mathematics

Article

On the Languages Accepted by Watson-Crick Finite Automata with Delays

José M. Sempere

Valencian Research Institute for Artificial Intelligence (VRAIN), Universitat Politècnica de València, Camino de Vera, 46022 Valencia, Spain; jsempere@dsic.upv.es

Abstract: In this work, we analyze the computational power of Watson-Crick finite automata (WKFA) if some restrictions over the transition function in the model are imposed. We consider that the restrictions imposed refer to the maximum length difference between the two input strands which is called the delay. We prove that the language class accepted by WKFA with such restrictions is a proper subclass of the languages accepted by arbitrary WKFA in general. In addition, we initiate the study of the language classes characterized by WKFAs with bounded delays. We prove some of the results by means of various relationships between WKFA and sticker systems.

Keywords: Watson-Crick finite automata; sticker systems; DNA computing; formal languages

Citation: Sempere, J.M. On the Languages Accepted by Watson-Crick Finite Automata with Delays. *Mathematics* **2021**, *9*, 813. https://doi.org/10.3390/math9080813

Academic Editor: Matteo Cavaliere

Received: 28 February 2021
Accepted: 7 April 2021
Published: 9 April 2021

1. Introduction

DNA computing is a research area that draws its inspiration from the processes that take place in nature based on the functioning and structure of biomolecules, DNA, RNA, and proteins [1]. For more than 20 years, several models have been proposed in this research area, most of them universal and complete models, inspired by natural processes carried out in living cells. One of the first computational models proposed in the framework of DNA computing was the Watson-Crick finite automaton (WKFA) [2], which is a type of finite state abstract machine that takes its input tape from the DNA double strand and takes into account the complementarity relationships between symbols as it happens in nature with respect to the DNA nucleotides. That is, adenine (A) is related with thymine (T) and cytosine (C) is related with guanine (G), according to the Watson-Crick complementarity rule. The other aspect taken under consideration in this model is the ability of DNA to recombine. DNA recombination occurs at the ends of a molecule that has been previously denatured and fractionated, that is the primers of the molecule that have a length that allows to keep the molecules stable. The length of these "sticky ends" will be used in this work to characterize the language acceptation by WKFA. Several variants of the WKFA model have been proposed in recent years. We can mention, among others, the reversible WK automata [3], the unary WK automata [4], the WK Jumping Finite Automata [5], and the $5' \to 3'$ sensing WKFA [6], among others. In [7], a system with several WKFA and parallel communication is proposed. A survey on the WKFA model can be found in [8].

Another computation model proposed in the DNA computing framework was the sticker systems [9]. The characteristics of this model are similar to WKFA (double strands and complementarity) but within a generative approach. In this case, we have a finite set of (possibly incomplete) double strands (the axioms) and a finite set of rules (dominoes) that can enlarge the axioms by using an operation over strands called sticking. The sticking operation is based again on the DNA recombination by the double strand and the complementarity relationship. It could be considered that sticker systems and WKFA are very similar models whose main difference is that WKFA are accepting systems (at least in their basic definition), while sticker systems are generating systems.

In this work, we propose an additional restriction in order to accept or reject a double strand in the WKFA. We consider that the length of the primers of the double strand during the computation time, does not exceed a predefined positive integer value. We can impose this feature in the computations performed during the application of transitions in the WKFA, as is the case in the delayed computations of sticker systems [10], or we can define it explicitly in the transitions of the model. In this work we will propose the latter approach, and explicitly define the delays in the definition of the model transition function. Once the characteristic feature we are working with has been explicitly defined, we will proceed to study its computational power by relating it to the classical theory of formal languages. It is important to study such kinds of restrictions in the model because if we wish to see practical applications of the model such as its application in bioinformatics [11], then working with the model in a generic way implies having to solve extremely difficult problems such as working with ambiguous and non-deterministic models. By imposing these types of constraints, the model can be put into a practical context that allows the application of machine learning techniques in order to solve problems of a very diverse nature [12]. Hence, it is important: (1) To study the scope of the restrictions from a formal point of view (i.e., study the classes of languages that can be defined with the restricted model), and (2) explicitly define the restrictions imposed for better use in other application areas.

The structure of this paper is as follows: In the next section we will introduce the basic concepts about formal language theory and the WKFA and sticker systems that we will use in the rest of the work. In addition, we define the delays in sticker systems and we relate sticker systems with WKFA. In Section 3, we will introduce the WKFA model with bounded delays and we define some relationships between their language classes. In addition, we provide a sufficient condition to define bounded delays as the intial delays in the WKFA. Finally, in Section 4, we provide some results about the language classes in WKFA with only initial delays. We finish our work with some conclusions and future work on the topic addressed in this work.

2. Basic Concepts and Notation

In this section, we introduce basic concepts from formal language theory according to [13,14] and from stickers according to [1] that we will use in the sequel.

An alphabet Σ is a finite nonempty set of elements named symbols. A string defined over Σ is a finite ordered sequence of symbols from Σ. The infinite set of all the strings defined over Σ is denoted by Σ^*. Given a string $x \in \Sigma^*$ we denote its length by $|x|$. The empty string is denoted by λ and Σ^+ denotes $\Sigma^* - \{\lambda\}$. Given a string x we denote the reversal string of x by x^r. Given two strings $x = x_1x_2\cdots x_p$ and $y = y_1y_2\cdots y_m$, we denote the product (concatenation) of x by y as the string $xy = x_1x_2\cdots x_py_1y_2\cdots y_m$. A language L defined over Σ is a set of strings over Σ.

A grammar is a construct $G = (N, \Sigma, P, S)$ where N and Σ are the alphabets of auxiliary and terminal symbols with $N \cap \Sigma = \emptyset$, $S \in N$ is the axiom of the grammar and P is a finite set of productions in the form $\alpha \to \beta$, where $\alpha \in (N \cup \Sigma)^*N(N \cup \Sigma)^*$ and $\beta \in (N \cup \Sigma)^*$. The language of the grammar is denoted by $L(G)$ and it is the set of terminal strings that can be obtained from S by applying symbol substitutions according to P. Formally, $w_1 \underset{G}{\Rightarrow} w_2$ if $w_1 = u\alpha v$, $w_2 = u\beta v$, and $\alpha \to \beta \in P$. We denote the reflexive and transitive closure of $\underset{G}{\Rightarrow}$ by $\overset{*}{\underset{G}{\Rightarrow}}$. The language generated by G is defined by the set $L(G) = \{w \in \Sigma^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$.

