Peptide Computing - Universality and Theoretical Model

M. Sakthi Balan[1]   Helmut Jürgensen[1,2]

[1]Department of Computer Science
University of Western Ontario
London, Ontario
Canada

[2]Institut für Informatik, Universität Potsdam,
August-Bebel-Str. 89, 14482 Potsdam, Germany

Unconventional Computation, 2006

Contents

Background

- Proposed by H. Hug et al.
- To solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Universal model.
  - Look-and-do method.
  - Unbounded numbers of peptides and antibodies.

Background

- Proposed by H. Hug et al.
- To solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Universal model.
  - Look-and-do method.
  - Unbounded numbers of peptides and antibodies.

Background

- Proposed by H. Hug et al.
- To solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Universal model.
  - Look-and-do method.
  - Unbounded numbers of peptides and antibodies.

Background

- Proposed by H. Hug et al.
- To solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Universal model.
  - Look-and-do method.
  - Unbounded numbers of peptides and antibodies.

Background

- Proposed by H. Hug et al.
- To solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Universal model.
  - Look-and-do method.
  - Unbounded numbers of peptides and antibodies.

Background

- Proposed by H. Hug et al.
- To solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem.
- Universal model.
  - Look-and-do method.
  - Unbounded numbers of peptides and antibodies.

Our Paper...

- Encoding of transitions of Turing machine that makes the simulation nearly automatic.
- Present a formal model of peptide computing to show the converse simulation under certain conditions.

Our Paper...

- Encoding of transitions of Turing machine that makes the simulation nearly automatic.
- Present a formal model of peptide computing to show the converse simulation under certain conditions.

Some Basic Definitions

- Peptides – sequence over 20 basic amino acids.
- Epitopes – binding sites in peptides which antibodies can recognize.
- Affinity – binding power of an antibody to a specific epitope.
- Affinity-based removal of antibodies.
- Epitope-based removal of antibodies.

Some Basic Definitions

- Peptides – sequence over 20 basic amino acids.
- Epitopes – binding sites in peptides which antibodies can recognize.
- Affinity – binding power of an antibody to a specific epitope.
- Affinity-based removal of antibodies.
- Epitope-based removal of antibodies.

Some Basic Definitions

- Peptides – sequence over 20 basic amino acids.
- Epitopes – binding sites in peptides which antibodies can recognize.
- Affinity – binding power of an antibody to a specific epitope.
- Affinity-based removal of antibodies.
- Epitope-based removal of antibodies.

Some Basic Definitions

- Peptides – sequence over 20 basic amino acids.
- Epitopes – binding sites in peptides which antibodies can recognize.
- Affinity – binding power of an antibody to a specific epitope.
- Affinity-based removal of antibodies.
- Epitope-based removal of antibodies.

Previous Simulation

Theorem

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F, \flat)$ be a Turing machine. There is a simulation of $\mathcal{M}$ by peptide computing with the following properties:

1. *There is a constant $c > 0$, independent of $\mathcal{M}$, such that the number of peptide antibody interactions needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is no greater than $c \cdot t_{\mathcal{M}}(w)$.*
2. *The length of the peptide sequence needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is in $\Theta(s_{\mathcal{M}}(w))$; moreover the number of antibodies needed is in $\Theta((|Q| + |\Sigma|) \cdot s_{\mathcal{M}}(w))$.*

Previous Simulation

Theorem

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F, \flat)$ be a Turing machine. There is a simulation of $\mathcal{M}$ by peptide computing with the following properties:

1. *There is a constant $c > 0$, independent of $\mathcal{M}$, such that the number of peptide antibody interactions needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is no greater than $c \cdot t_{\mathcal{M}}(w)$.*
2. *The length of the peptide sequence needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is in $\Theta(s_{\mathcal{M}}(w))$; moreover the number of antibodies needed is in $\Theta((|Q| + |\Sigma|) \cdot s_{\mathcal{M}}(w))$.*

Western

Previous Simulation

Theorem

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F, \flat)$ be a Turing machine. There is a simulation of $\mathcal{M}$ by peptide computing with the following properties:

1. *There is a constant $c > 0$, independent of $\mathcal{M}$, such that the number of peptide antibody interactions needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is no greater than $c \cdot t_{\mathcal{M}}(w)$.*
2. *The length of the peptide sequence needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is in $\Theta(s_{\mathcal{M}}(w))$; moreover the number of antibodies needed is in $\Theta\big((|Q| + |\Sigma|) \cdot s_{\mathcal{M}}(w)\big)$.*

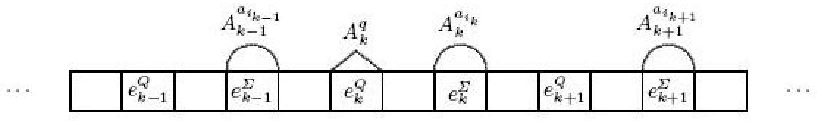

Figure: Peptide Sequence

Suppose the transition rule is $\delta(q, a_{i_k}) = \{(q', a'_{i_k}, R)\}$.

