Generation Problems

Elmar Böhler, Christian Glaßer, Bernhard Schwarz, and Klaus Wagner *

Theoretische Informatik
Julius–Maximilians–Universität Würzburg
97074 Würzburg, Germany

14th April 2004

Abstract

Given a fixed computable binary operation f, we study the complexity of the following generation problem: The input consists of strings $a_1, \ldots, a_n, b$. The question is whether b is in the closure of $\{a_1, \ldots, a_n\}$ under operation f.

For several subclasses of operations we prove tight upper and lower bounds for the generation problems. For example, we prove exponential-time upper and lower bounds for generation problems of length-monotonic polynomial-time computable operations. Other bounds involve classes like NP and PSPACE.

Here the class of bivariate polynomials with positive coefficients turns out to be the most interesting class of operations. We show that many of the corresponding generation problems belong to NP. However, we do not know this for all of them, e.g., for $x^2 + 2y$ this is an open question. We prove NP-completeness for polynomials $x^a y^b c$ where $a, b, c \geq 1$. Also, we show NP-hardness for polynomials like $x^2 + 2y$. As a by-product we obtain NP-completeness of the extended sum-of-subset problem $\mathrm{SOS}_c = \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\}(\sum_{i \in I} w_i^c = z)\}$ for any $c \geq 1$.

1 Introduction

No, this paper is not about problems between generations.[1] However, genealogy presents an example that explains the matter we are interested in. There is hardly any other prehistoric question where scientists grope in the dark as in the following: Are Neanderthals completely extinct or are there traces of them left in some of us? To examine whether a person, e.g., one of the authors, is not a descendant of a Neanderthal, one would usually build the whole family tree of the author and check whether every leaf of the tree is labeled with a homo sapiens. This becomes a generation problem in the following way. We go back to the time where Neanderthals and homo sapiens still lived segregated from each other. It is well-known that it is the operation of marriage (in a very natural sense) that produces children. We start with this first generation of homo sapiens and apply this operation to obtain their children. Then we apply the marriage operation again and again, until we reach today's people. Now we see whether our author has been generated.

Similar generation problems are for example:

- Does b belong to the closure of $\{a_1, \ldots, a_n\}$ under pairwise addition? This is equivalent to a modification of the sum-of-subset problem where factors other than 0 and 1 are allowed. It can be shown that this is NP-complete [vEB79].

*EMail: [boehler,glasser,schwarzb,wagner]@informatik.uni-wuerzburg.de

[1] Regardless of the different ages of the authors.

- Does the empty clause belong to the closure of the clauses $\{\phi_1, \ldots, \phi_n\}$ under the rule of the resolution proof system. This problem is coNP-complete.
- Does a given element of a monoid belong to the submonoid that is generated by a given set?

In this paper we investigate sets that are generated by arbitrary computable binary operations. For a fixed such operation we study the complexity of the question:

Does a given string b belong to the set that is generated from strings $\{a_1, \ldots, a_n\}$?

To make this precise, let $\Sigma = \{0, 1\}$ be the alphabet and let f be a computable binary operation on Σ^*, i.e., $f : \Sigma^* \times \Sigma^* \to \Sigma^*$. For $B \subseteq \Sigma^*$, let $[B]_f$ be the f-closure of B, i.e., the smallest set that contains B and that is closed under f. For fixed f we define the generation problem.

> **Generation problem** $\mathrm{GEN}(f)$
> INPUT: $a_1, \ldots, a_n, b \in \Sigma^*$
> QUESTION: Is b in $[\{a_1, \ldots, a_n\}]_f$?

Equivalently we can use this definition in the context of natural numbers, since these can be identified in the standard way with Σ^*. For convenience we write operations like addition in infix form.

In Section 3, we observe that generation problems for computable operations are recursively enumerable, and there exist associative, commutative, polynomial-time computable operations whose generation problems are many-one complete for recursively enumerable sets. There remain undecidable problems even if we further restrict the operation's resources like time and space. However, we achieve decidability when we demand the operation to be length-monotonic which means that in the generation tree of some x, the lengths of all intermediate results are bounded by the length of x. If the operations are length-monotonic and polynomial-time computable, all generation problems are solvable in exponential time and there are also such operations for which the generation problem is hard for $\mathrm{EXPTIME}$. We study the complexity of various restrictions of these operations. If additionally the operation is associative, then the corresponding generation problem belongs to PSPACE, and is even PSPACE-complete for suitable operations. If we further restrict the operations to be commutative, then we obtain generation problems that belong to NP, and some of them are even NP-complete (e.g., the usual integer addition).

The most interesting operations we consider in this paper are bivariate polynomials with positive coefficients which are studied in Section 4. Such polynomials are length-monotonic and hence, the corresponding generation problems are decidable. However, in general these polynomials are neither associative nor commutative, and hence the generation problems for such polynomials turn out to be non-trivial and exciting. For example, does $\mathrm{GEN}(x^2 + 2y)$ or $\mathrm{GEN}(x^2y^3)$ belong to NP? If so, are they NP-complete?

There are two main results in this section: For one, we show that if p is not of the form $q(x) + ky$ where q is non-linear and $k \geq 2$, then the generation problem belongs to NP. Besides that, we present a proof of NP-completeness for polynomials of the form $x^a y^b c$ where $a, b, c \geq 1$. Proving hardness is difficult already for such simple polynomials, since we have to cope with the various different trees that generate one number. As a tool to control the shape of generation trees we introduce (a, b)-*weighted trees* which are special trees with additional information. In the proof we force the generation trees into the shape of so-called complete (a, b)-weighted trees.

We do not know whether the generation problem belongs to NP, if the generating polynomial is of the form $q(x) + ky$ where q is non-linear and $k \geq 2$. In this regard, as an upper bound we can easily show that all bivariate polynomials with positive coefficients have generation problems in $\mathrm{NTIME\text{-}SPACE}(2^{\log^2 n}, n \log n)$. Our discussion in Section 5 suggests that this class appears to be a class not far from NP. As a special case of these polynomials, we consider $p(x, y) = x^c + ky$ where $c, k \geq 1$.

The main result of Section 5 shows that $\mathrm{GEN}(p)$ is NP-hard. Here the operation x^c brings the main difficulty for the proof. We have to find a way to encode information to numbers such that this information is not destroyed by taking the numbers to a high power. This is not easy to solve, since already squaring a number heavily changes its (binary) representation. To overcome this, for $c \geq 1$ we introduce generalized sum-of-subset problems

$$\mathrm{SOS}_c \overset{df}{=} \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\}(\textstyle\sum_{i \in I} w_i^c = z)\}.$$

We show that for all $c \geq 1$, SOS_c is NP-complete and then reduce these problems to $\mathrm{GEN}(p)$. Although all SOS_c are just auxiliary problems in our proof, we feel that this new NP-completeness result is interesting in its own right.

Finally, in Section 6 we summarize our results and give a table that shows a convenient overview of the upper and lower bounds of generation problems.

2 Preliminaries

For $a \geq 0$ let $\mathrm{bin}(a)$ be a's binary representation (without leading zeros, if $a > 0$). For convenience we use the operation mod in two ways: In $a \equiv b(\mathrm{mod}\ m)$ (or $a \equiv b(m)$ for short) it is used in the usual way, while the expression $(n \bmod m)$ denotes the remainder of n divided by m.

We work with pairs (A, B) of disjoint languages (where for example $A \in \mathrm{NP}$ and $B \in \mathrm{coNP}$). Say that pair (A, B) reduces to pair (C, D), in notation $(A, B) \leq_m^{pp} (C, D)$, if there exist a polynomial-time computable function f such that for all x,

$$\begin{aligned} x \in A \quad &\Rightarrow \quad f(x) \in C, \\ x \in B \quad &\Rightarrow \quad f(x) \in D. \end{aligned}$$

$A \leq_m^{pp} (C, D)$ and $(A, B) \leq_m^{pp} C$ are abbreviations for $(A, \overline{A}) \leq_m^{pp} (C, D)$ and $(A, B) \leq_m^{pp} (C, \overline{C})$.

A finite tree is called *binary tree*, if every node is either a leaf or has exactly two successors. Let $\mathrm{L}(T)$ be the set of leaves, $\mathrm{root}(T)$ be the root and $\mathrm{Node}(T)$ be the set of nodes of a tree T. We characterize a path from the root to a node by a word $w \in \{l, r\}^*$ where l defines a left turn and r defines a right turn. Let $\mathrm{path}(T) \overset{df}{=} \{w : w \text{ is a path of } T\}$. Every $v \in \mathrm{path}(T)$ that does not lead to a leaf node is called *initial path* of T. In contrast, every path in $\mathrm{path}(T)$ that is not an initial path is a *full path*. Let $\mathrm{ipath}(T)$ be the set of initial paths of T and $\mathrm{fpath}(T)$ be the set of full paths in T. For $q \in \mathrm{path}(T)$, let $l(q)$ and $r(q)$ be the number of left steps and right steps, resp., in q. For a node x of T with path v, let $l(x) \overset{df}{=} l(v)$ (resp., $r(x) \overset{df}{=} r(v)$).

The process of generating elements by an iterated application of a binary operation can be visualized by a *generation tree*.

Let $B \subseteq \Sigma^*$ be the base set. If f is a binary operation, then a binary tree is called *f-generation tree from B for z* if

- every leaf has a value from B,
- every node that has successors with values x and y has value $f(x, y)$,
- the root of the tree has value z.

Note that $z \in [B]_f$, if and only if there exists an f-generation tree from B for z.

3 Generation Problems for General Operations

Since we are mostly interested in complexity issues, we restrict ourselves to computable operations. All of the corresponding generation problems are recursively enumerable and we show that there are polynomial-time computable operations whose generation problems are undecidable. There remain undecidable problems even if we further restrict the operation's resources like time and space. The reason is that even with restricted resources it is possible to let a generation problem simulate grammatical derivation trees of arbitrary formal languages. We achieve decidability when we demand the operation to be length-monotonic. Hence we study the complexity of various restrictions of length-monotonic operations.

Theorem 3.1 $\mathrm{GEN}(\circ)$ *is recursively enumerable for every computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

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 $\mathrm{GEN}(\circ)$. □

We observe that polynomial-time computable operations are still too difficult for a complexity-oriented examination of generation problems. For example, with such an operation we can simulate single steps of arbitrary Turing machines.

Theorem 3.2 *There is an associative, commutative, polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $\mathrm{GEN}(\circ)$ *is m-complete for recursively enumerable sets.*

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

$$0^{n+1}1^{m_1} \circ 0^{n+1}1^{m_2} \stackrel{df}{=} \begin{cases} 0^{n+1}1^{m_1+m_2}, & \text{if } M \text{ on } n \text{ does not stop within } m_1+m_2 \text{ steps} \\ 1, & \text{otherwise,} \end{cases}$$

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

Observe, that $\circ$ is commutative and $\circ \in \mathrm{FP}$. For associativity let $x, y, z \in \Sigma^*$. In case that there are $n, m_1, m_2, m_3 \geq 0$ such that $x = 0^{n+1}1^{m_1}$, $y = 0^{n+1}1^{m_2}$, $z = 0^{n+1}1^{m_3}$ and M on n does not stop within $m_1 + m_2 + m_3$ 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 on n stops within m steps, then $[\{0^{n+1}1^1\}]_\circ = \{0^{n+1}1^1, 0^{n+1}1^2, \ldots, 0^{n+1}1^{m-1}, 1\}$. If M on n does not stop, then $[\{0^{n+1}1^1\}]_\circ = \{0^{n+1}1^1, 0^{n+1}1^2, \ldots\}$. Hence,

$$n \in D_\varphi \Leftrightarrow M \text{ on } n \text{ stops } \Leftrightarrow 1 \in [\{0^{n+1}1^1\}]_\circ \Leftrightarrow (0^{n+1}1, 1) \in \mathrm{GEN}(\circ).$$

□

3.1 Length-Monotonic Polynomial-Time Operations

We have seen that in order to get decidable generation problems we have to restrict the class of operations. Therefore, we demand that in the generation tree of some x, the lengths of all intermediate results are bounded by $|x|$, the *length* of x. This is equivalent to say that we restrict to operations $\circ$ that satisfy $|x \circ y| \geq \max(|x|, |y|)$. Call such operations *length-monotonic*. If $|x \circ y| = \max(|x|, |y|)$, then the operation is called *minimal length-monotonic*. Generation trees of such operations can be exhaustively searched by an alternating polynomial-space machine.

