Polynomial Time Generation Problems

Elmar Böhler and Klaus Wagner

Theoretische Informatik
Julius–Maximilians–Universität Würzburg
97074 Würzburg, Germany
[boehler,wagner]@informatik.uni-wuerzburg.de

Abstract. The generation problem for a binary polynomial time computable operation f on Σ^* is the question of whether a given element from Σ^* can be generated from other given elements from Σ^* by iterated application of f. For several subclasses of such operations like length-monotonic, associative and commutative operations, resp., we prove tight upper and lower bounds for their generation problems in terms of the classes of recursively enumerable languages and the complexity classes P, NP, PSPACE, and EXPTIME. Finally we consider the complexity of generation problems for bivariate polynomials with natural coefficients.

1 Introduction

How hard is it to find out whether an element can be generated from other elements by the iterated application of a given operation? Take, for example, the primality problem: A natural number n is non-prime if and only if it can be generated from $\{2, 3, \ldots, n-1\}$ by iterated multiplication. Only recently it became clear that this problem is in P, see [AKS02]. Or, as another example, what is the complexity of deciding whether an element of a monoid belongs to a finitely generated submonoid?

To make this precise, let f be an operation on A, i.e., a function $f : A \times A \to A$. For $B \subseteq A$, let $[B]_f$ be the f-closure of B, i.e. $[B]_f$ is the smallest set that contains B and which is closed under f. In this paper, we examine the following problem.

Generation problem GEN(A, f)
INPUT: $x_1, \ldots, x_m, z \in A$
QUESTION: Is z in $[\{x_1, \ldots, x_n\}]_f$?

To investigate this problem in the context of complexity theory, we consider $A = \Sigma^*$ for $\Sigma = \{0, 1\}$, and we assume f to be a polynomial time computable function. Depending on the context we will identify words from Σ^* with natural numbers by making use of one of the well-known bijections. Often we we will write the operation f in infix form. So for $x, y \in \Sigma^*$ the term xfy means $f(x, y)$. If we use infix form for a function we will usually denote it with symbols like $\circ$, $*$, $\cdot$, and $+$.

In Section 2 we will observe that the generation problem for every polynomial time operation is recursively enumerable, and that there are associative, commutative polynomial time operations whose generation problem is many-one complete for the class of all recursively enumerable sets.

To get decidable generation problems we restrict the class of operations in Section 3. We call an operation f on A *length-monotonic* if $|x| \leq |f(x,y)|$ and $|y| \leq |f(x,y)|$ for all $x, y \in A$. We call it *minimal length-monotonic* if there holds $|f(x,y)| = \max\{|x|, |y|\}$ for all $x, y \in A$. Obviously, every minimal length-monotonic operation is length-monotonic. We prove that the generation problem for every length-monotonic polynomial time operation is in EXPTIME and that there exists such an operation, which is additionally commutative, whose generation problem is many-one logspace complete for EXPTIME.

A further important restriction is associativity, which we deal with in Section 4. We prove that the generation problem for every length-monotonic associative polynomial time operation is in PSPACE and that there exists such an operation whose generation problem is many-one logspace complete for PSPACE. If such an operation is additionally commutative then its generation problem is in NP. And, in fact, the addition on natural numbers as a length-monotonic, associative, and commutative operation has an NP-complete generation problem.

This leads us in Section 5 directly to another class of operations, namely the class of bivariate polynomials with coefficients from natural numbers. If a polynomial is in fact univariate then its generation problem is in P. For all other polynomials we obtain an NP upper bound. That this bound is tight is shown by the fact that the generation problems for additon and multiplication on natural numbers is NP-complete.

2 All Polynomial Time Operations

In this section we will see what we can find out about the complexity of generation problems for all polynomial time computable operations. It will turn out that these problems do not fall in a definite complexity class. Moreover, there are polynomial time operations which have undecidable generation problems.

Theorem 1. *The generation problem* $\mathrm{GEN}(\Sigma^*, \circ)$ *is recursively enumerable for every polynomial time computable operation* $\circ$*.*

Proof: Consider the following algorithm working on $\circ$-formulae, i.e., formulae built up from words in Σ^* using the operation $\circ$. Given such a formula $F(x_1, \ldots, x_n)$ with $x_1, \ldots, x_n \in \Sigma^*$, the algorithm evaluates the formula and outputs $(x_1, \ldots, x_n, z)$ where z is the value of $F(x_1, \ldots, x_n)$. Obviously, the algorithm just enumerates $\mathrm{GEN}(\Sigma^*, \circ)$. $\square$

Theorem 2. *There is an associative, commutative, polynomial time computable operation* $\circ$ *such that the generation problem* $\mathrm{GEN}(\Sigma^*, \circ)$ *is m-complete for the class of recursively enumerable sets.*

Proof: Let $\varphi : \Sigma^* \to \Sigma^*$ be a recursive function such that $D_\varphi =_{\text{df}} \{x : \varphi(x) \text{ is defined}\}$ is the halting problem, and let M be a machine that computes φ. We define the function $\circ$ as follows: For $n, m_1, m_2 \geq 0$ let

$$0^{n+1}1^{m_1} \circ 0^{n+1}1^{m_2} =_{\text{df}} \begin{cases} 0^{n+1}1^{m_1+m_2}, & \text{if } M \text{ has not held after} \\ & \quad m_1 + m_2 \text{ steps on input } n \\ 1 & \text{otherwise,} \end{cases}$$

and for all other $x, y \in \Sigma^*$ let $x \circ y =_{\text{df}} 1$.