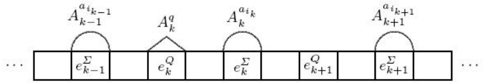

Figure: Before applying rule

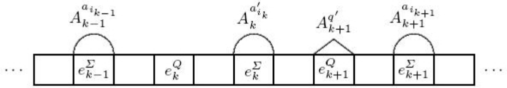

Figure: After applying rule

Analysis

- For each step of $\mathcal{M}$, there are two steps – removal of antibodies and adding of antibodies.
- The length of the peptide needed for simulating the computation of $\mathcal{M}$ on input w is $O(s_{\mathcal{M}}(w))$.
- The number of epitopes is $O(s_{\mathcal{M}}(w))$.
- The number of antibodies is $O((m + l) \cdot s_{\mathcal{M}}(w))$ where $m = |Q|$ and $l = |\Sigma|$.

Analysis

- For each step of $\mathcal{M}$, there are two steps – removal of antibodies and adding of antibodies.
- The length of the peptide needed for simulating the computation of $\mathcal{M}$ on input w is $O(s_{\mathcal{M}}(w))$.
- The number of epitopes is $O(s_{\mathcal{M}}(w))$.
- The number of antibodies is $O((m + l) \cdot s_{\mathcal{M}}(w))$ where $m = |Q|$ and $l = |\Sigma|$.

Analysis

- For each step of $\mathcal{M}$, there are two steps – removal of antibodies and adding of antibodies.
- The length of the peptide needed for simulating the computation of $\mathcal{M}$ on input w is $O(s_{\mathcal{M}}(w))$.
- The number of epitopes is $O(s_{\mathcal{M}}(w))$.
- The number of antibodies is $O((m+l) \cdot s_{\mathcal{M}}(w))$ where $m = |Q|$ and $l = |\Sigma|$.

Analysis

- For each step of $\mathcal{M}$, there are two steps – removal of antibodies and adding of antibodies.
- The length of the peptide needed for simulating the computation of $\mathcal{M}$ on input w is $O(s_{\mathcal{M}}(w))$.
- The number of epitopes is $O(s_{\mathcal{M}}(w))$.
- The number of antibodies is $O((m + l) \cdot s_{\mathcal{M}}(w))$ where $m = |Q|$ and $l = |\Sigma|$.

Analysis

- For each step of $\mathcal{M}$, there are two steps – removal of antibodies and adding of antibodies.
- The length of the peptide needed for simulating the computation of $\mathcal{M}$ on input w is $O(s_{\mathcal{M}}(w))$.
- The number of epitopes is $O(s_{\mathcal{M}}(w))$.
- The number of antibodies is $O((m + l) \cdot s_{\mathcal{M}}(w))$ where $m = |Q|$ and $l = |\Sigma|$.

Drawback

- Need for an extraneous computing agents for each step of the simulation.
  - Usually hidden in the definition of computational steps of any formal model.
  - How to limit the “power” of this agent.
- The size of the alphabets is unbounded.
  - Encoding of antibodies and epitopes over a finite alphabet increases resource and time requirements.
  - Theoretically possible; but, bio-chemically?

Drawback

- Need for an extraneous computing agents for each step of the simulation.
  - Usually hidden in the definition of computational steps of any formal model.
  - How to limit the “power” of this agent.
- The size of the alphabets is unbounded.
  - Encoding of antibodies and epitopes over a finite alphabet increases resource and time requirements.
  - Theoretically possible; but, bio-chemically?

Drawback

- Need for an extraneous computing agents for each step of the simulation.
  - Usually hidden in the definition of computational steps of any formal model.
  - How to limit the “power” of this agent.
- The size of the alphabets is unbounded.
  - Encoding of antibodies and epitopes over a finite alphabet increases resource and time requirements.
  - Theoretically possible; but, bio-chemically?

