"Sciences reach a point where they become mathematized..... It occurred in physics about the time of the Renaissance; it began in chemistry after John Dalton developed atomic theory; and it is just now happening in biology"

"They were built by 3 billion years of evolution, and we're just beginning to tap their potential to serve non-biological purposes. Nature has given us an incredible toolbox, and we're starting to explore what we might build."

Leonard Adleman

- M.Sakthi Balan, String Binding-Blocking Automata,Proceedings of Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science 2723, 2003, pp. 425-426.
- M.Sakthi Balan, Complexity Measures for Binding-Blocking Automata, communicated to Journal of Automata, Languages and Combinatorics, 2003.
- M. Sakthi Balan, K. Krithivasan, Blocking-Binding Automata, Preliminary proceedings of International Meeting on DNA Based Computers, M. Hagiya and A. Ohuchi (Eds.), 2002, pp. 327. (poster presentation)
- M. Sakthi Balan, K. Krithivasan, Normal-Forms of Blocking-Binding Automata, third International Conference on Unconventional Models of Computation, Published as CDMTCS Research Report at the University of Auckland, CDMTCS-195, C.S. Calude and M.J. Dinneen and F. Peper (Eds.), 2002, pp. 3 (poster presentation)
- M.Sakthi Balan and K. Krithivasan, Parallel Computation of Simple Arithmetic using Peptide-Antibody Interactions, International Workshop on Information Processing in Cells and Tissues, 2003.
- M. Sakthi Balan and K. Krithivasan, Binding-Blocking Automata, communicated to International Journal of Theory of Computing Systems, 2003.
- M. Sakthi Balan, Realizing Switching Functions using Peptide-Antibody Interactions, communicated, 2003.

List of Publications

- M. Sakthi Balan, K.Krithivasan and Y.Sivasubramanyam, Peptide Computing: Universality and Computing , Proceedings of Seventh International Conference on DNA Based Computers, Natasha Jonoska and Nedrian Seeman (Eds.): DNA7, LNCS 2340, pp. 290-299, 2002.
- M. Sakthi Balan, Complexity Issues in Binding-Blocking Automata, Pre-proceedings of International Workshop on Descriptional Complexity of Formal Systems, J. Dassow, M. Hoeberechts, H. Jürgensen and D. Wotschke (Eds.), 2002, pp. 43-54
- M.Sakthi Balan, Parallel Communicating Pushdown Automata with Filters in Communication, In J. Dassow and D. Wotschke, editors, Proc. of the Third International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures, Vienna, Austria, July 2001, pages 167-175. Preprint Nr. 16 of the Fakultat fur Informatik, Otto-von-Guericke-Universitat, Magdeburg, 2001.
- M. Sakthi Balan, Kamala Krithivasan and Mutyam Madhu, Some variants in Communication of Parallel Communicating Pushdown Automata, Journal of Automata, Languages and Combinatorics, Volume 8 (2003), No. 3, pp. 401-416.
- M. Sakthi Balan, Watson-Crick Distributed Automata, SIAM Discrete Mathematics Conference, San Diego, USA, 2002.
- M.Sakthi Balan, Algorithms for Peptide Computer, National Conference on Algorithms and Artificial Systems, 2003.

Acknowledgement: Saravanakumar Narayanan, TU-München, Germany for helpful discussions.

- Modeled switching operations and simple arithmetical operations using peptide-antibody interactions.
- Studied the complexity measures in binding-blocking automata
- Extended the definition of binding-blocking automata to string binding-blocking automata and rewriting binding-blocking automata
- Proved that the power of string binding-blocking automata is strictly more than that of binding-blocking automata
- Rewriting binding-blocking automata is universally complete

- Modeled switching operations and simple arithmetical operations using peptide-antibody interactions.
- Studied the complexity measures in binding-blocking automata
- Extended the definition of binding-blocking automata to string binding-blocking automata and rewriting binding-blocking automata
- Proved that the power of string binding-blocking automata is strictly more than that of binding-blocking automata

- Modeled switching operations and simple arithmetical operations using peptide-antibody interactions.
- Studied the complexity measures in binding-blocking automata
- Extended the definition of binding-blocking automata to string binding-blocking automata and rewriting binding-blocking automata

- Modeled switching operations and simple arithmetical operations using peptide-antibody interactions.
- Studied the complexity measures in binding-blocking automata

- Modeled switching operations and simple arithmetical operations using peptide-antibody interactions.

- We simulate a Turing machine using a RBBA
- Rewriting of Turing machine is taken care by the markers
- The states of the RBBA system is taken as 3-tuple $[q, a, p]$ where
  - $\rightarrow$ p denotes the system is in,
  - $\rightarrow$ q states the previous state of the system,
  - $\rightarrow$ a is the symbol read last.
- If a symbol a is rewritten by A when in the state p, which is got from the state p then the state-affinity has the pair (A, a) which gives more affinity to A than a.

