Proceedings of the International Workshop, Automata for Cellular and Molecular Computing, MTA SZTAKI, Budapest, pages 108 - 119, 2007.

Non-Determinism in Peptide Computer

M. Sakthi Balan

Department of Computer Science, The University of Western Ontario
London, Ontario, Canada, N6A 5B7
sakthi@csd.uwo.ca

Abstract

Peptide computer is a formal model for peptide computing. It involve reactions between various multiset of symbols and sequences. We study three kinds of non-determinism in peptide computer – global, locally-global and local. We show that (local) locally-global is a restrictive version of (locally-global) global. We also characterize conditions for a (locally-global) global system to be a system which is not (local) locally-global system.

1 Introduction

Peptide computing introduced by H. Hug and R. Schuler [6], takes interaction between peptides and antibodies as the basic frame work for computing. A formal model for this computing, called peptide computer, was proposed in [2]. This paper continues that study with an investigation of various kinds of non-determinism present in a peptide computer.

Peptide, a sequence of amino acids attached by covalent bonds called peptide bonds, consists of recognition sites, called *epitopes,* for the antibodies. A peptide can contain more than one epitope for the same or different antibodies. With each antibody, which attaches to a specific epitope, a binding power is associated, called its *affinity.* When antibodies compete for recognition sites – which may overlap in the given peptide – then the antibodies with greater affinity have higher priority. For further information regarding the bio-chemical processes themselves we refer to, for example, [5]. Dynamic global computing models for the immune system are presented in [7, 9].

Peptide computing refers to computational processes based on the elementary operations such as binding of antibodies to peptide sequences and removal of antibodies from peptide sequences.

In [6] it was shown how to solve the satisfiability problem using peptide computing and in the subsequent paper [3] it was shown to solve two further NP-complete problems – *Hamiltonian circuit* and *exact cover by 3-set*. Moreover in [3], a simulation of Turing machine by peptide computing is presented to show peptide computing is computationally complete. Towards formalizing the peptide computing model in a rigorous way a formal model called as peptide computer was proposed in [1, 2]. Peptide computer defines the notion of a step and it was also shown in [2] that it can be simulated by a Turing machine under some conditions. A survey on peptide computing depicting the state-of-the-art at its time is presented in [4].

A peptide computer, informally, consists of some symbols and sequences and the processing take place through reactions between symbols and sequences or between sequences. The presence of many symbols and sequences together with their multiple occurrences (multiset) brings in non-determinism to the system. Hence non-determinism can be studied in two ways: one, due to interactions between multiple occurrence of a sequence or a symbol and the other one where non-determinism happens when interactions occur between different sequences and different symbols. Non-determinism is an essential one for unconventional computing like peptide computing. Hence a study of non-determinism existing in a peptide computer would help us to understand how processing take place.

Our paper is organized as follows: in the following section we give some preliminary on peptide computer and introduce the basic notations that are necessary for the paper. In Section 3 we define three kinds of non-determinism in peptide computer and in Section 4 we study some of the properties of non-determinism. The paper concludes with some remarks in Section 5.

2 Preliminaries

For a set S, $|S|$ denotes the cardinality of S. When S is a singleton set, $S = \{x\}$ say, we often omit the set brackets, that is, we write x instead of $\{x\}$. For sets S and T, consider a relation $\varrho \subseteq S \times T$. Then ϱ^{-1} is the relation $\varrho^{-1} = \{(t, s) \mid (s, t) \in \varrho\}$ and, for $s \in S$, $\varrho(s) = \{t \mid (s, t) \in \varrho\}$. We use the notation $\varrho : S \overset{\circ}{\to} T$ to denote a partial mapping of S into T. In that case $\mathsf{dom}\,\varrho$ is the subset of S on which ϱ is defined. The notation $\varrho : S \to T$ means that ϱ is a total mapping of S into T, hence $\mathsf{dom}\,\varrho = S$ in this case. In addition to the standard symbols for operations on sets, we use the symbol $\uplus$ to denote disjoint union.

Let S be a non-empty set. A *multiset* on S is a pair $M = (I, \iota)$ where I is a set, the *index set,* and ι is a mapping of I into S, the *index mapping.* A multiset M is non-empty, if I is non-empty; it is finite if I is finite. For $s \in S$, the number $\mathsf{mult}(s) = |\{i \mid i \in I, \iota(i) = s\}|$ is the *multiplicity* of s. When I is countable, we write $M = \{m_i \mid i \in I\}$ where $m_i = \iota(i)$ is implied. With this notation, it is possible that $m_i = m_j$ while $i \neq j$ for $i, j \in I$. We use the standard symbols for set theoretic operations also for multisets. However, on multisets, union is disjoint union and both intersection and difference take multiplicities into account. Formally this can be handled by appropriate operations on the index sets.