We say that a grammar $G = (N, \Sigma, P, S)$ is right (left) linear (regular) if every production in P is in the form $A \to uB$ $(A \to Bu)$ or $A \to w$ with $A, B \in N$ and $u, w \in \Sigma^*$. The class of languages generated by right (left) linear grammars is the class of regular languages and is denoted by $\mathcal{REG}$. We say that a grammar $G = (N, \Sigma, P, S)$ is linear if every production in P is in the form $A \to uBv$ or $A \to w$ with $A, B \in N$ and $u, v, w \in \Sigma^*$. The class of languages generated by linear grammars is denoted by $\mathcal{LIN}$. A well-known result from formal language theory is the inclusion $\mathcal{REG} \subset \mathcal{LIN}$.

A deterministic finite automata (DFA) is an abstract machine $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of final states. The extension of the transition function over strings $\hat{\delta} : Q \times \Sigma^* \to Q$ is defined as $\hat{\delta}(q, \lambda) = q$, and $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ for every $x \in \Sigma^*$, and $a \in \Sigma$. In the following, we will not distinguish between δ and $\hat{\delta}$, and its meaning will depend on the function arguments. The language accepted by a DFA A is the set $L(A) = \{x \in \Sigma^* : \delta(q_0, x) \in F\}$. A classical result from formal language theory is that the class of languages accepted by deterministic finite automata is $\mathcal{REG}$. For the case of non-deterministic finite automata, the transition function is defined as $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, where $\mathcal{P}(Q)$ is the power set of Q. It is a classical result from formal language theory that the set of languages accepted by non-deterministic finite automata is again $\mathcal{REG}$.

Regular languages can be defined by regular expressions. They are a language specification based on definition rules that consider letters of the alphabet, the union, product and closure operators, and the use of parentheses to reorder the priority of operators.

A homomorphism h is defined as a mapping $h : \Sigma \to \Gamma^*$ where Σ and Γ are alphabets. We can extend the definition of homomorphisms over strings as $h(\lambda) = \lambda$ and $h(ax) = h(a)h(x)$ with $a \in \Sigma$ and $x \in \Sigma^*$. The homomorphism over a language $L \subseteq \Sigma^*$ is defined as $h(L) = \{h(x) : x \in L\}$.

Given an alphabet $\Sigma = \{a_1, \cdots, a_n\}$, we use the symmetric (and injective) relation of complementarity $\rho \subseteq \Sigma \times \Sigma$. For any string $x \in \Sigma^*$, we denote by $\rho(x)$ the string obtained by substituting the symbol a in x by the symbol b such that $(a, b) \in \rho$ (remember that ρ is injective) with $\rho(\lambda) = \lambda$.

Given an alphabet V, a sticker over V is the pair (x, y) such that $x = x_1vx_2$, $y = y_1wy_2$ with $x, y \in \Sigma^*$ and $\rho(v) = w$. The sticker (x, y) will be denoted by $\begin{pmatrix} x \\ y \end{pmatrix}$. A sticker $\begin{pmatrix} x \\ y \end{pmatrix}$ will be a complete and complementary molecule if $|x| = |y|$ and $\rho(x) = y$. A complementary and complete molecule $\begin{pmatrix} x \\ y \end{pmatrix}$ will be denoted as $\begin{bmatrix} x \\ y \end{bmatrix}$. The set of all complete and complementary molecules over V will be called the Watson-Crick domain and denoted by $WK_\rho(V)$.

We define the following sets, where ρ denotes the symmetric relation between the symbols in V:

$$L_\rho(V) = (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}) \begin{bmatrix} V \\ V \end{bmatrix}_\rho^*,$$

$$R_\rho(V) = \begin{bmatrix} V \\ V \end{bmatrix}_\rho^* (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}),$$

$$LR_\rho(V) = (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}) \begin{bmatrix} V \\ V \end{bmatrix}_\rho^+ (\begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}).$$

In addition, we can consider the set $W_\rho(V) = L_\rho(V) \cup R_\rho(V) \cup LR_\rho(V)$. Observe that $WK_\rho(V)$ is included in each set from $L_\rho(V)$, $R_\rho(V)$ and $LR_\rho(V)$.

Kari et al. [9] defined the sticking operation μ which can be considered as a product operation over stickers. Basically, given two stickers x and y, the operation $\mu(x, y)$ returns a new sticker by taking into account the complementarity relation ρ between the upper string in x and the lower string in y and vice versa.

Sticker systems were defined by Freund et al. in [15] and by Kari et al. in [9] as a generative model to apply the sticking operation μ (a product-like operation over stickers) to obtain languages. At the same time, Păun and Rozenberg [10] introduced some aspects which will be highly related to the present work such as the (bounded) delayed languages defined by sticker systems. A sticker system is defined by the tuple $\gamma = (V, \rho, A, D)$, where V is an alphabet, $\rho \subseteq V \times V$ is a symmetric (and injective) relation, D is a finite subset of $W_\rho(V) \times W_\rho(V)$ where its elements are called dominoes, and A is a finite subset of $LR_\rho(V)$ which are the axioms of the system. The derivation relation $\Rightarrow$ between stickers, according to a sticker system γ, can be defined as follows:

$$x \Rightarrow y \text{ iff } y = \mu(u_1, \mu(x, u_2)) \text{ such that } (u_1, u_2) \in D.$$

Given that $\Rightarrow$ is a relation between stickers subjected to a sticker system γ, we will denote the reflexive and transitive closure of $\Rightarrow$ by $\overset{*}{\Rightarrow}$. Now, we can define the language of complete and complementary molecules generated by the sticker system $\gamma = (V, \rho, A, D)$ with no restrictions as follows:

$$LM_n(\gamma) = \{w \in WK_\rho(V) : x \overset{*}{\Rightarrow} w, x \in A\}.$$

In addition, the language generated by γ is defined as:

$$L_n(\gamma) = \{x \in V^* : \begin{bmatrix} x \\ \rho(x) \end{bmatrix} \in LM_n(\gamma)\}.$$

The family of languages generated by arbitrary sticker systems is denoted by $\mathcal{ASL}(n)$.