- $\Gamma = (Q, \Sigma, V, \delta, M, \mathcal{R}, \mathcal{P}, q_0, F)$ where
  - Q is the finite set of states,
  - $q_0 \in Q$ 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 $Q \times \begin{matrix}\Sigma \\ V\end{matrix} \longrightarrow 2^{Q \times \begin{matrix}\Sigma \\ V\end{matrix} \times \{L,R\}}$,
  - $M \subseteq V$ 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 $M \times M$),
  - $\mathcal{P}$ is defined by, $\mathcal{P} : Q \longrightarrow \mathcal{R}$ called as state-affinity function and
  - $F \subseteq Q$ where F is the set of accepting states

- A finite control,
- An infinite tape which is divided into cells,
- A 2-way *tape head* which scans a cell to its right at a time.
- Each cell of the tape may hold exactly one of a finite number of *tape symbols*.
- When the symbols are read they are marked.
- Marking is done with help of a particular set called Marker set
- There is a set of poset relations, where each poset is defined on the set M.
- This poset relation helps to replace the markers when the necessity arises.

- The power of StrBBA in l transition is strictly more than BBA in l transition.
- The power of StrBBA in ll transition is strictly more than languages not accepted by BBA in ll.
- For every $L \in StrBBA_l$ there exists a random-context grammar RC with Context-free rules such that $L(RC) = L$.
- The set of all languages accepted by $strbba_l(Fin)$ is equal to the set of all regular languages.

- The basic model is very similar to a BBA.
- The head can read a sequence of symbols from its present position.
- When the symbols are read they are marked.
- String of symbols (starting form the head's position) can be blocked from being read by the head.
- Only those symbols which are not marked and not blocked can be read by the head.
- The fi nite control of the automaton is divided into three sets of states namely blocking states, unblocking states and general reading states.
- A marked symbol can not be read gain, but a blocked symbol can be unblocked and read again.
- Blocking is maximal.

(a) $REG \subset bba_D(*,1,*) \subset bba_D(*,2,*) \subset bba_D(*,3,*) \subset \cdots\cdots$ where $D \in \{(ll,b),(ll,f),(l,b),(l,f)\}$.

(b) $REG \subset bba_D(1,*,*) \subset bba_D(2,*,*) \subset bba_D(l,*,*) \subset \cdots\cdots$

(c) $bba_D(*,*,1) = bba_D(*,*,k),\ k \geq 2,\ D \in \{(l,b),(l,f),(ll,b),(ll,f)\}$

- Blocking number is the total number of sets of symbols blocked by the system at any point of time. Lies between 1 and 2^V.
- Blocking instant is defi ned as the maximum number of symbols blocked at any point of time.
- Blocking quotient of a subset X of alphabet is the length of the longest run from the blocking of X to the unblocking of X.
- Blocking quotient of a BBA system is defi ned as the maximum of blocking quotient over all the subsets of alphabet.
- $\mathcal{P}(k, m, n)$ denoted a BBA $\mathcal{P}$ with k the blocking number, m the blocking instant and n blocking quotient.

1. (l, b)-transition
2. (l, f)-transition
3. (ll, b)-transition
4. (ll, f)-transition

$l \longrightarrow$ leftmost transition

$ll \longrightarrow$ locally leftmost transition

$b \longrightarrow$ blocked transition

$f \longrightarrow$ free transition

- The symbols read by the head are called marked symbols.
- The symbols blocked are called as blocked symbols.
- The head can read a sequence of symbols from its present position.
- Only those symbols which are not marked and not blocked can be read by the head.

- $\mathcal{P} = (Q, V, E, \delta, q_0, R, \beta_b, \beta_{ub}, Q_{accept}, Q_{reject})$,
- $Q = Q_{block} \cup Q_{unblock} \cup Q_{general}$,
- $q_0 \in Q$ (start state), V is a fi nite set of symbols, E is the fi nite subset of V^*,
- δ is the transition function from $Q \times E \longrightarrow Q$,
- $R \subseteq E \times E$ is the partial order relation (called as affi nity relation) on E,
- β_b is the blocking function from $Q_{block} \longrightarrow 2^V$,
- β_{ub} is the unblocking function from $Q_{unblock} \longrightarrow 2^V$,
- $Q_{accept} \cup Q_{reject} \subseteq Q_{general}$ where Q_{accept} is the set of accepting states and Q_{reject} is the set of rejecting states.