Drawback

- Need for an extraneous computing agents for each step of the simulation.
  - Usually hidden in the definition of computational steps of any formal model.
  - How to limit the “power” of this agent.
- The size of the alphabets is unbounded.
  - Encoding of antibodies and epitopes over a finite alphabet increases resource and time requirements.
  - Theoretically possible; but, bio-chemically?

Nearly automated simulation

Theorem

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F, \flat)$ be a Turing machine. There is a simulation of $\mathcal{M}$ by peptide computing with the following properties:

1. *There is a constant $c > 0$, independent of $\mathcal{M}$, such that the number of peptide antibody interactions needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is no greater than $c \cdot t_{\mathcal{M}}(w)$.*
2. *The number of the peptide sequence needed for the simulation of a computation of $\mathcal{M}$ on input $w \in \Sigma^*$ is in $\Theta(s_{\mathcal{M}}(w))$; moreover the number of antibodies needed is in $\Theta((|Q| + |\Sigma|) \cdot s_{\mathcal{M}}(w))$.*

Proof Idea

- Assume that $\mathcal{M}$ has only a single final state.
- We use five multi-sets of peptide sequences:
  1. T to simulate the cells of the tape of $\mathcal{M}$;
  2. P to hold the program of $\mathcal{M}$;
  3. S to synchronize the operation; and
  4. I_1 and I_2 for carrying out intermediate steps.

Proof Idea

- Assume that $\mathcal{M}$ has only a single final state.
- We use five multi-sets of peptide sequences:
    1. T to simulate the cells of the tape of $\mathcal{M}$;
    2. P to hold the program of $\mathcal{M}$;
    3. S to synchronize the operation; and
    4. I_1 and I_2 for carrying out intermediate steps.

Proof Idea

- Assume that $\mathcal{M}$ has only a single final state.
- We use five multi-sets of peptide sequences:
    1. T to simulate the cells of the tape of $\mathcal{M}$;
    2. P to hold the program of $\mathcal{M}$;
    3. S to synchronize the operation; and
    4. I_1 and I_2 for carrying out intermediate steps.

Proof Idea

- Assume that $\mathcal{M}$ has only a single final state.
- We use five multi-sets of peptide sequences:
  1. T to simulate the cells of the tape of $\mathcal{M}$;
  2. P to hold the program of $\mathcal{M}$;
  3. S to synchronize the operation; and
  4. I_1 and I_2 for carrying out intermediate steps.

Proof Idea

- Assume that $\mathcal{M}$ has only a single final state.
- We use five multi-sets of peptide sequences:
  1. T to simulate the cells of the tape of $\mathcal{M}$;
  2. P to hold the program of $\mathcal{M}$;
  3. S to synchronize the operation; and
  4. I_1 and I_2 for carrying out intermediate steps.

Proof Idea

- Assume that $\mathcal{M}$ has only a single final state.
- We use five multi-sets of peptide sequences:
  1. T to simulate the cells of the tape of $\mathcal{M}$;
  2. P to hold the program of $\mathcal{M}$;
  3. S to synchronize the operation; and
  4. I_1 and I_2 for carrying out intermediate steps.

Proof Idea

- Each sequence in T consists of six epitopes.
- Uniquely denotes a cell on the tape of $\mathcal{M}$.
- Peptide sequence is represented by $p_i^{(T)} = e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$ with $e_{i,4}^{(T)} = x_i e_{i,2}^{(T)} y_i$ for some words x_i and y_i, where the epitopes are $e_{i,1}^{(T)}, \ldots, e_{i,4}^{(T)}, e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i$ and $x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$.

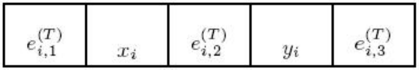

Figure: Cell i

Proof Idea

- Each sequence in T consists of six epitopes.
- Uniquely denotes a cell on the tape of $\mathcal{M}$.
- Peptide sequence is represented by $p_i^{(T)} = e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$ with $e_{i,4}^{(T)} = x_i e_{i,2}^{(T)} y_i$ for some words x_i and y_i, where the epitopes are $e_{i,1}^{(T)}, \ldots, e_{i,4}^{(T)}, e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i$ and $x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$.

