A Representation Theorem for languages accepted by Watson-Crick finite automata*

José M. Sempere
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
jsempere@dsic.upv.es

Abstract

Watson-Crick finite automata were first proposed in [1] inspired by formal language theory, finite states machines and some ingredients from DNA computing such as working with molecules as double stranded complementary strings. Here, we will give a representation theorem for the languages accepted by those machines in any arbitrary form. We will establish that any language (in the upper strand or complete molecule versions) can be expressed as the intersection of a linear language with an even linear one (together with homomorphisms and other operations if upper strand language is defined).

1 Preliminaries and basic concepts

Watson-Crick (WK) finite automata [1] is a good example of how DNA biological properties can be adapted to propose computation models in the framework of DNA computing. The WK automata model works with double strings inspired by double-stranded molecules with a complementary relation between symbols (here, inspired by classical complementary relation between nucleotides A-T and C-G).

First, we will introduce some basic concepts from formal language theory according to [2, 4] and from DNA computing according to [3].

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 will be denoted by Σ^*. Given a string $x \in \Sigma^*$ we will

*Work partially supported by the Ministerio de Ciencia y Tecnología under project TIC2003-09319-C03-02

denote its length by $|x|$. The empty string will be denoted by λ and Σ^+ will denote $\Sigma^* - \{\lambda\}$. Given a string x we will denote by x^r the reversal string of x. A language L defined over Σ is a set of strings from 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 $\alpha \to \beta$. The language of the grammar is denoted by $L(G)$ and 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 will denote by $\overset{*}{\underset{G}{\Rightarrow}}$ the reflexive and transitive closure of $\underset{G}{\Rightarrow}$.

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

$$\mathcal{REG} \subset \mathcal{ELIN} \subset \mathcal{LIN}$$

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 $x \in \Sigma^*$. Finally, 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 will use the symmetric (and injective) relation of complementarity $\rho \subseteq \Sigma \times \Sigma$. For any string $x \in \Sigma^*$, we will 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 Σ, a *sticker* over Σ will be the pair (x, y) such that $x = x_1 v x_2$, $y = y_1 w y_2$ with $x, y \in \Sigma^*$ and $\rho(v) = w$. The sticker (x, y) will be denoted by $\binom{x}{y}$. A sticker $\binom{x}{y}$ will be a complete and complementary *molecule* if $|x| = |y|$ and $\rho(x) = y$. A complementary and complete molecule $\binom{x}{y}$ will be denoted as $\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$. Obviously, any sticker $\binom{x}{y}$ or molecule $\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$ can be represented by $x\#y^r$ where $\# \notin \Sigma$. Here, we will use $x\#y^r$ instead of $x\#y$ due to the grammar construction that

we will propose in the next section. Furthermore, inspired by DNA structure $x\#y^r$ represents the upper and lower nucleotide strings within the same direction $3' - 5'$ (or $5' - 3'$).

Formally, an *arbitrary* WK finite automata is defined by the tuple $M = (V, \rho, Q, s_0, F, \delta)$, where Q and V are disjoint alphabets (states and symbols), s_0 is the initial state, $F \subseteq Q$ is a set of final states and $\delta : Q \times \binom{V^*}{V^*} \to \mathcal{P}(Q)$ (which denotes the power set of Q, that is the set of all possible subsets of Q).

The language of complete and complementary molecules accepted by M will be denoted by the set $L_m(M)$, while the upper strand language accepted by M will be denoted by $L_u(M)$ and defined as the set of strings x such that M, after analyzing the molecule $\begin{bmatrix} x \\ y \end{bmatrix}$ enters into a final state.

2 A Representation Theorem

Now, we give the basic representation theorem.

Theorem 2.1. *Let $M = (V, \rho, Q, s_0, F, \delta)$ be an arbitrary WK finite automata. Then there exists a linear language L_1 and an even linear language L_2 such that $L_m(M) = L_1 \cap L_2$.*

Proof. We will provide an algorithm to construct a linear grammar G_1 such that $L(G_1) = L_1$ and an even linear grammar G_2 such that $L(G_2) = L_2$.

First, the grammar $G_1 = (N, V, P, s_0)$ where $N = Q$, s_0 is the axiom of the grammar and P is defined as follows

- If $q \in F$ then $q \to \# \in P$
- If $p \in \delta(q, \binom{x_1}{x_2})$ then $q \to x_1\ p\ x_2^r \in P$.