- Consists of
  - → fi nite control
  - → fi nite tape
  - → tape head
  - → fi nite tape symbols
  - → transition function
  - → partial order relation
  - → blocking and unblocking functions

Automata Model Motivated by Peptide-Antibody Interaction

- For all described calculations peptides can be in an aqueous solution or bound to a chip surface.
- More than 20 amino acids that are used in the living world can be used for the peptide computer.
- The antibodies binding with postulated affi nities to their peptide epitopes can be obtained by screening phage display libraries.
- There are many biochemical methods to decipher the interaction of peptides with antibodies - Nuclear Magnetic Resonance Spectroscopy (NMR).
- Gates are reversible.
- The obvious limitation for the peptide computers is the laborious laboratory work in obtaining the monoclonal antibodies.

$Inverttozero(C, n)$

1. $Same(C)$
2. Add antibody T_{BAi},
3. Add antibody A_i

$SUB(A, B, C)$

1. $BlockInversion(I_1, B, B')$ where $I_1 = n-1.\cdots.0$,
2. $ADD(B', ONE, B'')$ where
   $ONE = a_{n-1}a_{n-2}\cdots a_1 1, a_i = 0,\ 1 \leq i \leq n-1$
3. $ADD(A, B'', C)$.
4. $Inverttozero(C, n)$

- First step gets the $1's$ complement of the number B
- The second step gives the $2's$ complement of the number B
- The penultimate step adds the numbers A and B'' to get the result in
- The Last step discards the n^{th} digit

$Same(C)$

1. Add excess of epitopes y_i which will remove all the antibodies T_{ABi} (it at all it is binding to the site y_i).
2. Add antibodies A_i
3. Add excess of epitopes z_i
4. Add antibodies B_i

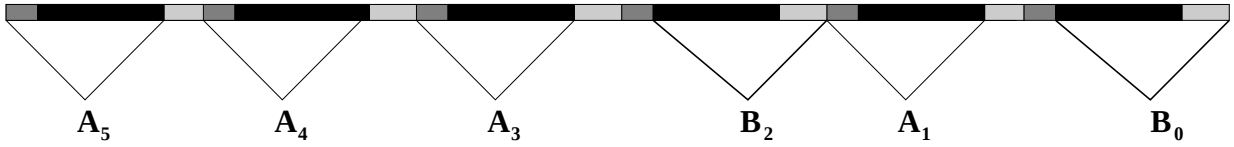
- The output is not suitable for performing additional operations.
- The reason - antibodies do not bind with the switching epitopes.
- Through a simple procedure we can make the peptide sequence again suitable for performing additional operations.

$ADD(A, B, C)$

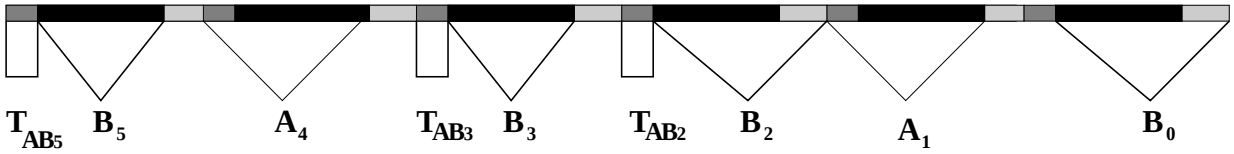
1. $XOR(A, B, C)$,
2. $BlockInversion(I_1, I_2, \cdots, I_k, C)$

where $I_j, 1 \leq j \leq k$ are carry blocks and k is the number of carry blocks.

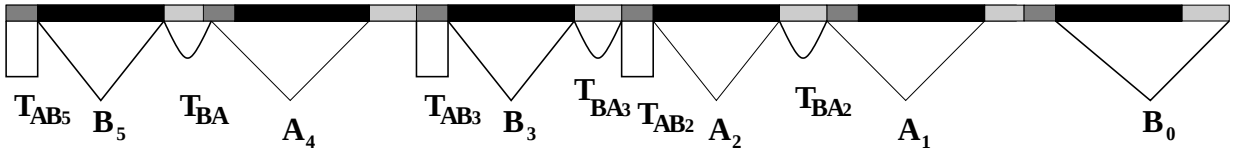
Addition - Example



After two steps



After the step 3(b)



After the step 3(d)

Example

- Two numbers be $A = 10110$ and $B = 10011$.
- At the end of the fi rst two steps the peptide sequence will be representing the number 000101.
- Third set of steps involves inverting all the digits occurring in the carry block.

