Automaton Models Inspired by Peptide Computing

M. Sakthi Balan

Department of Computer Science
University of Western Ontario
London, Ontario
Canada
Contents

1. Introduction
2. Peptide Computing
3. Automaton Models
4. Conclusion
About the paper

- String Binding-Blocking automata and Rewriting Binding-Blocking Automata.
- Blocking of string of symbols
  - to read them later, or
  - to store some information.
- Analyze the power and study their hierarchical structure.
Objective

- Imparting ideas from peptide computing into a finite state automata and study its behavior.
  - Blocking,
  - Unblocking.

- How a sequential machine behaves when ideas from peptide computing are added to it.
Motivation

- Peptide computing.
- Interaction between peptides and antibodies.
- Binding of antibodies to specific regions in peptides.
- Affinity power associated with binding.
- Permanent or temporary elimination of part of peptide sequences by attaching antibodies having higher affinity.
Peptide Computing

- Proposed by H. Hug and R. Schuler [Hug, Schuler 2001].
- Solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Theoretical model *peptide computer* defined in [Balan, Jurgensen 2007].
Epitopes
Peptide sequence with antibodies

Epitopes
Generic Automaton Model

- Finite State Automata with
  - Blocking function,
  - Unblocking function, and
  - Affinity function.

- Blocking of symbols: Binding-Blocking Automata.

- Blocking of strings: String Binding-Blocking Automata.
Variations in the Automaton Model

Position of the head, after unblocking occurs:

- Leftmost transition – moves to leftmost unmarked, unblocked symbol.
- Locally leftmost transition – no change in the position.
String Binding-Blocking Automaton

\[ P = (Q, V, E, \delta, q_0, R, \beta_b, \beta_{ub}, Q_{accept}) \] where

- \( Q = Q_{block} \cup Q_{unblock} \cup Q_{general} \) is the set of states (pairwise disjoint),
- \( q_0 \in Q \) is the start state,
- \( V \) is a finite set of symbols,
- \( E \) is the finite subset of \( V^* \),
- \( \delta \) is the transition function from \( Q \times (E \cup \{\epsilon\}) \to 2^Q \),
- \( R \) is the partial order relation (called affinity/priority relation) on \( E \), i.e., \( R \subseteq E \times E \);
- \( \beta_b \) is the blocking function from \( Q_{block} \to \mathcal{L} \);
- \( \beta_{ub} \) is the unblocking function from \( Q_{unblock} \to \mathcal{L}' \);
- \( \mathcal{L} \) and \( \mathcal{L}' \) are finite set of family of languages over \( V \), i.e., \( \mathcal{L} = \{L_1, L_2, \ldots, L_k\} \), and \( \mathcal{L}' = \{L'_1, L'_2, \ldots, L'_r\} \); and \( Q_{accept} \subseteq Q \) where \( Q_{accept} \) is the set of accepting states.
- \( L_i \in \mathcal{L} \) is said to be a blocking language.
- \( \mathcal{L} \) is called as the family of blocking.
- \( \mathcal{L}' \) is called as the family of unblocking languages.
**Transitions**

- In reading state, reads a (higher priority) string and the strings are marked.
  - Head can read a string only when all symbols are neither marked nor blocked.

- In blocking state, blocks the maximal $L$-string starting from the position of the head.

- In unblocking state, unblocks all $L$-strings.
Instantaneous Description

- Transition starts in the state $q_0$ from the first symbol.
- At any point of time system will be in any one of the state: reading, blocking, or unblocking.
- String is accepted if all symbols are marked and the state of the system is in $Q_{accept}$. 
Example

**State Set**

\[
Q_{\text{general}} = \{ q_0, q_{a_1}, q_{a_2}, q_f \}
\]

\[
Q_{\text{block}} = \{ q^{\text{block}}_{a} \}
\]

\[
Q_{\text{unblock}} = \{ q^{\text{unblock}}_{a} \}
\]

\[
Q_{\text{accept}} = \{ q_f \}
\]

**Transitions**

\[
\delta(q_0, a) = \{ q^{\text{block}}_{a} \}
\]

\[
\delta(q^{\text{block}}_{a}, \epsilon) = \{ q_{a_1} \}
\]

\[
\delta(q_{a_1}, a) = \{ q_{a_2} \}
\]

\[
\delta(q_{a_2}, \epsilon) = \{ q_{\text{unblock}}_{a} \}
\]

\[
\delta(q_{\text{unblock}}_{a}, \epsilon) = \{ q_0 \}
\]

\[
\delta(q_0, b) = \{ q_f \}
\]

**(Un)blocking functions**

\[
\beta_b(q^{\text{block}}_{a}) = \{ a^n b \mid n \geq 0 \}
\]

\[
\beta_{ub}(q^{\text{unblock}}_{a}) = \{ a^n b \mid n \geq 0 \}
\]