Observe, that $\circ$ is commutative and in FP. For the associativity let $x, y, z \in \Sigma^*$. In the case that there are $n, m_1, m_2, m_3 \geq 0$ such that $x = 0^{n+1}1^{m_1}$, $y = 0^{n+1}1^{m_2}$, $x = 0^{n+1}1^{m_3}$ and M has not held after $m_1 + m_2 + m_3$ steps on input n we obtain $x \circ (y \circ z) = (x \circ y) \circ z = 0^{n+1}1^{m_1+m_2+m_3}$. In all other cases we obtain $x \circ (y \circ z) = (x \circ y) \circ z = 1$.

Now, if M halts on input n after m steps then we obtain $[\{0^{n+1}1^1\}]_\circ = \{0^{n+1}1^1, 0^{n+1}1^2, \ldots, 0^{n+1}1^{m-1}, 1\}$. If M does not halt on input n then we have $[\{0^{n+1}1^1\}]_\circ = \{0^{n+1}1^1, 0^{n+1}1^2, \ldots\}$. Hence, $n \in D_\varphi \Leftrightarrow M$ halts on input $n \Leftrightarrow 1 \in [\{0^{n+1}1^1\}]_\circ \Leftrightarrow (0^{n+1}1, 1) \in \text{GEN}(\Sigma^*, \circ)$. □

3 Length-Monotonic Polynomial-Time Operations

We have seen in the last chapter that we have to restrict the class of our operations in order to get only decidable generation problems. Our way to do this is to bound the length of every intermediate result during the generation of an x by the length of x. That is just what we enforce with our notion of length-monotonicity.

Theorem 3. *The generation problem* $\text{GEN}(\Sigma^*, \circ)$ *is in* EXPTIME *for every length-monotonic, polynomial time computable operation* $\circ$.

Proof: Let $\circ$ be a length-monotonic, polynomial time computable operation. $\text{GEN}(\Sigma^*, \circ)$ can be decided by the following alternating algorithm that uses at most polynomial space:

```
function GEN(x1,...,xm,z)
  repeat
    if z ∈ {x1,...,xm} then accept;
    if |z| = 0 then reject;
    existentially choose z1 and z2 such that (z1 ∘ z2) = z;
    universally choose z from {z1,z2}
  forever
```

Since every polynomially space bounded alternating algorithm can be simulated by a deterministic exponentially time bounded algorithm [CKS81], the generation problem $\text{GEN}(\Sigma^*, \circ)$ is in EXPTIME. □

This exponential upper time bound for the generation problems of length-monotonic operations is tight. To show this we first establish the following technical lemma which will simplify some of the proofs below.

Lemma 4. *Let $A_1, \ldots, A_{k+l}$ be finite sets, and let $*$ be a polynomial time computable operation on $A_1^* \times \ldots \times A_k^* \times A_{k+1} \times \ldots \times A_{k+l}$. Then there exists a polynomial time computable operation $\circ$ on Σ^* such that the following holds.*

1. $\mathrm{GEN}(A_1^* \times \ldots \times A_k^* \times A_{k+1} \times \ldots \times A_{k+l}, *) \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\Sigma^*, \circ)$.
2. *If $*$ is commutative then $\circ$ is commutative.*
3. *If $*$ is associative then $\circ$ is associative.*
4. *If $*$ is minimal length-monotonic then $\circ$ is minimal length-monotonic.*

Proof: Let $m \geq 2$ be such that $|A_i| \leq 2^m$ for $i = 1, 2, \ldots, k+l$, let $h_i : A_i \to \Sigma^m$ be a block encoding with block length m for $i = 1, 2, \ldots, k+l$, let $d : \Sigma^* \to \Sigma^*$ be the homomorphism defined by $d(0) =_{\mathrm{df}} 00$ and $d(1) =_{\mathrm{df}} 11$, and let $\mathrm{code} : A_1^* \times \ldots \times A_k^* \times A_{k+1} \times \ldots \times A_{k+l} \to \Sigma^*$ be an encoding given by $\mathrm{code}(x_1, x_2, \ldots x_{k+l}) =_{\mathrm{df}} d(h_1(x_1))01d(h_2(x_2))01\ldots01d(h_{k+l}(x_{k+l}))$. Notice that $|\mathrm{code}(u)| = 2m|u| + 2(k+l-1)$. Now our operation $*$ on Σ^* can be defined by ($w_1, w_2 \in \Sigma^*$):

$$w_1 \circ w_2 =_{\mathrm{df}} \begin{cases} \mathrm{code}(u_1 * u_2) \text{ if } w_1 = \mathrm{code}(u_1) \text{ and } w_2 = \mathrm{code}(u_2) \\ 0^{\max(|w_1|,|w_2|)} \text{ otherwise.} \end{cases}$$

Obviously, if $*$ is commutative then so is $\circ$, and if $*$ is associative then so is $\circ$. Now let $*$ be minimal length-monotonic. If $w_1 = \mathrm{code}(u_1)$, $w_2 = \mathrm{code}(u_2)$, and $u_1 * u_2 = v$ then we can conclude:

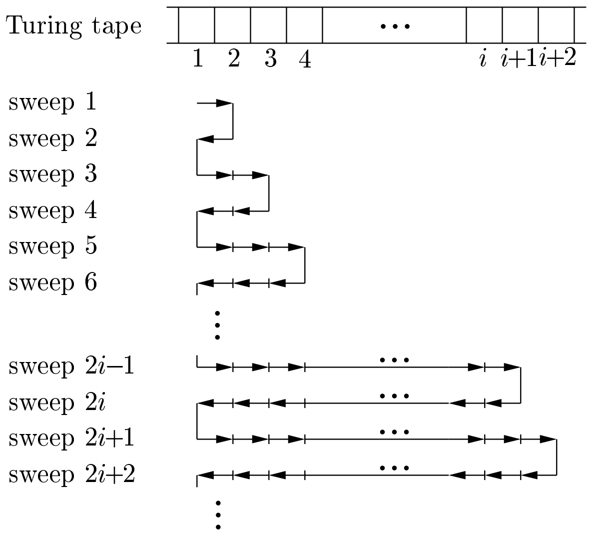
$$\begin{aligned} |w_1 \circ w_2| = |\mathrm{code}(u_1 * u_2)| = |\mathrm{code}(v)| &= 2m|v| + 2(k+l-1) \\ &= 2m \cdot \max(|u_1|, |u_2|) + 2(k+l-1) \\ &= \max(2m|u_1| + 2(k+l-1), 2m|u_2| + 2(k+l-1)) \\ &= \max(|\mathrm{code}(u_1)|, |\mathrm{code}(u_2)|) = \max(|w_1|, |w_2|). \end{aligned}$$

Otherwise, $|w_1 \circ w_2| = |0^{\max(|w_1|,|w_2|)}| = \max(|w_1|, |w_2|)$. Hence, $\circ$ is minimal length-monotonic.

