PAPER

The Relations among Watson-Crick Automata and Their Relations with Context-Free Languages

Satoshi OKAWA^{†a)} and Sadaki HIROSE^{††b)}, *Members*

SUMMARY Watson-Crick automata were introduced as a new computer model and have been intensively investigated regarding their computational power. In this paper, aiming to establish the relations among language families defined by Watson-Crick automata and the family of context-free languages completely, we obtain the following results. (1) $F1WK = FSWK = FWK$, (2) $FWK = AWK$, (3) there exists a language which is not context-free but belongs to NWK , and (4) there exists a context-free language which does not belong to AWK .

key words: DNA computer, Watson-Crick automata, formal languages, language family

1. Introduction

The limits of computational power in traditional silicon-based computers are becoming obvious as the demands to solve large-scale problems are increasing. To resolve this situation, many computing systems have been proposed and investigated in regards to their computational power. Among them, the DNA computer is one of the most attractive models.

Watson-Crick automata were introduced in [1] as a new computer model and have been intensively investigated regarding their computational power. The input is a double-stranded tape, called a Watson-Crick tape, which is considered as a double helix model of DNA strands discovered by Watson and Crick. A Watson-Crick automaton reads an input tape by scanning the upper strand and the lower strand separately and changes its state by a transition function. In [1], posing some restrictions such as stateless, all-final, simple, and 1-limited on Watson-Crick automata, several variants of Watson-Crick automata were defined, and the computational powers of such automata and the relations among language families defined by them and families in a Chomsky hierarchy were investigated. In [4], Hirose et al. investigated and obtained some relations of them by showing explicitly languages that belong to one family but do not belong to others.

In this paper, following the research mentioned above, aiming to fully establish the relations among language fam-

ilies defined by Watson-Crick automata (WK families) and the family of context-free languages, we obtain the following results. (1) $F1WK = FSWK = FWK$, (2) $FWK = AWK$, (3) there exists a language which is not context-free but belongs to NWK , and (4) there exists a context-free language which does not belong to AWK .

The paper is organized as follows. In Sect. 2, we introduce the definitions, notations, and known results from [1], [3], [4], and then offer technical preparation and some properties needed later in the paper. In Sects. 3 and 4, we investigate the relations among WK families and the relations between the WK families and the CF family, respectively. In Sect. 5, concluding remarks are stated.

2. Preliminaries

2.1 A Model of DNA Strands

DNA strands consist of polymer chains which are composed of nucleotides. Nucleotides differ only in the combination of bases which they are comprised of. There are four bases: A (Adenine), G (Guanine), C (Cytosine), T (Thymine). The double helix of DNA arises by the bonding of two separate strands. Bonding happens by the pairwise attraction of bases: The base A always bonds with T, and G with C. This phenomenon is known as the Watson-Crick complementarity. Based on these facts, we will define DNA strands.

Consider an alphabet Σ and a symmetric and complementarity relation $\rho \subseteq \Sigma \times \Sigma$. For a and b , when the relation ρ holds, that is, $(a, b) \in \rho$, we denote it by $\begin{bmatrix} a \\ b \end{bmatrix}$ and

$$\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho} = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \mid a, b \in \Sigma, (a, b) \in \rho \right\}.$$

As usual in the formal language field, we denote by $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho}^*$ the set of all strings of symbols $\begin{bmatrix} a \\ b \end{bmatrix}$ in $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho}$, and empty strings in Σ^* and $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho}^*$ by λ and $\begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$, respectively.

We can define the concatenation of strings in $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho}^*$ as follows.

The concatenation of two pairs $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ in $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho}^*$ is defined by $\begin{bmatrix} x_1 y_1 \\ x_2 y_2 \end{bmatrix}$ in $\left[\begin{array}{c} \Sigma \\ \Sigma \end{array} \right]_{\rho}^*$.

Manuscript received November 28, 2005.

Manuscript revised April 7, 2006.

[†]The author is with the Faculty of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu-shi, 965-8580 Japan.

^{††}The author is with the Faculty of Engineering, University of Toyama, Toyama-shi, 930-8555 Japan.

a) E-mail: okawa@u-aizu.ac.jp

b) E-mail: hirose@iis.toyama-u.ac.jp

DOI: 10.1093/ietisy/e89-d.10.2591

Since $\begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$ is not a DNA molecule as stated in [2], we consider for DNA molecules the set $WK_\rho(\Sigma) = \left[\begin{smallmatrix} \Sigma \\ \Sigma \end{smallmatrix} \right]_\rho^+$, which is called a Watson-Crick domain associated to the alphabet Σ and the complementarity relation ρ .

We call such an element $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ in $WK_\rho(\Sigma)$ well-formed double stranded sequence. The two component strings, w_1 and w_2 , are also called strands; w_1 is the upper strand and w_2 is the lower strand.

To describe pairs of strings such as fragments of DNA strands, we write the elements $(x_1, x_2) \in \Sigma^* \times \Sigma^*$ in the form $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, in accordance with the representation of DNA molecules. We also write $\begin{pmatrix} \Sigma^* \\ \Sigma^* \end{pmatrix}$ instead of $\Sigma^* \times \Sigma^*$. The concatenation of two pairs $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ is $\begin{pmatrix} x_1 y_1 \\ x_2 y_2 \end{pmatrix}$.

Note the essential difference between $\begin{pmatrix} x \\ y \end{pmatrix}$ and $\begin{bmatrix} x \\ y \end{bmatrix}$: While $\begin{pmatrix} x \\ y \end{pmatrix}$ is just another form for (x, y) , that is, no relation is assumed between the symbols appearing in x and y , $\begin{bmatrix} x \\ y \end{bmatrix}$ represents a molecule, that is, the two strands x and y are of the same length, with a precise bonding denoted by ρ between corresponding symbols in the two strands.

2.2 Watson-Crick Finite Automata

We will define a Watson-Crick finite automaton as an automaton whose input is any element in the Watson-Crick domain. We adopt a different notation in the definition of a Watson-Crick finite automaton from ones in [1]–[4], since it resembles the ordinary notation of a finite automaton.

