



Networks of Watson-Crick DOL systems with communication by substrings[☆]

Erzsébet Csuhaaj-Varjú^{a,*}, György Vaszil^b

^a Faculty of Informatics, ELTE Eötvös Loránd University, Pázmány Péter sétány 1/c, Budapest, 1117, Hungary

^b Faculty of Informatics, University of Debrecen, Kassai út 26, Debrecen, 4028, Hungary

ARTICLE INFO

Keywords:

DNA computing
Watson-Crick DOL system
Networks of Watson-Crick DOL systems
Recursively enumerable languages

ABSTRACT

Watson-Crick DOL systems (WDOL systems) are augmented variants of DOL systems defined over a DNA-like alphabet, where each letter has a complementary letter and this relation is symmetric. WDOL systems operate under a control that is inspired by the well-known phenomenon of Watson-Crick complementarity of the double helix of DNA. Depending on a trigger, the standard DOL rewriting step is applied either to the string or to its complementary string. In this paper, we examine extended networks of standard Watson-Crick DOL systems (ENSWDOL systems) with a variant of incomplete communication. An NWDOL system is a finite set of WDOL systems defined over a common DNA-like alphabet and operating in a synchronized manner. After rewriting their own strings in the WDOL manner, they communicate copies of certain generated strings (the so-called good strings) to the other nodes. In some previous papers, it was shown that ENSWDOL systems are computationally complete, and their computational power does not change if the communicated string is a non-empty prefix (non-empty suffix) of the generated string. We strengthen the previous results, namely we show that ENSWDOL systems are computationally complete even if the communicated string is an arbitrary substring of the generated string.

1. Introduction

Watson-Crick complementarity is a fundamental concept in DNA computing. The notion is motivated by the well-known phenomenon of the double helix of DNA, namely, that two DNA strands form a double strand if they are complement of each other. A paradigm where complementarity is considered in the operational sense, was introduced in [1,2]. The seminal model is called a Watson-Crick DOL system (a WDOL system for short).

A WDOL system is a DOL system given over a DNA-like alphabet and a mapping called the trigger for complementarity transition. In a DNA-like alphabet each letter has a complementary letter and this relation is symmetric. The letters of a DNA-like alphabet are called purines and pyrimidines, by the analogy of the nucleotides which are the sequence elements forming DNA strands. The complement of each purine is a pyrimidine and the complement of each pyrimidine is a purine. The trigger is a logical-valued mapping over the set of strings over the DNA-like alphabet with the following property: the value of the trigger of the axiom is 0, and whenever the value of the trigger of a string is 1, then the trigger value of its complementary string is 0. (The complementary string of a string is obtained by replacing each letter with its complementary letter.)

[☆] This article belongs to Section C: Theory of natural computing, Edited by Lila Kari.

* Corresponding author.

E-mail addresses: csuhaj@inf.elte.hu (E. Csuhaaj-Varjú), vaszil.gyorgy@inf.unideb.hu (G. Vaszil).

The derivation in the Watson-Crick DOL system is defined as follows: the string is rewritten in the standard DOL manner. Then the new string is checked according to the trigger. If the value of the trigger is 0 (the string is a so-called good string), then the derivation continues with the string. If the value of the trigger is 1 (the string is a so-called bad string), then the derivation continues with the complementary string.

One of the most important and popular variants of Watson-Crick DOL systems are the standard Watson-Crick DOL systems (SWDOL systems for short). In this case, the trigger is defined as follows: after applying the DOL rewriting step to the string, the number of occurrences of purines and the number of occurrences of pyrimidines in the new string are checked. If there are more occurrences of pyrimidines than that of purines, then the derivation continues with the complementary string of the new string, otherwise the derivation continues in the usual manner.

Watson-Crick DOL systems have been studied from several points of view. The interested reader can find further information on the computational power and different properties of these systems in [3–13]. A recent survey on Watson-Crick DOL systems and their variants is presented in [14].

WDOL systems were put in a distributed and communicating framework in [15] where networks of Watson-Crick DOL systems (NWDOL systems for short) were introduced. A network of Watson-Crick DOL systems (an NWDOL system) is a finite set of WDOL systems that are defined over a common DNA-like alphabet and operate in a synchronized manner. If no confusion arises, the WDOL systems of the network can also be called the nodes of the network. The WDOL systems perform derivation steps on their own strings in the WDOL manner and after each derivation step communicate copies of some of the new strings to the other nodes. The condition for communication is given by the trigger for turning to the complementary string, i.e. depending on whether or not a turn to the complement took place, the string or its complementary string is communicated.

For NSWDOL systems, two communication protocols were introduced in [15]: in the first case (protocol (a)), after rewriting the string, the WDOL system in the node keeps the good new strings and the complementary strings of the bad new strings and sends a copy of each good new string to every other node. In the second case (protocol (b)), the WDOL system keeps the good new strings and the complementary strings of the bad new strings and sends a copy of the complementary string of each bad new string to every other nodes.

Several important results were obtained on NSWDOL systems and their variants. In [16], two well-known NP-complete problems, namely, the Hamiltonian Path Problem and the Satisfiability Problem were shown to be solvable in linear time with NSWDOL systems. In [17] it was proved that any recursively enumerable language can be obtained as the language of an extended NSWDOL system using protocol (a). The language of an extended NSWDOL system is the set of strings which are given over a sub-alphabet of the system (the terminal alphabet) and appear at a dedicated node at some derivation step during the operation of the system.

In this paper, we deal with networks of standard WDOL systems with a certain type of incomplete information communication. The question was whether or not the computation power of NSWDOL systems changes if not the whole string but only one of its substrings is communicated to the other node.

In a predecessor paper, [18], we studied the computational power of NSWDOL systems with communication protocol (*pref, a*). In this case, the node sends to every other node a good non-empty prefix of every good non-empty string obtained by the rewriting step (this can be the whole good string itself) and keeps the new good strings and the complementary strings of the obtained bad strings.

An SWDOL system in the network is allowed to send different prefixes of the same string to different nodes, but it is allowed to send only one prefix of the string to a certain node. We proved that even with this type of incomplete information communication, networks of extended SWDOL systems (ENSWDOL systems) are computationally complete, i.e. generate any recursively enumerable language. An analogous statement was also formulated for the case when not a prefix but a suffix of the string is communicated (communication protocol (*suff, a*)).

In this paper, we examine the case when arbitrary non-empty substring of the good strings or complementary strings of the bad strings are communicated (communication protocol (*sub, a*)). In this case, after the rewriting step, the SWDOL system sends an arbitrary substring of its good strings and an arbitrary substring of the complementary strings of the obtained bad strings to every other node. We show that these variants of ENSWDOL systems are also computationally complete.

In the proof of the result on communication of prefixes in [18], marker symbols were added to the beginning and to the end of the string, and their presence was checked at the arrival to the target node, based on conditions being good or bad strings of SWDOL systems i.e. the relation between the number of purines and the number of pyrimidines. If the markers were present in the communicated string, then the whole string arrived at its target node. The proof of the main result, namely, that substring communication is sufficient to generate any recursively enumerable language also implements the communication of the whole string. But the technique we use in the proof is very much different from the one that is used in [18]. In this case, the string w in the recursively enumerable language is present in the nodes in the form of $\beta'\alpha\beta''\beta'''$, where α is a sentential form (according to a derivation in a DOL manner) and β' , β'' , β''' are the primed, double-primed, and triple-primed versions of β , where β is equal either to α or to $\bar{\alpha}$, i.e. the complementary string of α . The rules of the system are given in such way that if not the whole string $\beta'\alpha\beta''\beta'''$ is communicated, then the system is not able to produce a terminal string from this string any more. The primed copies of β together with α guarantee the necessary number of purines and pyrimidines in the string.