The maximal length of an overhang in a sticker $z = \begin{pmatrix} x \\ y \end{pmatrix}$ is called the delay of z. Observe that the delay of a sticker z is the maximal length of the right or left overhang located in the upper or lower strand. Formally, if $z = u\begin{bmatrix} v \\ \rho(v) \end{bmatrix}w$, with $u, w \in \begin{pmatrix} \lambda \\ V^* \end{pmatrix} \cup \begin{pmatrix} V^* \\ \lambda \end{pmatrix}$, then the delay of z equals to the maximum value between $|u|$ and $|w|$, where $|u| = |u'|$ if $u = \begin{pmatrix} u' \\ \lambda \end{pmatrix}$ or $u = \begin{pmatrix} \lambda \\ u' \end{pmatrix}$, and $|w| = |w'|$ if $w = \begin{pmatrix} w' \\ \lambda \end{pmatrix}$ or $w = \begin{pmatrix} \lambda \\ w' \end{pmatrix}$. The delay associated to a derivation in a sticker system is the maximal delay of the stickers that are produced during that derivation. For any sticker system γ, the language of strings generated by γ with a delay d is denoted by $L_d(\gamma)$. A sticker system γ is said to have a bounded delay if there is $d > 1$ such that $L_d(\gamma) = L_n(\gamma)\cdot$ The family of languages generated by arbitrary sticker systems with a bounded delay is denoted by $\mathcal{ASL}(b)$.

Watson-Crick finite automata (WKFA) were defined first by Freund et al. [2] as an acceptance model to deal with stickers. It can be defined by the tuple $M = (V, \rho, Q, s_0, F, \delta)$, where Q and V are disjoint alphabets (states and symbols), $\rho \subseteq V \times V$ is a symmetric (and injective) relation of complementarity, $s_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \to \mathcal{P}(Q)$. A transition step in an arbitrary WK finite automaton M is denoted by $\underset{M}{\mapsto}$. So, $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} q \begin{pmatrix} x_2x_3 \\ y_2y_3 \end{pmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1x_2 \\ y_1y_2 \end{pmatrix} p \begin{pmatrix} x_3 \\ y_3 \end{pmatrix}$ iff $p \in \delta(q, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix})$, with $x_1, x_2, x_3, y_1, y_2, y_3 \in V^*$. The transitive and reflexive closure of $\underset{M}{\mapsto}$ is denoted by $\overset{*}{\underset{M}{\mapsto}}$. We can use a normal form such that for every transition $q \in \delta(q, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})$ then $|x_1x_2| = 1$. This normal form defines the so-called 1-*limited* WK finite automata and they were proved to be equivalent to arbitrary ones [2].

The language of complete and complementary molecules accepted by M will be defined by the set $L_m(M) = \{\begin{bmatrix} x \\ y \end{bmatrix} : s_0 \begin{bmatrix} x \\ y \end{bmatrix} \overset{*}{\underset{M}{\mapsto}} \begin{bmatrix} x \\ y \end{bmatrix} q$, with $q \in F$ and $x, y \in V^*\}$, while the upper strand language accepted by M will be defined as:

$$L_u(M) = \{x \in V^* : s_0 \begin{bmatrix} x \\ y \end{bmatrix} \overset{*}{\underset{M}{\mapsto}} \begin{bmatrix} x \\ y \end{bmatrix} q, q \in F \wedge x, y \in V^*\}.$$

The class of languages accepted by arbitrary WKFA in the upper strand is denoted by $\mathcal{AWK}$.

We can use a normal form such that for every transition $q \in \delta(q, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})$ then $x_1 = \lambda$ or $x_2 = \lambda$. This normal form defines the so called simple WKFA and they were proved to be equivalent to arbitrary ones [2].

Now, we will relate the languages in $\mathcal{AWK}$ with the languages generated by sticker systems through the following result.

Theorem 1. *Let $M = (V, \rho, Q, s_0, F, \delta)$ be a simple WKFA. Then there exists a sticker system $\gamma = (V', \rho', A, D)$, a morphism h and a regular language R which satisfies $h(L_n(\gamma) \cap R) = L_u(M)$.*

Proof. Let us take $M = (V, \rho, Q, s_0, F, \delta)$ to be a simple WKFA with $Q = \{s_0, q_1, \ldots, q_n\}$ and $V = \{a_1, \cdots, a_n\}$. Observe that this restriction over WKFA does not affect the computational capacity of the model, given that simple is a normal form for WKFA. We can define the sticker system $\gamma = (V', \rho', A, D)$ where the elements of γ are defined as follows:

- $V' = V \cup Q \cup \{q_f, \#\}$, where $q_f, \# \notin Q \cup V$;
- ρ' is defined as:

$$\rho'(a) = \begin{cases} a & \text{if } a \in Q \cup \{q_f, \#\} \\ \rho(a) & \text{if } a \in V \\ ; & \end{cases}$$

- The set of axioms A is defined through the following rules:
    1. $\begin{pmatrix} \# \\ s_0\# \end{pmatrix}$

       This axiom places the initial state of M as the first state in the sequence defined to accept any string.
    2. $\begin{pmatrix} s_0\# \\ qs_0\# \end{pmatrix}$ where $q \in \delta(s_0, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix})$

       An axiom is created for every transition from the initial state of the WKFA, s_0. The overhang end on the left lower strand of the axiom corresponds to the state of the WKFA that is reached from the initial state with a λ-movement.
    3. $\begin{pmatrix} s_0\# \\ q_f s_0\# \end{pmatrix}$ if $s_0 \in F$

       If the initial state in the WKFA is final too, then this axiom is inserted. That is, the additional state q_f is produced from the initial state without any input symbol.
- The set D is defined by the pairs according to the following rules:
    1. $(\begin{pmatrix} q \\ p \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix})$ where $p \in \delta(q, \begin{pmatrix} x \\ y \end{pmatrix})$ and $x = \lambda$ or $y = \lambda$.

