Computational Models using Peptide- Antibody Interactions

M. Sakthi Balan
and
Kamala Krithivasan

Theoretical Computer Science Lab
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai – 600036.

Organization

- Peptide Computing
- Objective of the Work
- Solving NP-Complete problems
- Modeling gate operations
- Modeling arithmetic operations
- Binding-Blocking Automata (BBA)
- Normal-forms, Complexity measures of BBA
- String Binding-Blocking Automata
- Rewriting Binding-Blocking Automata
- Conclusion

Peptide Computing

- Introduced by Hubert Hug and Rainer Schuler in 2000.
- Solved satisfiability problem using peptide- antibody interactions.

Peptide Computing

- Uses peptides and antibodies
- Operation – binding of antibodies to epitopes in peptides
- ***Epitope*** – The site in peptide recognized by antibody
- Highly ***non- deterministic***
- Massive ***parallelism***

Peptide Computing Contd..

- Peptides – sequence of amino acids
- Twenty amino acids.

  Example – Glycine, Valine
- Connected by covalent bonds

Peptide Computing Contd..

- Antibodies recognizes epitopes by binding to it
- Binding of antibodies to epitopes has associated power called ***affinity***
- Higher priority to the antibody with larger affinity power

Objectives

- Using peptide computing for solving NP-Complete problems.
- To show this model is universal.
- Modeling switching operations, arithmetical operations.
- Defining automata model inspired by interactions between peptides and antibodies – Binding-Blocking Automata.

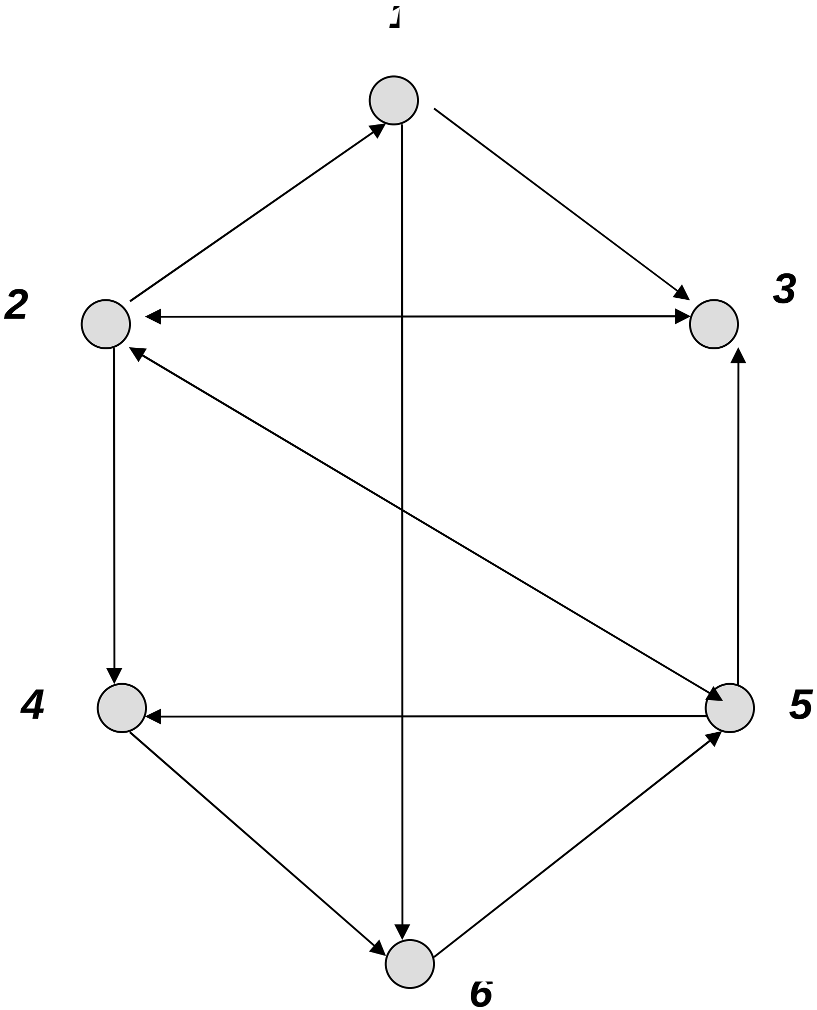
Objectives (Contd..)

- Analyze the power of Binding-Blocking Automata (BBA).
- Define and study some complexity measures for BBA.
- Define several variants of the model and study the acceptance power.

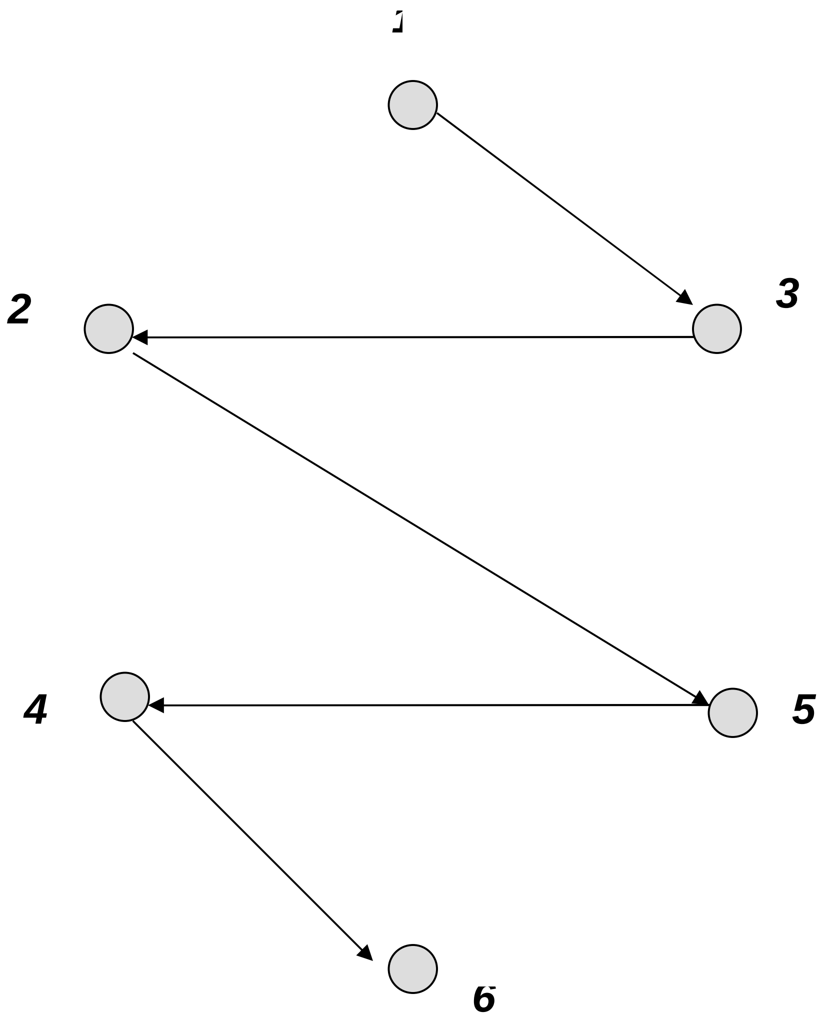
Solving NP- Complete Problems

Hamiltonian Path Problem

- $G = (V,E)$ is a directed graph
- $V = \{v_1,v_2,\ldots,v_n\}$ is the vertex set
- $E = \{e_{ij} \mid v_i$ is adjacent to $v_j\}$ is the edge set
- v_1 - source vertex, v_n – end vertex
- **Problem** – Test whether there exists a Hamiltonian path between v_1 and v_n