Theorem 3.3 $\mathrm{GEN}(\circ) \in \mathrm{EXPTIME}$ *for every length-monotonic, polynomial-space computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

Proof Let $\circ$ be a length-monotonic, polynomial-space computable operation. $\mathrm{GEN}(\circ)$ 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
```

Every polynomial-space-bounded alternating algorithm can be simulated by a deterministic exponential-time-bounded algorithm [CKS81]. So $\mathrm{GEN}(\circ) \in \mathrm{EXPTIME}$. □

This exponential-time upper bound for length-monotonic, polynomial-space computable operations is tight, even for polynomial-time computable operations. To see this we start with a technical lemma which simplifies the argumentation. It shows that for certain sets A, we can translate operations $* : A \times A \to A$ to operations $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that the complexity of the generation problem and other properties are preserved. This is done by an appropriate encoding of elements from A.

Lemma 3.4 *Let* $A_1, \ldots, A_{k+l}$ *be finite sets,* $A \stackrel{df}{=} A_1^* \times \ldots \times A_k^* \times A_{k+1} \times \ldots \times A_{k+l}$, *and let* $* : A \times A \to A$ *be a polynomial-time computable operation. Then there exists a polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that:*

1. $\mathrm{GEN}(*) \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\circ)$.
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 $d(0) \stackrel{df}{=} 00$ and $d(1) \stackrel{df}{=} 11$, and let $\mathrm{code} : A \to \Sigma^*$ be an encoding given by $\mathrm{code}(x_1, x_2, \ldots x_{k+l}) \stackrel{df}{=} d(h_1(x_1))01d(h_2(x_2))01 \ldots 01d(h_{k+l}(x_{k+l}))$. Note that $|\mathrm{code}(u)| = 2m|u| + 2(k+l-1)$. For $w_1, w_2 \in \Sigma^*$, $\circ$ can be defined as

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

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 conclude:

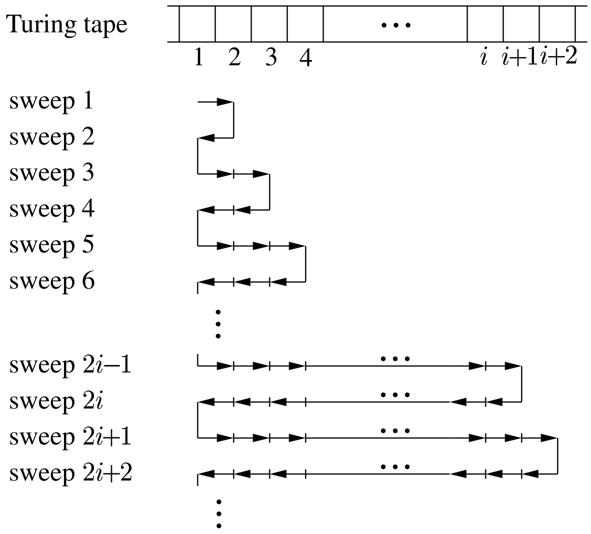
$$\begin{aligned} |w_1 \circ w_2| &= |\mathrm{code}(u_1 * u_2)| = |\mathrm{code}(v)| = 2m|v| + 2(k+l-1) \\ &= 2m \cdot \max(|u_1|, |u_2|) + 2(k+l-1) \\ &= \max(2m|u_1| + 2(k+l-1), 2m|u_2| + 2(k+l-1)) \\ &= \max(|\mathrm{code}(u_1)|, |\mathrm{code}(u_2)|) = \max(|w_1|, |w_2|). \end{aligned}$$

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 definition of $\circ$, $v \in [\{u_1, \dots, u_m\}]_*$ if and only if $\mathrm{code}(v) \in [\{\mathrm{code}(u_1), \dots, \mathrm{code}(u_m)\}]_\circ$. This completes the proof. $\square$

Theorem 3.5 *There is a commutative, minimal length-monotonic, polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $\mathrm{GEN}(\circ)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathrm{EXPTIME}$.

Proof We follow the idea of Cook [Coo71] to simulate deterministic exponential-time computations. Without loss of generality, a deterministic exponential-time one-tape Turing machine M deciding a set $A \subseteq \Sigma^*$ can be normalized in such a way that on input $x = a_1 a_2 \dots a_n$ it makes $2^{p(|x|)}$ *sweeps*, where p is a suitable polynomial. For $0 \leq 2i < 2^{p(|x|)}$, the $(2i+1)$-st 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 $(2i+2)$-nd 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 of states S, 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, then 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 i-th 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 $k \in \{j-1, j+1\}$. The idea of our operation is as follows: multiply $(x, i-1, j, s', a')$ with (x, i, k, s'', a'') and obtain (x, i, j, s, a). In an accepting computation of M on x (and only in this case) one generates finally the correct $(x, 2^{p(|x|)}, 2, s_1, \square)$.

To make this precise, let $a_j \stackrel{df}{=} \square$ for all $j > n$. Furthermore we assume that, in a quintuple (x, i, j, s, a) where $i, j \in \{0, 1, \dots, 2^{p(|x|)}\}$, the numbers i and j are given in binary presentation of length exactly $p(|x|) + 1$. Now define the operation $*$ as follows.

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

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

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

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

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

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

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

$(x, 2i, 1, s, a) * (x, 0, 0, s_0, \square) \stackrel{df}{=} (x, 2i+1, 1, s, a)$

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

$(x, 2i+1, i+2, s, a) * (x, 0, 0, s_0, \square) \stackrel{df}{=} (x, 2i+2, i+2, s, a)$

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

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 $(x, 0, 0, s_0, \square)$ one can generate exactly the correct quintuples of the form $(x, \dots)$ together with $(x, 0, 0, s_0, \square)$ and $(0^{|x|+p(|x|)+2}, \varepsilon, \varepsilon, s_0, \square)$. Hence, M accepts x if and only if $((x, 1, 1, \sigma(s_0, a_1), \lambda(s_0, a_1)), (x, 0, 0, s_0, \square), (x, 2^{p(|x|)}, 2, s_1, \square)) \in \mathrm{GEN}(*)$, and consequently $A \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(*)$. By Lemma 3.4, we finally obtain a polynomial-time computable, minimal length-monotonic and commutative operation $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that $A \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(\circ)$. $\square$

3.2 Length-Monotonic Associative Polynomial-Time Operations

We have seen that in general, commutativity does not lower the complexity of the generation problem for length-monotonic, polynomial-time computable operations. In this subsection we show that associativity does. Here we exploit that for associative operations $\circ$ we do not need to know the exact structure of an $\circ$-generation tree for z: Associativity makes all generation trees with the same sequence of leaves equivalent with respect to the generated element. We show that PSPACE is upper bound for all generation problems with associative, polynomial-space computable operations and that it is lower bound even for associative, polynomial-time computable operations.

Theorem 3.6 $\mathrm{GEN}(\circ) \in \mathrm{PSPACE}$ *for every length-monotonic and associative polynomial-space computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

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

