An Adaptive e-Learning Environment using Distributed Spiking Neural P Systems

Susan Elias, Sarath Chandar A.P
Sri Venkateswara College of Engineering
Sriperumbudur, Chennai, India
susan@svce.ac.in,sarathcse2008@gmail.com

Kamala Krithivasan, S.V. Raghavan
Indian Institute of Technology, Madras
Chennai, India
kamala@iitm.ac.in,svr@cs.iitm.ernet.in

***Abstract*—The motivation behind the proposed research work is the need for an innovative e-learning system that can adapt to the learning capability of every individual. Adaptive e-learning systems create new opportunities and at the same time have several research challenges that need to be addressed. The primary requirement of such adaptive systems is the need to create and represent adaptable content effectively. This paper presents a membrane computing model to demonstrate how adaptable content can be represented and used efficiently. The *Spiking Neural P System (SNP)* is a membrane computing model inspired by the way neurons communicate by means of *spikes*. This paper proposes the *Distributed Spiking Neural P System (DSNP)*, a variant of the existing *Distributed P System*, that can be used to represent dynamic and distibuted systems. Temporal relations captured on a *time line* during authoring of the e-course, can be automatically converted into an *SNP* system using the algorithm presented in the paper. An algorithm for the automatic generation of the *DSNP* from the e-course compositions represented using a linked list of *SNPs* is also presented in the paper along with experimental results to prove the efficiency and scalability of the proposed model.**

***Keywords*-Adaptive e-Learning; Spiking Neural P Systems; Multimedia presentation; Membrane Computing.**

I. Introduction and Related Work

E-Learning systems do exist and are successfully being utilized by several organizations and universities currently, but the challenge now is to personalize the content for each user automatically. Learning strategies and styles need to be clearly identified and suitably incorporated in the learning models. A survey of the various learning strategies and styles presented in [4] and [7] categorize learning strategies as *inquiry based, problem based, ubiquitous learning* and *blended learning*, while the various learning styles are classified as *diverging, assimilating, converging* and *accommodating*. These are incorporated into the current prevalent model of e-Learning which is primarily through Learning Management Systems (LMSs). The complexity in designing Adaptive e-learning environments is in personalising the content as different individuals have different learning patterns, and respond differently to the same content. This realization of the need for content adaptation has led to the development of the of the Adaptive e-Learning Technology [9],[3],[2]. Adaptive behaviour in a learning environment can have numerous manifestations.These have been broadly catogerised [1] as follows :

1) Adaptive Interactions
2) Adaptive Course Delivery
3) Adaptive Content Discovery and Assembly
4) Adaptive Collaboration Support

The first category refers to adaptations that support user interaction with the system without modifying the learning content itself. Adaptive Course delivery refers to adaptations that are intended to customise a course for an induvidual learner. The third category refers to the application of adaptive techniques for the discovery and assembly of learning content from potentially distributed sources or repositories. The last category captures adaptive support in learning processes that involve communication between multiple persons towards common objectives. Adaptive course delivery deals with personalisation of content and will be presented here as an application of the proposed membrane computing model called *Distributed Spiking Neural P System*.

Membrane computing [8] deals with distributed and parallel computing models by abstracting computing ideas from the structure and functioning of living cells as well as from the way cells are organised in tissues. Membrane systems are also called *P* systems. The basic types of *P* systems are the symbol object *P* system with multi-set rewriting rules, systems with symport/antiport rules, tissue like *P* systems, neural-like *P* system, etc. Several variants have since evolved for various applications and computing requirments.This paper is based on the *Spiking Neural P Systems* [6] and the *Distributed P system* [5]. Variants of these existing computing models have been defined and intergrated resulting in a new model having very innovative applications. *Spiking Neural P Systems* are computing models that are inspired by the way the neurons communicate by means of electrical impulses of identical shape, called *spikes*. Further the distributed manner in which the brain processes information is captured efficiently in the computing model by the associated rules and their applicability in the model.

The *Distributed Spiking Neural P System(DSNP)* proposed in this paper is a variant of existing *Distributed P system* [5]. The existing *Distributed P system (dp system)* represents a natural framework for solving problems in a distributed way. The problem to be solved is split into parts and introduced as sub-problems to the *P* systems that are components of the *dP* systems. These components solve the sub-problems in parallel and produce solutions to the initial problem by communicating and interacting with each other. In this the basic components of the distributed system will be *SNP*

systems and hence the proposed model is refered to as *Distributed Spiking Neural P System (DSNP)*. This paper presents an application of the proposed *DNSP* System by modeling an *Adaptive e-Learning Environment* using it.

