A FINITE SET OF FUNCTIONS WITH AN EXPTIME-COMPLETE COMPOSITION PROBLEM

MARCIN KOZIK

Abstract. We exhibit a finite family of functions over a finite set (i.e. a finite algebra) such that the problem whether a given function can be obtained as a composition of the members of this family (i.e. is a member of the clone generated by the algebra) is EXPTIME-complete.

1. Introduction

The class of EXPTIME-complete problems is well known for containing problems of finding the optimal strategies for chess [FL81], checkers [Rob84] as well as some versions of go (comp. [LS80], [BW94]). Another set of EXPTIME-complete problems consists of problems derived from questions of smaller complexity by means of succinct circuits which allow to present an input to the algorithms in exponentially smaller space (comp. [Bal96]). It was shown by H. Friedman about 1985 and published by Bergman, Juedes and Slutzki in [BJS99] that it is an EXPTIME-complete problem to decide whether a function can be obtained as a composition of given functions.

The problem of composition of functions was studied for a number of reasons. Finding a constant function as a composition of a given set of functions over a finite set is connected to Černý's Conjecture [Čer64] and was proved to be solvable in a polynomial time in [GJ85]. On the other hand Dexter Kozen had proved in [Koz77] that for a finite family of unary functions (over a finite set) with one distinguished member, it is a PSPACE-complete problem to decide whether the distinguished function can be obtained as a composition of the others. The mentioned above result of H. Friedman shows that allowing non-unary functions causes the problem to become EXPTIME-complete.

The problem of composition of functions has a very natural algebraic description. The result of H. Friedman describes the computational complexity of the following problem.

INPUT	a finite algebra $\mathbf{A}$ and a function $f : A^k \to A$
PROBLEM	decide if $f \in \mathrm{Clo}(\mathbf{A})$.

This problem splits into subproblems – for a fixed, finite algebra $\mathbf{A}$ we define

INPUT	a function $f : A^k \to A$
PROBLEM	decide if $f \in \mathrm{Clo}(\mathbf{A})$.

None of the such defined subproblems is more complex then the original problem. The natural question is how much easier are they. Does the complexity of the first problem rely on the fact that the algebra can vary? Can we find a fixed structure such that the connected subproblem remains complex? How complex is the problem of a membership in a finitely generated clone? This question follows the line of research devoted to describing the computational complexity of natural algebraic problems generated by fixed, finite structures. Such problems are studied extensively and a number of results is already known (comp. [Szé02], [JM06], [KK]).

The research was supported by Eduard Čech Center grant no. LC505.

The main theorem of our paper answers this question. We construct a finite family of binary functions (over a finite set) such that the problem whether a given function can be obtained as a composition of functions from this fixed family is EXPTIME-complete. This proves that the membership problem for a finitely generated clone can be EXPTIME-complete i.e. as complex as the general problem. Moreover our construction allows us also to restate the result of [BJS99] considering the computational complexity of the TERM-EQ problem in a more restrictive way.

In Section 4 we reprove the result of Dexter Kozen [Koz77] by showing that the computational complexity of a composition of unary functions is PSPACE-complete. In Section 5 we refine the construction to obtain the result of Friedman for the functions of higher arities. The alternative, and more formal, proof of this fact can be found in [BJS99]. Finally, in Section 6, we build on the intuition from previous sections to produce a finite family of binary functions such that the question whether a given function can be obtained as a composition of these binary functions is EXPTIME-complete. In Section 7 we briefly address the consequences of our result to the TERM-EQ problem studied in [BJS99].

2. Definitions

The following definitions allow us to state the problems formally. We define a set of functions obtained as compositions of unary functions in the following way.

Definition 2.1. *For a given set $\mathcal{U}$ and functions $f_0, \dots, f_n : \mathcal{U} \to \mathcal{U}$ we define the set of functions generated by $f_0, \dots, f_n$ to be*

$$\{\mathrm{id}_{\mathcal{U}}\} \cup \{s : \mathcal{U} \to \mathcal{U} \mid s = f_{a_0} \circ \dots \circ f_{a_k} \text{ where } a_i \leq n \text{ for all } i \leq k\}.$$

and denote it by $\mathrm{Clo}_1^{\mathcal{U}}(f_0, \dots, f_n)$.

In such a case the problem of of composition of unary functions follows.

Composition of unary functions. *Given a finite set $\mathcal{U}$ and a number of functions $s, f_0, \dots, f_n : \mathcal{U} \to \mathcal{U}$ decide whether $s \in \mathrm{Clo}_1^{\mathcal{U}}(f_0, \dots, f_n)$.*

In the case of composition of functions of higher arity we follow a standard notation which can be found in [BS81]. We define a composition of functions in the following way:

Definition 2.2. *For a set of functions $f_0, \dots, f_n$ such that $f_i : \mathcal{U}^{k_i} \to \mathcal{U}$ is the minimal set X such that*

1. *for any number m and any $i < m$ the function $f : \mathcal{U}^m \to \mathcal{U}$ defined to be $f(a_0, \dots, a_{m-1}) = a_i$ is a member of X, and*
2. *for any $i \leq n$ and any $g_0, \dots, g_{k_i - 1}$ members of X the function*

$$f_i(g_0, \dots, g_{k_i - 1}) \text{ is a member of } X.$$

we denote by $\mathrm{Clo}^{\mathcal{U}}(f_0, \dots, f_n)$.

The general problem of composition of functions is defined in the following way.

Composition of functions. *Given a finite set $\mathcal{U}$ together with a number of functions $s, f_0, \dots, f_n$ such that $f_i : \mathcal{U}^{k_i} \to \mathcal{U}$ and $s : \mathcal{U}^k \to \mathcal{U}$ decide whether $s \in \mathrm{Clo}^{\mathcal{U}}(f_0, \dots, f_n)$.*

Note that, according to Definitions 2.1, 2.2, projections are always members of the generated set. This assumption is introduced for the clarity of presentation and does not influence computational complexity of problems considered in this paper.

3. Notation and preliminaries

We identify natural numbers with their binary representations of a fixed length. That is, for a fixed n, a number between 0 and $2^n - 1$ is a word of length n over the alphabet $\{0, 1\}$ and the set of all such words is denoted by $\{0, 1\}^n$. Thus addition and substraction are partial functions on such words. We occasionally use regular expresions to denote a certain family of numbers i.e. 1^*0 is the set of all even numbers between 0 and $2^n - 1$.

For two words u and v we denote by $u \cdot v$ or simply uv a catenation of these two words. For a given word w we denote by $w(k)$ the k-th letter of w, where k is taken from the interval between 0 and $|w| - 1$ and $|w|$ denotes the number of letters in w.

All of the constructions in this paper are based on Turing machines. We usually denote a Turing machine by $\mathbf{T}$, its alphabet by $\mathcal{A}$ and the set of internal states by $\mathcal{S}$. We will denote the states of the machine by lowercase Greek letters with the starting state denoted by α and the accepting state denoted by ω. The instructions of the machine are five-tuples of the form $\beta a b D \gamma$, for $\beta, \gamma \in \mathcal{S}, a, b \in \mathcal{A}$ and $D = R$ or $D = L$ meaning: "while in state β reading a, write b move in the direction D and change the state to γ".

All of the computations of the machine $\mathbf{T}$ will take place on a bounded tape and we use two special symbols $\lozenge$ and $\blacklozenge$, taken from $\mathcal{A}$, to denote the leftmost and the rightmost, respectively, position on a tape. A configuration of the machine $\mathbf{T}$ is a triple $\langle \mathbf{w}, i, \beta \rangle$ where $\mathbf{w}$ is the word of fixed length denoting the tape of the machine $\mathbf{T}$, i is the number between 0 and $|\mathbf{w}| - 1$ describing the position of the head on a tape and β is an internal state of the machine $\mathbf{T}$.

We find it usefull to work with the formal expressions defining functions, and we follow [BS81] in defining them.