```
function GEN(z,x1,...,xn);
  choose an i ∈ {1,...,n} nondeterministically;
  z1 := xi;
  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
```

$\square$

The polynomial-space bound is tight even for polynomial-time operations $\circ$.

Theorem 3.7 *There is a minimal length-monotonic and associative polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $\mathrm{GEN}(\circ)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* PSPACE.

Proof Let $L \subseteq \Sigma^*$ be a set which is $\leq_{\mathrm{m}}^{\log}$-complete for PSPACE. By Lemma 3.4, it suffices to prove existence of a finite alphabet Δ and a minimal length-monotonic and associative polynomial-time computable operation $* : (\Sigma^* \times \Delta^*) \times (\Sigma^* \times \Delta^*) \to (\Sigma^* \times \Delta^*)$ such that $L \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(*)$.

Since $L \in \mathrm{PSPACE}$, it follows [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 $x \in \Sigma^*$,

$$x \in L \leftrightarrow f(x, 0) \cdot f(x, 1) \cdot \dots \cdot f(x, 2^{p(|x|)} - 1) = a_0$$

where $(\mathrm{A}_5,\cdot)$ is the group of even permutations on five elements with identity permutation a_0. Fix an $x \in \Sigma^*$ and let $K \stackrel{df}{=} p(|x|)$ and $M \stackrel{df}{=} 4K+3$. Further, for $i = 0, 1, \ldots, 2^K - 1$, let $\mathrm{b}(i)$ be that length K binary representation of i.

We consider the set

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

with multiplication $*$ whose essential part is given by

$$(x, \mathrm{b}(i)\, a\, \mathrm{b}(j)) * (x, \mathrm{b}(j{+}1)\, b\, \mathrm{b}(m)) \stackrel{df}{=} (x, \mathrm{b}(i)\, a \cdot f(x,j) \cdot b\, \mathrm{b}(m)).$$

We obtain

$$x \in L \leftrightarrow (x, 0^K a_0 1^K) \in [\{(x, \mathrm{b}(i) f(x,i) \mathrm{b}(i{+}1)) : 0 \leq i < 2^K{-}1\}]_*.$$

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

$$E_x \stackrel{df}{=} \{(x, \langle u \rangle) : u \in \Sigma^{\leq K} \cup \Sigma^{\leq K} \# \Sigma^{\leq K} \cup \Sigma^{\leq K} \# \Sigma^K \# \Sigma^{\leq K} \cup \Sigma^{\leq K} \# \Sigma^K \cdot \mathrm{A}_5 \cdot \Sigma^K \# \Sigma^{\leq K}\}$$

The operation $* : E_x \times E_x \to E_x$ is defined as follows. $(x, w_1) * \cdots * (x, w_r) \stackrel{df}{=}$

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

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 $f(x,i) \cdot f(x, i+1) \cdot \cdots \cdot f(x, j-1) = a$. 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. Without loss of generality assume that $\varepsilon \notin L$. If there is no $x \in \Sigma^*$ such that $(y,v), (z,w) \in E_x$, then we define $(y,v) * (z,w) \stackrel{df}{=} (\varepsilon, 2^{\max(|y|+|z|, |z|+|w|)})$. 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 $L \leq_{\mathrm{m}}^{\log} \mathrm{GEN}(*)$. $\square$

Now let us additionally assume $\circ$ to be commutative. Again, if we want to know whether or not $z \in [\{x_1, \ldots, x_n\}]_\circ$, associativity enables us to ignore the $\circ$-generation tree and instead search for a word over $\{x_1, \ldots, x_n\}$. Together with commutativity, we just have to guess exponents $k_1, \ldots, k_n$ and test whether $x_1^{k_1} \circ \cdots \circ x_n^{k_n} = z$. If the operation is computable in polynomial-time, then the exponentiations are computable in polynomial-time, too (by squaring and multiplying), which yields the following theorem.

Theorem 3.8 $\mathrm{GEN}(\circ) \in \mathrm{NP}$ *for all length-monotonic, associative, and commutative polynomial-time computable operations* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

Again, this upper bound is tight, i.e., there exist associative, commutative, and length-monotonic polynomial-time computable operations whose generation problems are NP-complete. Even the usual addition on natural numbers has this property.

Theorem 3.9 $\mathrm{GEN}(+)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP, *where* $+$ *is the addition on* $\mathbb{N}$.

Proof It is known that $\mathrm{GEN}(+)$ is NP-complete for the addition on integers [vEB79]. This proof exclusively uses natural numbers. □

4 Generation Problems for Polynomials

The previous section gave an overview over the complexity of generation problems for polynomial-time computable operations. Now we want to have a look at the more restricted class of generation problems whose operations are polynomials. The Davis-Putnam-Robinson-Matiyasevich theorem [Mat70] states that every recursively enumerable set is range of a polynomial with integer coefficients. Based on this, there are such polynomials where the generation problem is undecidable. To give an idea of this, take a polynomial p with undecidable positive range and replace every variable x by $x_1^2+x_2^2+x_3^2+x_4^2$. Take another polynomial q that is capable to generate all negative numbers and negative numbers only. Build a new polynomial out of p and q with an additional variable y such that for $y = 0$ the value of q is calculated, and for $y \neq 0$ the value of p is calculated. In this way it is possible to generate all negative numbers which in turn allow the generation of the positive range of p. However, to obtain this undecidability result, the polynomials must have negative coefficients and they usually contain a rather large number of variables. Therefore, we concentrate on bivariate polynomials with positive coefficients. These are always length-monotonic and hence, the corresponding generation problem is decidable. We show that many of them are even in NP and all of them belong to $\mathrm{NTIME\text{-}SPACE}(2^{\log^2 n}, n \log n)$. So far we have no evidence against the conjecture that all these generation problems belong to NP (see also the discussion in Section 5). However, we cannot prove this.

This section has two main results: First, we show that if p is not of the form $q(y) + ky$ where q is non-linear and $k \geq 2$, then the corresponding generation problem belongs to NP. Second, we prove NP-completeness for polynomials of the form $x^a y^b c$ where $a, b, c \geq 1$.

4.1 The Main Case

Let us start our investigation with univariate polynomials p, i.e., $p(x, y) = q(x)$ for a suitable polynomial q.

Theorem 4.1 *If* p *is a univariate polynomial, then* $\mathrm{GEN}(p)$ *is in* P.

Proof If $p(x, y) = q(x) = c$, then we have $[\{a_1, \ldots, a_n\}]_p = \{a_1, \ldots, a_n, c\}$. If $p(x, y) = q(x) = x + c$, then $[\{a_1, \ldots, a_n\}]_p = \{a_i + kc : i = 1, \ldots, n, k \geq 0\}$. In all other cases we have $q(x) \geq 2x$ or $q(x) \geq x^2$. It follows that $e \in [\{a_1, \ldots, a_n\}]_p \Leftrightarrow e \in \{p_k(a_i) : i = 1, \ldots, n, k = 0, 1, \ldots, |\mathrm{bin}(e)| + 1\}$ where $p_0(x) \stackrel{df}{=} x$ and $p_{k+1}(x) \stackrel{df}{=} p(p_k(x))$ for $k \geq 0$.
So in all cases the membership to $[\{a_1, \ldots, a_n\}]_p$ can be easily verified in polynomial time. □

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

Theorem 4.2 *If p is a bivariate polynomial that is not of the form $p(x, y) = kx+q(y)$ or $p(x, y) = q(x)+ky$ where q is non-linear and $k \geq 2$, then* $\mathrm{GEN}(p) \in \mathrm{NP}$.

Proof We show that p must have one of the following properties:

(1) $p(x, y) = x + q(y)$ or $p(x, y) = q(x) + y$ for some univariate polynomial q,

(2) $p(x, y) = ax + by + c$ for some $a, b, c \in \mathbb{N}$ such that $a, b \geq 2$, and

(3) $p(x, y) \geq x \cdot y$ for all x, y.

After this, the proof of the theorem is completed by the following three lemmas.

Assume that the polynomial p has non of the properties (1), (2), and (3). Since p does not fulfill (3) there are univariate polynomials q and r such that $p(x, y) = q(x) + r(y)$. Since $x^2 + y^2 \geq x \cdot y$ at least one of the polynomials q and r is linear. Consequently there exist a univariate polynomial q and an $k \geq 0$ such that $p(x, y) = kx + q(y)$ or $p(x, y) = q(x) + ky$. Since p does not fulfill (2), the polynomial q is not linear. Since p does not fulfill (1), we obtain $k \geq 2$. □

Lemma 4.3 *If $p(x, y) = x + q(y)$ for some univariate polynomial q, then* $\mathrm{GEN}(p) \in \mathrm{NP}$.

Proof It is sufficient to prove:

$$[\{a_1, \dots, a_r\}]_p = \{a_j + \textstyle\sum_{i=1}^{r} \alpha_i \cdot q(a_i) : j \in \{1, \dots, r\} \text{ and } \alpha_1, \dots, \alpha_r \in \mathbb{N}\}.$$

The inclusion from right to left is obvious. For the other direction, we observe that $\{a_1, \dots, a_r\}$ 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 $j, k \in \{1, \dots, r\}$. Then for some $c \geq 0$,

$$\begin{aligned} p(a_j + \sum_{i=1}^{r}(\alpha_i \cdot q(a_i)), a_k + \sum_{i=1}^{r}(\beta_i \cdot q(a_i))) &= a_j + \sum_{i=1}^{r}(\alpha_i \cdot q(a_i)) + q(a_k + \sum_{i=1}^{r}(\beta_i \cdot q(a_i))) \\ &= a_j + \sum_{i=1}^{r}(\alpha_i \cdot q(a_i)) + q(a_k) + c\sum_{i=1}^{r}(\beta_i \cdot q(a_i)) \quad (1) \\ &= a_j + \sum_{i=1}^{r}((\alpha_i + c\beta_i) \cdot q(a_i)) + q(a_k). \end{aligned}$$

To see equality (1), observe that by binomial theorem, for all $a, b \geq 0$, $q(a, b) = q(a) + cb$ for some $c \in \mathbb{N}$. □

Lemma 4.4 *If $p(x, y) = ax + by + c$ for $a, b, c \in \mathbb{N}$ and $a, b \geq 2$, then* $\mathrm{GEN}(p) \in \mathrm{NP}$.

Proof Let T be a p-generation tree for e. Without loss of generality we can assume that value 0 occurs only in the leaves of this tree T. Since $a, b \geq 2$, the depth of T is bounded by $|\mathrm{bin}(e)| + 1$.

Let T be an arbitrary binary tree whose leaves have values from $\{a_1, \dots, a_n\}$. For a full path q in T, choose $i(q) \in \{1, \dots, n\}$ such that the leaf of q has value $a_{i(q)}$. We obtain that $e \in [\{a_1, \dots, a_n\}]_p$ if and only if there exists a binary tree T whose leaves have values from $\{a_1, \dots, a_n\}$ such that

$$e = \sum_{q \,\in\, \mathrm{fpath}(T)} a_{i(q)} \cdot a^{l(q)} \cdot b^{r(q)} + \sum_{q \,\in\, \mathrm{ipath}(T)} c \cdot a^{l(q)} \cdot b^{r(q)}.$$

For a binary tree T of depth bounded by d and for $i, j \in \{0, \ldots, d\}$ we define the characteristics

$$\begin{aligned} s^T_{i,j} &\stackrel{\text{df}}{=} \#\{q : q \in \mathrm{ipath}(T), l(q) = i \text{ and } r(q) = j\} \text{ and} \\ r^T_{i,j} &\stackrel{\text{df}}{=} \#\{q : q \in \mathrm{fpath}(T), l(q) = i \text{ and } r(q) = j\}. \end{aligned}$$

Note that the $r^T_{i,j}$ can be computed from the $s^T_{i,j}$ by

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

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 $d \le |\mathrm{bin}(e)| + 1$ and a set of natural numbers $\{r_{i,j,k} : i, j \in \{0, \ldots, d\}, k \in \{1, \ldots, n\}\}$ 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 $s^T_{i,j}$ have the following properties.

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

On the other hand, we can prove the following.

Claim. Consider arbitrary natural numbers $s_{i,j}$ where $i, j \in \{0, \ldots, d\}$. If these $s_{i,j}$ fulfill $(**)$, then there exists a binary tree T such that $s^T_{i,j} = s_{i,j}$ for $i, j \in \{0, \ldots, d\}$.

Proof of the claim. By induction on $w(M) \stackrel{\text{df}}{=} \sum_{i=0}^d \sum_{j=0}^d s_{i,j}$.
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 $i, j \in \{0, \ldots, d\}$ there exists a pair $(i, j) \in \{0, \ldots, d\}^2$ 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 $M' \stackrel{\text{df}}{=} \{s'_{i,j} : i, j \in \{0, \ldots, d\}\}$ such that $s'_{i_0,j_0} \stackrel{\text{df}}{=} s_{i_0,j_0} - 1$ and $s'_{i,j} = s_{i,j}$ for all other $(i, j) \in \{0, \ldots, d\}^2$. 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 $i, j \in \{0, \ldots, d\}$. 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 this by considering four cases.

If $i_0 = j_0 = 0$ then $s^{T'}_{0,0} = s'_{0,0} < s_{0,0} \le 1$ and hence $s^{T'}_{0,0} = 0$.