Addition Algorithm

1. Add antibodies A_i where $1 \leq i \leq n$ such that either $a_i = 0$ and $b_i = 0$ or $a_i = 1$ and $b_i = 1$
2. Add antibodies B_i where $1 \leq i \leq n$ such that either $a_i = 0$ and $b_i = 1$ or $a_i = 0$ and $b_i = 1$
3. For all carry block $j_k.j_k - 1. \cdots .i_k + 1$ where k is the number of carry blocks in the given pair of binary numbers do the following in parallel. For $i_k + 1 \leq s \leq j_k$,
   (a) Add antibodies T_{ABs}
   (b) Add antibodies B_s
   (c) Add antibodies T_{BAs}
   (d) Add antibodies A_s

- Second step:
  - → Inverting the bits which are involved in carry propagations.
  - → Let $j.j-1.\cdots .i+1$ be a carry block.
  - → First we will put the antibodies $T_{ABk}, i+1 \leq k \leq j$ which will remove all the A_k, $i+1 \leq k \leq j$ antibodies
  - → Put the B_i antibodies, which will attach with all the vacant x_i sites. So all the 0 positions have been converted to 1 positions.
  - → Put antibodies $T_{BAk}, i+1 \leq k \leq j$ which removes all the antibodies B_i corresponding to the digit 1.
  - → We put A_i antibodies which binds to all the vacant positions x_i. This step converts all the 1's into 0's in the carry block.
- This above (second step) process can be done for all the carry blocks in parallel.

- The prepared peptide sequence is taken in an aqueous solution.
- The first step:
  - → Put antibody A_i in the aqueous solution if $a_i = b_i = 0$ or $a_i = b_i = 1$.
  - → Antibody B_i is put iff $a_i = 1,\ b_i = 0$ or $a_i = 0,\ b_i = 1$ where $0 \leq i \leq n-1$.
  - → The n^{th} digit is first initialized to *zero* so the antibody A_n is also put into the aqueous solution.

- Carry occurs only when both the bits a_i and b_i are 1.
- The carry propagates to the left until both the bits a_j and b_j ($j > i$) are 0.
- If there is no such j then the propagation carries on and ultimately n^{th} digit becomes 1.
- The sequence of bits $j.j-1.\cdots.i+1$ (if there is no such $j, 1 \leq j \leq n-1$ the $j = n$) where propagation carries and dies out as a *carry block*.
- For each of the carry block $j.j-1.\cdots.i+1$ we invert the digits $c_k, i+1 \leq k \leq j$ (inverting the digit means changing 0 to 1 and 1 to 0).

The value of $c_i, 1 \leq i \leq n$ is the same as the output of XOR operations:

	a_i	b_i	c_i
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

The bit c_n is initialized to zero

- Addition of any two binary numbers can be viewed in two steps.
  1. Guessing the answer for each bit without taking account of the carry.
  2. Propagation of a *carry*.
- $A = a_{n-1}a_{n-2}\cdots a_0$ and $B = b_{n-1}b_{n-2}\cdots b_0$ are two binary numbers, where $a_i, b_i \in \{0,1\}$.
- Let $C = A + B$ where $C = c_n c_{n-1}\cdots c_0$

- $t = t_{n-1} \cdots t_1 t_0$, where $t_i \in \{0, 1\}$, such that $t = \Sigma_{i=0}^{n-1} t_i . 2^i$.
- The exact binary number is stored by the way of binding the specifi ed antibodies to its specifi c binding sites.
- If the i^{th} bit from the right is 1 (0) then the antibody B_i (A_i) is bounded to the epitope $x_i z_i$ $(y_i x_i)$.
- If 10101 is the binary number the peptide-antibody representation is

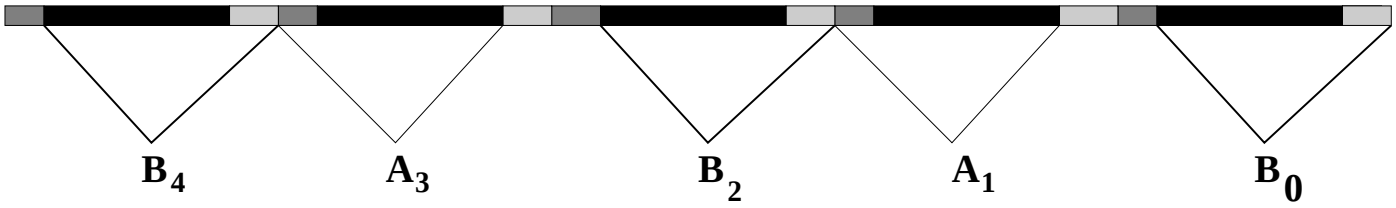


Figure 6: Number representation

- The binding site for the antibodies A_i are $y_i x_i$ and x_i with the former one having more affi nity,
- The binding site for the antibodies B_i are $x_i z_i$ and x_i with the former one having more affi nity,
- The binding site for the antibodies T_{ABi} are z_i,
- The binding site for the antibodies T_{BAi} are y_i
- The antibodies A_i are used to denote the i^{th} bit as *zero*.
- The antibodies B_i are used to denote the i^{th} bit as *one*.
- The antibodies T_{ABi} are used to switch i^{th} bit from $zero$ to one
- The antibodies T_{BAi} are used to switch i^{th} bit from one to $zero$.

