Quantifier Elimination for Regular Integer Linear-Exponential Programming

Mikhail R. Starchak
St. Petersburg State University
St. Petersburg, Russia
m.starchak@spbu.ru

***Abstract*— Regular integer linear-exponential programming (RegILEP) asks whether a system of inequalities of the form $\sum_{i=1..n}\left(a_i \cdot x_i + b_i \cdot 2^{x_i}\right) \leq c$, where all coefficients are integers, has a solution in the integers whose binary representations belong to some regular set over the alphabet $\{0,1\}$. RegILEP has recently been proved decidable in ExpSpace using purely automata-theoretic techniques. The first contribution of the paper is a novel decision procedure for RegILEP, which works in a quantifier elimination fashion: after specifying a total order on the variables, the procedure gradually excludes the exponential occurrences of the leading variable and then eliminates the linear ones. This decision procedure meets the existing ExpSpace upper bound for the problem. As a complementary result, we show that regular integer linear programming for the domain defined by the regular expression $(00 \cup 01)^*$ is PSpace-complete.**

***Index Terms*— arithmetic theories, exponentiation, automatic structures, quantifier elimination, integer linear programming**

I. Introduction

Integer linear programming feasibility (ILP) is a classical problem: given a matrix $A \in \mathbb{Z}^{m \times n}$ and a vector $\boldsymbol{b} \in \mathbb{Z}^m$, determine whether there exists a solution $\boldsymbol{x} \in \mathbb{Z}^n$ to the system of linear inequalities $A \cdot \boldsymbol{x} \leq \boldsymbol{b}$. Define the problem S-ILP as ILP, where the variables range over the set S, and let us call *regular integer linear programming* (RegILP) the S-ILP for the sets S of non-negative integers whose binary expansions are accepted by a DFA over the alphabet $\{0,1\}$, and this DFA is given as part of the input. Now suppose that we are given two matrices $A, B \in \mathbb{Z}^{m \times n}$ and a vector $\boldsymbol{c} \in \mathbb{Z}^m$. Then, *Integer Linear-Exponential Programming* feasibility (ILEP) over $S \subseteq \mathbb{N}$ is the problem of the existence of an $\boldsymbol{x}$ such that

$$A \cdot \boldsymbol{x} + B \cdot 2^{\boldsymbol{x}} \leq \boldsymbol{c},$$
$$\boldsymbol{x} \in S^n,$$

where $2^{\boldsymbol{x}}$ denotes the vector $(2^{x_1}, \ldots, 2^{x_n})$. When the binary expansions of the non-negative integers from the set S are accepted by a DFA, we obtain *regular ILEP* (RegILEP). In this paper, we develop a novel algorithm for deciding RegILEP and establish the computational complexity of RegILP.

From the logical point-of-view, ILP can be seen as the decision problem for the positive existential conjunctive fragment of the first-order theory of the integers $\mathbb{Z}$ with addition and order, which is usually called Presburger arithmetic (PrA). Indeed, this fragment comprises sentences $\exists \boldsymbol{x}.\varphi(\boldsymbol{x})$, where φ is a conjunction of linear inequalities with integer coefficients. While ILP can be decided using a variety of techniques [7], [17], [27], the only known algorithms for RegILEP (with restrictions on S) originate from the very recent developments in the decision procedures for existential Presburger arithmetic ($\exists$PrA) with the integer base 2 exponentiation $2^x : n \mapsto \lfloor 2^n \rfloor$, and its extensions. The algorithms from [1], [8] first specify an ordering over the variables and then, starting from the leading variable x, exclude the exponential occurrences 2^x (*linearisation*) and then eliminate the linear ones x (*elimination*) in the same way as in $\exists$PrA. These algorithms are purely algebraic, and the algorithm from [8] can decide S-ILEP in NP if the set S is represented via a formula of $\exists$PrA with the predicate $V_2(x,y)$, which is true whenever y is the largest power of 2 dividing x. The sets definable using such formulas form a proper subset in the class of all regular sets [14], [30].

There is only one algorithm for deciding RegILEP in its full generality, and it comes from the recent breakthrough result by Draghici, Haase, and Manea [10]. They prove that $\exists$PrA with the base 2 exponentiation and regular predicates (called *existential generalised Semënov arithmetic* ($\exists$GSA)) is decidable in ExpSpace. Here, a predicate R over $\mathbb{N}^n$ is *regular* if there is an NFA over the alphabet $\{0,1\}^n$ that recognises exactly the binary expansions of $\boldsymbol{a} \in \mathbb{N}^n$ such that $R(\boldsymbol{a})$. Since this theory contains all regular predicates, a purely algebraic approach is clearly not applicable, and the decidability of $\exists$GSA is proved via a reduction to the non-emptiness problem for a restricted version of *labelled affine vector addition system with states* (LAVASS). This generalisation of finite automata admits a representation of the atomic formula $(y = 2^x)$, and thus every formula of $\exists$GSA can be encoded into a restricted LAVASS. Since the non-emptiness problem for restricted LAVASS is decidable in ExpSpace, $\exists$GSA is decidable within the same class. However, the decision procedure is radically different from the quantifier elimination algorithms for ILEP from [1], [8], and the restricted LAVASS is a rather complex object for studying its non-emptiness problem. This stimulates the research on developing a new algorithm that combines algebraic and automata-theoretic paradigms. Our first main result is a novel decision procedure for $\exists$GSA, which gives us a new proof of the following fact.

Theorem 1. The RegILEP problem is decidable in ExpSpace.

To highlight the merits of the quantifier elimination approach compared to the restricted LAVASS, we first informally describe the main ideas underlying our procedure.

On the top-level, it has two steps, linearisation and elimination, which are applied to our formulas repeatedly, but now these steps are inevitably based on automata theory. Elimination of linearly occurring variables follows the algorithms for PrA [4], [7], [13]. To illustrate the key ideas of linearisation, consider the system

$$R(3\cdot 2^x+2\cdot 2^y+3\cdot y+1, 2^x+3\cdot 2^y+2)\wedge(y+3<x), \quad (1)$$

where R is recognised by some $\{0,1\}^2$-NFA. System (1) is first simplified by introducing new linearly occurring variables, so that in every regular predicate R' of the simplified system each coordinate of the NFA that recognises R' will correspond to either a linear or exponentiated variable. In particular, the term $3\cdot 2^x+2\cdot 2^y+3\cdot y+1$ will be replaced by a fresh variable $\hat{y}_1$ and $2^x+3\cdot 2^y+2$ by $\hat{y}_2$. Then, the resulting system is satisfiable in $\mathbb{N}$ if and only if the product $\mathcal{A}$ of the NFAs that recognise its predicates accepts a word of the form

$$\begin{bmatrix}1\\0\\0\\0\\\vdots\end{bmatrix}\underbrace{\begin{bmatrix}1\\1\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\0\\0\\\vdots\end{bmatrix}\dots\begin{bmatrix}0\\0\\0\\0\\\vdots\end{bmatrix}\begin{bmatrix}*\\*\\0\\0\\\vdots\end{bmatrix}\begin{bmatrix}*\\*\\0\\0\\\vdots\end{bmatrix}}_{x-y}\begin{bmatrix}*\\*\\0\\1\\\vdots\end{bmatrix}\begin{bmatrix}*\\*\\0\\0\\\vdots\end{bmatrix}\dots\begin{bmatrix}*\\*\\0\\0\\\vdots\end{bmatrix}\begin{matrix}\hat{y}_1\\\hat{y}_2\\2^x\\2^y\\\vdots\end{matrix} \quad (2)$$

Since the variables $\hat{y}_1$ and $\hat{y}_2$ occur in the system only linearly, they can be eliminated, and for this reason they are highlighted in red. The main concern of linearisation will be the blue part of the runs over such words. While reading the word (2) from right to left, the NFA $\mathcal{A}$ first reaches a transition with the unique 1 in the coordinate that corresponds to 2^y, then reads $(x-y-1)$ tuples with zeros in the coordinates corresponding to non-eliminated variables until reaching a transition with the unique 1 in the coordinate that corresponds to 2^x. The linearisation step updates the final states of $\mathcal{A}$ and introduces new constraints to express the existence of a run in $\mathcal{A}$ between two specific states over a word formed of tuples of zeros with length equal to $(x-y-1)$. The description of these lengths will be obtained via the so-called *Chrobak normal form* [9], [12], [26], [31] of NFAs over unary alphabets.

In contrast to [10], our procedure uses standard tools from automata theory, it can easily be decomposed into independent subroutines, and it also provides a direct way to generalise it to work with the real numbers (which seems to be difficult to do using the restricted LAVASS). By representing tuples of real numbers as ω-words [3], we obtain the following real-number version of GSA. Denote by $\mathscr{B}$ the set of all predicates over $\mathbb{R}^n_{\geq 0}$ recognisable by Büchi automata. Then the positive existential theory of the structure $\langle \mathbb{R}; 0, 1, +, \lfloor\cdot\rfloor, 2^{\lfloor\cdot\rfloor}, \mathscr{B}, \leq\rangle$, where $\lfloor\cdot\rfloor$ is interpreted as the floor function, is decidable in ExpSpace. Every real variable x is replaced by $(u+v)$, where u ranges over $\mathbb{Z}$ and v is a real variable ranging over $[0,1)$; then, $2^{\lfloor x\rfloor}$ becomes equal to 2^u, the variable v occurs in the formula only linearly. Following [3], the problem is split into deciding solvability of a system of regular constraints with integer variables and positive powers of 2 and of a system of ω-regular constraints with real variables from $[0,1)$ and negative powers of 2. Linearising the latter exponentiated variables via the technique described in the present paper, we reduce our problem to the decision problem for ∃GSA. Instead of giving this straightforward generalisation, we prove a complementary result to Theorem 1, which seems to be more surprising. As noted in [10, Conclusion], ∃GSA is PSpace-hard due to PSpace-completeness of the NFA intersection non-emptiness problem [18]. The next theorem strengthens this observation.

Theorem 2. The $(\mathtt{00}\cup\mathtt{01})^*$-ILP problem is PSpace-complete.

The computations of NFAs from the input of the intersection non-emptiness problem are expressed via a system of linear inequalities and constraints specifying that the binary expansion of the value of a linear term $f(\boldsymbol{x})$ is from $(\mathtt{00}\cup\mathtt{01})^*$. To complete the proof, it will be sufficient to notice that every non-negative integer can be represented as $(2x+y)$ for some x and y with binary expansions from the language $(\mathtt{00}\cup\mathtt{01})^*$.

The main results reflect the relationships between integer programming and decision procedures for arithmetic theories. Let us mention some closely related classical and recent works before proceeding to our quantifier elimination for ∃GSA.

Existential arithmetic theories and Integer Programming

For an extension ∃PrA, we can define a counterpart in the world of integer programming. Such problem may require the variables to take their values from some subset of integers and/or use specific kinds of constraints.

The undecidability of the Hilbert's 10th Problem [21] (∃PrA with multiplication [22]) implies that IP with quadratic constraints $\sum_{i=1..n}\left(a_i\cdot x_i+b_i\cdot {x_i}^2\right)\leq c$ is undecidable [16]. For the set of squares $Sq=\{0,1,4,9,\dots\}$, we see that Sq-ILP is expressible in the language of IP with quadratic constraints, but its decidability status is a long-standing open problem with only conditional undecidability results known [23]. The decision procedure for ∃PrA with divisibility [20] is, in essence, the algorithm that decides solvability in the integers of systems $\sum_{i=1..n}\left(a_i\cdot x_i+\sum_{j=1..n} b_{i,j}\cdot(x_i \bmod x_j)\right)\leq c$. The problem is decidable in NExpTime [19] and is undecidable over the set Sq [32]. The decision procedure from [19] gave rise to the decidability in NP of the IP with gcd-constraints [11], where a system of linear inequalities is supplemented with a system of constraints $\gcd(f(\boldsymbol{x}),g(\boldsymbol{x}))\sim d$ for the greatest common divisor function gcd, linear terms $f(\boldsymbol{x})$, $g(\boldsymbol{x})$, $d\in\mathbb{Z}_{>0}$, and a symbol $\sim\in\{\leq,=,\neq,\geq\}$. A quantifier elimination procedure for ∃PrA with exponentiation $x\mapsto 2^x$ and modulo powers of two operator $(x,y)\mapsto x \bmod 2^y$ has recently been applied [8] to integer linear-exponential programming, which was defined as $\sum_{i=1..n}\left(a_i\cdot x_i+\sum_{j=1..n} b_{i,j}\cdot(x_i \bmod 2^{x_j})+c_i\cdot 2^{x_i}\right)\leq d$. This problem was proved to be decidable in NP. It is easy to show that there is a regular predicate $R(u,v,w)$ such that $z=(x \bmod 2^y)$ if and only if $R(z,x,2^y)$. Therefore, linear-exponential systems from [8] are expressible in the language of ∃GSA. On the other hand, allowing in the system arbitrary modulo operators $(x_i \bmod x_j)$, we obtain an algorithmically undecidable problem [32].

II. Finite automata and linear-exponential terms

Our first goal is to perform several syntactic transformations of a given formula of $\exists$GSA so that the resulting formula will be equisatisfiable over $\mathbb{N}$ and prepared for quantifier elimination. This section recalls some necessary definitions, notations and results from automata theory and logic.

A. Regular predicates over non-negative integers

Our decision procedure operates with non-negative integers $\mathbb{N}=\{0,1,2,\dots\}$ and regular predicates over $\mathbb{N}^n$. We treat $\mathbb{B}=\{\mathtt{0},\mathtt{1}\}$ as the alphabet of bits and $\mathbb{B}^n$ as the alphabet of n-tuples of bits. The n-tuple of zeros will be denoted by $\mathbf{0}$ when its size n is clear from the context. The language $(\mathbb{B}^n)^*$ is the set of all words of finite length over $\mathbb{B}^n$ with the unique empty word ϵ of length 0. There is a naturally defined surjection $[\![\cdot]\!]$ from $(\mathbb{B}^n)^*$ to the set $\mathbb{N}^n$. For a word $\mathbf{a}_0\mathbf{a}_1...\mathbf{a}_t\in(\mathbb{B}^n)^*$, we define

$$[\![\mathbf{a}_0\mathbf{a}_1...\mathbf{a}_t]\!]=\mathbf{a}_0+\mathbf{a}_1\cdot 2+\dots+\mathbf{a}_t\cdot 2^t.$$

For a language $L\subseteq(\mathbb{B}^n)^*$, this function defines a subset of $\mathbb{N}^n$ in the following way: $[\![L]\!]=\{[\![\overline{\mathtt{a}}]\!]:\overline{\mathtt{a}}\in L\}$. Here, the vectors from $[\![L]\!]$ are encoded in the *least-significant-digit (lsd)*-first binary notation. Observe that $[\![\cdot]\!]$ becomes bijective if defined over the language $N^n:=\mathbf{0}\cup(\mathbb{B}^n)^*\cdot(\mathbb{B}^n\setminus\mathbf{0})$.