Multisets as defined above are also called *families* in the literature. The usual definition of a multiset as a set $\{(s, \mathsf{mult}(s)) \mid s \in S\}$ of pairs is adequate only when all multiplicities are finite.

By $\mathbb{N}$ and $\mathbb{N}_0$ we denote the sets of positive integers and of non-negative integers, respectively. The set $\mathbb{B} = \{0, 1\}$ represents the set of Boolean values. For $n \in \mathbb{N}_0$, the ordinal number $\mathfrak{n}$ is represented by the set $\mathfrak{n} = \{i \mid i \in \mathbb{N}_0, i < n\}$. Thus, for example, $\mathfrak{o} = \emptyset$, $\mathfrak{1} = \{0\}$ and, in general, $\mathfrak{n} = \{0, 1, \ldots, n-1\}$. By $\mathbb{R}$ we denote the set of real numbers, and $\mathbb{R}_+ = \{r \mid r \in \mathbb{R}, r \geq 0\}$.

An alphabet is a non-empty set. Let X be an alphabet. Then X^* is the set of all words over X including the empty word λ, and $X^+ = X^* \setminus \{\lambda\}$. For a word $w \in X^*$, $|w|$ is its length. Any word $u \in X^*$ with $w \in uX^*$ is a prefix of w; let $\mathsf{Pref}(w)$ be the set of prefixes of w; the words in $\mathsf{Pref}_+(w) = \{u \mid u \in X^+, w \in uX^+\}$ are the proper prefixes of w. Similarly, a word $u \in X^*$ with $w \in X^*uX^*$ is an infix of w, $\mathsf{Inf}(w)$ is the set of infixes of w and $\mathsf{Inf}_+(w) = \{u \mid u \in X^+, u \in \mathsf{Inf}(w), u \neq w\}$ is the set of proper infixes of w. A language over X is a subset of X^*. For a language L over X and $Y \in \{\mathsf{Pref}, \mathsf{Pref}_+, \mathsf{Inf}, \mathsf{Inf}_+\}$, $Y(L) = \bigcup_{w \in L} Y(w)$.

Let L be a language over X and $w \in X^*$. An *L-decomposition* of w is a pair of sequences $(u_0, u_1, \ldots, u_k)$, $(v_0, v_1, \ldots, v_{k-1})$ of words in X^* such that $u_0v_0u_1v_1 \cdots v_{k-1}u_k = w$, $v_0, v_1, \ldots, v_{k-1} \in L$ and $u_0, u_1, \ldots, u_k \notin X^*LX^*$. A language in X^+ such that every word has a unique L-decomposition is called a *solid code* [8]. Consider $w \in X^+$ of length n, say $w = x_0x_1 \cdots x_{n-1}$ with $x_i \in X$ for $i = 0, 1, \ldots, n-1$. An L-decomposition of w as above can be specified by a set of pairs $\{(i_l, j_l) \mid l = 0, 1, \ldots, k-1\}$ such that, for $l = 0, 1, \ldots, k-1$, $v_l = x_{i_l}x_{i_l+1} \cdots x_{j_l}$. Let $\partial_L(w)$ be the set of L-decompositions when represented in this way. Let $\mathcal{D}(L) = \{(w, d) \mid w \in X^*, d \in \partial_L(w)\}$ be the set of words together with all their L-decompositions.

Now we give a brief description of peptide computer presented in [2].

Definition 1 *A* peptide computer *is a quintuple $\mathcal{P} = (X, E, A, \alpha, \beta)$ where X is a finite alphabet (to represent basic building units like molecules), $E \subseteq X^+$ is a language (to represent epitopes), A is a countable alphabet with $A \cap X^* = \emptyset$ (to represent antibodies), $\alpha \subseteq E \times A$ is a relation (such that $a \in \alpha(e)$ means that antibody a can be attached to epitope e), $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$ (denoting the affinity between epitope e and antibody a).*