If $i_0 = 0$ and $j_0 > 0$ then $s^{T'}_{0,j_0} = s'_{0,j_0} < s_{0,j_0} \le s_{0,j_0-1} = s'_{0,j_0-1} = s^{T'}_{0,j_0-1}$.
If $i_0 > 0$ and $j_0 = 0$ then $s^{T'}_{i_0,0} = s'_{i_0,0} < s_{i_0,0} \le s_{i_0-1,0} = s'_{i_0-1,0} = s^{T'}_{i_0-1,0}$.
If $i_0 > 0$ and $j_0 > 0$ then $s^{T'}_{i_0,j_0} = s'_{i_0,j_0} < s_{i_0,j_0} \le s_{i_0-1,j_0} + s_{i_0,j_0-1} = s'_{i_0-1,j_0} + s'_{i_0,j_0-1} = s^{t'}_{i_0-1,j_0} + s^{T'}_{i_0,j_0-1}$.

Now choose a full path q in T' such that $l(q) = i_0$ and $r(q) = j_0$ and attach two successors 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 $(i, j) \in \{0, \ldots, d\}^2$. This completes the proof of the claim.

Consequently, we obtain that $e \in [\{a_1, \dots, a_n\}]_p$ if and only if for $d \stackrel{df}{=} |\mathrm{bin}(e)|+1$ and $i, j \in \{0, \dots, d\}$ there exist natural numbers $s_{i,j}$ and there exists a set of natural numbers $\{r_{i,j,k} : i, j \in \{0, \dots, d\}, k \in \{1, \dots, n\}\}$ such that

1. the $s_{i,j}$ fulfill $(**)$,
2. $\sum_{k=1}^{n} r_{i,j,k} = r_{i,j}$ for $i, j \in \{0, \dots, d\}$ (where the $r_{i,j}$ are computed from the $s_{i,j}$ as in $(*)$), and
3. $e = \sum_{i=0}^{d} \sum_{j=0}^{d} (\sum_{k=1}^{n} r_{i,j,k} \cdot a_k) \cdot a^i \cdot b^j + \sum_{i=0}^{d} \sum_{j=0}^{d} s_{i,j} \cdot c \cdot a^i \cdot b^j$.

This shows $\mathrm{GEN}(p) \in \mathrm{NP}$. □

Lemma 4.5 *If the polynomial p fulfills $p(x, y) \geq x \cdot y$ for all x, y, then* $\mathrm{GEN}(p) \in \mathrm{NP}$.

Proof Let $A \subseteq \mathbb{N}$ be finite. Let $A' \stackrel{df}{=} A \cup \{p(c, c) : c \in \{0\} \cap A\}$. Obviously we have $[A]_p = [A']_p$ and for every $z \in [A']_p$ there is a p-generation tree that has no node v that has only child nodes with value 0. If for every $x \in \mathbb{N}$ (resp., $y \in \mathbb{N}$), $(*)$

- $p(x, 0) \geq 2x$ (resp., $p(0, y) \geq 2y$) or
- $p(x, 0) \geq x^2$ (resp., $p(0, y) \geq y^2$) or
- $p(x, 1) \geq 2x$ (resp., $p(1, y) \geq 2y$) or
- $p(x, 1) \geq x^2$ (resp., $p(1, y) \geq y^2$),

then there is a p-generation tree for z from A' such that there are at most $|z|$ nodes with left (resp., right) child that has a value ≥ 2 $(*)$. Let D be a p-generation tree from A' for z. We can assume that there are at most $|z|$ leaves v in D that have a value greater than 1 and there can at most be $|z|$ nodes having two children with values greater than 1. Furthermore, we can assume that there are at most $|z|$ nodes v in D such that both children of v are leaves with values from $\{0, 1\}$. That means that if D has exponentially many nodes, then nearly every node (except polynomially many ones) $(**)$

- has one child with value ≤ 1 and another one that is no leaf, or
- is a leaf with value ≤ 1, and its parent's other child is no leaf.

We consider four cases:

- Let there be $x_1, \dots, x_8 \in \mathbb{N}$ such that ($p(x_1, 0) \not\geq 2x_1$ and $p(x_2, 0) \not\geq x_2^2$ and $p(x_3, 1) \not\geq 2x_3$ and $p(x_4, 1) \not\geq x_4^2$) and ($p(0, x_5) \not\geq 2x_5$ and $p(0, x_6) \not\geq x_6^2$ and $p(1, x_7) \not\geq 2x_7$ and $p(1, x_8) \not\geq x_8^2$). Then $p(x, y) = xy + c$, where $c \in \mathbb{N}$. Note that $p(x, 0) = p(0, y) = c$. Since $c \in A'$ if $0 \in A'$, we can assume that there are no leaves with value 0. Furthermore, $q(x) \stackrel{df}{=} x + c = p(1, x) = p(x, 1)$ for all $x \in \mathbb{N}$. Note that
$$\underbrace{q(q(\dots q}_{k}(x)\dots)) = x + kc,$$
so k applications of q can be guessed in one step. Using property $(**)$, we can guess a polynomially sized generation tree, where each node either represents a normal generation step or $k \leq z$ steps of the above form.

- Let there be $x_1, \dots, x_4 \in \mathbb{N}$ such that for all $x \in \mathbb{N}$ we have ($p(x,0) \geq 2x$ or $p(x,0) \geq x^2$ or $p(x,1) \geq 2x$ or $p(x,1) \geq x^2$) and ($p(0,x_1) \not\geq 2x_1$ and $p(0,x_2) \not\geq x_2^2$ and $p(1,x_3) \not\geq 2x_3$ and $p(1,x_4) \not\geq x_4^2$). Then $p(x,y) = x^k y + \sum_{i=1}^n b_i x^i + d$ where $k \geq 1, n, b_i, d \in \mathbb{N}$ $(1 \leq i \leq n)$. Because of $(*)$ there can only be polynomially many nodes in D with a left child that has a value greater than 1. So if there are exponentially many nodes in D, then all of them except polynomially many ones have a left child with value ≤ 1 and a right child that is not a leaf. Observe that $p(0,y) = d$ for all y, so we can assume that there is no left child labeled with 0 since if $0 \in A'$, so is d. Furthermore, $p(1,y) = y + \sum_{i=1}^n b_i + d$ and

$$\underbrace{p(1,p(1,\dots p}_{k}(1,y)\dots)) = y + k(\textstyle\sum_{i=1}^n b_i + d).$$

  Therefore we can guess a polynomial-sized generation tree for z where each node is either a normal generation step or $k \leq z$ subsumed steps of the form $p(1,y)$.

- Let there be $x_1, \dots, x_4 \in \mathbb{N}$ such that for all $x \in \mathbb{N}$ we have ($p(x_1,0) \not\geq 2x_1$ and $p(x_2,0) \not\geq x_2^2$ and $p(x_3,1) \not\geq 2x_3$ and $p(x_4,1) \not\geq x^2$) and ($p(0,x) \geq 2x$ or $p(0,x) \geq x^2$ or $p(1,x) \geq 2x$ or $p(1,x) \geq x^2$). Here a symmetrical argumentation holds.

- Let for all $x \in \mathbb{N}$ hold ($p(x,0) \geq 2x$ or $p(x,0) \geq x^2$ or $p(x,1) \geq 2x$ or $p(x,1) \geq x^2$) and ($p(0,x) \geq 2x$ or $p(0,x) \geq x^2$ or $p(1,x) \geq 2x$ or $p(1,x) \geq x^2$). By $(*)$ there is a polynomial sized p-generation tree from A' for b that can be guessed and checked in P.

□

4.2 GEN($x^a y^b c$) is NP-complete

By Theorem 4.2, if we consider a polynomial of the form $\sum_{a,b} x^a y^b c_{ab}$ where $a, b \geq 1$, then the generation problem belongs to NP. Here we pick out those polynomials that consist of only one term of the sum. For this special case we can show that $\mathrm{GEN}(x^a y^b c)$ is NP-complete if $c \geq 1$. For $a = 1$ or $b = 1$ this is easy to prove.

Proposition 4.6 *For $a, c \geq 1$, $\mathrm{GEN}(x^a y c)$ is $\leq_m^p$-complete for* NP.

Proof We reduce 1-IN-3-SAT to GEN(p), where $p(x,y) \stackrel{df}{=} x^a y c$. Let H be a 3-CNF formula with clauses $C_1, \dots, C_m$ and variables $x_1, \dots, x_n$. Let $p_1, p_2, \dots$ be the prime numbers larger than c. Define $a_1 \stackrel{df}{=} p_{m+1}^a \prod_{x_1 \in C_j} p_j^a$, $b_1 \stackrel{df}{=} p_{m+1}^a \prod_{\overline{x_1} \in C_j} p_j^a$, for $2 \leq i \leq n$, $a_i \stackrel{df}{=} p_{m+i} \prod_{x_i \in C_j} p_j$, $b_i \stackrel{df}{=} p_{m+i} \prod_{\overline{x_i} \in C_j} p_j$, $z \stackrel{df}{=} c^{n-1} \cdot \prod_{i=1}^{m+n} p_i^a$, and $g(H) \stackrel{df}{=} (a_1, \dots, a_n, b_1, \dots, b_n, z)$. Note that g is polynomial-time computable.

Assume $H \in$ 1-IN-3-SAT. Then there is an assignment $I : \{x_1, \dots, x_n\} \to \{0,1\}$ that satisfies exactly one literal in each clause. Therefore, we obtain $\prod_{i=1}^{m+n} p_i^a \cdot c^{n-1}$ by a linear generation tree that has leaf-values $c_n, \dots, c_1$ where $c_i = a_i$ if $I(x_i) = 1$ and $c_i = b_i$ otherwise. Hence $g(H) \in \mathrm{GEN}(p)$.

Assume that $g(H) \in \mathrm{GEN}(p)$, hence $z \in [\{a_1, \dots, a_n, b_1, \dots, b_n\}]_p$. Every prime p_i occurs exactly a times in the factorization of z. Therefore, either a_i or b_i (and not both) has to be a leaf-value in the generation tree. If $a > 1$ then additionally the generation tree has to be linear and the rightmost leaf has value a_1 or b_1. Hence, the assignment I such that $I(x_i) = 1$ if and only if a_i is a leaf-value in the generation tree satisfies H in the sense of 1-IN-3-SAT. Therefore $H \in$ 1-IN-3-SAT. □

Now let us consider $\mathrm{GEN}(x^a y^b c)$ for $a, b > 1$. In general, the crucial point in proving hardness for generation problems is to cope with the various different trees that generate a number. In our proofs we force the generation trees to have a specific shape such that the generation is possible only in a predefined way.

Consider an $x^a y^b c$-generation tree. Clearly, the generated number is a product that consists of various multiplicities of c and base elements. As a tool to control these multiplicities we introduce (a,b)-*weighted trees*, where we mark each node with the power its value appears in the root-value. By controlling the marks of the leaves, we can force an $x^a y^b c$-generation tree into the shape of a complete (a,b)-weighted tree.

Definition 4.7 *Let t be a binary tree. $T = (t,g)$ is called (a,b)-weighted tree, $a,b > 1$, if g is a marking-function $g : \mathrm{Node}(t) \to \mathbb{N}$ such that:*

If $x = \mathrm{root}(t)$, then $g(x) = 1$.

If $x \in \mathrm{Node}(t)$ has a left and a right successor x_l and x_r, then $g(x_l) = a \cdot g(x)$ and $g(x_r) = b \cdot g(x)$.

T is called balanced, if $\max_{x\in \mathrm{L}(t)} g(x) \leq \max(a,b) \cdot \min_{x\in \mathrm{L}(t)} g(x)$.
T is called complete, if $\max_{x\in \mathrm{L}(t)} g(x) < \max(a,b) \cdot \min_{x\in \mathrm{L}(t)} g(x)$.

From this definition it immediately follows that the marks have the desired properties. We obtain the following connection to $\mathrm{GEN}(x^a y^b c)$.

Property 4.8 *Let $a,b > 1$. If $T = (t,g)$ is an (a,b)-weighted tree, where t is an $x^a y^b c$-generation tree with values $I(v)$ for all $v \in \mathrm{L}(t)$, then*

$$I(\mathrm{root}(t)) = \prod\nolimits_{v\in \mathrm{L}(t)} I(v)^{g(v)} \cdot \prod\nolimits_{v\in \mathrm{Node}(t)-\mathrm{L}(t)} c^{g(v)}.$$

We want to remark that it is possible to define the notion of (a,b)-weighted trees for $a = 1$ and $b = 1$. However, if $a = 1$ and $b = 1$, then complete trees do not exist. In contrast, for all $a,b > 1$ complete trees exist. Therefore, we require $a,b > 1$.

Proposition 4.9 *Let $a,b > 1$. For every $n \geq 1$ there exists a balanced (a,b)-weighted tree that has n leaves.*

Proof For $n = 1$ take the tree that consists only of the root.
For arbitrary $n > 1$, let $T = (t,g)$ be a balanced (a,b)-weighted tree with $n-1$ leaves. Let $x_0 \in \mathrm{L}(t)$ be a leaf with minimal weight, i.e., $g(x_0) = \min_{x\in \mathrm{L}(t)} g(x)$. Define the tree t' by adding in t successors x_l and x_r to x_0, and define $g' : \mathrm{Node}(t') \to \mathbb{N}$ by $g'(x) \stackrel{df}{=} g(x)$ for all $x \in \mathrm{Node}(t)$, $g'(x_l) \stackrel{df}{=} a \cdot g(x_0)$, and $g'(x_r) \stackrel{df}{=} b \cdot g(x_0)$. This defines an (a,b)-weighted tree $T' \stackrel{df}{=} (t', g')$ with

$$\begin{aligned}
\max\nolimits_{x\in \mathrm{L}(t')} g'(x) &= \max(\max\nolimits_{x\in \mathrm{L}(t)} g(x), \max(g'(x_l), g'(x_r))) \\
&= \max(\max\nolimits_{x\in \mathrm{L}(t)} g(x), \max(a,b) \cdot g(x_0)) \\
&= \max(\max\nolimits_{x\in \mathrm{L}(t)} g(x), \max(a,b) \cdot \min\nolimits_{x\in \mathrm{L}(t)} g(x)) \\
&= \max(a,b) \cdot \min\nolimits_{x\in \mathrm{L}(t)} g(x) \\
&\leq \max(a,b) \cdot \min\nolimits_{x\in \mathrm{L}(t')} g'(x). \qquad (2)
\end{aligned}$$

Hence T' is balanced. □

Now we show that for each $n \geq 1$ there exists a complete (a,b)-weighted tree with nearly n leaves. Note that such a tree is polynomial-time constructible.

Proposition 4.10 *Let $a, b > 1$. For every $n \geq 1$ there exists a complete (a, b)-weighted tree with at least n and at most $2n - 1$ leaves.*

Proof Proposition 4.9 gives a balanced (a, b)-weighted tree T with n leaves. If all leaves have minimal weight, then T is complete. Otherwise, there are k, $1 \leq k \leq n - 1$, leaves of minimal weight. If we add two successors to each of these leaves, then the minimal weight increases. So in inequality (2), $\leq$ changes to $<$. So the resulting tree T' is complete. T' has $n - k + 2k = n + k$ leaves where $n \leq n + k \leq 2n - 1$. $\square$

Now we show that if the generation tree is not the desired complete tree, then at least one leaf-value is taken to a power that is too large.