II. Motivation and Problem Definition

Spiking Neural P Systems and *Distributed P Systems* were found to be models that were suitable for integration and for application in dynamic and real-time environments. The need for a model that could respond to changes in the environment and adapt itself suitably is the motivation for the research work presented in this paper. Mathematical models of computation aid in the effective analysis and verification and hence an appropriate model namely the *Spiking Neural P System* was identified and modified suitably for the challenging application of Adaptive e-Learning which inherently involves a lot of dynamism.

III. Proposed Models and Terminology

In this section the proposed *Distributed Spiking Neural P System* will be described and its features will be illustrated. A variant of the existing *Spiking Neural P System* has been used innovatively as the basic component of the proposed model. The formal definition of the proposed variant is presented in this section followed by the proposed *Distributed Spiking Neural P System*.

A. Variant of the Spiking Neural P System

Spiking Neural P Systems were introduced in [6] as a computationally complete model both in generating and accepting modes. Here a variant of the basic system that has distinct input neurons and a localized environment has been proposed. The proposed variant of the *spiking neural P system* (in short, an *SNP* System), of degree m ≥ 1, is of the form

$$\pi = (O, \sigma_1, ...\sigma_m, syn, i_i, i_o, ac) \quad (1)$$

where:

1) O = $\{a\}$ is the singleton alphabet (a is called spike);
2) $\sigma_1, ...\sigma_m$ are neurons of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
   a) $n_i \geq 1$ is the initial number of spikes contained by the cell;
   b) R_i is a finite set of rules of the following two forms:
      i) $E/a^r \rightarrow$ a;t, where E is a regular expression over O, r ≥ 1, and t ≥ 0;
      ii) $a^s \rightarrow \lambda$, for some s ≥ 1, with the restriction that a^s does not belong to any rule of type (i) from R_i
3) *syn* $\subseteq$ {1,2,..m} X {1,2,..m} with (i,i) $\notin$ syn for 1 $\leq$ i $\leq$ m (synapses among cells);

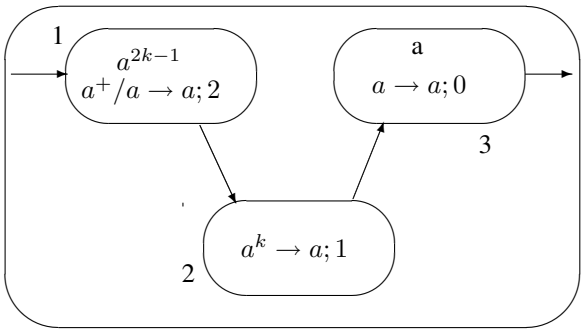

Figure 1. Illustration of the proposed variant of SNP System

4) $i_i \in \{1, 2, ..m\}$ indicates the input neuron which can obtain spikes from the local environment;
5) $i_o \in \{1, 2, ..m\}$ indicates the output neuron which releases the spikes to the environment;
6) *ac* is the initial number of spikes present in the local environment.

The rules of type (i) are called *firing rules* also referred to as *spiking rules*. The contents of the neuron (the number of spikes present in it) is indicated by a regular expression E. When the neuron is fired, r spikes are consumed and it produces a spike which will be sent to other connected neurons after t time units. The two important actions that can take place in a single step are *getting fired* and *spiking*. A neuron gets fired while using a rule $E/a^r \rightarrow a; t$, and this is possible only if the neuron contains n spikes such that $a^n \in L(E)$ and $n \geq r$. This means that the regular expression E represents exactly the contents of the neuron. The use of a rule $E/a^r \rightarrow a; t$ in a step q means *firing* in step q and *spiking* in step $q + t$. That is if $t = 0$, then the spike is produced immediately, in the same step when the rule is used. If $t = 1$, then the spike will leave the neuron in the next step, and so on. In the interval between using the rule and releasing the spike, the neuron is assumed closed, hence it cannot receive further spikes nor spike again.

The rules of type (ii) are called *forgetting rules*, s spikes are simply removed from the neuron (forgetten) when applying $a^s \rightarrow \lambda$. Like in the case of spiking rules, the left hand side of a forgetting rule must represent the contents of the neuron, that is, $a^s \rightarrow \lambda$ is applied only if the neuron contains exactly s spikes. The input neuron will take the available spikes from the environment and the output neuron will spike the spikes to the environment. The *SNP* Systems are intended here to be used primarily for controlling and communicating to achieve a centralized collaborative goal.