Consider a word $w \in X^+$ and $d \in \partial_E(w)$. An *A-attachment* is a partial mapping $\tau : d \overset{\circ}{\to} A$. Suppose $w = x_0x_1 \cdots x_n$ and $d = \{(i_l, j_l) \mid l = 0, 1, \ldots, k-1\}$. Then τ defines a word $w_\tau \in (X \cup (E \times A))^*$ as follows: For all $l = 0, 1, \ldots, k-1$, if $(i_l, j_l) \in \mathsf{dom}\,\tau$ replace $e = x_{i_l}x_{i_l+1} \cdots x_{j_l}$ by $(e, \tau(i_l, j_l))$ in w. Such a mapping τ is *legal* if $(e, \tau(i_l, j_l)) \in \alpha$ for all l. When τ is legal then $w_\tau \in (X \cup \alpha)^*$ and τ is called an *A-attachment to w*. For a language $L \subseteq X^+$, let $\mathcal{T}(L)$ be the set of A-attachments to words in L. Conversely, a word $z \in (X \cup \alpha)^*$ defines a word $w \in X^*$ and a set of A-attachments τ, such that $w_\tau = z$. Note that w is uniquely defined, but that τ may apply to several $d \in \partial_E w$.

Consider a word $z \in (X \cup \alpha)^+$ and a symbol $a \in A$. Let w and τ be such that $w_\tau = z$. Moreover, let $w = x_0x_1 \cdots x_n$ with $x_0, x_1, \ldots, x_n \in X$. Consider any $d \in \partial_E w$ with $\mathsf{dom}\,\tau \subseteq d$ and any $d' \in \partial_E w$. For $(i, j) \in d'$ let $e_{i,j} = x_ix_{i+1} \cdots x_j$. We say that *$a$ dominates* (i, j) in z when the following condition is satisfied: For all $(i', j') \in d$ such that $\{i', i'+1, \ldots, j'\} \cap \{i, i+1, \ldots, j\} \neq \emptyset$ and $(i', j') \in \mathsf{dom}\,\tau$,

$$\beta(e_{i,j}, a) > \beta(x_{i'}x_{i'+1} \cdots x_{j'}, \tau(i', j')).$$

In such a case, all pairs $(i', j') \in d$ with $\{i', i'+1, \ldots, j'\} \cap \{i, i+1, \ldots, j\} \neq \emptyset$ are said to be *affected.* If a dominates (i, j) in z, the following *basic reaction will happen* forming a multiset $R(z, a)$: For each affected pair (i', j'), a copy of $\tau(i'j')$ is put into $R(z,a)$; let $Y \subseteq \mathsf{dom}\,\tau$ be the set of pairs which are not affected and let $d'' \in \partial_E w$ be such that $Y \cup (i,j) \subseteq d''$. Define the A-attachment $\bar{\tau} : d'' \overset{\circ}{\to} A$ by $\bar{\tau}(p) = \tau(p)$ for $p \in Y$ and $\bar{\tau}(i,j) = a$. Put a copy of $w_{\bar{\tau}}$ into $R(z,a)$. The multiset $R(z,a)$ is the *result of a basic reaction* between z and a. If a is binding with z and some symbols are released from z when $R(z,a)$ is formed then we denote the set of released symbols by $Out(z,a)$. If nothing is released when a binds then $Out(z,a)$ will be $\{\lambda\}$.

We also need to consider *basic reactions* between words $z, z' \in (x \cup \alpha)^+$, where z and z' need not be different. Again we want to define the resulting multiset $R(z,z')$. We use w, d and τ as above. Now $z' = w'_{\tau'}$ where $\tau' : d' \overset{\circ}{\to} A$ for some $d' \in \partial_E w'$. Consider $(i',j') \in \mathsf{dom}\,\tau'$ and let $a = \tau'(i'j')$. Moreover, let $e'_{i',j'}$ be the infix of w' which starts at i' and ends at j'. Suppose a dominates (i,j) in z for some $(i,j) \in \bar{d} \in \partial_E w$ and $\beta(e_{i,j},a) > \beta(e'_{i',j'},a)$, then the reaction is as follows.

Since the basic reactions between two words z and z' are with respect to a, we represent these by $R_a(z,z')$. They take place in two steps:

1. The reaction $\mathsf{Sep}R_a(z,z')$ produces a multiset containing z, z'' and a, where z'' is defined as follows: let τ'' be the restriction of τ' to $\mathsf{dom}\,\tau' \setminus (i'j')$; then $z'' = w'_{\tau''}$.

2. Next is the reaction leading to $R(z,a)$.

As before $Out(z_1,z_2)$ denotes the set of symbols released from z_1 when a binds with z_1. When z and z' are the same occurrence of a word then $\mathsf{Sep}R_a(z,z')$ consists only of z'' and a.

The basic reactions resulting in $R(z,a)$ and $R_a(z,z')$ take place only when there is instability. Instability between z and a occurs when a dominates $(i,j) \in \partial_E w$ where $z = w_\tau$. Likewise instability between two words z and z' occurs when there is a symbol $a = \tau'(i',j')$ where $(i',j') \in \mathsf{dom}\,\tau'$ and $\tau' : d' \overset{\circ}{\to} A$ for some $d' \in \partial_E(w')$.