Finally, by the definition of $\circ$ there holds $v \in [\{u_1, \ldots, u_m\}]_*$ if and only if $\mathrm{code}(v) \in [\{\mathrm{code}(u_1), \ldots, \mathrm{code}(u_m)\}]_\circ$. This completes the proof. □

Theorem 5. *There is a commutative, minimal length-monotonic, polynomial time computable operation $\circ$ such that the generation problem $\mathrm{GEN}(\Sigma^*, \circ)$ is logarithmic space m-complete for* EXPTIME.

Proof: We follow the idea of Cook in [Coo71] to simulate deterministic exponential time computations. W.l.o.g., a deterministic exponential time one-tape Turing machine M deciding a set $A \subseteq \Sigma^*$ can be normalized in such a way that its work on input $x = a_1 a_2 \ldots a_n$ constitutes of $2^{p(|x|)}$ *sweeps*, where p is a suitable polynomial. For $0 \leq 2i < 2^{p(|x|)}$, the $(2i+1)$th sweep is a right move from tape cell 1 (with the first symbol of x) to tape cell $i+2$ within $i+1$ steps, and the $(2i+2)$th sweep is a left move from tape cell $i+2$ to tape cell 1 within $i+1$ steps. Each of the turning points belongs to two sweeps.



Furthermore, let M have the tape alphabet Δ, the set S of states, the initial state s_0, and the accepting state s_1. In the case of acceptance the tape of M is empty. If M is in state s and reads a than the next state is $\sigma(s,a)$, and the symbol printed is $\lambda(s,a)$.

We say that the quintuple (x,i,j,s,a) is *correct* if during the ith sweep on input x the machine M prints the symbol a in tape cell j and leaves that cell with state s. One can compute a correct (x,i,j,s,a) by knowing only two other correct quintuples, namely the correct $(x,i-1,j,s',a')$ and the correct (x,i,k,s'',a'') where $k \in \{j-1,j+1\}$. This is just the idea to define our operation: multiply $(x,i-1,j,s',a')$ with (x,i,k,s'',a'') and get (x,i,j,s,a). In an accepting computation of M on x (and only in this case) one generates finally the correct $(x,2^{p(|x|)},2,s_1,\Box)$.

To make this precise, let $a_j =_{\mathrm{df}} \Box$ for all $j > n$. Furthermore we sassume that, in a quintuple (x,i,j,s,a) such that $i,j \in \{0,1,\ldots,2^{p(|x)}\}$, the numbers i and j are given in binary presentation of length exactly $p(|x|)+1$. Now we define our multiplication $*$ as follows.

Right sweep, for $1 \leq 2i < 2^{p(|x|)}$ and $j = 1,2,\ldots,i$:

$$(x,2i,j+1,s,a) * (x,2i+1,j,s',b) =_{\mathrm{df}} (x,2i+1,j+1,\sigma(s',a),\lambda(s',a))$$

Left sweep, for $1 \leq 2i+1 < 2^{p(|x|)}$ and $j = 1,2,\ldots,i+1$:

$$(x,2i+1,j,s,a) * (x,2i+2,j+1,s',b) =_{\mathrm{df}} (x,2i+2,j,\sigma(s',a),\lambda(s',a))$$

New tape cell right, for $1 \leq 2i+1 < 2^{p(|x|)}$:

$$(x,2i+1,i+1,s,a) * (x,0,0,s_0,\Box) =_{\mathrm{df}} (x,2i+1,i+2,\sigma(s,a_{i+2}),\lambda(s,a_{i+2}))$$

Turning point left, for $1 \leq 2i < 2^{p(|x|)}$:

$$(x,2i,1,s,a) * (x,0,0,s_0,\Box) =_{\mathrm{df}} (x,2i+1,1,s,a)$$

Turning point right, for $1 \leq 2i+1 < 2^{p(|x|)}$:

$$(x,2i+1,i+2,s,a) * (x,0,0,s_0,\Box) =_{\mathrm{df}} (x,2i+2,i+2,s,a)$$

If $u * v$ is defined in this way then $v * u$ is defined in the same way. For all other products, not already defined, we define $(x, u, v, s, a) * (x', u', v', s', a') =_{\mathrm{df}} (0^{\max(|x|+|u|+|v|, |x'|+|u'|+|v'|)}, \varepsilon, \varepsilon, s_0, \Box)$.

