Artur Jeż | Alexander Okhotin

Complexity of solutions of equations over sets of numbers

Turku Centre for Computer Science

TUCS Technical Report
No 847, December 2007

Complexity of solutions of equations over sets of numbers

Artur Jeż

Institute of Computer Science,
University of Wrocław,
50–383 Wroclaw, Poland
aje@ii.uni.wroc.pl

Alexander Okhotin

Academy of Finland, *and*
Department of Mathematics, University of Turku, *and*
Turku Centre for Computer Science
Turku FIN–20014, Finland
alexander.okhotin@utu.fi

TUCS Technical Report

No 847, December 2007

Abstract

Systems of equations of the form $X_i = \varphi_i(X_1, \ldots, X_n)$ $(1 \leqslant i \leqslant n)$ are considered, in which the unknowns are sets of natural numbers. Expressions φ_i may contain the operations of union, intersection and pairwise sum $Y + Z = \{y + z \mid y \in Y, z \in Z\}$. These equations can be regarded as language equations or conjunctive grammars over a one-letter alphabet. A system with an EXPTIME-complete least solution is constructed in the paper, and it is established that least solutions of all such systems are in EXPTIME. The general membership problem for these equations is proved to be EXPTIME-complete.

Keywords: Language equations, integer expressions, conjunctive grammars, computational complexity

TUCS Laboratory
Discrete Mathematics for Information Technology

1 Introduction

The study of expressions over sets of numbers and of the computational complexity of their properties began in the paper by Stockmeyer and Meyer [17], who considered subsets of $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ as formal languages over a one-letter alphabet. In this case, concatenation of languages turns into a pairwise addition of elements of sets: $X + Y = \{x + y \,|\, x \in X, y \in Y\}$. Stockmeyer and Meyer established that the membership problem for expressions with union, intersection and addition is NP-complete.

Some extensions of this result were obtained by Yang [18], who considered integer circuits (that is, expressions in which subexpressions may be shared) with one more operation of pairwise multiplication, and established similar complexity results. A systematic study of complexity of expressions and circuits with different sets of operations was carried out by McKenzie and Wagner [9, 10].

In this paper we consider *equations over sets of natural numbers*, which are a more general device than expressions and circuits, and study the computational complexity of their least solutions, as well as of their membership problem. These equations naturally correspond to *language equations* over a one-letter alphabet. Language equations have recently become an active area of research, see a recent survey by Kunc [8]. In particular, unexpected hardness results on language equations have been obtained by Kunc [7] and by Okhotin [15, 16], and this connection gives another motivation for our study. Recent results by Jeż [5] on the expressive power of *conjunctive grammars* provide a technical foundation for our results.

We consider equations in the *resolved form*

$$\begin{cases} X_1 &= \varphi_1(X_1, \ldots, X_n) \\ &\vdots \\ X_n &= \varphi_n(X_1, \ldots, X_n) \end{cases} \tag{*}$$

in which every variable X_i assumes value of a set of nonnegative integers. The right-hand side φ_i of each equation may contain the operations of union, intersection and $+$, as well as singleton constants. Every such system has a least solution with respect to componentwise inclusion, which can be obtained by fixpoint iteration. Our result, established in Section 3, is a construction of a system (*), such that testing the membership of numbers in its least solution is an EXPTIME-hard problem (with the numbers given in binary notation). The result is obtained by a new kind of arithmetization of an alternating linear-space Turing machine. It is also shown that for every system (*) the membership of numbers in its least solution can be tested in exponential time, which makes the constructed set the hardest.

Let us compare our result to the existing results on expressions and circuits on sets of numbers. Previous research was concerned with the complexity of the general membership problem, where it was sufficient to encode an instance of some hard problem for numbers in an expression or a circuit. In our case, the task is to construct a system that represents a class of problems, while instances of that problem are to be encoded as numbers.

As compared to the research on language equations, our present approach studies a similar problem of constructing a representation of a hard set (cf. Kunc [7], Okhotin [15, 14], Jeż [5]). However, while encoding a computation of a Turing machine as a string over $\{a, b\}$ is an ordinary task, in our case we have to encode similar objects as numbers, that is, as strings over a one-letter alphabet. These strings have no apparent structure, and hence the proposed arithmetization is quite unobvious.

This result allows us to establish the complexity of the general membership problem for equations with $\{\cup, \cap, +\}$, which is stated as follows: *"Given a system and a number $n \geqslant 0$ in binary notation, determine whether n is in the first component of the least solution of the system"*. For integer expressions and integer circuits with the operations $\{\cup, \cap, +\}$, it is known from Stockmeyer and Meyer [17] and from McKenzie and Wagner [9, 10] that a similar problem is PSPACE-complete. Another weaker model are equations with $\{\cup, +\}$, that is, without intersection, for which the corresponding problem is NP-complete due to the result of Huynh [4] on the commutative case of the context-free grammars. In our case of equations with $\{\cup, \cap, +\}$, the general membership problem is EXPTIME-complete, which is established in Section 4. An exponential algorithm for solving this problem is given by a parsing algorithm on conjunctive grammars [13].

2 Language equations and conjunctive grammars

While our results are on the complexity of equations in sets of numbers, our methods are derived from the domain of formal language theory, in particular, from some recent results on language equations.

In language equations, the unknowns are formal languages over an alphabet Σ. If $|\Sigma| = 1$, they coincide with equations over sets of numbers, while for larger alphabets they constitute a more general notion. The main object of this study are equations of the resolved form (*), in which variables assume values of sets of non-negative integers, and the right-hand sides may contain the operations of union, intersection and addition of sets. These equations obviously correspond to *language equations* over a one-letter alphabet with the operations of union, intersection and concatenation, and the recent results on language equations of this kind provide a theoretical foundation, as well as a second motivation, for the present research.

The first type of language equations to be studied were equations of the same form (*) containing union and concatenation, but no intersection: Ginsburg and Rice [3] established that these equations provide a natural semantics for the context-free grammars. Equations with added intersection therefore constitute a generalization of the context-free grammars.

Definition 1 (Okhotin [12]). *A conjunctive grammar is a quadruple $G = (\Sigma, N, P, S)$, in which Σ and N are disjoint finite non-empty sets of terminal and nonterminal symbols respectively; P is a finite set of grammar rules, each of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_n \quad (\textit{where } A \in N,\ n \geqslant 1 \textit{ and } \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)$$

while $S \in N$ is a nonterminal designated as the start symbol.

The semantics of conjunctive grammars is defined by the least solution of the following system of language equations:

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \in P} \ \bigcap_{i=1}^{m} \alpha_i \quad (\textit{for all } A \in N) \tag{1}$$

The component corresponding to each $A \in N$ is then denoted by $L_G(A)$, and $L(G)$ is defined as $L_G(S)$.

The operations used in the right-hand sides of systems (1) are union, intersection and concatenation. Since they are monotone and continuous, a least solution always exists and can be obtained by fixpoint iteration as

$$\bigsqcup_{i \geqslant 0} \varphi^i(\varnothing, \ldots, \varnothing), \tag{2}$$

where φ is the right-hand side of (1) as a vector operator on $|N|$-tuples of languages, while $\sqcup$ denotes pairwise union of vectors of sets.

An equivalent definition of conjunctive grammars can be given using term rewriting [12], which generalizes Chomsky's word rewriting. The importance of these grammars lies with the fact that their expressive power is substantially greater than that of the context-free grammars, while the generated languages can still be parsed in time $O(n^3)$, and the practical context-free parsing algorithms, such as recursive descent and generalized LR, admit generalization to conjunctive grammars without an increase in their complexity.

The question of whether conjunctive grammars can generate any non-regular unary language has been an open problem for some years, until recently solved by Jeż [5], who constructed a grammar for the language $\{a^{4^n} \mid n \geqslant 0\}$. Let us reformulate this grammar as the following resolved system of four equations over sets of numbers:

Example 1 (Jeż [5]). *The system*

$$\begin{cases} X_1 &= \big((X_2 + X_2) \cap (X_1 + X_3)\big) \ \cup\ \{1\} \\ X_2 &= \big((X_{12} + X_2) \cap (X_1 + X_1)\big) \ \cup\ \{2\} \\ X_3 &= \big((X_{12} + X_{12}) \cap (X_1 + X_2)\big) \ \cup\ \{3\} \\ X_{12} &= \big((X_3 + X_3) \cap (X_1 + X_2)\big) \end{cases}$$

has least solution $X_i = \{\ell \mid \textit{base-4 notation of } \ell \textit{ is } i0\ldots0\}$, for $i = 1, 2, 3, 12$.