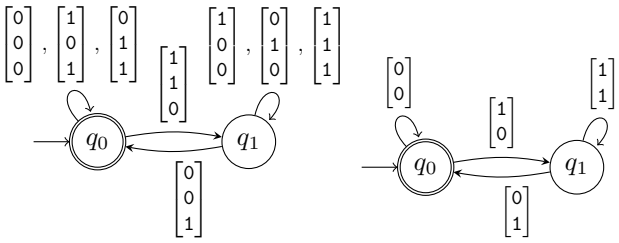
For an alphabet Σ, a *non-deterministic finite automaton over Σ (Σ-NFA)* is a 4-tuple $\mathcal{A}=(Q,S,F,\Delta)$, where Q is a finite set of *states*; $S,F\subseteq Q$ are, respectively, the sets of *initial* and *finial* states, and $\Delta\subseteq Q\times\Sigma\times Q$ is the *transition relation*. When Δ is a function $\Delta:Q\times\Sigma\to Q$ and S is a singleton set, the automaton $\mathcal{A}$ is *deterministic (Σ-DFA)*. A triple $(p,\mathtt{a},q)$ from Δ is called a *transition of* $\mathcal{A}$, and is denoted by $p\xrightarrow{\mathtt{a}}q$. A *run of* $\mathcal{A}$ from a state p_0 to a state p_{t+1} on an input word $\overline{\mathtt{a}}=\mathtt{a}_0\mathtt{a}_1...\mathtt{a}_t$ from Σ^* is a sequence of transitions $p_0\xrightarrow{\mathtt{a}_0}p_1\xrightarrow{\mathtt{a}_1}\dots\xrightarrow{\mathtt{a}_t}p_{t+1}$ such that for every $i\in[0,t]$ we have $p_i\xrightarrow{\mathtt{a}_i}p_{i+1}\in\Delta$. We say that *a word* $\overline{\mathtt{a}}\in\Sigma^*$ *is accepted* by a given Σ-NFA $\mathcal{A}$ if there is a run of $\mathcal{A}$ from a state $p\in S$ to a state $q\in F$ on the input $\overline{\mathtt{a}}$. The set of all words $\overline{\mathtt{a}}\in\Sigma^*$ accepted by $\mathcal{A}$ defines the language $L(\mathcal{A})$.

An n-ary predicate R over $\mathbb{N}$ is called *regular* if there is a $\mathbb{B}^n$-NFA $\mathcal{A}$ such that $L(\mathcal{A})=L'\cdot\mathbf{0}^*$ for a language $L'\subseteq N^n$ and we have $[\![L(\mathcal{A})]\!]=\{\boldsymbol{a}\in\mathbb{N}^n:R(\boldsymbol{a})\}$. In this case we say that $\mathcal{A}$ *recognises* R. In our paper, each regular predicate R will be defined via some NFA $\mathcal{A}=(Q,S,F,\Delta)$, and the size of this predicate is defined as $|R|:=\#Q+\#\Delta$, where $\#$ is the *cardinality* function.

Büchi-Bruyère's theorem [4], [5] states that a predicate R over $\mathbb{N}^n$ is regular (more precise: 2-regular) if and only if it is definable in the structure $\langle\mathbb{N};0,1,+,V_2,\le\rangle$, where $V_2(x,y)$ is true whenever y is the greatest power of 2 dividing x. For example, $V_2(12,y)$ is true only when $y=4$, and x is a power of 2 exactly when $V_2(x,x)$. For our purposes, we only need some basic tools from the "if" part of the proof from [4]. First of all, we use the standard $\mathbb{B}^3$-NFA for addition $(x+y=z)$ and a similar $\mathbb{B}^2$-NFA for doubling $(2\cdot x=y)$ both given in Fig. 1. The $\mathbb{B}^2$-NFA for $(x=y)$ is trivial: it has only one state q_0 (simultaneously initial and final) and two transitions labelled with $\begin{bmatrix}0\\0\end{bmatrix}$ and $\begin{bmatrix}1\\1\end{bmatrix}$. Using these simple automata we can construct a conjunction of formulas that defines the predicate $(a\cdot x=y)$ for a fixed positive integer a. Let $\mathtt{a}_0\mathtt{a}_1...\mathtt{a}_t$ be the binary expansion of a in the lsd-first notation with $t\ge 2$, i.e., we have $[\![\mathtt{a}_0\mathtt{a}_1...\mathtt{a}_t]\!]=a$. Then the desired expression is

$$\begin{aligned}a\cdot y=x\iff\exists\boldsymbol{u}\exists\boldsymbol{v}\Big[&2\cdot y=u_1\wedge\bigwedge_{i=2..t}(2\cdot u_{i-1}=u_i)\wedge\\&(\mathtt{a}_0\cdot y+\mathtt{a}_1\cdot u_1=v_1)\wedge\bigwedge_{i=2..t-1}(v_{i-1}+\mathtt{a}_i\cdot u_i=v_i)\wedge\\&(v_{t-1}+\mathtt{a}_t\cdot u_t=x)\Big].\end{aligned}\tag{3}$$

We will also need an NFA for $(y\le z)$ (projection over the first coordinate of the NFA from Fig. 1a) and NFAs for constants, i.e., that recognise the predicate $(x=a)$ for some fixed $a\in\mathbb{N}$. Such NFAs have a simple form: $\lceil log_2(a+1)\rceil+1$ states and the same number of transitions, which form a chain labelled with bits of the binary expansion of a in the lsd-first notation; the unique final state has a loop labelled with $\mathtt{0}$.



(a) NFA for $(x+y=z)$ (b) NFA for $(2\cdot x=y)$

Fig. 1: Two basic NFA

Remark 1. The NFAs for the predicates $(x+y=z)$, $(y\le z)$, and $(x=a)$ with $a>0$ have a unique final state q_0, do not have transitions $p\xrightarrow{\mathtt{0}}q_0$ for $p\ne q_0$, and q_0 has a unique predecessor (if defined) for every label.

The NFA for divisibility constraints $(d\mid x-r)$ may require exponentially many states in the bit-length of d. In our decision procedure, we will use a succinct representation of these NFA via a pair of integers $d>0$ and $r\in[0,d-1]$ encoded in binary. We recall the standard construction (see e.g. [2], [15]) to show that the existence of a transition $q_k\xrightarrow{\mathtt{a}}q_l$ in the NFA $\mathcal{D}_{d,r}$ for $(d\mid x-r)$ can be verified in polynomial time for every $\mathtt{a}\in\mathbb{B}$ and $k,l\in\mathbb{N}$ encoded in binary. Let $d=2^n\cdot m$, where m is odd. Then $\mathcal{D}_{d,r}$ will have $n+m$ states with the initial state q_m and the final state q_0. To define the transition function, we compute the remainder $c=r \bmod 2^n$ and then define the remainder k such that $2^n\cdot k+c\equiv r\,(\mathrm{mod}\ m)$. Let $\mathtt{a}_0\mathtt{a}_1...\mathtt{a}_{n-1}$ be the lsd-first encoding of c and let b be the multiplicative inverse of 2 modulo m. The NFA $\mathcal{D}_{d,r}$ has the following transitions: for $i\in[0,n-2]:q_{m+i}\xrightarrow{\mathtt{a}_i}q_{m+i+1}$ (reading $n-1$ least significant bits of r); then we have $q_{m+n-1}\xrightarrow{\mathtt{a}_{n-1}}q_k$ (after the n-th bit, the remaining part is congruent to k modulo m); and, finally, for $i\in[0,m-1]$ there are two transitions $q_i\xrightarrow{\mathtt{0}}q_{i\cdot b\,(\mathrm{mod}\ m)}$ and $q_i\xrightarrow{\mathtt{1}}q_{(i-1)\cdot b\,(\mathrm{mod}\ m)}$ (the standard NFA for $(m\mid x-k)$).

The proof of the following lemma is straightforward.

Lemma 1. For any integers $d > 0$ and $r \in [0, d-1]$ the NFA $\mathcal{D}_{d,r}$ recognises the divisibility $(d \mid x - r)$.

In Section IV, we need a description of words recognisable by a $\{\mathbf{0}\}$-NFA $\mathcal{A}$, and this can be done by constructing of the so-called *Chrobak normal form* [9], [12], [31] of $\mathcal{A}$. Instead of giving the formal definition of the normal form, we recall the main property of the algorithm from [26]. Since $\mathcal{A}$ is an NFA over a unary alphabet, the language $L(\mathcal{A})$ can be characterized in terms of arithmetic progressions describing only the lengths of words. For a state q of $\mathcal{A}$ denote by $sl(q)$ the shortest loop that can be done in q. Then we have the following lemma.

Lemma 2 (Sawa [26])**.** Let $\mathcal{A} = (Q, S, F, \Delta)$ be a $\{\mathbf{0}\}$-NFA with $\#Q = n$. Define a set C of pairs of non-negative integers as the union of

- C_1: all pairs $(c, 0)$ such that $c \in [0, n^2 - 1]$ and $\mathcal{A}$ accepts the string $\mathbf{0}^c$.
- C_2: all pairs (c, d) such that there is a run $q_0 \xrightarrow{n-1} \overset{d}{\curvearrowright} q \xrightarrow{l} q_f$ in $\mathcal{A}$ (where the lengths of paths are written instead of the words of the corresponding length) for some $q_0 \in S$, $q \in Q$, $q_f \in F$, and non-negative integers $d = sl(q)$, $c \in [n-1, n^2-1]$ and $l = c - (n-1)$.

Then $L(\mathcal{A}) = \bigcup_{(c,d)\in C} \{\mathbf{0}^k : k \in (c + d \cdot \mathbb{N})\}$.

This lemma is applied in the linearisation process, which is described in Section IV.

If a system of regular constraints $\bigwedge_{i=1..m} R_i(\boldsymbol{x}_i)$ has only linearly occurring variables, it can be decided for solvability in $\mathbb{N}$ as follows. Let $\boldsymbol{x}$ be the vector composed of the variables from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. For $i \in [1, m]$, define the regular predicates $R'_i(\boldsymbol{x})$ such that for all $\boldsymbol{x}$ we have $R_i(\boldsymbol{x}) \Leftrightarrow R'_i(\boldsymbol{x}_i)$. Then, in terms of binary representations of solutions, we must check for non-emptiness the intersection of $L(\mathcal{A}'_1), \ldots, L(\mathcal{A}'_m)$ for NFAs $\mathcal{A}'_i$ that recognise R'_i. Non-deterministic Algorithm 1 is an adaptation of a textbook procedure for the *intersection non-emptiness problem for NFAs* (see e.g. [18], where the problem is proved to be PSPACE-complete). In this algorithm, for every $\mathbf{b} \in \mathbb{B}^k$, where k is the size of $\boldsymbol{x}$, we denote by $\pi_{\mathcal{A}_i}(\mathbf{b})$ the tuple of bits that corresponds to the variables $\boldsymbol{x_i}$ in $\boldsymbol{x}$. This algorithm works in space polynomial in the size of the system. Savitch's theorem gives us the following lemma.

Lemma 3. For a system of regular constraints $\bigwedge_{i=1..m} R_i(\boldsymbol{x}_i)$ Algorithm 1 decides its solvability in non-negative integers $\mathbb{N}$ in space polynomial in $|R_1| + \cdots + |R_m|$.

This lemma will give the PSPACE upper bound for RegILP in Section VI, and modifications of Algorithm 1 will help us to prove two lemmas from Sections III and IV.

B. Existential arithmetic of addition and exponentiation

Presburger arithmetic is the first-order theory of the structure $\langle \mathbb{Z}; 0, 1, +, \leq \rangle$, where all symbols are interpreted in the usual way. *Semënov arithmetic* is the first-order theory of the

Algorithm 1 Solvability in $\mathbb{N}$ of systems of regular constraints

Input: $\varphi(x_1, ..., x_n)$: system $\bigwedge_{i=1..m} R_i(\boldsymbol{x}_i)$, where each R_i is recognised by the NFA $\mathcal{A}_i = (Q_i, S_i, F_i, \Delta_i)$.
Output: $\top$ if φ has solutions in $\mathbb{N}$, and otherwise $\bot$.

```
c ← ∏_{i=1..m} #Q_i
guess q = (q_1, ..., q_m) ← from S_1 × ··· × S_m
while c > 0 do
    if q ∈ F_1 × ··· × F_m then return ⊤
    else
        b ← guess from B^n, where for every i ∈ [1, m]
              there exists p_i ∈ Q_i with (q_i, π_{A_i}(b), p_i) ∈ Δ_i
        p ← guess from Q_1 × ··· × Q_n such that
              (q_i, π_{A_i}(b), p_i) ∈ Δ_i
        q ← p;   c ← c − 1
return ⊥
```

extension of this structure with the integer base 2 exponentiation $2^x : n \mapsto \lfloor 2^n \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. This, in particular, means that 2^x maps to zero every negative integer n. Both theories have quantifier elimination in some naturally defined extensions (see [13], [25] for the former and [24], [28] for the latter theory).