Illustration: Consider a *SNP* system Π_1 having 3 neurons, with labels 1,2 and 3 (Figure 1). Neuron 1 is the input neuron and neuron 3 is the output neuron. In the initial configuration we have spikes in neurons 1 and 3, and these neurons fire in the first step. The spike of neuron 3 exits the system. After firing, neuron 3 remains empty, so it cannot spike again before receiving a new spike. In turn, neuron 2 cannot fire untill collecting exactly k spikes.

After firing neuron 1 will be closed/blocked for the next 2 steps; in the third step it will release its spike, sending it to neuron 2, and in step 3 will fire again. Thus, neuron 1 fires in every third step, consuming one of the spikes; any number $n \geq 1$ of spikes in covered by the regular expression a^+. In the step $3k$, neuron 2 will receive the k^{th} spike emitted by neuron 1, hence in the next moment, $3k+1$, it will fire. The delay between firing and spiking is one time unit for neuron 2, hence its spike will reach neuron 3 in step $3k+2$, meaning that neuron 3 spikes again in step $3k+3$. Therefore, the interval between two spikes of neuron 3 will be $(3k+3)-1 = 3k+2$. There will be two spikes after this in the environment and $k-1$ spikes in neuron 1. The two spikes in the environment will be used by the neuron 1 and after next $3k+2$ steps, neuron 3 again spikes. Now neuron 1 will have only three spikes and the execution halts.

B. Distributed Spiking Neural P System

Distributed Neural P Systems were introduced in [5] in order to be able to explicitly handle the input in a distributed way. In [5] a distributed architecture based on cell like *P Systems* with their skin membranes communicating through channels rules as in a *tissue-like P system*, has been presented. This paper proposes a variant of the *Distributed P Systems* called *Distributed Spiking Neural P System* (in short as DSNP system) where n *SNP* Systems work independently on their problems, and communicate with each other using the associated skin-to-skin rules. In the proposed computing model one of the *SNP* systems is taken as a centralized *SNP* System which primarily monitors and controls the activities of the other *SNP* systems in the distributed environment.

A *Distributed Spiking Neural P System* is of the form

$$\Delta = (O, \pi_1, ...\pi_n, \pi_c, R) \tag{2}$$

where:

1) O is an alphabet of objects.
2) $\pi_1...\pi_n$ are SNP systems with O as the alphabet of objects and skin membranes(local environment) labelled with $S_1, ..S_n$ respectively.
3) π_c is the centralized control SNP system with skin membrane S_c.
4) R is a finite set of rules of the form $(S_i, u/v, S_j)$, where $1 \leq$ i, j $\leq$ n, i $\neq$ j and u,v $\in O^*$ with uv $\neq \lambda$; $|uv|$ is called the weight of the rule $(S_i, u/v, S_j)$.

The systems $\pi_1...\pi_n$ are called components of the system Δ and the rules in R are called inter-component communication rules. Rules in R are of form $(S_i, u/v, S_j)$ meaning that u number of spikes are taken from skin S_i and placed as v number of spikes in S_j.

As an illustration of the definition, consider the following simple example.

$$\Delta_1 = (\{a\}, \pi_1, \pi_2, \pi_c, R)$$

$$\begin{aligned} R \quad = \quad & \{(S_1, a^2/a^4, S_2), (S_c, a/a^2, S_1), \\ & (S_c, a^2/a^4, S_2), (S_2, a^8/a^2, S_1)\} \end{aligned}$$

This *DSNP System* consists of two *SNP Systems* π_1,π_2 and a centralized *SNP System* π_c. There are four communication rules. For example when there are two spikes in the local environment of π_1, it is transfered to the local environment of π_2 as 4 spikes and so on.

IV. Proposed Adaptive e-Learning Model

A. Design of the model

Each module in an e-course will have a layout that represents the presentation on a timeline. Given a timeline, a *Spiking Neural P System* can be created for it by running *Alogirithm 1* on the input time-line. After the modules are ordered and composed into an e-course, it can be converted into a *Distributed Spiking Neural P System* by running *Algorithm 2* over the linked list of modules selected for the e-course. This *DSNP* System will have a centralized component which will monitor and control the order of execution.The temporal layout of the presentation of the course is defined as

$$L_0 = \{SP_1, SP_2, SP_3....., SP_n\} \tag{3}$$

where $SP_i = (t_i, B_i, E_i)$, n represents the number of synchronization points in the presentation, B_i is the set of objects that start at time t_i, and E_i is the set of objects terminating at time t_i. For example consider the following time-line or temporal layout

$$\begin{aligned} L_0 \quad = \quad & \{\{(0, \{A, F\}, \{\}), (60, \{G\}, \{A\}), \\ & (300, \{B\}, \{\}), (310, \{\}, \{F, B\}), \\ & (900, \{C\}, \{\}), (1200, \{\}, \{C\})\} \end{aligned}$$

In this layout, A and F starts at 0 seconds; G starts at 60 seconds while A ends at the same time; B starts at 300 seconds; at 310 seconds F and B ends; C starts at 900 seconds and it ends at 1200 seconds.