Sets of this kind can be conveniently specified by regular expressions for the corresponding sets of base-k notations of numbers, which in this case are $\mathtt{10}^*$, $\mathtt{20}^*$, $\mathtt{30}^*$ and $\mathtt{120}^*$, respectively. In the following we shall omit some parentheses in the right-hand sides of equations, and assume the following default precedence of operations: addition has the highest precedence, followed by intersection, and then by union with the least precedence.

Using the same technique in a more elaborate construction, a general theorem on the expressive power of unary conjunctive grammars was established. It can be reformulated for equations over sets of numbers as follows:

Theorem 1 (Jeż [5]). *For every $k \geqslant 2$ and for every finite automaton M over the alphabet $\{0, \dots, k-1\}$ there exists a system of resolved language equations over $\mathbb{N}_0$ using $\cup, \cap, +$, such that its least solution is*

$$(S_1, S_2, \dots, S_n),$$

where $S_i \subseteq \mathbb{N}_0$ and $S_1 = \{\ell \mid k\text{-ary notation of } \ell \text{ is in } L(M)\}$.

Let us note in passing a recent paper by Jeż and Okhotin [6] establishing a generalization of this result to a larger family of automata recognizing positional notations.

Though representing sets of numbers with a regular positional notation using this type of formal grammars was an unexpected and strong result in terms of language theory, it has no implications on computational complexity, as all these sets are computationally easy. More general representation theorem of Jeż and Okhotin [6] also does not imply any better complexity results than P-completeness, which, as the present paper shows, is much below the actual complexity of these equations.

Therefore, a new method of constructing such equations is needed to understand their complexity. This step is made in the next section, which introduces an arithmetization technique based upon addition of sets of numbers.

3 Representing an EXPTIME-complete language

In this section it will be shown that languages defined by least solutions of resolved language equations using $+$, $\cup$ and $\cap$ can be EXPTIME-complete, and this is the hardest language in this family. Denote this family by $EQ(\cup, \cap, +)$.

Theorem 2. *The family $EQ(\cup, \cap, +)$ is contained in EXPTIME and contains an EXPTIME-complete language.*

The proof is by constructing such a system of equations. The given system encodes a computation of a linear-bounded alternating Turing machine (ATM) It is known that such machines can recognize some EXPTIME-complete languages [2].

In our case we shall consider ATMs operating on a circular tape and moving to the right at every step. Its tape originally contains the input word, and the squares containing it constitute all space available to the machine. Obviously, such machines are as powerful as linear-bounded ATMs of the general form.

Formally, such a machine is defined as $M = (\Omega, \Gamma, Q_E, Q_A, \delta, q_0, q_{\mathit{fin}})$, where Ω is the input alphabet, $\Gamma = \{a_0, a_1, \ldots, a_{\max}\} \supset \Omega$ is the tape alphabet, Q_E and Q_A are disjoint sets of existential and universal states, respectively, $Q = Q_E \cup Q_A$ and $q_0, q_{\mathit{fin}} \in Q$. Given an input $w \in \Omega^+$, M starts in state q_0 with the head over the first symbol of w. The transition function is $\delta : Q \times \Gamma \to 2^{Q \times \Gamma}$, and the head is moved one symbol to the right at every step. Once the head moves beyond the right-most symbol, it is moved back over the first symbol of w, maintaining its current state; this implements a circular tape. For technical reasons, assume that $(q, a') \notin \delta(q, a)$ for all $q \in Q$ and $a, a' \in \Sigma$, (that is, the machine never stays in the same state), and that $\delta(q, a) \neq \varnothing$ for all $q \in Q_A$ and $a \in \Sigma$.

Our construction of a system of equations over sets of numbers simulating a computation is based upon representing instantaneous descriptions of the ATM as *numbers*. We shall think of these numbers as written in base-$(8 + |Q| + \max(|Q| + 7, |\Gamma|))$ positional notation, and the entire argument is based upon mapping the symbols used by the machine to digits, and then using addition to manipulate individual digits in the positional notation of numbers. It must be noted that this positional notation is only a tool for our understanding of the constructions, while the actual equations deals with numbers as they are.

Let $\Sigma = \{\mathtt{0}, \mathtt{1}, \ldots, 7 + |Q| + \max(|Q| + 7, |\Gamma|)\}$ be the alphabet of digits, and define the mapping of symbols to digits, $\langle \cdot \rangle : Q \cup \Gamma \to \Sigma$, as follows:

$$\begin{aligned} \langle q_i \rangle &= 7 + i \quad (\text{for } q_i \in Q) \\ \langle a_i \rangle &= 7 + |Q| + i \quad (\text{for } a_i \in \Gamma) \end{aligned}$$

Furthermore, let $\langle Q \rangle = \{\langle q \rangle \mid q \in Q\}$ and $\langle \Gamma \rangle = \{\langle a \rangle \mid a \in \Gamma\}$. Now the tape of the ATM containing symbols $a_{i_1} \ldots a_{i_n}$, with the head over the j-th symbol and the machine in state q, is represented as the following string of digits:

$$\mathtt{0}\langle a_{i_1} \rangle \ldots \mathtt{0}\langle a_{i_{j-1}} \rangle \langle q \rangle \langle a_{i_j} \rangle \mathtt{0} \langle a_{i_{j+1}} \rangle \ldots \mathtt{0}\langle a_{i_n} \rangle \mathtt{0} \in \Sigma^*$$

For technical reasons, configurations in which the head has just moved over the last symbol but has not yet jumped to the first position are considered separately, and will be represented as strings of the form

$$\mathtt{0}\langle a_{i_1} \rangle \ldots \mathtt{0}\langle a_{i_n} \rangle \langle q \rangle,$$

where q is the current state. Note that digits denoting letters are written only in even positions, while odd positions are reserved for the states of the

Turing machine. The set of all strings of digits representing valid encodings of tapes is specified by the following regular expression over Σ:

$$\text{Tape} = (\mathtt{0}\langle\Gamma\rangle)^*\langle Q\rangle(\langle\Gamma\rangle\mathtt{0})^* \setminus \langle Q\rangle$$

The set Tape should be a considered as a formal language over Σ, which will be used later as a part of representations of some sets of numbers. Subsets of this set representing tapes with different states will be denoted as follows:

$$\text{Tape}_u = \{w \mid w \in \text{Tape},\ u \text{ is a substring of } w\}$$
$$\text{Tape}^\ell_u = \{w \mid w \in \text{Tape},\ u \text{ is a prefix of } w\}$$

Besides the contents of the tape, the encoding for Turing machine configurations uses a counter of rotations of the circular tape. This counter specifies the number of passes through the tape the machine is still allowed to make before it must halt. It is represented in binary notation using digits $\{\mathtt{0}, \mathtt{1}\}$, and the set of valid counter representations is

$$\text{Counter} = \mathtt{1}\{\mathtt{0}, \mathtt{1}\}^*$$

Normally the counter uses only digits $\{\mathtt{0}, \mathtt{1}\}$, but in order to implement the incrementation of the counter we shall use strings with one digit $\mathtt{2}$ representing zero with carry. The set of valid representations of counters with a carry is

$$\text{Counter}' = \mathtt{1}\{\mathtt{0}, \mathtt{1}\}^*\mathtt{2}\{\mathtt{0}, \mathtt{1}\}^* \cup \mathtt{2}\{\mathtt{0}, \mathtt{1}\}^*$$

For every string $c_{k-1} \dots c_0 \in \text{Counter} \cup \text{Counter}'$, define its value as

$$\text{Value}(c_{k-1} \dots c_0) = \sum_{j=0}^{k-1} c_j \cdot 2^j.$$

Now define the mapping from configurations of the Turing machine to numbers. A configuration with the tape contents, head position and current state given by a string of digits $w \in \text{Tape}$, and with the counter value given by $x \in \text{Counter}$ is represented by a string of digits

$$x\mathtt{55}w,$$

where two marker digits $\mathtt{55}$ separate the values. This string of digits in base-$|\Sigma|$ positional notation specifies a certain number, which accordingly represents the configuration.