The paper is organized as follows. The Introduction is followed by the section of Preliminaries, the notions and notations which are necessary to read the paper. In Section 3, we show that any EDTOL language can be generated by an ENSWDOL system using communication protocol (*sub, a*). The proof of the statement provides some kind of characterization of the strings that can be obtained in the master node of the ENSWDOL system and introduces the technique of enforcing the communication of the complete string, as we described above. The construction in this proof is used to prove the main result, presented in the next section. In Section 4, as the main result of the paper, we show that ENSWDOL systems using protocol (*sub, a*) are computationally complete. The proof of

this statement uses a simulation of the generation of a recursively enumerable language based on the Extended Post Correspondence and the construction in the proof of the result on generating EDTOL languages by ENSWDOL systems, given in Section 3. Finally, in Section 5 some conclusions are derived.

2. Preliminaries

2.1. Basic notions

We assume that the reader is familiar with formal language theory and the basics of DNA computing. For detailed information consult [19] and [20].

The set of strings over an alphabet V is denoted by V^* ; if the empty string, λ , is excluded, then the notation V^+ is used. A set of strings $L \subseteq V^*$ is called a language over alphabet V .

The number of occurrences of elements of U in w , where $U \subseteq V$ and $w \in V^*$ is denoted by $|w|_U$. If U is a singleton, say $\{a\}$, then the notation $|w|_a$ can also be used.

A string u is a substring of $w \in V^*$ if $w = xuy$ ($x, y \in V^*$). If $xy \neq \lambda$, then u is a proper substring of w . If $x = \lambda$, then u is a prefix of w and if $y \neq \lambda$, then we speak of a proper prefix. Analogously, if $y = \lambda$, then u is a suffix of w and if $x \neq \lambda$, then it is a proper suffix. Throughout the paper, $sub(w)$, $pref(w)$, and $suff(w)$ denote the set of substrings, the set of prefixes, and the set of suffixes of a string w .

A triple $H = (V, P, w)$ is a DOL system, where V is an alphabet, P is a finite non-empty set of rules of the form $a \rightarrow u$ with $a \in V$, $u \in V^*$, and $w \in V^+$. Furthermore, P is complete and deterministic, i.e. for every $a \in V$ there exists a rule $a \rightarrow u$ in P and for each $a \in V$ there is only one rule in P .

A direct derivation step in H is defined as follows: for two strings $x, y \in V^*$, x directly derives y in H , denoted by $x \Rightarrow_H y$, if $x = x_1 \dots x_k$, $y = y_1 \dots y_k$, and $x_i \rightarrow y_i \in P$, $1 \leq i \leq k$. If no confusion arises, index H can be omitted from $\Rightarrow_H$. The transitive (and reflexive) closure of $\Rightarrow$ is denoted by $\Rightarrow^+$ ($\Rightarrow^*$).

The language $L(H)$ of $H = (V, P, w)$ is defined by $L(H) = \{u \mid w \Rightarrow^* u\}$.

A fourtuple $G = (V, T, P, w)$ is an extended DOL system (an EDOL system for short), if (V, P, w) is a DOL system and $T \subset V$. T is called the set of terminal letters of G . The language $L(G)$ of G is defined by $L(G) = \{u \in T^* \mid w \Rightarrow^* u\}$.

A triple $\Gamma = (V, \mathcal{M}, w)$ is called a tabled DOL system (a DTOL system for short), where $\mathcal{M} = \{M_1, \dots, M_n\}$, $n \geq 1$, and for each i , $1 \leq i \leq n$, (V, M_i, w) is a DOL system. (We use notation M_i instead of P_i .)

A direct derivation step in Γ is defined as follows: for two strings $x, y \in \Sigma^*$, x directly derives y in Γ , denoted by $x \Rightarrow_{M_j} y$, if $x = x_1 \dots x_k$, $y = y_1 \dots y_k$, and there exists j , $1 \leq j \leq n$ such that $x_i \rightarrow y_i \in M_j$, $1 \leq i \leq k$ holds.

The language $L(\Gamma)$ of $\Gamma = (V, \mathcal{M}, w)$ is defined by $L(\Gamma) = \{u \mid w \Rightarrow_{M_{i_1}} u_1 \dots \Rightarrow_{M_{i_s}} u, 1 \leq i_j \leq n, 1 \leq j \leq s\}$.

A fourtuple $\Delta = (V, T, \mathcal{M}, w)$ is called an extended tabled DOL system (an EDTOL system for short), if $\mathcal{M} = \{M_1, \dots, M_n\}$, $n \geq 1$, and for each i , $1 \leq i \leq n$, (V, T, M_i, w) is an EDOL system.

The direct derivation step in Δ is defined in the same way as the direct derivation step in DTOL systems.

The language $L(\Delta)$ of Δ is $L(\Delta) = \{u \in T^* \mid w \Rightarrow_{M_{i_1}} u_1 \dots \Rightarrow_{M_{i_s}} u, 1 \leq i_j \leq n, 1 \leq j \leq s\}$.

2.2. Networks of Watson-Crick DOL systems

In the following, we recall some basic notions concerning Watson-Crick DOL systems and their variants. Standard Watson-Crick DOL systems were introduced in [1].

A DNA-like alphabet is an alphabet $\Sigma = \{a_1, \dots, a_n, \bar{a}_1, \dots, \bar{a}_n\}$, $n \geq 1$. Letters a_i and $\bar{a}_i$, $1 \leq i \leq n$, are said to be complementary letters. $\Sigma_1 = \{a_1, \dots, a_n\}$ is called the subalphabet of purines in Σ , and $\Sigma_2 = \{\bar{a}_1, \dots, \bar{a}_n\}$ is called the subalphabet of pyrimidines in Σ .

A string $w \in \Sigma^*$ is said to be a good string if $|w|_{\Sigma_1} \geq |w|_{\Sigma_2}$, otherwise it is called a bad string. The empty string is a good string.

The letter to letter endomorphism of a DNA-like alphabet Σ which maps each letter to its complementary letter is denoted by h_w . It is called the Watson-Crick morphism.

A standard Watson-Crick DOL system (an SWDOL system for short) is a triple $H = (\Sigma, P, w_0)$ where Σ , P , and w_0 are defined as for DOL systems, Σ is a DNA-like alphabet, and w_0 is a non-empty good string over Σ . The direct derivation step in H is defined in the following manner: for two strings $x, y \in \Sigma^*$, x directly derives y in H , denoted by $x \Rightarrow_H y$, if the following hold: $x = x_1 \dots x_m$, $z = z_1 \dots z_k$, where $x_i \rightarrow z_i \in P$, $1 \leq i \leq k$. Then $y = z$ if z is a good string and $y = h_w(z)$ if z is a bad string.

That is, if after applying a parallel rewriting to the letters of the string x , the obtained new string has less occurrences of purines than pyrimidines, then the new string turns to its complementary string and the derivation continues from this complementary string. If this is not the case, then the derivation continues in the usual manner.