Obviously, $*$ is polynomial time computable, minimal length-monotonic and commutative. By our definition, starting with $(x, 1, 1, \sigma(s_0, a_1), \lambda(s_0, a_1))$ and $(x, 0, 0, s_0, \Box)$ one can generate exactly the correct quintuples of the form $(x, \dots)$ together with $(x, 0, 0, s_0, \Box)$ and $(0^{|x|+p(|x|)+2}, \varepsilon, \varepsilon, s_0, \Box)$. Hence, M accepts x if and only if $((x, 1, 1, \sigma(s_0, a_1), \lambda(s_0, a_1)), (x, 0, 0, s_0, \Box), (x, 2^{p(|x|)}, 2, s_1, \Box)) \in \mathrm{GEN}(\Sigma^*, *)$, and consequently $A \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\Sigma^* \times \Sigma^* \times \Sigma^* \times S \times \Delta, *)$. By Lemma 4 we finally obtain a polynomial time computable, minimal length-monotonic and commutative operation $\circ$ on Σ^* such that $A \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\Sigma^*, \circ)$. $\Box$

4 Length-Monotonic Associative Polynomial-Time Operations

So commutativity does not lower the complexity of the generation problem for length-monotonic operations in general. In the following, we will see, that associativity does. If we ask whether $z \in [\{x_1, \dots, x_n\}]_f$ we search for a binary tree where the leaves are elements from $\{x_1, \dots, x_n\}$, each other node is labeled with results of applications of f on its two successor nodes and the root node is labeled with z. But if f is associative, we no longer need to know the structure of this tree; it suffices to know the sequence of its leaves.

Theorem 6. *For every length-monotonic and associative polynomial time computable operation* $\circ$ *the problem* $\mathrm{GEN}(\Sigma^*, \circ)$ *is in* PSPACE.

Proof: The following algorithm decides $\mathrm{GEN}(\Sigma^*, \circ)$ in polynomial space:

```
function GEN(z,x1,...,xn);
  choose an i ∈ {1,...,n} nondeterministically;
  z1 := zi;
  while (z1 ≠ z) and (|z1| ≤ |z|) do begin
    choose an i ∈ {1,...,n} nondeterministically;
    z1 := z1 ∘ xi
  end;
  if (z = z1) then accept else reject
```
$\Box$

The next theorem shows that the polynomial space bound for the generation problem of length-monotonic, associative operations is tight.

Theorem 7. *There is a minimal length-monotonic and associative polynomial time computable operation* $\circ$ *such that* $\mathrm{GEN}(\Sigma^*, \circ)$ *is logarithmic space m-complete for* PSPACE.

Proof: Let $L \subseteq \{0, 1\}^* = \Sigma^*$ be a set which is $\leq_{\mathrm{m}}^{\log}$-complete for PSPACE. By Lemma 4 it is sufficient to prove that there exist a finite alphabet Δ and a minimal length-monotonic and associative polynomial time computable operation $*$ such that $L \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\Sigma^* \times \Delta^*, *)$.

Since $L \in \mathrm{PSPACE}$, it follows from a result from [CF91] that there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \mathrm{A}_5$ and a polynomial p such that for all $x \in \Sigma^*$ there holds $x \in L \leftrightarrow f(x,0)\cdot f(x,1)\cdot\ldots\cdot f(x,2^{p(|x|)}-1) = a_0$, where $(\mathrm{A}_5,\cdot)$ is the group of even permutations on five elements with the identity permutation a_0. We fix an $x \in \Sigma^*$ and set $K =_{\mathrm{df}} p(|x|)$ as well as $M =_{\mathrm{df}} 4K+3$. Further, for $i = 0,1,\ldots,2^K-1$, let $\mathrm{b}(i)$ be that binary presentation of i which has length exactly K.

The basic idea is to consider the set

$$E'_x =_{\mathrm{df}} \{(x, \mathrm{b}(i)\, a\, \mathrm{b}(j)) : 0 \le i < j < 2^K, a \in \mathrm{A}_5\} \subseteq \{x\} \times (\Sigma^K \cdot \mathrm{A}_5 \cdot \Sigma^K)$$

with a multiplication $*$ whose most essential part is given by $(x, \mathrm{b}(i)\, a\, \mathrm{b}(j)) * (x, \mathrm{b}(j{+}1)\, b\, \mathrm{b}(m)) =_{\mathrm{df}} (x, \mathrm{b}(i)\, a \cdot f(x,j) \cdot b\, \mathrm{b}(j))$. Then we obtain

$$x \in L \leftrightarrow (x, 0^K f(x, 2^K{-}1)^{-1} 1^K) \in [\{(x, \mathrm{b}(i) f(x,i) \mathrm{b}(i{+}1)) : 0 \le i < 2^K{-}1\}]_*.$$

However, this cannot be used for a reduction $L \le_{\mathrm{m}}^{\log} \mathrm{GEN}(\Sigma^* \times (\{0,1\} \cup \mathrm{A}_5)^*, *)$ because the set $\{(x, \mathrm{b}(i) f(x,i) \mathrm{b}(i{+}1)) : 0 \le i < 2^K{-}1\}$ has exponentially many elements (in the length of x). So we have to generate with the same operation $*$ also the set $\{(x, \mathrm{b}(i) f(x,i) \mathrm{b}(i{+}1)) : 0 \le i < 2^K{-}1\}$ from a few basic pairs. Hence we modify the set E'_x and the operation $*$ as follows. We use a new *separation symbol* # and, to achieve minimal length-monotonicity, a new *padding* symbol 2. For $u \in \{0,1,\#\}^*$ we define $\langle u \rangle =_{\mathrm{df}} u 2^{M-|u|} \in \{0,1,2,\#\}^M$, and for $w \in \{0,1,2,\#\}^*$ we define $\overline{w} \in \{0,1,\#\}^*$ as the word w without the symbols 2. Now we consider the set $E_x \subseteq \{x\} \times (\{0,1,2,\#\} \cup \mathrm{A}_5)^M$ defined as

$$\begin{aligned} E_x =_{\mathrm{df}} \{(x, \langle u \rangle) : u \in {} & \Sigma^{\le K} \cup \Sigma^{\le K} \# \Sigma^{\le K} \cup \Sigma^{\le K} \# \Sigma^K \# \Sigma^{\le K} \\ & \cup \Sigma^{\le K} \# \Sigma^K \cdot \mathrm{A}_5 \cdot \Sigma^K \# \Sigma^{\le K}\} \end{aligned}$$