Now, we will prove that if M analyzes $\binom{x}{y}$ and enters into a state s then $s_0 \overset{*}{\underset{G_1}{\Rightarrow}} x\ s\ y^r$. The proof can be performed through an induction process over the number of transitions that M carries out.

Induction Basis. If M only applies one transition movement such that $s \in \delta(s_0, \binom{x}{y})$ then $s_0 \underset{G_1}{\Rightarrow} x\ s\ y^r$ as a consequence of the construction that we have proposed for G_1.

Induction Hypothesis. Let us suppose that if M applies up to n transition movements for analyzing $\binom{x}{y}$ and arrives to state s then $s_0 \overset{*}{\underset{G_1}{\Rightarrow}} x\ s\ y^r$

Induction Step. Now, M applies $n+1$ transition movements for analyzing $\binom{x}{y}$ and enters into state s. We will suppose that $\binom{x}{y} = \binom{x_1x_2}{y_1y_2}$, M enters into state s' after analyzing $\binom{x_1}{y_1}$ during the first n steps and, finally, $s \in \delta(s', \binom{x_2}{y_2})$. Then, by induction hypothesis, $s_0 \overset{*}{\underset{G_1}{\Rightarrow}} x_1\ s'\ y_1^r$ and, by construction, $s' \rightarrow x_2\ s\ y_2^r$, so $s_0 \overset{*}{\underset{G_1}{\Rightarrow}} x_1x_2\ s\ y_2^r y_1^r = x\ s\ y^r$.

We can observe that if M enters into a final state s, after analyzing $\binom{x}{y}$, then $x\#y^r \in L(G_1)$ (given that $s_0 \overset{*}{\underset{G_1}{\Rightarrow}} x\ s\ y_1^r \underset{G_1}{\Rightarrow} x\ \#\ y^r$).

The language L_2 is defined by the grammar $G_2 = (\{S\}, V, P, S)$ where P is defined as follows

- $S \rightarrow \# \in P$
- For every pair of symbols $a, b \in V$, such that $(a, b) \in \rho$, $S \rightarrow aSb \in P$

It can be easily proved that $L(G_2) = \{x_1\#x_2^r \in V^* : |x_1| = |x_2| \text{ and } \rho(x_1) = x_2\}$. That is, L_2 can be established as the set of complete and complementary molecules $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ in the form $x_1\#x_2^r$.

From L_1 and L_2 it is clear that $L_1 \cap L_2$ is the set of complete and complementary molecules accepted by M in the form $x\#y^r$ proposed in the previous section. □

In order to characterize the upper strand language we will provide the following result

Corollary 2.1. *Let $M = (V, \rho, Q, s_0, F, \delta)$ be an arbitrary WK finite automata. Then $L_u(M)$ can be expressed $g(h^{-1}(L_1 \cap L_2) \cap R)$ with L_1 being a linear language, L_2 an even linear language, R a regular language and g and h homomorphisms.*

Proof. Let us take L_1 and L_2 as in the previous theorem. Then, $L_1 \cap L_2$ is composed by strings in the form $x\#y^r$. In order to isolate the upper strand x we will apply an homomorphism h such that for any symbol $a_i \in \Sigma$ $h(i_1) = h(i_2) = a_i$ and $h(\#) = \#$. Then $h^{-1}(L_1 \cap L_2) = \{x_1\#y_1 : x_1 \in h^{-1}(x) \text{ and } y_1 \in h^{-1}(y)\}$. Now, let us define

the regular language $R = V_1^*\#V_2^*$ where $V_1 = \{i_1 : a_i \in \Sigma\}$ and $V_2 = \{i_2 : a_i \in \Sigma\}$. Finally, the homomorphism g is defined for all possible value i as $g(i_1) = a_i$, $g(i_2) = \lambda$ and $g(\#) = \lambda$. It can be easily proved that $L_u(M) = g(h^{-1}(L_1 \cap L_2) \cap R)$.

□

References

[1] R. Freund, G. Păun, G. Rozenberg, A. Salomaa. Watson-Crick finite automata In *Proceedings of DNA Based Computers III DIMACS Workshop (June, 1997)*, pp 297-327. The American Mathematical Society. 1999.

[2] J. Hopcroft, J. Ullman. Introduction to Automata Theory, Languages and Computation. Addison Wesley Publishing Co.,1979.

[3] G. Păun, G. Rozenberg, A. Salomaa. DNA Computing. New computing paradigms. Springer. 1998

[4] G. Rozenberg, A. Salomaa (Eds.). Handbook of Formal Languages Vol. 1. Springer. 1997.