In this paper we focus on the complexity of the existential Semënov arithmetic extended with all regular predicates. Over the integers, a predicate $R(\boldsymbol{x})$ is called *regular* if the predicate $\boldsymbol{x} \in \{|\boldsymbol{a}| : R(\boldsymbol{a}), \boldsymbol{a} \in \mathbb{Z}^n\}$ is regular (see Section II-A), where $|\boldsymbol{a}| = (|a_1|, ..., |a_n|)$. Denote by $\mathscr{R}$ the set of regular predicates over $\mathbb{Z}$ represented via the minimal DFAs that recognise them. *Generalised Semënov arithmetic* (GSA) is the first-order theory of the structure $\langle \mathbb{Z}; 0, 1, +, 2^x, \mathscr{R}, \leq \rangle$. Terms and formulas in the language of GSA we call, respectively, GSA-terms and GSA-formulas. The *length* $|\varphi|$ of a GSA-formula φ is parametric in the size of the predicates $R_1, \ldots, R_m$ that appear in φ, i.e., $|\varphi|$ is the number of symbols required to write down φ, assuming binary encoding of natural numbers and $|R_i|$ as the size of each symbol R_i. The existential GSA is decidable in EXPSPACE [10], however, for two quantifier blocks GSA is undecidable [6]. We will only deal with the existential fragment (∃GSA). Our goal now is to reduce in non-deterministic polynomial time the problem of deciding satisfiability in $\mathbb{Z}$ of quantifier-free GSA-formulas to the satisfiability in $\mathbb{N}$ of GSA-formulas in the following normalized form:

$$\bigwedge_{i=1..m} R_i(2^{x_1}, x_1, \ldots, 2^{x_n}, x_n), \tag{4}$$

where the regular predicates R_i are recognised by some NFAs $\mathcal{A}_i$. Such systems will be called *quantifier-free regular linear-exponential (lin-exp) systems*. Notice that each predicate R_i in (4) has $2n$ slots for variables, however, we do not require the NFA $\mathcal{A}_i$ to work over the alphabet $\mathbb{B}^{2n}$. For example, the system may contain the equality $(x_1 + 2^{x_2} = x_3)$ recognisable by the $\mathbb{B}^3$-NFA from Fig. 1a. Crucially, the relation $(x = 2^y)$ is not regular [10], and we only assign exponentiated variables 2^{x_j} to some coordinates of $\mathcal{A}_i$. The next lemma performs syntactic transformations of ∃GSA-formulas, which are standard for formulas of Semënov arithmetic [1], [8], [10], [24].

Lemma 4. The decision problem for ∃GSA is reducible in non-deterministic polynomial time to the problem of solvability in $\mathbb{N}$ of quantifier-free regular lin-exp systems.

The proof is simple and it is omitted for space reasons. In the future, by *exponentiated variables* we assume the exponential terms $2^{x_1}, \dots, 2^{x_n}$, and the terms $2^{x_1}, x_1, \dots, 2^{x_n}, x_n$ will be called *trivial terms over the variables* $x_1, \dots, x_n$.

III. Ordered regular lin-exp systems

This section is preparatory for the (non-deterministic) quantifier elimination procedure described in Sections IV and V. This procedure takes on input a regular quantifier-free lin-exp system $\varphi(2^{x_1}, x_1, \dots, 2^{x_n}, x_n)$ and operates during its execution with two sorts of variables called *eliminated* and *free*. For this reason, we introduce *regular linear-exponential (lin-exp) systems* $\varphi(\hat{y}_1, \dots, \hat{y}_k, 2^{x_1}, x_1, \dots, 2^{x_n}, x_n)$, which have the same definition as quantifier-free regular lin-exp systems, but the variables are now split into *eliminated* $\hat{y}_1, \dots, \hat{y}_k$, occurring in φ only linearly, and *free* $x_1, \dots, x_n$. It is clear that quantifier-free regular lin-exp systems are just regular lin-exp systems, where all variables are free.

As usual for decidable extensions of ∃PrA [8], [11], [19], we will work with systems supplemented with a total order on the trivial terms over free variables occurring in the system.

A. Orderings

For a regular lin-exp system φ we define a set

$$T_\varphi := \{2^{x_0}, 2^{x_1}, \dots, 2^{x_n}, x_0, x_1, \dots, x_n, 0\}$$

comprising all trivial terms over the free variables of φ and auxiliary terms 2^{x_0}, x_0, and 0. Then, by the *ordering for the set* T_φ we assume a quantifier-free regular lin-exp system θ that is composed of the constraints $(s \geq t)$ for every $s, t \in T_\varphi$ specifying a total order on T_φ such that

(i) $(t \geq 0)$ for every $t \in T_\varphi \setminus \{x_0\}$ and $(0 \geq x_0)$;
(ii) $(2^{x_i} \geq x_i)$ for every $i \in [1, n]$;
(iii) $(2^{x_i} \geq 2^{x_j} \iff x_i \geq x_j)$ for every $i, j \in [0, n]$.

Here and along the paper, we will represent such ordering θ as $(t_0 \geq t_1 \geq \dots \geq t_{2n} \geq x_0 = 0)$. It is clear that a regular lin-exp system φ has solutions in $\mathbb{N}$ if and only if there exists an ordering θ for T_φ such that $(\varphi \wedge \theta)$ also has solutions in $\mathbb{N}$. Having constructed (in non-deterministic polynomial time) an ordering θ for the set T_φ, we do the following updates in θ to define a *generalised ordering for* φ. First, for every free variable x we add to the ordering θ the constraint $P_2(2^x)$ specifying that 2^x is a power of 2. The NFA that recognises P_2 is trivial: it has an initial state q_0, a final state q_1, and three transitions: $q_0 \xrightarrow{0} q_0$, $q_0 \xrightarrow{1} q_1$, and $q_1 \xrightarrow{0} q_1$. These self-evident constraints are necessary for the linearisation procedure from Section IV. Second, for every $i, j \in [1, n]$ such that 2^{x_j} is the immediate successor of x_i in θ, we

1) introduce a new variable z_i;
2) add to θ the regular predicate $R_\lambda(x_i, 2^{z_i})$ specifying that 2^{z_i} is the largest power of 2 less than or equal to x_i;
3) insert 2^{z_i} into θ between x_i and 2^{x_j}: $x_i \geq 2^{z_i} \geq 2^{x_j}$.

The NFA that recognises $R_\lambda(u, v)$ has an initial state q_0, a final state q_1, and four transitions: loops in q_0 labelled with $\begin{bmatrix}0\\0\end{bmatrix}$ and $\begin{bmatrix}1\\0\end{bmatrix}$, a transition with $\begin{bmatrix}1\\1\end{bmatrix}$ from q_0 to q_1, a loop with $\begin{bmatrix}0\\0\end{bmatrix}$ in the state q_1. The free variables $z_1, \dots, z_l$ will be treated similarly to free variables: iteratively linearised and eliminated. However, the linearisation step will also rely on the fact that every such variable z is equal to $\lfloor \log x \rfloor$ for some free variable x. For this reason, we call these variables *logarithmic*. The next lemma summarises the transformations described above. It is used as a non-deterministic subroutine in Algorithm 3.

Lemma 5 (Order: specify a generalised ordering θ for a regular lin-exp system φ)**.** For the system φ we can construct in non-deterministic polynomial time in $|\varphi|$ a generalised ordering θ such that φ is equisatisfiable over $\mathbb{N}$ with $(\varphi \wedge \theta)$.

Example 1. Consider a regular lin-exp system

$$\varphi := (\hat{z} \geq 3 \cdot \hat{y} + x_1) \wedge (\hat{y} + 2^{x_2} = 2^{x_1}) \wedge (x_1 + x_2 = 2^{x_2}). \quad (5)$$

For this system there is the following generalised ordering:

$$\theta_1 := (2^{x_2} \geq x_2 \geq 2^{z_2} \geq 2^{x_1} \geq 2^{x_0} \geq x_1 \geq x_0 = 0) \wedge P_2(2^{x_1}) \wedge P_2(2^{x_2}) \wedge R_\lambda(x_2, 2^{z_2}),$$

where z_2 is a logarithmic variable for x_2. Observe that the regular lin-exp system $(\varphi \wedge \theta_1)$ does not have solutions in the non-negative integers. Indeed, from θ_1 it follows that x_1 is either equal to 0 or 1. When $x_1 = 0$, the third constraint from Equation (5) entails the unsatisfiable equality $(x_2 = 2^{x_2})$. If $x_1 = 1$, then the same constraint implies that either $x_2 = 0$ or $x_2 = 1$. However, from θ_1 we see that $x_2 \geq 2^{x_1} = 2$, and the system is unsatisfiable. If we choose another ordering

$$\theta_2 := (2^{x_1} \geq 2^{x_2} \geq x_1 \geq x_2 \geq 2^{z_2} \geq 2^{x_0} \geq x_0 = 0) \wedge P_2(2^{x_1}) \wedge P_2(2^{x_2}) \wedge R_\lambda(x_2, 2^{z_2}),$$

then the system $(\varphi \wedge \theta_2)$ is satisfiable. For example, we can take $z_2 = 1$, $x_2 = 3$, $x_1 = 5$, $\hat{y} = 24$, and $\hat{z} = 100$.

The following simple lemma will be useful in Section V.

Lemma 6. Let θ be a generalised ordering for a regular lin-exp system φ. Then the inequalities in θ have the form either

O1 $(2^x \geq 2^y \geq \dots \geq x_0 = 0)$ or
O2 $(2^x \geq x \geq 2^y \geq \dots \geq x_0 = 0)$, where the variable y is the logarithmic variable for x.

Proof. By (ii), the leading term of θ must be an exponentiated variable. Denote this term by 2^x. Due to items (ii) and (iii), the immediate successor of 2^x can be either an exponentiated variable 2^y or x. In the second case, the immediate successor of x cannot be a linear variable z, because otherwise (by (iii)) the exponential term 2^z must be greater than x in θ. Hence, the successor of x is a term 2^y, and, by definition of generalised orderings, y is the logarithmic variable for x. □

In Section IV, exponential terms are excluded from regular lin-exp systems, and thus an ordering for these systems we define as follows. Let θ be a generalised ordering for a regular

lin-exp system φ, and let θ' be the subformula of θ, where all constraints with terms $2^{x_i}, i \in [1,n]$ not featuring in φ are excluded. Then θ' is called an *ordering for* φ.

Example 2. Consider the system φ from Example 1, where the exponential term 2^{x_1} and the linear occurrences of x_2 are replaced, respectively, with eliminated variables $\hat{u}$ and $\hat{v}$:

$$\varphi \coloneqq (\hat{z} \geq 3 \cdot \hat{y} + x_1) \wedge (\hat{y} + 2^{x_2} = \hat{u}) \wedge (x_1 + \hat{v} = 2^{x_2}). \quad (6)$$

Now the free variable x_1 does not occur exponentiated in φ and the variable x_2 does not occur linearly. An ordering for φ will, however, have linear occurrences of x_2, e.g.

$$\theta'_1 \coloneqq (2^{x_2} \geq x_2 \geq 2^{z_2} \geq 2^{x_0} \geq x_1 \geq x_0 = 0) \wedge P_2(2^{x_2}) \wedge R_\lambda(x_2, 2^{z_2}).$$

If θ is an ordering for φ, then the regular lin-exp system $(\varphi \wedge \theta)$ is called an *ordered regular lin-exp system* and will be denoted by capital Greek letters Φ, Ψ, etc. In the future, we will often omit the word "ordered" because the presence of an ordering follows from the notation. Each system Φ can be regarded as a succinct representation of a $\mathbb{B}^{2(n+1+l)}$-NFA, where $n+1$ and l are, respectively, the numbers of free and logarithmic variables of Φ. Let us formally define this NFA.

B. NFAs associated with regular lin-exp systems

Let an ordered regular lin-exp system Φ have the form

$$\bigwedge_{i=1..m} R_i(\hat{y}_1, ..., \hat{y}_k, 2^{x_0}, x_0, ..., 2^{x_n}, x_n, 2^{z_1}, z_1, ..., 2^{z_l}, z_l), \quad (7)$$

where $\hat{y}_1, ..., \hat{y}_k$ are *eliminated*, $x_0, ..., x_n$ are *free*, $z_1, ..., z_l$ are *logarithmic* variables, and for every $i \in [1,m]$ the predicates R_i in (7) are recognised by NFAs $\mathcal{A}_i = (Q_i, S_i, F_i, \Delta_i)$. For simplicity of notation, we denote $s \coloneqq 2 \cdot (n+l+1)$. Using the NFAs $\mathcal{A}_i$, the *labelled transition system* $\mathcal{T}_\Phi \coloneqq (Q, \mathbb{B}^s, \Delta)$ *associated with the system* Φ is defined as follows:

- Q is the Cartesian product $Q_1 \times \cdots \times Q_m$;
- Δ is composed of the triples $(\boldsymbol{p}, \mathbf{a}, \boldsymbol{q}) \in Q \times \mathbb{B}^s \times Q$ for which there exists a vector $\mathbf{b} \in \mathbb{B}^k$ such that for every $i \in [1,m]$ the triple $(p_i, \pi_{\mathcal{A}_i}(\mathbf{b}, \mathbf{a}), q_i)$ is a transition in Δ_i.

To extend the definition of $\mathcal{T}_\Phi$ onto the NFA associated with the system Φ, we prove the following simple lemma.

Lemma 7. Let Φ be a regular lin-exp system, and let $\boldsymbol{q}, \boldsymbol{q}'$ be two states of the labelled transition system $\mathcal{T}_\Phi$. Then the problem of reachability in $\mathcal{T}_\Phi$ of the state $\boldsymbol{q}'$ from $\boldsymbol{q}$ via a run over a word from $\mathbf{0}^*$ is decidable in space polynomial in $|\Phi|$.

Proof. To decide reachability, we slightly modify Algorithm 1. First remove line 2 because now $\boldsymbol{q}$ is given as part of the input. In line 4 we will check the equality $\boldsymbol{q} = \boldsymbol{q}'$. Finally, in line 6 we now guess a vector $\mathbf{b} \in \mathbb{B}^k$ and in both lines 6 and 7 replace $\pi_{\mathcal{A}_i}(\mathbf{b})$ with $\pi_{\mathcal{A}_i}(\mathbf{b}, \mathbf{0})$ in order to consider transitions labelled with the tuple $\mathbf{0}$ in $\mathcal{T}_\Phi$. □

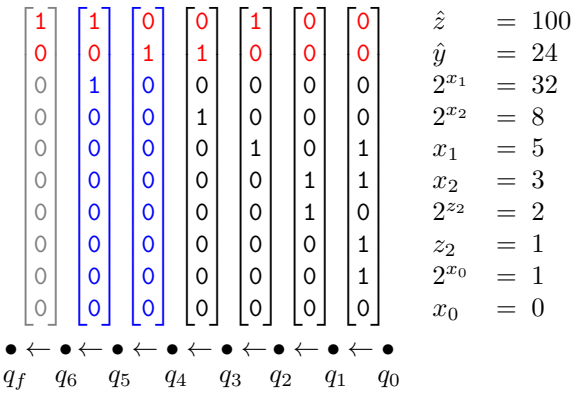
Remark 2. For an encoded in binary $c \leq \#Q$ given as part of the input, we can check in space polynomial in $|\Phi|$ that $\boldsymbol{q}$ is reachable from $\boldsymbol{q}'$ in $\mathcal{T}_\Phi$ via a run over $\mathbf{0}^c$: apply the changes of Lemma 7, remove lines 1 and 4 and return $\boldsymbol{q} = \boldsymbol{q}'$ in line 9.