The key property of this encoding is that *every transition of the ATM reduces the numerical value of its configuration.* Indeed, if the head is moved to the right, then a digit $\langle q\rangle$ is replaced with $\mathtt{0}$ and all other modifications are done on less signigicant digits. If the head jumps from the end to the beginning, then the counter is decremented, and since the counter occupies more significant positions in the number than the tape, this transition decreases the value of the configuration as well. This monotonicity allows us to encode dependence of configurations on each other by using addition of nonnegative numbers only.

The construction of equations representing the computation of the ATM begins with some expressions that will be used in the right-hand sides of equations. These expressions contain some constant sets of numbers given as regular languages over the alphabet Σ. Every such language represents the set of all numbers with $|\Sigma|$-ary notation of the given form. According to Theorem 1, every such set can be represented by a separate system of equations using only singleton constants. All these subsystems are assumed to be included in the constructed system, and each of the regular expressions in the system can be formally regarded as a reference to one of the auxiliary variables.

Definitions of a few of these regular languages incorporate positional notations of numbers obtained by subtracting one number from another. For convenience, these values are given in the form $u \boxminus v$, with $u, v \in \Sigma^*$ being positional notations of two numbers (the former shall be greater or equal to the latter). One can write, e.g., $(u \boxminus v)\mathtt{0}^*$ for the set of all numbers with their $|\Sigma|$-ary notation beginning with fixed digits determined by the given difference, followed with any number of zeroes.

$$\begin{aligned}
\mathrm{Step}(X) &= \Big(\bigcup_{\substack{q \in Q_E \\ a \in \Gamma}} \bigcup_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X) \Big) \cup \Big(\bigcup_{\substack{q \in Q_A \\ a \in \Gamma}} \bigcap_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X) \Big) \\
\mathrm{Move}_{q,a,q',a'}(X) &= \big(X \cap \mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}_{\langle a \rangle \langle q \rangle} \big) + \big(\langle q' \rangle \langle a' \rangle \mathtt{0} \boxminus \langle a \rangle \langle q \rangle \big) (\mathtt{00})^* \\
&\quad \cap \mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}_{\langle q' \rangle \langle a' \rangle} \\
\mathrm{Jump}(X) &= \bigcup_{q} \Big[\big(X \cap \mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}^{\ell}_{\langle q \rangle} \big) + (\mathtt{1000} \boxminus \langle q \rangle)(\mathtt{00})^+ + \langle q \rangle \Big] \\
&\quad \cap (\mathrm{Counter} \cup \mathrm{Counter}')\mathtt{55}\,\mathrm{Tape}_{\langle q \rangle} \\
\mathrm{Carry}(Y) &= \Big[\Big(\big(Y \cap \{\mathtt{0},\mathtt{1}\}^* \mathtt{2} \{\mathtt{0},\mathtt{1}\}^* \,\mathtt{55}\,\mathrm{Tape} \big) + \mathtt{10}^* \cap \{\mathtt{0},\mathtt{1}\}^* \mathtt{3} \{\mathtt{0},\mathtt{1}\}^* \,\mathtt{55}\,\mathrm{Tape} \Big) \\
&\quad + \big(\mathtt{10} \boxminus \mathtt{3} \big) \mathtt{0}^* \Big] \cap \big(\{\mathtt{0},\mathtt{1}\}^+ \cup \{\mathtt{0},\mathtt{1}\}^* \mathtt{2} \{\mathtt{0},\mathtt{1}\}^* \big) \,\mathtt{55}\,\mathrm{Tape}
\end{aligned}$$

In addition, define the set of final configurations of the machine:

$$\mathrm{Final} = \mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}_{\langle q_{fin} \rangle}$$

The construction uses two variables, X and Y. Either variable represents the set of proper configurations of the machine, starting from which the machine accepts. The variable X represents configurations belonging to the set $\mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}$, while Y represents configurations from $(\mathrm{Counter} \cup \mathrm{Counter}')\mathtt{55}\,\mathrm{Tape}$, in which the counter may contain one carry digit $\mathtt{2}$ that needs to be propagated to higher positions. The equations, using the above auxiliary functions, are as follows:

$$X = \mathrm{Final} \cup \mathrm{Step}(X) \cup \big(Y \cap \mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}\big) \tag{3}$$

$$Y = \mathrm{Jump}(X) \cup \mathrm{Carry}(Y) \tag{4}$$

In order to determine the least solution of this system, let us first establish some properties of the auxiliary functions.

The first quite elementary property is their distributivity over infinite union, which allows us to study these operations as operations on individual numbers, and then infer their action on sets of numbers.

Lemma 1 (Distributivity)**.** *Each function $f \in \{\mathrm{Move}_{q,a,q',a'}, \mathrm{Jump}, \mathrm{Carry}\}$ is distributive over infinite union, in the sense that $f(S) = \bigcup_{n \in S} f(\{n\})$ for every $S \subseteq \mathbb{N}_0$.*

This follows from the fact that each of these expressions consists of intersections with constant sets, sums with constant sets and unions.

Lemma 2. *Let $\varphi(X)$ be an expression defined as a composition of the following operations: (i) the variable X; (ii) constant sets; (iii) union; (iv) intersection with a constant set; (v) addition of a constant set. Then φ is distributive over infinite union, that is, $\varphi(X) = \bigcup_{n \in X} \varphi(\{n\})$.*

On the other hand, note that if an expression contains intersections or sums of multiple expressions involving X, then it is not necessarily distributive over infinite union; in particular, Step need not be distributive.

Proof. Induction on the structure of φ.

Basis. If $\varphi(X) = X$ or $\varphi(X) = C \subseteq \mathbb{N}_0$, the statement trivially holds.

Induction step I. Let $\varphi(X) = \psi(X) \cup \xi(X)$. By the induction hypothesis, $\psi(X) = \bigcup_{n \in X} \psi(\{n\})$ and $\xi(X) = \bigcup_{n \in X} \xi(\{n\})$. Therefore, $\varphi(X) = \bigcup_{n \in X} \big(\psi(\{n\}) \cup \xi(\{n\})\big) = \bigcup_{n \in X} \varphi(\{n\})$.

Induction step II. If $\varphi(X) = \psi(X) \cap C$ for some $C \subseteq \mathbb{N}_0$, then, by the induction hypothesis, $\psi(X) = \bigcup_{n \in X} \psi(\{n\})$. Since union and intersection are distributive, $\varphi(X) = \bigcup_{n \in X} \big(\psi(\{n\}) \cap C\big) = \bigcup_{n \in X} \varphi(\{n\})$.

Induction step III. The case of $\varphi(X) = \psi(X) + C$ is handled similarly to the previous case, using the distributivity of union and concatenation. □

One of the main technical devices used in these functions is addition of a constant set of numbers with $|\Sigma|$-ary notation $u\mathtt{0}^*$ (that is, a set $\{\, m \cdot |\Sigma|^i \mid i \geqslant 0 \,\}$) with one, two or three non-zero digits in u. The following lemma establishes that this addition can never rewrite the double markers $\mathtt{55}$, that is, every sum in which these markers are altered does not represent a valid tape contents. This means that every such addition manipulates the counter and the tape separately, and the changes do not mix.

Lemma 3 (Marker preservation). *For every $x, x' \in \{\mathtt{0},\mathtt{1},\mathtt{2},\mathtt{3}\}^* \setminus \mathtt{0}\Sigma^*$ and $w, w' \in \text{Tape}$, if $x'\mathtt{55}w' \in x\mathtt{55}w + (\Sigma^3 \cup \Sigma^2 \cup \Sigma)\mathtt{0}^*$, then $|w| = |w'|$.*

Proof. Let $y = ijk\mathtt{0}^\ell$, with $i, j, k \in \Sigma$, be a string representing a number, and assume $x'\mathtt{55}w' = x\mathtt{55}w + y$. The ℓ least significant digits of $x\mathtt{55}w$ and of $x'\mathtt{55}w'$ are the same.