Definition 3.1. *For a set of functional symbols $f_0, \ldots, f_n$ of arities $k_0, \ldots, k_n$, and a set of variables $\{x, y, \ldots\}$ we define* a term *in a recursive way*

(1) *any variable is a term, and*
(2) *for any $i \leq n$ and any terms $g_0, \ldots, g_{k_i - 1}$ the formal expresion*

$$f_i(g_0, \ldots, g_{k_i - 1}) \text{ is a term as well, and}$$

(3) *all the terms can be obtained in such a way.*

Thorough the paper we often abuse the notation by identifying the functional symbols denoting the operations and the functions themselves. The distinctions is always clear from the context.

It is a trivial observation that, having a fixed set of generating functions and a corresponding set of terms, each term defines a function, and each function that can be obtained as a composition is defined by some term. The notion of a subterm is intuitive – if we consider a term to be a labelled tree then each term being a labelled subtree of our term is its subterm. For a more involved study of algebraic concepts we refer the reader to [BS81] and [MMT87].

We introduce one more notion which allows for easier proofs of our results. We define a restricted composition of functions.

Restricted composition of functions. *Given sets $\mathcal{U}$ and $\mathcal{V}$ such that $\mathcal{V} \subseteq \mathcal{U}$ and functions $f_0, \ldots, f_n$ such that $f_i : \mathcal{U}^{k_i} \to \mathcal{U}$ and $s : \mathcal{U}^{k-1} \times \mathcal{V} \to \mathcal{U}$ decide whether there exists a function $s' : \mathcal{U}^k \to \mathcal{U}$ such that*

$$s' \in \mathrm{Clo}^{\mathcal{U}}(f_0, \ldots, f_n) \text{ and } s'_{|\mathcal{U}^{k-1} \times \mathcal{V}} = s.$$

The following proposition is a trivial consequence of the definitions.

Proposition 3.2. *The problem of composition of (unary) functions and of a restricted compositions of (unary) functions are computationally equivalent.*

Proof. It is obvious that a problem of a composition reduces to the problem of a restricted composition of functions. For the reduction in the other direction let us fix an instance of the restricted composition problem with the set $\mathcal{U}$, functions $f_0, \dots, f_n$, subset $\mathcal{V}$ and a function $s : \mathcal{U}^{k-1} \times \mathcal{V} \to \mathcal{U}$ as in the definition of a restricted composition. We construct a set $\mathcal{W} = \mathcal{U} \cup \mathcal{V}' \cup \{\perp\}$ which is a disjoint union of the three sets. Set $\mathcal{V}'$ consists of copies of the elements of $\mathcal{V}$ and there is a bijection $b \mapsto b'$ between $\mathcal{V}$ and $\mathcal{V}'$. We define the functions:

$$g_i(a_0, \dots, a_{k_i-1}) = \begin{cases} f_i(a_0, \dots, a_{k_i-1}) & \text{if } a_j \in \mathcal{U} \text{ for all } j < k_i \\ \perp & \text{else} \end{cases}$$

for all $i \leq n$, and an additional function

$$g_{n+1}(a) = \begin{cases} b & \text{if } a = b' \in \mathcal{V}' \\ \perp & \text{else} \end{cases}.$$

Finally we put

$$s''(a_0, \dots, a_{k-1}) = \begin{cases} s(a_0, \dots, a_{k-2}, a) & \text{if } a_0, \dots, a_{k-2} \in \mathcal{U} \text{ and } a_{k-1} = a' \in \mathcal{V}' \\ \perp & \text{else.} \end{cases}$$

The construction immediately implies that $s'' \in \mathrm{Clo}^{\mathcal{U}}(g_0, \dots, g_n, g_{n+1})$ if and only if the restricted composition problem for s and $f_0, \dots, f_n$ had a positive solution. This proves the reduction of the restricted composition problem to the composition problem. □

Note that the modification of the set $\mathcal{U}$ and the family of generating functions in the proof above depends only on the set $\mathcal{V}$. This fact will allow us to use the same reasoning while constructing a fixed family of functions which generate the set of functions with an EXPTIME-complete membership problem.

4. Composition of unary functions is PSPACE-complete

The results of this section have been proved by Kozen in [Koz77]. We reprove them to introduce the technique that is going to be applied in the following sections.

It is an easy fact that a non-deterministic Turing machine can solve a composition problem in a polynomial space. Thus, since by [Sav70] NPSPACE is equal to PSPACE, there exists a deterministic Turing machine solving the same problem in a polynomial space as well.

To put a lower bound on the computational complexity of the problem we fix a Turing machine $\mathbf{T}$ with the following properties:

(1) the language accepted by $\mathbf{T}$ is PSPACE-complete,
(2) for a given word w the machine $\mathbf{T}$ starts its computation on the first symbol of a tape equal to $\Diamond w \blacklozenge$ and never leaves it while computing,
(3) the machine accepts the word w if and only if it finishes the computations on $\Diamond$ on a tape consisting of $\Diamond w \blacklozenge$.

For a given word w we define an instance of a restricted composition problem equivalent to the acceptance of w by a machine $\mathbf{T}$.

Lemma 4.1. *The problem of acceptance of a word w by a machine $\mathbf{T}$ reduces to the restricted unary composition problem.*

Proof. For a given word w we define the set $\mathcal{U}$ to be the disjoint union of four sets

$$\{0, \dots, |w| + 1\} \times \mathcal{A} \cup \mathcal{S} \cup \{0, \dots, |w| + 1\} \cup \{\perp\}.$$

The sets $\{0, \dots, |w| + 1\}$ and $\mathcal{S}$ allow us to track the placement of the head and the internal state of the machine during the computations. The function of the

elements of $\{0,\ldots,|w|+1\}\times\mathcal{A}$ is to model the tape – the first coordinate describes the index on the tape, and the second a letter that is written there.

For each instruction $\beta abR\gamma$ and for each number i such that $0 \leq i < |w|+1$ we introduce a function on $\mathcal{U}$:

$$f^i_{\beta abR\gamma}(d) = \begin{cases} (j,c) & \text{if } d=(j,c) \text{ and } i \neq j,\\ (i,b) & \text{if } d=(i,a),\\ \gamma & \text{if } d=\beta\in\mathcal{S},\\ i+1 & \text{if } d=i,\\ \bot & \text{else.}\end{cases}$$

Similarly for each instruction $\beta abL\gamma$ and a number i such that $0 < i \leq |w|+1$

$$f^i_{\beta abL\gamma}(d) = \begin{cases} (j,c) & \text{if } d=(j,c) \text{ and } i \neq j,\\ (i,b) & \text{if } d=(i,a),\\ \gamma & \text{if } d=\beta\in\mathcal{S},\\ i-1 & \text{if } d=i,\\ \bot & \text{else.}\end{cases}$$

The construction allows us to imitate the computation of the machine $\mathbf{T}$ using compositions of appropriate functions. The final part of the restricted unary composition problem is defined in the following way:

$$s(d) = \begin{cases} (0,\Diamond) & \text{if } d=(0,\Diamond),\\ (i,w(i-1)) & \text{if } d=(i,w(i-1)) \text{ for } i \text{ between 1 and } |w|\\ (|w|+1,\blacklozenge) & \text{if } d=(|w|+1,\blacklozenge),\\ 0 & \text{if } d=0,\\ \omega & \text{if } d=\alpha\in\mathcal{S}.\end{cases}$$

Let's denote the domain of s by $\mathcal{V}$. If the composition of the functions from the set above is equal to s (on $\mathcal{V}$) then it doesn't evaluate to $\bot$ on any of the elements of $\mathcal{V}$. Since $\bot$ is the absorbing element then no subterm of a term defining the function can evaluate to $\bot$ on $\mathcal{V}$. Since α is in $\mathcal{V}$ the term has to "agree" on the states of functions adjacent in it. The same argument applies to the element of the set $\{0,\ldots,|w|+1\}$ implying that the computation agrees with respect to the placement of the head of the machine. Finally the images of the pairs force the composition to describe a computation of the machine $\mathbf{T}$ on a bounded tape equal to $\Diamond w\blacklozenge$. Thus a function s can be obtained as a composition if and only if $\mathbf{T}$ accepts w. The whole construction can be carried in a logarithmic space, and the reduction is proved. □

Thus, using Proposition 3.2 together with Lemma 4.1, we proved the theorem of Kozen.