       That means that, in M, it is possible to transit from q to p with the pair $\begin{pmatrix} x \\ y \end{pmatrix}$.
    2. $(\begin{pmatrix} \#q \\ q_f\# \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix})$ where $q \in F$.

       For each final state $q_i \in F$ of the WKFA, these stickers mark the end of the state sequence in the WKFA to accept the input.
    3. $(\begin{pmatrix} q_f \\ \lambda \end{pmatrix}, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix})$

       This rule is added to avoid overhang ends in the left side of the sticker system. When this rule is added there is no way to add another one according to the automata transitions.

  Now, observe the following accepting computation in M:

$$s_0 \begin{bmatrix} x \\ y \end{bmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} q_{i_1} \begin{pmatrix} x_2 \ldots x_n \\ y_2 \ldots y_n \end{pmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1 x_2 \\ y_1 y_2 \end{pmatrix} q_{i_2} \begin{pmatrix} x_3 \ldots x_n \\ y_3 \ldots y_n \end{pmatrix} \underset{M}{\mapsto} \cdots \underset{M}{\mapsto} \begin{pmatrix} x_1 x_2 \ldots x_n \\ y_1 y_2 \ldots y_n \end{pmatrix} q_{i_n},$$

  where $q_{in} \in F$. The following derivation is hold in γ:

$$\binom{\#}{s_0\#} \underset{\gamma}{\Rightarrow} \binom{s_0\#x_1}{q_{i_1}s_0\#y_1} \underset{\gamma}{\Rightarrow} \binom{q_{i_1}s_0\#x_1x_2}{q_{i_2}q_{i_1}s_0\#y_1y_2} \underset{\gamma}{\Rightarrow} \cdots \underset{\gamma}{\Rightarrow} \binom{q_{i_{n-1}}\cdots q_{i_1}s_0\#x_1\ldots x_n}{q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#y_1\ldots y_n} \underset{\gamma}{\Rightarrow} \cdots$$

$$\cdots \underset{\gamma}{\Rightarrow} \binom{\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#x_1\ldots x_n}{q_f\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#y_1\ldots y_n} \underset{\gamma}{\Rightarrow} \binom{q_f\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#x_1\ldots x_n}{q_f\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#y_1\ldots y_n},$$

and it can easily be shown that $x_1\ldots x_n \in L_u(M)$ iff $q_f\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#x_1\ldots x_n \in L_n(\gamma)$.

A regular set R is defined by the regular expression:

$$q_f\#(s_0 + q_1 + \cdots + q_n)^*\#(a_1 + a_2 + \cdots + a_n)^*.$$

The above regular expression guarantees that only strings containing a single symbol q_f can be produced and, therefore, in the sticker system, the molecules obtained correspond only to those that have reached a final state in the WKFA. On the other hand, note that without the set R acting as a control set, the sticker system could generate molecules that would not be recognized in the automaton. In other words, nothing prevents the sticker system from continuing to add stickers that could complete new molecules once the molecule accepted by the automaton has been generated.

Finally, we can define the morphism $h : V' \to V$ as follows:

$$h(a) = \begin{cases} a & \text{if } a \in V \\ \lambda & \text{if } a \notin V. \end{cases}$$

Hence, $q_f\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#x_1\ldots x_n \in L_n(\gamma) \cap R$, $h(q_f\#q_{i_n}q_{i_{n-1}}\cdots q_{i_1}s_0\#x_1\ldots x_n) = x_1\ldots x_n$, and the theorem holds.

It can be observed that the sequence of states needed to accept any string x in M is defined by the reverse of the sequence of states between the # marks in the stickers obtained in γ. $\square$

Example 1. *Let us take the simple WKFA defined by the transition diagram in Figure 1.*

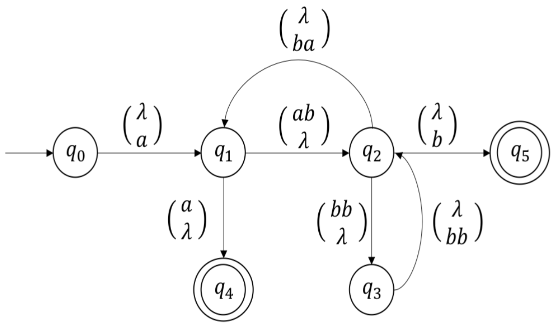

Figure 1. A simple Watson-Crick finite automata (WKFA).

From the previous WKFA, we build a sticker system, such that A and D are defined as follows:

$$A = \{\binom{\#}{q_0\#}\}.$$

$$D = \{(\binom{q_0}{q_1},\binom{\lambda}{a}),(\binom{q_1}{q_4},\binom{a}{\lambda}),(\binom{q_1}{q_2},\binom{ab}{\lambda}),(\binom{q_2}{q_1},\binom{\lambda}{ba}),(\binom{q_2}{q_3},\binom{\lambda}{bb}),$$
$$(\binom{q_2}{q_5},\binom{\lambda}{b}),(\binom{q_3}{q_2},\binom{\lambda}{bb}),(\binom{\#q_4}{q_f\#},\binom{\lambda}{\lambda}),(\binom{\#q_5}{q_f\#},\binom{\lambda}{\lambda}),(\binom{q_f}{\lambda},\binom{\lambda}{\lambda})\}.$$

The regular expresión defined to control the stickers generation is defined as follows:

$$q_f\#(q_0+q_1+q_2+q_3+q_4+q_5)^*\#(a+b)^*.$$

3. Accepting Languages with Bounded Delays in WKFA

In this section, we will introduce the delays during a WKFA computation in a way similar to sticker systems [10].