A basic reaction can trigger a sequence of reactions; this might even lead to a cycle which in turn will not result in a stable configuration. In the sequel we refer to $R(z,a)$ and $R_a(z_1,z_2)$ as the results of a basic reaction or as multisets, whichever is more appropriate in the context.

Definition 2 *Let $\mathcal{P}$ be a peptide computer. A* peptide configuration *is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.*

To a peptide configuration P a basic reaction may apply when instability exists in the configuration, that is, there may be $z, z' \in (X \cup \alpha)^+$ or $a \in A$ which occur in P such that $R(z,a)$ differs from the multiset consisting of z and a or $R(z,z')$ differs from the multiset consisting of z and z'. In either case a basic reaction non-deterministically removes (z,a) or (z,z') from P and adds $R(z,a)$ or $R(z,z')$, respectively. Let $R(P)$ be the set of peptide configurations which result from P through one basic reaction. For $n \in \mathbb{N}_0$, let R^n be the n-fold iteration of R.

Definition 3 *A peptide configuration P is said to be* stable *if $R(P) = \{P\}$.*

If $R^n(P)$ consists of stable configurations only, for some n, define $R^*(P) = R^n(P)$ for this n. Otherwise, $R^*(P) = \emptyset$. Let Γ be the class of stable peptide configurations.

To define peptide computations, we also need the following objects:

Definition 4 *A* peptide instruction *has the form $+P$ or $-P$ where P is a peptide configuration.*

When P' is a peptide configuration and I is a peptide instruction then

$$I(P') = \begin{cases} P' \cup P, & \text{if } I = +P, \\ P' \setminus P, & \text{if } I = -P, \end{cases}$$

with union and difference taken as multiset operations.

The instruction $-P$ is called a flushing instruction if $P = P' \cap A$; hence in this case all the symbols in A which are not binding to any sequence in X^+ are removed from the configuration.

Definition 5 *A* peptide program *is a pair $(\mathfrak{P}, \chi)$ where $\mathfrak{P}$ is a mapping from Γ^* into the set of peptide instructions and χ is a (halting) function $\chi : \Gamma \to \mathbb{B}$.*

Definition 6 *Let $\mathcal{P}$ be a peptide computing model and let $(\mathfrak{P}, \chi)$ be a peptide program for $\mathcal{P}$. A* peptide computation *is a word $c = c_0 c_1 \cdots c_t \in \Gamma^*$ with $c_0, c_1, \ldots, c_t \in \Gamma$ such that*

$$c_i \in R^*(\mathfrak{P}(c_0 c_1 \cdots c_{i-1})(c_{i-1}))$$

for $i = 0, 1, \ldots, c_t$.

A computation as above starts with $c_0 \in R^*(\mathfrak{P}(\lambda))$ and ends when $\chi(c_i) = 1$ for the first time.

To encode inputs we need a mapping γ from inputs to Γ, an input encoding; we also need an output decoding, that is, a mapping δ from Γ to outputs.

Definition 7 *A function f from inputs to outputs is* peptide computable *if there is a peptide program $\mathfrak{P}$, a computable input encoding γ of inputs into $\mathfrak{P}(\lambda)$ and a computable decoding of Γ into outputs such that, for every $x \in \mathsf{dom}\, f$, there is a peptide computation $c_0 c_1 \cdots c_t$ with $c_0, c_1, \ldots, c_t \in \Gamma$ and $\gamma(x) = c_0$ satisfying $\chi(c_t) = 1$ and $\delta(c_t) = f(x)$.*

3 Non-Determinism in Peptide Computer

As mentioned briefly in the introduction non-determinism comes into picture in peptide computer in two ways. First one is, many-to-many interactions between symbols and sequences and the other is, due to multiplicities of the sequences and symbols. But when we look upon a multiset as a pair $M = (I, \iota)$ we can visualize both the non-determinism as a single one since M can be virtually thought of as a set but holding all the original properties of multiset. With this important note we proceed to define non-determinism in peptide computer.

We define non-determinism in peptide computer in three levels: global, locally-global and local. We first describe all three of them very informally in the sequel.

Informally speaking there are two ways a non-determinism can occur in a peptide computer: one, when there are more than one epitopes defined for a symbol and two, when there are more than one symbols in competition to bind to their epitopes which overlap with each other. The global definition is similar to the definition of non-determinism in accepting devices like finite state automata, pushdown automata and so on. It is defined in a more generic way on the system as a whole.