Theorem 4.2 (Kozen)**.** *The composition of unary functions is a PSPACE-complete problem.*

5. Composition of functions of arbitrary arity is EXPTIME-complete

In this section we reprove the result of H. Friedman. The modification of the construction from Section 4 allows us to model the computations of an alternating Turing machine to obtain problems of higher complexity.

We follow [BS00] and [BJS99] in presenting an algorithm, working in an exponential time, and solving the problem of a composition of functions of arbitrary arity. The algorithm constructs all the k-ary members of $\mathrm{Clo}^{\mathcal{U}}(f_0,\ldots,f_n)$, where

k is the arity of the function s, and its computations follow Definition 2.2. For the list of functions $f_0, \dots, f_n$ given on input

(1) let `functions` be a an empty list of k-ary functions and `current` be an index in `functions`,
(2) add all the projections to `functions`
(3) set `current` to be the first element of `functions`
(4) while `current` is well defined
    (a) for each $i \leq n$ and for each choice of k_i-many functions enlisted in `functions` before `current` (the choice including at least one copy of `current`) apply f_i pointwise to generate a new k-ary function and add it to `functions` if it was not present there
    (b) move `current` to the next element of `functions`.

Let m denotes the maximal arity of the operation in the set $\{f_0, \dots, f_n\}$. Since there is at most $|\mathcal{U}|^{|\mathcal{U}|^k}$ functions from $\mathcal{U}^k \to \mathcal{U}$, the amount of the operations required to generate all the k-ary members of $\mathrm{Clo}^{\mathcal{U}}(f_0, \dots, f_n)$ is at most

$$C \cdot \left(|\mathcal{U}|^{|\mathcal{U}|^k}\right)^m \cdot |\mathcal{U}|^k,$$

for some constant C. This function is at most exponential with respect to the size of the input which is bounded from below by $\max\{|\mathcal{U}|^m, |\mathcal{U}|^k\}$.

It remains to prove the result of H. Friedman i.e that the problem of composition of functions of arbitrary arity is EXPTIME-hard. An alternative proof of this fact can be found in [BJS99]. Our proof introduces a mechanism that is going to be used extensively in the proof of the main result in Section 6.

We fix an alternating (comp. [CKS81]) Turing machine $\mathbf{T}$ such that:

(1) the language accepted by $\mathbf{T}$ is EXPTIME-complete,
(2) for a given word w the computations of the machine $\mathbf{T}$ never leave $\Diamond w \blacklozenge$,
(3) no instruction of $\mathbf{T}$ can be executed in the state ω,
(4) for each universal state of the machine $\mathbf{T}$ there are at most two instructions that can be executed (for a head reading any given symbol on a tape).

We prove an analogue of Lemma 4.1.

Lemma 5.1. *The problem of acceptance of a word w by the machine $\mathbf{T}$ reduces to the restricted composition problem.*

Proof. The construction of the set and functions is very similar to the construction already introduced in a proof of Lemma 4.1. For a given word w we define the set $\mathcal{U}$ to be the union of five sets

$$\{0, \dots, |w|+1\} \times \mathcal{A} \cup \mathcal{S} \cup \mathcal{S}' \cup \{0, \dots, |w|+1\} \cup \{\bot\},$$

where the four sets are precisely as in the above mentioned proof, and the set $\mathcal{S}'$ consists of copies of elements from $\mathcal{S}$ with the elements ω and ω' identified. Note that there is a natural bijection $\beta \mapsto \beta'$ between $\mathcal{S}$ and $\mathcal{S}'$.

The definition of the "computational" part of the problem follows exactly the previous construction. For each instruction $\beta a b R \gamma$ and for each number i such that $0 \leq i < |w|+1$ we introduce a function on $\mathcal{U}$:

$$f^i_{\beta a b R \gamma}(d) = \begin{cases} (j,c) & \text{if } d = (j,c) \text{ and } i \neq j, \\ (i,b) & \text{if } d = (i,a), \\ \gamma & \text{if } d = \beta \in \mathcal{S}, \\ i+1 & \text{if } d = i, \\ \bot & \text{else.} \end{cases}$$

Similarly for each instruction $\beta abL\gamma$ and a number i such that $0 < i \leq |w| + 1$

$$f^i_{\beta abL\gamma}(d) = \begin{cases} (j,c) & \text{if } d = (j,c) \text{ and } i \neq j, \\ (i,b) & \text{if } d = (i,a), \\ \gamma & \text{if } d = \beta \in \mathcal{S}, \\ i-1 & \text{if } d = i, \\ \bot & \text{else.} \end{cases}$$

These functions reproduce, exactly as in the case of unary composition, the configurations of a computation of a Turing machine.

The following functions allow us to check whether a computation of an alternating Turing machine was successful by backtracking it using additional states from the set $\mathcal{S}'$. Thus, for every pair of the instructions of $\mathbf{T}$ of the form $\beta abD\gamma$, $\beta acE\delta$ such that:

- if β is a universal state then $\beta abD\gamma$ and $\beta acE\delta$ are the only two instructions that can be executed in a state β with the head reading a;
- if β is an existential state then $\beta abD\gamma = \beta acE\delta$;

we define a function

$$g^i_{\beta abD\gamma,\beta acE\delta}(d,e) = \begin{cases} o & \text{if } d,e \notin \mathcal{S} \text{ and } f^i_{\beta abD\gamma}(o) = d \neq \bot \neq e = f^i_{\beta acE\delta}(o) \\ \beta' & \text{if } d = \gamma' \in \mathcal{S}' \text{ and } e = \delta' \in \mathcal{S}' \\ \bot & \text{else.} \end{cases}$$

It remains to define the function which is to be found as a composition of a functions defined above if and only if the machine $\mathbf{T}$ accepts w:

$$s(d) = \begin{cases} (0, \Diamond) & \text{if } d = (0, \Diamond), \\ (i, w(i-1)) & \text{if } d = (i, w(i-1)) \text{ for } i \text{ between 1 and } |w| \\ (|w|+1, \blacklozenge) & \text{if } d = (|w|+1, \blacklozenge), \\ 0 & \text{if } d = 0, \\ \alpha' & \text{if } d = \alpha \in \mathcal{S}. \end{cases}$$

The construction implies that, exactly like in the proof of Lemma 4.1, if a word w is accepted by the machine $\mathbf{T}$ then the restricted function s is in

$$\mathrm{Clo}^{\mathcal{U}}(f^i_{sabDl}, \dots, g^i_{sabDl}, \dots, g^i_{sabDl,sacEk}, \dots)$$

i.e. as the composition of the functions defined above. The other implication is a bit less obvious – the additional operations can generate configurations that are not a part of a computation starting at $\Diamond w \blacklozenge$, nevertheless these additional configurations cannot take part in the composition, since all of the operations follow the computation of the machine. This finishes the proof of a lemma. □

We proved that a language accepted by a machine $\mathbf{T}$ reduces to the problem of composition of functions which allows us to state the following theorem.

Theorem 5.2 (Friedman)**.** *The composition of functions of arbitrary arity is an EXPTIME-complete problem.*

Moreover the proof of Lemma 5.1 does not require functions of arity higher than two, which allows us to restate the theorem in a more restrictive way.