Let us suppose that we consider the following computation in a WKFA $M = (V, \rho, Q, s_0, F, \delta)$ with $x = x_1x_2\ldots x_n$ and $y = y_1y_2\ldots y_n$:

$$q_0\begin{bmatrix}x\\y\end{bmatrix} \underset{M}{\mapsto} \begin{pmatrix}x_1\\y_1\end{pmatrix} q_{i_1} \begin{pmatrix}x_2\ldots x_n\\y_2\ldots y_n\end{pmatrix} \underset{M}{\mapsto} \begin{pmatrix}x_1x_2\\y_1y_2\end{pmatrix} q_{i_2} \begin{pmatrix}x_3\ldots x_n\\y_3\ldots y_n\end{pmatrix} \underset{M}{\mapsto} \cdots \underset{M}{\mapsto} \begin{pmatrix}x_1x_2\ldots x_n\\y_1y_2\ldots y_n\end{pmatrix} q_{i_n},$$

where $x_j, y_j \in V^*$, $q_{i_j} \in Q$, if $0 \leq |x_1x_2\ldots x_j| - |y_1y_2\ldots y_j| \leq d$, for $1 \leq j \leq n$, and $q_{i_n} \in F$ then we say that M accepts $\begin{bmatrix}x\\y\end{bmatrix}$ with an upper bounded delay d. If $0 \leq |y_1y_2\ldots y_j| - |x_1x_2\ldots x_j| \leq d$, for $1 \leq j \leq n$ we say that M accepts $\begin{bmatrix}x\\y\end{bmatrix}$ with a lower bounded delay d. If $||x_1x_2\ldots x_j| - |y_1y_2\ldots y_j|| \leq d$, for $1 \leq j \leq n$ we say that M accepts $\begin{bmatrix}x\\y\end{bmatrix}$ with an arbitrary bounded delay d. The language accepted by M with upper (lower, arbitrary) bounded delay d in the upper strand is denoted by $L_{u,updel}(M,d)$ ($L_{u,lowdel}(M,d), L_{u,del}(M,d)$). The class of languages accepted by arbitrary WKFA in the upper strand with upper (lower, arbitrary) bounded delay equals to d will be denoted by $\mathcal{AWK}_{updel}(d)$ ($\mathcal{AWK}_{lowdel}(d)$, $\mathcal{AWK}_{del}(d)$).

The definition of delays in the computation of the automaton is a constraint that reduces the computational power of the model. This can be formalized by the following result.

Lemma 1. *For every integer value $d > 0$, $\mathcal{AWK}_{updel}(d) \subsetneq \mathcal{AWK}$.*

Proof. Let us consider the language $L = \{wcw : w \in \{a,b\}^*\}$. This language belongs to $\mathcal{AWK}$ given that it can be accepted by the following WKFA.

$$\begin{array}{lll}
\delta(q_0, \begin{pmatrix}a\\\lambda\end{pmatrix}) = \{q_0\} & \delta(q_0, \begin{pmatrix}b\\\lambda\end{pmatrix}) = \{q_0\} & \delta(q_0, \begin{pmatrix}c\\\lambda\end{pmatrix}) = \{q_1\} \\
\delta(q_1, \begin{pmatrix}a\\a\end{pmatrix}) = \{q_1\} & \delta(q_1, \begin{pmatrix}b\\b\end{pmatrix}) = \{q_1\} & \delta(q_1, \begin{pmatrix}\lambda\\c\end{pmatrix}) = \{q_2\} \\
\delta(q_2, \begin{pmatrix}\lambda\\a\end{pmatrix}) = \{q_2\} & \delta(q_2, \begin{pmatrix}\lambda\\b\end{pmatrix}) = \{q_2\}, &
\end{array}$$

with $F = \{q_2\}$ and $\rho = \{(a,a),(b,b),(c,c)\}$. Observe that the automaton reads first in the upper strand the string w until it reaches the symbol c always being in the state q_0. Once the symbol c has been read, the automaton changes its state and it then starts to match the string after the symbol c with the string starting on the lower strand. If the symbol c is found in the lower strand then it proceeds to read the rest of the input string in the lower strand.

It is easy to see that for any delay value $d > 0$, a string wcw with $|w| > d$ will not be parsed with a delay less than or equal to d given that the automaton needs to read the string w first before arriving to the symbol c. In such a case, the automaton produces a delay greater than d given that $|w| > d$. □

Now, we are going to consider the relationships between the upper, lower, and arbitrary delays. In this case, we will work with arbitrary WKFA and the identity as the relation ρ defined in the automata. Observe that this does not detract from the generality of our results since the complementarity relationship can be reduced to the identity as shown in [16].

Lemma 2. *For every integer value $d > 0$, $\mathcal{AWK}_{lowdel}(d) = \mathcal{AWK}_{updel}(d)$.*

Proof. Let $M_1 = (V, \rho, Q, s_0, F, \delta_1)$ be an arbitrary WKFA with ρ being the identity relation, and L be the language accepted by M_1 with a bounded upper (lower) delay d. Then, from M_1 we can obtain the WKFA $M_2 = (V, \rho, Q, s_0, F, \delta_2)$ where $p \in \delta_2(q, \begin{pmatrix} y \\ x \end{pmatrix})$ if $p \in \delta_1(q, \begin{pmatrix} x \\ y \end{pmatrix})$. It is easy to see that if w is accepted by M_1 with an upper (lower) delay d, then w is accepted by M_2 by a lower (upper) delay given that the relation ρ is the identity and the transitions in M_2 swap the contents of the upper and lower strands. □

Lemma 3. $\mathcal{AWK}_{updel}(0) = \mathcal{REG}$.

Proof. Let $M = (V, \rho, Q, s_0, F, \delta)$ be an arbitrary WKFA and $L = L_{u,updel}(M, 0)$. For every $x \in L_{u,updel}(M, 0)$ there exists a computation in M as follows:

$$q_0 \begin{bmatrix} x \\ x \end{bmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1 \\ x_1 \end{pmatrix} q_{i_1} \begin{pmatrix} x_2 \ldots x_n \\ x_2 \ldots x_n \end{pmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1 x_2 \\ x_1 x_2 \end{pmatrix} q_{i_2} \begin{pmatrix} x_3 \ldots x_n \\ x_3 \ldots x_n \end{pmatrix} \underset{M}{\mapsto} \cdots \underset{M}{\mapsto} \begin{pmatrix} x_1 x_2 \ldots x_n \\ x_1 x_2 \ldots x_n \end{pmatrix} q_{i_n}.$$

Observe that ρ is the identity relation and, we can assume that $x_i \in V$, $1 \leq i \leq n$. Then, from M we can obtain a non-deterministic finite automata $A = (Q, V, s_0, f, F)$, such that $p \in f(q, a)$ iff $p \in \delta(q, \begin{pmatrix} a \\ a \end{pmatrix})$.