Consider the $(\ell+4)$th digit of $x\mathtt{55}w$, let it be c. Since y has less than $\ell+4$ digit, any change at this position can only be due to a carry from position $\ell+3$. As $|\Sigma|-1$ is not a proper encoding, then clearly $c < |\Sigma|-1$. Since the carry digit is at most 1, the $(\ell+4)$th digit in $x'\mathtt{55}w'$ is less or equal to $c+1$, that is, it is less or equal to $|\Sigma|-1$. Therefore, there is no carry in $x\mathtt{55}w+y$ at position $\ell+4$. Since y has $\ell+3$ digits and there is no carry from position $\ell+4$, all digits in positions greater than $\ell+4$ in $x\mathtt{55}w+y$ are the same as in $x\mathtt{55}w$. Hence, $x'\mathtt{55}w'$ has at most four digits different from $x\mathtt{55}w$, which may be at positions $\ell+1$, $\ell+2$, $\ell+3$ and $\ell+4$.

Assume for the sake of contradiction that $|w| \neq |w'|$. Since w and w' are both of odd length, the positions of $\mathtt{5}$ in words $x\mathtt{55}w$ and $x'\mathtt{55}w'$ are different. Hence $x\mathtt{55}w$ and $x'\mathtt{55}w'$ differ at exactly four positions, which are the positions of $\mathtt{5}$ in them.

Note that if four digits are modified by adding y, then the digit in position $\ell+4$ can only be incremented by 1 due to a carry from the previous position. Since one of the words $x\mathtt{55}w$, $x'\mathtt{55}w'$ has digit $\mathtt{5}$ in position $\ell+4$, the other word should have digit $\mathtt{4}$ or $\mathtt{6}$ in the same position. Because the latter digits are not encodings of any symbols, this yields a contradiction. □

The next statement describes the operation of Carry: applied to a configuration with the counter having a single carry digit $\mathtt{2}$, Carry changes this digit to $\mathtt{0}$ and increments the next digit, making it $\mathtt{1}$ or $\mathtt{2}$. Note that all operations are in $|\Sigma|$-ary notation. The tape contents is not altered.

Lemma 4 (Carry propagation). *For every $x \in \text{Counter}'$ and for every $w \in \text{Tape}$, $\text{Carry}\big(\{x\mathtt{55}w\}\big) = \{x'\mathtt{55}w\}$, where $x' \in \text{Counter} \cup \text{Counter}'$ and $\text{Value}(x') = \text{Value}(x)$. If $x' \in \text{Counter}'$, then the position of $\mathtt{2}$ in x' is greater than the position of $\mathtt{2}$ in x.*

Proof. It has to be proved that if $x = \mathtt{2}\widetilde{x} \in \text{Counter}'$ and $w \in \text{Tape}$, then

$$\text{Carry}(\{\mathtt{2}\widetilde{x}\mathtt{55}w\}) = \{\mathtt{10}\widetilde{x}\mathtt{55}w\},$$

and if $x = \widehat{x}c\mathtt{2}\widetilde{x} \in \text{Counter}'$ and $w \in \text{Tape}$, then

$$\text{Carry}(\{\widehat{x}c\mathtt{2}\widetilde{x}\mathtt{55}w\}) = \{\widehat{x}(c+1)\mathtt{0}\widetilde{x}\mathtt{55}w\}.$$

If a string $x\mathtt{55}w$, with $x \in \text{Counter}' \cup \text{Counter}$ and $w \in \text{Tape}$, is substituted into the expression Carry, then the first subexpression contains all strings of the form

$$u \in x\mathtt{55}w{+}\mathtt{10}^* \;\cap\; \{\mathtt{0},\mathtt{1}\}^*\mathtt{3}\{\mathtt{0},\mathtt{1}\}^*\,\mathtt{55}\,\text{Tape}\,.$$

Consider the possible changes done to $x\mathtt{55}w$ to obtain u. As 1 is added only to one digit, there cannot be a carry, because the last digit in Σ is not used for encoding. Therefore, only one digit is modified in $x\mathtt{55}w$. Since $x\mathtt{55}w$ does not contain the digit $\mathtt{3}$ that occurs in u, the unique digit $\mathtt{2}$ in x must be replaced by $\mathtt{3}$. Denote $u = x''\mathtt{55}w$.

Consider the string u' (any such string if it is not unique) obtained from operation

$$u' \in u + \big(\mathtt{10} \boxminus \mathtt{3}\big)\mathtt{0}^* \cap \Big(\{\mathtt{0},\mathtt{1}\}^+ \cup \{\mathtt{0},\mathtt{1}\}^*\mathtt{2}\{\mathtt{0},\mathtt{1}\}^*\Big)\,\mathtt{55}\,\mathrm{Tape}$$

Let $u' = u+y$, with $y \in (\mathtt{10}\boxminus\mathtt{3})\mathtt{0}^* = (k-3)\mathtt{0}^*$. By Lemma 3, $u' = x'\mathtt{55}w'$ and $|w'| = |w|$.

Consider the changes in $x'\mathtt{55}w'$ as compared to $x''\mathtt{55}w$. Since in x'' there is $\mathtt{3}$ and in x' there is none, the position of $\mathtt{3}$ in x'' is one of the modified positions. Denote the number of this position by k. There is only one non-zero digit in y. Since the addition of y has modified digit $\mathtt{3}$, this means that the non-zero digit in y is in position k or $k-1$. If it is in position $k-1$, then one can only modify $\mathtt{3}$ by adding 1 as a carry from position $k-1$. This is a contradiction, as $\mathtt{3}+1$ is not a proper encoding. Otherwise, if it is in position k, then adding y to $x''\mathtt{55}w$ replaces $\mathtt{3}$ with $\mathtt{0}$ and results in a carry, thus increasing the digit in position $k+1$ by 1.

Note, that in particular no changes were applied to w, hence $w' = w$. Also if there is a $\mathtt{2}$ in x' then its position is greater than k, as no digits on positions smaller than k were changed and on position k it has $\mathtt{0}$.

Consider also the values of the counters x and x'. The value of x is $\sum c_i 2^i$. If x has no digit it position $k+1$, then assume for the purposes of calculation that $c_{k+1} = 0$ (this does not influence the value of the counter). In x', the digit $\mathtt{2}$ was replaced by $\mathtt{0}$, hence $c'_k = \mathtt{0}$. On the other hand, c_{k+1} was replaced with $c_{k+1}+1$. If c_{k+1} did not exist, then a new digit $c'_{k+1} = \mathtt{1}$ has been created. In any case $c'_{k+1} = c_{k+1}+1$. All other digits of the counters are left intact. Then the difference of the values of the counters is determined by positions k and $k+1$, and

$$\mathrm{Value}(x) - \mathrm{Value}(x') = (c_{k+1}\cdot 2^{k+1} + \mathtt{2}\cdot 2^k) - ((c_{k+1}+1)\cdot 2^{k+1} + \mathtt{0}\cdot 2^k) = 0,$$

that is, the value of the counter has been preserved. □

According to Lemma 4, Carry moves the carry by one position higher. The next lemma shows that sufficiently many iterations of Carry always eliminate the carry digit: given a counter with the notation $x = \widetilde{x}\mathtt{0}\mathtt{1}^{k-1}\mathtt{2}$, Carry^k transforms it to $x = \widetilde{x}\mathtt{1}\mathtt{0}^{k-1}\mathtt{0}$.

Lemma 5 (Termination of carry propagation). *For every $x \in \mathrm{Counter} \cup \mathrm{Counter}'$ and $w \in \mathrm{Tape}$ there exists $x' \in \mathrm{Counter}$ and $k \geqslant 0$, such that $\mathrm{Carry}^k(x\mathtt{55}w) = x'\mathtt{55}w$ and $\mathrm{Value}(x) = \mathrm{Value}(x')$.*

Proof. If $x \in \text{Counter}$, then $k = 0$ and $x' = x$ clearly satisfy the statement of the lemma.