1
2
3
4
5
6



1
2
3
4
5
6

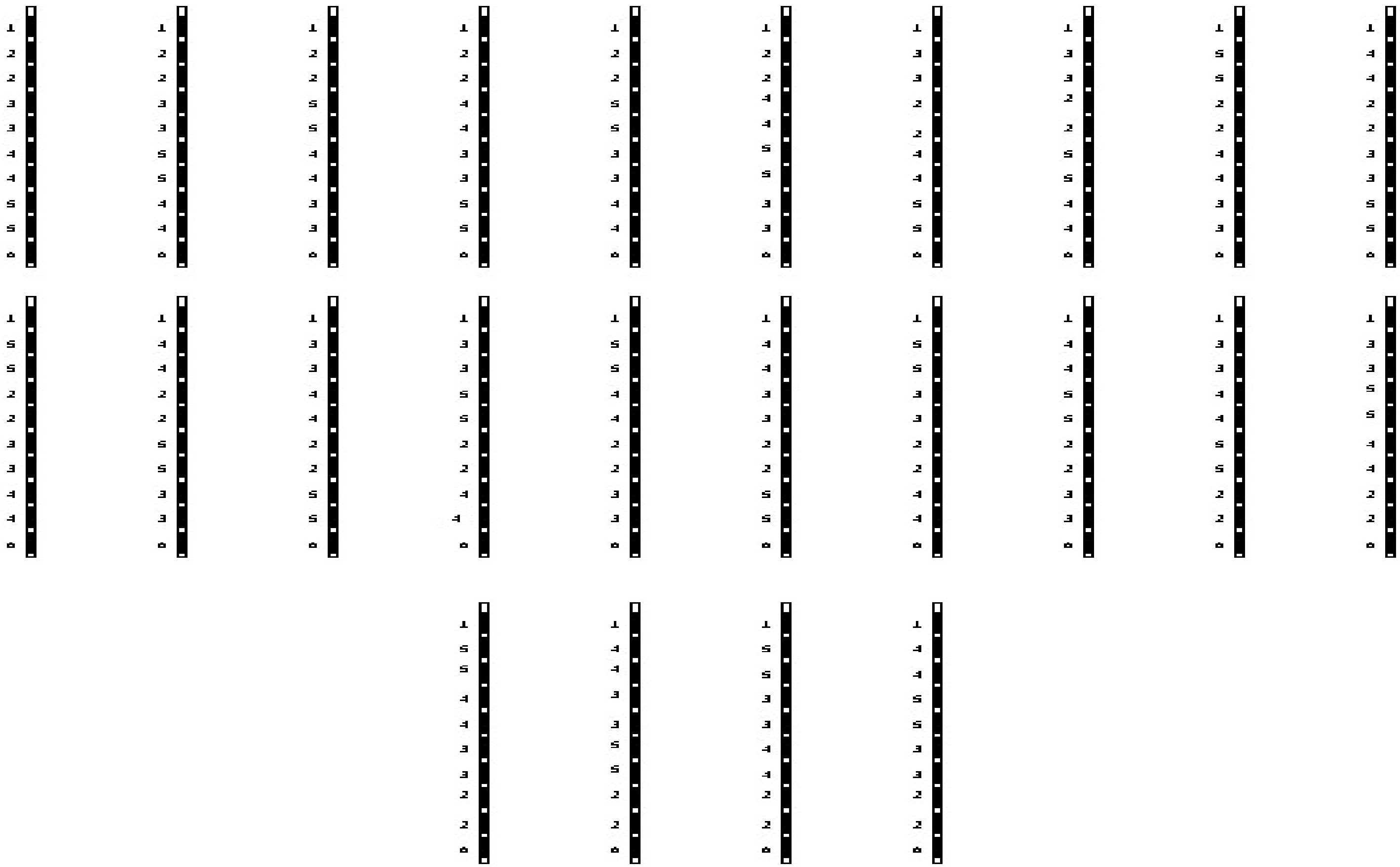
Peptides Formation

- For each possible solution there is a peptide sequence formed.
- Each of the pair of epitopes present in the peptide sequence denote a possible edge of the graph.

Antibody Formation

- Three sets of antibodies are formed.
- One set to recognize all the legal edges.
- One set to recognize all edges not present in the graph.
- Last one to recognize the Hamiltonian path of the graph.