Proposition 4.11 *Let $a, b > 1$. Let $T = (t, g)$ be a complete (a, b)-weighted tree with n leaves. If $T' = (t', g')$ is an (a, b)-weighted tree with more than n leaves, then there exists a leaf $y \in \mathrm{L}(t')$ such that*

$$g'(y) > \max_{x \in \mathrm{L}(t)} g(x).$$

Proof Without loss of generality we can assume $a \geq b$. Fix a shortest way in terms of deleting and adding leaves that transforms t to t'. We have to change at least one leaf $x_0 \in \mathrm{L}(t)$ to an inner node of t'. Let x_l and x_r be the successors of x_0. We obtain

$$g'(x_l) = a \cdot g(x_0) \geq \max(a, b) \cdot \min_{x \in \mathrm{L}(t)} g(x) > \max_{x \in \mathrm{L}(t)} g(x).$$

Hence, every $y \in t'$ that is reachable from x_l fulfills $g'(y) \geq g'(x_l) > \max_{x \in \mathrm{L}(t)} g(x)$. $\square$

Next we show that balanced (a, b)-weighted trees have a height which is bounded logarithmically in the number of leaves.

Proposition 4.12 *Let $a \geq b > 1$. Let $T = (t, g)$ be a balanced (a, b)-weighted tree with n leaves. If d denotes the maximal depth of a leaf of t, then*

$$d \leq \log_b(a) \cdot (1 + \log_2(n))$$

Proof Let $m \stackrel{df}{=} \min_{v \in \mathrm{L}(t)} g(v)$. Hence t contains a complete binary tree of depth $\geq \log_a(m)$, hence $\log_a(m) \leq \log_2(n)$. T is balanced, so $b^d \leq am$ which is equivalent to $d \leq \log_b(am)$. Therefore,

$$d \leq \log_b(am) = \log_b(a) \cdot \log_a(am) \leq \log_b(a) \cdot (1 + \log_2(n)).$$

$\square$

Theorem 4.13 *For $a, b, c \geq 1$ and $p(x, y) \stackrel{df}{=} x^a y^b c$, $\mathrm{GEN}(p)$ is $\leq_m^p$-complete for* NP.

Proof By Proposition 4.6, we can assume $a, b > 1$. Containment in NP follows from Theorem 4.2. We reduce 1-IN-3-SAT to $\mathrm{GEN}(p)$. Let H be a 3-CNF formula with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$. Let $p_1, p_2, \ldots$ be the prime numbers larger than c. Define $a_i \stackrel{df}{=} p_{m+i} \prod_{x_i \in C_j} p_j$ and $b_i \stackrel{df}{=} p_{m+i} \prod_{\overline{x_i} \in C_j} p_j$. Let $T = (t, g)$ be a complete (a, b)-weighted tree with k leaves where $n \leq k \leq 2n-1$ and $\mathrm{L}(t) = \{v_1, \ldots, v_k\}$ (such a tree exists by Proposition 4.10). Furthermore, let d be the maximal depth of a leaf of t. Define $a_i \stackrel{df}{=} p_{m+i}$ for $i = n + 1, \ldots, k$,

$$
\begin{aligned}
B &\stackrel{\text{df}}{=} \left\{a_i' \stackrel{\text{df}}{=} a_i^{a^d b^d / g(v_i)} : 1 \le i \le k\right\} \cup \left\{b_i' \stackrel{\text{df}}{=} b_i^{a^d b^d / g(v_i)} : 1 \le i \le n\right\}, \text{ and} \\
z &\stackrel{\text{df}}{=} \prod\nolimits_{i=1}^{m+k} p_i^{a^d b^d} \cdot \prod\nolimits_{v \in t - \mathrm{L}(t)} c^{g(v)}.
\end{aligned}
$$

Proposition 4.12 shows that (B, z) is polynomial-time computable.

If $H \in$ 1-IN-3-SAT, then there is an assignment $I_H : \{x_1, \ldots, x_n\} \to \{0,1\}$ that satisfies exactly one literal in each clause. We obtain

$$
\prod_{I_H(x_i)=1} a_i \cdot \prod_{I_H(x_i)=0} b_i \cdot \prod_{i=n+1}^{k} p_{m+i} = \prod_{i=1}^{m+k} p_i.
$$

We consider t as an p-generation tree with values

$$
I_t(v_i) \stackrel{\text{df}}{=} \begin{cases} a_i' & \text{, if } i = 1, \ldots, n \text{ and } I_H(x_i) = 1 \\ b_i' & \text{, if } i = 1, \ldots, n \text{ and } I_H(x_i) = 0 \\ a_i' & \text{, if } i = n+1, \ldots, k. \end{cases}
$$

By Property 4.8, $I_t(\mathrm{root}(t))$, the value of the root, can be evaluated as follows.

$$
\begin{aligned}
I_t(\mathrm{root}(t)) &= \prod_{v \in \mathrm{L}(t)} I_t(v)^{g(v)} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)} \\
&= \prod_{I_H(x_i)=1} (a_i')^{g(v_i)} \cdot \prod_{I_H(x_i)=0} (b_i')^{g(v_i)} \cdot \prod_{i=n+1}^{k} (a_i')^{g(v_i)} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)} \\
&= \prod_{I_H(x_i)=1} a_i^{a^d b^d} \cdot \prod_{I_H(x_i)=0} b_i^{a^d b^d} \cdot \prod_{i=n+1}^{k} a_i^{a^d b^d} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)} \\
&= \left(\prod_{I_H(x_i)=1} a_i \cdot \prod_{I_H(x_i)=0} b_i \cdot \prod_{i=n+1}^{k} p_{m+i} \right)^{a^d b^d} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)} \\
&= \prod_{i=1}^{m+k} p_i^{a^d b^d} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)} = z.
\end{aligned}
$$

Hence $(B, z) \in \mathrm{GEN}(p)$.

Assume $(B, z) \in \mathrm{GEN}(p)$. So there exists an (a, b)-weighted tree $T' = (t', g')$, where t' is a p-generation tree from B for z. For each $v \in \mathrm{Node}(t')$ define $I_{t'}(v)$ as the value of node v. Each element of B has exactly one prime factor from $p_{m+1}, \ldots, p_{m+k}$. Since z has all these prime factors at least once, t' must have at least k leaves. Assume t' has more than k leaves. By Proposition 4.11, there exists $v \in \mathrm{L}(t')$ such that $g'(v) > \max_{x \in \mathrm{L}(t)} g(x)$. $I_{t'}(v)$ has exactly one prime factor from $p_{m+1}, \ldots, p_{m+k}$; say p_{m+i} with exponent $a^d b^d / g(v_i)$. Hence

$$
p_{m+i}^{a^d b^d / g(v_i) \cdot g'(v)}
$$

is a factor of $I_{t'}(\mathrm{root}(t'))$. From $a^d b^d / g(v_i) \cdot g'(v) > a^d b^d$ it follows that $I_{t'}(\mathrm{root}(t')) \neq z$.
So t' has exactly k leaves. Each prime $p_{m+1}, \ldots, p_{m+k}$ must appear as factor in a value of some leaf. Therefore, for $i = 1, \ldots, n$, either a_i' or b_i' is a value of a leaf (but not both). Define $I_H : \{x_1, \ldots, x_n\} \to \{0,1\}$ such that $I_H(x_i) = 1 \Leftrightarrow_{\mathrm{df}} a_i'$ is a leaf-value of t'. Observe that I_H shows $H \in$ 1-IN-3-SAT. $\square$

5 The Generation Problem $\mathrm{GEN}(\boldsymbol{x^c + ky})$

So far we do not have upper bounds for generation problems of polynomials $p(x, y) = q(x) + ky$, where q is non-linear and $k \geq 2$. The obvious algorithm guesses and verifies generation trees. How large are these trees? To answer this, observe that the trees are of a special form: When we go from the root to the leaves in y-direction, then in each step, the length of the value decreases by one bit. When we go in x-direction, then in each step, the length is bisected. It follows that the size of such trees grows faster than any polynomial, but not as fast as $2^{\log^2 n}$. Therefore, $\mathrm{GEN}(p) \in \mathrm{NTIME}(2^{\log^2 n})$. We do not have to guess complete generation trees. If a subtree generates some value b, then it suffices to store b instead of the whole subtree. We need to store a value b every time we go in x-direction. So we need space $O(n \log n)$.

Proposition 5.1 $\mathrm{GEN}(p) \in \mathrm{NTIME\text{-}SPACE}(2^{\log^2 n}, n \log n)$ *for all* $p(x, y) = q(x) + ky$ *where* $k \geq 2$ *and* q *is a non-linear polynomial.*

Because of the special form of a generation tree for such polynomials, the generation problem can be solved by special alternating machines: Some z can be generated via p from A if and only if there exist $z_1, \ldots, z_n \leq z$ such that $n \leq |z|$, $z = z_1$, $z_n \in A$, and for all $1 \leq i < n$, $z_i = p(z_{i+1}, y_i)$ where y_i can be generated via p from A and $|y_i| \leq \frac{1}{2}|z_i|$. An alternating machine can check this predicate in polynomial time with a logarithmic number of alternations. Furthermore, in existential parts the machine guesses polynomially many bits. In contrast, in universal parts it guesses logarithmically many bits.

This discussion shows that $\mathrm{GEN}(p)$ can be solved with quite restricted resources. However, we do not know whether $\mathrm{GEN}(p)$ belongs to NP. Standard diagonalizations show that there exist oracles A and B such that $\mathrm{BPP}^A \not\subseteq \mathrm{NTIME}(2^{\log^2 n})^A$ and $\mathrm{coNP}^B \not\subseteq \mathrm{NTIME}(2^{\log^2 n})^B$. Therefore, we should not expect $\mathrm{GEN}(p)$ to be hard for any class that contains BPP or coNP. This rules out many reasonable classes above NP to be reducible to $\mathrm{GEN}(p)$. We consider this as a hint that $\mathrm{GEN}(p)$ could be contained in NP, but we do not have a proof for this. We leave this as an open question.

Nevertheless, in this section we prove lower bounds. The main result, Theorem 5.16, shows that if $p(x, y) = x^c + ky$ where $c, k \geq 1$, then $\mathrm{GEN}(p)$ is $\leq_m^p$-hard for NP. The proof is difficult for two reasons which we want to explain for $p(x, y) = x^2 + 2y$.

1. We have to encode NP-computations into generation problems. For this, we need to construct an instance (B, z) of $\mathrm{GEN}(p)$ that represents information about a given NP-computation. The elements of B must be chosen in a way so that squaring will not destroy this information. This is difficult, since squaring a number heavily changes its (binary) representation.

2. We construct (B, z) such that if z can be generated, then x must be chosen always from B (and is not a generated number). So the generation tree is linear. Because of the factor 2, in any step, the number generated so far is shifted to the left. We have to cope with this shifting.

With regard to item 2, our construction makes sure that the size of the linear generation tree is bounded. So the number of shifts is bounded. For B we choose numbers that are much longer than this bound such that each number is provided with a unique stamp. The stamps make sure that there is at most one possible tree that generates z. In particular, this fixes the sequence of numbers from B that are chosen for x. This keeps the shifting under control.

The problem in item 1 is more complicated and also more interesting. It comes down to prove NP-hardness of the following extended sum-of-subset problem.

$$\mathrm{SOS}_2 \stackrel{df}{=} \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\}(\textstyle\sum_{i \in I} w_i^2 = z)\}$$

(In the proof we use a promise problem related to SOS_2, but for simplicity we argue with SOS_2 in this sketch.) First we reduce 1-IN-3-SAT to SOS and obtain an SOS instance $w = (w_1, \ldots, w_{2n}, z)$. The reduction is such that either $w \notin \mathrm{SOS}$ or there is a selection of exactly n weights which sum up to z. We choose a base b larger than z and all w_i^2. So in the system to base b, z and all w_i^2 fit into one digit. For each w_i, define the following 6-digit numbers in the system to base b.

$$\begin{aligned} a_i &\stackrel{df}{=} [11000w_i]_b \\ r_i &\stackrel{df}{=} [10001w_i]_b \end{aligned}$$

The set of all a_i and all r_i build the weights for the SOS_2 instance we want to construct. The intention is to use the weight a_i whenever w_i is used in the sum that yields z, and to use r_i whenever w_i is not used. The squares of a_i and r_i look as follows with respect to base b.

$$\begin{array}{lcl} a_i^2 & \stackrel{df}{=} & [\ 1\quad 2\ \ 1\ \ 0\ \ 0\ \ 2w_i\ \ 2w_i\ \ 0\ \ 0\quad 0\quad w_i^2]_b \\ r_i^2 & \stackrel{df}{=} & [\ 1\quad 0\ \ 0\ \ 0\ \ 2\ \ 2w_i\quad 0\quad\ 0\ \ 1\ \ 2w_i\ \ w_i^2]_b \end{array}$$

Note that a_i^2 and r_i^2 have the same first digit, the same last digit, and the same digit at the middle position. At all other positions, either a_i^2 or r_i^2 has digit 0. In the sum for SOS_2, for every i, either a_i or r_i is used. Therefore, in system b, the last digit of this sum becomes predictable: It must be $\sum_i w_i^2$. This is the most important point in our argumentation. Also, we choose exactly n weights a_i and n weights r_i. With $s_1 \stackrel{df}{=} \sum_i w_i$, $s_2 \stackrel{df}{=} \sum_i w_i^2$, and $\overline{z} \stackrel{df}{=} s_1 - z$ we can easily describe the destination number for the SOS_2 instance.

$$z' \stackrel{df}{=} [2n\ \ 2n\ \ n\ \ 0\ \ 2n\ \ 2s_1\quad 2z\quad 0\ \ n\ \ 2\overline{z}\quad s_2]_b$$

We obtain the instance $(a_1, r_1, \ldots, a_{2n}, r_{2n}, z')$ which belongs to SOS_2 if and only if $(w_1, \ldots, w_{2n}, z) \in \mathrm{SOS}$. This shows NP-hardness for SOS_2 and solves the difficulty mentioned in item 2.

We inductively use this technique to show that for all $c \geq 1$, the following extended sum-of-subset problem is NP-complete.