An extended standard WDOL system (an ESWDOL system for short) is a fourtuple $G = (\Sigma, T, P, w_0)$, where (Σ, P, w_0) is an SWDOL system and $T \subset \Sigma$. T is called the set of terminal letters.

Networks of standard Watson-Crick DOL systems were introduced in [15,17]. Here, we present a slightly modified version of that seminal definition, namely, we indicate a master node.

A network of standard Watson-Crick DOL systems (an NSWOL system for short) is an $(n+2)$ -tuple

$$\Gamma = (\Sigma, (P_1, w_1), \dots, (P_n, w_n), e),$$

where

- Σ is a DNA-like alphabet (the alphabet of the system),
- (Σ, P_i, w_i) , $1 \leq i \leq n$ is an SWDOL system, called the i th component of Γ ,
- (Σ, P_e, w_e) is the master node, $1 \leq e \leq n$.

If no confusion arises, then P_i can also be called the i th component of Γ .

NSWDOL systems operate by changing their states. A direct change of a state consists of rewriting the strings in the WDOL manner and communicating them according to a given protocol.

A state of an NSWDOL system $\Gamma = (\Sigma, (P_1, w_1), \dots, (P_n, w_n), e)$, $n \geq 1$, is an n -tuple $(L_1, \dots, L_n)$, where L_i is a set of good strings over Σ , $1 \leq i \leq n$. The initial state of the system is $(\{w_1\}, \dots, \{w_n\})$, where w_i is the axiom of the i th component, $1 \leq i \leq n$.

In [18] modified communication protocols were introduced and studied. The original protocols are given in [15] as follows: after the rewriting step, the node sends a copy of every obtained good non-empty string to each other node and keeps the obtained good strings and the complementary strings of the generated bad strings (protocol (a)). In the case of protocol (b), it sends a copy of the complementary string of every generated bad string to each other node and keeps the good strings and the complementary strings of the bad strings.

In [18], communication protocols (x, a) , where $x \in \{pref, suff\}$ were introduced. In this case, after the parallel rewriting step, the node sends a good non-empty prefix (a good non-empty suffix) of every obtained good non-empty string to each other node and keeps the obtained good strings and the complementary strings of the generated bad strings. It was shown that any recursively enumerable language can be generated by an NSWDOL system using communication protocol $(pref, a)$. It was also stated, without a proof, that the statement holds for communication protocol $(suff, a)$ as well.

In this paper, we deal with the case when arbitrary non-empty substring is communicated, i.e. we examine the case of communication protocol (sub, a) . This protocol is defined as follows.

Let $\Gamma = (\Sigma, (P_1, w_1), \dots, (P_n, w_n), e)$, $n \geq 1$, be an NSWDOL system and let $s_1 = (L_1, \dots, L_n)$ and $s_2 = (L'_1, \dots, L'_n)$ be two states of Γ .

We say that s_1 directly derives s_2 according to protocol (sub, a) , denoted by $s_1 \xrightarrow{(sub, a)}_{\Gamma} s_2$, if the following conditions are met: for every i , $1 \leq i \leq n$, let

$$L'_i = A'_i \cup B'_i \bigcup_{j=1, j \neq i}^n C'_j,$$

where

$$A'_i = \{z \mid z = h_w(y), x \Rightarrow_{P_i} y, x \in L_i, y \text{ is a bad string}\},$$

$$B'_i = \{y \mid x \Rightarrow_{P_i} y, x \in L_i, y \text{ is a good string}\},$$

and C_j is a set of elements obtained from the elements of

$$B'_j = \{v_{j_1}, \dots, v_{j_{r_j}}\} = \{y \mid x \Rightarrow_{P_j} y, x \in L_j \text{ is a good string}\}$$

as follows:

$$C'_j = \{v'_{j_1}\} \cup \dots \cup \{v'_{j_{r_j}}\},$$

where $v'_{j_k} \in sub(v_{j_k})$, v'_{j_k} is a non-empty good string and $v_{j_k} \in B'_j$, $1 \leq k \leq r_j$. If B'_j is the empty set, then C'_j is also the empty set.

The transitive and reflexive closure of $\xrightarrow{(sub, a)}_{\Gamma}$ is denoted by $\xRightarrow{(sub, a)}_{\Gamma}$.

The language of an NSWDOL system Γ using protocol (sub, a) is

$$L_{(sub, a)}(\Gamma) = \{u_1 \in L_e \mid (\{w_1\}, \dots, \{w_n\}) \xRightarrow{(sub, a)}_{\Gamma} (L_1, \dots, L_n)\}.$$

That is, the language of Γ is the set of strings which are present at the master node at some derivation step of the operation of the system. The axiom is also an element of the language.

By an extended NSWDOL system (an ENSWDOL system for short) we mean an $(n+3)$ -tuple $\Gamma = (\Sigma, T, (P_1, w_1), \dots, (P_n, w_n), e)$, $n \geq 1$, where $T \subseteq \Sigma$ and all other components of Γ are defined in the same way as in the case of NSWDOL systems.

The language of an extended NSWDOL system Γ using protocol (sub, a) is defined by

$$L_{(sub, a)}(\Gamma) = \{u_1 \in (T^* \cap L_e) \mid (\{w_1\}, \dots, \{w_n\}) \xRightarrow{(sub, a)}_{\Gamma} (L_1, \dots, L_n)\}.$$

3. EDTOL systems versus ENSWDOL systems with communication protocol (sub, a)

In this section, we show that every EDTOL language can be obtained as the language of an ENSWDOL system with communication protocol (sub, a) . Since EDTOL systems resemble ENSWDOL systems using protocol (a), this statement is an interesting result on the relation of these two models. In the proof of the statement, a novel technique is provided for checking whether in a communication step of the ENSWDOL system a substring or the whole string itself is communicated. Furthermore, the construction gives some kind

of characterization of the strings that can be obtained at the master node of the ENSWDOL system. The construction of the proof is used in the proof of the main result of the paper, Theorem 2, in Section 4, which states the computational completeness of ENSWDOL systems working with communication protocol (sub, a) .

Theorem 1.

$$\mathcal{L}(EDTOL) \subseteq \mathcal{L}_{(sub,a)}(ENSWDOL)$$

Proof. Let $G = (V, T, \mathcal{M}, w)$ with $\mathcal{M} = (M_1, \dots, M_n)$ be an EDTOL system. We construct an ENSWDOL system Γ , such that $L(\Gamma) = L(G)$. Let

$$\Gamma = (\Sigma, T, (P_e, S), (P_1, \tilde{S}), \dots, (P_n, \tilde{S}), e)$$

with $\Sigma = \Sigma_1 \cup \tilde{\Sigma}_1$ with

$$\begin{aligned} \Sigma_1 = \{ & a, a_0, a'_0, a''_0, a'''_0 \mid a \in V \} \cup \{ a_{i,j}, a_{e,j}, a'_{i,j}, a'_{e,j}, a''_{i,j}, a''_{e,j}, \\ & a'''_{i,j}, a'''_{e,j} \mid a \in V, 1 \leq i \leq n, 1 \leq j \leq 3 \} \cup \{ F, S, \tilde{S} \}. \end{aligned}$$

Note that we use the notation $a^\beta_\alpha \in \Sigma$ for some superscript α and subscript β to denote the indexed and/or primed variant of the corresponding symbol $a \in V$. Similarly, $u^\beta_\alpha \in \Sigma^*$ will denote the strings corresponding to some $u \in V^*$ by being composed of the indexed or the indexed and primed versions of the symbols of u . Moreover, we write u^β_α and $\bar{u}^\beta_\alpha$ if the string is composed of non-barred or barred letters only.