Theorem 5.3 (Friedman)**.** *The composition of binary functions is an EXPTIME-complete problem.*

6. A finite set of functions with an EXPTIME-complete composition problem

In this section we exhibit a finite family of functions, such that the membership in the set of functions generated by this family is EXPTIME-complete. We fix an alternating (comp. ([CKS81]) Turing machine $\mathbf{T}$ satisfying the following conditions:

(1) the language accepted by $\mathbf{T}$ is EXPTIME-complete and each word in it is of length $2^n - 2$ for some n,
(2) for a given word w the computations of the machine $\mathbf{T}$ never leave $\Diamond w \blacklozenge$, and at least one instruction is executed on each position of such a tape for each computation branch,
(3) no instruction of $\mathbf{T}$ can be executed in the state ω,
(4) for each universal state of the machine $\mathbf{T}$ there are at most two instructions that can be executed (for a head reading some symbol on a tape).

We introduce the set $\mathcal{U}$ and the functions on this set gradually. The functions split into three disjoin sets $\mathcal{F} \cup \mathcal{G} \cup \mathcal{H}$. All the functions in sets $\mathcal{F}$ and $\mathcal{H}$ are binary, while the functions in $\mathcal{G}$ are unary.

6.1. **The basic properties of the set.** The set $\mathcal{U}$ contains an absorbing element $\perp$ i.e.

$$\perp \in \mathcal{U} \tag{1}$$

and for any $f \in \mathcal{F}$, $g \in \mathcal{G}$ and $h \in \mathcal{H}$

$$f(\perp, a) = f(a, \perp) = g(\perp) = h(\perp, a) = h(a, \perp) = \perp \text{ for all } a \in \mathcal{U}.$$

The set $\mathcal{U}$ contains also

$$\mathcal{M} = \{0, 1\} \subset \mathcal{U} \tag{2}$$

and, for any $f \in \mathcal{F}$, $g \in \mathcal{G}$ and $h \in \mathcal{H}$, the following implications are true

$$\begin{aligned} f(a, b) \neq \perp &\implies a \in \mathcal{M} \wedge b \notin \mathcal{M}, \\ a \in \mathcal{M} &\implies g(a) = h(a, b) = h(b, a) = \perp \text{ for all } b \in \mathcal{U}, \end{aligned}$$

and the elements of the set $\mathcal{M}$ are outside of the range of all the functions in $\mathcal{F} \cup \mathcal{G} \cup \mathcal{H}$. This definition implies the following corollary.

Corollary 6.1. *If a function $s : \mathcal{U}^n \to \mathcal{U}$ is not constant and $s \in \mathrm{Clo}^{\mathcal{U}}(\mathcal{F} \cup \mathcal{G} \cup \mathcal{H})$, then whenever $f \in \mathcal{F}$ appears in a term defining s then its appearance is of the form $f(x_i, t(\bar{x}))$ for some variable x_i and term $t(\bar{x})$.*

The corollary leads to the following definition.

Definition 6.2. *A term $f^{(k)}(x_{i_k}, \dots f^{(0)}(x_{i_0}, y))$ (for $f^{(j)} \in \mathcal{F}$ for all j) is an $\mathcal{F}$-factor of a term $t(\bar{x})$ if and only if $f^{(k)}(x_{i_k}, \dots f^{(0)}(x_{i_0}, t'(\bar{x})))$ is a subterm of $t(\bar{x})$, for some term $t'(\bar{x})$, and the operation applied to it in $t(\bar{x})$ as well as the out-most operation of $t'(\bar{x})$ are not in $\mathcal{F}$.*

6.2. **The structure of some terms.** We introduce a second part of the set $\mathcal{U}$. The set

$$\mathcal{P} = \{A, B, C, D\} \subset \mathcal{U} \tag{3}$$

allows us to characterize the $\mathcal{F}$-factors of terms defining certain functions. For any

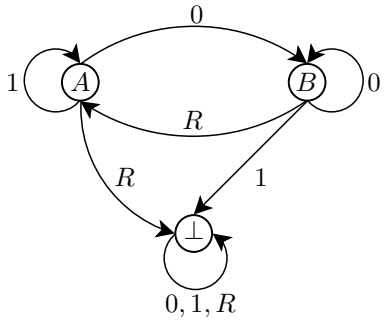

FIGURE 1. Forcing the order of the variables

$f \in \mathcal{F}$, $g \in \mathcal{G}$ and $h \in \mathcal{H}$ and any $a \in \mathcal{M}, b, d \in \mathcal{P}$ we set:

$$f(a,b) = d \text{ iff } (b \xrightarrow{a} d \text{ in Figure 1 or 2})$$

$$g(b) = d \text{ iff } (b \xrightarrow{R} d \text{ in Figure 1 or 2})$$

$$h(b,c) = \begin{cases} d & \text{if } b = c \wedge (b \xrightarrow{R} d \text{ in Figure 1 or 2}) \\ \bot & \text{if } b \neq c. \end{cases}$$

Moreover the above definitions are the only possibilities to obtain an element of $\mathcal{P}$

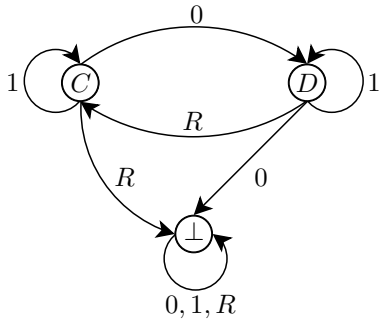

FIGURE 2. Forcing the arity of the variables

as a result of an application of a function from $\mathcal{F} \cup \mathcal{G} \cup \mathcal{H}$. Note that the conditions on the functions in $\mathcal{F} \cup \mathcal{G} \cup \mathcal{H}$ fully define them on $\mathcal{M} \cup \mathcal{P} \cup \{\bot\}$.

We are ready to characterize terms defining certain functions on $\mathcal{U}$. The function $s : \mathcal{U}^{n+1} \to \mathcal{U}$, for which we need to decide whether it can be obtained as a composition of the functions from $\mathcal{F} \cup \mathcal{G} \cup \mathcal{H}$, will satisfy these conditions.

[CI]. *For any $a \in 1^*0^+ \subset \{0,1\}^n$ we have $s(a, B) = A$.*

We present the first corollary.

Corollary 6.3. *If $t(\bar{x}, y)$ is a term defining a function satisfying [CI], then each $\mathcal{F}$-factor of $t(\bar{x}, y)$ – $f^{(k)}(x_{i_k}, \dots f^{(0)}(x_{i_0}, y))$ satisfies $i_0 \leq \dots \leq i_k$.*

Proof. Let us fix $f^{(k)}(x_{i_k}, \dots f^{(0)}(x_{i_0}, t'(\bar{x})))$ as in the Definition 6.2 for some $t(\bar{x}, y)$ defining a function satisfying [CI]. Since the outmost relation of $t'(\bar{x}, y)$ is not in $\mathcal{F}$ and $t'(a, B) \neq \bot$ we get $t'(a, B) = A$. Suppose, for a contradiction, that $i_l < i_j$ for some $j < l$ then for $a = 1^{i_j - 1}0^{n - i_j + 1}$ we have $f^{(j)}(0, \dots, f^{(0)}(a(i_0), A)) \in \{B, \bot\}$. Thus $f^{(l)}(1, \dots, f^{(0)}(a(i_0), A)) = \bot$ which is a contradiction with [CI]. □

We introduce the second condition.

[CII]. *For any $a \in 1^*01^* \subset \{0,1\}^n$ we have $s(a, D) = C$.*

The following corollary is proved in exactly the same way as Corollary 6.3 and we omit the proof of it.

Corollary 6.4. *If $t(\bar{x}, y)$ is a term defining a function satisfying [CI] and [CII], then all $\mathcal{F}$-factors of $t(\bar{x}, y)$ are of the form $f^{(n-1)}(x_{n-1}, \dots, f^{(0)}(x_0, y))$.*

A straightforward analysis of domains and ranges of the operations implies the following corollary.