together with the multiplication $*$ which works on E_x as follows:

$$(x, w_1) * \ldots * (x, w_r) =$$

$$\begin{cases} (x, \langle w_1 \ldots w_r \rangle) \text{ if } \overline{w_1 \ldots w_r} \in \Sigma^{\le K} \cup \Sigma^{\le K} \# \Sigma^{\le K} \cup \Sigma^{\le K} \# \Sigma^K \# \Sigma^{\le K} \\ (x, \langle u \# \mathrm{b}(i_1) b_1 \cdot b_2 \cdot \ldots \cdot b_{s-1} \mathrm{b}(i_s) \# v \rangle), \text{ where } b_j = c_j \text{ if } c_j \in \mathrm{A}_5 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and } b_j = f(x, i_j) \text{ otherwise} \\ \quad \text{if } \overline{w_1 \ldots w_r} = u \# \mathrm{b}(i_1) c_1 \mathrm{b}(i_2) c_2 \ldots c_{s-1} \mathrm{b}(i_s) \# v \text{ such that } s \ge 2, \\ \quad u, v \in \Sigma^{\le K}, 0 \le i_1 < i_2 < \ldots < i_s < 2^K, c_1, c_2, \ldots, c_{s-1} \in \mathrm{A}_5 \cup \{\#\}, \\ \quad \text{and } c_j = \# \text{ implies } i_j + 1 = i_{j+1} \text{ for } j = 1, \ldots, s-1 \\ (x, \langle \varepsilon \rangle) \text{ otherwise} \end{cases}$$

For $r = 2$, take this as the definition of $*$. For $r > 2$ this is an observation which shows that in the case of more than two factors the result does not depend on the order in which the multiplications are carried out. Hence the operation $*$ is associative on E_x. Furthermore we observe $(x, \langle u \# \mathrm{b}(i) a \mathrm{b}(j) \# v \rangle)$ is in $[\{(x, \langle 0 \rangle), (x, \langle 1 \rangle), (x, \langle \# \rangle)\}]_*$ if and only if $i < j$ and $f(x,i) \cdot f(x,i+1) \cdot \ldots \cdot f(x,j-1) = a$. Consequently, we obtain

$$x \in L \leftrightarrow (x, \langle \# 0^K f(x, 2^K{-}1)^{-1} 1^K \# \rangle) \in [\{(x, \langle 0 \rangle), (x, \langle 1 \rangle), (x, \langle \# \rangle)\}]_*.$$

So far the operation $*$ has been defined separately on every single set E_x. W.l.o.g assume that $\varepsilon \notin L$. If there is no $x \in \Sigma^*$ such that $(y,v), (z,w) \in E_x$ then we define $(y, v) * (z, w) =_{\mathrm{df}} (\varepsilon, 2^{\max(|y|+|z|,|z|+|w|)})$. In such a way we obtain an associative, minimal length-monotonic operation $*$ on the entire set $\Sigma^* \times (\{0, 1, 2, \#\} \cup \mathrm{A}_5)^*$ such that $L \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\Sigma^* \times (\{0, 1, 2, \#\} \cup \mathrm{A}_5)^*, *)$. □

What about the generation problem with an operation $\circ$ that is length-monotonic, associative, and commutative? Again, if we want to know whether or not $z \in [\{x_1, \ldots, x_n\}]_\circ$, associativity enables us to ignore the derivation tree and just search for a word over $\{x_1, \ldots, x_n\}$. Together with commutativity, we just have to search guess exponents $k_1, \ldots, k_n$ and test whether $x^{k_1} \circ x^{k_2} \circ \cdots \circ x^{k_n} = z$. The exponentiations are computable in polynomial time, using the well-known technique of squaring and multiplying, hence the following theorem is obvious.

Theorem 8. *For all length-monotonic, associative, and commutative operations $f \in \mathrm{FP}$ the problem $\mathrm{GEN}(\Sigma^*, f)$ is in NP.*

Also this upper bound is tight, i.e., there exist associative, commutative, and length-monotonic operations whose generation problems are NP-complete. We will see that the operation of addition on natural numbers has this property.

Theorem 9. *The problems $\mathrm{GEN}(\mathbb{N}, +)$ is $\leq_{\mathrm{m}}^{\log}$-complete for NP, where $+$ is the addition on $\mathbb{N}$.*

Proof: In [vEB79] it is shown, that $\mathrm{GEN}(\mathbb{Z}, +)$ is $\leq_{\mathrm{m}}^{\log}$-complete for NP. The reduction there uses only natural numbers, i.e., it is in fact a reduction to $\mathrm{GEN}(\mathbb{N}, +)$. □

5 Polynomials

We have seen that the generation problem for addition on natural numbers is NP-complete. But what about other bivariate polynomials with natural coefficents? We start with bivariate polynomials p which are in fact univariate, i.e., for which there exists a univariate polynomial q such that $p(x, y) = q(x)$ for all $x, y \in \mathbb{N}$. In what follows we denote the binary description of a natural number n by $\mathrm{bin}(n)$.

Theorem 10. *If p is a univariate polynomial then $\mathrm{GEN}(\mathbb{N}, p)$ is in P.*

Proof: If $p(x, y) = q(x) = c$ then we have $[\{a_1, a_2, \ldots, a_n\}]_p = \{a_1, a_2, \ldots, a_n, c\}$. If $p(x, y) = q(x) = x+c$ then $[\{a_1, a_2, \ldots, a_n\}]_p = \{a_i+kc : i = 1, 2, \ldots, r, k \geq 0\}$. In all other cases we have $q(x) \geq 2x$ or $q(x) \geq x^2$ from which we conclude $e \in [\{a_1, a_2, \ldots, a_n\}]_p \Leftrightarrow e \in \{p_k(a_i) : i = 1, \ldots, n, k + 0, 1, \ldots, |\mathrm{bin}(e)| + 1\}$ where $p_0(x) =_{\mathrm{df}} x$ and $p_{k+1}(x) =_{\mathrm{df}} p(p_k(x))$ for $k \geq 0$.