A Watson-Crick finite automaton is a construct

$$M = (Q, \Sigma, \rho, \delta, q_0, F),$$

where Q is a set of states, Σ is an alphabet, $\rho \subseteq \Sigma \times \Sigma$ is a complementarity relation, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \begin{pmatrix} \Sigma^* \\ \Sigma^* \end{pmatrix} \rightarrow \mathcal{P}(Q)^\dagger$ is a transition function such that $\delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right) \neq \emptyset$ only for finitely many triples $(q, x, y) \in Q \times \Sigma^* \times \Sigma^*$.

The interpretation of $p \in \delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right)$ is that the automaton M in the state q , passes over x in the upper strand and y in the lower strand of a double stranded sequence, and enters the state p .

For $\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}$, $\begin{pmatrix} x \\ y \end{pmatrix}$, and $\begin{pmatrix} u_2 \\ v_2 \end{pmatrix}$ in $\begin{pmatrix} \Sigma^* \\ \Sigma^* \end{pmatrix}$ such that $\begin{bmatrix} u_1 x u_2 \\ v_1 y v_2 \end{bmatrix}$ in $WK_\rho(\Sigma)$, and q and p in Q with p in $\delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right)$, we write a

transition of M as follows:

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} q \begin{pmatrix} x u_2 \\ y v_2 \end{pmatrix} \Rightarrow \begin{pmatrix} u_1 x \\ v_1 y \end{pmatrix} p \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}.$$

We denote by $\Rightarrow^*$ the reflexive and transitive closure of the relation $\Rightarrow$. The language $L(M)$ recognized by M is defined as follows.

$$L(M) = \left\{ w_1 \left| q_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} q_f, \right. \right. \\ \left. \left. \text{for some } q_f \in F, \text{ and } \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(\Sigma) \right\}.$$

Note that an automaton M recognizes only the upper strand w_1 and that λ is not an element of $L(M)$ by the definition of $WK_\rho(\Sigma)$.

2.3 Some Variants of WK Families and Known Facts

We consider several variants of Watson-Crick finite automata. We say that $M = (Q, \Sigma, \rho, \delta, q_0, F)$ is:

- stateless, if $Q = F = \{q_0\}$.
- all-final, if $F = Q$.
- simple, if $x = \lambda$ or $y = \lambda$ for all $\delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right) \neq \emptyset$.
- 1-limited, if $|xy| = 1$ for all $\delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right) \neq \emptyset$.

For the stateless case, since the components Q , q_0 , and F can be omitted and the transition function can be defined by a set of pairs of strings $\begin{pmatrix} x \\ y \end{pmatrix}$ instead of the form

$\delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right)$, a stateless automaton can be written in the form $M = (\Sigma, \rho, \delta)$. For the all-final case, we can omit the set of final states, and denote them as $M = (Q, \Sigma, \rho, \delta, q_0)$.

Adopting the notation in [1], we denote by AWK , NWK , FWK , SWK , $1WK$, $NSWK$, $N1WK$, $FSWK$, and $F1WK$, the families of languages recognized by Watson-Crick finite automata which are arbitrary (A), stateless (N , from no state), all-final (F), simple (S), 1-limited (1), stateless and simple (NS), stateless and 1-limited ($N1$), all-final and simple (FS), and all-final and 1-limited ($F1$), respectively.

We will refer to all these language families and the families of FWK , $FSWK$, and $F1WK$ as WK families and FWK families, respectively.

Directly from the definitions, we obtain:

- $XWK \subseteq AWK$
for $X \in \{N, F, S, 1, NS, N1, FS, F1\}$.
- $1WK \subseteq SWK$.
- $NWK \subseteq FWK, NSWK \subseteq FSWK$,
and $N1WK \subseteq F1WK$.

[†]For a set A , the family of subsets of A is denoted by $\mathcal{P}(A)$.

- First of all, we fix an alphabet as Σ and assume $|\Sigma| \geq 2$ throughout the rest of the paper. For the complementarity

[†]Word reversal w^R for a word w , is defined by: $\lambda^R = \lambda$ and $(wa)^R = a(w^R)$ for $w \in \Sigma^*$ and $a \in \Sigma$.

$$w_{i+2}w_{i+2}^R f(\pi_{(i+1)})$$

$$w_{i+3}w_{i+3}^R f(\pi_{(i+2)})$$

...

$$w_{k-1}w_{k-1}^R f(\pi_{(k-2)})$$

$$w_k w_k^R f(\pi_{(k-1)})$$

Proof. The first expression is the definition itself, only using the $\pi_{(k-1)}$ notation.

The second expression is obtained from the first one by the repeated expansion of the left parts of $f(\pi_{(k-j)})$ for $j = 1, 2, \dots, k-i$ as follows.

$$f(\pi) = f(\pi_{(k-1)})w_k w_k^R f(\pi_{(k-1)})$$

$$= f(\pi_{(k-2)})w_{k-1}w_{k-1}^R f(\pi_{(k-2)})w_k w_k^R f(\pi_{(k-1)})$$

$$= f(\pi_{(k-3)})w_{k-2}w_{k-2}^R f(\pi_{(k-3)})w_{k-1}w_{k-1}^R$$

$$f(\pi_{(k-2)})w_k w_k^R f(\pi_{(k-1)})$$

⋮

$$= f(\pi_{(i)})w_{i+1}w_{i+1}^R f(\pi_{(i)})$$

$$w_{i+2}w_{i+2}^R f(\pi_{(i+1)})$$

$$w_{i+3}w_{i+3}^R f(\pi_{(i+2)})$$

...

$$w_{k-1}w_{k-1}^R f(\pi_{(k-2)})$$

$$w_k w_k^R f(\pi_{(k-1)}).$$

□

Moreover, since the expression of the second part of $f(\pi)$ is rather long compared to the first part $f(\pi_{(i)})$ as seen in the second expression in Lemma 1 when $f(\pi)$ is decomposed into two parts at the end of $f(\pi_{(i)})$, we employ s_i for describing the second part, that is,

$s_i = w_{i+1}w_{i+1}^R f(\pi_{(i)}) \cdots w_k w_k^R f(\pi_{(k-1)})$. Then, $f(\pi)$ can be expressed as $f(\pi) = f(\pi_{(i)})s_i$.