**Language**

\[
L_1 = \{ a^n ba^n \mid n \geq 1 \}
\]
Example

State Set

\[ Q_{\text{general}} = \{ q_0, q_{a_1}, q_{a_2}, q_f \} \]
\[ Q_{\text{block}} = \{ q_{\text{block}} \} \]
\[ Q_{\text{unblock}} = \{ q_{\text{unblock}} \} \]
\[ Q_{\text{accept}} = \{ q_f \} \]

(Un)blocking functions

\[ \beta_b(q_{\text{block}}) = \{ a^n b \mid n \geq 0 \} \]
\[ \beta_{ub}(q_{\text{unblock}}) = \{ a^n b \mid n \geq 0 \} \]

Transitions

\[ \delta(q_0, a) = \{ q_{\text{block}} \} \]
\[ \delta(q_{\text{block}}, \epsilon) = \{ q_{a_1} \} \]
\[ \delta(q_{a_1}, a) = \{ q_{a_2} \} \]
\[ \delta(q_{a_2}, \epsilon) = \{ q_{\text{unblock}} \} \]
\[ \delta(q_{\text{unblock}}, \epsilon) = \{ q_0 \} \]
\[ \delta(q_0, b) = \{ q_f \} \]

Language

\[ L_1 = \{ a^n b a^n \mid n \geq 1 \} \]
Example

**State Set**

$Q_{general} = \{ q_0, q_{a_1}, q_{a_2}, q_f \}$

$Q_{block} = \{ q_{block}^a \}$

$Q_{unblock} = \{ q_{unblock}^a \}$

$Q_{accept} = \{ q_f \}$

**Transitions**

$\delta(q_0, a) = \{ q_{block}^a \}$

$\delta(q_{block}^a, \epsilon) = \{ q_{a_1} \}$

$\delta(q_{a_1}, a) = \{ q_{a_2} \}$

$\delta(q_{a_2}, \epsilon) = \{ q_{unblock}^a \}$

$\delta(q_{unblock}^a, \epsilon) = \{ q_0 \}$

$\delta(q_0, b) = \{ q_f \}$

**(Un)blocking functions**

$\beta_b(q_{block}^a) = \{ a^n b \mid n \geq 0 \}$

$\beta_{ub}(q_{unblock}^a) = \{ a^n b \mid n \geq 0 \}$

**Language**

$L_1 = \{ a^n ba^n \mid n \geq 1 \}$
**Example**

**State Set**
- $Q_{general} = \{ q_0, q_{a1}, q_{a2}, q_f \}$
- $Q_{block} = \{ q_{block}^a \}$
- $Q_{unblock} = \{ q_{unblock}^a \}$
- $Q_{accept} = \{ q_f \}$

**Transitions**
- $\delta(q_0, a) = \{ q_{block}^a \}$
- $\delta(q_{block}^a, \epsilon) = \{ q_{a1} \}$
- $\delta(q_{a1}, a) = \{ q_{a2} \}$
- $\delta(q_{a2}, \epsilon) = \{ q_{unblock}^a \}$
- $\delta(q_{unblock}^a, \epsilon) = \{ q_0 \}$
- $\delta(q_0, b) = \{ q_f \}$

**(Un)blocking functions**
- $\beta_b(q_{block}^a) = \{ a^n b \mid n \geq 0 \}$
- $\beta_{ub}(q_{unblock}^a) = \{ a^n b \mid n \geq 0 \}$

**Language**
- $L_1 = \{ a^n b a^n \mid n \geq 1 \}$
Example

1. $q_0 \ a \ a \ a \ b \ a \ a \ a$
   \[\uparrow - - - - - - - - -\]

2. $a \ a \ a \ b \ q^{\text{block}} \ a \ a \ a$
   \[\# \$ \$ \$ \uparrow - - - -\]

3. $a \ a \ a \ b \ q^{a_1} \ a \ a \ a$
   \[\# \$ \$ \$ \uparrow - - - -\]

4. $a \ a \ a \ b \ a \ q^{a_2} \ a \ a$
   \[\# \$ \$ \$ \# \uparrow - - - -\]

5. $a \ q^{\text{unblock}} \ a \ a \ b \ a \ a \ a$
   \[\# \uparrow - - - - \# - - - -\]

6. $a \ q_0 \ a \ a \ b \ a \ a \ a$
   \[\# \uparrow - - - - \# - - - -\]
Example

Automaton Models Inspired by Peptide Computing
## Automaton Models Inspired by Peptide Computing