Four sets of antibodies $\mathcal{A}$, $\mathcal{B}$, $\mathcal{T}_{AB}$ and $\mathcal{T}_{BA}$

- $\mathcal{A} = \{A_0, A_1, \cdots, A_{n-1}\}$,
- $\mathcal{B} = \{B_0, B_1, \cdots, B_{n-1}\}$,
- $\mathcal{T}_{AB} = \{T_{AB0}, T_{AB1}, \cdots, T_{AB(n-1)}\}$,
- $\mathcal{T}_{BA} = \{T_{BA0}, T_{BA1}, \cdots, T_{BA(n-1)}\}$.

Proposed Model

- Peptide sequence and set of antibodies.
- Peptide sequence consists of n position specifi c epitopes.
- Each epitope has three parts of which two parts are for switching purpose, other part is a general one.
- $ep_i = y_i x_i z_i,\ 0 \leq i \leq n-1$ denote one epitope then y_i and z_i are called the *switching epitopes* for the i^{th} bit.

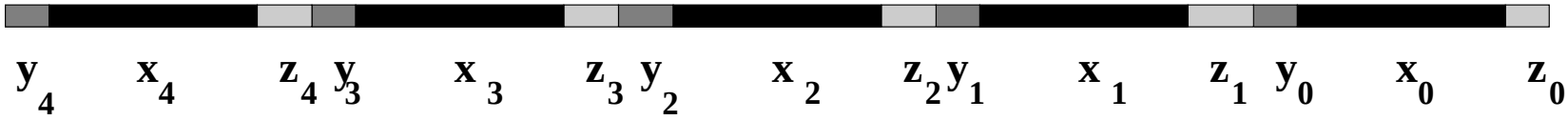


Figure 5: Peptide sequence for a 5-digit number

Modeling Simple Arithmetic Operations using Peptide-Antibody Interactions

NOR Gate

In the OR gate construction, the following changes are made,

- C_{init} denotes the bit 1 and
- C_f denotes 0

If at least one of the input is 1 the output is 0.

NAND Gate

In the AND gate construction we do the following changes

- The antibody C_{init} denotes the bit 0.
- The antibody C_f (labeled antibody) denotes the bit 0.

If at least one of the input is 0 the output is 1.

Algorithm

1. Take the peptide sequence P in an aqueous solution.
2. Add the antibody C_{init}.
3. Add antibodies corresponding to the input bits.
4. Add antibody C_f.
5. Add antibody C_0.

- If the inputs are same either the epitope $x_1 x x_2$ is bounded by the antibody C_{init}, or
- The epitope x will be free for the antibody C_0 to come and bind to it.
- Above two steps makes sure that the output is 0 whenever the inputs are same.
- When the inputs are different either the epitope $x_1 x$ or $x x_2$ will be free - the antibody C_f can bind to either of them.
- The above step guarantee that the output will be 1 when the inputs are different.

- Input bits 0 and 1 are represented by the antibodies A_i and B_i respectively where $1 \leq i \leq 2$.
- The antibodies C_{init} and C_0 denotes the bit 0.
- The antibody C_f (labeled antibody) denotes the bit 1.
- $epitope(A_1) = \{y\}, epitope(A_2) = \{z\},$
- $epitope(B_1) = \{yx_1\}, epitope(B_2) = \{x_2z\},$
- $epitope(C_{init}) = \{x_1xx_2\}, epitope(C_f) = \{x_1x, xx_2\}$ and $epitope(C_0) = \{x\},$
- $aff(B_i) > aff(C_{init}) > aff(C_f) > aff(C_0)$, $1 \leq i \leq 2$.

- The model for XOR gate requires little change
- One more antibody is needed and
- The binding sites for the output antibodies are different.