It is easy to see that $x \in L(A) = L$. Hence, $\mathcal{AWK}_{updel}(0) \subseteq \mathcal{REG}$. In order to see that $\mathcal{REG} \subseteq \mathcal{AWK}_{updel}(0)$, we take any arbitrary deterministic finite automata $A = (Q, \Sigma, f, q_0, F)$, and we obtain the WKFA $M = (\Sigma, \rho, Q, q_0, F, \delta)$ where ρ is the identity relation, and $p \in \delta(q, \begin{pmatrix} a \\ a \end{pmatrix})$ iff $f(q, a) = p$. Again, it is easy to see that $x \in L_{u,updel}(M, 0)$ iff $x \in L(A)$. Then, $\mathcal{REG} \subseteq \mathcal{AWK}_{updel}(0)$, and the lemma holds. □

A Sufficient Condition for Bounded Delays in WKFA

We have seen how delays during computation, regardless of whether they occur on the upper or lower strand, can be variable. That is, given a delay value d, the computation can produce that delay incrementally (i.e., starting with a delay less than d and then increasing it to the required value). In this section, we will propose a way to produce the required delay instantaneously. Informally, the approach we are going to propose requires producing the desired delay in the first movement of the computation and, subsequently, maintaining it until a point in time where it is completely reduced. This, without being a normal form, can be interesting when dealing with WKFA inductive inference as proposed in [12].

We say that M accepts x in the upper strand with an initial delay k iff M applies the following transition sequences to accept $x = x_1 x_2 \cdots x_l$. Remember that we can use the identity relation without loss of generality.

1. **First movement (initial delay)**

$$s_0 \begin{pmatrix} x_1 x_2 \cdots x_l \\ x \end{pmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1 x_2 \cdots x_k \\ \lambda \end{pmatrix} s_1 \begin{pmatrix} x_{k+1} \cdots x_l \\ x \end{pmatrix}.$$

2. **Delay propagation** $\forall i \geq 1$ such that $(i+1) \cdot k < l$

$$\begin{pmatrix} x_1 \cdots x_{i \cdot k} \\ \rho(x_1 \cdots x_{(i-1) \cdot k}) \end{pmatrix} p \begin{pmatrix} x_{i \cdot k+1} \cdots x_l \\ \rho(x_{(i-1) \cdot k+1} \cdots x_l) \end{pmatrix} \underset{M}{\mapsto} \begin{pmatrix} x_1 \cdots x_{(i+1) \cdot k} \\ \rho(x_1 \cdots x_{i \cdot k}) \end{pmatrix} q \begin{pmatrix} x_{(i+1) \cdot k+1} \cdots x_l \\ \rho(x_{i \cdot k+1} \cdots x_l) \end{pmatrix}.$$

3. **Final movement** $\forall m \geq 0$ such that $l - m + 1 < k$

$$\binom{x_1\cdots x_m}{\rho(x_1\cdots x_{m-k})}p\binom{x_{m+1}\cdots x_l}{\rho(x_{m-k+1}\cdots x_l)}\underset{M}{\mapsto}\binom{x_1\cdots x_l}{\rho(x_1\cdots x_l)}q.$$

Therefore, M accepts x in the upper strand by first processing k symbols of the upper strand, and then processing alternately k symbols of the lower and upper strand. When there are less than k symbols left in the upper strand, all the remaining symbols are processed. Figure 2 shows graphically how the stickers are organized as they are processed with initial delays.

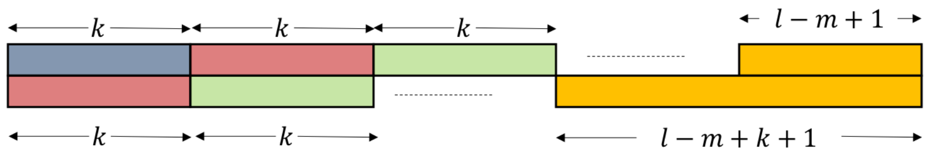

Figure 2. Stickers organization to be processed with initial delays.

Observe that the movements of the automata can adapted to be simple: The movement $q \in \delta(p, \binom{x}{y})$, with $|x| = |y| = k$ can be replaced by the following two movements $q' \in \delta(p, \binom{\lambda}{y})$ and $q \in \delta(q', \binom{x}{\lambda})$. Hence, the scheme to process the stickers in a simple WKFA with initial delays is shown in Figure 3.

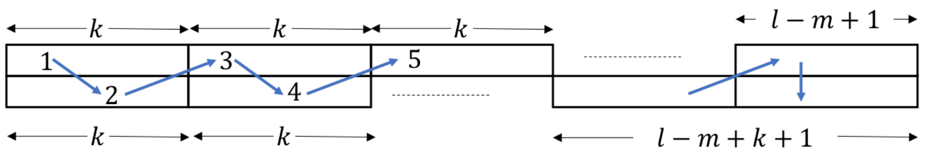

Figure 3. Order in processing stickers with initial delays.

Observe that according to the previous figure, in the WKFA the delay always equals to k given that first it is produced in the upper strand and then the lower strand is completed. In the following we will work with simple WKFA with only initial delays.

4. Languages Accepted by WKFA with Only Initial Delays

Let M be a simple WKFA with only initial delay movements, and let d be the delay at every computation step in M. If a sticker system γ is built from M as established in Theorem 1, then $L_n(\gamma) \in \mathcal{ASL}(b)$. This is due to the fact that, according to the definition of a WKFA with an initial delay d, the length of the delay that may appear in any of the two strands when processing a sequence cannot be longer than d. Therefore, in the sticker system, during its construction, the overhang end in the right side cannot be longer than d, and, in the left side, longer than one.