We define a language which plays a very important role later. Let L be a context-free language $\{ww^R \mid w \in \Sigma^+\}$. We give some lemmas related to L here. For a word w , let $\text{Pref}(w)$ be a set of proper prefixes of w , $\text{Pref}(w) = \{u \mid w = uv, u \neq \lambda\}$.

Lemma 2: There exist infinitely many words w such that $\text{Pref}(w) \cap L = \emptyset$.

Proof. For example, for any non-negative integer i , ab^i and for any non-negative integers i and j , $ab^{2i+1}a^j$ are such words. □

Lemma 3: For any integer $k > 0$, there exist infinitely many sets $W = \{w_1, w_2, \dots, w_{k-1}, w_k\}$ of size k such that for any pair of words w_i and w_j ($i \neq j$), they are different from each other and one is not a prefix of the other.

Proof. For example, choose $W = \{ab^{2i_1+1}a^{j_1}, ab^{2i_2+1}a^{j_2}, \dots, ab^{2i_k+1}a^{j_k}\}$ with $0 < i_1 < i_2 < \dots < i_k$ and $0 < j_1 < j_2 < \dots < j_k$.

$\dots < j_k$. Then it is clear that such a set W satisfies the condition, and that there are infinitely many sets W by choosing sequences for i and j with the properly increasing conditions. □

Lemma 4: For any list $\pi \in \Pi$ with $|\pi| = k > 0$, $f(\pi) \in L$.

Proof. It is clear by the definition of f . □

Lemma 5: For any list $\pi \in \Pi$ with $|\pi| = k > 0$, $|\text{Pref}(f(\pi)) \cap L| \geq k$. Moreover, for $k > 0$, we can find $\pi \in \Pi$ such that $|\text{Pref}(f(\pi)) \cap L| = k$.

Proof. First, by induction on the length of π , we prove the first sentence.

For $k = 1$, since $\pi = \langle w \rangle$, $f(\pi) = ww^R$, and $\text{Pref}(f(\pi)) \cap L = \text{Pref}(ww^R) \cap L \supseteq \{ww^R\}$, the first sentence holds.

Assume the proposition holds for $k - 1$. Let π be a list $\langle w_1, w_2, \dots, w_{k-1}, w_k \rangle$. Then, since $\text{Pref}(f(\pi)) \cap L \supseteq \text{Pref}(f(\pi_{(k-1)})) \cap L$ and $f(\pi) \in L - \text{Pref}(f(\pi_{(k-1)}))$, $|\text{Pref}(f(\pi)) \cap L| \geq |\text{Pref}(f(\pi_{(k-1)})) \cap L| + 1 \geq k - 1 + 1 = k$. This completes the proof of the first sentence.

For the second sentence, by Lemma 3, we can choose a list $\pi = \langle w_1, w_2, \dots, w_{k-1}, w_k \rangle$ satisfying that for every word w_i , $\text{Pref}(w_i) \cap L = \emptyset$, and that for any pair of words w_i and w_j ($i \neq j$), they are different from each other and one is not a prefix of the other. Then, since $\text{Pref}(f(\pi)) \cap L = \{f(\pi_{(i)}) \mid 1 \leq i \leq k\}$, $|\text{Pref}(f(\pi)) \cap L| = k$. This completes the proof of the second sentence. □

3. The Relations among WK Families

In this section, we investigate the relations among WK families. First we investigate the relations in FWK families, and then the relations among AWK and FWK families.

3.1 The Relations in the FWK Families

Theorem 1: $F1WK = FSWK$.

Proof. Since $F1WK \subseteq FSWK$, it is enough to show $FSWK \subseteq F1WK$. Let L be any language in $FSWK$ and accepted by an all-final and simple Watson-Crick automaton $M = (Q, \Sigma, \rho, \delta, q_0)$. Then, we construct an all-final and 1-limited Watson-Crick automaton $M' = (Q', \Sigma, \rho, \delta', q'_0)$ which accepts L . The state set Q' is defined as

$$Q' = \bigcup_{q \in Q} \left\{ q^u \mid \delta \left(q, \begin{pmatrix} x \\ \lambda \end{pmatrix} \right) \neq \emptyset, u \in \text{Pref}(x) \right\} \\ \bigcup_{q \in Q} \left\{ q_v \mid \delta \left(q, \begin{pmatrix} \lambda \\ y \end{pmatrix} \right) \neq \emptyset, v \in \text{Pref}(y) \right\} \cup Q.$$

Next, we define the transition function δ' as follows.

For $\delta \left(q, \begin{pmatrix} x \\ \lambda \end{pmatrix} \right) \neq \emptyset$ with $x = x_1 x_2 \cdots x_k$ for $k > 1$ and $x_i \in \Sigma$ for $1 \leq i \leq k$,

$$\delta' \left(q, \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \right) = \{q^{x_1}\},$$

$$\begin{aligned}\delta'\left(q^{x_1}, \begin{pmatrix} x_2 \\ \lambda \end{pmatrix}\right) &= \{q^{x_1 x_2}\}, \\ &\vdots \\ \delta'\left(q^{x_1 \cdots x_{k-2}}, \begin{pmatrix} x_{k-1} \\ \lambda \end{pmatrix}\right) &= \{q^{x_1 \cdots x_{k-1}}\}, \text{ and} \\ \delta'\left(q^{x_1 \cdots x_{k-1}}, \begin{pmatrix} x_k \\ \lambda \end{pmatrix}\right) &= \delta\left(q, \begin{pmatrix} x \\ \lambda \end{pmatrix}\right).\end{aligned}$$

For $\delta\left(q, \begin{pmatrix} \lambda \\ y \end{pmatrix}\right) \neq \emptyset$ with $y = y_1 y_2 \cdots y_k$ for $k > 1$ and $y_i \in \Sigma$ for $1 \leq i \leq k$,

$$\begin{aligned}\delta'\left(q, \begin{pmatrix} \lambda \\ y_1 \end{pmatrix}\right) &= \{q_{y_1}\}, \\ \delta'\left(q_{y_1}, \begin{pmatrix} \lambda \\ y_2 \end{pmatrix}\right) &= \{q_{y_1 y_2}\}, \\ &\vdots \\ \delta'\left(q_{y_1 \cdots y_{k-2}}, \begin{pmatrix} \lambda \\ y_{k-1} \end{pmatrix}\right) &= \{q_{y_1 \cdots y_{k-1}}\}, \text{ and} \\ \delta'\left(q_{y_1 \cdots y_{k-1}}, \begin{pmatrix} \lambda \\ y_k \end{pmatrix}\right) &= \delta\left(q, \begin{pmatrix} \lambda \\ y \end{pmatrix}\right).\end{aligned}$$