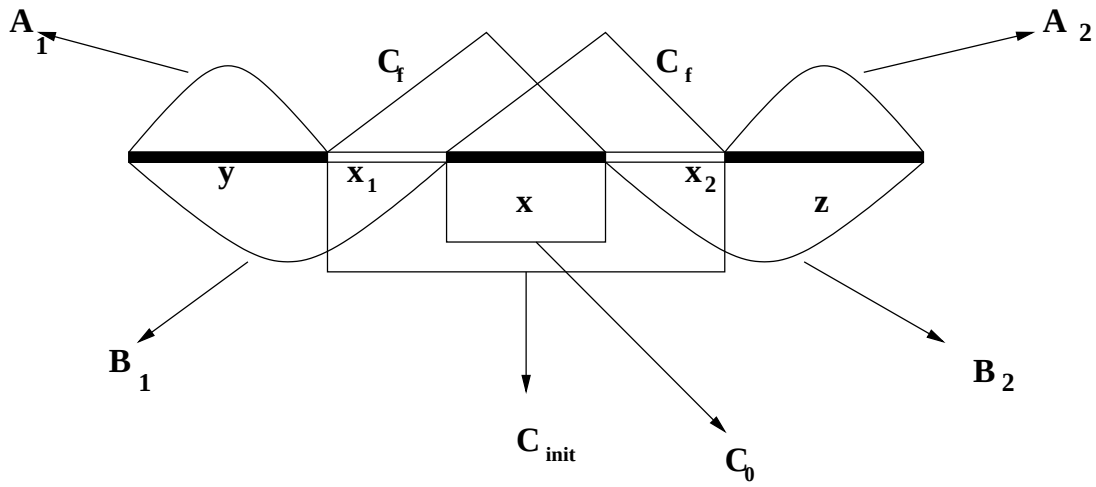


Figure 4: Peptide sequence with possible antibodies

Algorithm

- Take the peptide sequence P in an aqueous solution.
- Add the antibody C_{init}.
- Add antibody corresponding to the input bit.
- Add antibody C_f.

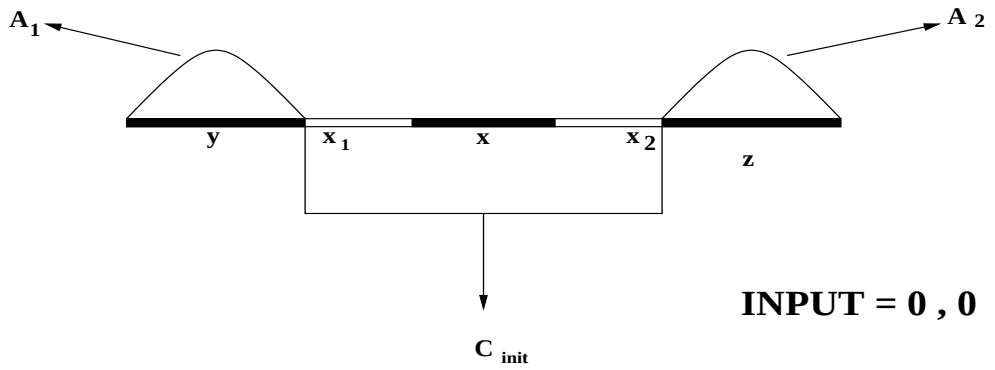
The initial bit denoting 0 is toggled only if the input bit is 0.

- Peptide sequence is $P = xx_2z$
- The antibodies are A_1 and B_1.
- The antibody C_{init} denotes the bit 0.
- The antibody C_f (labeled antibody) denotes the bit 1.
- $epitope(B_2) = \{z\}$,
- $epitope(A_2) = \{x_2z\}$,
- $epitope(C_{init}) = \{xx_2\}, epitope(C_f) = \{x\}$.
- $aff(A_i) > aff(C_{init}) > aff(C_f), 1 \leq i \leq 2$.

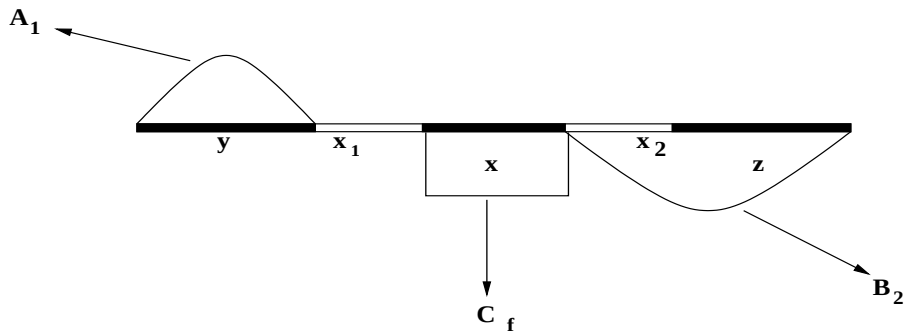
AND Gate

- The antibody C_{init} denotes the bit 1.
- The antibody C_f (labeled antibody) denotes the bit 0.
- $epitope(B_1) = \{y\}, epitope(B_2) = \{z\},$
- $epitope(A_1) = \{yx_1\}, epitope(A_2) = \{x_2z\},$
- $epitope(C_{init}) = \{x_1xx_2\}, epitope(C_f) = \{x\}.$
- $aff(A_i) > aff(C_{init}) > aff(C_f), 1 \leq i \leq 2.$

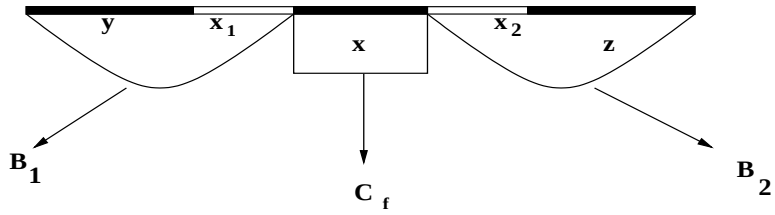
Output is 0 if at least one of the input is 0.