Example 2. *Let us take the simple WKFA with initial delays defined by the transition diagram of Figure 4. Observe that the bounded delay in this case equals to 2.*

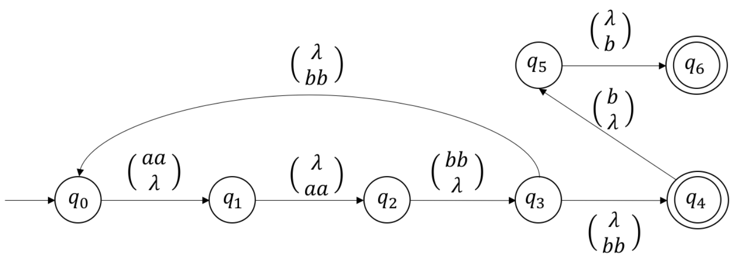
$$\begin{tikzcd}
 & & & q_5 \arrow[r, "\binom{\lambda}{b}"] & q_6 \\
\arrow[r] & q_0 \arrow[r, "\binom{aa}{\lambda}"] & q_1 \arrow[r, "\binom{\lambda}{aa}"] & q_2 \arrow[r, "\binom{bb}{\lambda}"] & q_3 \arrow[lll, bend right, "\binom{\lambda}{bb}"'] \arrow[r, "\binom{\lambda}{bb}"'] & q_4 \arrow[llu, "\binom{b}{\lambda}"']
\end{tikzcd}$$

Figure 4. A simple WKFA with initial delays equal to 2.

Observe that the bounded delay in this case equals to 2 and $L_u(M) = L_{u,updel}(M,2)$.
From the previous WKFA, we build a sticker system γ, such that A and D are defined as follows:

$A = \{\binom{\#}{q_0\#}\}$.
$D = \{(\binom{q_0}{q_1}, \binom{aa}{\lambda}), (\binom{q_1}{q_2}, \binom{\lambda}{aa}), (\binom{q_2}{q_3}, \binom{bb}{\lambda}), (\binom{q_3}{q_0}, \binom{\lambda}{bb}), (\binom{q_3}{q_4}, \binom{\lambda}{bb}),$
$(\binom{q_4}{q_5}, \binom{b}{\lambda}), (\binom{q_5}{q_6}, \binom{\lambda}{b})(\binom{\#q_4}{q_f\#}, \binom{\lambda}{\lambda}), (\binom{\#q_6}{q_f\#}, \binom{\lambda}{\lambda}), (\binom{q_f}{\lambda}, \binom{\lambda}{\lambda})\}$.

In this case, $L_u(M) = L_n(\gamma)$ which can be defined by the regular expression:

$$(aabb)(aabb)^*(b+\lambda),$$

given that, once a molecule is completed in γ, after having reached the final state q_4, there is no possibility of lengthening the molecule and completing a larger one, as the upper and lower strand would not be complementary.

Thus, the definition of a regular expression for derivative control, as in the proof of Theorem 1, is completely unnecessary in the case of initial delays. This aspect will be formalized by the following theorem.

Theorem 2. *Let $M = (V, \rho, Q, s_0, F, \delta)$ be a simple WKFA with only initial delays. Then there exists a sticker system $\gamma = (V', \rho', A, D)$ and a morphism h which satisfies $h(L_n(\gamma)) = L_u(M)$.*

Proof. Let $M = (V, \rho, Q, s_0, F, \delta)$ be a simple WKFA with only initial delays. Then we define a sticker system γ and a morphism h as in the proof of Theorem 1. Observe that in this case, there is no way to keep on derivating a sticker once the pairs $(\binom{\#q}{q_f\#}, \binom{\lambda}{\lambda})$ and $(\binom{q_f}{\lambda}, \binom{\lambda}{\lambda})$ have been applied given that the rest of elements in D do not preserve the complementarity since the relation ρ is the identity. □

It is known that $\mathcal{ASL}(b) \subseteq \mathcal{LIN}$ (Theorem 4.3 in [10]). That theorem is proven by building a linear grammar $G = (N, T, S, P)$ from a sticker system $\gamma = (V, \rho, A, D)$. In the grammar G, the molecules are obtained by a derivation from the ends of the molecule until reaching its central part in a reverse process to the one that happens in the sticker system. We can take advantage of this result together with the results we have proven before to characterize the language accepted by the WKFA with only initial delays. It is established by the following result.

Theorem 3. *Let $M = (V, \rho, Q, s_0, F, \delta)$ be a simple WKFA with only initial delays. Then $L_u(M) \in \mathcal{REG}$.*

Proof. A sticker system γ can be obtained from M, according to the proof of Theorem 1. Let $\gamma = (V', \rho', A, D)$ and $V' = V \cup Q \cup \{\#, q_f\}$ be such a sticker system. Let $G = (N, \begin{bmatrix} V' \\ V' \end{bmatrix}, S, P)$ be a linear grammar built from γ as described in [10]. It can be observed that, for every production in G in the form $A \to \alpha B \beta$ the sticker α is defined from the symbols in $Q \cup \{\#, q_f\}$ while the sticker β is defined from the symbols in V.

Now, a left linear grammar $G' = (N, V, S, P')$ can be constructed from G. The set of productions in P' is defined as follows:

- $A \to Bv_1$ if $A \to \binom{u_1}{u_2} B \binom{v_1}{v_2} \in P$;

- $A \to h(\omega_1)$ if $A \to \binom{\omega_1}{\omega_2} \in P$,

where the morphism $h : V' \to V$ is defined as $h(a) = a$ if $a \in V$ and $h(a) = \lambda$ if $a \in Q \cup \{\#, q_f\}$. The language generated by the grammar G' is the result of removing the symbols in $Q \cup \{\#, q_f\}$ from the strings generated by G in the upper strand. These strings are the strings in $L_u(M)$. Since G' is a left linear grammar and $L(G') = L_u(M)$ then $L_u(M) \in \mathcal{REG}$. □

5. Conclusions

In this paper we addressed some aspects of the languages defined by WKFA where the delay between the two strands in obtaining the upper strings is bounded. This aspect is interesting because, on the one hand, it allows us to relate two classical DNA computational models such as Watson-Crick finite automata and sticker systems. On the other hand, it allows us to establish a result that limits the computational capacity of the WKFA model, i.e., by introducing a bounded delay, we are reducing the class of languages that the model is able to accept.