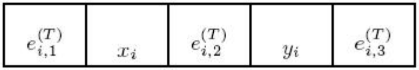

Figure: Cell i

Proof Idea

- Each sequence in T consists of six epitopes.
- Uniquely denotes a cell on the tape of $\mathcal{M}$.
- Peptide sequence is represented by $p_i^{(T)} = e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$ with $e_{i,4}^{(T)} = x_i e_{i,2}^{(T)} y_i$ for some words x_i and y_i, where the epitopes are $e_{i,1}^{(T)}, \ldots, e_{i,4}^{(T)}, e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i$ and $x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$.

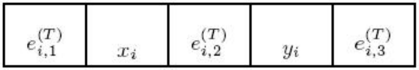

Figure: Cell i

Proof Idea

- Each sequence in T consists of six epitopes.
- Uniquely denotes a cell on the tape of $\mathcal{M}$.
- Peptide sequence is represented by $p_i^{(T)} = e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$ with $e_{i,4}^{(T)} = x_i e_{i,2}^{(T)} y_i$ for some words x_i and y_i, where the epitopes are $e_{i,1}^{(T)}, \ldots, e_{i,4}^{(T)}, e_{i,1}^{(T)} x_i e_{i,2}^{(T)} y_i$ and $x_i e_{i,2}^{(T)} y_i e_{i,3}^{(T)}$.

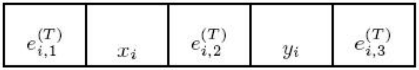

Figure: Cell i

Proof Idea

- The set P contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will capture the transition applied when $\mathcal{M}$ is in state q and reading the symbol a.
- Has three epitopes $e^{(P)}_{(q,a),1}$, $e^{(P)}_{(q,a),2}$ and $e^{(P)}_{(q,a),3}$.

Proof Idea

- The set P contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will capture the transition applied when $\mathcal{M}$ is in state q and reading the symbol a.
- Has three epitopes $e^{(P)}_{(q,a),1}$, $e^{(P)}_{(q,a),2}$ and $e^{(P)}_{(q,a),3}$.

Proof Idea

- The set P contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will capture the transition applied when $\mathcal{M}$ is in state q and reading the symbol a.
- Has three epitopes $e^{(P)}_{(q,a),1}$, $e^{(P)}_{(q,a),2}$ and $e^{(P)}_{(q,a),3}$.

Proof Idea

- The set P contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will capture the transition applied when $\mathcal{M}$ is in state q and reading the symbol a.
- Has three epitopes $e^{(P)}_{(q,a),1}$, $e^{(P)}_{(q,a),2}$ and $e^{(P)}_{(q,a),3}$.

Proof Idea

- Has the form $p^{(P)}_{(q,a)} = e^{(P)}_{(q,a),1} e^{(P)}_{(q,a),2}$ with $e^{(P)}_{(q,a),3} \in \mathrm{Inf}_{+}(p^{(P)}_{(q,a)})$ and which overlaps both $e^{(P)}_{(q,a),1}$ and $e^{(P)}_{(q,a),2}$.

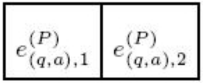

Figure: Peptide sequence in P

Proof Idea

- Has the form $p^{(P)}_{(q,a)} = e^{(P)}_{(q,a),1} e^{(P)}_{(q,a),2}$ with $e^{(P)}_{(q,a),3} \in \mathrm{Inf}_+(p^{(P)}_{(q,a)})$ and which overlaps both $e^{(P)}_{(q,a),1}$ and $e^{(P)}_{(q,a),2}$.

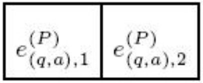

Figure: Peptide sequence in P

Proof Idea

- The set S contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will control the execution of a transition step.
- Has the form $p^{(S)}_{(q,a)} = z_{(q,a)} e^{(S)}_{(q,a),1} e^{(S)}_{(q,a),2}$.
- Has the three epitopes $e^{(S)}_{(q,a),1}$, $e^{(S)}_{(q,a),2}$ and the whole sequence itself is an epitope.

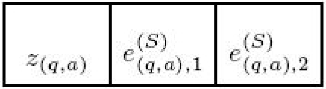

Figure: Peptide sequence in S

Proof Idea

- The set S contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will control the execution of a transition step.
- Has the form $p^{(S)}_{(q,a)} = z_{(q,a)} e^{(S)}_{(q,a),1} e^{(S)}_{(q,a),2}$.
- Has the three epitopes $e^{(S)}_{(q,a),1}$, $e^{(S)}_{(q,a),2}$ and the whole sequence itself is an epitope.