Then, any one step transition of M caused by $\delta\left(q, \begin{pmatrix} x \\ \lambda \end{pmatrix}\right)$ with $x = x_1 x_2 \cdots x_k$, for p in $\delta\left(q, \begin{pmatrix} x \\ \lambda \end{pmatrix}\right)$,

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} q \begin{pmatrix} x \\ \lambda \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \Rightarrow \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} p \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}$$

can be simulated by M' as follows.

$$\begin{aligned}\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} q \begin{pmatrix} x_1 x_2 \cdots x_k \\ \lambda \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \\ \Rightarrow \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} q^{x_1} \begin{pmatrix} x_2 \cdots x_k \\ \lambda \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \\ \Rightarrow \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} x_1 x_2 \\ \lambda \end{pmatrix} q^{x_1 x_2} \begin{pmatrix} x_3 \cdots x_k \\ \lambda \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \\ \Rightarrow^* \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} x_1 x_2 \cdots x_{k-1} \\ \lambda \end{pmatrix} q^{x_1 x_2 \cdots x_{k-1}} \begin{pmatrix} x_k \\ \lambda \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \\ \Rightarrow \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} x_1 x_2 \cdots x_k \\ \lambda \end{pmatrix} p \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}.\end{aligned}$$

The other case,

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} q \begin{pmatrix} \lambda \\ y \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} \Rightarrow \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} \lambda \\ y \end{pmatrix} p \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}$$

can be simulated by M' in the similar way. Therefore, $L(M) \subseteq L(M')$.

For another inclusion, the following observation holds: By the construction of the transition function δ' , if M' starts with q , then possible sequences of transitions of M' are those which spell out $x = x_1 x_2 \cdots x_k$ for an upper strand,

or $y = y_1 y_2 \cdots y_k$ for a lower strand, and M processes $\begin{pmatrix} x \\ \lambda \end{pmatrix}$ or $\begin{pmatrix} \lambda \\ y \end{pmatrix}$. Therefore, $L(M') \subseteq L(M)$.

Combining the two inclusions, we have $L(M) = L(M')$. $\square$

Theorem 2: $FSWK = FWK$.

Proof. Since $FSWK \subseteq FWK$, it is enough to show that $FWK \subseteq FSWK$. Let L be any language in FWK and accepted by an all-final Watson-Crick finite automaton $M = (Q, \Sigma, \rho, \delta, q_0)$. Then, we construct an all-final and simple Watson-Crick finite automaton $M' = (Q', \Sigma, \rho, \delta', q'_0)$ which accepts L . The state set Q' is defined as

$$Q' = \left\{ q^u \mid \delta\left(q, \begin{pmatrix} u \\ v \end{pmatrix}\right) \neq \emptyset, u, v \neq \lambda \right\} \cup Q.$$

Next, we define the transition function δ' as follows.

$$\begin{aligned}\delta'\left(q, \begin{pmatrix} u \\ \lambda \end{pmatrix}\right) &= \{q^u\}, \text{ and} \\ \delta'\left(q^u, \begin{pmatrix} \lambda \\ v \end{pmatrix}\right) &= \delta\left(q, \begin{pmatrix} u \\ v \end{pmatrix}\right).\end{aligned}$$

For the other cases, $\delta\left(q, \begin{pmatrix} u \\ \lambda \end{pmatrix}\right)$ and $\delta\left(q, \begin{pmatrix} \lambda \\ v \end{pmatrix}\right)$,

$$\begin{aligned}\delta'\left(q, \begin{pmatrix} u \\ \lambda \end{pmatrix}\right) &= \delta\left(q, \begin{pmatrix} u \\ \lambda \end{pmatrix}\right), \text{ and} \\ \delta'\left(q, \begin{pmatrix} \lambda \\ v \end{pmatrix}\right) &= \delta\left(q, \begin{pmatrix} \lambda \\ v \end{pmatrix}\right).\end{aligned}$$

Then, any one step transition of M can be simulated by M' for $L(M) \subseteq L(M')$ and $L(M') \subseteq L(M)$ can be deduced by the similar observation in the proof of Theorem 1. This completes the proof. $\square$

Combining the fact $NWK \subseteq FWK$ with Theorems 1 and 2, we obtain:

Corollary 1: $NWK \subseteq F1WK = FSWK = FWK$.

3.2 The Relation between AWK and FWK Families

Here, we consider the relation between FWK and AWK . By their definitions, it is clear that $FWK \subseteq AWK$ holds. Therefore, we would like to know whether this inclusion is proper or $FWK = AWK$ holds.

For any Watson-Crick finite automaton $M = (Q, \Sigma, \rho, \delta, q_0, F)$, we define two sets, for $q \in Q$

$$\Delta(q) = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \mid \delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right) \neq \emptyset \right\} \text{ and}$$

$$\Delta_l^{-1}(q) = \left\{ y \mid \exists p \in Q, \exists x \in \Sigma^*, q \in \delta\left(p, \begin{pmatrix} x \\ y \end{pmatrix}\right) \right\},$$

Here, $\Delta(q)$ is a set of word pairs $\begin{pmatrix} x \\ y \end{pmatrix}$ which can be processed in one step by M in the state q . on the other hand, $\Delta_l^{-1}(q)$ is a set of lower words y of word pairs $\begin{pmatrix} x \\ y \end{pmatrix}$ such that M can change its state to q in one step by processing such word pairs, that is, the superscript -1 and the subscript l mean one-step before and lower words of word pairs, respectively.

Furthermore, we introduce a notation $S_v = \{q_v | q \in S\}$ for any subset S of Q and $v \in \Sigma^*$, which is used in the proof of Theorem 3 to define the transition function δ' . For example, if $\delta\left(q, \begin{pmatrix} x \\ y \end{pmatrix}\right) = \{q_1, q_2, \dots, q_k\}$, then we denote $\delta'\left(s_0, \begin{pmatrix} x \\ \lambda \end{pmatrix}\right) = \delta\left(q_0, \begin{pmatrix} x \\ y \end{pmatrix}\right)_y$, instead of the notation $\delta'\left(q', \begin{pmatrix} x \\ \lambda \end{pmatrix}\right) = \{q_{1y}, q_{2y}, \dots, q_{ky}\}$.