INPUT = 0 , 0



INPUT = 0 , 1



INPUT = 1 , 1

Figure 3: OR gate

Algorithm

1. Take the peptide sequence P in an aqueous solution.
2. Add the antibody C_{init}.
3. Add antibodies corresponding to the input bits. For example if the fi rst bit is 1 and the second bit is 0 then add antibodies B_1 and A_2.
4. Add antibody C_f.

If the output has to be seen the antibody C_f can be given some color so that at the end of the algorithm if fluorescence is detected the output will be 1 or else it will be 0.

- The output 1 occurs if at least one of the inputs is 1.
- Start with an initial output of 0 - antibody C_{init} binds to its epitope.
- C_{init} is toggled if at least one 1 comes as an input. For this to be carried out
  - → the epitopes for the antibody C_{init} and the antibody $B_i, 1 \leq i \leq 2$ are taken as overlapping ones.
  - → $aff(B_i) > aff(C_{init}) > aff(C_f), 1 \leq i \leq 2$.
  - → This facilitates toggle of output bit to 1 - antibody C_f binds to its epitope.

- Input bits 0 and 1 are represented by the antibodies A_i and B_i respectively where $1 \leq i \leq 2$.
- The antibody C_{init} denotes the bit 0.
- The antibody C_f (labeled antibody) denotes the bit 1.
- $epitope(A_1) = \{y\}, epitope(A_2) = \{z\},$
- $epitope(B_1) = \{yx_1\}, epitope(B_2) = \{x_2z\},$
- $epitope(C_{init}) = \{x_1xx_2\}, epitope(C_f) = \{x\}.$
- $aff(B_i) > aff(C_{init}) > aff(C_f), 1 \leq i \leq 2.$

- Input bits 0 and 1 are represented by the antibodies A_i and B_i respectively where $1 \leq i \leq 2$.
- The antibody C_{init} denotes the bit 0.
- The antibody C_f (labeled antibody) denotes the bit 1.
- $epitope(A_1) = \{y\}, epitope(A_2) = \{z\},$
- $epitope(B_1) = \{yx_1\}, epitope(B_2) = \{x_2z\},$
- $epitope(C_{init}) = \{x_1xx_2\}, epitope(C_f) = \{x\}.$

- Input bits 0 and 1 are represented by the antibodies A_i and B_i respectively where $1 \leq i \leq 2$.

- $epitope(A_1) = \{y\}, epitope(A_2) = \{z\},$
- $epitope(B_1) = \{yx_1\}, epitope(B_2) = \{x_2z\},$