Corollary 6.5. *If $t(\bar{x}, y)$ is a term defining a function satisfying [CI] and [CII], then the operations of $\mathcal{G} \cup \mathcal{H}$ are never applied directly after each other in $t(\bar{x}, y)$. Moreover the outmost operation and all the innermost operations of $t(\bar{x}, y)$ belong to $\mathcal{G} \cup \mathcal{H}$.*

6.3. **Modeling the tape of the machine.** We move to define a next part of the set $\mathcal{U}$. This part will allow us to control the tape of the Turing machine. We put

$$\mathcal{T} = \{T, F\}^2 \times \{L, H, R\} \times \mathcal{A} \subset \mathcal{U}. \tag{4}$$

The final coordinate of an element from $\mathcal{T}$ is a letter of the tape alphabet, the third coordinate indicates if the position on a tape is to the left, right or exactly where the head of the machine is. The two first coordinates are used to identify the future position of the head. The set $\mathcal{F}$ consists of four functions and we introduce them first. For any $b \in \mathcal{M}$ and any $(I, J, D, a) \in \mathcal{T}$ we define the function $f_0^0(x, y)$ to be

I	J	D	a	b	$f_0^0(b, (I, J, D, a))$
F	F	any	any	any	(F, F, D, a)
T	T	any	any	0	(T, T, D, a)
T	T	any	any	1	(F, F, D, a)
else	else	else	else	else	$\perp$

dually we define $f_1^1(x, y)$

I	J	D	a	b	$f_1^1(b, (I, J, D, a))$
F	F	any	any	any	(F, F, D, a)
T	T	any	any	0	(F, F, D, a)
T	T	any	any	1	(T, T, D, a)
else	else	else	else	else	$\perp$

and two more complicated functions $f_1^0(x, y)$

I	J	D	a	b	$f_1^0(b, (I, J, D, a))$
F	F	any	any	any	(F, F, D, a)
T	T	H or L	any	0	(T, F, D, a)
T	T	R	any	1	(F, T, D, a)
F	T	L	any	0	(F, F, D, a)
F	T	L	any	1	(F, T, D, a)
T	F	R or H	any	0	(T, F, D, a)
T	F	R or H	any	1	(F, F, D, a)
else	else	else	else	else	$\perp$

and $f_0^1(x,y)$

I	J	D	a	b	$f_0^1(b,(I,J,D,a))$
F	F	any	any	any	(F,F,D,a)
T	T	L	any	0	(F,T,D,a)
T	T	R or H	any	1	(T,F,D,a)
F	T	R	any	0	(F,T,D,a)
F	T	R	any	1	(F,F,D,a)
T	F	L or H	any	0	(F,F,D,a)
T	F	L or H	any	1	(T,F,D,a)
else	else	else	else	else	$\perp$

These functions allow us to identify the future position of the head. The following definition allows for an easier analysis of the $\mathcal{F}$-factors of some terms.

Definition 6.6. *For any words u,v of equal length over the alphabet $\{0,1\}$ we define the following terms:*

$$f_\varepsilon^\varepsilon(\bar{x},y)=y,$$

where ε denotes the empty word, and

$$f_{vb}^{ua}(\bar{x},y)=f_b^a(x_{|ua|-1},f_v^u(\bar{x},y))$$

recursively, where $a,b\in\{0,1\}$ and $f_b^a(x,y)$ is one of the four functions in $\mathcal{F}$.

Note that, according to Corollaries 6.3 and 6.4, each $\mathcal{F}$-factor of the term defining function satisfying conditions [CI] and [CII] is of the form $f_v^u(\bar{x},y)$ for some words u,v.

We need one more auxiliary concept. Let $P_i\subset\{0,1\}^n\times\mathcal{T}$ be such that $(b,(I,J,D,a))\in P_i$ if and only if $I=J=T$ and

$$(b<i\wedge D=L)\vee(b=i\wedge D=H)\vee(b>i\wedge D=R).$$

The following proposition is very usefull in our construction.

Proposition 6.7. *For any u,v, words of length n over the alphabet $\{0,1\}$, the following conditions are equivalent:*

(1) *$f_v^u(b,(I,J,D,a))\neq\perp$ for all $(b,(I,J,D,a))\in P_i$,*
(2) *$u=v$ or $i=u=v-1$ or $i=u=v+1$.*

Proof. We begin with a proof of the implication from 2 to 1. The case when $u=v$ is a trivial analysis of the definitions of functions $f_0^0(x,y)$ and $f_1^1(x,y)$. Assume now, that $i=u=v+1$ and that w is the longest common prefix of u and v. Lets fix an arbitrary $(b,(T,T,D,a))\in P_i$. If w is not a prefix of b then $f_w^w(b,(T,T,D,a))=(F,F,D,a)$ and thus $f_v^u(b,(T,T,D,a))\neq\perp$ as required. Assume now that w is a prefix of b. The word $w0$ is a prefix of v and $w1$ is a prefix of u - basing on this fact we consider two cases:

- if $b<i=u$, then $w0$ is a prefix of b and by the definition of P_i we obtain $D=L$ and thus

$$f_{w0}^{w1}(b,(T,T,L,a))=f_0^1(0,f_w^w(b,(T,T,L,a)))=$$
$$=f_0^1(0,(T,T,L,a))=(F,T,L,a).$$

  Since the function $f_1^0(x,y)$ applied to any element of the form (F,T,L,a) produces either the same element, or (F,F,L,a) we imply that, as required, $f_v^u(b,(T,T,L,a))\neq\perp$.

- if $b \geq i = u$ then $w1$ is a prefix of b and by the definition of P_i we infer that $D \in \{R, H\}$. In such a case

$$f^{w1}_{w0}(b,(T,T,D,a)) = f^1_0(1, f^w_w(b,(T,T,D,a))) = \\ = f^1_0(1,(T,T,D,a)) = (T,F,D,a),$$

and exactly the same reasoning shows that $f^u_v(b,(T,T,L,a)) \neq \perp$ as required.

This finishes the case of $i = u = v + 1$ and the remaining case of the implication, $i = u = v - 1$, is an alphabetical variant of the same proof. The implication from 2 to 1 is proved.

To start with the reverse implication assume that $u \neq v$ and let w denotes the maximal common prefix of u and v. We assume that $u > v$, that is $w1$ is a prefix of u and $w0$ is a prefix of v. Thus, for each $(b,(T,T,D,a)) \in P_i$ such that w is a prefix of b we have

$$f^{w1}_{w0}(b,(T,T,D,a)) = f^1_0(b_{|w1|-1}, f^w_w(b,(T,T,D,a))) = f^1_0(b_{|w1|-1},(T,T,D,a)) \neq \perp,$$

which, by the definition of $f^1_0(x,y)$, implies that $D = L$ whenever $b_{|w1|-1} = 0$ and $D \in \{R, H\}$ otherwise. The construction of P_i immediately implies that $i = w10^{n-|w1|}$. Let us fix an element of P_i of the form $(w01^{n-|w1|},(T,T,L,a))$, then

$$f^{w1}_{w0}(w01^{n-|w1|},(T,T,L,a)) = f^1_0(0,(T,T,L,a)) = (F,T,L,a),$$

and, since $f^0_1(x,y)$ is the only operation that does not produce $\perp$ on (F,T,L,a) we imply that $u = w10^{n-|w1|}$ and $v = w01^{n-|w1|}$ as required. The case of $u < v$ is an alphabetical variant of the same proof, and the proposition is proved. □

The following corollary is an easy consequence of the proposition.

Corollary 6.8. *Let u, v be members of $\{0,1\}^n$ such that $f^u_v(b,(T,T,D,a)) \neq \perp$ for all $(b,(T,T,D,a)) \in P_i$. Then, for any element $(b,(T,T,D,a)) \in P_i$,*

$$\big[f^u_v(b,(T,T,D,a)) = (T,J,D,a) \text{ for some } J \in \{T,F\}\big] \text{ if and only if } b = u;$$

and

$$\big[f^u_v(b,(T,T,D,a)) = (I,T,D,a) \text{ for some } I \in \{T,F\}\big] \text{ if and only if } b = v.$$

That is u is the only choice for b with $(b,(T,T,D,a)) \in P_i$ if we require the first projection of $f^u_v(b,(T,T,D,a))$ to be equal to T. Similarly v is the only choice for b if we require the second projection of $f^u_v(b,(T,T,D,a))$ equal to T. The proof of the corollary is a straightforward consequence of the fact that if $f^i_j(k,(I,J,D,a)) \neq \perp$, then $f^i_j(k,(I,J,D,a)) = (T,J,D,a)$ if and only if $I = T$ and $i = k$, for any choice of $i, j, k \in \{0,1\}$ (and similarly for the second coordinate).