If the algorithm from Lemma 7 returns $\top$ for states $\boldsymbol{q}, \boldsymbol{q}'$, then we will say that $\boldsymbol{q}'$ is $\mathbf{0}^*$-*reachable* from $\boldsymbol{q}$ in $\mathcal{T}_\Phi$. Now the NFA $\mathcal{A}_\Phi$ *associated with (the ordered regular lin-exp system)* Φ is defined as a tuple (Q, S, F, Δ), where Q and Δ come from $\mathcal{T}_\Phi$, and the sets of initial and final states are defined using the NFAs $\mathcal{A}_i$ as follows:

- S is exactly $S_1 \times \cdots \times S_m$;
- F is composed of the states $\boldsymbol{q} \in Q$ such that $(\boldsymbol{p}, \mathbf{a}, \boldsymbol{q}) \in \Delta$ with the coordinate of $\mathbf{a}$ that corresponds to the leading exponential term of θ equal to 1, and, moreover, a state from $F_1 \times \cdots \times F_m$ is $\mathbf{0}^*$-reachable from $\boldsymbol{q}$ in $\mathcal{T}_\Phi$.

From Lemma 7, we know that checking that $\boldsymbol{q}$ is a final state of $\mathcal{A}_\Phi$ can be done in space polynomial in $|\Phi|$. The NFA $\mathcal{A}_\Phi$ can be regarded as the result of projection over the coordinates that correspond to the variables $\hat{y}_1, \ldots, \hat{y}_k$ of the product of NFAs $\mathcal{A}_1, \ldots, \mathcal{A}_m$, and the definition of the set F reflects the usual problem of cutting the leading zeros in the accepting runs via the right quotient construction. This is how the existential quantifiers are eliminated in Büchi-Bruyère's theorem [4], [5]. To illustrate the definition of $\mathcal{A}_\Phi$, consider an example.

Example 3. Let $\Phi = (\varphi \wedge \theta_2)$ for the regular lin-exp system φ and the ordering θ_2 from Equation (5). Then there is a run in the $\mathbb{B}^8$-NFA $\mathcal{A}_\Phi$, which reads the assignment from Example 1:



The leading exponential term of θ_2 is 2^{x_1}. The state q_6 is a final state of $\mathcal{A}_\Phi$, because q_f (which was a final state in the product of the NFAs from Φ) is $\mathbf{0}^*$-reachable from q_6 and there is a transition $(q_5, [1,0,0,0,0,0,0,0], q_6)$ in Δ. We focus on the blue columns in Section IV. Notice that the length of the run from q_4 to q_6 is equal to $2 = x_1 - x_2$.

Let us prove the main property of the NFA $\mathcal{A}_\Phi$. In the proof, we assume that the first k bits of tuples from $\mathbb{B}^{k+s}$ correspond to eliminated variables $\hat{\boldsymbol{y}}$, and the last s bits to free variables $\boldsymbol{x}$ (occurring both linearly and exponentiated) of Φ. By $\pi_{\boldsymbol{z}}$ we denote the projection of a tuple from $\mathbb{B}^{k+s}$ over the coordinates that correspond to variables from $\boldsymbol{z}$. The functions $\pi_{\mathcal{A}_i}$ and $\pi_{\boldsymbol{z}}$ are extended over words from $(\mathbb{B}^{k+s})^*$ in the usual way.

Lemma 8. Let $\mathcal{A}_\Phi$ be the $\mathbb{B}^s$-NFA associated with an ordered regular lin-exp system Φ. Then we have

$$[\![L(\mathcal{A}_\Phi)]\!] = \{\boldsymbol{a} \in \mathbb{N}^s : \exists \hat{\boldsymbol{y}}.\Phi(\hat{\boldsymbol{y}}, \boldsymbol{a})\}.$$

Proof. If $\overline{\mathbf{w}} \in (\mathbb{B}^s)^*$ is accepted by the NFA $\mathcal{A}_\Phi$, then by the

definition of $\mathcal{A}_\Phi$ there exists a word $\overline{\mathbf{w}}' \in (\mathbb{B}^{k+s})^*$ such that $\pi_{\boldsymbol{x}}(\overline{\mathbf{w}}') \in \overline{\mathbf{w}} \cdot \mathbf{0}^*$ and for every $i \in [1,m]$ the NFA $\mathcal{A}_i$ accepts $\pi_{\mathcal{A}_i}(\overline{\mathbf{w}}')$. Therefore, if $\boldsymbol{a} = [\![\overline{\mathbf{w}}]\!]$ and $\boldsymbol{b} = [\![\pi_{\hat{\boldsymbol{y}}}(\overline{\mathbf{w}}')]\!]$, then the formula $\Phi(\boldsymbol{b}, \boldsymbol{a})$ evaluates to true and we have $\exists \hat{\boldsymbol{y}}.\Phi(\hat{\boldsymbol{y}}, \boldsymbol{a})$.

Conversely, if $\boldsymbol{a} \in \mathbb{N}^s$ is such that $\exists \hat{\boldsymbol{y}}.\Phi(\hat{\boldsymbol{y}}, \boldsymbol{a})$, then we take the corresponding values $\boldsymbol{b} \in \mathbb{N}^k$ of the eliminated variables $\hat{\boldsymbol{y}}$ and use the fact that NFAs $\mathcal{A}_i$ recognise the regular predicates from Φ, ensuring the existence of $\overline{\mathbf{w}}' \in [\![(\boldsymbol{b}, \boldsymbol{a})]\!]^{-1}$ accepted by NFAs $\mathcal{A}_i$ for all $i \in [1,m]$. From the definition of $\mathcal{A}_\Phi$ it follows that $\pi_{\boldsymbol{x}}(\overline{\mathbf{w}}')$ is accepted by $\mathcal{A}_\Phi$, and thus $\boldsymbol{a} \in [\![L(\mathcal{A}_\Phi)]\!]$. □

IV. Linearisation and elimination

The key ingredient in our quantifier elimination procedure for regular lin-exp systems is the process of linearisation of free variables. Let us first informally describe its main ideas.

A. Linearisation of the leading exponential term

Given a system $\Phi = (\varphi \wedge \theta)$, where 2^x is the leading term of the generalised ordering θ, we are going to construct an equisatisfiable over $\mathbb{N}$ regular lin-exp system $\Psi = (\psi \wedge \eta)$ such that η is θ without 2^x and ψ has the same free variables as φ, but x occurs in ψ only linearly. By Lemma 6, the successor of 2^x in θ is either 2^y or x so that y is the logarithmic variable for x, i.e., there is a subformula $R_\lambda(x, 2^y)$ in θ. Then, the $\mathbb{B}^s$-NFA $\mathcal{A}_\Phi$ associated with the system Φ accepts the words that can be described via the following pattern:

$$\begin{bmatrix}0\\0\\0\\\vdots\\0\end{bmatrix}^* \cdot \left(\begin{bmatrix}1\\1\\\mathbb{B}\\\vdots\\\mathbb{B}\end{bmatrix} \cup \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} \begin{bmatrix}0\\0\\0\\\vdots\\0\end{bmatrix}^* \begin{bmatrix}0\\1\\\mathbb{B}\\\vdots\\\mathbb{B}\end{bmatrix} \right) \cdot \begin{bmatrix}0\\0\\\mathbb{B}\\\vdots\\\mathbb{B}\end{bmatrix}^* \quad \begin{matrix}2^x\\2^y\\t_3\\\vdots\\t_s\end{matrix} \tag{8}$$

Here, the NFA $\mathcal{A}_\Phi$ reads digits from right to left, because the non-negative integers are represented in the lsd-first notation. The rightmost column shows the correspondence between the coordinates in tuples from $\mathbb{B}^s$ and the primitive terms over free variables of Φ. Observe that the length of the fragment enclosed in parentheses in (8) must be equal to $(x - y + 1)$, and the length of the blue part is exactly $(x - y)$. Suppose that in the NFA $\mathcal{A}_\Phi$ there is a transition $\tau = (\boldsymbol{p}, [1, \mathtt{a}, \mathbf{b}], \boldsymbol{q})$, which is the last transition in every accepting run (τ is an *accepting transition*). Here, $\boldsymbol{q}$ is a final state of $\mathcal{A}_\Phi$, and the first two coordinates of the labels correspond to 2^x and 2^y. Then, the term 2^x is replaced in Φ with a fresh eliminated variable $\hat{u}$ and excluded from the ordering θ resulting in a regular lin-exp system $\Psi = (\psi \wedge \eta)$ and an accepting transition $\tau' = (\boldsymbol{p}', \mathbf{c}, \boldsymbol{q}')$ with a final state $\boldsymbol{q}'$ of $\mathcal{A}_\Psi$. Then we consider two cases in each non-deterministic branch that correspond to the value of the bit $\mathtt{a}$ in the transition τ. If $\mathtt{a} = 1$, then we add $(x = y)$ to ψ and use $(\boldsymbol{p}, [1, \mathbf{b}], \boldsymbol{q})$ as a new accepting transition of $\mathcal{A}_\Psi$. Otherwise, if $\mathtt{a} = 0$, we use Lemma 2 to construct a finite set of pairs $C \subseteq \mathbb{N}^2$ to describe the lengths of all runs in $\mathcal{A}_\Psi$ from $\boldsymbol{q}'$ to $\boldsymbol{p}$ with transitions labelled with $\mathbf{0}$. After guessing a pair $(c-1, d) \in C$, we add the constraint $(x-y) \in \{c + d \cdot n : n \in \mathbb{N}\}$ to ψ in each non-deterministic branch, and further use τ' as a new accepting transition of the resulting system.

Algorithm 2 Linearisation of the leading exponential term in an ordered regular lin-exp system

Input: Φ : regular lin-exp system $(\varphi \wedge \theta)$ with a generalised ordering θ; 2^x is the leading term of θ and 2^y is the second largest exponential term of θ;
τ : transition $(\boldsymbol{p}, [1, \mathtt{a}, \mathbf{b}], \boldsymbol{q})$ from $Acc(\Phi)$
Output of branch β**:** a pair (Ψ_β, τ_β) of a regular lin-exp system Ψ_β, where x does not occur exponentiated, and a transition τ_β from $Acc(\Psi_\beta)$.

replace in Φ the term 2^x with a fresh variable u
$(\Psi = (\psi \wedge \eta), \tau' = \boldsymbol{p}' \xrightarrow{\mathbf{c}} \boldsymbol{q}') \leftarrow$ ELIMINATE u in (Φ, τ)
if $(\mathtt{a} = 1)$ **then**
  assert(η does not contain $R_\lambda(x, 2^y)$)
  $\psi \leftarrow \psi \wedge (x = y)$
else ▷ *when* $\mathtt{a} = 0$
  let n be the cardinality of Q, the set of states of $\mathcal{A}_\Psi$
  guess $(c, \boldsymbol{q}'') \leftarrow$ a pair from $[1, n^2] \times \{\star\} \cup [n, n^2] \times Q$
  $c' \leftarrow$ the minimal integer $z > 0$ s.t. $z - \lfloor log_2(z) \rfloor \geq c$
  if $(\boldsymbol{q}'' = \star)$ **then**
    assert($\mathbf{0}^{c-1}$-reachability of $\boldsymbol{p}$ from $\boldsymbol{q}'$ in $\mathcal{T}_\Psi$)
    if η contains $R_\lambda(x, 2^y)$ **then**
      assert($c' - \lfloor log_2(c') \rfloor = c$)
      if $c = 1$ **then guess** $c' \leftarrow$ an integer from $\{1, 2\}$
      $\psi \leftarrow \psi \wedge (x = c')$
    else
      $\psi \leftarrow \psi \wedge (x = y + c)$
  else ▷ *when* $\boldsymbol{q}''$ *is a state of* $\mathcal{A}_\Psi$
    $d \leftarrow$ the length of the shortest $\mathbf{0}^*$-loop in $\boldsymbol{q}''$
    assert($\mathbf{0}^{n-1}$-reachability of $\boldsymbol{q}''$ from $\boldsymbol{q}'$ and $\mathbf{0}^{c-n}$-reachability of $\boldsymbol{p}$ from $\boldsymbol{q}''$ in $\mathcal{T}_\Psi$)
    if η contains $R_\lambda(x, 2^y)$ **then**
      $\psi \leftarrow \psi \wedge (x \geq c')$
    else
      $\psi \leftarrow \psi \wedge (x \geq y + c)$
    guess $r \leftarrow$ a remainder from $[0, d-1]$
    $\psi \leftarrow \psi \wedge (d \mid x - (r + c)) \wedge (d \mid y - r)$
$\tau \leftarrow$ extend τ' to be a transition in $Acc(\psi \wedge \eta)$
return the pair $((\psi \wedge \eta), \tau)$

B. Regular lin-exp systems with an accepting transition

As it can be seen from the previous paragraph, during the linearisation process applied to a system Φ, it is important to control the last transition with a non-zero label in the accepting runs of the NFA $\mathcal{A}_\Phi = (Q, S, F, \Delta)$. Both Algorithms 2 and 3 operate with a pair $(\Phi, \boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q})$, of an ordered regular lin-exp system Φ and a transition to a final state $\boldsymbol{q}$ of $\mathcal{A}_\Phi$. To describe the main properties of these algorithms, we first define a set

$$Acc(\Phi) \coloneqq \{\boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q} \,:\, \boldsymbol{q} \in F, \mathbf{a} \neq \mathbf{0}\},$$

and for every transition $\boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q} \in Acc(\Phi)$ we introduce an NFA $\mathcal{A}_\Phi(\boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q}) = (Q \cup \{\hat{\boldsymbol{q}}\}, S, \{\hat{\boldsymbol{q}}\}, \Delta')$, where Δ' is the extension of the transition relation Δ with $\boldsymbol{p} \xrightarrow{\mathbf{a}} \hat{\boldsymbol{q}}$ and the loop $\hat{\boldsymbol{q}} \xrightarrow{\mathbf{0}} \hat{\boldsymbol{q}}$. The lemma below shows that the NFA $\mathcal{A}_\Phi(\boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q})$ recognises the subset of $[\![L(\mathcal{A}_\Phi)]\!]$ composed of $\boldsymbol{a} \in \mathbb{N}^s$ which are accepted by $\mathcal{A}_\Phi$ via a run of the form $\boldsymbol{q}_0 \xrightarrow{\mathbf{a}_0} \boldsymbol{q}_1 \xrightarrow{\mathbf{a}_1} \ldots \xrightarrow{\mathbf{a}_{t-1}} \boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q}$.

