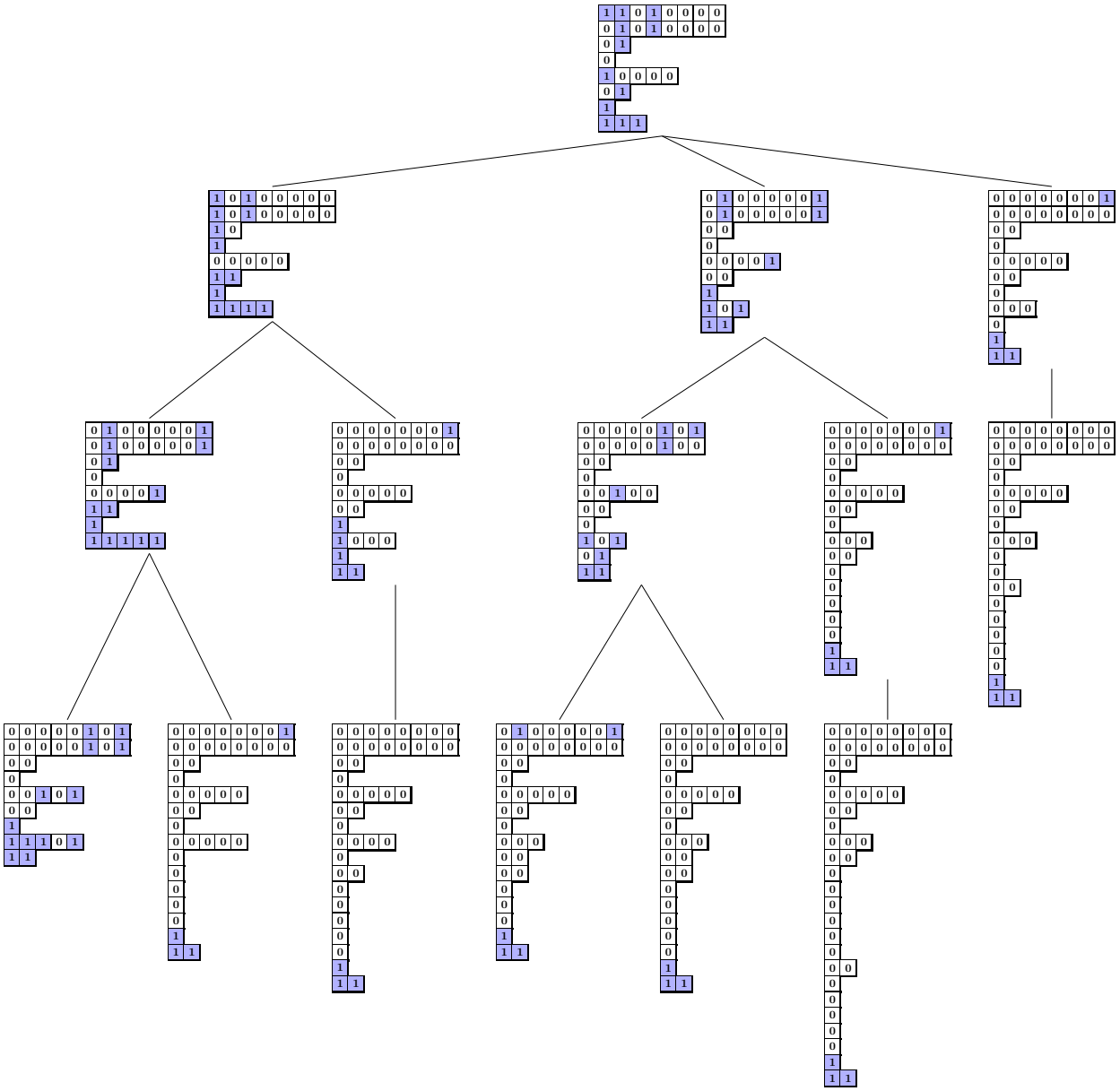

Figure 7: Seed tables of the children, grandchildren, and great grandchildren of $\{0, 8, 16, 18, 19, 24, 26, 27, 30, \dots\}$