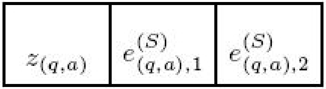

Figure: Peptide sequence in S

Proof Idea

- The set S contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will control the execution of a transition step.
- Has the form $p^{(S)}_{(q,a)} = z_{(q,a)} e^{(S)}_{(q,a),1} e^{(S)}_{(q,a),2}$.
- Has the three epitopes $e^{(S)}_{(q,a),1}$, $e^{(S)}_{(q,a),2}$ and the whole sequence itself is an epitope.

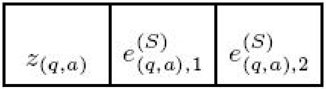

Figure: Peptide sequence in S

Proof Idea

- The set S contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will control the execution of a transition step.
- Has the form $p^{(S)}_{(q,a)} = z_{(q,a)} e^{(S)}_{(q,a),1} e^{(S)}_{(q,a),2}$.
- Has the three epitopes $e^{(S)}_{(q,a),1}$, $e^{(S)}_{(q,a),2}$ and the whole sequence itself is an epitope.

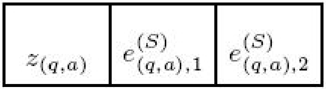

Figure: Peptide sequence in S

Proof Idea

- The set S contains a peptide sequence for each pair $(q, a) \in Q \times \Sigma$.
- Will control the execution of a transition step.
- Has the form $p^{(S)}_{(q,a)} = z_{(q,a)} e^{(S)}_{(q,a),1} e^{(S)}_{(q,a),2}$.
- Has the three epitopes $e^{(S)}_{(q,a),1}$, $e^{(S)}_{(q,a),2}$ and the whole sequence itself is an epitope.

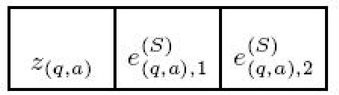

Figure: Peptide sequence in S

Proof Idea

The sets I_1 and I_2 contain peptide sequence as follows:

- Each sequence in I_1 contains epitopes $e^{(I_1)}_{(q,a),1}$ and $e^{(I_1)}_{(q,a),2}$.
- It is represented by $p^{(I_1)}_{(q,a)} = e^{(I_1)}_{(q,a),1} e^{(I_1)}_{(q,a),2}$.
- All the peptide sequences in I_1 are initialized with antibodies $A_{q,a}$ which binds to the epitope $e^{(I_1)}_{(q,a),1}$.
- Each sequence in the set I_2 is represented by $p^{(I_2)}_{(q,a)} = e^{(I_2)}_{(q,a)}$.

Proof Idea

The sets I_1 and I_2 contain peptide sequence as follows:

- Each sequence in I_1 contains epitopes $e^{(I_1)}_{(q,a),1}$ and $e^{(I_1)}_{(q,a),2}$.
- It is represented by $p^{(I_1)}_{(q,a)} = e^{(I_1)}_{(q,a),1} e^{(I_1)}_{(q,a),2}$.
- All the peptide sequences in I_1 are initialized with antibodies $A_{q,a}$ which binds to the epitope $e^{(I_1)}_{(q,a),1}$.
- Each sequence in the set I_2 is represented by $p^{(I_2)}_{(q,a)} = e^{(I_2)}_{(q,a)}$.

Proof Idea

The sets I_1 and I_2 contain peptide sequence as follows:

- Each sequence in I_1 contains epitopes $e^{(I_1)}_{(q,a),1}$ and $e^{(I_1)}_{(q,a),2}$.
- It is represented by $p^{(I_1)}_{(q,a)} = e^{(I_1)}_{(q,a),1} e^{(I_1)}_{(q,a),2}$.
- All the peptide sequences in I_1 are initialized with antibodies $A_{q,a}$ which binds to the epitope $e^{(I_1)}_{(q,a),1}$.
- Each sequence in the set I_2 is represented by $p^{(I_2)}_{(q,a)} = e^{(I_2)}_{(q,a)}$.

Proof Idea

The sets I_1 and I_2 contain peptide sequence as follows:

- Each sequence in I_1 contains epitopes $e^{(I_1)}_{(q,a),1}$ and $e^{(I_1)}_{(q,a),2}$.
- It is represented by $p^{(I_1)}_{(q,a)} = e^{(I_1)}_{(q,a),1} e^{(I_1)}_{(q,a),2}$.
- All the peptide sequences in I_1 are initialized with antibodies $A_{q,a}$ which binds to the epitope $e^{(I_1)}_{(q,a),1}$.
- Each sequence in the set I_2 is represented by $p^{(I_2)}_{(q,a)} = e^{(I_2)}_{(q,a)}$.