### Example

<table>
<thead>
<tr>
<th>Step 13</th>
<th>Step 14</th>
<th>Step 15</th>
<th>Step 16</th>
<th>Step 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a\ a\ a\ b\ a\ a\ q_{a_1}\ a$</td>
<td>$a\ a\ a\ b\ a\ a\ q_{a_2}$</td>
<td>$a\ a\ a\ q_{\text{unblock}}\ b\ a\ a\ a$</td>
<td>$a\ a\ a\ q_0\ b\ a\ a\ a\ a$</td>
<td>$a\ a\ a\ b\ q_f\ a\ a\ a$</td>
</tr>
<tr>
<td>$#\ #\ #\ #\ #\ #\ #\ \uparrow\ -$</td>
<td>$#\ #\ #\ #\ #\ #\ #\ \uparrow\ -$</td>
<td>$#\ #\ #\ #\ #\ #\ #\ \uparrow\ -$</td>
<td>$#\ #\ #\ #\ #\ #\ #\ \uparrow\ -$</td>
<td>$#\ #\ #\ #\ #\ #\ #\ \uparrow\ -$</td>
</tr>
</tbody>
</table>

(Automaton Models)
Results

- $L_1$ accepted by $StrBBA$ is not accepted by any $BBA_i$.
- In order to equate the number of $a$'s on either side of $b$
  - $BBA$ system has to first block the symbol $a$.
  - blocking $a$’s will block both the strings of $a$.
  - if the system unblocks to equate the second string with the first string then the head comes to the first string of $a$, since the transition is leftmost.
- Shows $StrBBA$ accept languages not accepted by $BBA_i$. 
$L_2 = \{ a^{2n}(aca)^n \mid n \geq 1 \}$.

$L_2$ is accepted by $StrBBA_{||}$ whereas, it is not accepted by $BBA_{||}$.

- to match $a$ with a $aca$, the system has to know from where the substring $(aca)^n$ starts.
- in order to equate each $a$ with the substring $aca$ the system has to block all $a'$s then look for $aca$.
- blocking of $a$ will block all $a'$s in the substring $aca$.
- This shows the system can neither equate $a$ with $aca$ nor it knows the position where the string $aca$ starts.
StrBBA is more powerful than BBA

- In BBA, symbols are blocked; in StrBBA strings are blocked.
- The proof idea is:
  - Use states of the form $q^X$ where $X$ denotes symbols which are blocked.
  - For each reading transition we have two transitions one that blocks a string over $X$ and the other, the normal reading transition.
Conjecture: StrBBA is simulated by Random-context grammars

- Random-context grammars without forbidden context.
- We assume that the system StrBBA has no iterative blocking.
- The main idea:
  - Have one non-terminal to generate symbols when no blocking is present.
  - When blocking occurs transfer the control to a new non-terminal which generates symbols.
  - Likewise when unblocking occurs transfer the control the first non-terminal.
Rewriting Binding-Blocking Automaton

Definition

\[ \Gamma = (Q, \Sigma, V, \delta, M, \mathcal{R}, \mathcal{P}, q_0, F) \] where

- \( Q \) is the finite set of states and \( q_0 \in Q \) is the start state,
- \( \Sigma \) is the finite set of tape alphabet,
- \( V \subseteq \Sigma \) is a finite set of symbols called input alphabet,
- \( \delta \) is the transition function from \( Q \times V \rightarrow 2^{Q \times \{L,R\}} \),
- \( M \subseteq V \) is called the set of markers;
- \( \mathcal{R} \) is the set of posets over \( M \) called as affinity set (i.e, each \( R \in \mathcal{R} \) is a subset of \( M \times M \)),
- \( \mathcal{P} : Q \rightarrow \mathcal{R} \) called as state-affinity function, and
- \( F \subseteq Q \) where \( F \) is the set of accepting states.
### Instantaneous Description

\[
\begin{array}{cccccccc}
  a_1 & a_2 & \cdots & a_{i-1} & q & a_i & a_{i+1} & \cdots & a_n & b' & b' & \cdots & b' & \cdots \\
  A_1 & A_2 & \cdots & A_{i-1} & \uparrow & A_i & A_{i+1} & \cdots & A_n & A_{n+1} & A_{n+2} & \cdots & - & \cdots \\
\end{array}
\]
Result

For any Turing machine $TM$ there is an equivalent $RBBA$ system which accepts the same language as $TM$. 
String BBA

- Blocking of strings.
- Defined two transitions \( l \) and \( ll \).
- StrBBA more powerful than BBA.
- Bounded by RC without forbidden context.

Rewriting BBA

- Blocking symbols are more than one called markers.
- Equivalent to Turing machine.
Conclusion

String BBA
- Blocking of strings.
- Defined two transitions / and //.
- StrBBA more powerful than BBA.
- Bounded by RC without forbidden context.

Rewriting BBA
- Blocking symbols are more than one called markers.
- Equivalent to Turing machine.
Is $\mathcal{L}$ more powerful than $\mathcal{L}'$?

Is the power of Random-context grammars a tighter bound for the power of StrBBA?

Is StrBBA with finite blocking languages strictly contained in StrBBA?

Does affinity play an important role?
Thank You