Lemma 9. For every regular lin-exp system Φ we have

$$[\![L(\mathcal{A}_\Phi)]\!] = \bigcup_{\tau \in Acc(\Phi)} [\![L(\mathcal{A}_\Phi(\tau))]\!].$$

Proof. If $\boldsymbol{a} \in [\![L(\mathcal{A}_\Phi)]\!]$, then there is a run in $\mathcal{A}_\Phi$ of the form

$$\boldsymbol{q}_0 \xrightarrow{\mathbf{a}_0} \boldsymbol{q}_1 \xrightarrow{\mathbf{a}_1} \dots \xrightarrow{\mathbf{a}_{t-1}} \boldsymbol{p} \xrightarrow{\mathbf{a}_t} \boldsymbol{q} \xrightarrow{\mathbf{0}} \dots \xrightarrow{\mathbf{0}} \boldsymbol{q}_f,$$

where $\boldsymbol{a} = [\![\mathbf{a}_0 ... \mathbf{a}_t]\!]$ with $\mathbf{a}_t \neq \mathbf{0}$ and $\boldsymbol{q}_f$ is a final state of $\mathcal{A}_\Phi$. By Lemma 8, we know that $\exists \hat{\boldsymbol{y}}.\Phi(\hat{\boldsymbol{y}}, \boldsymbol{a})$, and thus $\mathbf{a}_t$ must have 1 in the coordinate that corresponds to the leading exponential term of θ. Therefore, since $\boldsymbol{q}_f$ is $\mathbf{0}^*$-reachable from $\boldsymbol{q}$ in $\mathcal{T}_\Phi$, the state $\boldsymbol{q}$ is also a final state of $\mathcal{A}_\Phi$, the transition $\tau := \boldsymbol{p} \xrightarrow{\mathbf{a}_t} \boldsymbol{q}$ is in $Acc(\Phi)$ and the word $\mathbf{a}_0 ... \mathbf{a}_t$ is accepted by $\mathcal{A}_\Phi(\tau)$.

The converse direction is trivial because for every transition $\tau = \boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q}$ in $Acc(\Phi)$ the final state $\hat{q}$ of $\mathcal{A}_\Phi(\tau)$ is a copy of a final state $\boldsymbol{q}$ of $\mathcal{A}_\Phi$, to which we have directed a copy of τ. □

Whereas Lemma 8 provides a direct representation of the set $[\![L(\mathcal{A}_\Phi)]\!]$ via an existential formula, there is no such representation for the set $[\![L(\mathcal{A}_\Phi(\tau))]\!]$. For the convenience of subsequent proofs, we rewrite the fact that $\boldsymbol{a} \in [\![L(\mathcal{A}_\Phi(\tau))]\!]$ as $\exists \hat{\boldsymbol{y}}.(\Phi, \tau)(\hat{\boldsymbol{y}}, \boldsymbol{a})$ and the phrase *there exists* $a \in \mathbb{N}$ *such that* $(a, \boldsymbol{b}) \in [\![L(\mathcal{A}_\Phi(\tau))]\!]$ using the formula $\exists \hat{\boldsymbol{y}}.\exists x.(\Phi, \tau)(\hat{\boldsymbol{y}}, x, \boldsymbol{b})$. With this notation and Lemma 8, we can rewrite Lemma 9.

Lemma 10. For every regular lin-exp system Φ we have

$$\exists \hat{\boldsymbol{y}}.\Phi(\hat{\boldsymbol{y}}, \boldsymbol{x}) \iff \bigvee_{\tau \in Acc(\Phi)} \exists \hat{\boldsymbol{y}}.(\Phi, \tau)(\hat{\boldsymbol{y}}, \boldsymbol{x}). \tag{9}$$

In the informal descriptions from Section IV-A, we replaced the leading exponential term 2^x of the ordering θ with a fresh eliminated variable $\hat{u}$. This operation essentially performs two steps: replacement of 2^x with a free variable u and elimination of u. Let us now specialise the elimination of linearly occurring variables onto a pair (Φ, τ).

Lemma 11 (ELIMINATE:free variable x occurring only linearly in regular lin-exp system $\Phi = (\varphi \wedge \theta)$ for $\tau = \boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q}$ in $Acc(\Phi)$)**.** For a fresh eliminated variable $\hat{x}$, let us define

1: $\zeta \leftarrow$ the subsystem with x in θ
2: $\eta \leftarrow$ the ordering θ, where ζ is removed
3: $\psi \leftarrow (\varphi \wedge \zeta)$
4: replace in ψ all occurrences of x with $\hat{x}$

Then $\Psi = (\psi \wedge \eta)$ is an ordered regular lin-exp system and

$$\exists \hat{\boldsymbol{y}}.\exists x.(\Phi, \tau)(\hat{\boldsymbol{y}}, x, \boldsymbol{z}) \Leftrightarrow \bigvee_{\tau' \in A} \exists \hat{\boldsymbol{y}}.\exists \hat{x}.(\Psi, \tau')(\hat{\boldsymbol{y}}, \hat{x}, \boldsymbol{z}), \tag{10}$$

where for $\mathbf{b} = \pi_{\boldsymbol{z}}(\mathbf{a})$ the set A is defined as

E1 $A := \{\boldsymbol{p} \xrightarrow{\mathbf{b}} \boldsymbol{q}\}$ if $\mathbf{b} \neq \mathbf{0}$. Otherwise, if $\mathbf{b} = \mathbf{0}$, then
E2 A comprises transitions $\tau' = \boldsymbol{p}' \xrightarrow{\mathbf{c}} \boldsymbol{q}'$ from $Acc(\Psi)$ such that $\boldsymbol{p}$ is $\mathbf{0}^*$-reachable from $\boldsymbol{q}'$ in $\mathcal{T}_\Psi$.

Proof. Clearly, Ψ is an ordered regular lin-exp system. Also observe that the NFAs $\mathcal{A}_\Phi$ and $\mathcal{A}_\Psi$ have the same sets of states because the only difference between the systems Φ and Ψ is that the variable x has been replaced with $\hat{x}$. Now consider the case when $\mathbf{a} = [\mathtt{a}, \mathbf{b}]$ and $\mathbf{b} \neq \mathbf{0}$. Suppose that there is $a \in \mathbb{N}$ such that $(a, \boldsymbol{b}) \in [\![L(\mathcal{A}_\Phi(\tau))]\!]$. Then, in $\mathcal{A}_\Phi$ there is a run

$$\boldsymbol{q}_0 \xrightarrow{\begin{bmatrix}\mathtt{a}_0\\ \mathbf{b}_0\end{bmatrix}} \boldsymbol{q}_1 \to \dots \xrightarrow{\begin{bmatrix}\mathtt{a}_{k-1}\\ \mathbf{b}_{k-1}\end{bmatrix}} \boldsymbol{p} \xrightarrow{\begin{bmatrix}\mathtt{a}\\ \mathbf{b}\end{bmatrix}} \boldsymbol{q},$$

where $a = [\![\mathtt{a}_0 ... \mathtt{a}_{k-1} \mathtt{a}]\!]$. Since $\boldsymbol{q}$ is a final state of $\mathcal{A}_\Phi$, then due to $\mathbf{b} \neq \mathbf{0}$, $\boldsymbol{q}$ will be a final state of $\mathcal{A}_\Psi$. Therefore, we have $\boldsymbol{b} \in [\![L(\mathcal{A}_\Psi(\tau'))]\!]$, and the equivalence (10) is proved from left to right in the case E1. For the converse direction, the fact that $\boldsymbol{b} \in [\![L(\mathcal{A}_\Psi(\tau'))]\!]$ means that there is a run in $\mathcal{A}_\Phi$ of the form

$$\boldsymbol{q}_0 \xrightarrow{\begin{bmatrix}\mathtt{a}_0\\ \mathbf{b}_0\end{bmatrix}} \boldsymbol{q}_1 \to \dots \xrightarrow{\begin{bmatrix}\mathtt{a}_{k-1}\\ \mathbf{b}_{k-1}\end{bmatrix}} \boldsymbol{p} \xrightarrow{\begin{bmatrix}\mathtt{a}_k\\ \mathbf{b}\end{bmatrix}} \boldsymbol{q} \xrightarrow{\begin{bmatrix}\mathtt{a}_{k+1}\\ \mathbf{0}\end{bmatrix}} \dots \xrightarrow{\begin{bmatrix}\mathtt{a}_t\\ \mathbf{0}\end{bmatrix}} \boldsymbol{q}_f$$

for a final state $\boldsymbol{q}_f$ of $\mathcal{A}_\Phi$. But we know that there is a transition $\tau = \boldsymbol{p} \xrightarrow{\mathbf{a}} \boldsymbol{q}$ in $Acc(\Phi)$, and thus for the integer $a = [\![\mathtt{a}_0 ... \mathtt{a}_{k-1} \mathtt{a}]\!]$ we have $\exists \hat{\boldsymbol{y}}.(\Phi, \tau)(\hat{\boldsymbol{y}}, a, \boldsymbol{b})$, and the equivalence (10) is proved.

Now let $\mathbf{a} = [\mathtt{a}, \mathbf{b}]$ and $\mathbf{b} = \mathbf{0}$. Then there is $a \in \mathbb{N}$ such that $(a, \boldsymbol{b}) \in [\![L(\mathcal{A}_\Phi(\tau))]\!]$ if there is a run in $\mathcal{A}_\Phi$ of the form

$$\boldsymbol{q}_0 \xrightarrow{\begin{bmatrix}\mathtt{a}_0\\ \mathbf{b}_0\end{bmatrix}} \boldsymbol{q}_1 \to \dots \to \boldsymbol{p}' \xrightarrow{\begin{bmatrix}\mathtt{a}_k\\ \mathbf{b}_k\end{bmatrix}} \boldsymbol{q}' \xrightarrow{\begin{bmatrix}\mathtt{a}_{k+1}\\ \mathbf{0}\end{bmatrix}} \dots \to \boldsymbol{p} \xrightarrow{\begin{bmatrix}\mathtt{a}\\ \mathbf{0}\end{bmatrix}} \boldsymbol{q}, \tag{11}$$

where $\mathbf{b}_k \neq \mathbf{0}$, $a = [\![\mathtt{a}_0 ... \mathtt{a}_k ... \mathtt{a}]\!]$, and $\mathtt{a} = 1$. Since $\boldsymbol{q}$ is a final state of $\mathcal{A}_\Phi$, the state $\boldsymbol{q}'$ is a final state of $\mathcal{A}_\Psi$, $\tau' = \boldsymbol{p}' \xrightarrow{\mathbf{b}_k} \boldsymbol{q}'$ belongs to $Acc(\Psi)$ and we have $\boldsymbol{b} = [\![\mathbf{b}_0 ... \mathbf{b}_k]\!] \in [\![L(\mathcal{A}_\Psi(\tau'))]\!]$. Conversely, if we have $\boldsymbol{b} \in [\![L(\mathcal{A}_\Psi(\tau'))]\!]$, then we use the fact that $\boldsymbol{p}$ is $\mathbf{0}^*$-reachable from $\boldsymbol{q}'$ in $\mathcal{T}_\Phi$, which ensures the existence of a run in $\mathcal{A}_\Phi$ of the form (11). Since τ belongs to $Acc(\Phi)$, for $a = [\![\mathtt{a}_0 ... \mathtt{a}]\!]$ we obtain $\exists \hat{\boldsymbol{y}}.(\Phi, \tau)(\hat{\boldsymbol{y}}, a, \boldsymbol{b})$, and the equivalence (10) is proved for the case E2. □

Lemma 11 will be used as a non-deterministic procedure which takes on input a pair (Φ, τ) and returns in each branch a pair (Ψ, τ') for some $\tau' \in A$. Non-determinism is used in the case E2 to guess $\tau' \in A$ and to check $\mathbf{0}^*$-reachability of $\boldsymbol{p}$ from $\boldsymbol{q}'$ in $\mathcal{T}_\Psi$ using Lemma 7. Importantly, before application of Lemma 11 in line 2, the replacement performed in $\Phi = (\varphi \wedge \theta)$ by line 1 moves the subsystem $P_2(u)$ from θ to φ.

During its execution, Algorithm 2 supplements regular lin-exp system $\Psi = (\psi \wedge \eta)$, which is obtained using Lemma 11 in line 2, with new equalities $(x = c')$ and $(x = y + c)$ in lines 5, 15, and 17, or inequalities $(x \geq c')$ and $(x \geq y + c)$ in lines 22 and 24 together with the divisibility $(d \mid x - (r + c))$ in line 26. For (in)equalities we use basic NFAs from Section II-A and fresh eliminated variables: $(x \geq c') \Leftrightarrow \exists \hat{y}_1 (\hat{y}_1 = c' \wedge \hat{y} \leq x)$ and $(x \geq y + c) \Leftrightarrow \exists \hat{y}_1 \exists \hat{y}_2 (\hat{y}_1 \leq x \wedge y + \hat{y}_2 = \hat{y}_1 \wedge \hat{y}_2 = c)$.

Introduction of new constraints (denote them by ψ') changes the set of states of the NFA associated with the system Ψ, and the transition $\boldsymbol{p}' \xrightarrow{\mathbf{c}} \boldsymbol{q}' \in Acc(\Psi)$ must be updated. To do this, in line 27 we guess (using Lemma 7) $\boldsymbol{p}'' \xrightarrow{\mathbf{c}} \boldsymbol{q}''$ in $Acc(\psi')$ and extend the state $\boldsymbol{p}'$ with $\boldsymbol{p}''$ and $\boldsymbol{q}'$ with $\boldsymbol{q}''$. If there is no such transition in $Acc(\psi')$, the execution of the branch aborts, similarly to the unsuccessful assertions that we use in the pseudocode. Observe that, by Remark 1, for (in)equalities this extension is unique and can be performed deterministically.

Another remark concerns line 19, where we apply Lemma 2. By $\mathbf{0}^*$*-loop in* $\boldsymbol{q}$ we call a run in $\mathcal{T}_\Phi$ from the state $\boldsymbol{q}$ to $\boldsymbol{q}$ over a word from $\mathbf{0}^*$. Combining the binary search with Remark 2, we obtain the following lemma.

Lemma 12. Let Φ be a regular lin-exp system, and let $\boldsymbol{q}$ be a state of the labelled transition system $\mathcal{T}_\Phi$ associated with Φ. Then the length of the shortest $\mathbf{0}^*$-loop in $\boldsymbol{q}$ can be computed in space polynomial in $|\Phi|$.

We can now describe the main property of Algorithm 2.