The other two local definitions are more interesting ones. Basically when reaction take place we have a peptide configuration that contains all the sequences and symbols participating in the reaction. The locally-global non-determinism is defined by restricting the global definition of non-determinism to the sequences and the symbols present in the configuration.

The third one is defined on the reactions taking place between symbols and a sequence or between two sequences. As defined earlier, these reactions happen only when there is an instability in the medium. When instability is present in the medium and either of the following conditions are satisfied: there is a possibility of more than one epitopes for a symbol to bind or there are more than one symbol that can attach to an epitope, then we say that it is locally non-deterministic.

All the above definitions will be explained more in detail when we define them formally.

It can be seen that the local definitions are more restrictive versions of the global definition. We prove it formally in the next section. There can be many instances that even though the definition might be non-deterministic globally it might not be non-deterministic when reaction occurs. We will present under what conditions this happens.

The reason behind defining non-determinism in three levels is explained in the sequel. The peptide computer is a generic system used to solve a set of problems. Hence it can be defined as a non-deterministic or deterministic one. To solve a problem by peptide computer we write a peptide program to work with the available sequences and symbols. Even when the generic system is a non-deterministic one, when writing a program for a problem there might not be a need for non-determinism. Hence we have separated out the non-determinism as global and locally-global ones. Similarly even when a program uses non-determinism when actual processing take place due to the structure of the sequence and the binding properties of the symbols there might not be any non-determinism present when reactions take place. These arguments show that we have to separate non-determinism into three levels.

Before defining non-determinism formally, we define few technical terms which we use in the paper later.

Definition 8 *For a peptide configuration P we say a sequence $z \in (X \cup \alpha)^+$ as a* participating sequence *if $z \in P$. Likewise a symbol a is a* participating symbol *if $a \in P$.*

The set of all participating sequences is denoted by *Pseq* and the set of all participating symbols is denoted by *Psym*. A participating sequence is denoted by *pseq* and a participating symbol by *psym*.

Definition 9 *For a peptide configuration P, an epitope $e \in E$ is said to be a* participating epitope *if $e \in Sub(w)$ where $w_\tau = z$ is a pseq and τ is an arbitrary $A-attachment$.*

The set of all participating epitopes is denoted by *Pepi*. A participating epitope is denoted by *pepi*.

We recall that only when the configuration attains stability the next peptide instruction is applied. When a peptide instruction is carried out by peptide computer, the configuration becomes instable and reactions occur to attain stability. Now we have the following definition:

Definition 10 *The time period between applying a peptide instruction and attaining stability is defined as the* instability period. *The series of reactions happening in the instability period are collectively called as a* step.

Definition 11 *For a sequence $x \in V^+$ represented as $x = x_1 x_2 \cdots x_n$ where $x_i \in V$, any epitope e in x is of the form $x_i \cdots x_j$ where $i \leq j \leq n$.*

For any two epitopes e, e' in x with $e = x_i \cdots x_j$ and $e' = y_k \cdots y_l$ where $i \leq j \leq n$ and $k \leq l \leq n$ we say e and e' overlap when either $i \leq k \leq j$ or $k \leq i \leq l$.

Now we present formal definitions for three types of non-determinism in peptide computer.

Definition 12 *A* peptide computer $\mathcal{P} = (X, E, A, \alpha, \beta)$ *is said to be* globally non-deterministic *if either of the following conditions are satisfied:*

- *there exists a symbol $a \in A$ and epitopes $e_1, e_2, \cdots, e_n, n \geq 2$ such that $\{(e_1, a), (e_2, a), \cdots, (e_n, a)\} \subseteq \alpha$*
- *there are symbols $a_1, a_2, \cdots, a_m, m \geq 2$ and there exists epitopes $e_1, e_2, \cdots, e_p$, $p \geq 1$ such that*
  - *each e_j overlaps with each other e_i where $1 \leq i \neq j \leq p$, and*
  - *for each a_i there exists at least one j such that $(e_j, a_i) \in \alpha$ where $1 \leq i \leq m$ and $1 \leq j \leq p$.*

Definition 13 *A* peptide computer $\mathcal{P} = (X, E, A, \alpha, \beta)$ *is said to be* locally-global non-deterministic *if either of the following conditions are true for all peptide configuration P:*