Another aspect we want to highlight is that the study of language families characterized by bounded delays is a prior step to establishing efficient methods for their inductive inference, that is the application of machine learning techniques to obtain DNA computational models. In this sense, some previous results allow us to approach this aspect in a reasonable way [12]. The imposition of some kind of constraints and normal forms for some classes of languages accepted by WKFA is important in establishing the machine learning algorithms to infer WKFA from positive examples. It is well known in formal language theory that from linear languages there exist inherently ambiguous languages [17]. Since WKFA can be represented by intersections of linear and even linear languages [18], then the inherent ambiguity of some languages accepted by this model is unavoidable. This leads to multiple options, when specifying a set of transitions during the elaboration of a model by using machine learning techniques. The imposition of features such as initial delays makes the ambiguity disappear given that a predefined form is imposed on the transitions and makes it feasible to automatically learn some subclasses of this model.

There are additional aspects of this work that deserve to be explored in future research. For example, one aspect to consider is whether delays induce a hierarchy of languages in a gradual way, i.e., whether any language accepted by a WKFA with a bounded delay d can be accepted by a WKFA with a smaller delay. In the case of a negative answer to this question, we could obtain an infinite hierarchy, where the smallest class would be defined by the class of regular languages (null delays) which, in turn, would contain the languages that can be accepted only with initial delays.

Funding: This work has been developed with the financial support of the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952215 corresponding to the TAILOR project.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Păun, G.; Rozenberg, G.; Salomaa, A. *DNA Computing. New Computing Paradigms*; Springer: Berlin/Heidelberg, Germany, 1998.
2. Freund, R.; Păun, G.; Rozenberg, G.; Salomaa, A. Watson-Crick finite automata. In Proceedings of the DNA Based Computers III DIMACS Workshop, Pennsylvania, PA, USA, 23–25 June 1997; Rubin, H., Wood, D.H., Eds.; The American Mathematical Society: Providence, RI, USA, 1999; pp. 297–327.
3. Chatterjee, K.; Sankar Ray, K. Reversible Watson-Crick automata. *Acta Inform.* **2017**, *54*, 487–499. [CrossRef]
4. Chatterjee, K.; Sankar Ray, K. Unary Watson-Crick automata. *Theor. Comput. Sci.* **2019**, *782*, 107–112. [CrossRef]

5. Mahalingam, K.; Mishra, U.K.; Raghavan, R. Watson-Crick Jumping Finite Automata. *Int. J. Found. Comput. Sci.* **2020**, *31*, 891–913. [CrossRef]
6. Nagy, B. On $5' \to 3'$ Sensing Watson-Crick Finite Automata. In Proceedings of the 13th International Meeting on DNA Computing, DNA13, Memphis, TN, USA, 4–8 June 2007; Garzon, M.H., Yan, H., Eds.; DNA Computing; Springer LNCS: Berlin/Heidelberg, Germany, 2008; Volume 4848, pp. 256–262.
7. Czeizler, E.; Czeizler, E. Parallel communicating Watson-Crick automata systems. *Acta Cybern.* **2006**, *17*, 685–700. [CrossRef]
8. Czeizler, E.; Czeizler, E. A short survey on Watson-Crick automata. *Bull. EATCS* **2006**, *88*, 104–119.
9. Kari, L.; Păun, G.; Rozenberg, G.; Salomaa, A.; Yu, S. DNA computing, sticker systems, and universality. *Acta Inform.* **1998**, *35*, 401–420. [CrossRef]
10. Păun, G.; Rozenberg, G. Sticker Systems. *Theor. Comput. Sci.* **1998**, *204*, 183–203. [CrossRef]
11. Sempere, J.M. On the Application of Watson-Crick Finite Automata for the Resolution of Bioinformatic Problems. In Proceedings of the Invited Talk at the Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018), Košice, Slovakia, 21–22 August 2018; Freund, R., Hospodár, M., Jirásková, G., Pighizzini, G., Eds.; Österreichische Computer Gesellschaft: Vienna, Austria, 2018; pp. 29–30.
12. Sempere, J.M. Learning Context-Sensitive Languages from Linear Structural Information. In Proceedings of the 9th International Colloquium on Grammatical Inference ICGI 2008, Saint-Malo, France, 22–24 September 2008; Clark, A., Coste, F., Miclet, L., Eds.; Springer LNAI: Berlin/Heidelberg, Germany, 2008; Volume 5278, pp. 175–186.
13. Hopcroft, J.; Ullman, J. *Introduction to Automata Theory, Languages and Computation*; Addison Wesley Publishing Co.: Boston, MA, USA, 1979.
14. Rozenberg, G.; Salomaa, A. (Eds.) *Handbook of Formal Languages*; Springer: Berlin/Heidelberg, Germany, 1997.
15. Freund, R.; Păun, G.; Rozenberg, G.; Salomaa, A. Bidirectional sticker systems. In Proceedings of the Third Annual Pacific Conference on Biocomputing, Maui, HI, USA, 4–9 January 1998; Altman, R.B., Dunker, A.K., Hunter, L., Klein, T.E., Eds.; World Scientific: Singapore, 1998; pp. 535–546.
16. Kuske, D.; Weigel, P. The Role of the Complementarity Relation in Watson-Crick Automata and Sticker Systems. In Proceedings of the 8th International Conference DLT 2004, Auckland, New Zealand, 13–17 December 2004; Calude Elena Calude, C.S., Dinneen, M.J., Eds.; Springer LNCS: Berlin/Heidelberg, Germany, 2004; Volume 3340, pp. 272–283.
17. Greibach, S. The Undecidability of the Ambiguity Problem for Minimal Linear Grammars. *Inf. Control* **1963**, *6*, 119–125. [CrossRef]
18. Sempere, J.M. A Representation Theorem for Languages accepted by Watson-Crick Finite Automata. *Bull. EATCS* **2004**, *83*, 187–191.