Lemma 13 (LINEARISE: the leading exponential term 2^x in a regular lin-exp system $\Phi=(\varphi\wedge\theta)$ for $\tau=\boldsymbol{p}\xrightarrow{[1,\mathtt{a},\mathbf{b}]}\boldsymbol{q}$ in $Acc(\Phi)$)**.** Let the successor of 2^x in θ be either 2^y or x so that y is the logarithmic variable for x. If (Ψ_β,τ_β) is the output of a branch $\beta\in B$ of Algorithm 2 on input (Φ,τ), then we have

$$\exists\hat{\boldsymbol{y}}.(\Phi,\tau)(\hat{\boldsymbol{y}},2^x,x,2^y,y,\boldsymbol{z}) \tag{12}$$

$$\iff \bigvee_{\beta\in B}\exists\hat{\boldsymbol{y}}_\beta.(\Psi_\beta,\tau_\beta)(\hat{\boldsymbol{y}}_\beta,x,2^y,y,\boldsymbol{z}). \tag{13}$$

Proof. The fact that $(2^a,a,2^b,b,\boldsymbol{c})\in[\![L(\mathcal{A}_\Phi(\tau))]\!]$ means that there is a run in $\mathcal{A}_\Phi$ of the form

$$\boldsymbol{q}\xleftarrow{\begin{bmatrix}1\\1\\\mathbf{b}\end{bmatrix}}\boldsymbol{p}\xleftarrow{\begin{bmatrix}0\\0\\\mathbf{b}_t\end{bmatrix}}\dots\xleftarrow{\begin{bmatrix}0\\0\\\mathbf{b}_0\end{bmatrix}}\boldsymbol{q}_0 \tag{14}$$

or of the form

$$\boldsymbol{q}\xleftarrow{\begin{bmatrix}1\\0\\\mathbf{0}\end{bmatrix}}\boldsymbol{p}\xleftarrow{\begin{bmatrix}0\\0\\\mathbf{0}\end{bmatrix}}\dots\xleftarrow{\begin{bmatrix}0\\0\\\mathbf{0}\end{bmatrix}}\boldsymbol{q}'\xleftarrow{\begin{bmatrix}0\\1\\\mathbf{b}\end{bmatrix}}\boldsymbol{p}'\dots\xleftarrow{\begin{bmatrix}0\\0\\\mathbf{b}_0\end{bmatrix}}\boldsymbol{q}_0, \tag{15}$$

where τ is the last transition in these runs, the first coordinates of the labels represent 2^a and 2^b, respectively, and the blue part has length $a-b$. On the other hand, after elimination of the variable u in $\exists\hat{\boldsymbol{y}}.\exists u.(\Phi,\tau)(\hat{\boldsymbol{y}},u,x,2^y,y,\boldsymbol{z})$ via Lemma 11, we obtain a pair (Ψ,τ'), where τ' is equal to either $(\boldsymbol{p},[1,\mathbf{b}],\boldsymbol{q})$ for the transition τ from (14), or $(\boldsymbol{p}',[1,\mathbf{b}],\boldsymbol{q}')$ for the case (15). The equivalence (10) gives us the proof of the implication from (12) to (13) if B is the set A from Lemma 11. Let us add new constraints to Ψ in every non-deterministic branch so that for every tuple $(2^a,a,2^b,b,\boldsymbol{c})$ that satisfies (12), the tuple $(a,2^b,b,\boldsymbol{c})$ will still satisfy (13), but now the converse direction will also be true.

The case of τ from (14) is easy: we add to Ψ the constraint $(x=y)$. Indeed, if the tuple $(a,2^b,b,\boldsymbol{c})$ satisfies (13) and we have $(a=b)$, then there is a run of the form (14) without the red row. Since Ψ contains the constraints $P_2(\hat{u})$ and $\hat{u}\geq 2^y$, then every transition in this run before τ' can be supplemented with $\mathtt{0}$ in the coordinate that corresponds to $\hat{u}$, and since the transition τ belongs to $Acc(\Phi)$, we can replace τ' with τ. This results in a run in $\mathcal{A}_\Phi$ that recognises the tuple $(2^a,a,2^b,b,\boldsymbol{c})$, and thus this tuple satisfies (12).

Now consider τ from (15). To rewrite $\mathbf{0}^{a-b-1}$-reachability of $\boldsymbol{p}$ from $\boldsymbol{q}'$ in $\mathcal{T}_\Psi=(Q,\mathbb{B}^{s-1},\Delta)$, we apply Lemma 2 to the unary NFA $\mathcal{C}=(Q,\{\boldsymbol{q}'\},\{\boldsymbol{p}\},\Delta')$, where $(\boldsymbol{q}_1,\mathbf{0},\boldsymbol{q}_2)\in\Delta'$ if and only if $(\boldsymbol{q}_1,\mathbf{0},\boldsymbol{q}_2)\in\Delta$. Let the set of pairs $C=C_1\cup C_2$ be the result of application of Lemma 2 to $\mathcal{C}$. Then we have

$$a-b\in\{c+d\cdot n:(c-1,d)\in C,\ \text{and}\ n\in\mathbb{N}\}. \tag{16}$$

Since C completely describes the lengths of runs over words in $\mathbf{0}^*$ from $\boldsymbol{q}'$ to $\boldsymbol{p}$, then there is a pair (c,d) that describes the length of the run from $\boldsymbol{q}'$ to $\boldsymbol{p}$ in (15). For the converse, observe that for $\tau'\in A$ there is a particular NFA $\mathcal{C}_{\tau'}$ and a set $C_{\tau'}$. Again, the system Ψ contains the constraints $P_2(\hat{u})$ and $\hat{u}\geq 2^y$, and there is a transition $\tau=(\boldsymbol{p},[1,0,\mathbf{0}],\boldsymbol{q})$ in $Acc(\Phi)$. Therefore, if for $\tau'\in Acc(\Psi)$ a tuple $(a,2^b,b,\boldsymbol{c})$ is recognised by $\mathcal{A}_\Psi(\tau')$ and satisfies the constraint (16) for $C_{\tau'}$, then, from a run in $\mathcal{A}_\Psi(\tau')$ that recognises $(a,2^b,b,\boldsymbol{c})$ we can reconstruct a run in $\mathcal{A}_\Phi(\tau)$ that recognises $(2^a,a,2^b,b,\boldsymbol{c})$.

Summarizing and simultaneously correlating with the pseudocode of Algorithm 2, we obtain the following equivalence. For the transition $\tau=(\boldsymbol{p},[1,\mathtt{a},\mathbf{b}],\boldsymbol{q})$ in $Acc(\Phi)$ and a tuple $(a,b,\boldsymbol{c})$ we have $\exists\hat{\boldsymbol{y}}.(\Phi,\tau)(\hat{\boldsymbol{y}},2^a,a,2^b,b,\boldsymbol{c})$ if and only if one of the following two alternatives holds:

- A (lines 4–5) $\mathtt{a}=1$, $(a,2^b,b,\boldsymbol{c})\in[\![L(\mathcal{A}_\Psi(\tau'))]\!]$ for the transition $\tau'=(\boldsymbol{p},[1,\mathbf{b}],\boldsymbol{q})$, and $a=b$.
- B (lines 8–26) $\mathtt{a}=0$, there is a transition $\tau'\in Acc(\Psi)$ with $(a,2^b,b,\boldsymbol{c})\in[\![L(\mathcal{A}_\Psi(\tau'))]\!]$ and for the set $C_{\tau'}=C_1\cup C_2$ constructed for the NFA $\mathcal{C}_{\tau'}$ and $\#Q=n$ either
  - B.1 (lines 11–17) there is an integer $c\in[1,n^2]$ such that $(c-1,0)\in C_1$ and $(a=b+c)$; or
  - B.2 (lines 19–26) there is an integer $c\in[n,n^2]$ and a state $\boldsymbol{q}''\in Q$ such that $(c-1,d)\in C_2$, where d is the length of the shortest loop in $\mathcal{C}_{\tau'}$ that can be done in $\boldsymbol{q}''$, and we have $(a\geq b+c)\wedge(d\mid a-(b+c))$.

Lines 1 and 2 apply Lemma 11 to produce a pair (Ψ,τ') of an ordered regular lin-exp system $\Psi=(\psi\wedge\eta)$ and a transition τ' from $Acc(\Psi)$. Our aim is to show that the updates of system Ψ performed by lines 3–27 indeed introduce the auxiliary (in)equalities and divisibilities from A and B.

If $(\mathtt{a}=1)$, from E1 we obtain in line 2 $\tau'=(\boldsymbol{p},[1,\mathbf{b}],\boldsymbol{q})$. When x is the successor of 2^x in θ and y is the logarithmic variable for x (*the logarithmic case*), then the alternative A would imply $R_\lambda(a,2^a)$, and we rule out this case in line 4. Otherwise, if 2^y is the successor of 2^x in θ (*the general case*), then in line 5 we define $\Psi_\beta=(\psi\wedge(x=y)\wedge\eta)$, and in line 27 the transition τ_β is defined as (the unique) extension of τ'. Thus, A holds if and only if $(a,2^b,b,\boldsymbol{c})\in[\![L(\mathcal{A}_{\Psi_\beta}(\tau_\beta))]\!]$.

When $(\mathtt{a}=0)$, the choice between the alternatives B.1 and B.2 is done in line 8. If we have guessed a pair $(c,\star)$ for $c\in[1,n^2]$, then we check $\mathbf{0}^{c-1}$-reachability of the state $\boldsymbol{p}$ from $\boldsymbol{q}'$ in $\mathcal{T}_\Psi$ and, if successful, add to the system Ψ the equality $(x=y+c)$ to rewrite B.1. This is explicitly done in line 17 for the general case. However, in the logarithmic case we actually have the equality $a-\lfloor log_2(a)\rfloor=c$, and we first compute (in line 9) the minimal z such that $z-\lfloor log_2(z)\rfloor\geq c$. If $c=1$, we take $c'=1$; otherwise guess an integer $c'\in[c,2c]$ and check that the inequality is true for $z=c'$ and is false for $z=c'-1$. The only case when the equation $x-\lfloor log_2(x)\rfloor=c$ has two solutions is $c=1$: we have either $x=1$ or $x=2$ (a solution is guessed in line 14). Otherwise, if there is any solution, then there is only one solution due to the growth of the functions $f(x)=x$ and $g(x)=\lfloor log_2(x)\rfloor+c$. The system Ψ is now supplemented with the equality $(x=c')$. Again, after the usual extension of the transition τ' we obtain that B.1 holds if and only if $(a,2^b,b,\boldsymbol{c})\in[\![L(\mathcal{A}_{\Psi_\beta}(\tau_\beta))]\!]$.

Now consider the case when we have guessed in line 8 a pair $(c, \boldsymbol{q}'')$ for an integer $c \in [n, n^2]$ and a state $\boldsymbol{q}'' \in Q$. Using Lemma 12, we compute d in line 19 and then check $\mathbf{0}^{n-1}$-reachability of $\boldsymbol{q}''$ from $\boldsymbol{q}'$ and $\mathbf{0}^{c-n}$-reachability of $\boldsymbol{q}$ from $\boldsymbol{q}''$ in $\mathcal{T}_\Psi$ using Remark 2. To rewrite B.2, we must add to Ψ the inequality $(x \geq y{+}c)$ and divisibility $(d \mid x{-}(y{+}c))$. Again, in the logarithmic case we rewrite the inequality using the equivalent one $(x \geq c')$. By guessing a remainder of y modulo d in line 25, the divisibility $(d \mid x{-}(y{+}c))$ is split into $(d \mid x-(r+c)) \wedge (d \mid (y-r))$ and added to Ψ in line 26. In line 27, τ_β is defined as an extension of τ' using the simple NFAs for (in)equalities and divisibilities added to Ψ. Now B.2 holds if and only if $(a, 2^b, b, \boldsymbol{c}) \in [\![L(\mathcal{A}_{\Psi_\beta}(\tau_\beta))]\!]$. □

V. Master Procedure

This section describes Algorithm 3, which combines Lemmas 11 and 13 to decide solvability in $\mathbb{N}$ of a regular lin-exp system φ. After a simple elimination of the variables occurring only linearly in φ (line 1) and specification of a generalised ordering θ (line 2), there are two cases depending on the two alternatives for θ described in Lemma 6. In both cases, Algorithm 3 performs linearisation and elimination of the variable x (lines 7–8). However, if the inequalities in θ have the form O2, we also eliminate the logarithmic variable y, which is handled differently from the free ones. Namely, after line 8, the system Ψ will have only exponential occurrences of y except for the divisibility $(d \mid y-r)$, which was added by line 7 (see line 26 of Algorithm 2). Now consider the following equivalences:

$$\begin{aligned} &\exists y.(\Psi(2^y, \boldsymbol{z}) \wedge (d \mid y-r)) \iff \\ &\exists y.\exists y'.(\Psi(y', \boldsymbol{z}) \wedge (y' = 2^y) \wedge (d \mid y-r)) \iff \\ &\exists y'.(\Psi(y', \boldsymbol{z}) \wedge \exists y.((y' = 2^y) \wedge (d \mid y-r))). \end{aligned} \tag{17}$$

The variable y' now occurs only linearly in Ψ, and the extra existential formula $\exists y.((y' = 2^y) \wedge (d \mid y-r))$ represents a regular predicate on y', which is recognised by a simple NFA. This predicate will be denoted by $R_{d,r}$, and in our system it will be represented via a pair of integers (d, r) encoded in binary. The corresponding NFA has $d+1$ states $\{q_0, \ldots, q_d\}$; it has transitions $q_i \xrightarrow{0} q_{(i \bmod d)+1}$ for every $i \in [1, d]$, and also $q_{r+1} \xrightarrow{1} q_0$ with a loop $q_0 \xrightarrow{0} q_0$. Its initial state is q_1 and its final state is q_0. For example, consider Fig. 2.

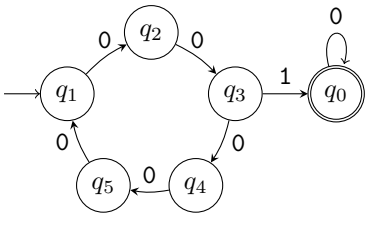


Fig. 2: NFA for $R_{5,2}(y') := \exists y.((y' = 2^y) \wedge (5 \mid y-2))$

It is clear that the NFA that recognises $R_{d,r}$ has the same properties as the simple NFAs from Remark 1. After these preliminary remarks, let us proceed to the proof of correctness of Algorithm 3.

