

Simulation of a Turing Machine using EndoII splicing rules

Kamala Krithivasan, Department of CSE, IIT Madras 600036, India, kamala@iitm.ac.in
Anshu Bhatia, Department of Biotechnology, IIT Madras 600036, India, anshu@iitm.ac.in
Chandra T.S., Department of Biotechnology, IIT Madras 600036, India, chandra@iitm.ac.in

Abstract— In this paper we define splicing rules for class II restriction endonucleases and show how a Turing Machine can be simulated using such rules.

1) Introduction

There exist numerous theoretical models in DNA computing which have power equal to that of Turing Machines [1]. One of them is splicing system which models the recombinant behaviour of DNA strands. In many of the theoretical models to show the power of a system, simulation of type 0 grammar is used or other methods like simulation of matrix grammar is used. Rothmund has described the implementation of a Turing Machine using type II restriction endonucleases [2].

In this paper we define a slightly different type of splicing system. We show the equivalence of the new splicing system to the original system as described in [1]. We also show how to simulate a Turing Machine using this model.

In the next section we give the basic definitions needed for the paper. In section 3 we define the new type of splicing system and show its equivalence to the original one. In section 4 we use the system to simulate a TM. The paper concludes with a brief note in section 5.

2) Basic Definitions

In this section we give some basic definitions needed for the paper.

a) Turing Machine:

A Turing machine can be described by the following parameters:

- An infinite set of cells which can store any of the set of symbols from $S = \{s_0, s_1, \dots, s_n\}$.
- A set of states described, a member of the set $Q = \{q_0, q_1, q_2, \dots, q_p\}$.
- The head of the machine, which points to one of the infinite cells.
- Transition rules which are of the form:
 $\delta(q_x, s_v) \rightarrow (q_y, s_w, m)$, where q_x is the current state of the machine, s_v is the symbol pointed out by the head, q_y is the new state of the machine, s_w is the symbol which replaces

s_v and m is the movement of the head of machine which can be to the left or to the right.

The size of a machine with j symbols and k non halting states is $j \times k$. Given the input on the tape, the TM starts on the leftmost non-blank symbol in the initial state and if it halts in an accepting state, the input is accepted.

b) Splicing System:

Splicing systems are usually defined in the following manner. [1]

Consider an alphabet V and two special symbols, $\#$, $\$$ not in V . A splicing rule (over V) is a string of the form

$$r = u_1 \# u_2 \$ u_3 \# u_4$$

where $u_1, u_2, u_3, u_4 \in V^*$. For the above splicing rule $(x, y) \models_r z, w$ iff $x = x_1 u_1 u_2 x_2$, $y = y_1 u_3 u_4 y_2$, $z = x_1 u_1 u_4 y_2$ $w = y_1 u_3 u_2 x_2$ for some $x_1, x_2, y_1, y_2 \in V^*$

An H scheme is a pair $\sigma = (V, R)$, where V is an alphabet and $R \subseteq V^* \# V^* \$ V^* \# V^*$ is a set of splicing rules.

$\sigma_1^1(L) = \{z, w \in V^* \mid x, y \models_r z, w \text{ for some } x, y \in L, r \in R\}$.

For an H scheme $\sigma = (V, R)$ and a language

$L \in V^*$,

$\sigma_1^0 = L$,

$\sigma_1^{i+1}(L) = \sigma_1^i(L) \cup \sigma_1(\sigma_1^i(L))$, $i \geq 0$, and

$\sigma_1^*(L) = \bigcup_{i \geq 0} \sigma_1^i(L)$

$H_1(FL_1, FL_2) = \{\sigma_1^*(L) \mid L \in FL_1 \text{ and } \sigma = (V, R) \text{ with } R \in FL_2\}$

An extended H system is one where distinction is made between total and target alphabet. An EH system is defined as:

$\gamma = (V, T, A, R)$ where V is an alphabet, $T \subseteq V$, $A \subseteq V^*$ and $R \subseteq V^* \# V^* \$ V^* \# V^*$.

The language generated by γ is defined by: $L(\gamma) = \sigma_1^*(A) \cap T^*$. It has been shown that $EH(\text{REG}, \text{REG}) = \text{RE}$, where RE is the class of recursively enumerable languages.