The components of Γ are defined as follows.

$$\begin{aligned} P_e = \{ & S \rightarrow w'_0 w_0 w''_0 w'''_0 \mid w \text{ is the axiom of } G \} \cup \{ a \rightarrow F \mid a \in V \} \cup \\ & \{ a'_0 \rightarrow a'_{e,1}, \bar{a}'_{e,1} \rightarrow F, a''_0 \rightarrow \bar{a}''_{e,1}, a'_{e,1} \rightarrow F, a'''_0 \rightarrow a'''_{e,1}, \\ & \bar{a}'''_{e,1} \rightarrow F, a_0 \rightarrow \bar{a}_{e,1}, a_{e,1} \rightarrow F \mid a \in V \} \cup \\ & \{ a'_{e,1} \rightarrow \bar{a}'_{e,2}, a'_{e,2} \rightarrow F, \bar{a}''_{e,1} \rightarrow a''_{e,2}, \bar{a}''_{e,2} \rightarrow F, a'''_{e,1} \rightarrow \bar{a}'''_{e,2}, \\ & a'''_{e,2} \rightarrow F, \bar{a}_{e,1} \rightarrow \bar{a}_{e,2}, a_{e,2} \rightarrow F \mid a \in V \} \cup \\ & \{ \bar{a}'_{e,2} \rightarrow a'_{e,3}, \bar{a}'_{e,3} \rightarrow F, a''_{e,2} \rightarrow \bar{a}''_{e,3}, a''_{e,3} \rightarrow F, \bar{a}'''_{e,2} \rightarrow \bar{a}'''_{e,3}, \\ & a'''_{e,3} \rightarrow F, \bar{a}_{e,2} \rightarrow a_{e,3}, \bar{a}_{e,3} \rightarrow F \mid a \in V \} \cup \\ & \{ a'_{e,3} \rightarrow \lambda, a_{e,3} \rightarrow a, \bar{a}''_{e,3} \rightarrow \lambda, \bar{a}'''_{e,3} \rightarrow \lambda \mid a \in V \} \cup \\ & \{ a'_{i,j} \rightarrow F, \bar{a}'_{i,j} \rightarrow F, a''_{i,j} \rightarrow F, \bar{a}''_{i,j} \rightarrow F, a'''_{i,j} \rightarrow F, \bar{a}'''_{i,j} \rightarrow F, \\ & F \rightarrow F, \bar{F} \rightarrow \bar{F} \mid a \in V, 1 \leq i \leq n, 1 \leq j \leq 3 \}. \end{aligned}$$

If $\lambda \in L(G)$, then we also add rule $S \rightarrow \lambda$ to P_e . This component starts and finishes the generation of the strings. The process starts when it introduces the string $w'_0 w_0 w''_0 w'''_0$ which is a concatenation of four indexed and primed copies of the axiom w of the EDTOL system G . This is a good string, so after it is introduced, substrings are sent to all other components.

The other components correspond to the tables $M_i, 1 \leq i \leq n$ of G , and they are defined as

$$\begin{aligned} P_i = \{ & a'_0 \rightarrow a'_{i,1}, \bar{a}'_{i,1} \rightarrow F, a''_0 \rightarrow \bar{a}''_{i,1}, a'_{i,1} \rightarrow F, a'''_0 \rightarrow a'''_{i,1}, \\ & \bar{a}'''_{i,1} \rightarrow F, a_0 \rightarrow \bar{a}_{i,1}, a_{i,1} \rightarrow F \mid a \in V \} \cup \\ & \{ a'_{i,1} \rightarrow \bar{a}'_{i,2}, a'_{i,2} \rightarrow F, \bar{a}''_{i,1} \rightarrow a''_{i,2}, \bar{a}''_{i,2} \rightarrow F, a'''_{i,1} \rightarrow \bar{a}'''_{i,2}, \\ & a'''_{i,2} \rightarrow F, \bar{a}_{i,1} \rightarrow \bar{a}_{i,2}, a_{i,2} \rightarrow F \mid a \in V \} \cup \\ & \{ \bar{a}'_{i,2} \rightarrow a'_{i,3}, \bar{a}'_{i,3} \rightarrow F, a''_{i,2} \rightarrow \bar{a}''_{i,3}, a''_{i,3} \rightarrow F, \bar{a}'''_{i,2} \rightarrow \bar{a}'''_{i,3}, \\ & a'''_{i,3} \rightarrow F, \bar{a}_{i,2} \rightarrow a_{i,3}, \bar{a}_{i,3} \rightarrow F \mid a \in V \} \cup \\ & \{ a'_{i,3} \rightarrow \alpha'_0, a_{i,3} \rightarrow \alpha_0, \bar{a}''_{i,3} \rightarrow \alpha''_0, \bar{a}'''_{i,3} \rightarrow \alpha'''_0 \mid a \rightarrow \alpha \in M_i \} \cup \\ & \{ a'_{j,k} \rightarrow F, \bar{a}'_{j,k} \rightarrow F, a''_{j,k} \rightarrow F, \bar{a}''_{j,k} \rightarrow F, a'''_{j,k} \rightarrow F, \bar{a}'''_{j,k} \rightarrow F, \\ & a'_{e,k} \rightarrow F, \bar{a}'_{e,k} \rightarrow F, a''_{e,k} \rightarrow F, \bar{a}''_{e,k} \rightarrow F, a'''_{e,k} \rightarrow F, \bar{a}'''_{e,k} \rightarrow F, \\ & F \rightarrow F, \bar{F} \rightarrow \bar{F}, a \rightarrow F \mid a \in V, j \neq i, 1 \leq j \leq n, 1 \leq k \leq 3 \} \cup \\ & \{ \tilde{S} \rightarrow F, \bar{\tilde{S}} \rightarrow F \}. \end{aligned}$$

These components start with $\tilde{S}$ as axiom that is immediately rewritten to the trap symbol. The components rewrite the strings they receive from the other components. When the i th component receives a string, it either simulates the application of the table M_i , or if something goes wrong, it turns the string into a string of trap symbols $F \dots F$ which can only be rewritten identically.

To avoid the introduction of the trap symbols, the component must receive a string of the form $u'_0 u_0 u''_0 u'''_0$ for some $u \in V^*$. In the first phase of its functioning, the components check if the lengths of the four differently primed parts of the string coincide, that is, if $|u'_0| = |u_0| = |u''_0| = |u'''_0|$ holds. This equality holds, if and only if, no part of the string was “lost” during the substring communication. We call such a string *complete*. The checking of the completeness is done as follows. Given a substring

$$x'_0 u_0 y''_0 z'''_0$$

of a complete string, the i th component changes it to

$$x'_{i,1} \bar{u}_{i,1} \bar{y}''_{i,1} \bar{z}'''_{i,1}.$$

If this is a bad string, then it turns to its complement, and a string of the form $F \dots F$ is obtained after the next rewriting step. If $x'_{i,1} \bar{u}_{i,1} \bar{y}''_{i,1} \bar{z}'''_{i,1}$ is a good string, then copies of it are communicated to the other components which turn it into strings of the form $F \dots F$ since they have no other rules for symbols indexed with i .