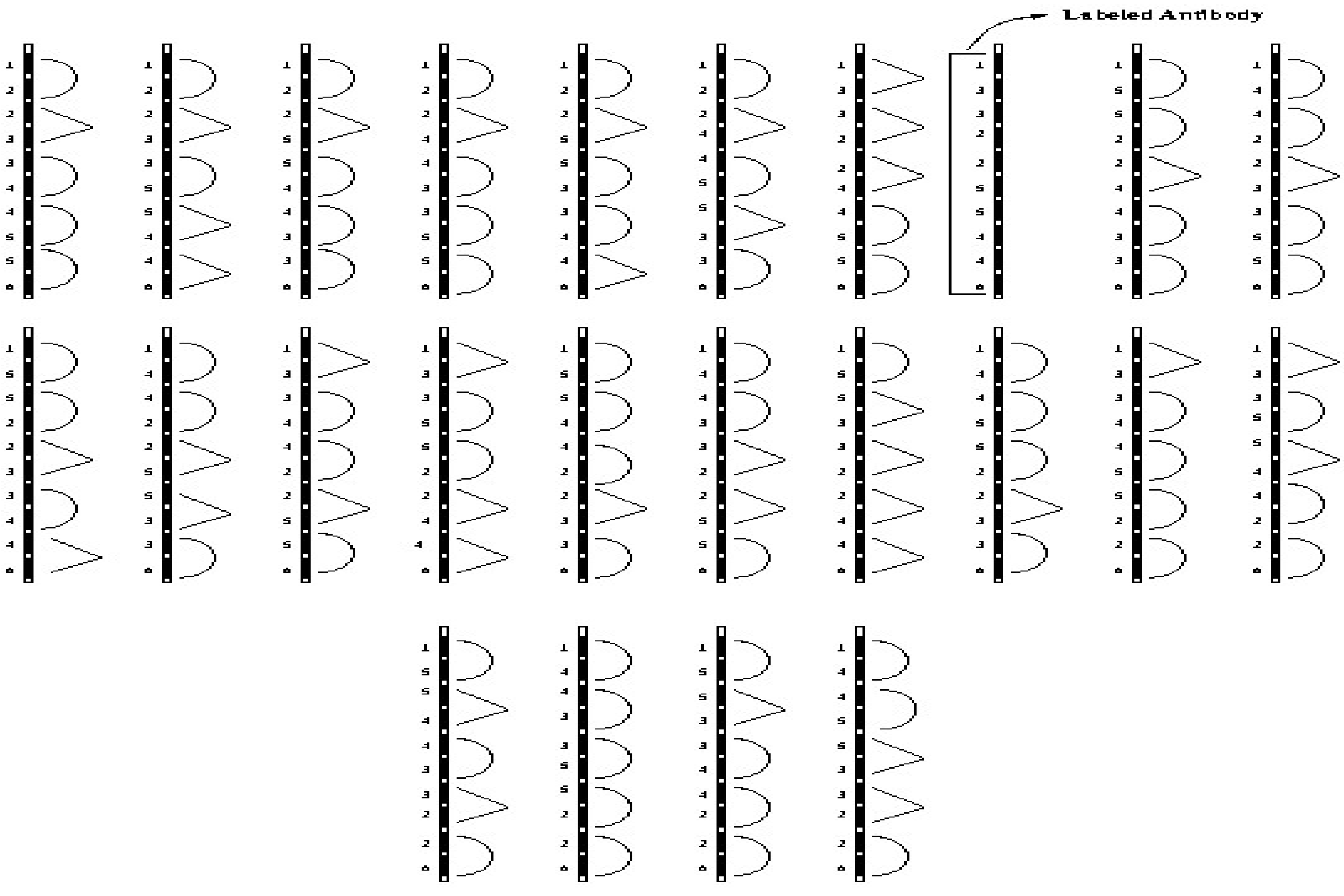
Peptide Solution Space



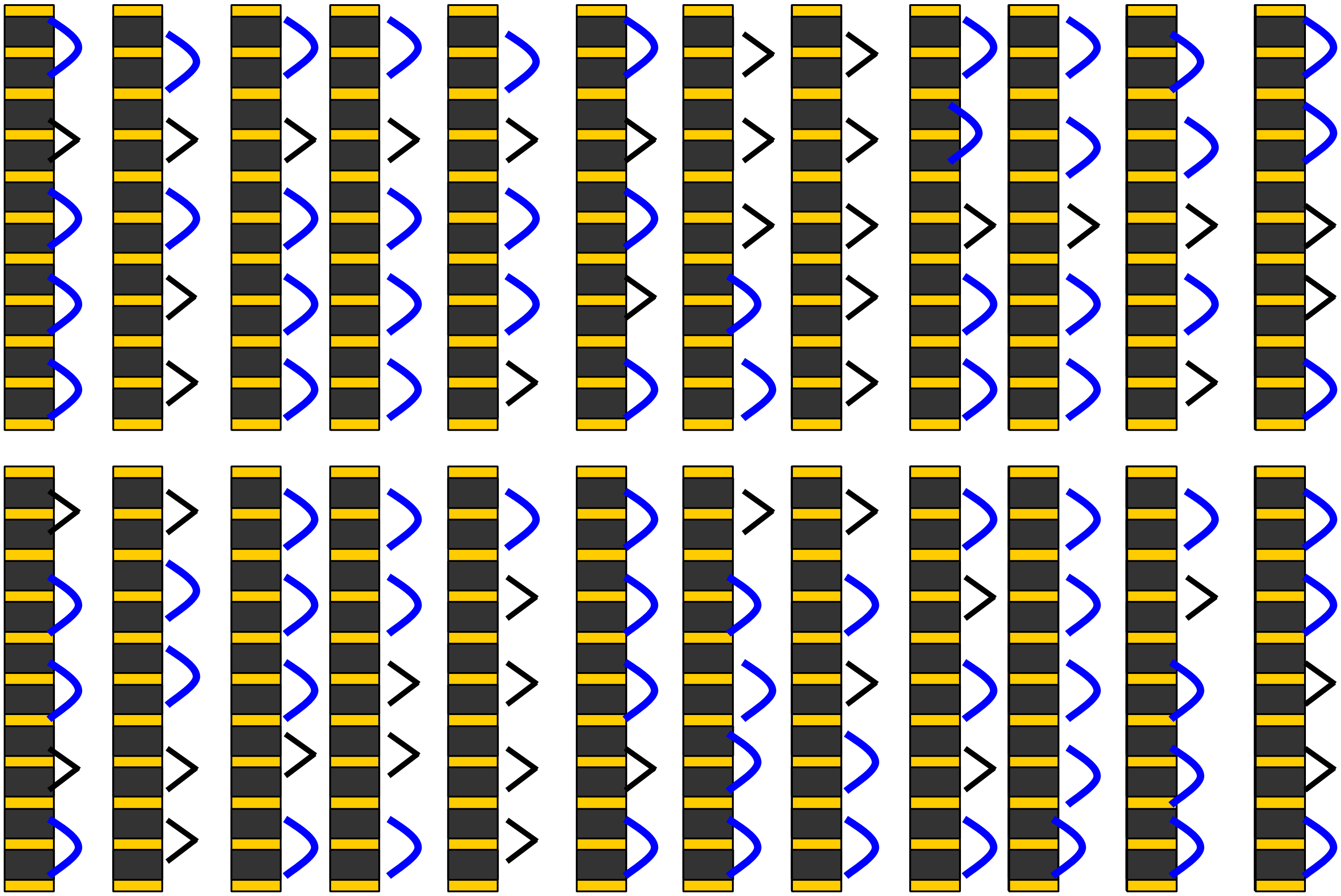
Algorithm

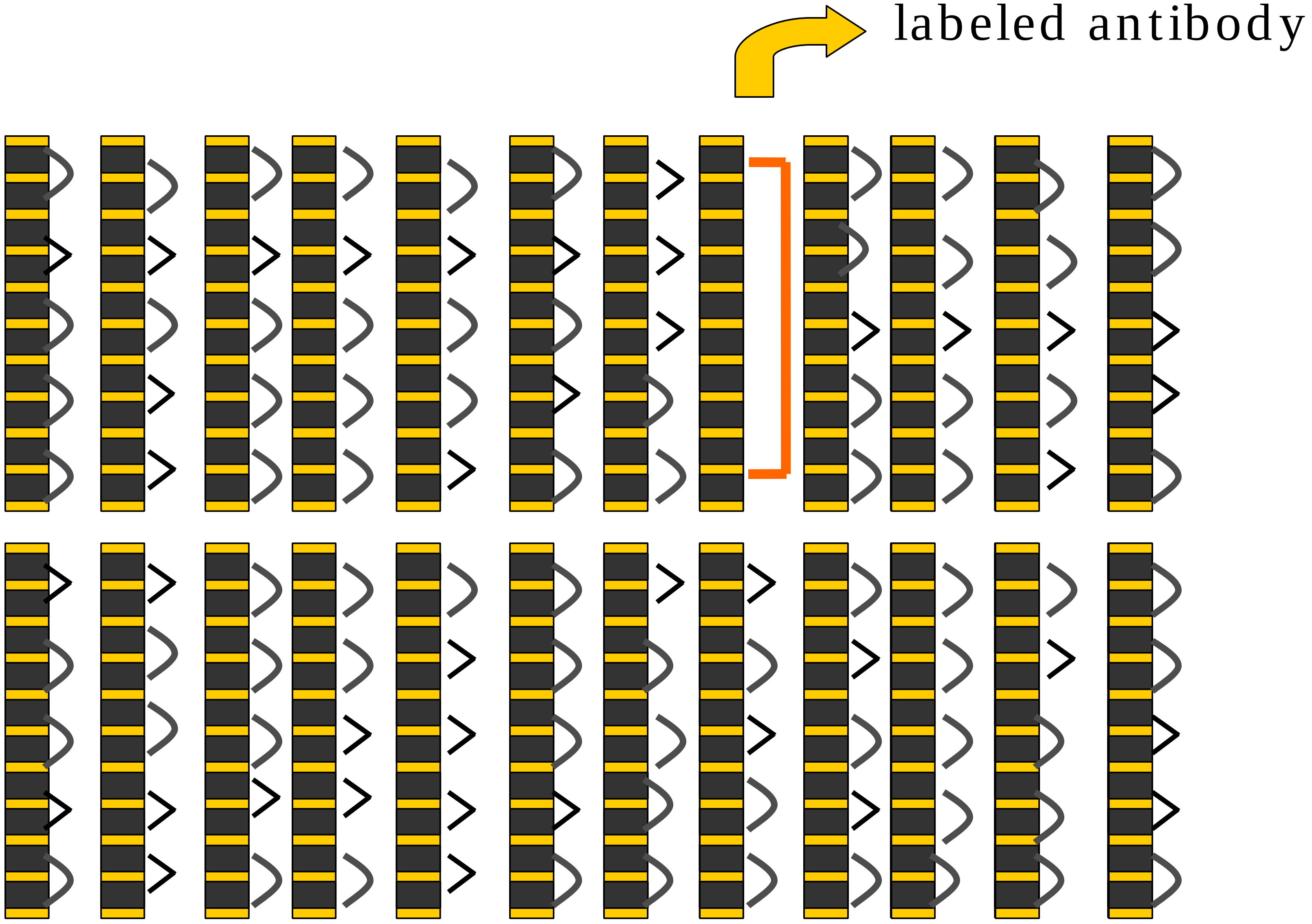
1. Take all the peptides in an aqueous solution
2. Add antibodies A_{ij}
3. Add antibodies B_{ij}
4. Add labeled antibody C
5. If fluorescence is detected answer is *yes* or else the answer is *no*

Peptides with Antibodies



Peptide with Antibodies




labeled antibody

Complexity

- Number of peptides = $(n-2)!$
- Length of peptides = $O(n)$
- Number of antibodies = $O(n^2)$
- Number of Bio-steps is *constant*