The operations of $\mathcal{F}$ are defined on the set $\mathcal{T}$ in such a way that the last coordinate of the element of $\mathcal{T}$ has "no influence" on other coordinates (or result being equal to $\perp$) thus the following corollary holds.

Corollary 6.9. *For any $u, v \in \{0,1\}^n$ and any $P \subset P_i$ such that for any $b \in \{0,1\}^n$ there exists $(T,T,D,a) \in \mathcal{T}$ such that $(b,(T,T,D,a)) \in P$, the following conditions are equivalent:*

- *for any $(b,(T,T,D,a)) \in P$ we have $f^u_v(b,(T,T,D,a)) \neq \perp$;*
- *for any $(b,(T,T,D,a)) \in P_i$ we have $f^u_v(b,(T,T,D,a)) \neq \perp$.*

We move on to define the operations of the sets $\mathcal{G}$ and $\mathcal{H}$ on $\mathcal{T}$. We set $\bar{R} = L$ and $\bar{L} = R$ which allows us to present the definitions in a more compact way. We define the set $\mathcal{G}$ – the instructions from this set generate the configurations of $\mathbf{T}$

following the computations of the machine. For each instruction of $\mathbf{T}$, denoted by $\beta abD\gamma$, we put $g^{\beta abD\gamma}$ to be, for any $c \in \mathcal{T}$,

$$g^{\beta abD\gamma}(c) = \begin{cases} (T,T,\bar{D},b) & \text{if } c = (T,F,H,a) \\ (T,T,H,d) & \text{if } c = (F,T,D,d) \\ (T,T,G,d) & \text{if } c = (F,F,G,d) \text{ for some } G \neq H \\ \bot & \text{else.} \end{cases}$$

We move on to define the instructions of the set $\mathcal{H}$ on $\mathcal{T}$. This instructions allow us to check whether an alternating Turing machine accepts given configuration. Each of the instructions will provide an "accepted configuration" backtracking the computations of $\mathbf{T}$. Thus, for any pair of the instructions of $\mathbf{T}$ of the form $\beta abD\gamma$, $\beta acE\delta$ such that:

- if β is a universal state then $\beta abD\gamma$ and $\beta acE\delta$ are the only two instructions that can be executed in a state β with the head reading a;
- if β is an existential state then $\beta abD\gamma = \beta acE\delta$;

we define $h^{\beta abD\gamma}_{\beta acE\delta}$ on $d, e \in \mathcal{T}$ to be

$$h^{\beta abD\gamma}_{\beta acE\delta}(d,e) = \begin{cases} (T,T,H,a) & \text{if } d = (F,T,\bar{D},b) \text{ and } c = (F,T,\bar{E},c) \\ (T,T,D,o) & \text{if } D = E \text{ and } d = (T,F,H,o), e = (T,F,H,o) \\ (T,T,D,o) & \text{if } d = (T,F,H,o) \text{ and } e = (F,F,D,o) \\ (T,T,E,o) & \text{if } d = (F,F,E,o) \text{ and } e = (T,F,H,o) \\ (T,T,G,o) & \text{if } d = e = (F,F,G,o) \text{ for some } G \neq H \\ \bot & \text{else.} \end{cases}$$

Note that the following corollary is a simple consequence of the definitions:

Corollary 6.10. *For each $h^{\beta abD\gamma}_{\beta acE\delta} \in \mathcal{H}$ and for any $(I,J,G,d), (I',J',G'',d') \in \mathcal{T}$ and any $D'' \in \{L,H,R\}, d'' \in \mathcal{A}$ the following conditions are equivalent*

(1) $h^{\beta abD\gamma}_{\beta acE\delta}((I,J,D,d),(I',J',D',d')) = (T,T,D'',d'')$,
(2) $g^{\beta abD\gamma}((J,I,D'',d'')) = (T,T,D,d)$, *similarly* $g^{\beta acE\delta}((J',I',D'',d'')) = (T,T,D',d')$.

6.4. **The states of the machine.** We introduce the final part of the set $\mathcal{U}$. We put

$$\mathcal{R} = \mathcal{S} \cup \mathcal{S}' \subset \mathcal{U} \tag{5}$$

where the set $\mathcal{S}'$ consists of copies of elements from $\mathcal{S}$ with the elements ω and ω' identified. There is a natural bijection from $\mathcal{S}$ onto $\mathcal{S}'$ mapping $\beta \mapsto \beta'$ for each state β. For any $f(x,y) \in \mathcal{F}$ and any $a \in \mathcal{M}$ and $\beta \in \mathcal{R}$ we put

$$f(a,\beta) = \beta.$$

For any $g^{\beta abD\gamma}(x) \in \mathcal{G}$ and any $\delta \in \mathcal{R}$ we put

$$g^{\beta abD\gamma}(\delta) = \begin{cases} \gamma & \text{if } \beta = \delta \\ \bot & \text{else.} \end{cases}$$

For any $h^{\beta abD\gamma}_{\beta acE\delta} \in \mathcal{H}$ and any $\pi, \tau \in \mathcal{R}$ we put

$$h^{\beta abD\gamma}_{\beta acE\delta}(\pi,\tau) = \begin{cases} \beta' & \text{if } \pi = \gamma' \text{ and } \tau = \delta' \\ \bot & \text{else.} \end{cases}$$

All the applications of the operations which were not defined explicitly are equal to $\bot$.

6.5. **Computation and checking functions.** For any configuration of the machine $\mathbf{T}$, denoted by $\langle \mathbf{w}, i, \beta\rangle$, where $\mathbf{w} \in \mathcal{A}^{2^n}$, $i < 2^n$ and $\beta \in \mathcal{S}$, we define the set $P_{\langle \mathbf{w},i,\beta\rangle}$. The first part of it is a subset of P_i and is defined to be

$$(b, (I, J, D, a)) \in P_{\langle \mathbf{w},i,\beta\rangle} \cap P_i \text{ if and only if } \mathbf{w}(b) = a.$$

The second part of $P_{\langle \mathbf{w},i,\beta\rangle}$, which is outside of P_i, is equal to $\{0,1\}^n \times \{\beta\}$. Such a set describes, in a natural way, a configuration of the machine. The next definition allows us to follow the computations of the machine using these sets.

For any configuration $\langle \mathbf{w}, i, \beta\rangle$ and an operation of the Turing machine of the form $\gamma abD\delta$ we put

$$P_{\langle \mathbf{w},i,\beta\rangle:\gamma abD\delta} = \{(c, d) \mid (c, e) \in P_{\langle \mathbf{w},i,\beta\rangle} \text{ and } g^{\gamma abD\delta}(f_v^i(c, e)) = d\}$$

where $v = i + 1$ if $D = R$ and $v = i - 1$ if $D = L$. The following facts follow from the definitions.

Corollary 6.11. *For any configuration $\langle \mathbf{w}, i, \beta\rangle$, any instruction of the machine $\mathbf{T}$ $\gamma abD\delta$ and any pair of words $u, v \in \{0,1\}^n$ the following conditions are equivalent.*

(1) $\perp \notin g^{\gamma abD\delta}(f_v^u(P_{\langle \mathbf{w},i,\beta\rangle}))$,
(2) *the set*
$$P_{\langle \mathbf{w},i,\beta\rangle:\gamma abD\delta} = P_{\langle \mathbf{w}',j,\pi\rangle}$$
for some configuration $\langle \mathbf{w}', j, \pi\rangle$, and $u = i$ and $v = i + 1$ if $D = R$ and $v = i - 1$ if $D = L$,
(3) *the instruction $\gamma abD\delta$ can be executed in a configuration $\langle \mathbf{w}, i, \beta\rangle$ and $u = i$ and $v = i + 1$ if $D = R$ and $v = i - 1$ if $D = L$.*

Moreover, if these conditions are satisfied, the configuration obtained from $\langle \mathbf{w}, i, \beta\rangle$ after executing $\gamma abD\delta$ is equal to $\langle \mathbf{w}', j, \pi\rangle$.