Hence in all cases the membership to $\{a_1, a_2, \ldots, a_n\}]_p$ can easily be verified in polynomial time. □

Theorem 11. *If the polynomial p is bivariate then $\mathrm{GEN}(\mathbb{N}, p)$ is in NP.*

Proof: The proof is given by the following three lemmas. □

A polynomial $p(x, y)$ is said to be *linear* if there are $a, b, c \in \mathbb{N}$ such that $p(x, y) = ax + by + c$.

Lemma 12. *If the polynomial p with natural coefficients is not linear then* $\mathrm{GEN}(\mathbb{N}, p)$ *is in* NP.

Proof: Let $A \subseteq \mathbb{N}$ be finite. To decide $b \in [A]_p$ we consider three cases.
Case $b = 0$. Observe that $p(x, y) = 0$ implies $x = 0$ or $y = 0$. Hence $0 \in [A]_p$ if and only if $0 \in A$.
Case $b = 1$. Observe that $p(x, y) = 1$ implies

- If the absolute term of p is 1 then ($x = 0$ or $y = 0$) and $p(0, 0) = 1$
- If the absolute term of p is 0 and p has no mixed term $x^i y^j$ then ($x = 0$, $y = 1$, and $p(0, 1) = 1$) or ($x = 1$, $y = 0$, and $p(1, 0) = 1$).
- If the absolute term of p is 0 and p has mixed terms then ($x = 1$, $y = 1$, and $p(1, 1) = 1$).

Hence $1 \in [A]_p$ if and only if $1 \in A$ or ($0 \in A$ and $p(0, 0) = 1$).
Case $b \geq 2$. The previous cases have shown that there exists a generation tree for b where 0 and 1 can appear only in the leaf nodes or in the nodes next to them. Let $p(A) =_{\mathrm{df}} A \cup \{p(x, y) : x, y \in A\}$, and let $B =_{\mathrm{df}} (A \cup \{p(x, y) : x, y \in p^1(A)\}) \setminus \{0, 1\}$. Now we have $b \in [A]_p$ if and only if $b \in [B]_p$. Since p is not linear we have $p(x, y) \geq x \cdot y$ or $p(x, y) \geq x^2$ or $p(x, y) \geq y^2$. If $b \in [B]_p$ and the binary generation tree for b has m leaves (with elements from B) then $b \geq 2^m$. Hence the generation tree for b has at most $2|b|$ nodes which can be guessed and checked nondeterministically in polynomial time. □

Lemma 13. *If $p(x, y) = x + by + c$ such that $b, c \in \mathbb{N}$ then* $\mathrm{GEN}(\mathbb{N}, p)$ *is in* NP.

Proof: We show the following equation:

$$[\{a_1, \ldots, a_r\}]_p = \{a_j + \Sigma_{i=1}^r \alpha_i \cdot (ba_i + c) : j \in \{1, \ldots, r\} \text{ and } \alpha_1, \ldots, \alpha_r \in \mathbb{N}\}$$

The inclusion from right to left is obvious. For the other direction, we prove that $\{a_1, \ldots, a_r\}$ is included in the right hand side (which is obvious) and that the right hand side is closed under p. For the latter let $\alpha_i, \beta_i \in \mathbb{N}$ for $1 \leq i \leq r$ and $j, k \in \{1, \ldots, r\}$. Then

$$\begin{aligned} & p(a_j + \Sigma_{i=1}^r(\alpha_i \cdot (ba_i + c)), a_k + \Sigma_{i=1}^r \beta_i \cdot (ba_i + c)) \\ & \quad = a_j + \Sigma_{i=1}^r(\alpha_i \cdot (ba_i + c)) + b(a_k + \Sigma_{i=1}^r(\beta_i \cdot (ba_i + c))) + c \\ & \quad = a_j + \Sigma_{i=1}^r(\alpha_i \cdot (ba_i + c)) + (ba_k + c) + b \cdot \Sigma_{i=1}^r(\beta_i \cdot (ba_i + c)) \\ & \quad = a_j + \Sigma_{i=1}^r((\alpha_i + b\beta_i) \cdot (ba_i + c)) + (ba_k + c) \end{aligned}$$

□

Lemma 14. *If $p(x, y) = ax + by + c$ such that $a, b \in \mathbb{N}$ and $a, b \geq 2$ then* $\mathrm{GEN}(\mathbb{N}, p)$ *is in* NP.

Proof: We observe that $e \in [\{a_1, \ldots, a_n\}]_p$ if and only if there exists a binary tree t whose nodes have labels from natural numbers such that

- every leaf of t has a label from $\{a_1, \ldots, a_n\}$,
- if the left son and the right son of a node have label k and l, resp., then this node has label $ak + bl + c$, and
- the root of t has label e.

W.l.o.g. we can assume that label 0 occurs only in the leaves of this tree t. Since $a, b \geq 2$, the depth of t is bounded by $|\mathrm{bin}(e)| + 1$.

Now let t be an arbitrary binary tree whose leaves have labels from $\{a_1, \ldots, a_n\}$. We call a path in t from the root upward to a leaf a *full path*, and we call a proper initial part of a full path an *initial path*. For a full or initial path q, let $l(q)$ and $r(q)$ be the number of left steps and right steps, resp., in q. For a full path q in t, let $i(q) \in \{1, \ldots, n\}$ be such that the leaf of q has label $a_{i(q)}$.