The i th component continues the processing of $x'_{i,1} \bar{u}_{i,1} \bar{y}''_{i,1} \bar{z}'''_{i,1}$ based on the following considerations. If $x'_{i,1} \bar{u}_{i,1} \bar{y}''_{i,1} \bar{z}'''_{i,1}$ is a good string, then one of the following cases must hold:

1. $|x'_{i,1}| = |\bar{u}_{i,1}| = |\bar{y}''_{i,1}| = |\bar{z}'''_{i,1}|$, which means that $x'_{i,1} \bar{u}_{i,1} \bar{y}''_{i,1} \bar{z}'''_{i,1}$ is complete, thus, the communicated substring $x'_0 u_0 y''_0 z'''_0$ coincides with the complete sentential form.
2. $\bar{y}''_{i,1} \bar{z}'''_{i,1} = \lambda$ and $|x'_{i,1}| \geq |\bar{u}_{i,1}|$, which means that the double and triple primed parts of the string were lost completely, together with possibly some of the primed and non-primed symbols.
3. $x'_{i,1} \bar{u}_{i,1} = \lambda$ and $|\bar{y}''_{i,1}| \leq |\bar{z}'''_{i,1}|$, which means that the primed and non-primed parts of the string were lost completely, together with possibly some of the double and triple primed symbols.

No matter which of the above cases holds, the rewriting process continues by changing the string to

$$\bar{x}'_{i,2} \bar{u}_{i,2} \bar{y}''_{i,2} \bar{z}'''_{i,2}.$$

In the first case, that is, when the communicated substring is the complete string, this is again a good string since the number of non-barred and barred symbol occurrences are equal. After the successful rewriting step, substrings are communicated to the other components where strings of the form $F \dots F$ are introduced, as we have seen above.

In the second case, this newly obtained string is a bad string, so it turns to its complement which can only be rewritten to $F \dots F$ in the next step, producing no terminal string in any possible derivation that might follow.

In the third case, the obtained string is also a good string, so the derivation continues and produces $\bar{y}''_{i,3} \bar{z}'''_{i,3}$, which is a bad string, so it turns to its complement and then it is changed to $F \dots F$ producing no terminal string afterwards.

If, on the other hand, we started with a completely communicated string (in the first case, according to the numbering above), we get the string

$$x'_{i,3} u_{i,3} y''_{i,3} z'''_{i,3},$$

which is a good string, so (after communicating with the other components where strings of the form $F \dots F$ are introduced), we get $v'_0 v_0 v''_0 v'''_0$ corresponding to $v \in V^*$, the result of the application to the i th table of the EDTOL system G to the sentential form $u \in V^*$.

As this is a good string, it is communicated to all the other components where the application of another table is simulated (after the completeness of the communicated string is checked), or in the case of the master component, the string $v \in V^*$, a possible sentential form of the EDTOL system G is produced. If this string is terminal, that is, if $v \in L(G)$, then v is also an element of $L_{(sub,a)}(\Gamma)$, the language generated by the ENSWDOL system Γ .

To see how the master component works, consider the rules of the production set P_e . Similarly to the other components, after receiving a substring of the sentential form $v'_0 v_0 v''_0 v'''_0$, it produces

$$v'_{e,3} v_{e,3} \bar{v}''_{e,3} \bar{v}'''_{e,3}$$

in three rewriting steps, if and only if, it received the complete string $v'_0 v_0 v''_0 v'''_0$ by the substring communication. Now, in the next rewriting step the primed symbols are erased, $v \in V^*$ is obtained, and then the trap symbols are introduced

$$v'_{e,3} v_{e,3} \bar{v}''_{e,3} \bar{v}'''_{e,3} \Rightarrow v \Rightarrow F \dots F$$

which are then rewritten identically for the rest of the derivation. These are all good strings, so their substrings are sent to the other components, but they can only be rewritten to strings of trap symbols by any of them, so these communications will not contribute to the results of the derivation. On the other hand, if $v \in T^*$, then it is considered to be part of the language generated by Γ .

We can see by the construction above that the strings produced by the components either turn into strings of the form $F \dots F$, or they correspond to a sentential form of the EDTOL system, in particular, the terminal strings appearing at the master component of

Γ can also be generated by G . We may also see that any series of applications of the tables of G can be reproduced in Γ by applying the corresponding components to the string $w'_0 w''_0 w'''_0$ (which is initially received by all components) in the same sequence as the tables are applied to the axiom w in G . $\square$

4. Computational completeness of ENSWDOL systems with communication protocol (sub, a)

In this section, we show that every recursively enumerable language can be generated by an ENSWDOL system working with the protocol (sub, a) . The proof of the statement is based on a representation of recursively enumerable languages by the Extended Post Correspondence [21] and the on the construction in the proof of Theorem 1.

Let $\Sigma = \{a_1, a_2, \dots, a_l\}$ be an alphabet, where $l \geq 1$. An Extended Post Correspondence (an EPC) is a pair

$$\Pi = (\{(u_1, v_1), \dots, (u_m, v_m)\}, (z_{a_1}, \dots, z_{a_l})),$$

where $m \geq 1$, $u_i, v_i, z_{a_j} \in \{0, 1\}^*$ for $1 \leq i \leq m$ and $1 \leq j \leq l$. The language represented by Π , written as $L(\Pi, \Sigma)$, is defined by

$$L(\Pi, \Sigma) = \{x_1 \dots x_n \in \Sigma^* \mid \text{there is } (s_1, s_2, \dots, s_r) \in \{1, 2, \dots, m\}^r, r \geq 1, \\ \text{such that } u_{s_1} u_{s_2} \dots u_{s_r} = v_{s_1} v_{s_2} \dots v_{s_r} z_{x_1} z_{x_2} \dots z_{x_n}\}.$$

It is shown in [21] that for each recursively enumerable language $L \subseteq \Sigma^*$, there exists an Extended Post Correspondence Π such that $L(\Pi, \Sigma) = L$.

We first note that the statement that every recursively enumerable language is the language of an instance of the EPC also holds if the strings u_i, v_i, z_{a_j} are defined over $\{1, 2\}$ instead of $\{0, 1\}$. Therefore, we may consider the strings $u_{s_1} u_{s_2} \dots u_{s_r}$, $v_{s_1} \dots v_{s_r}$, and $z_{x_1} \dots z_{x_n}$ as numbers in the base three notation and represent them with the values of these numbers.

For a string $u \in \{1, 2\}^*$, let $val(u) \in \mathbb{N}$ denote the number that is represented by u in the base three notation, and let $dig(u) \in \mathbb{N}$ denote the length of u (the number of digits in u). In the following, we will use the modified version of the EPC defined with strings over the binary alphabet $\{1, 2\}$, where a string $w = x_1 \dots x_n$, $x_i \in \Sigma$, $1 \leq i \leq n$, is in L if and only if there exist indices $s_1, \dots, s_r \in \{1, \dots, m\}$, such that the two strings $u_{s_1} \dots u_{s_r} \in \{1, 2\}^*$ and $v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n} \in \{1, 2\}^*$ have the same value as numbers in the base three notation, $val(u_{s_1} \dots u_{s_r}) = val(v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n})$.