Exact Cover by 3-Sets Problem

- **Instance** A finite set $X = \{x_1, x_2, \ldots, x_n\}$, $n = 3q$ and a collection C of *3*-elements subsets of X
- **Question**: Does C contain an ***Exact Cover*** for X

Peptide Computing is Computationally Complete

A Turing Machine can be

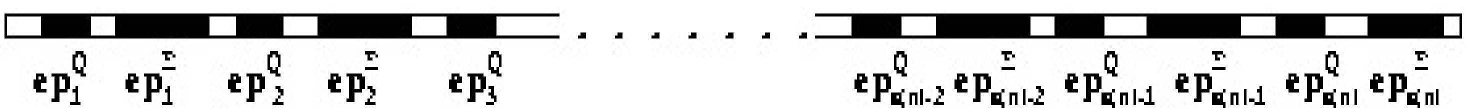
simulated by a Peptide System

Universality Result

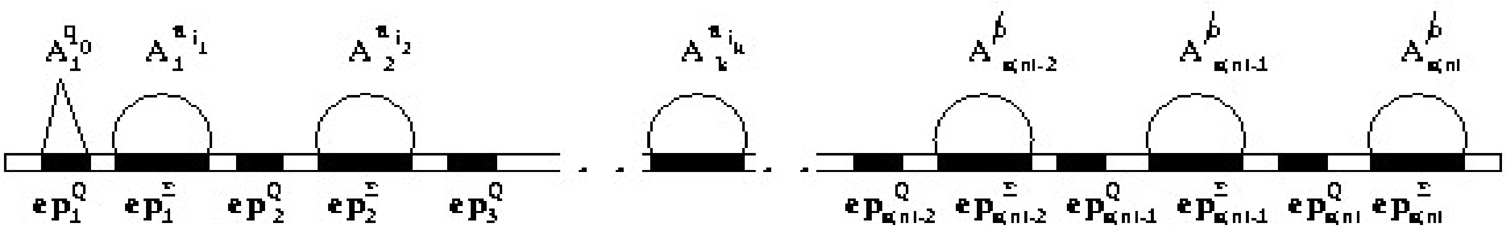
- Turing machine, $M = (Q, \Sigma, \delta, s_0, F)$
- $Q = \{q_1, q_2, \ldots, q_m\}$
- $\Sigma = \{a_1, a_2, \ldots, a_l\}$
- B is the blank symbol

Universality Result Contd..