- *there exists a symbol $a \in Psym$ and epitopes $e_1, e_2, \cdots, e_n, n \geq 2, e_i \in Pepi$, $1 \leq i \leq n$ such that $\{(e_1, a), (e_2, a), \cdots, (e_n, a)\} \subseteq \alpha$*
- *there are symbols $a_1, a_2, \cdots, a_m, m \geq 2, a_i \in Psym, 1 \leq i \leq m$ and there exists epitopes $e_1, e_2, \cdots, e_p, p \geq 1, e_j \in Pepi, 1 \leq j \leq p$ such that*
  - *each e_j overlaps with each other e_k where $1 \leq j \neq k \leq p$, and*
  - *for each a_i there exists at least one l such that $(e_l, a_i) \in \alpha$ where $1 \leq l \leq p$ and $1 \leq i \leq m$.*

Before defining locally non-deterministic we recall that a step in peptide computer consists of a sequence of set of reactions $R^1(P), R^2(P), \cdots\cdots$ where P is the configuration of the peptide computer. At each i, $R^i(P)$ consists of reactions between z and a and reactions between two sequences z_1 and z_2. Now we define local non-determinism:

Definition 14 *A step in peptide computer is said to be non-deterministic if there exists a $m \geq 1$ such that $R^m(P)$ satisfies either of the following conditions:*

- *if $a \in Psym$ dominates more than one pair, say the set of pairs $\{(i_1, j_1), (i_2, j_2), \cdots, (i_n, j_n)\}$*
- *if there are symbols $a_1, a_2, \cdots, a_m, m \geq 2, a_i \in Psym, 1 \leq i \leq m$ and there exists epitopes $e_1, e_2, \cdots, e_p, e_i \in Pepi, 1 \leq i \leq p$ such that*
  - *for each a_i there exists at least one j such that $(e_j, a_i) \in \alpha$ and a_i dominates e_j, and*
  - *each e_j overlaps with each other e_i.*

In all the three definitions presented above we note that there are two different views of non-determinism – one, with respect to a symbol which can non-deterministically bind to one of its epitopes and the other one with respect to epitopes, where non-deterministically one of many symbols can bind to it. We note here that epitopes for those symbols need not be the same epitope but a set of epitopes with the property that epitopes overlap with each other.

Definition 15 *A symbol $a \in A$ is a* non-deterministic symbol *if $\mid \alpha(a) \mid > 1$.*

The set of all non-deterministic symbols in A is denoted by A_{nd}. In the following definitions we classify the set of epitopes as non-deterministic and deterministic epitopes.

Definition 16 *A set $F \subseteq E$ is said to be* overlapping set *if every two epitopes $e_i, e_j \in F$ overlap.*

By definition it should be obvious to note that every singleton set of E is an overlapping set. For any sequence x over V^* if any epitope in x is associated with the overlapping set F then we denote it as x_F.

Definition 17 *An overlapping set $F = \{f_1, f_2, \cdots, f_m\}, m \geq 1$ is said to be a* non-deterministic epitope-set *if there is a set $A' \subseteq A$, say $A' = \{a_1, a_2, \cdots, a_n\}$ where $n \geq 2$ such that for all $a_i \in A'$ there is at least one j satisfying the condition that $(e_j, a_i) \in \alpha$.*

The set of all non-deterministic epitope set is denoted by $\mathcal{E}_{nd}$. If $\mathcal{E}_{nd} = \{E_1, E_2, \cdots, E_n\}$ and $F \subseteq E$ then we define $\mathcal{E}_{nd}^F$ as the family of set $\{E_1 \cap F, E_2 \cap F, \cdots E_n \cap F\}$.

Definition 18 *Let F be an overlapping set and $e \in F$. The* weight *of F is defined as $\beta(e, a)$ where (e, a) is a subsequence of x_F.*

We note that the above definition is not ambiguous since all epitopes in F overlap and so at any instance only one epitope will be bounded by a symbol. We denote weight of F as $w(F)$.

Definition 19 *Let $e \in E$. We say e is* closed *if any overlapping set F containing e has a non-zero weight. If all overlapping sets containing E are zero weight then it is said to be* open.

Using the definitions closed and open we define a characteristic function $\chi : E \longrightarrow \{0,1\}$ with $\chi(e) = 0$ if e is closed and $\chi(e) = 1$ if e is open.

Definition 20 *Let $X = \{x_1, x_2, \cdots, x_n\}, n \geq 1$ be a set. We define tuple(X) as a n-tuple $(x_{i_1}, x_{i_2}, \cdots, x_{i_n})$ where $(i_1, i_2, \cdots, i_n)$ is any permutation of the set $\{1, 2, \cdots, n\}$. We extend the definition of χ to tuples over any subset of E, say $(e_1, e_2, \cdots, e_k)$, as $(\chi(e_1), \chi(e_2), \cdots, \chi(e_n))$.*

Definition 21 *A peptide computer is said to be strictly globally non-deterministic if it is globally non-deterministic but not locally-global non-deterministic. Likewise a peptide computer is said to be strictly locally-global non-deterministic if it is locally-global non-deterministic but not locally non-deterministic.*

We use the following notations in our paper. The set of all peptide computer $\mathcal{P}$ is denoted by PC. The set of all peptide computers which are globally non-deterministic is denoted by PC_{gnd}. Likewise peptide computers which are locally-global non-deterministic (locally non-deterministic) is denoted by PC_{lgnd} (PC_{lgd}).