Based on an EPC Π , we can generate the elements of $L = L(\Pi, \Sigma)$ as follows. We start the generation with a string of the form $u_{s_1} v_{s_1}$, $s_1 \in \{1, \dots, m\}$. Then, based on the pairs (u, v) from Π , we add u -s and v -s to the string in the correct manner to obtain a string of the form $\alpha\beta$ with $\alpha = u_{s_1} \dots u_{s_t}$ and $\beta = v_{s_1} \dots v_{s_t}$, for some $t \geq 1$. Then, in a second phase, we add x -s and z -s to the string to obtain $x_1 \dots x_m u_{s_1} \dots u_{s_t} v_{s_1} \dots v_{s_t} z_{x_1} \dots z_{x_m}$. Now, in the final phase, we check whether $\alpha = u_{s_1} \dots u_{s_t}$ and $\gamma = v_{s_1} \dots v_{s_t} z_{x_1} \dots z_{x_m}$ are equal or not. If the equality holds, then we delete both substrings from the string. (If the empty string is in L , we can generate it by continuing with the final generation phase directly after the first phase of the above procedure.)

The strings in L can also be obtained if in the previous procedure we represent α and β (or γ) with strings with exactly as many occurrences of certain letters, say A and B , as the values of α and β (or γ), respectively, according to the base three notation. Thus, we can simulate the appending of a pair (u_j, v_j) or (a_j, z_{a_j}) to the string in generation by modifying the number of occurrences of letters A and B in the string.

Based on the above discussion, we can establish a connection between EDTOL systems and the Extended Post Correspondence. First, we introduce some auxiliary notions.

Definition 1. Let $\Sigma = \{a_1, a_2, \dots, a_l\}$ and let $\Pi = (\{(u_1, v_1), \dots, (u_m, v_m)\}, (z_{a_1}, \dots, z_{a_l}))$ be an EPC with $m \geq 1$ and $u_i, v_i, z_{a_j} \in \{1, 2\}^*$ for $1 \leq i \leq m$, $1 \leq j \leq l$.

The string $x_{i_1} \dots x_{i_m} u_{s_1} u_{s_2} \dots u_{s_r} v_{s_1} v_{s_2} \dots v_{s_r} z_{x_1} z_{x_2} \dots z_{x_n}$ is called a *valid string* of Π in Σ , if $x_1 \dots x_n \in \Sigma^*$ and $(s_1, s_2, \dots, s_r) \in \{1, 2, \dots, m\}^r$, $r \geq 1$. The set of all valid strings of Π in Σ is denoted by $S(\Pi, \Sigma)$.

Lemma 1. Let $\Sigma = \{a_1, \dots, a_l\}$, $l \geq 1$, be an alphabet, and let

$$\Pi = (\{(u_1, v_1), \dots, (u_m, v_m)\}, (z_{a_1}, \dots, z_{a_l})),$$

be an Extended Post Correspondence with $1 \leq m$ and $u_i, v_i, z_{a_j} \in \{1, 2\}^*$ for $1 \leq i \leq m$, $1 \leq j \leq l$.

We can construct an EDTOL system $G = (V, T, \mathcal{M}, w_0)$ with $T = \Sigma \cup \{A, B\} \subset V$, such that the following hold. Let $h : V^* \rightarrow \Sigma^*$ be a projection defined as $h(a) = a$ for $a \in \Sigma$ and $h(b) = \lambda$ otherwise. Then

- for any $w \in L(G)$, there exists a valid string

$$y = x_{i_1} \dots x_{i_n} u_{s_1} \dots u_{s_r} v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n} \in S(\Pi, \Sigma),$$

such that $|w|_A = val(u_{s_1} \dots u_{s_r})$, $|w|_B = val(v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n})$, $h(y) = x_{i_1} \dots x_{i_n}$, and

- for any $y = x_{i_1} \dots x_{i_n} u_{s_1} \dots u_{s_r} v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n} \in S(\Pi, \Sigma)$, there exists a string

$$w \in L(\Gamma),$$

such that $|w|_A = \text{val}(u_{s_1} \dots u_{s_r})$, $|w|_B = \text{val}(v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n})$, and $h(w) = x_{i_1} \dots x_{i_n}$.

Proof. We construct an EDTOL system $G = (V, T, \mathcal{M}, w_0)$ with $V = \Sigma \cup \{A, A_1, B, B_1, S_1, S_2, F\}$, $T = \Sigma \cup \{A, B\}$, and $w_0 = S_1$. The set of tables $\mathcal{M}$ of Γ is given as follows.

For each pair (u_h, v_h) , $1 \leq h \leq m$, $\mathcal{M}$ has a table

$$M_{(u_h, v_h)} = \{S_1 \rightarrow S_1 A_1^{\text{val}(u_h)} B_1^{\text{val}(v_h)}, A_1 \rightarrow A_1^{\text{dig}(u_h)}, B_1 \rightarrow B_1^{\text{dig}(v_h)}, \\ S_2 \rightarrow F, F \rightarrow F\} \cup \{a \rightarrow F \mid a \in \Sigma\},$$

for each pair (a_i, z_{a_i}) , $1 \leq i \leq l$, $\mathcal{M}$ has a table

$$M_{(a_i, z_{a_i})} = \{S_2 \rightarrow S_2 a_i B_1^{\text{val}(z_{a_i})}, B_1 \rightarrow B_1^{\text{dig}(z_{a_i})}, A_1 \rightarrow A_1, F \rightarrow F\} \cup \\ \{a \rightarrow a \mid a \in \Sigma\}.$$

In addition, $\mathcal{M}$ also has a table to switch between the two phases of the derivation

$$M_{\text{switch}} = \{S_1 \rightarrow S_2, A_1 \rightarrow A_1, B_1 \rightarrow B_1, S_2 \rightarrow F, F \rightarrow F\} \cup \\ \{a \rightarrow a \mid a \in \Sigma\},$$

and a table to produce terminal strings

$$M_{\text{term}} = \{S_2 \rightarrow \lambda, A_1 \rightarrow A, B_1 \rightarrow B, S_1 \rightarrow F, F \rightarrow F\} \cup \\ \{a \rightarrow a \mid a \in \Sigma\}.$$

Every derivation in G starts with the axiom S_1 . Then, in the first phase of the derivation, tables of the form $M_{(u_h, v_h)}$, $1 \leq h \leq m$, are applied. This derivation phase results in a string u over $\{S_1, A_1, B_1\}$ such that it represents a string $\alpha = u_{s_1} \dots u_{s_r} v_{s_1} \dots v_{s_r}$, $1 \leq s_k \leq m$, $1 \leq k \leq r$, i.e. $|\alpha|_{A_1} = \text{val}(u_{s_1} \dots u_{s_r})$ and $|\alpha|_{B_1} = \text{val}(v_{s_1} \dots v_{s_r})$. The strings obtained in this derivation phase correspond to the first phase of the generation of a string in Π in Σ .

The derivation can be continued by the applications of M_{switch} and then the tables $M_{(a_i, z_{a_i})}$, $1 \leq i \leq l$. The resulting string, γ , corresponds to a valid string of Π in Σ . Then, after applying M_{term} a terminal version of the sentential form is produced which represents a string $x_{i_1} \dots x_{i_n} u_{s_1} \dots u_{s_r} v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n}$, where $|\gamma|_A = \text{val}(u_{s_1} \dots u_{s_r})$ and $|\gamma|_B = \text{val}(v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n})$. Furthermore, $h(\gamma) = x_{i_1} \dots x_{i_n}$.