Splicing Operation for a circular and linear string:

A circular string over V is a sequence $x =$

$a_1, a_2 \dots a_n$ for $a_i \in V$, $1 \leq i \leq n$, with the assumption that a_1 follows a_n . The set of all circular strings over V is denoted by V° . For a linear string x the corresponding circular string is \hat{x} . Consider an alphabet V and a splicing rule $r = u_1 \# u_2 \$ u_3 \# u_4$ over V . For $\hat{x} \in V^\circ$ and $v, w \in V^*$ $(\hat{x}, v) \models_r w$ iff $x = x_1 u_1 u_2, v = v_1 u_3 u_4 v_2, w = v_1 u_3 u_2 x_1 u_1 u_4 v_2$. This may be shown as:

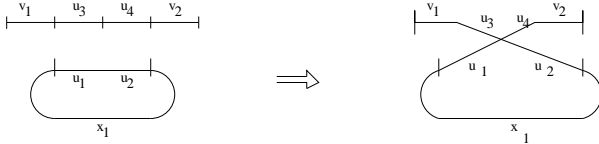


Fig. 1.

3) EndoII Systems

In this section we define new class of splicing systems which in reality have the same effect of the original systems. Since they model the behaviour of type II restriction endonucleases we call them as EndoII systems. Note that original splicing systems model type I restriction enzymes. We define these splicing systems in the following manner.

Consider an alphabet V , special symbols $\#, \$, L, R$ not in V and $l, k \in \mathbb{N}$, the set of natural numbers. A splicing rule (over V) is a string of the form:

$$r = \langle u_1, l, M_1 \$ u_2, k, M_2 \rangle$$

where M_1 and $M_2 \in \{L, R\}$. The above rule means that while the recognition site is the string u_1 in strand I the splicing site is l nucleotides upstream of the rightmost nucleotide of u_1 in case of $M=L$ and l nucleotides downstream of the leftmost nucleotide in case of $M=R$. Similar interpretations may be drawn for strand II also. As examples consider the following splicing rules and their results:

$$r = \langle u_1, l, R \$ u_2, k, R \rangle$$

For the above splicing rule $(x, y) \models_r z, w$ iff $x = x_1 u_1 x_2 x_3, y = y_1 u_2 y_2 y_3, z = x_1 u_1 x_2 y_3, w = y_1 u_2 y_2 x_3$ for some $x_1, x_2, y_1, y_2, u_1, u_2 \in V^*$ and $|u_1| + |x_2| = l$ and $|u_2| + |y_2| = k$, where $|t|$ represents the length of sequence t .

Pictorially this may be represented as:

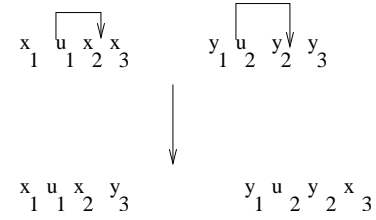


Fig. 2.

Consider another example:

$$r = \langle u_1, l, L \$ u_2, k, R \rangle$$

For the above splicing rule $(x, y) \models_r z, w$ iff $x = x_1 x_2 u_1 x_3, y = y_1 u_2 y_2 y_3, z = x_1 y_3, w = y_1 u_2 y_2 x_2 u_1 x_3$ for some $x_1, x_2, y_1, y_2, u_1, u_2 \in V^*$ and $|u_1| + |x_2| = l$ and $|u_2| + |y_2| = k$. This may be represented as:

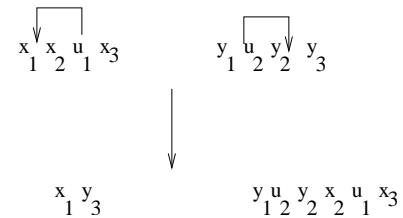


Fig. 3.

Similar rules can be written for $M_1=R, M_2=L$ and $M_1=L, M_2=L$

An EndoII H scheme is a pair $\sigma = (V, R)$, where V is an alphabet and $R \subseteq \langle V^*, (0, 1)^*, (L, R) \$ V^*, (0, 1)^*, (L, R) \rangle$, is a set of splicing rules.

$\sigma_1^1(L) = \{z, w \in V^* \mid x, y \models_r z, w \text{ for some } x, y \in L, r \in R\}$.

For an H scheme $\sigma = (V, R)$ and a language $L \in V^*$,

$$\sigma_1^0(L) = L,$$

$$\sigma_1^{i+1}(L) = \sigma_1^i(L) \cup \sigma_1(\sigma_1^i(L)), i \geq 0, \text{ and } \sigma_1^*(L) = \bigcup_{i \geq 0} \sigma_1^i(L)$$

$$H_1(FL_1, FL_2) = \{\sigma_1^*(L) \mid L \in FL_1 \text{ and } \sigma = (V, R) \text{ with } R \in FL_2\}$$

An extended H system is one where distinction is made between total and target alphabet. An EndoII EH system is defined as:

$\gamma = (V, T, A, R)$ where V is an alphabet, $T \subseteq V, A \subseteq V^*$ and R is the set of splicing rules as defined before.

The language generated by γ is defined by:

$$L(\gamma) = \sigma_1^*(A) \cap T^*.$$

These rules are applied for splicing between a circular and linear string in a similar way as in the original splicing system as elaborated in the previous section.