4 Results on Non-Determinism

Theorem 1

1. *$PC_{lgnd} \subseteq PC_{gnd}$,*
2. *$PC_{lnd} \subseteq PC_{lgnd}$.*

Proof. Let $\mathcal{P} \in PC_{lgnd}$. We prove $\mathcal{P}$ is also a globally non-deterministic one. This simply follows from definition. Since $Psym \subseteq A$ and $Pepi \subseteq E$, it directly follows that if condition (1) of Definition 13 is true then condition (1) of Definition 12 is true, or if the condition (2) of Definition 13 is true then condition (2) of Definition 12 is true. This shows that $\mathcal{P} \in PC_{gnd}$.

Let $\mathcal{P}' \in PC_{lnd}$. We will show that $\mathcal{P}' \in PC_{lgnd}$. Our assumption implies that either the condition (1) or condition (2) of Definition 14 is satisfied. If condition (1) is true then $a \in Psym$ dominates more than one pair, i.e., the set of pairs $\{(i_1, j_1), (i_2, j_2), \cdots (i_n, j_n)\}$ where $n \geq 2$. By the definition of a symbol dominating a sequence it implies that for the symbol $a \in Psym$, $\{(e_1, a), (e_2, a), \cdots, (e_n, a)\} \subseteq \alpha$ where if (i_k, j_k) is a pair from the sequence $x^k \in Pseq$ then $e_k = x^k_{i_k} x^k_{i_k+1} \cdots x^k_{j_k}, 1 \leq k \leq n$. This proves condition (1) of Definition 13 is true.

If condition (2) of Definition 14 is satisfied, then there are symbols $a_1, a_2, \cdots, a_m$, $m \geq 2, a_i \in Psym, 1 \leq i \leq m$ and there exists epitopes $e_1, e_2, \cdots, e_p, p \geq 1, e_i \in$

$Pepi, 1 \leq i \leq p$ such that for each a_i there exists at least one j such that $(e_j, a_i) \in \alpha$. This shows condition (2) of Definition 13 is satisfied.

Hence in either case we have $\mathcal{P} \in PC_{lgnd}$. □

Now we study some properties of peptide computer which will help us to exhibit the conditions for a peptide computer to be strictly globally-non-deterministic. Similarly we also examine under what conditions peptide computer is strictly locally-global non-deterministic one.

Theorem 2 *A peptide computer $\mathcal{P}$ is strictly globally non-deterministic if it satisfies either of the following conditions: for all i,*

- *P_i contains no symbols from A_{nd} and no epitopes from E_{nd}.*
- *For all $a \in A_{nd} \cap P_i seq$, $\mid \alpha^{-1}(a) \cap P_i epi \mid= 1$, and for all $E \in \mathcal{E}_{nd}^{P_i epi}$, $\mid \alpha(E) \mid= 1$.*

Proof. If P_i contains no symbols from A_{nd} and no epitopes from E_{nd} then it is trivial that $\mathcal{P}$ is strictly globally non-deterministic since no other symbols and epitopes will contribute to non-determinism.

We assume that P_i contain symbols from A_{nd}. Let $a \in A_{nd} \cap P_i seq$. Suppose a satisfies the condition $\mid \alpha^{-1}(a) \cap P_i epi \mid= 1$ then it signifies that there is exactly one $e \in P_i epi$ such that $(e, a) \in \alpha$. Hence the configuration P_i has only one epitope for all non-deterministic symbols in the configuration. This implies the condition (1) is not satisfied in the Definition 13.

If there is a non-deterministic epitope set E in the configuration P_i and satisfies the condition $\mid \alpha(E) \mid= 1$ then it shows that there is only one $a \in P_i sym$ such that $(e, a) \in \alpha$ where $e \in E$. The possibility of more than one e is ruled out by our first assumption that $\mid \alpha^{-1}(a) \cap P_i epi \mid= 1$. This shows that $\mathcal{P}$ is not locally-global non-deterministic.