Proof Idea

The sets I_1 and I_2 contain peptide sequence as follows:

- Each sequence in I_1 contains epitopes $e^{(I_1)}_{(q,a),1}$ and $e^{(I_1)}_{(q,a),2}$.
- It is represented by $p^{(I_1)}_{(q,a)} = e^{(I_1)}_{(q,a),1} e^{(I_1)}_{(q,a),2}$.
- All the peptide sequences in I_1 are initialized with antibodies $A_{q,a}$ which binds to the epitope $e^{(I_1)}_{(q,a),1}$.
- Each sequence in the set I_2 is represented by $p^{(I_2)}_{(q,a)} = e^{(I_2)}_{(q,a)}$.

Proof Idea

Encoding of transition function δ of $\mathcal{M}$: suppose $\delta(q, a) = (q', a', \mathrm{D})$ for $\mathrm{D} \in \{\mathrm{L}, \mathrm{R}\}$.

- We have a peptide sequence $p^{(P)}_{(q,a)}$ in P with antibodies $A_{q'}$ and $A_{a',\mathrm{D}}$ attached to it at epitopes $e^{(P)}_{(q,a),1}$ and $e^{(P)}_{(q,a),2}$, respectively.
- Each sequence in P encodes the transition for state q and symbol a.
- The antibodies $A_{q'}$ and $A_{a',\mathrm{D}}$ need to be ‘read,’ that is, removed, to execute the transition.
- If $q' \in F$ then the antibody $A_{q'}$ will be a labelled one.

Proof Idea

Encoding of transition function δ of $\mathcal{M}$: suppose $\delta(q, a) = (q', a', \mathtt{D})$ for $\mathtt{D} \in \{\mathtt{L}, \mathtt{R}\}$.

- We have a peptide sequence $p^{(P)}_{(q,a)}$ in P with antibodies $A_{q'}$ and $A_{a',\mathtt{D}}$ attached to it at epitopes $e^{(P)}_{(q,a),1}$ and $e^{(P)}_{(q,a),2}$, respectively.
- Each sequence in P encodes the transition for state q and symbol a.
- The antibodies $A_{q'}$ and $A_{a',\mathtt{D}}$ need to be ‘read,’ that is, removed, to execute the transition.
- If $q' \in F$ then the antibody $A_{q'}$ will be a labelled one.

Proof Idea

Encoding of transition function δ of $\mathcal{M}$: suppose $\delta(q, a) = (q', a', \mathtt{D})$ for $\mathtt{D} \in \{\mathtt{L}, \mathtt{R}\}$.

- We have a peptide sequence $p^{(P)}_{(q,a)}$ in P with antibodies $A_{q'}$ and $A_{a',\mathtt{D}}$ attached to it at epitopes $e^{(P)}_{(q,a),1}$ and $e^{(P)}_{(q,a),2}$, respectively.
- Each sequence in P encodes the transition for state q and symbol a.
- The antibodies $A_{q'}$ and $A_{a',\mathtt{D}}$ need to be 'read,' that is, removed, to execute the transition.
- If $q' \in F$ then the antibody $A_{q'}$ will be a labelled one.

Proof Idea

Encoding of transition function δ of $\mathcal{M}$: suppose $\delta(q, a) = (q', a', \mathtt{D})$ for $\mathtt{D} \in \{\mathtt{L}, \mathtt{R}\}$.

- We have a peptide sequence $p^{(P)}_{(q,a)}$ in P with antibodies $A_{q'}$ and $A_{a',\mathtt{D}}$ attached to it at epitopes $e^{(P)}_{(q,a),1}$ and $e^{(P)}_{(q,a),2}$, respectively.
- Each sequence in P encodes the transition for state q and symbol a.
- The antibodies $A_{q'}$ and $A_{a',\mathtt{D}}$ need to be 'read,' that is, removed, to execute the transition.
- If $q' \in F$ then the antibody $A_{q'}$ will be a labelled one.

Proof Idea

Encoding of a configuration of $\mathcal{M}$:

- $p_i^{(T)}$ has A_{i-1}, A_a (or $A_{a,\mathbb{D}}$) and A_{i+1} attached to its epitopes $e_{i,1}^{(T)}$, $e_{i,4}^{(T)}$ and $e_{i,2}^{(T)}$, respectively.
- A_a or $A_{a,\mathbb{D}}$ denotes the content of the cell i.
- A_{i+1} and A_{i-1} are for initiating the right and left movement of $\mathcal{M}$.