Algorithm 3 Solvability in $\mathbb{N}$ of regular lin-exp systems

Input: $\varphi(2^{x_1}, x_1, \ldots, 2^{x_n}, x_n)$: regular lin-exp system.
Output: $\top$ if φ is satisfiable in $\mathbb{N}$, and otherwise $\bot$.

1: replace in φ each variable x occurring only linearly with $\hat{x}$
2: $\Phi = (\varphi \wedge \theta) \leftarrow$ ORDER trivial terms over free variables of φ
3: **guess** $\tau \leftarrow$ a transition from $Acc(\Phi)$
4: **while** φ contains an exponentiated variable **do**
5: $\quad 2^x \leftarrow$ leading exponential term of θ
6: $\quad 2^y \leftarrow$ second largest exponential term in θ
7: $\quad (\Psi, \tau') \leftarrow$ LINEARISE the term 2^x in (Φ, τ)
8: $\quad (\Phi, \tau) \leftarrow$ ELIMINATE the variable x in (Ψ, τ')
9: $\quad$ **if** the subformula $R_\lambda(x, 2^y)$ occurs in Ψ **then**
10: $\quad\quad$ **let** $(d \mid y-r)$ be the divisibility with y in Φ
11: $\quad\quad$ replace in Φ the term 2^y with a fresh variable y'
12: $\quad\quad \Phi \leftarrow \Phi \wedge R_{d,r}(y')$
13: $\quad\quad \tau \leftarrow$ extend τ to be a transition in $Acc(\Phi)$
14: $\quad\quad (\Psi, \tau') \leftarrow$ ELIMINATE the variable y' in (Φ, τ)
15: $\quad\quad (\Phi, \tau) \leftarrow$ ELIMINATE the variable y in (Ψ, τ')
16: **return** $(\tau = \boldsymbol{q}_0 \xrightarrow{[1,0]} \boldsymbol{q}$ for an initial state $\boldsymbol{q}_0$ of $\mathcal{A}_\Phi)$

Lemma 14. Let $\varphi(2^{x_1}, x_1, \ldots, 2^{x_n}, x_n)$ be the regular lin-exp system given on input to Algorithm 3. Then, φ has solutions in $\mathbb{N}$ if and only if Algorithm 3 returns $\top$.

Proof. The first lines are preparatory: line 1 replaces each free variable x occurring only linearly in φ with a fresh eliminated variable $\hat{x}$, and line 2 applies Lemma 5 to specify a generalised ordering θ for φ. For simplicity, assume that there are still n free variables in $\Phi = (\varphi \wedge \theta)$. In line 3, we guess a transition τ from $Acc(\Phi)$ in order to work with the pair (Φ, τ) relying on Lemma 10. The next claim is a simple observation, which is easy to prove. It ensures that we can always call Algorithm 2 in line 7 and linearise logarithmic variables in lines 10–13.

Claim 1. For $i \in [1, n]$, the i-th iteration of the **while** loop in line 4 of Algorithm 3 starts with the variables (Φ, τ) storing a regular lin-exp system $(\varphi \wedge \theta)$ and a transition in $\mathcal{A}_\Phi$, where **1**. Φ has no linear occurrences of logarithmic variables; **2**. θ is a generalised ordering for φ; **3**. τ is a transition from $Acc(\Phi)$.

Now, applying Lemma 10, we obtain the equivalence

$$\exists \hat{\boldsymbol{y}}.\Phi(\hat{\boldsymbol{y}}, \boldsymbol{x}) \iff \bigvee_{\tau \in Acc(\Phi)} \exists \hat{\boldsymbol{y}}.(\Phi, \tau)(\hat{\boldsymbol{y}}, \boldsymbol{x}). \tag{18}$$

Each non-deterministic branch works with its particular pair (Φ, τ), and every iteration of the **while** loop updates the values of the variables (Φ, τ). Denote by B_i the set of all non-deterministic branches after i iterations of this loop. Each branch is represented via a sequence of guesses of transitions and integers (encoded in binary). Every branch $\beta_{i+1} \in B_{i+1}$ can be decomposed as $\beta_{i+1} = \beta_i \beta'$ for some $\beta_i \in B_i$. Our aim now is to prove that for every $i \in [0, n]$ we have

$$\begin{aligned} &\exists \hat{\boldsymbol{y}}.\exists \boldsymbol{x}.\exists \boldsymbol{y}.(\Phi, \tau)(\hat{\boldsymbol{y}}, 2^{x_1}, x_1, ..., 2^{x_i}, x_i, 2^{y_1}, y_1, ..., 2^{y_j}, y_j, \boldsymbol{z}) \\ &\qquad \iff \bigvee_{\beta \in B_i} \exists \hat{\boldsymbol{y}}_{\boldsymbol{\beta}}.(\Phi_\beta, \tau_\beta)(\hat{\boldsymbol{y}}_{\boldsymbol{\beta}}, \boldsymbol{z}), \end{aligned}$$

where $\boldsymbol{x} = x_1, ..., x_i$ are free variables, $\boldsymbol{y} = y_1, ..., y_j$ for $j \leq i$

are logarithmic variables for $\boldsymbol{x}$, and $\boldsymbol{z}$ comprises the remaining variables of Φ. We prove this equivalence by induction on i.

The base case $i = 0$ is trivial: the formula $\exists\hat{\boldsymbol{y}}.(\Phi,\tau)(\hat{\boldsymbol{y}},\boldsymbol{x})$ remains unchanged. Now suppose that the equivalence is true for some $i-1 \geq 0$. Let us fix some branch $\beta \in B_{i-1}$. By Lemma 13, after line 7 we obtain

$$\begin{aligned}&\exists\hat{\boldsymbol{y}}_\beta.(\Phi_\beta,\tau_\beta)(\hat{\boldsymbol{y}}_\beta,2^{x_i},x_i,2^y,y,\boldsymbol{z})\\ \iff &\bigvee_{\gamma\in B_\beta}\exists\hat{\boldsymbol{y}}_\gamma.(\Psi_\gamma,\tau_\gamma)(\hat{\boldsymbol{y}}_\gamma,x_i,2^y,y,\boldsymbol{z}),\end{aligned}\tag{19}$$

where y is either x_{i+1} or the logarithmic variable y_{j+1} for x_i. Line 8 eliminates the variable x_i in each system Ψ_γ for the transition τ_γ. Applying Lemma 11, we obtain a regular lin-exp system Ψ'_γ such that

$$\begin{aligned}&\exists\hat{\boldsymbol{y}}_\gamma.\exists x_i.(\Psi_\gamma,\tau_\gamma)(\hat{\boldsymbol{y}}_\gamma,x_i,2^y,y,\boldsymbol{z})\\ \iff &\bigvee_{\tau'\in A_\gamma}\exists\hat{\boldsymbol{y}}_\gamma.\exists\hat{x}_i.(\Psi'_\gamma,\tau')(\hat{\boldsymbol{y}}_\gamma,\hat{x}_i,2^y,y,\boldsymbol{z}).\end{aligned}\tag{20}$$

Induction hypothesis combined with the equivalences (19) and (20) completes the proof of the case when the inequalities in the ordering of Φ_β have the form $(2^{x_i}\geq 2^{x_{i+1}}\geq\theta')$. Indeed, for $\beta' = \gamma\tau'$ we have $\Phi_{\beta\beta'} \coloneqq \Psi'_\gamma$ and $\tau_{\beta\beta'} \coloneqq \tau'$.

Now consider the case when the inequalities in the ordering of Φ_β have the form $(2^{x_i}\geq x_i\geq 2^{y_j}\geq\theta')$, where the variable y_j is logarithmic for x_i. By Claim 1, 2^{y_j} does not occur linearly in Φ_β. Hence, its only linear occurrence of Ψ'_γ comes from the divisibility $(d_\gamma \mid y_j - r_\gamma)$. We apply (17) to obtain

$$\begin{aligned}&\exists\hat{\boldsymbol{y}}_\gamma.\exists\hat{x}_i.\exists y_j.(\Psi'_\gamma,\tau')(\hat{\boldsymbol{y}}_\gamma,\hat{x}_i,2^{y_j},y_j,\boldsymbol{z})\\ \iff\ &\exists\hat{\boldsymbol{y}}_\gamma.\exists\hat{x}_i.\exists y'_j.\exists y_j.(\Psi''_\gamma,\tau'')(\hat{\boldsymbol{y}}_\gamma,\hat{x}_i,y'_j,y_j,\boldsymbol{z}),\end{aligned}\tag{21}$$

where Ψ''_γ is defined as the conjunction of Ψ'_γ, where 2^{y_j} is replaced with y'_j, and $R_{d_\gamma,r_\gamma}(y'_j)$. The variable y_j now only occurs in $(d_\gamma \mid y_j - r_\gamma)$, and we did not excluded this constraint from Ψ''_γ for the presentational convenience only. The transition τ'' is the extension of τ' obtained in line 13. It remains to apply Lemma 11 twice. In line 14 we eliminate y'_j in Ψ''_γ for $\tau' \in Acc(\Psi''_\gamma)$. In the next line we eliminate the variable y_j. This elimination actually only decrements the dimension of the NFA associated with the system. By Lemma 11, we obtain

$$\begin{aligned}&\exists\hat{\boldsymbol{y}}_\gamma.\exists\hat{x}_i.\exists y'_j.\exists y_j.(\Psi''_\gamma,\tau'')(\hat{\boldsymbol{y}}_\gamma,\hat{x}_i,y'_j,y_j,\boldsymbol{z})\\ &\iff \bigvee_{\delta\in B_{\beta'}}\exists\hat{\boldsymbol{y}}_\delta.(\Psi_\delta,\tau_\delta)(\hat{\boldsymbol{y}}_\delta,\boldsymbol{z}),\end{aligned}\tag{22}$$

where $\hat{\boldsymbol{y}}_\delta$ comprises the variables $\hat{\boldsymbol{y}}_\gamma,\hat{x}_i,\hat{y}'_j,\hat{y}_j$. It remains to notice that for $\beta'' = \beta'\delta$ we have $\Phi_{\beta\beta''} \coloneqq \Psi_\delta$ and $\tau_{\beta\beta''} \coloneqq \tau_\delta$.

After n iterations of the **while** loop, we obtain

$$\begin{aligned}&\exists\hat{\boldsymbol{y}}.\exists x_1...\exists x_n.\exists y_1...\exists y_l.\big[\\ &\quad(\Phi,\tau)(\hat{\boldsymbol{y}},2^{x_1},x_1,...,2^{x_n},x_n,2^{y_1},y_1,...,2^{y_l},y_l,2^{x_0},x_0)\big]\\ &\qquad\iff \bigvee_{\beta\in B_n}\exists\hat{\boldsymbol{y}}_{\boldsymbol{\beta}}.(\Phi_\beta,\tau_\beta)(\hat{\boldsymbol{y}}_{\boldsymbol{\beta}},2^{x_0},x_0).\end{aligned}$$

Here, the only coordinates of the NFA $\mathcal{A}_{\Phi_\beta}$ associated with the system $\Phi_\beta = (\varphi_\beta,\theta_\beta)$ correspond to the terms 2^{x_0} and x_0, and the ordering θ_β now has the form $(2^{x_0}\geq x_0 = 0)$. From the definition of the final states of $\mathcal{A}_{\Phi_\beta}$, it follows that in order to check that $(1,0) \in [\![L(\mathcal{A}_{\Phi_\beta}(\tau_\beta))]\!]$, we must check that τ_β is a transition of the form $(\boldsymbol{q}_0,[1,0],\boldsymbol{q})$ in $\mathcal{A}_{\Phi_\beta}$ for some initial state $\boldsymbol{q}_0$. The evaluation of this condition is returned as the output of the non-deterministic branch β. □

To establish the computational complexity of Algorithm 3, we estimate the growth of the size the pair of a regular lin-exp system and a transition stored in the variables (Φ,τ). Recall that we represent the divisibility constraint $(d \mid x - r)$ and the predicate $R_{d,r}$ using the binary expansions of d and r, and that a state $\boldsymbol{q}$ of $\mathcal{A}_\Phi = (Q,S,F,\Delta)$ stores a tuple from Q, where the states that correspond to the automata for these predicates are represented via the binary representations of their numbers. For this reason, $|\tau| \leq |\Phi| + 4\cdot n + |\Phi| \leq 6\cdot|\Phi|$, and we are now going to consider only the systems in Φ.

Lemma 15. Let Φ_i be the regular lin-exp system stored in the variable Φ of Algorithm 3 after $i \in [0,n]$ iterations of the **while** loop, where Φ_0 is defined in line 2. Denote by m_i the number of constraints in Φ_i and by s_i the number of states in $\mathcal{A}_{\Phi_i}$. Then, assuming $m_0 \geq 1$ and $s_0 \geq 2$, for $i \in [1,n]$ we have

$$m_i \leq m_0 + 6\cdot i;\quad s_i \leq 2^{5^{i+1}}\cdot s_0^{5^i};\quad |\Phi_i| \leq i\cdot 5^{i+2}\cdot m_0\cdot|\Phi_0|.$$

This lemma is proved by induction on i. Only notice that at each iteration of the **while** loop we add at most 6 constraints due to lines 7 (where we call Algorithm 2) and 12. Within Algorithm 2, we can add at most 5 constraints: three simple regular constraints to rewrite $(x \geq y + c)$ in line 24 and two divisibilities in line 26. We can now prove our first theorem.

Proof of Theorem 1. By Lemma 4, the problem of deciding a formula ξ of ∃GSA is reducible in non-deterministic polynomial time to the problem of solvability in $\mathbb{N}$ of a quantifier-free regular lin-exp system φ. By Lemma 14, the solvability can be decided via Algorithm 3. In this algorithm, line 1 does not change the size of the formula; by Lemma 5, line 2 constructs an equisatisfiable ordered regular lin-exp system Φ_0 in time polynomial in $|\varphi|$. The **while** loop then works with the pairs (Φ,τ) of a regular lin-exp system Φ and a transition τ of $\mathcal{A}_\Phi = (Q,S,F,\Delta)$. The size of τ is at most $6\cdot|\Phi|$, and by Lemma 15, the size of Φ after the n-th iteration of this loop is bounded by $n\cdot 5^{n+2}\cdot m_0\cdot|\Phi_0|$, which is at most $n\cdot 5^{n+2}\cdot|\Phi_0|^2$ because m_0 is the number of constraints in Φ_0. By Lemmas 11 and 13, we see that each line of Algorithm 3 is performed in space polynomial in the size of the system currently stored in the variable Φ. Therefore, this non-deterministic algorithm works in exponential space in $|\Phi_0|$, and thus ξ can be decided in non-deterministic exponential space in $|\xi|$ and by Savitch's theorem, ∃GSA is decidable in ExpSpace. □

VI. Regular ILP complexity

The proof of Theorem 1 implies that ∃GSA is decidable in PSpace for every *fixed* number of applications of exponentiation in the formula. Let us show PSpace-hardness already for the positive existential conjunctive fragment of ∃GSA without exponentiation and for a fixed regular predicate.