Theorem 3: $FWK = AWK$.

Proof. By definition, it is enough to show that for any language L in AWK , we can find an all-final Watson-Crick finite automaton which accepts L .

Let L be a language in AWK accepted by a Watson-Crick finite automaton $M = (Q, \Sigma, \rho, \delta, q_0, F)$. We construct an all-final Watson-Crick finite automaton $M' = (Q', \Sigma, \rho, \delta', s_0)$ from M .

For the state set Q' , we define

$$Q' = \{s_0, s_f\} \cup \bigcup_{q \in Q} \{q_y | y \in \Delta_l^{-1}(q)\}, \text{ where } s_0, s_f \notin Q.$$

Next, we define the transition function δ' as follows.

For $\begin{pmatrix} x \\ y \end{pmatrix} \in \Delta(q_0)$,

$$\delta'\left(s_0, \begin{pmatrix} x \\ \lambda \end{pmatrix}\right) = \delta\left(q_0, \begin{pmatrix} x \\ y \end{pmatrix}\right)_y.$$

For $q \in Q, y \in \Sigma^*$, and $\begin{pmatrix} x' \\ y' \end{pmatrix} \in \Delta(q)$,

$$\delta'\left(q_y, \begin{pmatrix} x' \\ y \end{pmatrix}\right) = \delta\left(q, \begin{pmatrix} x' \\ y' \end{pmatrix}\right)_y.$$

For $q \in F$ and $y \in \Sigma^*$,

$$\delta'\left(q_y, \begin{pmatrix} \lambda \\ y \end{pmatrix}\right) = \{s_f\}.$$

Now, we show $L(M) = L(M')$. Let w be a word in $L(M)$. Assume that M processes $\begin{bmatrix} w \\ \rho(w) \end{bmatrix}$ as follows:

$$\begin{aligned} q_0 \begin{bmatrix} w \\ \rho(w) \end{bmatrix} &\Rightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} p^{(1)} \begin{pmatrix} w_1 \\ w'_1 \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \dots \begin{pmatrix} x_i \\ y_i \end{pmatrix} p^{(i)} \begin{pmatrix} w_i \\ w'_i \end{pmatrix} \end{aligned}$$

$$\Rightarrow^* \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \dots \begin{pmatrix} x_k \\ y_k \end{pmatrix} p^{(k)} = \begin{bmatrix} w \\ \rho(w) \end{bmatrix} p^{(k)},$$

and $p^{(k)} \in F$.

Then, M' can process this word as follows:

$$\begin{aligned} s_0 \begin{bmatrix} w \\ \rho(w) \end{bmatrix} &\Rightarrow \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} p_{y_1}^{(1)} \begin{pmatrix} w_1 \\ y_1 w'_1 \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \begin{pmatrix} x_2 \\ y_1 \end{pmatrix} \dots \begin{pmatrix} x_i \\ y_{i-1} \end{pmatrix} p_{y_i}^{(i)} \begin{pmatrix} w_i \\ y_i w'_i \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \begin{pmatrix} x_2 \\ y_1 \end{pmatrix} \dots \begin{pmatrix} x_k \\ y_{k-1} \end{pmatrix} p_{y_k}^{(k)} \begin{pmatrix} \lambda \\ y_k \end{pmatrix} \\ &\Rightarrow \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \begin{pmatrix} x_2 \\ y_1 \end{pmatrix} \dots \begin{pmatrix} x_k \\ y_{k-1} \end{pmatrix} \begin{pmatrix} \lambda \\ y_k \end{pmatrix} s_f \\ &= \begin{bmatrix} w \\ \rho(w) \end{bmatrix} s_f. \end{aligned}$$

Therefore, $L(M) \subseteq L(M')$.

For another inclusion, assume that w does not belong to $L(M)$. Then, there are two possibilities for M not to accept $\begin{bmatrix} w \\ \rho(w) \end{bmatrix}$. That is, (1) M processes it completely and enters in $q \notin F$, or (2) M stops during the process.

For the former case (1), since the process $\begin{bmatrix} w \\ \rho(w) \end{bmatrix}$ by M is:

$$\begin{aligned} q_0 \begin{bmatrix} w \\ \rho(w) \end{bmatrix} &\Rightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} p^{(1)} \begin{pmatrix} w_1 \\ w'_1 \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \dots \begin{pmatrix} x_i \\ y_i \end{pmatrix} p^{(i)} \begin{pmatrix} w_i \\ w'_i \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \dots \begin{pmatrix} x_k \\ y_k \end{pmatrix} p^{(k)} \\ &= \begin{bmatrix} w \\ \rho(w) \end{bmatrix} p^{(k)} \end{aligned}$$

with $p^{(k)} \notin F$, the process by M' is:

$$\begin{aligned} s_0 \begin{bmatrix} w \\ \rho(w) \end{bmatrix} &\Rightarrow \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} p_{y_1}^{(1)} \begin{pmatrix} w_1 \\ y_1 w'_1 \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \begin{pmatrix} x_2 \\ y_1 \end{pmatrix} \dots \begin{pmatrix} x_i \\ y_{i-1} \end{pmatrix} p_{y_i}^{(i)} \begin{pmatrix} w_i \\ y_i w'_i \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \begin{pmatrix} x_2 \\ y_1 \end{pmatrix} \dots \begin{pmatrix} x_k \\ y_{k-1} \end{pmatrix} p_{y_k}^{(k)} \begin{pmatrix} \lambda \\ y_k \end{pmatrix}, \end{aligned}$$

and M' stops, because there is no transition from $p_{y_k}^{(k)}$ with $\begin{pmatrix} \lambda \\ y_k \end{pmatrix}$ by the definition of δ' since $p^{(k)} \notin F$.

Hence, w is not a word in $L(M')$.