Next we prove the equivalence of the endoII system

with the original splicing system:

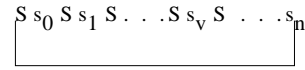


Fig. 4.

Theorem:

The class of languages generated by EndoII system is the same as the class of languages generated by the original system.

Proof:

We show that one type of rules can be simulated by the other type.

- a) Let $\Gamma = (V,T,A,R)$ be a original splicing system. We construct an equivalent EndoII system as: $\Gamma' = (V,T,A,R')$ as follows: If $r = u_1 \# u_2 \$ u_3 \# u_4$ is a rule in Γ then, $r = \langle u_1 u_2, l, R \$ u_3, u_4, k, R \rangle$ is a rule in Γ' , where $l = |u_1| + |u_2|$ and $k = |u_3| + |u_4|$.
- b) If $r = \langle u_1, l, R \$ u_2, k, R \rangle$ is a rule in Γ' , then,
 - i) for $l \leq |u_1|$ and $k \leq |u_2|$: $r = u_1' \# u_2' \$ u_3' \# u_4'$ is a rule in Γ where $u_1 = u_1' u_2'$, $|u_1'| = l$ and $u_2 = u_3' u_4'$, $|u_3'| = k$
 - ii) for $l \geq |u_1|$ and $k \leq |u_2|$: $r = u_1' x \# u_3' \# u_4'$ is a rule in Γ where $u_1 x = |l|$ and $|x| = l - |u_1|$. and u_3' and u_4' are same as defined in (i). Similar forms of proofs exist for
 - iii) $l \leq |u_1|$ and $k \geq |u_2|$
 - iv) $l \geq |u_1|$ and $k \geq |u_2|$

Thus we see that EndoII rules have the same effect as those in the original splicing systems.

4) Simulation of Turing Machine

In this section we first represent a TM using circular oligonucleotides(strings in the formal model) and then show how to simulate the working of a TM using an Endo II circular splicing system.

- a) **Symbols:** Here we are considering a general Turing machine with many symbols. Each of these symbols is represented by a specific sequence of nucleotides (strings in the formal model). Two symbols are always separated by a specific sequence which we label as the spacer sequence S. The initial sequence represents the starting state of the machine. The sequences representing the first and last cells are joined together. The example of such a representation is:

b) The State and Head Representation:

Each of the states is represented by a specific sequence of nucleotides(strings). The sequence representing the current state surrounds the symbol to which the head of the machine points from both ends.

For e.g. if the head currently points to a specific symbol s_v and the Turing machine is in the state q_y , then the corresponding instantaneous description(ID) of the Turing Machine is:

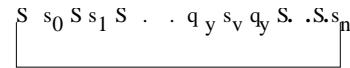


Fig. 5.

c) Transition rules for a Turing Machine:

All transition rules for a Turing Machine can be one of the following forms:

1. $\delta(q_x, s_v) \rightarrow (q_y, s_w, L)$
2. $\delta(q_x, s_v) \rightarrow (q_y, s_w, R)$

The first rule represents a left transition, the second rule represents a right transition. Without loss of generality we can assume a Turing machine to have two symbols and a number of states.

Next we present a set of splicing rules which simulate each of the above transition rules of a Turing Machine. This will make our simulation of a Turing Machine complete.

Simulating the left transition of a Turing Machine:

We want to apply the rule $\delta(q_x, s_v) \rightarrow (q_y, s_w, L)$. To simulate this move in the EndoII splicing system we use the following set of rules.

1. Introducing q_y to the left of the current symbol:

(a) $\langle q_x s_v q_x, |S| + |q_x s_v q_x|, L \$ R_1, |R_2| + |R_1|, R \rangle$, where $|S|$ and $|R|$ represent the length of these sequences.

Now consider the linear sequence, which is an axiom. (In fact for every move we have a set of axioms.)

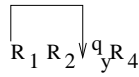


Fig. 6.

and the circular Turing Machine representation

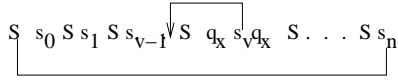


Fig. 7.

Application of (a) leads to

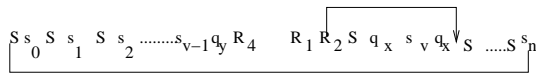


Fig. 8.

Rule 2: Changing s_v to s_w

$$(b) \langle R_2, |S| + |q_x s_v q_x| + |R_2|, R \$ R_5, |X| + |R_5|, L \rangle$$

Consider the above rule, the last sequence and the linear sequence(axiom):



Fig. 9.