$$\mathrm{SOS}_c \stackrel{df}{=} \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\}(\textstyle\sum_{i \in I} w_i^c = z)\}.$$

We need SOS_c as a auxiliary problem for generation problems. However, we feel that this new NP-completeness result is interesting in its own right.

5.1 Notations

In the proofs below we have to construct natural numbers that contain information about NP computations. In addition, these numbers have to contain this information in a way such that exponentiation will not destroy it. For this we need to consider numbers with respect to several bases b. Therefore, we introduce the following notations. For $b \geq 2$ define $A_b = \{0, \ldots, b-1\}$ to be the alphabet that contains b digits. As abbreviation we write A instead of A_2. For digits $a_0, \ldots, a_{n-1} \in A_b$, let $[a_{n-1} \cdots a_0]_b \stackrel{df}{=} \sum_{i=0}^{n-1} a_i b^i$. This means that $[a_{n-1} \cdots a_0]_b$ is the number that is represented by $a_{n-1} \cdots a_0$ with respect to base b.

We will consider vectors of weights $W = (w_1, \ldots, w_{2n})$ such that certain selections of these weights sum up to given destination numbers $z_1, \ldots, z_c$. We group W into pairs (w_1, w_2), (w_3, w_4), and so on. Each pair has a unique stamp u in its binary representation such that the destination number z_c shows the same stamp, but all other pairs have 0's at this position. This allows us to argue that if we want to reach z_c, then from each pair we have to use at least one weight. Moreover, in view of generation problems, we need the stamps still working if the weights are multiplied by small numbers. Therefore, additionally we demand that the stamp u is embedded in s digits 0. We make this precise:

Definition 5.2 *Let* $W = (w_1, \ldots, w_{2n})$ *and* $Z = (z_1, \ldots, z_c)$ *where* $n, c \geq 1$. *Define* $\overline{z}_c \overset{df}{=} (\sum_{w \in W} w^c) - z_c$. *We call* (W, Z) s-distinguishable, $s \geq 1$, *if all* $\mathrm{bin}(w_i^c)$ *have same length* l *where* $l \equiv 1(c)$, *and if for every* $j < n$ *there exist* $t \geq 1$ *and* $u \in 1A^*$ *such that*

1. $\mathrm{bin}(z_c), \mathrm{bin}(\overline{z}_c), \mathrm{bin}(w_{2j+1}^c), \mathrm{bin}(w_{2j+2}^c) \in A^*0^s u 0^s A^t$ *and*
2. *for all* $i \neq j$, $\mathrm{bin}(w_{2i+1}^c), \mathrm{bin}(w_{2i+2}^c) \in A^*0^s 0^{|u|} 0^s A^t$.

5.2 NP-Hardness of Modified Sum-of-Subset Problems

We want to show that for $c, k \geq 1$, the generation problem $\mathrm{GEN}(x^c + ky)$ is $\leq_m^p$-hard for NP. The proof is such that the NP-hardness of modified sum-of-subset problems is shown first, and then this is transferred to the generation problems. Our argumentation for the modified sum-of-subset problems is restricted to instances that meet several requirements. Therefore, it is convenient to define these problems as pairs $(L_{c,s}, R_{c,s})$ of disjoint sets.

Definition 5.3 *Let* $c, s \geq 1$.

$$L_{c,s} \overset{df}{=} \big\{(W, Z) : W = (w_1, \ldots, w_{2n}),\ Z = (z_1, \ldots, z_c),\ (W, Z) \text{ is } ns\text{-distinguishable, and } (\exists I \subseteq \{1, \ldots, 2n\} \text{ s.t. for all } i,\ 2i+1 \in I \Leftrightarrow 2i+2 \notin I)(\forall m \in \{1, \ldots, c\})[\sum_{i \in I} w_i^m = z_m]\big\}$$

$$R_{c,s} \overset{df}{=} \big\{(W, Z) : W = (w_1, \ldots, w_{2n}),\ Z = (z_1, \ldots, z_c),\ (W, Z) \text{ is } ns\text{-distinguishable, and } (\forall I \subseteq \{1, \ldots, 2n\})(\forall m \in \{1, \ldots, c\})[\sum_{i \in I} w_i^m \neq z_m]\big\}$$

Observe that for $c, s \geq 1$, $L_{c,s} \cap R_{c,s} = \emptyset$, $L_{c,s} \in \mathrm{NP}$, and $R_{c,s} \in \mathrm{coNP}$. We show NP-hardness for $c = 1$ first, and then inductively for higher c's.

Lemma 5.4 *For* $s \geq 1$, $(L_{1,s}, R_{1,s})$ *is* $\leq_m^{pp}$*-hard for* NP.

Proof For any word $a_{n-1} \ldots a_0 \in A^*$, let $a[i] \overset{df}{=} a_i$.

$$\text{1-IN-3-SAT} \overset{df}{=} \{H : H \text{ is a 3-CNF formula having an assignment that satisfies exactly one literal in each clause}\}$$

1-IN-3-SAT is NP-complete. For $s \geq 1$, we show 1-IN-3-SAT$\leq_m^{pp}(L_{1,s}, R_{1,s})$ via reduction f. Let H be a 3-CNF formula with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$ where $n \geq 2$. For $0 \leq i \leq n-1$ let

$$\begin{aligned} a &\overset{df}{=} 0^{sn}10^{sn}, \\ a_i &\overset{df}{=} 0^{i(2sn+1)}a0^{(n-i-1)(2sn+1)}, \\ w_{2i+1} &\overset{df}{=} [1a_ic_{i1}\ldots c_{im}]_2, \text{ and} \\ w_{2i+2} &\overset{df}{=} [1a_i\bar{c}_{i1}\ldots\bar{c}_{im}]_2 \end{aligned}$$

where

$$c_{ij} = \begin{cases} 0^{n-1}1 & \text{, if } x_i \text{ is a literal in } C_j \\ 0^n & \text{, otherwise} \end{cases}$$

and

$$\bar{c}_{ij} = \begin{cases} 0^{n-1}1 & \text{, if } \overline{x_i} \text{ is a literal in } C_j \\ 0^n & \text{, otherwise.} \end{cases}$$

Finally, define the reduction as $f(H) \stackrel{df}{=} ((w_1, \ldots, w_{2n}), (z))$ where $d \stackrel{df}{=} n(2sn + 1) + mn$ and

$$z \stackrel{df}{=} n2^d + [a^n(0^{n-1}1)^m]_2.$$

Note that $|\mathrm{bin}(w_i)| = d + 1$. Let $\overline{z} \stackrel{df}{=} \sum_{i=1}^{2n} w_i - z$ and observe that

$$\overline{z} = n2^d + [a^n 0^{nm}]_2 + 2 \cdot [(0^{n-1}1)^m]_2.$$

Therefore, $((w_1, \ldots, w_{2n}), (z))$ is ns-distinguishable.

Let $H \in$ 1-IN-3-SAT. So there exists an assignment $\Phi : \{x_1, \ldots, x_n\} \to \{0, 1\}$ such that each clause is satisfied by exactly one literal. Let

$$I \stackrel{df}{=} \{2i + 1 : 0 \le i < n \text{ and } \Phi(x_i) = 1\} \cup \{2i + 2 : 0 \le i < n \text{ and } \Phi(x_i) = 0\}.$$

It follows $\sum_{i\in I} w_i = z$ and hence $((w_1, \ldots, w_{2n}), (z)) \in L_{1,s}$.

Let $H \notin$ 1-IN-3-SAT and suppose there exists $I \subseteq \{1, \ldots, 2n\}$ such that $\sum_{i\in I} w_i = z$. For all i, $w_i > 2^d$. Also, $z < (n + 1)2^d$, since $[a^n(0^{n-1}1)^m]_2 < 2^d$. Therefore, I contains at most n elements. On the other hand, for all i, $w_i < 2^d + 2^{d-n}$. Since $(n - 1)(2^d + 2^{d-n}) < n2^d$ we obtain $|I| = n$.

Since $((w_1, \ldots, w_{2n}), (z))$ is ns-distinguishable, I must contain exactly one element from each pair (w_{2i+1}, w_{2i+2}). For every $k \in \{0, \ldots, m - 1\}$ there exists exactly one $j \in I$ such that $w_j[kn] = 1$: Otherwise, in $\mathrm{bin}(\sum_{i\in I} w_i)$ there is a 1 at position $kn + t$ where $1 \le t < n$. This is impossible. Therefore, if Φ is defined such that $\Phi(x_i) = 1 \Leftrightarrow 2i + 1 \in I$, then Φ satisfies exactly one literal in each clause. This contradicts our assumption. Hence, $((w_1, \ldots, w_{2n}), (z)) \in R_{1,s}$. □

So far we know that $(L_{1,s}, R_{1,s})$ is NP-hard. This is the induction base of our argumentation. Now we turn to the induction step and show how to transfer hardness to pairs $(L_{c,s}, R_{c,s})$ where $c > 0$.

Lemma 5.5 *For $c, s \ge 1$, $(L_{c,2s+c}, R_{c,2s+c}) \le_m^{pp} (L_{c+1,s}, R_{c+1,s})$.*

Proof We describe the reduction f on input (W, Z) where $W = (w_1, \ldots, w_{2n})$ and $Z = (z_1, \ldots, z_c)$. Let $w = \max(W)$ and choose $l' \equiv 0(c + 1)$ such that $b \stackrel{df}{=} 2^{l'} > 4n(c + 1)! \cdot w^{c+1}$. All w_i belong to A_b. For $1 \le k \le 2n$, define the following weights (where a means *accepted* weight and r means *rejected* weight).

$$\begin{aligned} a_k &\stackrel{df}{=} [110^c0w_k]_b \\ r_k &\stackrel{df}{=} [100^c1w_k]_b \end{aligned}$$

Fix any m such that $1 \le m \le c + 1$. In the following we show how to define the right destination number y_m. After that we define $f(W, Z) = (W', Z')$ where $W' = (a_1, a_2, r_1, r_2, a_3, a_4, r_3, r_4, \ldots, r_{2n-1}, r_{2n})$ and $Z' = (y_1, \ldots, y_{c+1})$. By binomial theorem,

$$a_k^m = \sum_{i=0}^{m} \sum_{j=0}^{i} \binom{m}{i} \binom{i}{j} w_k^{m-i} \cdot b^{(c+2)i+j}, \tag{3}$$

$$r_k^m = \sum_{i=0}^{m} \sum_{j=0}^{i} \binom{m}{i} \binom{i}{j} w_k^{m-i} \cdot b^{(c+2)j+i}. \tag{4}$$

Observe that in (3) each term $b^{(c+2)i+j}$ appears uniquely: If $(c + 2)i + j = (c + 2)i' + j'$, then (since $j < c + 2$ and $j' < c + 2$) $j = j'$ and $i = i'$. Similarly, in (4) each term $b^{(c+2)j+i}$ appears uniquely. Now the idea is, to let $a_k(t)$ denote the coefficient of b^t in equation (3), and to let $r_k(t)$ denote the coefficient of

b^t in equation (4). First, we define $a_k(t)$ and $r_k(t)$ for $0 \le t \le d$, where $d \stackrel{df}{=} (c+3)m$, and then we show that this definition fits to our idea.

$$a_k(t) \stackrel{df}{=} \begin{cases} \binom{m}{i}\binom{i}{j} w_k^{m-i} & : \text{ if } t = (c+2)i + j \text{ for } 0 \le j < i \le m \\ 0 & : \text{ if } t = (c+2)j + i \text{ for } 0 \le j < i \le m \\ \binom{m}{i} w_k^{m-i} & : \text{ otherwise, i.e., if } t = (c+3)i \end{cases} \tag{5}$$

$$r_k(t) \stackrel{df}{=} \begin{cases} 0 & : \text{ if } t = (c+2)i + j \text{ for } 0 \le j < i \le m \\ \binom{m}{i}\binom{i}{j} w_k^{m-i} & : \text{ if } t = (c+2)j + i \text{ for } 0 \le j < i \le m \\ \binom{m}{i} w_k^{m-i} & : \text{ otherwise, i.e., if } t = (c+3)i \end{cases} \tag{6}$$

Note that $a_k(t)$ and $r_k(t)$ depend on m. We abstain from taking m as additional index, since m will always be clear from the context. Observe that the three cases in these definitions are indeed disjoint. So $a_k(t)$ and $r_k(t)$ are well-defined. It follows that $a_k(t)$ and $r_k(t)$ are the announced coefficients from equations (3) and (4). It follows that

$$\begin{aligned} a_k^m &= \sum_{t=0}^{d} a_k(t) \cdot b^t \quad \text{and} \\ r_k^m &= \sum_{t=0}^{d} r_k(t) \cdot b^t. \end{aligned}$$

All $a_k(t)$ and all $r_k(t)$ are less than $b/4n$ and therefore belong to A_b. Hence,

$$a_k^m = [a_k(d) \cdots a_k(1) a_k(0)]_b \quad \text{and} \tag{7}$$

$$r_k^m = [r_k(d) \cdots r_k(1) r_k(0)]_b. \tag{8}$$

Equations (5) and (6) tell us that these representations to base b differ only at positions $t \not\equiv 0 (c+3)$.

In order to define the destination number y_m, we show how to transfer a selection of weights w_k to a corresponding selection of weights a_k^m and r_k^m. Suppose $\sum w_k = z_1$ where the sum ranges over a suitable collection of n weights. Now choose a_k^m for every weight w_k that is used (i.e., *accepted*) in the sum $\sum w_k$; and choose r_k^m for every weight w_k that is not used (i.e., *rejected*) in this sum. The choice of whether to take a_k^m or r_k^m only matters for positions $t \not\equiv 0(c+3)$. By equations (5) and (6), at these positions, either r_k^m has digit 0 and a_k^m has digit $\binom{m}{i}\binom{i}{j} w_k^{m-i}$, or a_k^m has digit 0 and r_k^m has digit $\binom{m}{i}\binom{i}{j} w_k^{m-i}$ (note that $i > 0$ since $i \ne j$). So when we consider the sum of all chosen a_k^m and r_k^m at such a position, then either we see digit

$$\binom{m}{i}\binom{i}{j} \sum_{k \text{ accepted}} w_k^{m-i} = \binom{m}{i}\binom{i}{j} z_{m-i},$$

or we see digit

$$\binom{m}{i}\binom{i}{j} \sum_{k \text{ rejected}} w_k^{m-i} = \binom{m}{i}\binom{i}{j} \overline{z}_{m-i}$$

where $z_0 \stackrel{df}{=} n$ and $\overline{z}_i = \sum_{w \in W} w^i - z_i$ as defined above. This motivates the following digits of the destination number y_m.

$$y(t) \stackrel{df}{=} \begin{cases} \binom{m}{i}\binom{i}{j} z_{m-i} & : \text{ if } t = (c+2)i + j \text{ for } 0 \le j < i \le m \\ \binom{m}{i}\binom{i}{j} \overline{z}_{m-i} & : \text{ if } t = (c+2)j + i \text{ for } 0 \le j < i \le m \\ \sum_{w \in W} \binom{m}{i} w^{m-i} & : \text{ otherwise, i.e., if } t = (c+3)i \end{cases}$$

Here again we abstain from taking m as index, since m will be clear from the context. Define the m-th destination number as

$$y_m \stackrel{df}{=} [y(d) \cdots y(1)y(0)]_b.$$

To finish f's definition, let $f(W, Z) \stackrel{df}{=} (W', Z')$ where $W' = (a_1, a_2, r_1, r_2, a_3, a_4, r_3, r_4, \ldots, r_{2n-1}, r_{2n})$ and $Z' = (y_1, \ldots, y_{c+1})$.