Proof of Theorem 2. The PSPACE upper bound follows from Lemmas 3 and 4. Let us prove PSPACE-hardness.

We are going to construct a polynomial-time reduction from the intersection non-emptiness problem for NFAs over $\mathbb{B} = \{0, 1\}$, a well-known PSPACE-complete problem [18]. Recall that we take on input $\mathbb{B}$-NFAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and ask whether the intersection $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_n)$ is non-empty. Our first step is an adaptation of the existential representation of computations of non-deterministic finite automata from [29]: for every $\mathcal{A}_i$ there exists an existential formula $\exists \boldsymbol{y}.\varphi_i(x, \boldsymbol{y})$ in the language of PrA with bitwise AND operation $\&$ such that

$$[\![L(\mathcal{A}_i)]\!] = \{x \in \mathbb{N} : \exists \boldsymbol{y}.\varphi_i(x, \boldsymbol{y})\}.$$

Here we allow the NFA $\mathcal{A}_i$ to accept any string in $\mathtt{a}_0 \ldots \mathtt{a}_t \cdot \mathtt{0}^*$, where $[\![\mathtt{a}_0\mathtt{a}_1 \ldots \mathtt{a}_t]\!] = x$. Bitwise AND has its natural semantics, e.g., $23 \,\&\, 13 = 10111_2 \,\&\, 01101_2 = 00101_2 = 5$. Having this function, the bitwise precedence relation $x \preceq y$ for $x, y \in \mathbb{N}$ is defined as $x \,\&\, y = x$. For example, we have $5 \preceq 13$ because $5 \,\&\, 13 = 0101_2 \,\&\, 1101_2 = 0101_2 = 5$. The description from [29] is not in the form of a conjunction of positive atomic formulas. Let us show that this can nevertheless be achieved.

Let $\mathcal{A} = (Q, S, F, \Delta)$ be a $\mathbb{B}$-NFA, where the set of states is $Q = \{q_0, \ldots, q_s\}$. For convenience of notation, assume that $Q = [0, s]$ and $S, F \subseteq [0, s]$, i.e., that these sets contain the numbers of states instead of the states themselves. Let us also treat Δ as a transition function that maps a pair from $Q \times \mathbb{B}$ to a subset of Q. For $\mathtt{a} \in \mathbb{B}$, it is convenient to denote by $f_{\mathtt{a}}(x, t)$ the term $(t - x - 1)$ if $\mathtt{a} = \mathtt{0}$ and the variable x if $\mathtt{a} = \mathtt{1}$. In the formula below, this alias is used only when t is a power of 2 greater than x; therefore, $f_{\mathtt{0}}(x, t)$ is just a 2-complement of the binary expansion of x supplemented with some number of leading zeros. Then the desired formula is

$$x \in [\![L(\mathcal{A})]\!] \iff \exists t.\exists \boldsymbol{q}.\Big[t \,\&\, (t-1) = 0 \wedge (x < t) \wedge \quad (23)$$

$$\bigwedge_{0 \le i < j \le s} (q_i \,\&\, q_j = 0) \wedge (q_0 + \cdots + q_s = 2 \cdot t - 1) \wedge \quad (24)$$

$$(1 \preceq \sum_{i \in S} q_i) \wedge (t \preceq \sum_{i \in F} q_i) \wedge \quad (25)$$

$$\bigwedge_{(i,\mathtt{a}) \in Q \times \mathbb{B}} 2 \cdot \big(q_i \,\&\, f_{\mathtt{a}}(x, t)\big) \preceq \Big(\sum_{j \in \Delta(i,\mathtt{a})} q_j\Big)\Big]. \quad (26)$$

Constraint (23) restricts t to be a power of 2 greater than x; the variable t corresponds to the length of an accepting run of $\mathcal{A}$ on a binary expansion of x. The variables in $\boldsymbol{q} = (q_0, ..., q_s)$ accumulate the positions of each state in this accepting run. By (24), the vector $\boldsymbol{q}$ describes a sequence of states of length $\log(t) + 1$, and by (25) this sequence starts in an initial state of $\mathcal{A}$ and terminates in one of its final states. The sums in (25) are actually bitwise ORs because by (24) the variables for the states are pairwise disjoint. Constraints (26) are responsible for the transitions in $\mathcal{A}$: each time a non-deterministic computation on input x reaches q_i and reads a symbol $\mathtt{a}$, in the next step the computation will be in a state from $\Delta(q_i, \mathtt{a})$. If a sequence of states satisfies (25) and (26), then it is an accepting run of $\mathcal{A}$ on input (a binary expansion of) x.

It is clear that the size of the constructed formula $\varphi(x, t, \boldsymbol{q})$ is polynomial in $|\mathcal{A}|$. Conjoining these descriptions for the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, we obtain

$$L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_n) \neq \varnothing \iff \exists x \exists \boldsymbol{y} \bigwedge_{i=1..n} \varphi_i(x, \boldsymbol{y}), \quad (27)$$

where the vector $\boldsymbol{y}$ comprises the variables from the existential descriptions of the languages $L(\mathcal{A}_i)$.

Our next goal is to replace the expressions $z = x \& y$ using a positive conjunctive formula in the language of PrA with a regular predicate $R(x)$ defined as $x \in [\![(\mathtt{00} \cup \mathtt{10})^*]\!]$. Since in the definition of the function $[\![\cdot]\!]$ we used the *lsd*-first notation, this will mean that in the *msd*-first notation the variable x can be represented using a string from $(\mathtt{00} \cup \mathtt{01})^*$; for example, $R(5)$ is true because $\mathtt{0101} \in (\mathtt{00} \cup \mathtt{01})^*$. First observe that it is sufficient to define $x \,\&\, y = 0$ because we have

$$z = x \,\&\, y \iff (x - z) \,\&\, z = 0 \wedge x \,\&\, (y - z) = 0 \wedge z \le x \wedge z \le y.$$

The first equality says that for every $\mathtt{1}$ in the binary expansion of z, there is $\mathtt{1}$ at the same position in x; the next equality requires that there does not exist a position i in the binary expansions of x, y, and z such that the tuple of bits $[x_i, y_i, z_i]$ is either $[\mathtt{1}, \mathtt{1}, \mathtt{0}]$ or $[\mathtt{1}, \mathtt{0}, \mathtt{1}]$. Now consider the formula

$$x \,\&\, y = 0 \iff \exists u.\exists v.\exists z.\exists t.\Big[R(u) \wedge R(z) \wedge R(u + z) \wedge R(2v) \wedge R(2t) \wedge R(2v + 2t) \wedge x = u + v \wedge y = z + t\Big].$$

We have introduced two pairs of *disjoint* non-negative integers (u, z) and (v, t), where $\mathtt{1}$ can occur only at even positions of u, z, and at odd positions of v, t. Indeed, if u and z have $\mathtt{1}$ in the same position n, which must be even, then their sum will have $\mathtt{1}$ in the odd position $n + 1$, and thus the constraint $R(u+z)$ will not be satisfied. Observe that every non-negative integer x can be represented as a sum $u + v$ for some u and v satisfying $R(u) \wedge R(2v)$: the variable u represents the even bits of x and $2v$ is responsible for the odd ones.

When all bitwise ANDs in Equation (27) are excluded using these definitions, the size of the resulting system is linear in the size of the initial system. After introduction of new non-negative integer variables for each term in the scope of R, we obtain a system of linear equalities and inequalities where some variables satisfy R while the other variables are from $\mathbb{N}$. For each x ranging over $\mathbb{N}$ we introduce a pair of variables (u, v) such that $R(u) \wedge R(2v)$ and replace all occurrences of x with the sum $(u + v)$. Equalities and strict inequalities can be excluded in the usual way: $(x = y) \Leftrightarrow (x \le y) \wedge (y \le x)$ and $(x < y) \Leftrightarrow (x + 1 \le y)$. This completes the proof. □

ACKNOWLEDGMENTS

The author thanks the anonymous reviewers for their valuable remarks and suggestions. This work was supported by the Russian Science Foundation, project 23-71-01041.

References

[1] M. Benedikt, D. Chistikov, and A. Mansutti, "The complexity of Presburger arithmetic with power or powers," in *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), K. Etessami, U. Feige, and G. Puppis, Eds., vol. 261. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 112:1–112:18.

[2] B. Boigelot, "Symbolic methods and automata," in *Handbook of Automata Theory*, J. Pin, Ed. European Mathematical Society Publishing House, Zürich, Switzerland, 2021, pp. 1189–1215.

[3] B. Boigelot, S. Rassart, and P. Wolper, *On the expressiveness of real and integer arithmetic automata: Extended abstract*. Springer Berlin Heidelberg, 1998, pp. 152–163.

[4] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire, "Logic and p-recognizable sets of integers," *Bulletin of the Belgian Mathematical Society - Simon Stevin*, vol. 1, no. 2, pp. 191–238, jan 1994.

[5] R. J. Büchi, "Weak second-order arithmetic and finite automata," *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 6, no. 1-6, pp. 66–92, 1960.

[6] G. Cherlin and F. Point, "On extensions of Presburger arithmetic," in *Proceedings of the fourth Easter conference on model theory in Groß Köris*, ser. Seminarberichte 86. Groß Köris, Germany: Humboldt University, Berlin, 1986, pp. 17–34.

[7] D. Chistikov, "An Introduction to the Theory of Linear Integer Arithmetic," in *44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2024)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), S. Barman and S. Lasota, Eds., vol. 323. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, pp. 1:1–1:36.

[8] D. Chistikov, A. Mansutti, and M. R. Starchak, "Integer linear-exponential programming in NP by quantifier elimination," in *51th International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, ser. LIPIcs, vol. 297, Dagstuhl, Germany, 2024, pp. 132:1–132:20.

[9] M. Chrobak, "Finite automata and unary languages," *Theoretical Computer Science*, vol. 47, pp. 149–158, 1986.

[10] A. Draghici, C. Haase, and F. Manea, "Semënov arithmetic, affine VASS, and string constraints," in *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), 2024, pp. 29:1–29:19.

[11] R. Défossez, C. Haase, A. Mansutti, and G. A. Pérez, "Integer programming with gcd constraints," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2024, pp. 3605–3658.

[12] P. Gawrychowski, "Chrobak normal form revisited, with applications," in *International Conference on Implementation and Application of Automata (CIAA) 2011. LNCS*, B. Bouchou-Markhoff, P. Caron, J.-M. Champarnaud, and D. Maurel, Eds. Springer Berlin Heidelberg, 2011, pp. 142–153.

[13] C. Haase, "A survival guide to Presburger arithmetic," *ACM SIGLOG News*, vol. 5, no. 3, pp. 67–82, jul 2018.

[14] C. Haase and J. Różycki, "On the expressiveness of Büchi arithmetic," in *Foundations of Software Science and Computation Structures (FoSSaCS) 2021. LNCS*, vol. 12650. Luxembourg City, Luxembourg: Springer International Publishing, 2021, pp. 310–323.

[15] P. Habermehl, V. Havlena, M. Hečko, L. Holík, and O. Lengál, "Algebraic reasoning meets automata in solving linear integer arithmetic," in *Computer Aided Verification (CAV) 2024. LNCS*, vol. 14681. Springer Nature Switzerland, 2024, pp. 42–67.

[16] R. G. Jeroslow, "There cannot be any algorithm for integer programming with quadratic constraints," *Operations Research*, vol. 21, no. 1, pp. 221–224, 1973.

[17] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds., *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer Berlin Heidelberg, Nov. 2009.

[18] D. Kozen, "Lower bounds for natural proof systems," in *18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, 1977, pp. 254–266.

[19] A. Lechner, J. Ouaknine, and J. Worrell, "On the complexity of linear arithmetic with divisibility," in *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. USA: IEEE Computer Society, 2015, pp. 667–676.

[20] L. Lipshitz, "The diophantine problem for addition and divisibility," *Transactions of the American Mathematical Society*, vol. 235, pp. 271–283, Jan. 1978.

[21] Y. V. Matiyasevich, "Enumerable sets are diophantine," *Doklady Akademii Nauk SSSR*, vol. 191, no. 2, pp. 279–282, 1970.

[22] ——, *Hilbert's Tenth Problem*, ser. Foundations of Computing Series. Cambridge, Massachusetts, USA: MIT Press, 1993.

[23] T. Pheidas and X. Vidaux, "Extensions of Büchi's problem: questions of decidability for addition and kth powers," *Fundamenta Mathematicae*, vol. 185, no. 2, pp. 171–194, 2005.

[24] F. Point, "On the expansion $\langle \mathbb{N}; +, 2^x \rangle$ of Presburger arithmetic," 2007.

[25] M. Presburger, "Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt," in *Comptes Rendus du I congrès de Mathématiciens des Pays Slaves*, Warszawa, Poland, 1929, p. 92–101.

[26] Z. Sawa, "Efficient construction of semilinear representations of languages accepted by unary nondeterministic finite automata," *Fundamenta Informaticae*, vol. 123, no. 1, pp. 97–106, 2013.

[27] A. Schrijver, *Theory of linear and integer programming*. USA: John Wiley & Sons, Inc., 1986.

[28] A. L. Semënov, "Logical theories of one-place functions on the set of natural numbers," *Mathematics of the USSR-Izvestiya*, vol. 22, no. 3, pp. 587–618, jun 1984.

[29] M. R. Starchak, "On the existential arithmetics with addition and bitwise minimum," in *Foundations of Software Science and Computation Structures (FoSSaCS) 2023. LNCS*, vol. 13992. Paris, France: Springer International Publishing, 2023, pp. 176–195.

[30] ——, "Existential definability of unary predicates in Büchi arithmetic," in *Computability in Europe (CiE) 2024. LNCS*, vol. 14773. Springer Nature Switzerland, 2024, pp. 218–232.

[31] A. W. To, "Unary finite automata vs. arithmetic progressions," *Information Processing Letters*, vol. 109, no. 17, pp. 1010–1014, Aug. 2009.

[32] J. Utreras, "A logical approach to the problem of representation of integers by systems of diagonal forms," *Bulletin of the London Mathematical Society*, vol. 43, no. 2, pp. 299–310, jan 2010.