Application of (b) leads to:

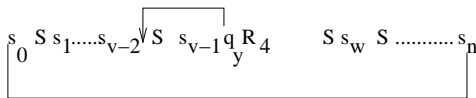


Fig. 10.

Rule 3: Changing the head of the machine to the left:

$$(c) \langle R_4 q_y s_{v-1}, |S| + |R_4 q_y s_{v-1}|, L \$ R_6, |X| + |R_6|, R \rangle$$

Consider the above rule, the last sequence and the linear sequence(axiom):

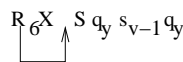


Fig. 11.

Application of (c) leads to fig.12.

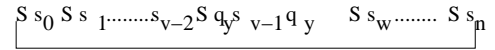


Fig. 12.

A ligase enzyme which ligates only state sequences ($q_1, q_2 \dots q_n$) to spacer sequence S may be used to recircularize the linear strand. Formally a linear string turns circular when the sequence at the beginning of string is a state sequence and end of the string is a spacer sequence or vice versa. In this case q_y and S are ligated and hence the left transition of the TM is completely simulated and results in a:

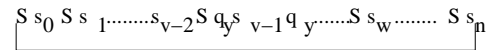


Fig. 13.

Simulating the right transition of a Turing Machine:

We want to apply $\delta(q_x, s_v) \rightarrow (q_y, s_w, R)$ to the above representation. We use the following set of rules.

Introducing q_y to the right of the current symbol:

$$(a) \langle q_x s_v q_x, |S| + |q_x s_v q_x|, R \$ R_1, |R_2| + |R_1|, L \rangle$$

Now consider the linear sequence(axiom): and the

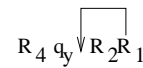


Fig. 14.

circular Turing Machine

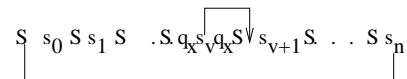


Fig. 15.

Application of (a) leads to

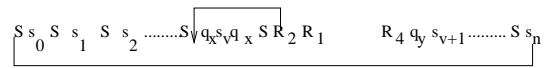


Fig. 16.

Rule 2: Changing s_v to s_w

$$(b) \langle R_2, |S| + |q_x s_v q_x| + |R_2|, L \$ R_5, |X| + |R_5|, R \rangle$$

Consider the above rule, the last sequence and the linear sequence:

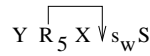


Fig. 17.

Application of (b) leads to:

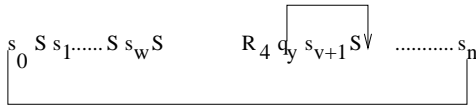


Fig. 18.

Rule 3: Changing the head of the machine to the right:

$$(c) \langle R_4 q_y s_{v+1}, |S| + |R_4 q_y s_{v+1}|, R \$ R_6, |X| + |R_6|, L \rangle$$

Consider the above rule, the last sequence and the linear sequence(axiom):

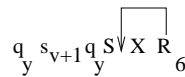


Fig. 19.

Application of (c) leads to:

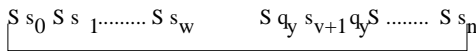


Fig. 20.

A similar ligase enzyme as explained in the left simulation is used to recircularize the linear strand. In this case q_y and S are ligated and hence the right transition of the TM is completely simulated and results in a:

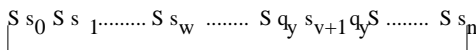


Fig. 21.

A specific state and symbol will put the TM in the halt state, and the circular representation of the TM will then no longer be modified.

5) Conclusion

In this paper we have defined a new splicing system (EndoII splicing system). We can generate a recursively

enumerable set (type 0 language) using such a EndoII system in three different stages. In the first all possible circular strings as inputs to TM may be generated in some sort of duplicated form using EndoII rules. One part of the duplicated string will then be the input for the Turing Machine. Rules similar to those described in the last section will be applied to one half of the duplicated strings in stage II. In the third stage the original half will be separated from the half modified by Turing Machine rules in case of all those representations which reach an accept stage. Thus this will provide us with the input accepted by the TM. In this way a recursively enumerable language can be generated. The details of stage I and III can be worked out.

Here we used EndoII rules to simulate a Turing Machine. This is interesting as any formal model defined so far only uses type 0 grammar for proving universality whereas here we show direct simulation of TM. It may be noted that type II endonucleases are the most widely available restriction enzymes and we have simulated a TM using a formal model corresponding to that. Further study of this model is being done.

References

- 1) Gh.Paun, G.Rozenberg and A.Salomaa, DNA Computing-New Computing Paradigms, Springer, 1998.
- 2) P.W.K Rothemund: A DNA and restriction enzyme implementation of Turing Machines,75-120 Proceedings of DIMACS workshop on DNA based computers, 1995.