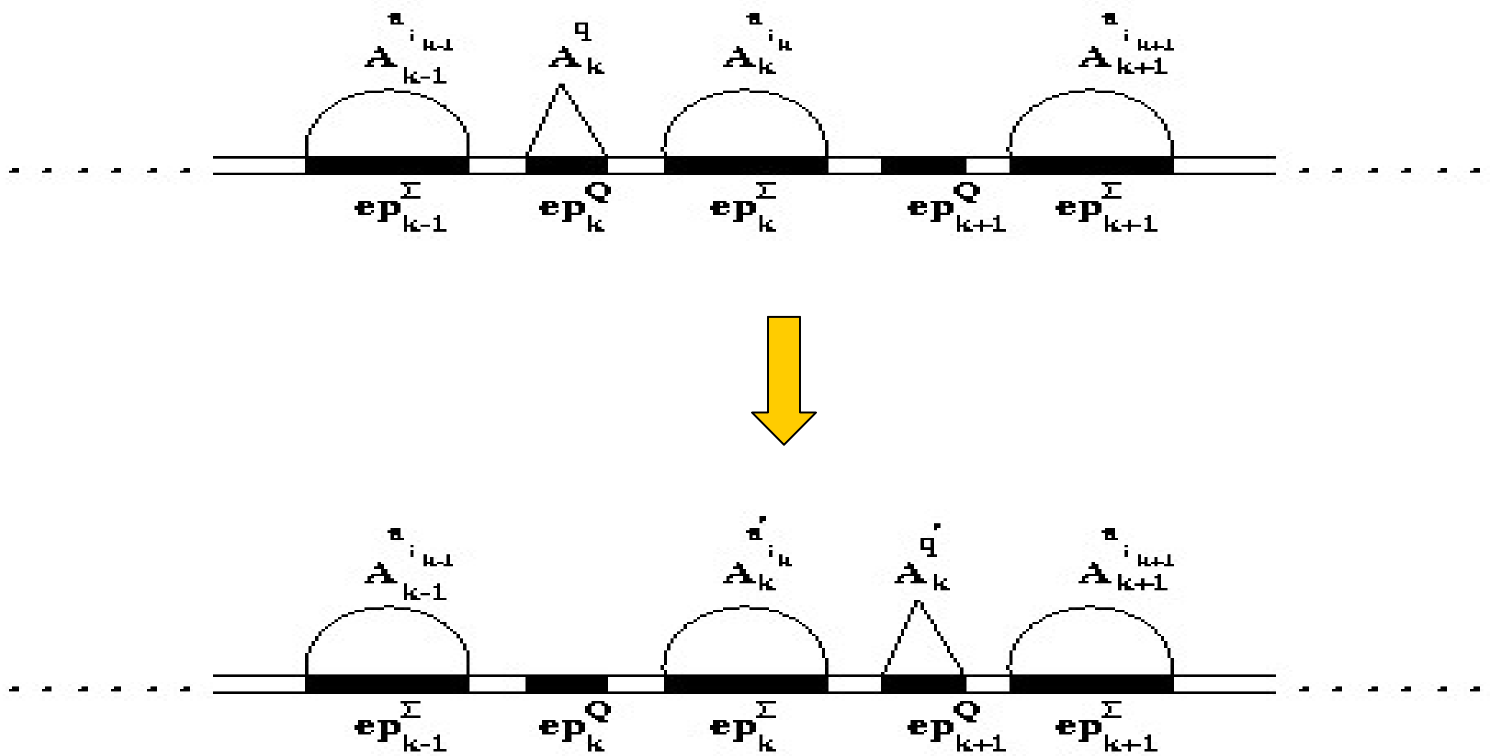
Peptide without antibodies



Initial Configuration of Peptide



Simulating the Right Move Contd..



Complexity

- Peptide system takes $O(t(n))$ time
- Length of the peptide is $O(s(n))$
- Amount of antibodies is

$$O(m.s(n)+l.(s(n))$$

Switching Operations - OR, AND & NOT gates

Why modeling gate operations?

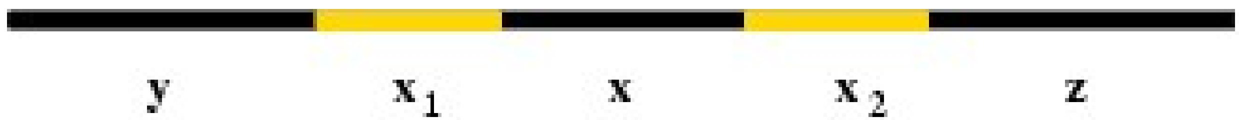
- Peptide and antibody formation is dependent on the problem.
- Defining basic operations makes the model independent of the problem.

Proposed Model

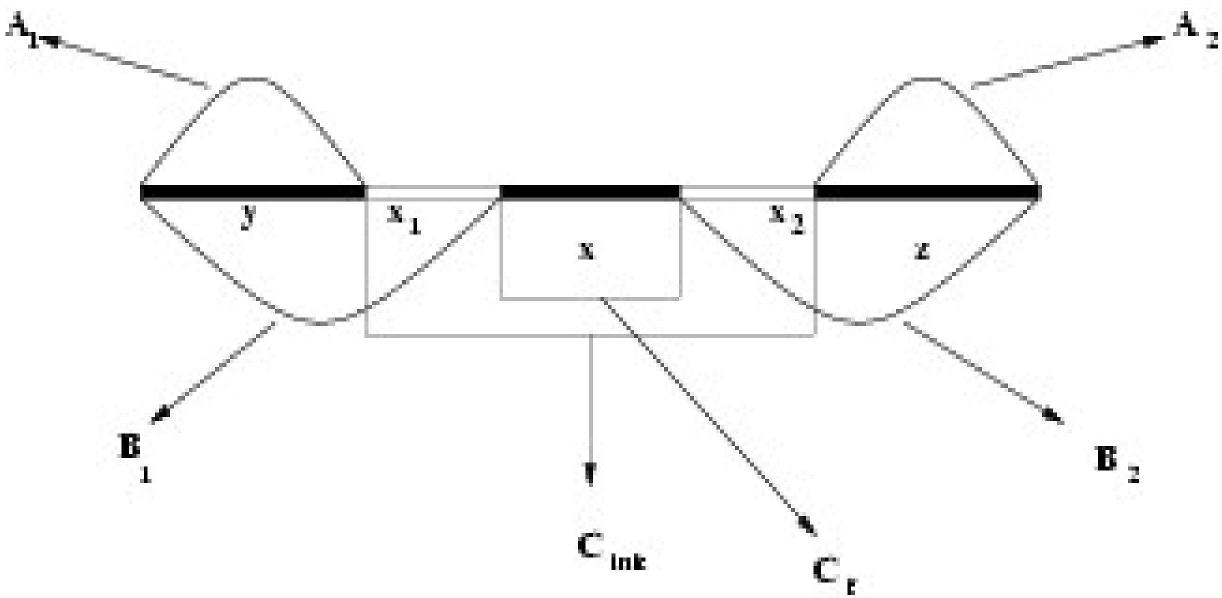
- Consists of a peptide sequence and some set of antibodies.
- Peptide sequence consists of five epitopes,
- Six antibodies denoted by A_1, A_2, B_1, B_2, C_{init} and C_f.
- Antibodies A_1, A_2, B_1 and B_2 denote the inputs.
- C_{init} denote the initial value of the result of the operation.
- C_{init} and C_f denote the output of the operation.

