Automaton Models Inspired by Peptide Computing

M. Sakthi Balan

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

Contents

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

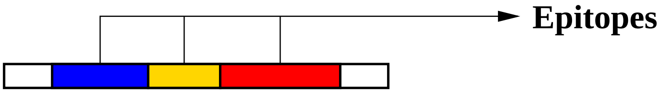
- Imparting ideas from peptide computing into a finite state automata and study its behavior.
  - Blocking,
  - Unblocking.
- How a sequential machine behaves when ideas from peptide computing are added to it.

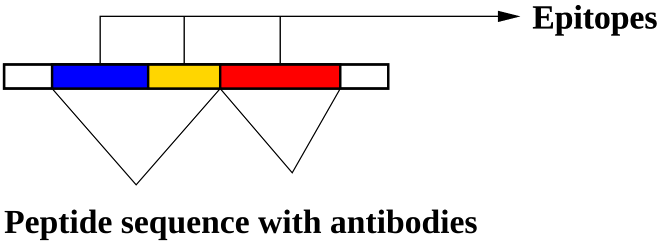
Motivation

- 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


Epitopes
Peptide sequence with antibodies

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.

Western

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

- $\mathcal{P} = (Q, V, E, \delta, q_0, R, \beta_b, \beta_{ub}, Q_{accept})$ 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^*,
  - δ is the transition function from $Q \times (E \cup \{\epsilon\}) \longrightarrow 2^Q$,
  - R is the partial order relation (called affinity/priority relation) on E, i.e., $R \subseteq E \times E$;
  - β_b is the blocking function from $Q_{block} \longrightarrow \mathcal{L}$;
  - β_{ub} is the unblocking function from $Q_{unblock} \longrightarrow \mathcal{L}'$
  - $\mathcal{L}$ and $\mathcal{L}'$ are finite set of family of languages over V, i.e., $\mathcal{L} = \{L_1, L_2, \cdots, L_k\}$, and $\mathcal{L}' = \{L'_1, L'_2, \cdots L'_r\}$; 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.

Western

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}.

Western

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

$$L_1 = \{a^n b a^n \mid n \geq 1\}$$

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

$$L_1 = \{a^n b a^n \mid n \geq 1\}$$

Western

Example

State Set

$$Q_{general} = \{q_0, q_{a_1}, q_{a_2}, q_f\}$$
$$Q_{block} = \{q^{block_a}\}$$
$$Q_{unblock} = \{q^{unblock_a}\}$$
$$Q_{accept} = \{q_f\}$$

(Un)blocking functions

$$\beta_b(q^{block_a}) = \{a^n b \mid n \geq 0\}$$
$$\beta_{ub}(q^{unblock_a}) = \{a^n b \mid n \geq 0\}$$

Transitions

$$\delta(q_0, a) = \{q^{block_a}\}$$
$$\delta(q^{block_a}, \epsilon) = \{q_{a_1}\}$$
$$\delta(q_{a_1}, a) = \{q_{a_2}\}$$
$$\delta(q_{a_2}, \epsilon) = \{q_{unblock_a}\}$$
$$\delta(q^{unblock_a}, \epsilon) = \{q_0\}$$
$$\delta(q_0, b) = \{q_f\}$$

Language

$$L_1 = \{a^n b a^n \mid n \geq 1\}$$

Western

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

$$L_1 = \{a^n b a^n \mid n \geq 1\}$$

Example

1

q_0	a	a	a	b	a	a	a
$\uparrow$	–	–	–	–	–	–	–

$\downarrow$

2

a	a	a	b	q^{block_a}	a	a	a
#	\$	\$	\$	$\uparrow$	–	–	–

$\downarrow$

3

a	a	a	b	q^{a_1}	a	a	a
#	\$	\$	\$	$\uparrow$	–	–	–

$\downarrow$

4

a	a	a	b	a	q^{a_2}	a	a
#	\$	\$	\$	#	$\uparrow$	–	–

$\downarrow$

5

a	$q^{unblock_a}$	a	a	b	a	a	a
#	$\uparrow$	–	–	–	#	–	–

$\downarrow$

6

a	q_0	a	a	b	a	a	a
#	$\uparrow$	–	–	–	#	–	–

$\downarrow$

Example

7

a	a	a	b	a	q^{block_a}	a	a
#	#	$\$$	$\$$	#	↑	—	—

↓

8

a	a	a	b	a	q_{a_1}	a	a
#	#	$\$$	$\$$	#	↑	—	—

↓

9

a	a	a	b	a	a	q_{a_2}	a
#	#	$\$$	$\$$	#	#	↑	—

↓

10

a	a	$q^{unblock_a}$	a	b	a	a	a
#	#	↑	—	—	#	#	—

↓

11

a	a	q_0	a	b	a	a	a
#	#	↑	—	—	#	#	—

↓

12

a	a	a	b	a	a	q^{block_a}	a
#	#	#	$\$$	#	#	↑	—

↓

Example

13

a	a	a	b	a	a	q_{a_1}	a
#	#	#	$	#	#	↑	–

↓

14

a	a	a	b	a	a	a	q_{a_2}
#	#	#	$	#	#	#	↑

↓

15

a	a	a	$q^{unblock_a}$	b	a	a	a
#	#	#	↑	–	#	#	#

↓

16

a	a	a	q_0	b	a	a	a
#	#	#	↑	–	#	#	#

↓

17

a	a	a	b	q_f	a	a	a
#	#	#	#	↑	#	#	#

Western

Results

- L_1 accepted by *StrBBA* is not accepted by any BBA_l.
- 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_l.

Western

Results

- $L_2 = \{a^{2n}(aca)^n \mid n \geq 1\}$.
- 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.

Western

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,
- Σ is the finite set of tape alphabet,
- $V \subseteq \Sigma$ is a finite set of symbols called input alphabet,
- δ is the transition function from $Q \times \frac{\Sigma}{V} \longrightarrow 2^{Q \times \{L,R\}}$,
- $M \subseteq V$ 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 $M \times M$),
- $\mathcal{P} : Q \longrightarrow \mathcal{R}$ called as state-affinity function, and
- $F \subseteq Q$ where F is the set of accepting states.

Instantaneous Description

a_1	a_2	$\cdots$	a_{i-1}	q	a_i	a_{i+1}	$\cdots$	a_n	$\not b$	$\not b$	$\cdots$	$\not b$	$\cdots$	
A_1	A_2	$\cdots$	A_{i-1}	$\uparrow$	A_i	A_{i+1}	$\cdots$	A_n	A_{n+1}	A_{n+2}	$\cdots$	$-$	$\cdots$	

Result

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

Conclusion

String BBA

- Blocking of strings.
- Defined two transitions *I* and *II*.
- 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 *I* and *II*.
- 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

- Is *II* more powerful than *I*?
- 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