String Binding-Blocking Automata*

M. Sakthi Balan

Department of Computer Science and Engineering,
Indian Institute of Technology, Madras
Chennai – 600036, India
sakthi@cs.iitm.ernet.in

In a similar way to DNA hybridization, antibodies which specifically recognize peptide sequences can be used for calculation [3,4]. In [4] the concept of peptide computing via peptide-antibody interaction is introduced and an algorithm to solve the satisfiability problem is given. In [3], (1) it is proved that peptide computing is computationally complete and (2) a method to solve two well-known NP-complete problems namely Hamiltonian path problem and exact cover by 3-set problem (a variation of set cover problem) using the interactions between peptides and antibodies is given.

In our earlier paper [1], we proposed a theoretical model called as binding-blocking automata (BBA) for computing with peptide-antibody interactions. In [1] we define two types of transitions - leftmost(l) and locally leftmost(ll) of BBA and prove that the acceptance power of multihead finite automata is sandwiched between the acceptance power of BBA in l and ll transitions. In this work we define a variant of binding-blocking automata called as string binding-blocking automata and analyze the acceptance power of the new model.

The model of binding-blocking automaton can be informally said as a finite state automaton (reading a string of symbols at a time) with (1) blocking and unblocking functions and (2) priority relation in reading of symbols. Blocking and unblocking facilitates skipping [1] some symbols at some instant and reading it when it is necessary.

In the sequel we state some results from [1,2] - (1) for every BBA there exists an equivalent BBA without priority, (2) for every language accepted by BBA with l transition, there exists BBA with ll transitions accepting the same language, (3) for every language accepted by BBA with l transition there is an equivalent multi-head finite automata which accepts the same language and (4) for every language L accepted by a multi-head finite automaton there is a language L' accepted by BBA such that L can be written in the form $h^{-1}(L')$ where h is a homomorphism from L to L'.

The basic model of the string binding-blocking automaton is very similar to a BBA but for the blocking and unblocking. Some string of symbols (starting form the head's position) can be blocked from being read by the head. So only those symbols which are not already read and not blocked can be read by the head. The finite control of the automaton is divided into three sets of states namely blocking states, unblocking states and general reading states. A read symbol can not be read gain, but a blocked symbol can be unblocked and read.

* Financial support from Infosys Technologies Limited, India is acknowledged

[1] running through the symbols without reading

Let us suppose the input string is y. At any time the system can be in any one of the three states - reading state, blocking state or unblocking state. In reading state the system can read a string of symbols (say l symbols) at a time and move its head l positions to the right. In the blocking state q, the system blocks a string of symbols as specified by the blocking function (say $x \in L$ where $L \in \beta_b(q)$, $x \in Sub(y)$ [2]) starting from the position of the head. The string x satisfies the maximal property i.e., there exists no $z \in L$ such that $x \in Pre(z)$ [3] and $z \in Sub(y)$. When the system is in the unblocking state q the recently blocked string $x \in Sub(y)$ and $x \in L$ where $L \in \beta_{ub}(q)$ is unblocked. We note that the head can only read symbols which are neither read nor blocked. The symbols which are read by the head are called *marked* symbols, which are blocked are called as *blocked* symbols.

A string binding-blocking automaton with D-transition is denoted by $strbba_D$ and the language accepted by the above automaton is denoted by $StrBBA_D$. If the blocking languages are finite languages then the above system is represented by $strbba(Fin)$.

We show that $strbba_l$ system is more powerful than *bba* system working in l transition by showing that $L = \{a^n b a^n \mid n \geq 1\}$ is accepted by $strbba_l$ but not by any *bba* working in l transition.

The above language is accepted by $strbba_{ll}$.

The language $L = \{a^{2n+1}(aca)^{2n+1} \mid n \geq 1\}$ shows that $strbba_l l$ system is more powerful than *bba* system working in ll transition.

We also prove the following results,

1. For any bba_{ll} we can construct an equivalent $strbba_{ll}$.
2. For every $L \in StrBBA_l$ there exists a random-context grammar RC with Context-free rules such that $L(RC) = L$.
3. For every $strbba_D$, $\mathcal{P}$ there is an equivalent $strbba_D$, $\mathcal{Q}$ such that there is only one accepting state and there is no transition from the accepting state.

Hence by above examples and results we have $\mathcal{L}(bba_{ll}) \subset \mathcal{L}(strbba_{ll})$ and $\mathcal{L}(bba_l) \neq \mathcal{L}(strbba_l)$

References

1. M.Sakthi Balan and Kamala Krithivasan. Blocking-binding automata. poster presentation in Eigth International Confernce on DNA based Computers, 2002.
2. M.Sakthi Balan and Kamala Krithivasan. Normal-forms of binding-blocking automata. poster presentation in Unconventional Models of Computing, 2002.
3. M.Sakthi Balan, Kamala Krithivasan, and Y.Sivasubramanyam. Peptide computing – universality and complexity. In Natasha Jonoska and Nadrian Seeman, editors, *Proceedings of Seventh International Conference on DNA Based Computers – DNA7, LNCS*, volume 2340, pages 290–299, 2002.
4. Hubert Hug and Rainer Schuler. Strategies for the developement of a peptide computer. *Bioinformatics*, 17:364–368, 2001.

[2] $Sub(y)$ is the set of all sub-strings of y
[3] $Pre(z)$ is the set of all prefixes of z