For the latter case (2), the sequence of transitions for M with $\begin{bmatrix} w \\ \rho(w) \end{bmatrix}$ is assumed as

$$q_0 \begin{bmatrix} w \\ \rho(w) \end{bmatrix} \Rightarrow \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} p^{(1)} \begin{pmatrix} w_1 \\ w'_1 \end{pmatrix}$$

$$\Rightarrow^* \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \cdots \begin{pmatrix} x_i \\ y_i \end{pmatrix} p^{(i)} \begin{pmatrix} w_i \\ w'_i \end{pmatrix}$$

and M stops, which means that $\delta \left(p^{(i)}, \begin{pmatrix} x \\ y \end{pmatrix} \right) = \emptyset$ for any prefix strand $\begin{pmatrix} x \\ y \end{pmatrix}$ of $\begin{pmatrix} w \\ w'_i \end{pmatrix}$. Then, M' processes and stops as follows.

$$\begin{aligned} s_0 \left[\begin{pmatrix} w \\ \rho(w) \end{pmatrix} \right] &\Rightarrow \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} p_{y_1}^{(1)} \begin{pmatrix} w_1 \\ y_1 w'_1 \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} x_1 \\ \lambda \end{pmatrix} \begin{pmatrix} x_2 \\ y_1 \end{pmatrix} \cdots \begin{pmatrix} x_i \\ y_{i-1} \end{pmatrix} p_{y_i}^{(i)} \begin{pmatrix} w_i \\ y_i w'_i \end{pmatrix}. \end{aligned}$$

Hence, w is not a word in $L(M')$. Therefore, $L(M') \subseteq L(M)$.

Combining these two inclusions, we have $L(M) = L(M')$. This completes the proof. $\square$

Combining previous theorems and a proposition, we obtain the following result as a corollary to this theorem.

Corollary 2: $F1WK = FSWK = FWK = AWK$.

4. The Relations between WK Families and CF

In this section, we investigate the relations between WK families and CF. By the definitions of language families, any language in WK never contains λ , while some languages in CF can contain it. This leads to a trivial conclusion that the language families WK and CF are incomparable, since a context-free language which contains an empty word is not accepted by any Watson-Crick finite automaton and some context-sensitive language such as $L = \{a^n b^n c^n \mid n \geq 1\}$ can be accepted by some Watson-Crick finite automaton. In order to make our investigation more meaningful, we need some consideration for the treatment of an empty word.

Recalling that we encountered a similar difficulty and resolved it by ignoring the difference whether or not L contains λ when comparing CF and CS, we adopt the similar resolution, that is, we consider only languages obtained by $L' = L - \{\lambda\}$ for $L \in CF$ in this section.

First we show that there exists a non-context-free language which belongs to NWK. Then, we show that there exists a context-free language which does not belong to AWK.

4.1 The Incomparability of NWK and CF

Theorem 4: $NWK - CF \neq \emptyset$.

Proof. Since a language $L = \{a^n b^n c^n, c^n b^n a^n \mid n \geq 1\}^+$ is not a context-free language, it is enough to construct a stateless Watson-Crick finite automaton M which recognizes L .

Let $M = (\Sigma, \rho, \delta)$ be a stateless Watson-Crick finite automaton, where $\Sigma = \{a, b, c\}$, $\rho = \{(a, a), (b, b), (c, c)\}$, and

$$\delta = \left\{ \begin{pmatrix} a \\ \lambda \end{pmatrix}, \begin{pmatrix} b \\ a \end{pmatrix}, \begin{pmatrix} c \\ b \end{pmatrix}, \begin{pmatrix} \lambda \\ c \end{pmatrix} \right\}.$$

Then, since a sequence $\begin{bmatrix} a^n b^n c^n \\ a^n b^n c^n \end{bmatrix}$ can be decomposed into $\begin{pmatrix} a \\ \lambda \end{pmatrix}^n, \begin{pmatrix} b \\ a \end{pmatrix}^n, \begin{pmatrix} c \\ b \end{pmatrix}^n$ and $\begin{pmatrix} \lambda \\ c \end{pmatrix}^n$, it is clear that the sequence can be successfully processed by M . Similarly, a sequence $\begin{bmatrix} c^n b^n a^n \\ c^n b^n a^n \end{bmatrix}$ can also be successfully processed by M . Therefore, it is clear that a sequence consisting of concatenations of such sequences is successfully processed by M . Hence, $L \subseteq L(M)$.

Next, we will show the converse inclusion $L(M) \subseteq L$. Assuming the contrary, that is, $L(M) \not\subseteq L$, there exists a word w which is accepted by M but is not in L . Moreover, w is assumed to be the shortest word among such words.

Assume w is decomposed into uv with $u \in L$ and $u, v \neq \lambda$. If v is in L , then w , the concatenation of u and v , is an element of L , leading to the contradiction. Therefore, v is not in L . Since $u \in L$, from the fact proved above, u is accepted by M , and as a result,

$$q_0 \begin{bmatrix} u \\ u \end{bmatrix} \Rightarrow^* \begin{bmatrix} u \\ u \end{bmatrix} q_0.$$

By the assumption that w is accepted by M ,

$$q_0 \begin{bmatrix} w \\ w \end{bmatrix} \Rightarrow^* \begin{bmatrix} w \\ w \end{bmatrix} q_0.$$

Therefore,

$$q_0 \begin{bmatrix} w \\ w \end{bmatrix} \Rightarrow^* \begin{bmatrix} u \\ u \end{bmatrix} q_0 \begin{bmatrix} v \\ v \end{bmatrix} \Rightarrow^* \begin{bmatrix} u \\ u \end{bmatrix} \begin{bmatrix} v \\ v \end{bmatrix} q_0.$$

Hence,

$$q_0 \begin{bmatrix} v \\ v \end{bmatrix} \Rightarrow^* \begin{bmatrix} v \\ v \end{bmatrix} q_0.$$

This indicates that v is accepted by M . However, v is not an element of L and the length of v is shorter than one of w , which contradicts the assumption that w is the shortest among such words. This means that any proper prefix of w is not in L .