Hence $\mathcal{P}$ is strictly globally non-deterministic. □

Theorem 3 *A peptide computer $\mathcal{P}$ is strictly locally-global non-deterministic if it satisfies either of the following conditions: for all i,*

- *For all $a \in A_{nd} \cap P_i seq$, if $\chi(tuple(\alpha^{-1}(a) \cap P_i epi)$ is a zero vector or an unit vector.*
- *For all $E \in \mathcal{E}_{nd}^{P_i}$, $\mid \alpha(E) \mid\leq 1$ if $w(E) = 0$ and $\mid \alpha(E) \mid\geq 0$ if $w(E) = 1$.*

Proof. Let $\mathcal{P}$ be locally-global non-deterministic satisfying the above conditions we will show that it is not locally non-deterministic. Since $\mathcal{P}$ is locally-global non-deterministic either of the conditions in Definition 13 is true. Suppose the condition (1) is true for the configuration P_i. Let $a \in P_i sym \cap A_{nd}$. Then there exists epitopes $e_1, e_2, \cdots, e_n, n \geq 2$ such that $e_j \in P_i epi$ and $(e_j, a) \in \alpha$. Hence a has n choices of epitopes to bind. But since we are looking for a locally deterministic one, these choices should not exist when reaction take place, i.e., there should be at most one choice for the symbol a. For this to happen except at most one epitope, say e_i,

a should not dominate any other epitope. This implies all epitopes $e_j (j \neq i)$ are bounded by a symbol – if some of them are not bounded by symbols any one of the epitope overlapping with it is bounded by a symbol. This implies that if we consider $(e_1, e_2, \cdots, e_n)$ as an n-dimensional vector then $\chi((e_1, e_2, \cdots, e_n))$ is a unit vector or zero vector. Hence if $\chi((e_1, e_2, \cdots, e_n))$ is a unit vector or zero vector for all $a \in P_i sym \cap A_{nd}$, then condition (1) of Definition 14 is not satisfied.

Now suppose there is a non-deterministic epitope set E in the configuration. Since we look for a peptide computer that is not locally deterministic there should not be $n (n \geq 2)$ possibilities of symbols binding with epitopes in E. There are only two choices for that: (1) $w(E) > 0$ and (2) $w(E) = 0$ and $n = 1$. If $w(E) > 0$ then there are no open epitopes and hence n can be arbitrary. In other case since $w(E) = 0$ all epitopes in E are open. Hence there should be at most one symbol in competition for an epitope in E.

The discussion shows that $\mathcal{P}$ is strictly locally-global non-deterministic. $\square$

5 Conclusion

We defined three levels of non-determinism in peptide computer: global, locally-global and local. We showed global is a more general definition, locally-global is a restrictive version of global and local is further restrictive version of locally-global. We also characterized conditions for a global system to not to be a locally-global one and locally-global to not to be a local one.

The three levels of non-determinism defined in peptide computer is helpful in the following way: once a (locally-global) global non-deterministic peptide computer is given we can either select the system to be (locally) locally-global non-deterministic or strictly (locally-global) globally-non-deterministic. More interesting question is to study how dynamically under some contextual conditions the system can pick a step to be a locally non-deterministic one or a locally deterministic one. This will control the use of non-determinism to a greater extent.

References

[1] M. S. Balan and H. Jürgensen. Peptide computing: Universality and theoretical model. In *Unconventional Computation*, volume LNCS 4135, pages 57–71. Springer-Verlag, 2006.

[2] M. S. Balan and H. Jürgensen. On the universality of peptide computing. *Natural Computing*, 2007. In print.

[3] M. S. Balan, K. Krithivasan, and Y. Sivasubramanyam. Peptide computing: Universality and computing. In N. Jonoska and N. Seeman, editors, *Proceedings of Seventh International Conference on DNA based Computers, LNCS 2340*, pages 290–299, 2002.

[4] M.S. Balan, H. Jürgensen, and K. Krithivasan. Peptide computing: A survey. Technical Report preprint 4/2005, ISSN 0946-7580, Universität Potsdam, Germany, 2005.

[5] C.R. Cantor and P.R. Schimmel. *Biophysical Chemistry.* W. H. Freeman and Company, San Francisco, 1980. 3 volumes.

[6] H. Hug and R. Schuler. Strategies for the development of a peptide computer. *Bioinformatics*, 17:364–368, 2001.

[7] Y. Ishida. *Immunity-Based Systems: A Design Perspective.* Springer, Berlin, 2004.

[8] H. Jürgensen and S. Konstantinidis. Codes. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 511–607. Springer-Verlag, Berlin, 1997.

[9] A. O. Tarakanov, V. A. Skormin, and S. P. Sokolova. *Immunocomputing, Principles and Applications.* Springer, New York, 2003.