Notice that if table of the form $M_{(u_h, v_h)}$ is applied after introducing S_2 in the string in generation, then no terminal string can be generated since nonterminal F is introduced in the string that does not disappear from the string in the following steps. Thus, a derivation of a terminal string in Γ corresponds to a generation of a valid string in Π .

By the rules of the tables of G , it can be seen that the application of a table corresponds to appending a pair (u_h, v_h) or (a_i, z_{a_i}) to the string in generation. Furthermore, the tables of G are defined in such way that to obtain a terminal string, tables of the form $M_{(u_h, v_h)}$ are applied first and after that the application of tables of the form $M_{(a_i, z_{a_i})}$ follows. No mixed application of the two types of tables is possible. By these observations we conclude that each derivation in Γ resulting in a terminal string corresponds to a generation of a valid string in Π . $\square$

Using Lemma 1 and the construction in the proof of Theorem 1 we can now prove the main statement of the paper.

Theorem 2. For every recursively enumerable language L , we can construct an ENSWDOL system Γ such that $L_{(\text{sub}, a)}(\Gamma) = L$.

Proof. To prove the statement, we modify the construction presented in the proof of Theorem 1.

Let $\Sigma = \{a_1, \dots, a_l\}$, let $L \subseteq \Sigma^*$ be a recursively enumerable language, and let $L = L(\Pi, \Sigma)$ for the EPC $\Pi = ((u_1, v_1), \dots, (u_m, v_m)), (z_{a_1}, \dots, z_{a_l})$, where $m \geq 1$, $u_i, v_i, z_{a_j} \in \{1, 2\}^*$ for $1 \leq i \leq m$, $1 \leq j \leq l$. Thus,

$$L(\Pi, \Sigma) = \{x_1 x_2 \dots x_n \in \Sigma^* \mid (s_1, s_2, \dots, s_r) \in \{1, 2, \dots, m\}^r, r \geq 1,$$

$$u_{s_1} u_{s_2} \dots u_{s_r} = v_{s_1} v_{s_2} \dots v_{s_r} z_{x_1} z_{x_2} \dots z_{x_n}\}.$$

By Lemma 1, we can construct an EDTOL system $G = (V, T, \mathcal{M}, w_0)$ for the EPC Π , such that $T = \Sigma \cup \{A, B\} \subset V$, and for any $w \in L(G)$ there exists a string

$$y = x_{i_1} \dots x_{i_n} u_{s_1} \dots u_{s_r} v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n} \in S(\Pi, \Sigma),$$

such that $|w|_A = \text{val}(u_{s_1} \dots u_{s_r})$, $|w|_B = \text{val}(v_{s_1} \dots v_{s_r} z_{x_1} \dots z_{x_n})$, and $h(y) = x_{i_1} \dots x_{i_n}$, where the projection $h : V^* \rightarrow \Sigma^*$ is defined by $h(a) = a$ for $a \in \Sigma$ and $h(a) = \lambda$ otherwise. (Note that $h(\lambda) = \lambda$.)

By Theorem 1, we can construct an ENSWDOL system Γ such that $L(\Gamma) = L(G)$. By the proof of Theorem 1, any string is a terminal string of G , if and only if it can be obtained at the master node of Γ . The terminal strings of G consist of letters from $\Sigma \cup \{A, B\}$. To

satisfy the conditions set in the Extended Post Correspondence, we need to select those strings where the number of letters A and B are equal.

To meet this requirement, we add further symbols to the alphabet of Γ , modify the rule set of the master P_e , and add further rules to the other components. The new symbols and rules will assist the checking of the equality of the number of occurrences of letters A and letters B in the terminal strings of G .

We add to the nonterminal alphabet of Γ the set of letters together with their complementary letters

$$V_2 = \{a_{e,i}, C_{e,i} \mid 4 \leq i \leq 6, a \in T\},$$

and add to the rule sets

$$\{a \rightarrow F \mid a \in V_2 \cup \bar{V}_2\}$$

to the components P_i corresponding to the i th tables of G .

We modify the rule set of P_e as follows. We replace the rules

$$\{a'_{e,3} \rightarrow \lambda, a_{e,3} \rightarrow a, \bar{a}''_{e,3} \rightarrow \lambda, \bar{a}'''_{e,3} \rightarrow \lambda \mid a \in V\}$$

with

$$\begin{aligned} &\{a'_{e,3} \rightarrow \lambda, a_{e,3} \rightarrow a_{e,4}\bar{C}_{e,4}, \bar{a}''_{e,3} \rightarrow \lambda, \bar{a}'''_{e,3} \rightarrow \lambda \mid a \in T\} \cup \\ &\{a'_{e,3} \rightarrow \bar{F}, a_{e,3} \rightarrow \bar{F}, \bar{a}''_{e,3} \rightarrow \bar{F}, \bar{a}'''_{e,3} \rightarrow \bar{F} \mid a \in V \setminus T\}. \end{aligned}$$

These rules, instead of finishing the derivation, select the terminal strings. If a string $w'_{e,3}w_{e,3}w''_{e,3}w'''_{e,3}$ corresponding to a sentential form w of the simulated EDTOL system is produced by the ENSWDOL system at P_e , then it is rewritten to $w_{e,4}$ (if $w \in T^*$) or it is rewritten to a bad string (if w contains at least one nonterminal). In the latter case, the string turns to the complement, and then is rewritten to $F \dots F$ by the rules

$$\{\bar{a}_{e,4} \rightarrow F, C_{e,4} \rightarrow F \mid a \in T\}$$

which we also add to P_e .

If $w_{e,4}$ is successfully generated, we can start to check the equality of the occurrences of symbols $A_{e,4}$ and $B_{e,4}$. First we check that $|w_{e,4}|_{A_{e,4}} \geq |w_{e,4}|_{B_{e,4}}$ by using the rules (also added to P_e)

$$\begin{aligned} &\{a_{e,4} \rightarrow a_{e,5}, \bar{a}_{e,5} \rightarrow F \mid a \in T\} \cup \\ &\{A_{e,4} \rightarrow A_{e,5}, \bar{A}_{e,5} \rightarrow F, B_{e,4} \rightarrow \bar{B}_{e,5}, B_{e,5} \rightarrow F, \bar{C}_{e,4} \rightarrow \bar{C}_{e,5}, C_{e,5} \rightarrow F\}. \end{aligned}$$

If the number of symbols $B_{e,4}$ was greater than the number of symbols $A_{e,4}$, then the rewritten string turns to its complement, and then to $F \dots F$. Next, in a similar way, we check that $|w_{e,4}|_{A_{e,4}} \leq |w_{e,4}|_{B_{e,4}}$ by using the rules

$$\begin{aligned} &\{a_{e,5} \rightarrow a_{e,6}, \bar{a}_{e,6} \rightarrow F \mid a \in T\} \cup \\ &\{A_{e,5} \rightarrow \bar{A}_{e,6}, A_{e,6} \rightarrow F, \bar{B}_{e,5} \rightarrow B_{e,6}, \bar{B}_{e,6} \rightarrow F, \bar{C}_{e,5} \rightarrow \bar{C}_{e,6}, C_{e,6} \rightarrow F\}. \end{aligned}$$