By the definition of M , M cannot process any word starting with b . We assume w starts with a . Furthermore, if $w = a^i c v (i > 0)$, then

$$q_0 \begin{bmatrix} w \\ w \end{bmatrix} = q_0 \begin{bmatrix} a^i c v \\ a^i c v \end{bmatrix} \Rightarrow^* \begin{pmatrix} a \\ \lambda \end{pmatrix}^i q_0 \begin{pmatrix} c v \\ a^i c v \end{pmatrix},$$

and M can not move further. Therefore, w should be of the form $a^i b^j v (v \neq b v')$. If $i > j$, then

$$\begin{aligned} q_0 \begin{bmatrix} w \\ w \end{bmatrix} &= q_0 \begin{bmatrix} a^i b^j v \\ a^i b^j v \end{bmatrix} \Rightarrow^* \begin{pmatrix} a \\ \lambda \end{pmatrix}^i q_0 \begin{pmatrix} b^j v \\ a^i b^j v \end{pmatrix} \\ &\Rightarrow^* \begin{pmatrix} a \\ \lambda \end{pmatrix}^i \begin{pmatrix} b \\ a \end{pmatrix}^j q_0 \begin{pmatrix} v \\ a^{i-j} b^j v \end{pmatrix}, \end{aligned}$$

and M can not move further. For the case $i < j$, by the similar argument, M can not reach the right end of $\begin{bmatrix} w \\ w \end{bmatrix}$.

Therefore, $i = j$ and $w = a^i b^i v$.

Repeating the similar argument, we can conclude that $w = a^i b^i c^i v$. Since $a^i b^i c^i$ is in L , it leads to the contradiction to the fact that w is not decomposed into uv with $u \in L$.

For the case where w starts with c , the same argument leads to the contradiction. This completes the proof. $\square$

We obtain a new relation as a corollary to Theorem 4 and the fact in [4] that *REG* and *NWK* are incomparable, leading to the fact that *REG* – *NWK* $\neq \emptyset$ as a consequence.

Corollary 3: *NWK* and *CF* are incomparable.

4.2 The Incomparability of *AWK* and *CF*

Here, we investigate the relation between *AWK* and *CF*. Since it is known that there is a non-context-free language accepted by some Watson-Crick finite automaton, there are two possibilities: (1) *AWK* contains *CF* properly, or (2) *AWK* and *CF* are incomparable. We show that there exists a context-free language which can not be accepted by any Watson-Crick finite automaton, and that the second case holds as a result.

Theorem 5: Let L be a context-free language $\{ww^R \mid w \in \Sigma^+\}$ over Σ with $|\Sigma| \geq 2$. Then, L is not accepted by any Watson-Crick finite automaton.

Proof. Assume the contrary that $L = L(M)$ holds for some Watson-Crick finite automaton $M = (Q, \Sigma, \rho, \delta, q_0, F)$. Without loss of generality, we can assume M is a simple Watson-Crick finite automaton with $|F| = k > 0$. Let π be a list of length $k + 2$, that is, $\pi = \langle w_1, w_2, \dots, w_k, w_{k+1}, w_{k+2} \rangle$. Furthermore, by Lemmata 2 and 3, we can assume that for every word w_i , $\text{Pref}(w_i) \cap L = \emptyset$, and that for any pair of words w_i and w_j ($i \neq j$), they are different from each other and one is not a prefix of the other.

Throughout the proof, we fix the list π which satisfies conditions stated above. Then, since $f(\pi)$ is in L by Lemma 4, it is accepted by M . By Lemma 5, the number of words in $\text{Pref}(f(\pi)) \cap L$ is $k + 2$, and every word in $\text{Pref}(f(\pi)) \cap L$ is of the form $f(\pi_{(i)})$ and accepted by M . Therefore, we can decompose $f(\pi)$ at the point M enters a final state during M processes $f(\pi)$. We have $k + 2$ decompositions of $f(\pi)$: For all i ($0 \leq i \leq k + 2$), $f(\pi) = f(\pi_{(i)})s_i$.

Now we consider the process when M recognizes $f(\pi)$. Since every word in $\text{Pref}(f(\pi)) \cap L$ should be recognized by M , every time M recognizes it, M enters a final state. So the process is as follows:

$$\begin{aligned} q_0 \left[\begin{array}{c} f(\pi) \\ \rho(f(\pi)) \end{array} \right] &\Rightarrow^* \left[\begin{array}{c} f(\pi_{(1)}) \\ \rho(f(\pi_{(1)})) \end{array} \right] p^{(1)} \left[\begin{array}{c} s_1 \\ \rho(s_1) \end{array} \right] \\ &\Rightarrow^* \left[\begin{array}{c} f(\pi_{(i)}) \\ \rho(f(\pi_{(i)})) \end{array} \right] p^{(i)} \left[\begin{array}{c} s_i \\ \rho(s_i) \end{array} \right] \\ &\Rightarrow^* \left[\begin{array}{c} f(\pi_{(j)}) \\ \rho(f(\pi_{(j)})) \end{array} \right] p^{(j)} \left[\begin{array}{c} s_j \\ \rho(s_j) \end{array} \right] \\ &\Rightarrow^* \left[\begin{array}{c} f(\pi_{(k+1)}) \\ \rho(f(\pi_{(k+1)})) \end{array} \right] p^{(k+1)} \left[\begin{array}{c} s_{k+1} \\ \rho(s_{k+1}) \end{array} \right] \end{aligned}$$

$$\Rightarrow^* \left[\begin{array}{c} f(\pi_{(k+2)}) \\ \rho(f(\pi_{(k+2)})) \end{array} \right] p^{(k+2)}$$

where $p^{(i)} \in F$ for all i ($1 \leq i \leq k + 2$).

By assumption $|F| = k$, there exists a state such that it appears at least twice in the first $k + 1$ parts of this sequence. Assume $p^{(i)}$ and $p^{(j)}$ are the same final state and $1 \leq i < j \leq k + 1$.