Proof. The implication from 2 to 1 is trivial. For the implication 1 to 3 we use Proposition 6.7 and Corollary 6.9 to obtain $i = u$ and $|v - u| \leq 1$. Moreover Corollary 6.8 together with the definition of $g^{\gamma abD\delta}(x)$ implies that $u \neq v$ (since $g^{\gamma abD\delta}((T, T, E, c)) = \perp$ for any choice of E and c). Similarly, since $g^{\gamma abD\delta}((F, T, E, c)) \neq \perp$ implies $E = D$, we obtain $v = i+1$ if $D = R$ and $v = i-1$ if $D = L$. Finally, since $g^{\gamma abD\delta}((T, F, H, c)) \neq \perp$ implies $c = a$ we infer that $w(i) = a$ and similarly since $g^{\gamma abD\delta}(f_v^u(0^n, \beta)) \neq \perp$ we imply that $\beta = \gamma$ and 3 is proved. It remains to show the implication from 3 to 2. The arguments above applied to $g^{\gamma abD\delta}(f_{i+1}^i(\bar{x}, y))$ if $D = R$ and to $g^{\gamma abD\delta}(f_{i-1}^i(\bar{x}, y))$ if $D = L$ together with Proposition 6.7, Corollary 6.8 and the definion of functions in $\mathcal{G}$ proves the implication. A proof of this fact provides also an argument that the configuration obtained from $\langle \mathbf{w}, i, \beta\rangle$ after execution of $\gamma abD\delta$ is equal to $\langle \mathbf{w}', j, \pi\rangle$. □

By an inductive argument, using the previous corollary together with Corollaries 6.3 and 6.4 we obtain the following result.

Corollary 6.12. *Let $t(\bar{x}, y)$ be a term using functional symbols from $\mathcal{F} \cup \mathcal{G}$. If $t(\bar{x}, y)$ defines a function satisfying [CI] and [CII] and such that $\perp \notin t(P_{\langle \mathbf{w},i,\beta\rangle})$ for some configuration $\langle \mathbf{w}, i, \beta\rangle$, then the set*

$$\{(c, d) \mid (c, e) \in P \text{ and } t(c, e) = d\} = P_{\langle \mathbf{w}',j,\gamma\rangle}$$

for some configuration $\langle \mathbf{w}', j, \gamma\rangle$ which can be obtained by a Turing machine $\mathbf{T}$ starting its computations on $\langle \mathbf{w}, i, \beta\rangle$. Moreover for each such a configuration $\langle \mathbf{w}', j, \gamma\rangle$ there exists a term producing an appropriate set.

To exhibit an analogue of Corollary 6.11 for the functions from the set $\mathcal{H}$ we introduce, for each configuration, a new set $P'_{\langle \mathbf{w},i,\beta\rangle}$. The P_i part of the set is the same as of $P'_{\langle \mathbf{w},i,\beta\rangle}$ i.e.

$$(b,(I,J,D,a)) \in P'_{\langle \mathbf{w},i,\beta\rangle} \cap P_i \text{ if and only if } \mathbf{w}(b) = a$$

and the second part of $P_{\langle \mathbf{w},i,\beta\rangle}$, from outside of P_i, is equal to $\{0,1\}^n \times \{\beta'\}$. These sets allow us to backtrack the computations of the machine in the same way the previous sets were used to follow them. Note that $P'_{\langle \mathbf{w},i,\beta\rangle} \cap P_i = P_{\langle \mathbf{w},i,\beta\rangle} \cap P_i$ and $P'_{\langle \mathbf{w},i,\omega\rangle} = P_{\langle \mathbf{w},i,\omega\rangle}$ for any $\mathbf{w}, i$ and β.

We define $P^{\delta abD\pi:\langle \mathbf{w},i,\beta\rangle}_{\delta acE\tau:\langle \mathbf{w}',i',\gamma\rangle}$ to consist of the pairs (c,d) such that

$$\exists (c,e) \in P'_{\langle \mathbf{w},i,\beta\rangle}, \exists (c,e') \in P'_{\langle \mathbf{w}',i',\gamma\rangle} \text{ such that } h^{\delta abD\pi}_{\delta acE\tau}(f^i_v(c,e), f^{i'}_{v'}(c,e')) = d$$

where $v = i-1$ if $D = R$ and $v = i+1$ if $D = L$ and $v' = i'-1$ if $E = R$ and $v' = i'+1$ if $E = L$. The analogue of Corollary 6.11 states

Corollary 6.13. *For any configurations $\langle \mathbf{w},i,\beta\rangle$, $\langle \mathbf{w}',i',\gamma\rangle$ and any instructions of the machine $\mathbf{T}$: $\delta abD\pi$, $\delta acE\tau$ and any four words $u,v,u',v' \in \{0,1\}^n$ the following conditions are equivalent.*

1. *$h^{\delta abD\pi}_{\delta acE\tau}(f^u_v(c,e), f^{u'}_{v'}(c,e')) \neq \perp$, for any $(c,e) \in P'_{\langle \mathbf{w},i,\beta\rangle}$ and any $(c,e') \in P'_{\langle \mathbf{w}',i',\gamma\rangle}$,*
2. *the set*
$$P^{\delta abD\pi:\langle \mathbf{w},i,\beta\rangle}_{\delta acE\tau:\langle \mathbf{w}',i',\gamma\rangle} = P'_{\langle \mathbf{w}'',i'',\sigma\rangle}$$
for some configuration $\langle \mathbf{w}'',i'',\sigma\rangle$, and $u = i, u' = i'$ and $v = v'$ and $v = u-1$ if $D = R$ and $v = u+1$ if $D = L$ and $v' = u'-1$ if $E = R$ and $v' = u'+1$ if $E = L$
3. *there exists a configuration $\langle \mathbf{w}'',i'',\sigma\rangle$ such that the instructions $\delta abD\pi$ and $\delta acE\tau$ can be executed in it producing $\langle \mathbf{w},i,\beta\rangle$ and $\langle \mathbf{w}',i',\gamma\rangle$ respectively. Moreover if δ is a universal state then $\delta abD\pi$ and $\delta acE\tau$ are the only two instructions that can be executed in a state δ with the head reading a and if δ is an existential state then $\delta abD\pi = \delta acE\tau$. Finally $u = i, u' = i'$ and $v = v'$ and $v = u-1$ if $D = R$ and $v = u+1$ if $D = L$ and $v' = u'-1$ if $E = R$ and $v' = u'+1$ if $E = L$*

and the configurations denoted by $\langle \mathbf{w}'',i'',\sigma\rangle$ in 2 and 3 are equal.

The proof of the corollary is a carbon copy of the proof of Corollary 6.11 using Corollary 6.10. Finally, using inductive argument, we can show

Corollary 6.14. *Let $t(\bar{x},y)$ be a term such that the outmost functional symbol is from $\mathcal{H}$. If $t(\bar{x},y)$ defines a function satisfying [CI] and [CII] and such that $\perp \notin t(P_{\langle \mathbf{w},i,\beta\rangle})$ for some configuration $\langle \mathbf{w},i,\beta\rangle$, then the set*

$$\{(c,d) \mid (c,e) \in P \text{ and } t(c,e) = d\} = P'_{\langle \mathbf{w}',i',\gamma\rangle}$$

for some configuration $\langle \mathbf{w}',i',\gamma\rangle$ which is an accepting configuration of a Turing machine $\mathbf{T}$. Moreover for each accepting configuration which can be obtained working backwords from the accepting configurations obtained from $\langle \mathbf{w},i,\beta\rangle$ there exists a term producing the appropriate set.