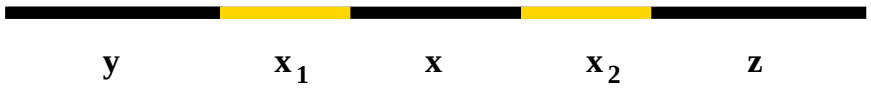


Figure 1: Peptide sequence

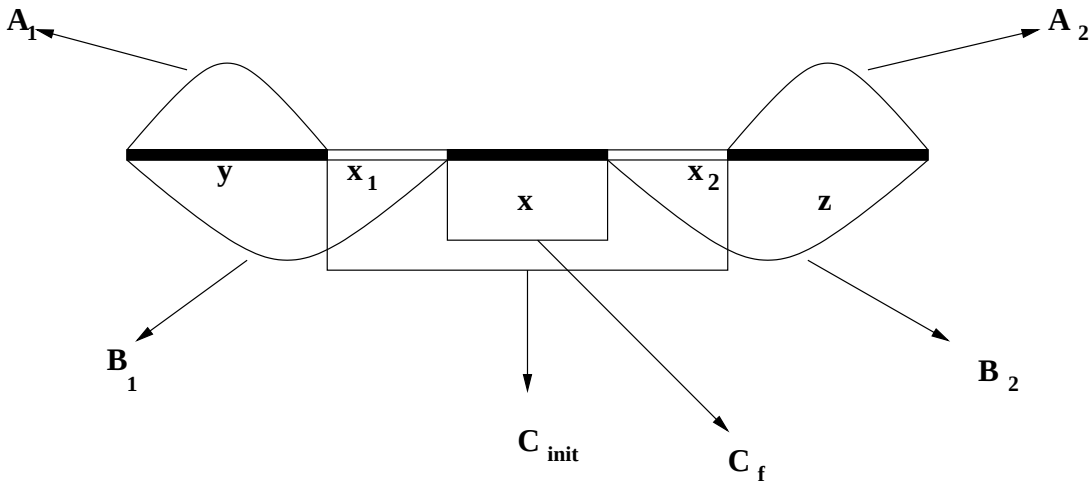


Figure 2: Peptide sequence with possible antibodies

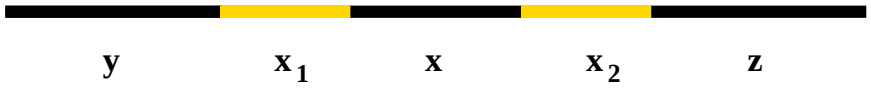


Figure 1: Peptide sequence

Proposed Model

- Epitopes x and y are the binding places for the antibodies denoting the input.
- Epitope $x_1 x x_2$ is the place where the antibody representing the initial output binds.
- Epitopes $x_1 x$, $x x_2$ and x are the binding places for the antibodies denoting the output.

Proposed Model

- Consists of a peptide sequence and some set of antibodies.
- Peptide sequence consists of fi ve epitopes, $P = yx_1xx_2z$ where each y, x_1, x, x_2 and z are epitopes.
- Seven antibodies denoted by $A_1, A_2, B_1, B_2, C_{init}$ and C_f.
- Antibodies A_1, A_2, B_1 and B_2 denote the inputs.
- C_{init} denote the initial value of the result of the operation.
- C_{init} and C_f denote the output of the operation.

Modeling Switching Operations using Peptide-Antibody Interactions

Peptide, Antibodies

- Sequence of amino acids attached by covalent bonds - peptide bonds
- Peptide consists of recognition sites - epitopes
- Peptide can contain more than one epitope for the same or different antibodies.
- Each antibody which attaches to a specifi c epitope there is a binding power associated with it called affi nity.
- The antibody with greater affi nity gets the higher priority for binding to its epitope.

- Hubert Hug and Rainer Schuler introduced the model.
- Solved satisfi ability problem.
- We solved
  - → Hamiltonian path problem and set cover problem.
  - → Universality of peptide computing.
  - → Introduced Binding-Blocking Automata(BBA).
  - → Analyzed the power of BBA in four variants $\{left, locally - leftmost\} \times \{free, blocked\}$

- Hubert Hug and Rainer Schuler introduced the model.
- Solved satisfi ability problem.
- We solved
  - → Hamiltonian path problem and set cover problem.
  - → Universality of peptide computing.
  - → Introduced Binding-Blocking Automata(BBA).

- Hubert Hug and Rainer Schuler introduced the model.
- Solved satisfi ability problem.
- We solved
  - → Hamiltonian path problem and set cover problem.
  - → Universality of peptide computing.

- Hubert Hug and Rainer Schuler introduced the model.
- Solved satisfi ability problem.
- We solved
  - → Hamiltonian path problem and set cover problem.

- Hubert Hug and Rainer Schuler introduced the model.
- Solved satisfi ability problem.

Peptide Computing

- Peptides, antibodies and their interactions.
- Peptides represent the sample space of a given problem.
- Antibodies are used to select certain subsets of this sample space.
- Parallel interactions between peptide sequences and antibodies.

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions
- Modeling Simple Arithmetic using Peptide Antibody Interactions
- Automata Model - Binding-Blocking Automata
- Complexity Issues in Binding-Blocking Automata
- String Binding-Blocking Automata
- Rewriting Binding-Blocking Automata
- Conclusion

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions
- Modeling Simple Arithmetic using Peptide Antibody Interactions
- Automata Model - Binding-Blocking Automata
- Complexity Issues in Binding-Blocking Automata
- String Binding-Blocking Automata
- Rewriting Binding-Blocking Automata

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions
- Modeling Simple Arithmetic using Peptide Antibody Interactions
- Automata Model - Binding-Blocking Automata
- Complexity Issues in Binding-Blocking Automata
- String Binding-Blocking Automata

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions
- Modeling Simple Arithmetic using Peptide Antibody Interactions
- Automata Model - Binding-Blocking Automata
- Complexity Issues in Binding-Blocking Automata

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions
- Modeling Simple Arithmetic using Peptide Antibody Interactions
- Automata Model - Binding-Blocking Automata

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions
- Modeling Simple Arithmetic using Peptide Antibody Interactions

Contents

- Peptide Computing
- Modeling Switching Operations using Peptide Antibody Interactions

Contents

- Peptide Computing

Computational Models using Peptide-Antibody Interactions

by

M. Sakthi Balan

Theoretical Computer Science Lab

Department of Computer Science and Engineering

Indian Institute of Technology Madras