Claim 5.6 *If (W, Z) is $(2s+c)n$-distinguishable, then $f(W, Z) = (W', Z')$ is $2ns$-distinguishable.*

Proof Fix $m = c+1$ and let $d = (c+3)m$. Observe that for every k, $a_k(d) = r_k(d) = 1$. By assumption, $b = 2^{l'}$ for $l' \equiv 0(c+1)$. Hence one digit from A_b corresponds exactly to l' bits. By equations (7) and (8), for every k, $|\mathrm{bin}(a_k^{c+1})| = |\mathrm{bin}(r_k^{c+1})| = d \cdot l' + 1$. This number is $\equiv 1(c+1)$.

We need to understand the structure of $\overline{y}_{c+1} = (\sum_{w \in W'} w^{c+1}) - y_{c+1}$, the complement of y_{c+1}. For this end, define

$$\overline{y}(t) \quad \stackrel{df}{=} \quad \begin{cases} \binom{m}{i}\binom{i}{j}\overline{z}_{m-i} & : \text{ if } t = (c+2)i + j \text{ for } 0 \leq j < i \leq m \\ \binom{m}{i}\binom{i}{j} z_{m-i} & : \text{ if } t = (c+2)j + i \text{ for } 0 \leq j < i \leq m \\ \sum\limits_{w \in W} \binom{m}{i} w^{m-i} & : \text{ otherwise, i.e., if } t = (c+3)i. \end{cases}$$

Observe that for all t, $y(t) + \overline{y}(t) = \sum_{k=1}^{2n} (a_k(t) + r_k(t))$. Hence

$$[y(d) \cdots y(1)y(0)]_b + [\overline{y}(d) \cdots \overline{y}(1)\overline{y}(0)]_b = \sum_{w \in W'} w^m$$

and therefore,

$$\overline{y}_{c+1} = [\overline{y}(d) \cdots \overline{y}(1)\overline{y}(0)]_b.$$

Choose any $j < n$ and consider a_{2j+1} and a_{2j+2}. By assumption, (W, Z) is $(2s+c)n$-distinguishable. So there exist $t \geq 1$ and $u \in 1A^*$ such that

1. $\mathrm{bin}(z_c), \mathrm{bin}(\overline{z}_c), \mathrm{bin}(w_{2j+1}^c), \mathrm{bin}(w_{2j+2}^c) \in A^* 0^{(2s+c)n} u 0^{(2s+c)n} A^t$ and
2. for all $i \neq j$, $\mathrm{bin}(w_{2i+1}^c), \mathrm{bin}(w_{2i+2}^c) \in A^* 0^{(2s+c)n} 0^{|u|} 0^{(2s+c)n} A^t$.

If one multiplies a binary number of the form $A^* 0^{i'} u 0^{i'} A^t$ by $m = c+1 \leq 2^c$, then this yields a number of the form $A^* 0^{i'-c} u' 0^{i'-c} A^{t+c}$ where $u' \in A^{|u|+c}$. So in our case, there exist $t' \geq 1$ and $u' \in 1A^*$ such that

1. $\mathrm{bin}(mz_c), \mathrm{bin}(m\overline{z}_c), \mathrm{bin}(mw_{2j+1}^c), \mathrm{bin}(mw_{2j+2}^c) \in A^* 0^{2sn} u' 0^{2sn} A^{t'}$ and
2. for all $i \neq j$, $\mathrm{bin}(mw_{2i+1}^c), \mathrm{bin}(mw_{2i+2}^c) \in A^* 0^{2sn} 0^{|u'|} 0^{2sn} A^{t'}$.

Let $t_a = c+2$. For all i, $a_i(t_a) = mw_i^c$, $r_i(t_a) = 0$, $y(t_a) = mz_c$, and $\overline{y}(t_a) = m\overline{z}_c$. So for $t'' = t' + l' \cdot t_a$,

1. $\mathrm{bin}(y_m), \mathrm{bin}(\overline{y}_m), \mathrm{bin}(a_{2j+1}^m), \mathrm{bin}(a_{2j+2}^m) \in A^* 0^{2sn} u' 0^{2sn} A^{t''}$,
2. for all $i \neq j$, $\mathrm{bin}(a_{2i+1}^m), \mathrm{bin}(a_{2i+2}^m) \in A^* 0^{2sn} 0^{|u'|} 0^{2sn} A^{t''}$, and
3. for all i, $\mathrm{bin}(r_{2i+1}^m), \mathrm{bin}(r_{2i+2}^m) \in A^* 0^{2sn} 0^{|u'|} 0^{2sn} A^{t''}$.

We obtain the analogous three statements for r_{2j+1} and r_{2j+2} by looking at the position $t_r = 1$. Here for all i, $a_i(t_r) = 0$, $r_i(t_r) = mw_i^c$, $y(t_r) = m\overline{z}_c$, and $\overline{y}(t_r) = mz_c$. Hence (W', Z') is $2ns$-distinguishable. □

Claim 5.7 *If $(W,Z) \in L_{c,2s+c}$, then $f(W,Z) = (W',Z') \in L_{c+1,s}$.*

Proof By Claim 5.6, (W',Z') is $2ns$-distinguishable. Let I be as in the definition of $L_{c,2s+c}$, and let $\overline{I} \stackrel{df}{=} \{1,\ldots,2n\} - I$. Note $|I| = |\overline{I}| = n$. We choose all a_i such that $i \in I$ and all r_i such that $i \in \overline{I}$. Note that this collection of weights from W' is suitable to show that (W',Z') belongs to $L_{c+1,s}$ (i.e., when numbering the weights of W' from 1 to $4n$, then the indices of chosen weights form an I' where $2i+1 \in I' \Leftrightarrow 2i+2 \notin I'$). Fix any $m \in \{1,\ldots,c+1\}$. Our selection of weights induces the following sum.

$$\begin{aligned} z' &\stackrel{df}{=} \sum_{k\in I} a_k^m + \sum_{k\in\overline{I}} r_k^m \\ &= \sum_{k\in I} [a_k(d)\cdots a_k(1)a_k(0)]_b + \sum_{k\in\overline{I}} [r_k(d)\cdots r_k(1)r_k(0)]_b \end{aligned}$$

We have seen that all $a_k(t)$ and all $r_k(t)$ are less than $b/4n$. So for every t,

$$z'(t) \stackrel{df}{=} \sum_{k\in I} a_k(t) + \sum_{k\in\overline{I}} r_k(t)$$

is less than b. This means that if we consider the weights to base b and sum up digit by digit, then there is no sum that is carried forward. It follows that

$$z' = [z'(d)\ldots z'(1)z'(0)]_b.$$

From equations (5) and (6) we obtain

$$z'(t) = \begin{cases} \binom{m}{i}\binom{i}{j}\sum_{k\in I} w_k^{m-i} & : \text{ if } t = (c+2)i+j \text{ for } 0 \le j < i \le m \\ \binom{m}{i}\binom{i}{j}\sum_{k\in\overline{I}} w_k^{m-i} & : \text{ if } t = (c+2)j+i \text{ for } 0 \le j < i \le m \\ \binom{m}{i}\sum_{w\in W} w^{m-i} & : \text{ otherwise, i.e., if } t = (c+3)i. \end{cases}$$

So for all t, $z'(t) = y(t)$ and therefore, $z' = y_m$. This shows $(W',Z') \in L_{c+1,s}$. □

Claim 5.8 *If $(W,Z) \in R_{c,2s+c}$, then $f(W,Z) = (W',Z') \in R_{c+1,s}$.*

Proof By Claim 5.6, (W',Z') is $2ns$-distinguishable. Assume $(W',Z') \notin R_{c+1,s}$, i.e., there exists I_a and I_r, subsets of $\{1,\ldots,2n\}$, and there exists some $m \in \{1,\ldots,c+1\}$ such that

$$\sum_{k\in I_a} a_k^m + \sum_{k\in I_r} r_k^m = y_m. \tag{9}$$

Let $t_a = (c+2)(m-1)$. For all k, $a_k(t_a) = mw_k$, $r_k(t_a) = 0$, and $y(t_a) = mz_1$. In Equation (9), we can consider the weights to base b and can sum up digit by digit without obtaining a sum that is carried forward. By looking at position t_a we obtain $y(t_a) = \sum_{k\in I_a} a_k(t_a)$ and hence

$$z_1 = \sum_{k\in I_a} w_k.$$

So we found a collection of weights from W whose sum is z_1. This is a contradiction. □

This complete the proof of Lemma 5.5. □

Lemma 5.9 *For $c, s \geq 1$, $(L_{c,s}, R_{c,s})$ is $\leq_m^{pp}$-hard for* NP.

Proof The proof is by induction on c. The induction base follows from Lemma 5.4 while the induction step follows from Lemma 5.5. □

Theorem 5.10 *For $c \geq 1$, the following sum-of-subset problem is $\leq_m^{p}$-complete for* NP.

$$\mathrm{SOS}_c \overset{df}{=} \{(a_1, \dots, a_n, b) : \exists I \subseteq \{1, \dots, n\}(\textstyle\sum_{i\in I} a_i^c = b)\}.$$

Proof Clearly, $\mathrm{SOS}_c \in \mathrm{NP}$. For given (W, Z) where $W = (w_1, \dots, w_{2n})$ and $Z = (z_1, \dots, z_c)$ let $f(W, Z) \overset{df}{=} (w_1, \dots, w_{2n}, z_c)$. Observe $(L_{c,1}, R_{c,1}) \leq_m^{pp} \mathrm{SOS}_c$ via f. So by Lemma 5.9, SOS_c is NP-hard. □

5.3 NP-Hardness of $\mathrm{GEN}(x^c + ky)$

Starting from Lemma 5.9 we transfer NP-hardness to generation problems. First, we show this for $c > 1$ and then we treat $\mathrm{GEN}(x + ky)$ in a separate lemma.

Lemma 5.11 *For $c \geq 2$, $k \geq 1$ and $s \overset{df}{=} 5k^2(c + 5)$, $(L_{c,s}, R_{c,s}) \leq_m^{pp} \mathrm{GEN}(x^c + ky)$.*

Proof We describe the reduction f on input (W, Z) where $W = (w_1, \dots, w_{2n})$ and $Z = (z_1, \dots, z_c)$. We may assume that all w_i and z_j are divisible by 2^{cns}. Otherwise, use $W' = (2^{cns}w_1, \dots, 2^{cns}w_{2n})$ and $Z' = (2^{cns}z_1, 2^{2cns}z_2, \dots, 2^{ccns}z_c)$ instead of W and Z. Let $l \overset{df}{=} |\mathrm{bin}(w_1^c)|$ and note that $l > cns$. If $k = 1$, then we use $a = 0$ as auxiliary weight. Otherwise, if $k \geq 2$, then we use $a = 2^{(l-1)/c}$.

$$B \overset{df}{=} \{a, 2^{l-1} + 1\} \cup \{w_1 k, w_2 k, w_3 k^2, w_4 k^2, \dots, w_{2n-1} k^n, w_{2n} k^n\} \tag{10}$$

$$d \overset{df}{=} k^{cn}(z_c + 2^{l-1} + 1) + a^c \cdot \sum_{i=1}^{c-1} k^i \cdot \sum_{i=0}^{n-1} k^{ic} \tag{11}$$

If $n2^{l-1} \leq z_c < n2^l$, then $f(W, Z) \overset{df}{=} (B, d)$, otherwise $f(W, Z) \overset{df}{=} (\emptyset, 0)$. In the following we show $(L_{c,s}, R_{c,s}) \leq_m^{pp} \mathrm{GEN}(x^c + ky)$ via f.

Case 1: Assume $(W, Z) \in L_{c,s}$. Hence there exist weights $x_1, \dots, x_n \in W$ such that $\sum_{i=1}^n x_i^c = z_c$ where $x_1 \in \{w_1, w_2\}$, $x_2 \in \{w_3, w_4\}$, and so on. Therefore, $n2^{l-1} \leq z_c < n2^l$ and so $f(W, Z) = (B, d)$. We describe the generation of d. Clearly, $y_0 \overset{df}{=} 2^{l-1} + 1$ can be generated. For $j \geq 1$, let

$$y_j \overset{df}{=} k^c \cdot y_{j-1} + (k^j x_j)^c + a^c \cdot \sum_{i=1}^{c-1} k^i. \tag{12}$$

If y_{j-1} can be generated, then so can y_j: For $k = 1$ this is trivial. For $k \geq 2$, start with y_{j-1} and apply the generation $y_{\mathrm{new}} = a^c + k \cdot y_{\mathrm{old}}$ for $c - 1$ times. Then apply the generation $y_{\mathrm{new}} = (k^j x_j)^c + k \cdot y_{\mathrm{old}}$ (note that $k^j x_j \in B$). This yields y_j. Hence y_n can be generated. From equation (12) we obtain

$$y_n = k^{cn} \sum_{i=1}^{n} x_j^c + k^{cn}(2^{l-1} + 1) + a^c \cdot \sum_{i=1}^{c-1} k^i \cdot \sum_{i=0}^{n-1} k^{ic}.$$

It follows that $d = y_n$ and therefore, $(B, d) \in \mathrm{GEN}(x^c + ky)$.

Case 2: Assume $(W, Z) \in R_{c,s}$. If $z_c < n2^{l-1}$ or $z_c \geq n2^l$, then $f(W, Z) = (\emptyset, 0) \notin \mathrm{GEN}(x^c + ky)$ and we are done. So let us assume $n2^{l-1} \leq z_c < n2^l$ and $f(W, Z) = (B, d) \in \mathrm{GEN}(x^c + ky)$. In the remaining proof we will derive a contradiction which will prove the lemma.

If $k \geq 2$, then from equation (11) and $z_c < n2^l$ we obtain $d < (n+1)2^l k^{cn} + a^c k^{cn+1}$. Hence

$$k = 1 \quad \Rightarrow \quad d < (n+1)2^l, \tag{13}$$

$$k \geq 2 \quad \Rightarrow \quad d < 2^l k^{n(c+3)}. \tag{14}$$

Claim 5.12 *There exist $m \geq 1$, $y_0 \in B$ and $x_1, \ldots, x_m \in B - \{0, 2^{l-1} + 1\}$ such that*

$$d = k^m y_0 + \sum_{i=1}^{m} k^{m-i} x_i^c. \tag{15}$$

Proof We have seen that $l > cns$. From equations (13) and (14) it follows that $d < 2^{l+ns-2} < 2^{2l-2}$. For all $x \in B - \{0\}$, $|\mathrm{bin}(x^c)| \geq l$. So if z can be generated and is not already in B, then $|\mathrm{bin}(z)| \geq l$ and therefore $z \geq 2^{l-1}$. If we apply the generation rule $x^c + ky$ for $x = z$ and any y, then, since $c \geq 2$, we obtain $z' \geq 2^{2l-2} > d$ which cannot be used to generate d. Similarly, if we apply the generation rule $x^c + ky$ for $x = 2^{l-1} + 1 \in B$ and any y, then we obtain $z' \geq 2^{2l-2} > d$ which cannot be used to generate d. Hence, there exists a generation of d such that in each step, x is chosen from $B - \{0, 2^{l-1} + 1\}$. From $z_c \geq 2^{l-1}$ and equation (11) it follows that $d \geq 2^l k^{cn}$ and hence $d \notin B$. Therefore, d can be generated in the following linear way: There exist $m \geq 1$, $y_0 \in B$, and $x_1, \ldots, x_m \in B - \{0, 2^{l-1} + 1\}$ such that if $y_i \stackrel{df}{=} x_i^c + k \cdot y_{i-1}$ for $1 \leq i \leq m$, then $y_m = d$. This is equivalent to the statement in the claim. $\square$

Claim 5.13

1. $y_0 = 2^{l-1} + 1$.
2. *If $k = 1$, then $m \leq 2n$.*
3. *If $k \geq 2$, then $m = cn$.*

Proof First, we show $m < ns/k^2$. Assume $m \geq ns/k^2$ and $k = 1$. By Claim 5.12, $d > m2^{l-1}$. From equation (13) it follows that $d > 2^{l-1} ns/k^2 \geq 2^{l+1} \cdot n(c+5) > d$ which is a contradiction. Assume $m \geq ns/k^2$ and $k \geq 2$. By Claim 5.12, $d > 2^{l-1} k^{m-1}$. From equation (14) it follows that $d > 2^l k^{(ns/k^2)-2} \geq 2^l k^{5n(c+3)} > d$ which is a contradiction. Therefore,

$$m < ns/k^2. \tag{16}$$

Assume $y_0 \neq 2^{l-1} + 1$, i.e., $y_0 \in B - \{2^{l-1} + 1\}$. By assumption, all w_i and z_i are $\equiv 0(2^{cns})$. So all elements in $B - \{2^{l-1} + 1\}$ are $\equiv 0(2^{ns})$ (if $k \geq 2$, then $a = 2^{(l-1)/c} \geq 2^{ns}$). From Claim 5.12 we obtain $d \equiv 0(2^{ns})$. However, equation (11) says that $d \equiv k^{cn}(2^{ns})$. Since $0 < k^{cn} < 2^{kcn} < 2^{ns}$ we have $d \not\equiv 0(2^{ns})$. This is a contradiction and we obtain $y_0 = 2^{l-1} + 1$.