6.6. **The searched function.** It remains to define a function $s : \mathcal{U}^n \times \mathcal{V} \to \mathcal{U}$ for each given input word w of length $2^n - 2$. We put $\langle \Diamond w \blacklozenge, 0^n, \alpha\rangle$ to be the starting

configuration of the machine $\mathbf{T}$ and define s to be

$$s(b,a) = \begin{cases} (b,a) & \text{if } (b,a) \in P_{\langle \Diamond w \blacklozenge, 0^n, \alpha \rangle} \cap P_{0^n} \\ \alpha' & \text{if } a = \alpha \\ A & \text{if } b \in 1^*0^+ \text{ and } a = B \\ C & \text{if } b \in 1^*01^* \text{ and } a = D \\ \bot & \text{else,} \end{cases}$$

for $\mathcal{V} = \{(I,J,D,a) \in \mathcal{T} \mid I = J = T\} \cup \{\alpha\} \cup \{B, D\}$. The Corollary 6.14 immediately implies that if the function s can be found as a restriction of the element of $\mathrm{Clo}^{\mathcal{U}}(\mathcal{F} \cup \mathcal{G} \cup \mathcal{H})$ then the word w is accepted by the machine $\mathbf{T}$. If, on the other hand, the word is accepted by $\mathbf{T}$ the same corollary provides us with an existence of a function, say s', satisfying [CI], [CII] and coinciding with s on $P_{\langle \Diamond w \blacklozenge, 0^n, \alpha \rangle}$. It remains to show that

- $s'(b, B) = \bot$ whenever $b \notin 1^*0^+$,
- $s'(b, D) = \bot$ whenever $b \notin 1^*01^*$,
- $s'(b, (T,T,D,a)) = \bot$ whenever $(b, (T,T,D,a)) \notin P_{\langle \Diamond w \blacklozenge, 0^n, \alpha \rangle}$.

The first two points are obvious consequences of the definition of the functions on the set $\mathcal{P}$. For the last one we remark that for each $(b, (T,T,D,a)) \notin P_{\langle \Diamond w \blacklozenge, 0^n, \alpha \rangle}$ there exists $(b, (T,T,D',a')) \in P_{\langle \Diamond w \blacklozenge, 0^n, \alpha \rangle}$, and since the computation of $\mathbf{T}$ visits each square of the tape and the term does not evaluate to $\bot$ on $(b, (T,T,D',a'))$ it has to on $(b, (T,T,D,a))$. Thus the main theorem of this section is proved.

Theorem 6.15. *The set* $\mathrm{Clo}^{\mathcal{U}}(\mathcal{F} \cup \mathcal{G} \cup \mathcal{H})$ *has an EXPTIME-complete membership problem.*

It is a trivial observation that all of the unary operations in our construction can be substituted by binary (putting $\bot$ as a result of an application to non-identical arguments). This allows us to restate the theorem in a more uniform way.

Theorem 6.16. *There exists a finite set* $\mathcal{U}$ *and a family of functions* $f_0, \dots, f_n : \mathcal{U}^2 \to \mathcal{U}$ *such that the question whether a given function can be found as a composition of* $f_0, \dots, f_n$ *is EXPTIME-complete.*

7. Consequences

The authors in [BJS99] tackle the problem, denoted by TERM-EQ, which is defined in the following way

INPUT	a pair of finite algebras $(\mathbf{A}, \mathbf{B})$ over the same set
PROBLEM	decide if $\mathrm{Clo}(\mathbf{B}) = \mathrm{Clo}(\mathbf{A})$.

and prove that it is EXPTIME-complete. Our reasoning implies that one of the algebras $\mathbf{A}, \mathbf{B}$ can be fixed (to be the algebra constructed in Section 6) and the computational complexity of the problem will not decrease. This proves that there is a finite algebra $\mathbf{A}$ such that the problem

INPUT	a finite algebra $\mathbf{B}$
PROBLEM	decide if $\mathrm{Clo}(\mathbf{B}) = \mathrm{Clo}(\mathbf{A})$.

is EXPTIME-complete which strengthens the known result.

References

[Bal96] José L. Balcázar, *The complexity of searching implicit graphs*, Artificial Intelligence **86** (1996), no. 1, 171–188. MR **MR1410132 (98a:**68088)

[BJS99] Clifford Bergman, David Juedes, and Giora Slutzki, *Computational complexity of term-equivalence*, Internat. J. Algebra Comput. **9** (1999), no. 1, 113–128. MR **MR1695293 (2000b:**68088)

[BS81] Stanley Burris and H. P. Sankappanavar, *A course in universal algebra*, Graduate Texts in Mathematics, vol. 78, Springer-Verlag, New York, 1981. MR **MR648287 (83k:**08001)

[BS00] Clifford Bergman and Giora Slutzki, *Complexity of some problems concerning varieties and quasi-varieties of algebras*, SIAM Journal on Computing **30** (2000), no. 2, 359–382.

[BW94] Elwyn Berlekamp and David Wolfe, *Mathematical Go*, A K Peters Ltd., Wellesley, MA, 1994, Chilling gets the last point, With a foreword by James Davies. MR **MR1274921 (95i:**90131)

[Čer64] Ján Černý, *A remark on homogeneous experiments with finite automata*, Mat.-Fyz. Časopis Sloven. Akad. Vied **14** (1964), 208–216. MR MR0168429 (29 #5692)

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer, *Alternation*, J. Assoc. Comput. Mach. **28** (1981), no. 1, 114–133. MR **MR603186 (83g:**68059)

[FL81] Aviezri S. Fraenkel and David Lichtenstein, *Computing a perfect strategy for $n\times n$ chess requires time exponential in n*, J. Combin. Theory Ser. A **31** (1981), no. 2, 199–214. MR **MR629595 (83b:**68044)

[GJ85] M. R. Garey and D. S. Johnson, *Composing functions to minimize image size*, SIAM J. Comput. **14** (1985), no. 2, 500–503. MR **MR784752 (86d:**68031)

[JM06] Marcel Jackson and Ralph McKenzie, *Interpreting graph colorability in finite semigroups*, Internat. J. Algebra Comput. **16** (2006), no. 1, 119–140. MR **MR2217645 (2006m:**20081)

[KK] Marcin Kozik and Gábor Kun, *The subdirectly irreducible algebras in the variety generated by graph algebras*, Accepted.

[Koz77] Dexter Kozen, *Lower bounds for natural proof systems*, 18th Annual Symposium on Foundations of Computer Science (Providence, R.I., 1977), IEEE Comput. Sci., Long Beach, Calif., 1977, pp. 254–266. MR MR0495200 (58 #13931)

[LS80] David Lichtenstein and Michael Sipser, *GO is polynomial-space hard*, J. Assoc. Comput. Mach. **27** (1980), no. 2, 393–401. MR **MR567056 (81b:**68052)

[MMT87] Ralph N. McKenzie, George F. McNulty, and Walter F. Taylor, *Algebras, lattices, varieties. Vol. I*, The Wadsworth & Brooks/Cole Mathematics Series, Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1987. MR **MR883644 (88e:**08001)

[Rob84] J. M. Robson, *N by N checkers is Exptime complete*, SIAM J. Comput. **13** (1984), no. 2, 252–267. MR **MR739988 (86f:**90168)

[Sav70] Walter J. Savitch, *Relationships between nondeterministic and deterministic tape complexities*, Journal of Computer and System Sciences **4** (1970), 177–192.

[Szé02] Zoltán Székely, *Computational complexity of the finite algebra membership problem for varieties*, Internat. J. Algebra Comput. **12** (2002), no. 6, 811–823. MR **MR1949698 (2003k:**08009)

Algorithmics Research Group, Jagiellonian University, Golebia 24, 31-007 Krakow, Poland

Eduard Čech Center, Charles University, Sokolovska 83, 186 75 Praha 8, Czech Republic

E-mail address: Marcin.Kozik@tcs.uj.edu.pl