Now we obtain that $e \in [\{a_1, \ldots, a_n\}]_p$ if and only if there exists a binary tree t whose leaves have labels from $\{a_1, \ldots, a_n\}$ such that

$$e = \sum_{q \text{ full path in } t} a_{i(q)} \cdot a^{l(q)} \cdot b^{r(q)} + \sum_{q \text{ initial path in } t} c \cdot a^{l(q)} \cdot b^{r(q)}.$$

For a binary tree t of depth bounded by d we define the characteristics

$$\begin{aligned} s^t_{i,j} &=_{\mathrm{df}} \#\{q : q \text{ initial path in } t \text{ with } l(q) = i \text{ and } r(q) = j\} \text{ and} \\ r^t_{i,j} &=_{\mathrm{df}} \#\{q : q \text{ full path in } t \text{ with } l(q) = i \text{ and } r(q) = j\} \end{aligned}$$

for $i, j \in \{0, 1, \ldots, d\}$. Note that the $r^t_{i,j}$ can be computed from the $s^t_{i,j}$ by

$$\left.\begin{array}{l} - \ r^t_{0,0} = 1 - s^t_{0,0}, \\ - \ r^t_{0,j+1} = s^t_{0,j} - s^t_{0,j+1} \text{ for } j \in \{0, 1, \ldots, d\}, \\ - \ r^t_{i+1,0} = s^t_{i,0} - s^t_{i+1,0} \text{ for } i \in \{0, 1, \ldots, d\}, \text{ and} \\ - \ r^t_{i+1,j+1} = s^t_{i,j+1} + s^t_{i+1,j} - s^t_{i+1,j+1} \text{ for } i, j \in \{0, 1, \ldots, d\}. \end{array}\right\} \qquad (*)$$

Using these characteristics we obtain that $e \in [\{a_1, \ldots, a_n\}]_p$ if and only if there exist a binary tree t of depth $d \leq |\mathrm{bin}(e)| + 1$ and $\{r_{i,j,k} : i, j \in \{0, 1, \ldots, d\}, k \in \{1, \ldots, n\}\} \subset \mathbb{N}$ such that $\sum_{k=1}^{n} r_{i,j,k} = r^t_{i,j}$ and

$$e = \sum_{i=0}^{d} \sum_{j=0}^{d} \left(\sum_{k=1}^{n} r_{i,j,k} \cdot a_k \right) \cdot a^i \cdot b^j + \sum_{i=0}^{d} \sum_{j=0}^{d} s^t_{i,j} \cdot c \cdot a^i \cdot b^j.$$

Observe that the characteristics $s^t_{i,j}$ of t have the following properties (**):

$$\left.\begin{array}{l} - \ s^t_{0,0} \leq 1, \\ - \ s^t_{0,j+1} \leq s^t_{0,j} \text{ for } j \in \{0, 1, \ldots, d-1\}, \\ - \ s^t_{i+1,0} \leq s^t_{i,0} \text{ for } i \in \{0, 1, \ldots, d-1\}, \\ - \ s^t_{i+1,j+1} \leq s^t_{i+1,j} + s^t_{i,j+1}, \text{ for } i, j \in \{0, 1, \ldots, d-1\}, \mathit{and} \\ - \ s^t_{i,d} = s^t_{d,j} = 0 \text{ for } i, j \in \{0, 1, \ldots, d\}. \end{array}\right\} \qquad (**)$$

On the other hand, we can prove the following

Claim. If the collection $M = \{s_{i,j} : i, j \in \{0, 1, \ldots, d\}\} \subset \mathbb{N}$ fulfills (**) then there exists a binary tree t such that $s^t_{i,j} = s_{i,j}$ for $i, j \in \{0, 1, \ldots, d\}$.

Proof of the claim. By induction on $w(M) =_{\mathrm{df}} \sum_{i=0}^{d} \sum_{j=0}^{d} s_{i,j}$.
If $w(M) = 0$ then the tree with only one node fulfills the statement.
If $w(M) > 0$ then we have $s_{0,0} > 0$. Since $s_{i,d} = s_{d,j} = 0$ for $i, j \in \{0, 1, \ldots, d\}$ there exists a pair $(i, j) \in \{0, 1, \ldots, d\}^2$ such that $s_{i,j} > 0$ and $s_{i+1,j} = s_{i,j+1} = 0$. Let (i_0, j_0) be such a pair. Define $M' = \{s'_{i,j} : i, j \in \{0, 1, \ldots, d\}\}$ such that

$s'_{i_0,j_0} =_{\mathrm{df}} s_{i_0,j_0} - 1$ and $s'_{i,j} = s_{i,j}$ for all other $(i,j) \in \{0,1,\ldots,d\}^2$. Obviously, M' fulfills (**) and $w(M') = w(M) - 1$. By the induction hypothesis, there exists a binary tree t' such that $s^{t'}_{i,j} = s'_{i,j}$ for $i,j \in \{0,1,\ldots,d\}$. To know that there exists a full path q in t' such that $l(q) = i_0$ and $r(q) = j_0$ we have to prove $r^{t'}_{i_0,j_0} > 0$. We do the latter by considering four cases.
If $i_0 = j_0 = 0$ then $s^{t'}_{0,0} = s'_{0,0} < s_{0,0} \leq 1$ and hence $s^{t'}_{0,0} = 0$.
If $i_0 = 0$ and $j_0 > 0$ then $s^{t'}_{0,j_0} = s'_{0,j_0} < s_{0,j_0} \leq s_{0,j_0-1} = s'_{0,j_0-1} = s^{t'}_{0,j_0-1}$.
If $i_0 > 0$ and $j_0 = 0$ then $s^{t'}_{i_0,0} = s'_{i_0,0} < s_{i_0,0} \leq s_{i_0-1,0} = s'_{i_0-1,0} = s^{t'}_{i_0-1,0}$.
If $i_0 > 0$ and $j_0 > 0$ then $s^{t'}_{i_0,j_0} = s'_{i_0,j_0} < s_{i_0,j_0} \leq s_{i_0-1,j_0} + s_{i_0,j_0-1} = s'_{i_0-1,j_0} + s'_{i_0,j_0-1} = s^{t'}_{i_0-1,j_0} + s^{t'}_{i_0,j_0-1}$.
Now choose a full path q in t' such that $l(q) = i_0$ and $r(q) = j_0$ and attach two sons to it. For the binary tree t defined in such a way we have $s^t_{i_0,j_0} = s^{t'}_{i_0,j_0} + 1 = s'_{i_0,j_0} + 1 = s_{i_0,j_0}$ and $s^t_{i,j} = s^{t'}_{i,j} = s'_{i,j} = s_{i,j}$ for all other $(i,j) \in \{0,1,\ldots,d\}^2$. This completes the proof of the claim.