We have seen that all elements in $B - \{2^{l-1} + 1\}$ are $\equiv 0(2^{ns})$. By Claim 5.12, $d \equiv k^m(2^{ns})$. By equation (11), $d \equiv k^{cn}(2^{ns})$. By equation (16), $k^m \leq 2^{km} < 2^{ns}$ and $k^{cn} < 2^{ns}$. Therefore, if $k \geq 2$, then $m = cn$. If $k = 1$, then by Claim 5.12, $d \geq (m+1)2^{l-1}$. So by equation (13), $m \leq 2n$. $\square$

Claim 5.14 *For every j, $1 \leq j \leq n$ there exists exactly one i such that $x_i \in \{w_{2j-1}k^j, w_{2j}k^j\}$. If $k \geq 2$, then this i is determined by $i = jc$.*

Proof Fix j. By assumption, (W, Z) is ns-distinguishable. So there exist $t \geq 1$ and $u \in 1A^*$ such that $\mathrm{bin}(z_c), \mathrm{bin}(w^c_{2j-1}), \mathrm{bin}(w^c_{2j}) \in A^* 0^{ns} u 0^{ns} A^t$ and for all $i \neq j$, $\mathrm{bin}(w^c_{2i-1}), \mathrm{bin}(w^c_{2i}) \in A^* 0^{ns} 0^{|u|} 0^{ns} A^t$. Let $r \stackrel{df}{=} 2ns + |u| + t$. In the following calculation we are mainly interested in the lower r bits of all x^c_i. If $\alpha \stackrel{df}{=} [u]_2$, then

$$
\begin{aligned}
(w^c_{2j-1} \bmod 2^r) &= \alpha 2^{ns+t} + \beta_1 \quad \text{and} && (17)\\
(w^c_{2j} \bmod 2^r) &= \alpha 2^{ns+t} + \beta_2, && (18)
\end{aligned}
$$

where $\beta_1, \beta_2 < 2^t$. We partition the set of indices $\{1, \ldots, m\}$.

$$
\begin{aligned}
J_1 &\stackrel{df}{=} \{i : 1 \leq i \leq m \wedge x_i = w_{2j-1} k^j\}\\
J_2 &\stackrel{df}{=} \{i : 1 \leq i \leq m \wedge x_i = w_{2j} k^j\}\\
J_3 &\stackrel{df}{=} \{1, \ldots, m\} - (J_1 \cup J_2)
\end{aligned}
$$

From equation (15) we obtain

$$
d = \sum_{i \in J_1} k^{m-i} (w_{2j-1} k^j)^c + \sum_{i \in J_2} k^{m-i} (w_{2j} k^j)^c + \sum_{i \in J_3} k^{m-i} x^c_i + k^m y_0. \qquad (19)
$$

Now we study equation (19) modulo 2^r. We start with the first two sums and consider w^c_{2j-1} and w^c_{2j} modulo 2^r. By equations (17) and (18), these terms consist of an upper part (i.e., $\alpha 2^{ns+t}$) and of a lower part (i.e., β_1 or β_2). Let e_1 (resp., e_2) denote the sum of the upper (resp., lower) parts:

$$
\begin{aligned}
e_1 &\stackrel{df}{=} \sum_{i \in J_1} k^{m-i} k^{jc} \cdot \alpha 2^{ns+t} + \sum_{i \in J_2} k^{m-i} k^{jc} \cdot \alpha 2^{ns+t} && (20)\\
e_2 &\stackrel{df}{=} \sum_{i \in J_1} k^{m-i} k^{jc} \beta_1 + \sum_{i \in J_2} k^{m-i} k^{jc} \beta_2 && (21)
\end{aligned}
$$

Moreover, let e_3 denote the sum (this time modulo 2^r) of the last two terms in equation (19):

$$
e_3 \stackrel{df}{=} ((\sum_{i \in J_3} k^{m-i} x^c_i + k^m y_0) \bmod 2^r) \qquad (22)
$$

Clearly, $d \equiv e_1 + e_2 + e_3 \ (2^r)$. We argue that $(d \bmod 2^r) = e_1 + e_2 + e_3$.

For all $i \in J_3$, either $x^c_i = a^c \neq 0$ or $x^c_i = x' k^{ci'}$ where $\mathrm{bin}(x') \in A^* 0^{ns} 0^{|u|} 0^{ns} A^t$ and $1 \leq i' \leq n$. Therefore, for all $i \in J_3$,

$$
(x^c_i \bmod 2^r) < 2^t k^{cn}. \qquad (23)
$$

Moreover, $(y_0 \bmod 2^r) = 1$. Equations (22) and (23) allow an estimation of e_3.

$$
e_3 \leq \sum_{i \in J_3} k^{m-i} 2^t k^{cn} + k^m
$$

If $k = 1$, then by Claim 5.13, $e_3 \leq 2n2^t + 1$. If $k \geq 2$, then $e_3 \leq 2^t k^{cn} k^m + k^m$ and $m = cn$. So for all k,

$$
e_3 < 2^{ns+t-1}. \qquad (24)
$$

Estimate e_2 with help of equation (21) and Claim 5.13:

$$
e_2 \leq m k^{m-1} k^{jc} 2^t \leq 2^{5knc+t} < 2^{ns+t-1} \qquad (25)
$$

Together with (24) this yields

$$e_2 + e_3 < 2^{ns+t}. \tag{26}$$

Finally we turn to e_1. Equation (20) can be written as

$$e_1 = \alpha 2^{ns+t} k^{jc+m} \sum_{i \in J_1 \cup J_2} k^{-i}. \tag{27}$$

Therefore, $e_1 < 2^{|u|+ns+t+5knc} \leq 2^{r-1}$. Together with (26) we obtain $e_1 + e_2 + e_3 < 2^r$ and hence

$$(d \bmod 2^r) = e_1 + e_2 + e_3. \tag{28}$$

By equation (11), $d \equiv k^{cn}(z_c + 1)\ (2^r)$. Recall that $\mathrm{bin}(z_c) \in A^* 0^{ns} u 0^{ns} A^t$. Therefore, $(z_c \bmod 2^r) = \alpha 2^{ns+t} + \gamma$ where $\gamma < 2^t$. Observe $k^{cn}(\alpha 2^{ns+t} + \gamma + 1) < 2^{kcn} 2^{|u|} 2^{ns+t+1} \leq 2^r$. This yields

$$(d \bmod 2^r) = \alpha 2^{ns+t} k^{cn} + k^{cn}(\gamma + 1). \tag{29}$$

Compare equations (28) and (29). The terms e_1 and $\alpha 2^{ns+t} k^{cn}$ are divisible by 2^{ns+t}, while the terms $e_2 + e_3$ and $k^{cn}(\gamma+1)$ are less than 2^{ns+t}. It follows that $e_1 = \alpha 2^{ns+t} k^{cn}$ and therefore, by equation (27),

$$\sum_{i \in J_1 \cup J_2} k^{m-i} = k^{c(n-j)}. \tag{30}$$

For $k = 1$ this implies $|J_1 \cup J_2| = 1$, while for $k \geq 2$ this implies $|J_1 \cup J_2| \geq 1$. Assume $k \geq 2$ and let i' be the maximum of $J_1 \cup J_2$. The left hand side of (30) is $\equiv k^{nc-i'}\ (k^{nc-i'+1})$. So it must be that $k^{c(n-j)} < k^{nc-i'+1}$ and therefore, $k^{nc-i'} = k^{c(n-j)}$. Hence $J_1 \cup J_2 = \{jc\}$. This proves Claim 5.14. $\square$

Assume $k = 1$. By Claims 5.12, 5.13, and 5.14, there exist $\overline{x}_i \in \{w_{2i-1}, w_{2i}\}$ such that

$$d = (2^{l-1} + 1) + \sum_{i=1}^{n} \overline{x}_i^c. \tag{31}$$

Together with equation (11) this shows $z_c = \sum_{i=1}^{n} \overline{x}_i^c$. So $(W, Z) \notin R_{c,s}$ which contradicts our assumption.

Assume $k \geq 2$. By Claim 5.14, for every j, $x_{jc} = \overline{x}_j \cdot k^j$ where $\overline{x}_j \in \{w_{2j-1}, w_{2j}\}$. Moreover, it follows that for every i, if $i \not\equiv 0(c)$, then $x_i = a$. So equation (15) can be written as:

$$\begin{aligned} d &= k^m y_0 + \sum_{j=1}^{n} k^{m-jc} x_{jc}^c + \sum_{\substack{i \in \{1,\ldots,m\},\\ i \not\equiv 0(c)}} k^{m-i} x_i^c & (32)\\ &= k^{nc}(a^c + 1) + k^{nc} \sum_{j=1}^{n} \overline{x}_j^c + a^c \sum_{\substack{i \in \{1,\ldots,nc\},\\ i \not\equiv 0(c)}} k^{nc-i} & (33) \end{aligned}$$

Observe that the right-most sum in (33) can be written as

$$\frac{k^{nc} - 1}{k - 1} - \frac{k^{cn} - 1}{k^c - 1} = \sum_{i=1}^{c-1} k^i \cdot \sum_{i=0}^{n-1} k^{ic}.$$

So we can continue to transform d.

$$d = k^{nc}(a^c + 1) + k^{nc}\sum_{j=1}^{n}\overline{x}_j^c + a^c \cdot \sum_{i=1}^{c-1} k^i \cdot \sum_{i=0}^{n-1} k^{ic}$$

Together with equation (11),

$$z_c = \sum_{j=1}^{n}\overline{x}_j^c. \tag{34}$$

So again $(W, Z) \notin R_{c,s}$ which contradicts our assumption. □

Lemma 5.15 *If* $p(x, y) = x + ky$ *where* $k \geq 1$ *then* $\mathrm{GEN}(p)$ *is* $\leq_m^p$*-complete for* NP.

Proof We have already seen the upper bound (Lemma 4.3) and the lower bound for the case $k = 1$ [vEB79], so let us focus on the lower bound for $k \geq 2$. We $\leq_m^{pp}$-reduce $(L_{1,2k}, R_{1,2k})$ to $\mathrm{GEN}(x + ky)$. Let $W \stackrel{df}{=} (w_1, \ldots, w_{2n})$, $Z \stackrel{df}{=} (z)$ such that $w_i, z \in \mathbb{N}$ $(1 \leq i \leq 2n)$. Let $\ell \stackrel{df}{=} |\mathrm{bin}(k\Sigma_{i=1}^{2n} w_i)|$ and $G \stackrel{df}{=} 2^{\ell+1}$. Define

$$\begin{aligned}
v_1 &\stackrel{df}{=} k(G + w_1),\\
v_2 &\stackrel{df}{=} k(G + w_2),\\
v_i &\stackrel{df}{=} G + w_i, \text{ for } 3 \leq i \leq 2n, \text{ and}\\
z' &\stackrel{df}{=} k(nG + z).
\end{aligned}$$

Now let $(W, Z) \in L_{1,2k}$. Then there is an $I = \{i_1, \ldots, i_n\} \subseteq \{1, \ldots, 2n\}$ such that for all $i \in \{0, \ldots, n-1\}$ exactly one of $\{2i + 1, 2i + 2\}$ is in I and $\sum_{i \in I} w_i = z$. Assume that $i_j < i_t$ if $j < t$. Then $p(p(\ldots p(p(v_{i_1}, v_{i_2}), v_{i_3}), \ldots, v_{i_{n-1}}), v_{i_n}) = k(G + w_{i_1}) + k\sum_{j=2}^{n} G + w_{i_j} = k(nG + z) = z'$.

Now let $(W, Z) \in R_{1,2k}$ and assume that $(v_1, \ldots, v_{2n}, z') \in \mathrm{GEN}(p)$. Observe that $v_i \geq G$ for all $i \in \{1, \ldots, 2n\}$. Let T be a generation tree for z' from $\{v_1, \ldots, v_{2n}\}$ with m leaves. Then obviously $z' \geq \sum_{q \,\in\, \mathrm{fpath}(T)} k^{r(q)}G$. Since for every leaf in T except one there is a path q with $r(q) \geq 1$ we have $nkG + G > nkG + kz = z' \geq (m-1)kG + G$ and therefore $m \leq n$. Suppose there is an $i \in \{0, \ldots, n-1\}$ such that neither v_{2i+1} nor v_{2i+2} is a value of a leaf in T. We know that (W, Z) is $2kn$ distinguishable. Adding G to a w_j $(1 \leq j \leq 2n)$ and nG to z does not interfere with the distinguishing gaps of the values by the choice of G. Multiplying some of the values with k, decreases the size of the distinguishing gaps by at most $\lfloor \log k + 1 \rfloor$. Hence there is a $u \in 1A^*$ and a $t \geq 1$ such that $\mathrm{bin}(z') \in A^*0^{kn}u0^{kn}A^t$ and for all $j \neq i$, both $\mathrm{bin}(v_{2j+1})$ and $\mathrm{bin}(v_{2j+2})$ are in $A^*0^{kn}0^{|u|}0^{kn}A^t$. Since in every step of the generation the size of the distinguishing gap is reduced by at most $\lfloor \log k + 1 \rfloor$ and since there are at most $n - 1$ steps in the whole generation process, z' can not be generated. Hence for all $i \in \{0, \ldots, n-1\}$ exactly one element of $\{v_{2i+1}, v_{2i+2}\}$ is a value of a leaf in T and $m = n$. If there were a path q in T with $r(q) > 1$ then $nkG + G > z' \geq k^2G + (n-2)kG + G \geq nkG + G$ $(*)$ would hold. Therefore $\mathrm{fpath}(T) = \{l^{n-1}\} \cup \{l^i r : 0 \leq i \leq n-2\}$. Since $v_1, v_2 \geq kG$ the value of the leaf with the path l^{n-1} has to be one of $\{v_1, v_2\}$ otherwise again $(*)$ would hold. So there are $\{i_1, \ldots, i_n\}$ such that $i_1 \in \{1, 2\}$ and

$$\begin{aligned}
z' &= p(p(\ldots p(p(v_{i_1}, v_{i_2}), v_{i_3}), \ldots, v_{i_{n-1}}), v_{i_n})\\
&= k(G + w_{i_1}) + k\textstyle\sum_{j=2}^{n} G + w_{i_j}\\
&= k(nG + \textstyle\sum_{j=1}^{n} w_{i_j})\\
&= k(nG + z)
\end{aligned}$$

and therefore $\sum_{j=1}^{n} w_{i_j} = z$ which is a contradiction. Hence $(v_1, \ldots, v_{2n}, z') \notin \mathrm{GEN}(p)$. □

We combine the auxiliary results proved so far and formulate the main result of this section.

Theorem 5.16 *For* $c, k \geq 1$, $\mathrm{GEN}(x^c + ky)$ *is* $\leq_m^p$*-hard for* NP.

Proof Follows from Lemmas 5.9, 5.11, and 5.15. □

6 Conclusion

We summarize our results on the complexity of $\mathrm{GEN}(f)$ in the following table. Every lower bound is given by the fact that there exists an f from the considered class of operations whose generation problem is complete for the respective class. All operations are polynomial-time computable.

The gap between NP and $\mathrm{NTIME}(2^{\log^2 n})$ in the last rows of the table below calls the attention to an interesting open question: Does $\mathrm{GEN}(q(x) + ky)$ belong to NP if q is non-linear and $k \geq 2$? Since the generation trees for these polynomials may be of superpolynomial size, the obvious algorithm of guessing and verifying the tree is not applicable. Also, we could not find more compact representations as in Theorem 4.2. There are generation trees where almost all nodes take different values. Therefore it may be possible that we really have to calculate all of them. Perhaps there are special polynomials of the form $q(x) + ky$ for which the closure is very regular, as in Theorem 4.2, case (1)? Another possibility to solve the problem could be to have a closer look at the restricted alternating machines we describe in Section 5. What are the exact capabilities of these machines?

operation	lower bound	Theorem	upper bound	Theorem
arbitrary	recursively enumerable	3.2	recursively enumerable	3.1
length-monotonic	EXPTIME	3.5	EXPTIME	3.3
length-monotonic and commutative	EXPTIME	3.5	EXPTIME	3.3
length-monotonic and associative	PSPACE	3.7	PSPACE	3.6
length-mon., assoc., and commutative	NP	3.9	NP	3.8
all Polynomials $\neq q(x) + ky$	NP	3.9	NP	4.2
$x + y$	NP	3.9	NP	3.8
$x \cdot y$	NP	4.6	NP	3.8
$x^a y^b c$	NP	4.13	NP	4.2
all Polynomials $= q(x) + ky$	NP	5.16	$\mathrm{NTIME}(2^{\log^2 n})$	5.1
$x^c + ky$	NP	5.16	$\mathrm{NTIME}(2^{\log^2 n})$	5.1

References

[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.

[Mat70] Y. V. Matiyasevich. Enumerable sets are diophantine. *Doklady Akad. Nauk SSSR*, 191:279–282, 1970. Translation in Soviet Math. Doklady, 11:354–357, 1970.

[vEB79] P. van Emde Boas. Complexity of linear problems. In *Proceedings of the Fundamentals of Computation Theory Conference*, pages 117–120, 1979.