Let $x \in \text{Counter}'$ and construct a sequence $\{x_i\}_{i \geqslant 0}$, with $x_i \in \text{Counter}'$ and $\text{Value}(x_i) = \text{Value}(x)$, as follows. Let $x_0 := x$. For every $i \geqslant 1$, consider $\text{Carry}(x_{i-1}\mathtt{55}w)$, which, by Lemma 4, equals $\{y\mathtt{55}w\}$ for some $y \in \text{Counter}' \cup \text{Counter}$ with $\text{Value}(y) = \text{Value}(x_{i-1})$. If $y \in \text{Counter}'$, let $x_i := y$. Otherwise, if $y \in \text{Counter}$, then $k := i$ and $x' := y$ satisfy the statement of the lemma.

Note that, by Lemma 4, the position of $\mathtt{2}$ in each x_{i+1} is greater than in x_i, hence all elements of the sequence are distinct. Since there exist finitely many elements of $\text{Counter}'$ having the same value, the sequence cannot be infinite and eventually $y \in \text{Counter}$ is obtained. □

The next lemma states the functionality of Jump, which can be described as follows. If Jump is applied to a configuration in which the head scans over the first symbol, then the result of the operation is the *previous* configuration, in which the head is at the right-most position beyond the end of the string, while the value of the counter x is greater by 1.

Lemma 6. *Let* $x = \widetilde{x}c \in \text{Counter}$ *with* $c \in \{\mathtt{0}, \mathtt{1}\}$ *and* $w = \langle q \rangle \widetilde{w}\mathtt{0} \in \text{Tape}$ *with* $q \in Q$*, that is,* w *encodes a configuration with the head over the first symbol. Then* $\text{Jump}(x\mathtt{55}w) = \{\widetilde{x}(c+1)\mathtt{550}\widetilde{w}\langle q \rangle\}$.

For any string $\alpha \in \Sigma^*$ *of a different form,* $\text{Jump}(\alpha) = \varnothing$.

Proof. The inner subexpression of $\text{Jump}(x\mathtt{55}w)$,

$$\{x\mathtt{55}w\} \cap \text{Counter}\,\mathtt{55}\,\text{Tape}^{\ell}_{\langle q \rangle},$$

ensures that $w = \langle q \rangle \widetilde{w}\mathtt{0}$ for some $\widetilde{w} \in \langle \Gamma \rangle (\mathtt{0}\langle \Gamma \rangle)^*$, that is, that the state symbol is the left-most one. If w is of a different form, then $\text{Jump}(x\mathtt{55}w) = \varnothing$. Fix an arbitrary state $q \in Q$; as the outermost operation in Jump, a union over all q will be taken.

The next subexpression performs an addition

$$x\mathtt{55}\langle q \rangle \widetilde{w}\mathtt{0} + (\mathtt{1000} \boxminus \langle q \rangle)(\mathtt{00})^+ + \langle q \rangle.$$

Consider an arbitrary $y = (\mathtt{1000} \boxminus \langle q \rangle)(\mathtt{00})^k \ + \langle q \rangle$, with $k > 0$. Denote $u = x\mathtt{55}w + y$. Assume that

$$u \in (\text{Counter}' \cup \text{Counter})\mathtt{55}\,\text{Tape}_q,$$

as the next operation in Jump is an intersection with this set. In particular, there are $x' \in \text{Counter} \cup \text{Counter}'$ and $w' \in \text{Tape}$, such that $u = x'\mathtt{55}w'$. By Lemma 3, $|w| = |w'|$.

Notice that $w' = \mathtt{0}\widetilde{w}'\langle q \rangle$, as the right-most digit in y is $\langle q \rangle$ and the right-most digit in w is $\mathtt{0}$ and w' has only one state digit.

As y has non-zero digits only on positions $2k+1, 2k+2, 2k+3$ and the digit $|\Sigma|-1$ does not encode any symbol, adding y cannot change any digit in $x\mathtt{55}w$ in positions greater than $2k+4$. Let $2\ell = |\widetilde{w}\mathtt{0}|$. Then adding y to w modifies the digit in position $2\ell+1$, which is $\langle q\rangle$. Hence $2\ell+1 = 2k+1$ or $2\ell+1 = 2k+3$. If $2\ell+1 = 2k+3$, then in position $2\ell+1$ in $x\mathtt{55}w+y$ there is either $\langle q\rangle$ or $\langle q\rangle - 1$. And those are clearly not $\langle 0\rangle$, which is in position $2\ell+1$ in w'.

Hence $2\ell+1 = 2k+1$. Let $x = \widetilde{x}c$. Then $x\mathtt{55}w + y = \widetilde{x}(c+1)\mathtt{550}\widetilde{w}\langle q\rangle$, hence $x' = \widetilde{x}(c+1)$ and $w' = \mathtt{0}\widetilde{w}\langle q\rangle$. □

It follows from Lemma 6 that Jump is a reversible function, that is, the previous configuration given by Jump($x\mathtt{55}w$) corresponds to $x\mathtt{55}w$ only. This is stated as follows:

Lemma 7. *Let* $x'\mathtt{55}w' \in \text{Jump}(x\mathtt{55}w)$*. Then* $w' = \mathtt{0}\widetilde{w}\langle q\rangle$ *and* $w = \langle q\rangle\widetilde{w}\mathtt{0}$ *for some state* q*, and* $\text{Value}(x') = \text{Value}(x) + 1$*.*

Proof. Let $x'\mathtt{55}w' \in \text{Jump}(x\mathtt{55}w)$. We use Lemma 6. Let $x = \widetilde{x}c$, $w = \langle q\rangle\widetilde{w}\mathtt{0}$ for some state q, by Lemma 6 they are of this form. Then $x' = \widetilde{x}(c+1)$, $w' = \mathtt{0}\widetilde{w}\langle q\rangle$ by the same lemma. □

Let us now proceed with specifying the action of Move, which represents symbol manipulation, head movement and state change of a Turing machine according to the membership of states and symbols specified in δ. Generally, when $\text{Move}_{q,a,q',a'}$ is applied to a valid configuration, it computes the *preceding configuration* of the machine. This configuration is unique because of the restriction built in $\text{Move}_{q,a,q',a'}$ in its subscripts. The symbols and states used as the subscript restrict its applicability to the following case: in the current configuration the machine is in state q and the symbol to the left rewritten at the previous step is a, while in the previous configuration the machine was in state q' and scanned the symbol a'. For all other configurations and in all other cases, the function produces the empty set.

Lemma 8. *Let* $q, q' \in Q$ *and* $a, a' \in \Gamma$*. Let* $x \in \text{Counter}$ *and* $w = \widehat{w}\mathtt{0}\langle a\rangle\langle q\rangle\widetilde{w} \in \text{Tape}$ *for some* $\widehat{w} \in (\mathtt{0}\langle\Gamma\rangle)^*$ *and* $\widetilde{w} \in (\langle\Gamma\rangle\mathtt{0})^*$*. Then* $\text{Move}_{q,a,q',a'}(x\mathtt{55}w) = x\mathtt{55}\widehat{w}\langle q'\rangle\langle a'\rangle\mathtt{0}\widetilde{w}$*.*

For every string $\alpha \in \Sigma^*$ *of a different form,* $\text{Move}_{q,a,q',a'}(\alpha) = \varnothing$*.*

Proof. We deal with fixed a', q', a, q. The inner subexpression of $\text{Move}_{q,a,q',a'}$,

$$x\mathtt{55}w \cap \text{Counter}\,\mathtt{55}\,\text{Tape}_{aq},$$

ensures that $w = \widehat{w}\mathtt{0}\langle a\rangle\langle q\rangle\widetilde{w}$ for some $\widehat{w} \in (\mathtt{0}\langle\Gamma\rangle)^*$ and $\widetilde{w} \in (\langle\Gamma\rangle\mathtt{0})^*$ Also $\text{Move}_{q,a,q',a'}$ is empty for w of a different form.