If the result of the last rewriting step is a good string, we have $w_{e,6}$, such that $|w_{e,6}|_{\bar{A}_{e,6}} = |w_{e,6}|_{B_{e,6}}$. If we now delete the unnecessary nonterminals, we get a string of the language represented by the EPC Π . For this purpose, we can use the rules (which we also add to P_e)

$$\{a_{e,6} \rightarrow a \mid a \in T\} \cup \{\bar{A}_{e,6} \rightarrow \lambda, B_{e,6} \rightarrow \lambda, \bar{C}_{e,6} \rightarrow \lambda\}.$$

As we have seen above, the modified construction is able to use the capabilities of standard Watson-Crick DOL systems to check the equality of the number of occurrences of different symbols in the valid strings of an Extended Post Correspondence, so it can be used to make sure that the ENSWDOL system only generates strings, if and only if, they are in the language of the EPC. $\square$

5. Conclusions

In this paper, we examined extended networks of standard Watson-Crick DOL systems (ENSWDOL systems) with a new variant of incomplete communication, namely, when the SWDOL systems in the nodes communicate to the other nodes arbitrary substrings of the good strings they generated. It was shown in previous papers that ENSWDOL systems are computationally complete and their computational power does not change if the communicated string is a non-empty prefix (non-empty suffix) of the generated string. Here we strengthened these results, namely we proved that ENSWDOL systems are computationally complete even if the communicated string is an arbitrary substring of the generated string. To prove the result, we present a novel technique for simulating EDTOL systems with ENSWDOL systems where the derivation leads to a terminal string if and only if after each rewriting step the whole string and not any of its proper substrings is communicated.

CRedit authorship contribution statement

Erzsébet Csuhaj-Varjú: Writing – original draft, Investigation, Conceptualization. **György Vaszil:** Writing – original draft, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Open access is provided by Eötvös Loránd University under an agreement between Elsevier and the Hungarian EISZ consortium.

References

- [1] V. Mihalache, A. Salomaa, Lindenmayer and DNA: Watson-Crick DOL systems, *Bull. Eur. Assoc. Theor. Comput. Sci.* 62 (1997).
- [2] V. Mihalache, A. Salomaa, Language-theoretic aspects of DNA complementarity, *Theor. Comput. Sci.* 250 (1–2) (2001) 163–178, [https://doi.org/10.1016/S0304-3975\(99\)00129-2](https://doi.org/10.1016/S0304-3975(99)00129-2).
- [3] J. Csimá, E. Csuhaj-Varjú, A. Salomaa, Power and size of extended Watson-Crick L systems, *Theor. Comput. Sci.* 290 (3) (2003) 1665–1678, [https://doi.org/10.1016/S0304-3975\(02\)00074-9](https://doi.org/10.1016/S0304-3975(02)00074-9).
- [4] A. Salomaa, P. Sosík, Watson-Crick DOL systems: the power of one transition, *Theor. Comput. Sci.* 301 (1–3) (2003) 187–200, [https://doi.org/10.1016/S0304-3975\(02\)00580-7](https://doi.org/10.1016/S0304-3975(02)00580-7).
- [5] P. Sosík, Universal computation with Watson-Crick DOL systems, *Theor. Comput. Sci.* 289 (1) (2002) 485–501, [https://doi.org/10.1016/S0304-3975\(01\)00328-0](https://doi.org/10.1016/S0304-3975(01)00328-0).
- [6] P. Sosík, Watson-Crick DOL systems: generative power and undecidable problems, *Theor. Comput. Sci.* 306 (1–3) (2003) 101–112, [https://doi.org/10.1016/S0304-3975\(03\)00214-7](https://doi.org/10.1016/S0304-3975(03)00214-7).
- [7] A. Salomaa, Aspects of DNA complementarity, in: C.S. Calude, J. Casti, M. Dinneen (Eds.), *Unconventional Models of Computation*, Springer-Verlag, Singapore, 1998, pp. 94–107.
- [8] A. Salomaa, Watson-Crick walks and roads on DOL graphs, *Acta Cybern.* 14 (1) (1999) 179–192.
- [9] A. Salomaa, Iterated morphisms with complementarity on the DNA alphabet, in: M. Ito, G. Păun, S. Yu (Eds.), *Words, Semigroups, and Transductions - Festschrift in Honor of Gabriel Thierrin*, World Scientific, 2001, pp. 405–420.
- [10] A. Salomaa, Uni-translational Watson-Crick DOL systems, *Theor. Comput. Sci.* 281 (1–2) (2002) 537–553, [https://doi.org/10.1016/S0304-3975\(02\)00026-9](https://doi.org/10.1016/S0304-3975(02)00026-9).
- [11] J. Honkala, A. Salomaa, Watson-Crick DOL systems with regular triggers, *Theor. Comput. Sci.* 259 (1–2) (2001) 689–698, [https://doi.org/10.1016/S0304-3975\(01\)00010-X](https://doi.org/10.1016/S0304-3975(01)00010-X).
- [12] D. Sears, K. Salomaa, Extended Watson-Crick L systems with regular trigger languages and restricted derivation modes, *Nat. Comput.* 11 (4) (2012) 653–664, <https://doi.org/10.1007/S11047-012-9329-6>.
- [13] E. Csuhaj-Varjú, R. Freund, G. Vaszil, Watson-Crick TOL systems and red-green register machines, *Fundam. Inform.* 155 (1–2) (2017) 111–129, <https://doi.org/10.3233/FI-2017-1578>.
- [14] P. Sosík, Watson-Crick DOL systems: a survey, *J. Membr. Comput.* 5 (3) (2023) 182–189, <https://doi.org/10.1007/S41965-023-00130-Z>.
- [15] E. Csuhaj-Varjú, A. Salomaa, Networks of Watson-Crick DOL systems, in: M. Ito, T. Imaoka (Eds.), *Proceedings of the International Colloquium on Words, Languages & Combinatorics III*, Kyoto, Japan, March 14–18, 2000, World Scientific, 2000, pp. 134–150.
- [16] E. Csuhaj-Varjú, A. Salomaa, The power of networks of Watson-Crick DOL systems, in: N. Jonoska, G. Păun, G. Rozenberg (Eds.), *Aspects of Molecular Computing, Essays Dedicated to Tom Head on the Occasion of His 70th Birthday*, in: *Lecture Notes in Computer Science*, vol. 2950, Springer, 2004, pp. 106–118.
- [17] E. Csuhaj-Varjú, Computing by networks of Watson-Crick DOL systems, in: M. Ito (Ed.), *Proc. Algebraic Systems, Formal Languages and Computation*, in: *RIMS Kokyuroku*, vol. 1166, Research Institute for Mathematical Sciences, Kyoto University, 2000, pp. 43–51.
- [18] E. Csuhaj-Varjú, Networks of standard Watson-Crick DOL systems with incomplete information communication, in: J. Karhumäki, H.A. Maurer, G. Păun, G. Rozenberg (Eds.), *Theory Is Forever, Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday*, in: *Lecture Notes in Computer Science*, vol. 3113, Springer, 2004, pp. 35–48.
- [19] G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages, Word, Language, Grammar*, vol. 1, Springer, 1997.
- [20] G. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag Berlin Heidelberg, 1998.
- [21] V. Geffert, Normal forms for phrase-structure grammars, *RAIRO Theor. Inform. Appl.* 25 (1991) 473–496, <https://doi.org/10.1051/ITA/1991250504731>.