Proof Idea

Encoding of a configuration of $\mathcal{M}$:

- $p_i^{(T)}$ has A_{i-1}, A_a (or $A_{a,\mathrm{D}}$) and A_{i+1} attached to its epitopes $e_{i,1}^{(T)}$, $e_{i,4}^{(T)}$ and $e_{i,2}^{(T)}$, respectively.
- A_a or $A_{a,\mathrm{D}}$ denotes the content of the cell i.
- A_{i+1} and A_{i-1} are for initiating the right and left movement of $\mathcal{M}$.

Proof Idea

Encoding of a configuration of $\mathcal{M}$:

- $p_i^{(T)}$ has A_{i-1}, A_a (or $A_{a,\mathbb{D}}$) and A_{i+1} attached to its epitopes $e_{i,1}^{(T)}$, $e_{i,4}^{(T)}$ and $e_{i,2}^{(T)}$, respectively.
- A_a or $A_{a,\mathbb{D}}$ denotes the content of the cell i.
- A_{i+1} and A_{i-1} are for initiating the right and left movement of $\mathcal{M}$.

Proof Idea

Encoding of a configuration of $\mathcal{M}$:

- $p_i^{(T)}$ has A_{i-1}, A_a (or $A_{a,\mathbb{D}}$) and A_{i+1} attached to its epitopes $e_{i,1}^{(T)}$, $e_{i,4}^{(T)}$ and $e_{i,2}^{(T)}$, respectively.
- A_a or $A_{a,\mathbb{D}}$ denotes the content of the cell i.
- A_{i+1} and A_{i-1} are for initiating the right and left movement of $\mathcal{M}$.

Proof Idea

- We assume that peptide sequences for enough cells are available to conduct the computation.
- The cells not occupied by input symbols are initialized to $A_\flat$.

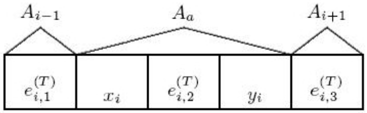

Figure: Cell i with Antibodies

Proof Idea

- We assume that peptide sequences for enough cells are available to conduct the computation.
- The cells not occupied by input symbols are initialized to $A_\flat$.

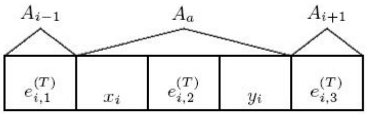

Figure: Cell i with Antibodies

Proof Idea

Simulation of a step of $\mathcal{M}$:

- Each such step consists of a cycle of reactions.
- Initiated by antibodies A_q, denoting the current state, and antibodies A_i denoting the position of the head.
- The computation is started by adding antibodies A_{q_0} and A_1 corresponding to the initial state q_0 and the first cell respectively.

Proof Idea

Simulation of a step of $\mathcal{M}$:

- Each such step consists of a cycle of reactions.
- Initiated by antibodies A_q, denoting the current state, and antibodies A_i denoting the position of the head.
- The computation is started by adding antibodies A_{q_0} and A_1 corresponding to the initial state q_0 and the first cell respectively.

Proof Idea

Simulation of a step of $\mathcal{M}$:

- Each such step consists of a cycle of reactions.
- Initiated by antibodies A_q, denoting the current state, and antibodies A_i denoting the position of the head.
- The computation is started by adding antibodies A_{q_0} and A_1 corresponding to the initial state q_0 and the first cell respectively.

Proof Idea

Simulation of a step of $\mathcal{M}$:

- Each such step consists of a cycle of reactions.
- Initiated by antibodies A_q, denoting the current state, and antibodies A_i denoting the position of the head.
- The computation is started by adding antibodies A_{q_0} and A_1 corresponding to the initial state q_0 and the first cell respectively.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathrm{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathrm{D}}}$ from P.
- $A_{\bar{a},\bar{\mathrm{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathrm{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathrm{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathrm{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathrm{D}}}$ from P.
- $A_{\bar{a},\bar{\mathrm{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathrm{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathrm{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathrm{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathrm{D}}}$ from P.
- $A_{\bar{a},\bar{\mathrm{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathrm{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathrm{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathrm{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathrm{D}}}$ from P.
- $A_{\bar{a},\bar{\mathrm{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathrm{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathrm{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathrm{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathrm{D}}}$ from P.
- $A_{\bar{a},\bar{\mathrm{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathrm{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathrm{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathrm{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathrm{D}}}$ from P.
- $A_{\bar{a},\bar{\mathrm{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathrm{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathrm{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathbb{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathbb{D}}}$ from P.
- $A_{\bar{a},\bar{\mathbb{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathbb{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathbb{D}'}$ attaches to S.
- System ready for next transition.