Proposed Model

- Epitopes for the antibodies denoting the input to bind.
- Epitope *for* the antibody representing the initial output to bind.
- Epitopes for the antibodies denoting the output to bind.



Peptide sequence



Peptide Sequence with Antibodies

OR Gate

- 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), $1 \leq i \leq 2$.

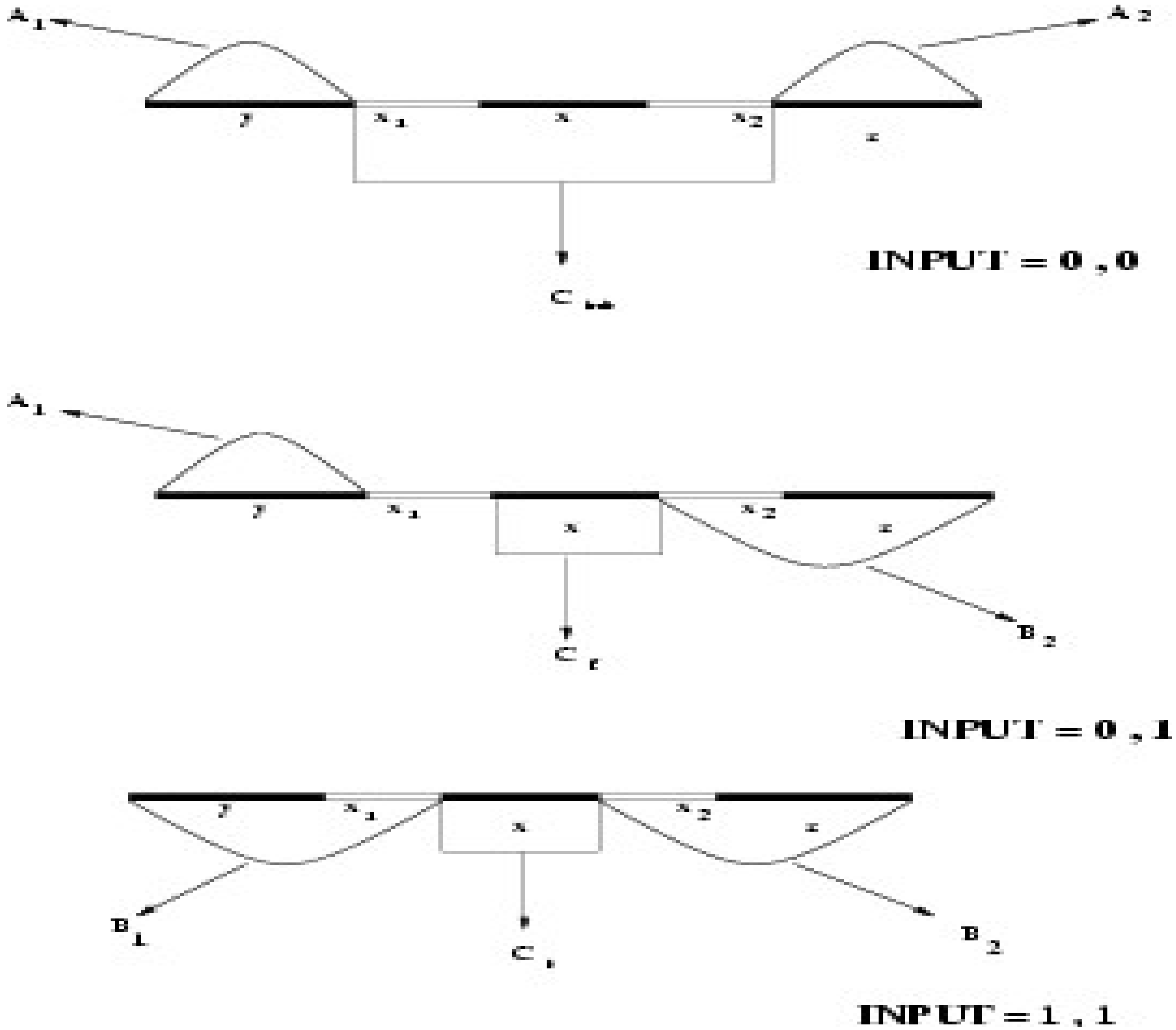
OR Gate

- 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, $1 \le i \le 2$ are taken as overlapping ones.
- aff(B_i) > aff(C_{init}) > aff(C_f), $1 \le i \le 2$.
- This facilitates toggle of output bit to *1* - antibody C_f binds to its epitope.

Algorithm

- Take the peptide sequence *P* in an aqueous solution.
- Add the antibody *C*.
- Add antibodies corresponding to the input bits. For example if the first bit is *1* and the second bit is *0* then add antibodies B_1 and A_2.
- Add antibody C_f.

If the output has to be seen the antibody C_f can be gives 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*.


A 1
A 2
y
x 1
x
x 2
z
INPUT = 0 , 0
A 1
B 2
INPUT = 0 , 1
B 1
B 2
INPUT = 1 , 1

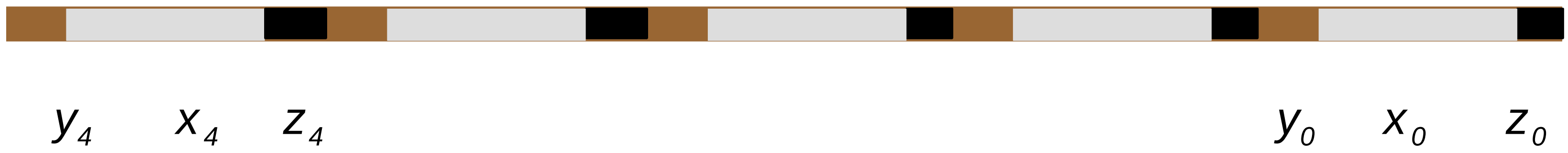
Other Gates

- This model has been extended to other gates – *AND, NOT, NAND, NOR* and *XOR*
- We also extend this model to simulate Boolean circuits.

Modeling Arithmetic Operations

Arithmetic Operations Proposed Model

- Consists of a peptide and set of antibodies
- Peptide sequence has *n* position specific epitopes
- Epitopes $ep_i = y_i x_i z_i$, y_i and z_i are *switching epitopes* for the i^{th} bit.