B Implementation of the improved seeds algorithm

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef unsigned __int128 bbint;
int gamma;
long long int Sdesc(const bbint G, bbint S, const int c, const int m, const int u, const int v, int r, int gd) {
  int newc, s = 1, sold = 0;
  bbint A;
  long long int ng = 0;
  if (gd) {
    --gd;
    A = G;
    if (S & (bbint)1) {
      if (S & ((bbint)1 << m))
        ng += Sdesc(G | ((bbint)1 << (c - 1)), (S >> 1) | ((bbint)3 << (c - 1)), c + 1, m, u, v, r--, gd);
      else
        ng += Sdesc(G | ((bbint)1 << (c - 1)), (S >> 1) | ((bbint)3 << (c - 1)), c + 1, m, u, v, --r, gd);
    }
    while (r) {
      if (S & (bbint)1 << s) {
        newc = c + s + 1;
        for (int i = sold; i < s; i++) {
          A <<= 1;
          S &= A;
        }
        if (S & ((bbint)1 << (m + s)))
          ng += Sdesc(G | ((bbint)1 << (newc - 2)), S >> (s + 1) | ((bbint)7 << (newc - 3)), newc, m, u, v, r--, gd);
        else
          ng += Sdesc(G | ((bbint)1 << (newc - 2)), S >> (s + 1) | ((bbint)7 << (newc - 3)), newc, m, u, v, --r, gd);
        sold = s;
      }
      s++;
    }
  } else {
    A = S & (S >> m) & (((bbint)1 << u) - 1);
    s = 0;
    while (A) {
      s++;
      A &= A - 1;
    }
    ng += s * (r - 1);
    ng += r * (r - 1) * (r - 2) / 6;
    if (c > m + u) {
      A = (((bbint)1 << v) - 1) & S & (S >> u) & (S >> (m + u));
      while (A) {
        ng++;
        A &= A - 1;
      }
    }
  }
  return ng;
}
long long int lowrank(const int m, const int u, const int gamma) {
  int gd, min, mu, c;
  bbint G, S;
  long long int ng = 0;
  mu = m + u;
  if (u == m) {
    gd = gamma - mu;
    if (m < gd)
      min = m;
    else
      min = gd;
    c = mu;
    for (int v = 1; v < min; v++) {
      gd--;
      G = ((bbint)1 << c) - 1;
      c++;
      G -= ((bbint)1 << (m - 1));
      G -= ((bbint)1 << (mu - 1));
      S = ((bbint)1 << c) - 1;
      S -= ((bbint)1 << (m - v));
      S -= ((bbint)1 << (mu - v));
      ng += Sdesc(G, S - 1, c, m, u, v, m - 2, gd);
    }
    gd--;
    G = ((bbint)1 << c) - 1;
    c++;
    G -= ((bbint)1 << (m - 1));
    G -= ((bbint)1 << (mu - 1));
    S = ((bbint)1 << c) - 1;
    S -= ((bbint)1 << (m - min));
    S -= ((bbint)1 << (mu - min));
    ng += Sdesc(G, S, c, m, u, min, m - 1, gd);
  } else {
    if (m - u < gamma - mu)
      min = m - u;
    else
      min = gamma - mu;
    c = mu;
    gd = gamma - m - u;
    for (int v = 1; v < min; v++) {
```

```
93          gd−−;
94          G = ((bbint)1 << c) − 1;
95          c++;
96          G −= ((bbint)1 << (m − 1));
97          G −= ((bbint)1 << (mu − 1));
98          S = ((bbint)1 << c) − 1;
99          S −= ((bbint)1 << (m − u − v));
100         S −= ((bbint)1 << (m − v));
101         S −= ((bbint)1 << (mu − v));
102         if (u < v)
103           ng += Sdesc(G, S − 1, c, m, u, v, m − 4, gd);
104         else
105           ng += Sdesc(G, S − 1, c, m, u, v, m − 3, gd);
106       }
107       gd−−;
108       G = ((bbint)1 << c) − 1;
109       c++;
110       G −= ((bbint)1 << (m − 1));
111       G −= ((bbint)1 << (mu − 1));
112       S = ((bbint)1 << c) − 1;
113       S −= ((bbint)1 << (m − u − min));
114       S −= ((bbint)1 << (m − min));
115       S −= ((bbint)1 << (mu − min));
116       if (u < min)
117         ng += Sdesc(G, S, c, m, u, min, m − 3, gd);
118       else
119         ng += Sdesc(G, S, c, m, u, min, m − 2, gd);
120     }
121     return ng;
122   }
123   int main(int numvars, char **vars) {
124     long long int ng = 0;
125     bbint G, S;
126     int x, min;
127     time_t seconds, secondsafter;
128     gamma = (int)atoi(vars[1]);
129     seconds = time(NULL);
130     ng = (gamma − 4) * (gamma − 5) * (gamma − 6) / 6 + (gamma − 2) * (gamma − 3) + 8; // case m=gamma−2 and the hyperelliptic;
131     for (int m = 3; m < gamma − 2; m++) {
132       x = gamma − m − 1;
133       if (m − 1 < x)
134         min = m − 1;
135       else
136         min = x;
137       ng += Sdesc(((bbint)1 << (m − 1)) − 1, ((bbint)1 << m) − 8, m, m, 1, 1, m − 3, x − 1);
138       for (int v = 2; v < min; v++) {
139         G = ((bbint)1 << (m + v)) − 1;
140         G −= ((bbint)3 << (m − 1));
141         S = ((bbint)1 << (m + v + 1)) − 1;
142         S −= ((bbint)7 << (m − v − 1));
143         ng += Sdesc(G, S − 1, m + v + 1, m, 1, v, m − 4, x − v);
144       }
145       G = ((bbint)1 << (m + min)) − 1;
146       G −= ((bbint)3 << (m − 1));
147       S = ((bbint)1 << (m + min + 1)) − 1;
148       S −= ((bbint)7 << (m − min − 1));
149       ng += Sdesc(G, S, m + min + 1, m, 1, min, m − 3, x − min);
150       if (x > m) {
151         for (int u = 2; u <= m; u++)
152           ng += lowrank(m, u, gamma);
153       } else {
154         for (int u = 2; u < x; u++) {
155           ng += lowrank(m, u, gamma);
156         }
157         ng += (m − 1) * (m − 2) * (m − 3) / 6;
158         if (m < 2 * x)
159           ng += (x − 1) * (m − 2);
160         else
161           ng += x * (m − 2);
162         if (x == m || 2 * x != m)
163           ng++;
164         if (x == m − 1)
165           ng++;
166         if (x < m − 1 && 2 * x != m − 1)
167           ng += 2;
168       }
169     }
170     secondsafter = time(NULL);
171     printf("n%d=%lld-(without-parallelization)-time-taken-%d\n", (int)gamma, ng, (int)(secondsafter − seconds));
172     return 0;
173   }
```