The next subexpression performs the operation

$$x\mathtt{55}w + \big(\langle q'\rangle\langle a'\rangle\mathtt{0} \boxminus \langle a\rangle\langle q\rangle\big)(\mathtt{00})^* \cap \text{Counter}\,\mathtt{55}\,\text{Tape}_{q'a'}$$

Let

$$y = \big(\langle q'\rangle\langle a'\rangle\mathtt{0} \boxminus \langle a\rangle\langle q\rangle\big)\mathtt{0}^{2k} \in \big(\langle q'\rangle\langle a'\rangle\mathtt{0} \boxminus \langle a\rangle\langle q\rangle\big)(\mathtt{00})^*$$

and consider the string $x'\mathtt{55}w' = x\mathtt{55}w + y$, where $x' \in \mathrm{Counter}$ and $w' \in \mathrm{Tape}$. By Lemma 3, $|w'| = |w|$.

As y has non-zero digits only in positions $2k+1, 2k+2, 2k+3$, while the digit $|\Sigma| - 1$ is not a valid encoding of any symbol, adding y cannot change any digit in position greater than $2k+4$ in $x\mathtt{55}w$.

Let $2\ell = |\widetilde{w}|$. Since $q \neq q'$, by the technical assumption that the machine does not stay in the same state, then w and w' differ at the position $2\ell + 1$. Therefore, $2\ell + 1 = 2k + 3$ or $2\ell + 1 = 2k + 1$.

Suppose $2\ell+1 = 2k+3$. Then $x\mathtt{55}w+y$ on position number $2\ell+1 =$ has digit $(\langle q'\rangle + \langle q\rangle) \mod |\Sigma|$ or $(\langle q'\rangle + \langle q\rangle + 1) \mod |\Sigma|$ ($+1$ is possible due to carry). Since $\langle q\rangle, \langle q'\rangle \leqslant 6+|Q|$ and $q \neq q'$, it follows that $\langle q\rangle + \langle q'\rangle \leqslant 11+|Q|$ and $\langle q\rangle + \langle q'\rangle + 1 \leqslant 12 + |Q|$. Each sum is smaller than $|\Sigma|$ and is therefore represented by a single digit. Each of them is greater than $\langle q'\rangle$. Hence both are filtered out by the intersection with $\mathrm{Counter}\,\mathtt{55}\,\mathrm{Tape}_{q'a'}$.

Suppose $2\ell + 1 = 2k + 1$. Then $x' = x$ and $w' = \widehat{w}\langle q'\rangle\langle a'\rangle\mathtt{0}\widetilde{w}$, as stated in the lemma. □

Similarly to Lemma 7, reversibility of $\mathrm{Move}_{q,a,q',a'}$ directly follows from Lemma 8.

Lemma 9. *Let $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(x'\mathtt{55}w')$. Then $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$ and $w' = \widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$ for some $\widehat{w} \in (\mathtt{0}\langle\Gamma\rangle)^*$ and $\widetilde{w} \in (\langle\Gamma\rangle\mathtt{0})^*$, and $x = x'$.*

Proof. By Lemma 8, $w' = \widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$ for some $\widehat{w} \in (\mathtt{0}\langle\Gamma\rangle)^*$ and $\widetilde{w} \in (\langle\Gamma\rangle\mathtt{0})^*$, otherwise $\mathrm{Move}_{q',a',q,a}(x\mathtt{55}w') = \varnothing$. Then, by the same lemma, $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$ and $x = x'$. □

The flow control of the alternating Turing machine includes existential and universal nondeterminism in the corresponding states, and a single step is in fact a disjunction or conjunction of several transitions as specified in Move. This logic is transcribed in the expression $\mathrm{Step}(X)$, which computes the set of all *previous configurations*, from which machines in a universal state make all their transitions to configurations in X and machines in an existential state make at least one of their transitions to some configuration in X. This implements one step of the computation of the machine, backwards.

Lemma 10. *Let $x \in$ Counter and $w \in$ Tape, let $q \in Q$ be the state encoded in w. Then $x\mathtt{55}w \in \mathrm{Step}(X)$ if and only if*

- *the configuration w has the head* not *in the position beyond the rightmost symbol, that is, $w = \widehat{w}\langle q\rangle\widetilde{w}\mathtt{0}$ for some $\widehat{w}, \widetilde{w} \in \Sigma^*$.*
- *if $q \in Q_E$, then for some string w' encoding next configuration of the ATM there holds $x\mathtt{55}w' \in X$.*

- *if $q \in Q_A$, then for every string w' encoding next configuration of the ATM there holds $x\mathtt{55}w' \in X$.*

Proof. Let a be the symbol under the head of the machine in the configuration w.

⇒ Consider the definition of Step:

$$\mathrm{Step}(X) = \Big(\bigcup_{\widehat{q} \in Q_E, \widehat{a} \in \Gamma} \ \bigcup_{(q',a') \in \delta(\widehat{q},\widehat{a})} \mathrm{Move}_{q',a',\widehat{q},\widehat{a}}(X) \Big) \cup \Big(\bigcup_{\widehat{q} \in Q_A, \widehat{a} \in \Gamma} \ \bigcap_{(q',a') \in \delta(\widehat{q},\widehat{a})} \mathrm{Move}_{q',a',\widehat{q},\widehat{a}}(X) \Big).$$

Let $x\mathtt{55}w \in \mathrm{Step}(X)$ and first suppose that q is an existential state. By Lemma 9,

$$x\mathtt{55}w \notin \mathrm{Move}_{q',a',\widehat{q},\widehat{a}}(X)$$

for $(\widehat{q},\widehat{a}) \neq (q,a)$, and therefore

$$x\mathtt{55}w \in \bigcup_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X),$$

that is, there exist $q' \in Q$ and $a' \in \Sigma$ with $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(X)$. Then, since $\mathrm{Move}_{q',a',q,a}$ is distributive over infinite union (Lemma 1), there exists a number $n \in X$ with $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(\{n\})$. By Lemma 8, n must be of the form $x\mathtt{55}w'$ with $w' \in$ Tape. Applying Lemma 9 to $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(\{x\mathtt{55}w'\})$, one obtains that $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$ and $w' = \widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$. Since $(q',a') \in \delta(q,a)$, w' is a configuration next to w, and $x\mathtt{55}w' \in X$. Clearly, the position of the head in w is left to the right-most symbol, as $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$.

The case of $q \in Q_A$ is similar. It follows from $x\mathtt{55}w \in \mathrm{Step}(X)$ that

$$x\mathtt{55}w \in \bigcap_{(q',a') \in \delta(q,a)} \mathrm{Move}_{q',a',q,a}(X),$$

that is, for all $q' \in Q$ and $a' \in \Sigma$ with $(q',a') \in \delta(q,a)$ it holds that $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(X)$. As in the previous case, this implies that $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$ and there is $x\mathtt{55}w'_{q',a'} \in X$ with $w'_{q',a'} = \widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$. These are two consecutive configurations, and every configuration next to w is of this form for some $(q',a') \in \delta(q,a)$. Then the required element $x\mathtt{55}\widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$ is in X for all q' and a' with $(q',a') \in \delta(q,a)$. Also note that, by assumption, there is at least one such pair (q',a'), hence w is of the required form with the head not beyond the right-most symbol.

⇐ Let $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$ and first consider the case of $q \in Q_E$. Let w' be one of the next configurations of the machine with $x\mathtt{55}w' \in X$. Then $w' = \widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$ for some $(q',a') \in \delta(q,a)$. By Lemma 8, $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(x\mathtt{55}w')$. Since $\mathrm{Move}_{q',a',q,a}(x\mathtt{55}w') \subseteq \mathrm{Step}(X)$, this shows that $x\mathtt{55}w \in \mathrm{Step}(X)$.