Here, we consider a word $z = f(\pi_{(i)})s_j$. Then, M will process a sequence $\left[\begin{array}{c} z \\ \rho(z) \end{array} \right]$ as follows.

$$\begin{aligned} q_0 \left[\begin{array}{c} z \\ \rho(z) \end{array} \right] &= q_0 \left[\begin{array}{c} f(\pi_{(i)})s_j \\ \rho(f(\pi_{(i)})s_j) \end{array} \right] \\ &\Rightarrow^* \left[\begin{array}{c} f(\pi_{(i)}) \\ \rho(f(\pi_{(i)})) \end{array} \right] p^{(i)} \left[\begin{array}{c} s_j \\ \rho(s_j) \end{array} \right]. \end{aligned}$$

Then, recalling $p^{(i)} = p^{(j)}$ and referring the latter part of the process that M accepts $f(\pi)$, we can recognize that M continues the process as follows.

$$\begin{aligned} &\left[\begin{array}{c} f(\pi_{(i)}) \\ \rho(f(\pi_{(i)})) \end{array} \right] p^{(i)} \left[\begin{array}{c} s_j \\ \rho(s_j) \end{array} \right] \\ &= \left[\begin{array}{c} f(\pi_{(i)}) \\ \rho(f(\pi_{(i)})) \end{array} \right] p^{(j)} \left[\begin{array}{c} s_j \\ \rho(s_j) \end{array} \right] \\ &\Rightarrow^* \left[\begin{array}{c} f(\pi_{(i)}) \\ \rho(f(\pi_{(i)})) \end{array} \right] \left[\begin{array}{c} s_j \\ \rho(s_j) \end{array} \right] p^{(k+2)}. \end{aligned}$$

Hence, M accepts z which does not belong to L , which leads to the contradiction. This completes the proof[†]. $\square$

As a corollary to this theorem, we obtain the following result.

Corollary 4: *AWK* and *CF* are incomparable.

5. Concluding Remarks

In this paper, aiming to establish the relations among language families defined by Watson-Crick finite automata, we obtain the following results.

1. $F1WK = FSWK = FWK$.
2. $FWK = AWK$.
3. There exists a language which is not context-free but belongs to *NWK*.
4. There exists a context-free language which does not belong to *AWK*.

The second result means that the power of recognizing languages is not influenced by whether or not the sets of final states are specified. The first and second results

[†]Note that the language L employed here is a linear context-free language. As our intention in this paper is to establish the relation among *WK* families and language families in the Chomsky hierarchy, we only point out this fact and do not restate this theorem in the stronger form.

Fig. 2 The refined relations among WK and Chomsky families.

mean that in language recognition, restrictions on how automata read strands at once, such as from one strand or both strands, one character, or a string, does not cause any difference in the recognition ability, if they have enough states, although stateless automata are strictly less powerful than Watson-Crick finite automata having states. The third result is implicitly known since it can be shown as a corollary to Corollary 5.7 in [1] which is stated that:

For each family of languages FL such that $FL \subseteq RE$ and FL is closed under intersection with regular languages and arbitrary morphisms, we have $XWK - FL \neq \emptyset$, $X \in \{A, S, F, 1, N, FS\}$.

However, it is worthwhile to state this result since we give a language $L = \{a^n b^n c^n, c^n b^n a^n | n \geq 1\}^+$ as an explicit example which belongs to NWK but does not belong to CF . With the known fact that there exists a context-sensitive language in AWK , the fourth result shows us that AWK is incomparable with CF .

We established the relations among WK families and language families in a Chomsky hierarchy. As a summary of this paper, we illustrate the refined relations among them in Fig. 2. In this figure, we added a result but not stated anywhere within our survey, $N1WK \subsetneq NSWK^\dagger$. As a result, we could establish the relations among WK families completely.

Acknowledgement

Mr. Masashi Furuya, who graduated from the University of

[†]It is easily proved that a language $L = \{(ab)^n | n \geq 0\}$, for example, can be accepted by a stateless simple Watson-Crick automaton. However, it is not accepted by any stateless 1-limited Watson-Crick automaton, since if such an automaton accepts L , then it should contain $\begin{pmatrix} a \\ \lambda \end{pmatrix}$, $\begin{pmatrix} b \\ \lambda \end{pmatrix}$, $\begin{pmatrix} \lambda \\ a \end{pmatrix}$, and $\begin{pmatrix} \lambda \\ b \end{pmatrix}$ in its transition function δ and should accept words not in L such as a .

Aizu in March 2005, greatly contributed to this research through his graduation thesis, as he discovered a language and constructed a WK automaton which accepts it. Both his language and the automaton are used in Theorem 4. Mr. Kazuya Yamashita, a student of the master's degree program at University of Toyama, pointed out the fact $N1WK \subsetneq NSWK$ and gave an example stated in the footnote.

The authors are grateful to anonymous referees who kindly pointed out many defects in the first version of this paper for its improvement.

References

- [1] R. Freund, Gh. Păun, G. Rozenberg, and A. Salomaa, DNA Computing: New Computing Paradigms, Springer, 1998.
- [2] C. Martin-Vide, Gh. Păun, G. Rozenberg, and A. Salomaa, "Universality results for finite H systems and for Watson-Crick finite automata," in Computing with Bio-Molecules, Theory and Experiments, ed. Gh. Păun, pp.200–220, Springer, 1998.
- [3] R. Freund, Gh. Păun, G. Rozenberg, and A. Salomaa, "Watson-Crick automata," DIMACS Series in Discrete Mathematics and Theoretical Computer Science, ed. H. Rubin and D.H. Wood, vol.48, pp.297–327, 1999.
- [4] S. Hirose, K. Tsuda, and Y. Ogoshi, "Some relations between Watson-Crick finite automata and Chomsky hierarchy," IEICE Trans. Inf. & Syst., vol.E87-D, no.5, pp.1261–1264, May 2004.

Satoshi Okawa received the B.E., the M.E., and the D.E. degrees from Tohoku University in Electrical and Communication Engineering in 1972, 1975, and 1979, respectively. In 1980, he joined Iwate University, then moved to Hachinohe Institute of Technology in 1982 and became a professor there. In 1993, he joined the University of Aizu as a professor. His research interests include automata and language theory, graph and network theory and algorithms, computational complexity, and parallel computation.

He is a member of Information Processing Society of Japan, Japan Society of Software Science and Technology, the Japan Society of Symbolic and Algebraic Computation, The Association for Natural Language Processing, and others.

Sadaki Hirose received the B.E. degree in Electronics from Toyama University in 1974, the M.E. and the D.E. degrees in Information Engineering from Tohoku University in 1976 and 1980, respectively. From 1980 to 1984, he was with Fujitsu Laboratories, Ltd., and from 1984 to 1989, he was an associate professor at Kanagawa University. In 1989, he joined Toyama University where he is presently a Professor of Engineering. His research interests include formal language theory, automata theory,

algorithm and complexity theory. He is a member of the Information Processing Society of Japan.