Consequently, we obtain that $e \in [\{a_1,\ldots,a_n\}]_p$ if and only if there exist a collection $\{s_{i,j} : i,j \in \{0,1,\ldots,d\}\} \subset \mathbb{N}$ with $d =_{\mathrm{df}} |\mathrm{bin}(e)| + 1$ that fulfills (**) and $\{r_{i,j,k} : i,j \in \{0,1,\ldots,d\}, k \in \{1,\ldots,n\}\} \subset \mathbb{N}$ such that $\sum_{k=1}^{n} r_{i,j,k} = r_{i,j}$ for $i,j \in \{0,1,\ldots,d\}$ (where the $r_{i,j}$ are computed from the $s_{i,j}$ as in (*)) and

$$e = \sum_{i=0}^{d}\sum_{j=0}^{d}\left(\sum_{k=1}^{n} r_{i,j,k} \cdot a_k\right) \cdot a^i \cdot b^j + \sum_{i=0}^{d}\sum_{j=0}^{d} s_{i,j} \cdot c \cdot a^i \cdot b^j.$$

This shows that $\mathrm{GEN}(\mathbb{N},p)$ is in NP. $\square$

That the upper bound given in Theorem 11 is tight, is shown by the fact that $\mathrm{GEN}(\mathbb{N},+)$ is $\leq^{\log}_{\mathrm{m}}$-complete for the addition $+$ on natural numbers (Theorem 9). However, we can also show that the generation problem for the multiplication on natural numbers is NP-complete. For this we need the following NP-complete problem (cf. [Sch78]) which we will reduce to our problem.

One-in-Three-Satisfiability 1-IN-3-SAT
INPUT: Formula F in CNF with exactly three literals per clause.
QUESTION: Is there an assignment to the variables of F such that in each clause exactly one literal becomes true.

Theorem 15. *The problem* $\mathrm{GEN}(\mathbb{N},\cdot)$ *is* $\leq^{\log}_{\mathrm{m}}$*-complete for* NP*, where* $\cdot$ *is the multiplication on* $\mathbb{N}$*.*

Proof: We will reduce 1-IN-3-SAT to $\mathrm{GEN}(\mathbb{N},\cdot)$. Let $p_1,p_2,p_3,\ldots$ be the prime numbers in natural ordering. Let H be an instance of 1-IN-3-SAT with the clauses $C_1,\ldots,C_m$ and the variables $x_1,\ldots,x_n$. We define $a_i =_{\mathrm{df}} p_{m+i} \cdot \Pi_{x_i \in C_j} p_j$, $b_i =_{\mathrm{df}} p_{m+i} \cdot \Pi_{\overline{x_i} \in C_j} p_j$, and $g(H) = (a_1,\ldots,a_n,b_1,\ldots,b_n,\Pi_{i=1}^{m+n} p_i)$. Note that g is polynomial time computable. If $H \in$ 1-IN-3-SAT then there is an assignment $I : \{x_1,\ldots,x_n\} \to \{0,1\}$ which satisfies in each clause of H exactly one literal. Then we obtain $\Pi_{I(x_i)=1} a_i \cdot \Pi_{I(x_i)=0} b_i = \Pi_{i=1}^{m+n} p_i$.

On the other hand, assume that $\Pi_{i=1}^{m+n} p_i \in [\{a_1, \ldots, a_n, b_1, \ldots, b_n\}]_.$. Then there are disjoint sets P and N such that $P \cup N = \{1, \ldots, n\}$ and $\Pi_{i=1}^{m+n} p_i = \Pi_{i \in P} a_i \cdot \Pi_{i \in N} b_i$. The assignment I such that $I(x_i) = 1$ if and only if $i \in P$ satisfies each clause of H exactly once. $\square$

6 Conclusion

We summarize our results on the complexity of GEN(Σ^*, f) in the following table. Every lower bounds is given by the fact that there exists an f from the considered class of operations whose generation problem is complete for the respective complexity class. All operations are polynomial time computable.

operation	lower bound	Theorem	upper bound	Theorem
arbitrary	recursively enumerable	2	recursively enumerable	1
length-monotonic	EXPTIME	5	EXPTIME	3
length-monotonic and commutative	EXPTIME	5	EXPTIME	3
length-monotonic and associative	PSPACE	7	PSPACE	6
length-mon., associative, and commutative	NP	9	NP	8
bivariate polynomials	NP	15	NP	11
univariate polynomials			P	10

We conclude with an interesting open problem: Is the generation problem for every (bivariate) polynomial either NP-complete or in P? If yes, is there a criterion which tells us wich of them is true?

References

[AKS02] M. Agrawal, N. Kayal, and N. Saxena. PRIMES in P. Technical report, Indian Institute of Technology Kanpur, 2002.

[CF91] J.-Y. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Science*, 2:67–76, 1991.

[CKS81] A.K. Chandra, D. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.

[Coo71] S.A. Cook. Characterizations of pushdown machines in terms of time bounded computers. *Journal of the ACM*, 18:4–18, 1971.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proccedings* 10^{th} *Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.

[vEB79] P. van Emde Boas. Complexity of linear problems. pages 117–120, 1979.