If $q \in Q_A$, then, by assumption, $x\mathtt{55}w' \in X$ for all configurations w' immediately following w. That is, for all $(q',a') \in \delta(q,a)$, $x\mathtt{55}w'_{q',a'} \in X$, where $w'_{q',a'} = \widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w}$. For every such pair, by Lemma 8, $x\mathtt{55}w \in \mathrm{Move}_{q',a',q,a}(x\mathtt{55}w'_{q',a'})$. Hence,

$$x\mathtt{55}w \in \bigcap_{(q',a')\in\delta(q,a)} \mathrm{Move}_{q',a',q,a}(X),$$

and therefore $x\mathtt{55}w \in \mathrm{Step}(X)$. □

Having established the formal meaning of the auxiliary operations, let us return to the equations. The equation for X states that a configuration leads to acceptance if and only if it is accepting itself (Final), or one can directly proceed from it to a configuration leading to acceptance ($\mathrm{Step}(X)$), or that it is a configuration obtained in Y. The equation for Y specifies circular rotation of the tape by $\mathrm{Jump}(X)$ and implements iterated carry propagation as in Lemma 5 by a self-reference $\mathrm{Carry}(Y)$. Altogether, the least solution of these equations corresponds to the computation of the machine as follows:

Lemma 11. *Let (L_X, L_Y) be the least solution of the equations* (3)–(4)*.*

- (⇒) *Let $x \in$ Counter, $w \in$ Tape and $x\mathtt{55}w \in L_X$. Then M accepts starting from the configuration represented by w.*

- (⇐) *Conversely, if M accepts starting from the configuration represented by $w \in$ Tape, and the longest path in the tree of the accepting computation has length ℓ, then for each $x \in$ Counter with $\mathrm{Value}(x) \geqslant \ell$, there holds $x\mathtt{55}w \in L_X$.*

Proof. The least solution of the system is computed by fixpoint iteration (2), Denote by $L_X^{(k)}$ and $L_Y^{(k)}$ the X- and Y-components of the vector $\varphi^k(\varnothing, \ldots, \varnothing)$ obtained after $k \geqslant 0$ iterations. Then $x\mathtt{55}w \in L_X$ if and only if $x\mathtt{55}w \in L_X^{(k)}$ for some $k \geqslant 1$.

(⇒) Assume that $x\mathtt{55}w \in L_X$, that is, $x\mathtt{55}w \in L_X^{(k)}$. It has to be proved that w encodes a configuration from which the machine accepts. The proof is an induction on k.

By (3), $x\mathtt{55}w \in L_X^{(k)}$ means that either $x\mathtt{55}w \in$ Final, or $x\mathtt{55}w \in \mathrm{Step}(L_X^{(k-1)})$ or $x\mathtt{55}w \in L_Y^{(k-1)}$.

If $x\mathtt{55}w \in$ Final, then clearly w represents an accepting configuration, as the Turing machine is already in an accepting state.

Let $x\mathtt{55}w \in \mathrm{Step}(L_X^{(k-1)})$, and let $w = \widehat{w}\langle q\rangle\langle a\rangle\mathtt{0}\widetilde{w}$; the configuration is of this form by Lemma 10. Consider the set of numbers $S = \{x\mathtt{55}\widehat{w}\mathtt{0}\langle a'\rangle\langle q'\rangle\widetilde{w} \mid (q',a') \in \delta(q,a)\}$ representing all next configurations of the machine. Suppose that $q \in Q_A$ ($q \in Q_E$, respectively). Then, by Lemma 10, all numbers in S (at least one number from S, respectively) are in $L_X^{(k-1)}$. By the induction hypothesis, all numbers from $L_X^{(k-1)} \cap S$ represent configurations that have accepting computations. Hence w also represents a configuration with an accepting computation, as all (at least one, respectively) its consecutive configurations have accepting computations.

If $x\mathsf{55}w \in L_Y^{(k-1)}$, then let $x_{k-1} = x$ and construct a sequence $\{x_i\}_{i=k-1,k-2,\ldots}$, where $x_i\mathsf{55}w \in L_Y^{(i)}$ for all $i \leqslant k-1$ and $x_{i+1}\mathsf{55}w = \text{Carry}(x_i\mathsf{55}w)$ for all $i < k-1$. For every $i \leqslant k-2$, since $x_{i+1}\mathsf{55}w \in L_Y^{(i+1)}$, then either $x_{i+1}\mathsf{55}w \in \text{Carry}(L_Y^{(i)})$ or $x_{i+1}\mathsf{55}w \in \text{Jump}(L_X^{(i)})$. In the former case, by Lemma 1, there exists $x'\mathsf{55}w' \in L_Y^{(i)}$ with $\{x_{i+1}\mathsf{55}w_{i+1}\} = \text{Carry}(\{x'\mathsf{55}w'\})$. According to Lemma 4, the latter means that $w' = w$. Then $x_i = x'$ forms the next element of the sequence.

In the latter case, if $x_{i+1}\mathsf{55}w \in \text{Jump}(L_X^{(i)})$, Lemma 1 similarly implies that there is $x'\mathsf{55}w' \in L_X^{(i)}$ with $\{x_{i+1}\mathsf{55}w\} = \text{Jump}(\{x'\mathsf{55}w'\})$. According to Lemma 7, $w' = \langle q\rangle\widetilde{w}\mathsf{0}$ and $w = \mathsf{0}\widetilde{w}\langle q\rangle$, that is, the machine jumps from w to w'. Since, by the induction hypothesis, w' represents a configuration with an accepting computation, so does w.

It is left to mention that the case of $x_{i+1}\mathsf{55}w \in \text{Jump}(L_X^{(i)})$ in the above proof eventually occurs, since otherwise the sequence would continue until $x_0\mathsf{55}w \in L_Y^{(0)} = \varnothing$, which is impossible.

$\textcircled{\Leftarrow}$ Consider the converse statement. That is, we are given a configuration w that has an accepting computation with the longest path of length ℓ and want to prove that $x\mathsf{55}w \in L_X$ for x with value greater than ℓ. We proceed by induction on ℓ.

If $\ell = 0$, then w represents a configuration in the accepting state and therefore, by the equation for X in the system, $x\mathsf{55}w \in \text{Final} \subseteq L_X$ for all $x \in \text{Counter}$.

Now assume w has an accepting computation with length $\ell+1$. Suppose first that $w = \widehat{w}\langle q\rangle\langle a\rangle\mathsf{0}\widetilde{w}$ with $\widehat{w}, \widetilde{w} \in \Sigma^*$, $a \in \Sigma$ and $q \in Q$, that is, the configuration denoted by w has the head anywhere except in the position beyond the right-most symbol. Assume that $q \in Q_A$ ($q \in Q_E$, respectively). Then, by definition, all words (at least one word, respectively) representing next configurations of ATM have accepting computations of length at most ℓ. Hence all words (at least one word, respectively) of the form $x\mathsf{55}w'$ are in L_X, where $x \in \text{Counter}$ represents counter of value at least n and w' represents a configuration next to w. Then, by Lemma 10, $x\mathsf{55}w \in \text{Step}(L_X)$. By the equation for X, we obtain $x\mathsf{55}w \in L_X$, as desired.

Consider the case of the head of the Turing machine in the position beyond the right-most symbol, and let $w = \mathsf{0}\widetilde{w}\langle q\rangle$. Let w' be the next configuration, that is, $w' = \langle q\rangle\widetilde{w}\mathsf{0}$. Let $x' \in \text{Counter}$ represent a counter of value at least ℓ. Then, by the induction assumption, $x'\mathsf{55}w' \in L_X$. Hence, by Lemma 6, there is a word $x''\mathsf{55}w \in \text{Jump}(L_X) \subseteq L_Y$, where $x'' \in \text{Counter}' \cup \text{Counter}$ and $\text{Value}(x'') = \text{Value}(x') + 1$. Then, by Lemma 5, $x'''\mathsf{55}w \in L_Y$, where x''' is the unique element of Counter such that $\text{Value}(x'') = \text{Value}(x''')$. Then, by (3), also $x'''\mathsf{55}w \in L_X$, as desired. □

It remains to observe that the number of steps of the machine is exponentially bounded, hence the acceptance of a word by the machine is represented by the following number in the least solution of the constructed system:

Main Lemma. *ATM M accepts a string $a_1 \dots a_n \in \Omega^+$ if and only if*

$$\mathtt{1}^{2+\log n+\log(|\Gamma|)n+\log(|Q|)}\mathtt{55}\langle q_0\rangle\langle a_1\rangle\mathtt{0}\langle a_1\rangle\mathtt{0}\dots\langle a_n\rangle\mathtt{0} \in L_X.$$

Proof. The initial configuration of M on $a_1 \dots a_n$ is represented by the sequence of digits $w = \langle q_0\rangle\langle a_1\rangle\mathtt{0}\langle a_1\rangle\mathtt{0}\dots\langle a_n\rangle\mathtt{0} \in \text{Tape}$.