Proof Idea

Suppose the floating antibodies are A_q and A_i and antibody A_a is attached to $p_i^{(T)}$.
Let the transition is $\delta(q, a) = \{(\bar{q}, \bar{a}, \bar{\mathbb{D}})\}$

- A_i attaches to T and removes A_a.
- A_q and A_a are ready to choose the next transition.
- A_q and A_a through I_1 and I_2 chooses the antibody $A_{q,a}$. (important to discard any circular arguments)
- $A_{q,a}$ chooses the antibodies $A_{\bar{q}}$ and $A_{\bar{a},\bar{\mathbb{D}}}$ from P.
- $A_{\bar{a},\bar{\mathbb{D}}}$ attaches to T and removes A_{i+1} or A_{i-1}.
- A_{i+1} or A_{i-1} removes the next antibody A_b or $A_{b,\mathbb{D}'}$ representing the next symbol b.
- $A_{\bar{q}}$ and A_b or $A_{b,\mathbb{D}'}$ attaches to S.
- System ready for next transition.

About the Proof

- Requires an infinite number of antibodies which, however is recursively enumerable.
- We can consider antibodies as being encoded over a finite alphabet.
- To encode n symbols by a solid code the maximal code word length is in $\Theta(\log n)$.

Corollary

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F, \flat)$ be a Turing machine. There is a simulation of $\mathcal{M}$ by peptide computing with the following properties:

1. *Only a finite alphabet is required,*
2. *A step is simulated in $\Theta(\log s_{\mathcal{M}})$ steps.*

Why?

- Rigorous notion of a computation step.
- Capabilities and limitations of this computing paradigm.
- Computability implies peptide computability. Converse?
- If converse true, under what conditions?

Why?

- Rigorous notion of a computation step.
- Capabilities and limitations of this computing paradigm.
- Computability implies peptide computability. Converse?
- If converse true, under what conditions?

Why?

- Rigorous notion of a computation step.
- Capabilities and limitations of this computing paradigm.
- Computability implies peptide computability. Converse?
- If converse true, under what conditions?

Why?

- Rigorous notion of a computation step.
- Capabilities and limitations of this computing paradigm.
- Computability implies peptide computability. Converse?
- If converse true, under what conditions?

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$ (to represent antibodies);
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \rightarrow \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$ (to represent antibodies);
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$ (to represent antibodies);
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$ (to represent antibodies);
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^{+}$ is a language;
- A is a countable alphabet with $A \cap X^{*} = \emptyset$ (to represent antibodies);
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_{+}$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

What else do we need...

- A-attachment: partial mapping τ from decomposition of $w \in X^*$ with respect to E to A. $z = w_\tau$.
- If affinity of a is more in z we say it dominates.
- Reaction between words and symbols – if a dominates (i, j) in z then multiset $R(z, a)$ is formed and $\tau \rightarrow \tau'$.
- Reaction between words – if a in z' dominates some position in z.

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^{+} \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Converse of the simulation

Theorem

For every peptide computer $\mathcal{P} = (X, E, A, \alpha, \beta)$ with the following conditions:

1. *E and A are (at least) computably enumerable;*
2. *α is decidable;*
3. *β and χ are computable;*

and for every computably enumerable peptide program $\mathfrak{P}$ for $\mathcal{P}$, there is a Turing machine simulating the peptide computations of $\mathcal{P}$ according to $\mathfrak{P}$.

On our Results

- Ideal Assumptions:
  - $3 - D$ structure of peptides not considered.
  - There is no cross-reactions.
- Biochemical feasibility?

On our Results

- Ideal Assumptions:
  - $3 - D$ structure of peptides not considered.
  - There is no cross-reactions.
- Biochemical feasibility?

On our Results

- Ideal Assumptions:
  - $3 - D$ structure of peptides not considered.
  - There is no cross-reactions.
- Biochemical feasibility?

Future work

- Fault-tolerance to address cross-reaction.
- To study various kinds of non-determinism existing in the binding of symbols to words.

Future work

- Fault-tolerance to address cross-reaction.
- To study various kinds of non-determinism existing in the binding of symbols to words.