Peptide Sequence for a 5-bit number

Antibodies

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

Binding Sites

For A_i: $\quad A_i \rightarrow y_i x_i, \qquad A_i \rightarrow x_i$

$$y_i x_i > x_i$$

For B_i: $\quad B_i \rightarrow x_i z_i, \qquad B_i \rightarrow x_i$

$$x_i z_i > x_i$$

$$T_{ABi} \longrightarrow Z_i \qquad T_{BAi} \longrightarrow y_i$$

Affinity

- $aff(T_{ABi}) > aff(A_i)$
- $aff(T_{BAi}) > aff(B_i)$
- $aff(T_{ABi}) = aff(T_{BAi})$

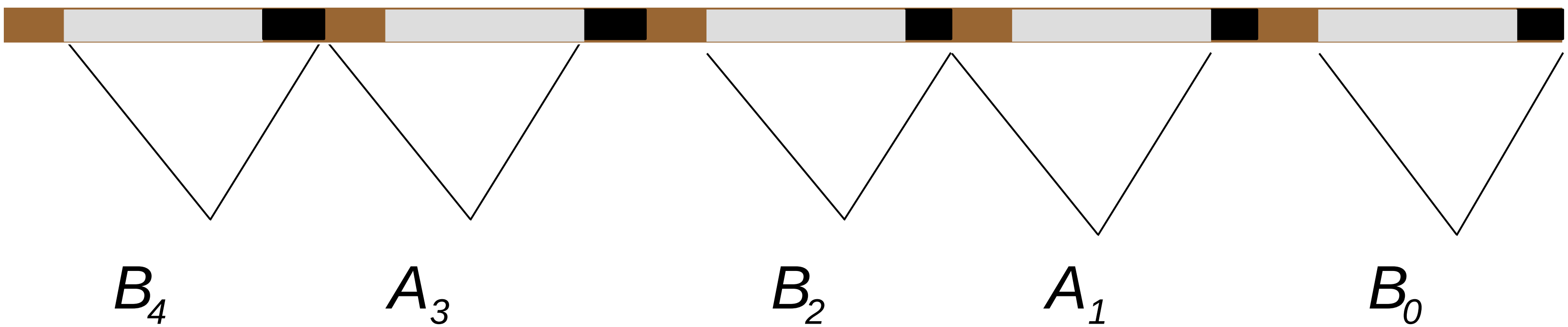
What it denotes?

- A_i – denotes i^{th} bit is zero
- B_i – denotes i^{th} bit is one
- T_{ABi} – used to switch i^{th} bit from zero to one
- T_{BAi} – used to switch i^{th} bit from one to zero

Representation of Binary Numbers

- If the i^{th} bit is 0 then the antibody A_i is bounded to the epitope $y_i x_i$
- If the i^{th} bit is 1 then the antibody B_i is bounded to the epitope $x_i z_i$

Example



10101

Addition of Two Binary Numbers

$$A = a_{n-1} a_{n-2} \ldots a_0 \qquad B = b_{n-1} b_{n-2} \ldots b_0$$

$$C = c_n c_{n-1} c_{n-2} \ldots c_0$$

XOR

	a_i	b_i	C_i
1	*0*	*0*	*0*
2	*0*	*1*	*1*
3	*1*	*0*	*1*
4	*1*	*1*	*0*

Addition (Contd..)

- First step – guessing equivalent to XOR gate.
- The bit c_n is initialized to zero.
- Carry propagation.

Addition (Contd..)

- Carry occurs only when both the bits a_i and b_i are *1*.
- Carry is propagated to the left until both the bits a_j and b_j $(j > i)$ are *0*.
- If no such *j* exists then propagation stops making n^{th} bit *1*.
- $j.j-1....i+1$ is called the carry block.
- For each carry block $j.j-1....i+1$ invert the digits c_k

  $(i+1 \leq k \leq j)$

Algorithm

1. Add antibodies A_i where $a_i = 0$ and $b_i = 0$ or $a_i = 1$ and $b_i = 1$.
2. Add antibodies B_i where $a_i = 0$ and $b_i = 1$ or $a_i = 1$ and $b_i = 0$.
3. For all carry block $j_k j_k - 1 \ldots i_k + 1$ 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}, and
   d) Add antibodies A_s.