⇒⃝ If M accepts starting from this configuration, then the longest path in the accepting computation consists of at most

$$|n+1| \cdot |Q| \cdot |\Gamma|^n \leqslant 2^{\log n+1} \cdot 2^{\log(|Q|)+\log(|\Gamma|)n} < 2^{2+\log n+\log(|\Gamma|)n+\log(|Q|)} - 1.$$

steps, since all configurations forming this path must be different. Then, by Lemma 11, for $x = \mathtt{1}^{2+\log(|w|)+\log(|\Gamma|)|w|+\log(|Q|)}$ with $\text{Value}(x) = 2^{2+\log(|w|)+\log(|\Gamma|)|w|+\log(|Q|)} - 1$ it holds that $x\mathtt{55}w \in L_X$.

⇐⃝ Conversely, if there exists $x \in \text{Counter}$ with $x\mathtt{55}w \in L_X$, then, according to Lemma 11, M accepts starting from the configuration w. □

Proof of Theorem 2. The system of equations constructed above has an EXPTIME-complete least solution.

To see that the least solution of every system is in EXPTIME, it is sufficient to represent it as a conjunctive grammar over a unary alphabet. Then, given a number n, its membership in the least solution can be tested by supplying the string a^n to a known cubic-time parsing algorithm for conjunctive grammars [12]. Its time is cubic in n, hence exponential in the length of the binary notation of n. □

Having established a solution complexity theorem for equations over sets of numbers, let us discuss its implications on conjunctive grammars over a one-letter alphabet.

Every conjunctive language is in P [12], and some conjunctive languages over a multiple-letter alphabet are known to be P-complete [14]. The case of a unary alphabet is special, as it is known that no sparse language, in particular no unary language, can be P-complete unless DLOGSPACE = P [11, 1], that is, unless the notion of P-completeness is trivial. However, from Theorem 2 one can infer the following result slightly weaker than P-completeness:

Corollary 1. *There exists a EXPTIME-complete set of numbers $S \subseteq \mathbb{N}$, such that the language $L = \{a^n \mid n \in S\}$ of unary notations of numbers from S is generated by a conjunctive grammar.*

Note that for every unary language generated by a conjunctive grammar, the corresponding set of numbers is in EXPTIME. The set constructed in Corollary 1 can thus be regarded as the computationally hardest among unary conjunctive languages.

A simple consequence of Corollary 1 refers to the complexity of parsing for conjunctive grammars.

Corollary 2. *Unless* PSPACE = EXPTIME*, there is no logarithmic-space parsing algorithm for conjunctive languages over a unary alphabet.*

4 The membership problem

Consider the general membership problem for our equations, stated as follows: "Given a system $X_i = \varphi_i(X_1, \ldots, X_m)$ and a number n in binary notation, determine whether n is in the first component of the least solution of the given system". Its complexity is now easy to establish.

Theorem 3. *The membership problem for resolved systems of equations over sets of numbers with operations $\{\cup, \cap, +\}$ is EXPTIME-complete.*

Proof. Membership in EXPTIME. The algorithm begins with representing the given system as a conjunctive grammar over a unary alphabet, with a linearly bounded blow-up. The given number n is represented as a string a^n with an exponential blow-up. Then it is sufficient to apply the known polynomial-time algorithm for solving the membership problem for conjunctive grammars [13].

The EXPTIME-hardness of the general membership problem immediately follows from Theorem 2 by fixing the system of equations. □

Let us conclude by comparing the complexity of the membership problem for expressions, circuits and equations, as well as the families of sets representable by their solutions. All known results are given in Table 1.

	Representable sets	Membership problem
expressions with $\{\cup, +\}$	Finite	NP-complete [17]
circuits with $\{\cup, +\}$	Finite	NP-complete [4, 9, 10]
equations with $\{\cup, +\}$	Ultimately periodic	NP-complete [4]
expressions with $\{\cup, \cap, +\}$	Finite	PSPACE-complete [17]
circuits with $\{\cup, \cap, +\}$	Finite	PSPACE-complete [9, 10]
equations with $\{\cup, \cap, +\}$	$\subsetneq$ ***EXPTIME***, contains ***EXPTIME-complete*** *set*	***EXPTIME-complete***

Table 1: Comparison of formalisms over sets of integers.

The new complexity results for the equations over sets of numbers naturally fit into the framework of the existing research. On the other hand, the new results on the expressive power of equations come in a sharp contrast with the previous work: these equations can represent non-trivial sets of numbers, which are computationally as hard as the general membership problem for this class.

It remains an open question, what is the exact family of sets of natural numbers defined by these equations. For instance, is it possible to represent the set of all primes?

Acknowledgements

Research of the first author supported by MNiSW grant number N206 024 31/3826, 2006–2008. Research of the second author supported by the Academy of Finland under grant 118540.

The research was done during the visit to Turku of the first author, supported by the European Science Foundation short visit grant as part of the project "Automata: from Mathematics to Applications", reference number 1763.

References

[1] J.-Y. Cai, D. Sivakumar, "Sparse hard sets for P: resolution of a conjecture of Hartmanis". *Journal of Computer and System Sciences*, 58:2 (1999), 280–296.

[2] A. K. Chandra, D. C. Kozen, L. J. Stockmeyer, "Alternation", *Journal of the ACM*, 28:1 (1982) 114–133.

[3] S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.

[4] D. T. Huynh, "Commutative grammars: the complexity of uniform word problems", *Information and Control*, 57:1 (1983), 21–39.

[5] A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *DLT 2007* (Turku, Finland, July 3–6, 2007), LNCS 4588, 242–253.

[6] A. Jeż, A. Okhotin, "Conjunctive grammars over a unary alphabet: undecidability and unbounded growth", *Computer Science in Russia* (CSR 2007, Ekaterinburg, Russia, September 3–7, 2007), LNCS 4649, 168–181.

[7] M. Kunc, "The power of commuting with finite sets of words", *Theory of Computing Systems*, 40:4 (2007), 521–551.

[8] M. Kunc, "What do we know about language equations?", *Developments in Language Theory* (DLT 2007, Turku, Finland, July 3–6, 2007), LNCS 4588, 23–27.

[9] P. McKenzie, K. Wagner, "The complexity of membership problems for circuits over sets of natural numbers", *20th Annual Symposium on Theoretical Aspects of Computer Science* (STACS 2003, Berlin, Germany, February 27–March 1, 2003), LNCS 2607, 571–582.

[10] P. McKenzie, K. Wagner, "The complexity of membership problems for circuits over sets of natural numbers", *Computational Complexity*, 16 (2007), to appear.

[11] M. Ogihara, "Sparse hard sets for P yield space-efficient algorithms", *Chicago J. Theor. Comput. Sci.*, 1996.

[12] A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.

[13] A. Okhotin, "A recognition and parsing algorithm for arbitrary conjunctive grammars", *Theoretical Computer Science*, 302 (2003), 365–399.

[14] A. Okhotin, "The hardest linear conjunctive language", *Information Processing Letters*, 86:5 (2003), 247–253.

[15] A. Okhotin, "Decision problems for language equations with Boolean operations", *Automata, Languages and Programming* (ICALP 2003, Eindhoven, The Netherlands, June 30–July 4, 2003), LNCS 2719, 239–251.

[16] A. Okhotin, "Unresolved systems of language equations: expressive power and decision problems", *Theoretical Computer Science*, 349:3 (2005), 283–308.

[17] L. J. Stockmeyer, A. R. Meyer, "Word problems requiring exponential time", *STOC 1973*, 1–9.

[18] K. Yang, "Integer circuit evaluation is PSPACE-complete", *Computational Complexity 2000*, 204–211.

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku

- Department of Information Technology
- Department of Mathematical Sciences

Åbo Akademi University

- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration

- Institute of Information Systems Sciences

ISBN 978–952–12–1966–5
ISSN 1239-1891