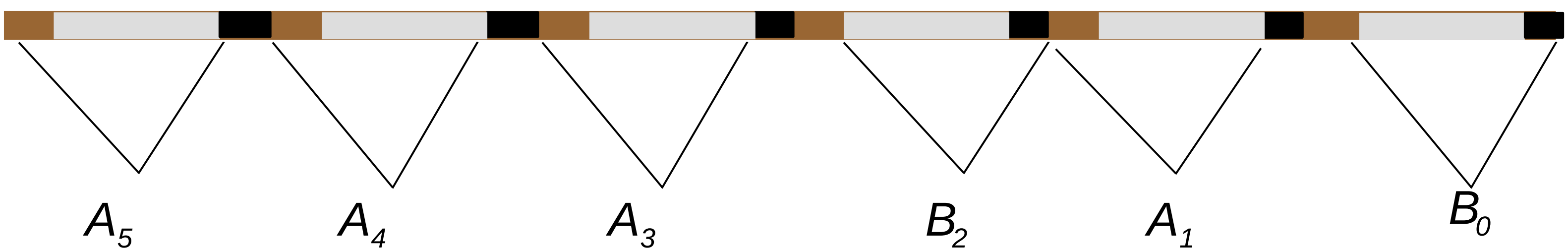
Example

1011
0

+

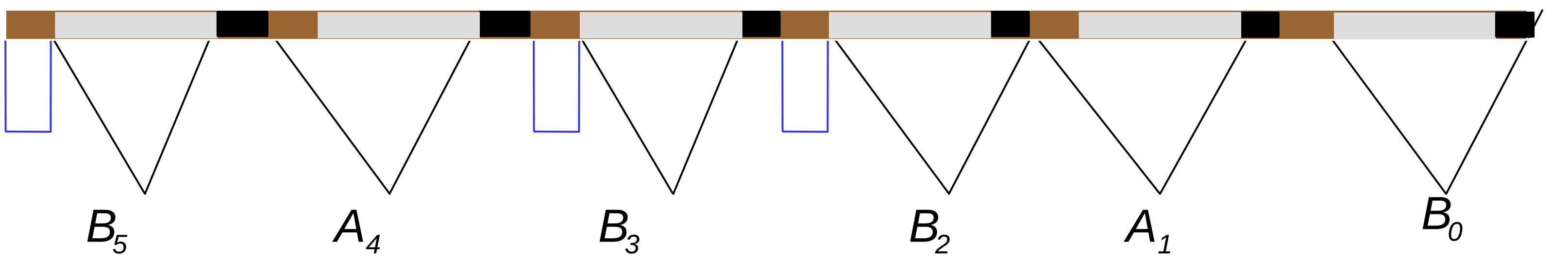
000101

1001
1



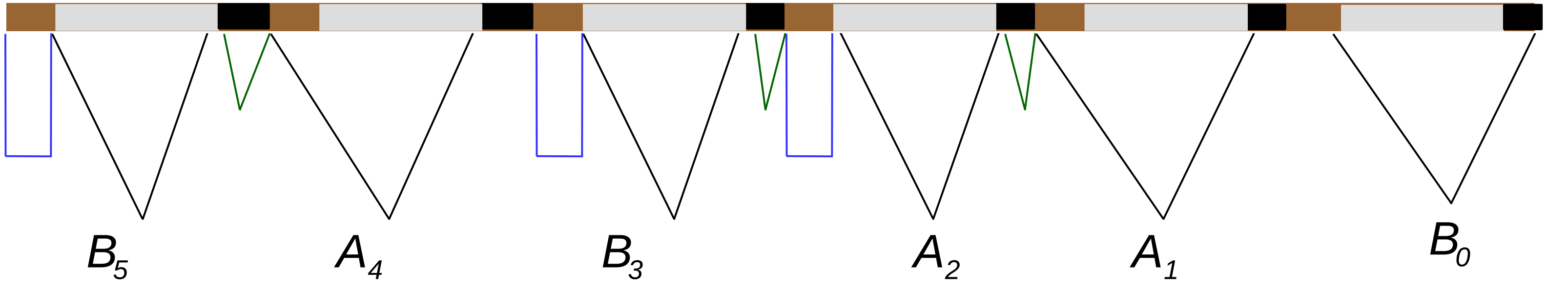
Example (Contd..)

101101



Example (Contd..)

101001



Algorithm

ADD(A,B,C)

1. XOR(A,B,C)
2. BlockInversion(I_1,I_2,...I_k,C) where I_j are carry blocks and k is the number of carry blocks.

Binding-Blocking Automata

Binding-Blocking Automata

- finite control
- finite tape
- tape head
- finite tape symbols
- transition function
- partial order relation
- blocking and unblocking functions

BBA Formal Definition

$$P = (Q, V, E, \delta, q_0, R, \beta_b, \beta_{ub}, Q_{accept}, Q_{reject})$$

where $Q = Q_{block} \cup Q_{unblock} \cup Q_{general}$,

$q_0 \in Q$ (start state),

V is a finite set of symbols,

E is the finite subset of V^*,

δ is the transition function from $Q \times E \rightarrow Q$,

$R \subseteq E \times E$ is the partial order relation,

β_b is the blocking function from $Q_{block} \rightarrow 2^V$,

β_{ub} is the unblocking function from $Q_{unblock} \rightarrow 2^V$,

$Q_{accept} \cup Q_{reject} \subseteq Q_{general}$ where Q_{accept} is the set of accepting states and Q_{reject} is the set of rejecting states.

BBA (Contd..)

- 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.

Four kinds of Transition

- *(l,b)*- transition
- *(l,f)*- transition
- *(ll,b)*- transition
- *(ll,f)*- transition
- *l* – leftmost, *ll* – locally leftmost,
- *b* – blocked, *f* - free

BBA - Results

- $BBA_{(I,b)} \subset BBA_{(I,f)}$
- $BBA_{(II,b)} \subseteq BBA_{(II,f)}$
- $BBA_{p,(I,x)} = BBA_{np,(I,x)}$, $x = b$ *or* f,
- $BBA_{(I,x)} = SBBA_{(I,x)}$, $x = b$ *or* f,
- For every language $L \in k\text{-}SNFA$ there is a language

  $L' \in BBA_{(II,b)}$ such that L can be written in the form

  $h^{-1}(L')$ where h is a homomorphism from L to L'.
- For every $L \in BBA_{(I,x)}$ there exists $BBA_{(II,x)}$ P such

  that $L(P) = L$ where $x \in \{b,f\}$.
- $BBA_{(I,b)} \subset BBA_{(II,b)}$.

Complexity Issues

- 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 defined as the maximum number of symbols blocked at any point of time.

Complexity Issues (Contd..)

- 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 defined as the maximum of blocking quotient over all the subsets of alphabet.
- P(*k*,*m*,*n*) denote a BBA P with *k* the blocking number, *m* the blocking instant and *n* blocking quotient.

Complexity Issues of BBA Results

- $REG \subseteq BBA_D(*,1,*) \subseteq BBA_D(*,2,*) \subseteq BBA_D(*,3,*) \subseteq$ … …

  where $D \in \{(ll,b),(ll,f),(l,b),(l,f)\}$.
- $REG \subseteq BBA_D(1,*,*) \subseteq BBA_D(2,*,*) \subseteq BBA_D(l,*,*) \subseteq$ … …
- $BBA_D(*,*,1) = BBA_D(*,*,k),\ k \leq 2,$

  $D \in \{(l,b),(l,f),(ll,b),(ll,f)\}$

String Binding-Blocking Automata

- String of symbols (starting from 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.
- Blocking is maximal.

Results in StrBBA

- The power of *StrBBA* in *I* transition is strictly more than *BBA* in *I* transition.
- The power of *StrBBA* in *II* transition is strictly more than languages not accepted by *BBA* in *II*.
- For every $L \in StrBBA_I$ there exists a random-context grammar *RC* with Context-free rules such that

  $L(RC) = L$.
- The set of all languages accepted by $strbba_I(Fin)$ is equal to the set of all regular languages.

Rewriting BBA

- A *2*- way tape head which scans a cell to its right at a time.
- 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 marker set.
- This poset relation, depending on the state, helps to replace the markers.

RBBA - Result

RBBA is Universally Complete

- 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

  - *p* denotes the system is in,
  - *q* states the previous state of the system,
  - *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 *q* then the state-affinity has the pair *(A,a)* which gives more affinity to *A* than *a*.

Summary

- Solved two NP-Complete problems using peptide-antibody interactions.
- Modeled switching operations and simple arithmetical operations using peptide-antibody interactions.
- Proposed an automata model – binding-blocking automata.
- Studied normal-forms and complexity measures for binding-blocking automata
- Extended the definition of binding-blocking to some more variants
  - string binding-blocking automata > BBA
  - rewriting binding-blocking automata = Turing Machine

Conclusion

- The peptide computational model has the potential to solve difficult combinatorial problems (Note: it hides pre-processing time)
- Will it be a complete computational model or a hybrid version with silicon computers or a model used to solve some subset of problems?
- Defining basic operations for the model.
- Solving hard problems using these basic operations.
- Preprocessing time – time for building peptides and antibodies is more.
- Implementation difficulties (cross-reactivity, structure of peptides)
- Can exhaustive search be replaced by other efficient methods (step-by-step elimination method)
- Turing machine simulation in which the transition is automatically taken over by the interactions between peptides and antibodies.
- Can we bound the number of antibodies in the Turning machine simulation.