B. Proposed algorithms

Algorithm 1 will convert the given time-line into a *SNP System*. Line 1 will set O to be a singleton alphabet set {a}. The *for* loop from line 2 to 27 will convert the time-line into a *SNP System*. *x* is the *id* of that particular module and *m* will be assigned a value greater than the *id* of all modules. Input neurons will be created in lines 4 to 14 with appropriate forgetting rules which will be used in controlling the order of presentation. Lines 15 to 25 will create the remaining neurons and the output neuron is set in line 27. This algorithm can be used to convert all the modules into corresponding *SNP Systems*.

Algorithm 2 will convert a linked list of modules which are in the form of *SNP Systems* into a *DSNP System*. O is set to a singleton alphabet set {a} in line 1. The *for* loop in lines 2-5 will add the *SNP Systems* present in the linked list to the *DSNP System* and corresponding skin-to-skin communication rules are added to maintain the link. In rule in line 4, *u* will be set to *a* and *v* to a_j. Lines 14-17 will generate communication rules that will be used by π_c to control the system. The rule in line 15 can be used to initiate any system while the rule in line 16 will be used for broadcasting *stop* message to all the modules.

Algorithm 1: *The algorithm for converting timeline to SNP*

Input: Timeline $L_0 = (SP_0, SP_1, ...SP_n)$, x
Output: SNP $\pi = (O, \sigma_1, ..., \sigma_m, syn, i_i, i_o, ac)$

Set $O \leftarrow \{a\}$
for *each* SP_i *in* T **do**
  ac $\leftarrow 0$
  if *i==0* **then**
    for *each element in* B_i **do**
      $n_i \leftarrow 0$
      $R \leftarrow R \cup \{a^x \rightarrow a; t_{i+1}\}$
      $R \leftarrow R \cup \{a^m \rightarrow \lambda\}$
      $R \leftarrow R \cup \{a^{m+x} \rightarrow \lambda\}$
      $R \leftarrow R \cup \{a^{m+1} \rightarrow \lambda\}$
      $\pi \leftarrow \pi \cup \{\sigma_i = (n_i, R_i)\}$
      $i_i \leftarrow i_i \cup \{\sigma_i\}$
    end
  end
  else
    for *each element in* B_i **do**
      $n_i \leftarrow 0$
      $e \leftarrow |E_i|$
      $R \leftarrow R \cup \{a^e \rightarrow a; t_{i+1}\}$
      $\pi \leftarrow \pi \cup \{\sigma_i = (n_i, R_i)\}$
      for *each element* σ_k *in* E_i **do**
        $syn \leftarrow syn \cup \{(k, i)\}$
      end
    end
  end
end
$i_o \leftarrow \{\sigma_n\}$

Algorithm 2: *The algorithm for converting linked list of modules to DSNP system*

Input: Linked List of modules L = $(M_1, M_2, M_3..., M_n)$
Output: DSNP system $\Delta = (O, \pi_1, ..., \pi_n, \pi_c, R)$

$O \leftarrow \{a\}$
for *each link from* π_i *to* π_j *in* L **do**
  $\Delta \leftarrow \Delta \cup \{\pi_i, \pi_j\}$
  $R \leftarrow R \cup \{(S_i, u/v, S_j)\}$
end
$O_c \leftarrow \{a\}$
$i_i \leftarrow 1$
$i_o \leftarrow 1$
$n_1 \leftarrow 0$
$ac \leftarrow 0$
$R_1 \leftarrow R_1 \cup \{a^m \rightarrow a^m; 0\}$
$\pi_c \leftarrow \pi_c \cup \{O, \sigma_1 = (n_1, R_1), i_i, i_o, ac\}$
$\Delta \leftarrow \Delta \cup \{\pi_c\}$
for *each* $\pi_j in \Delta - \{\pi_c\}$ **do**
  $R \leftarrow R \cup \{(S_c, j/j, S_j)\}$
  $R \leftarrow R \cup \{(S_c, m/m, S_j)\}$
end

C. Illustration

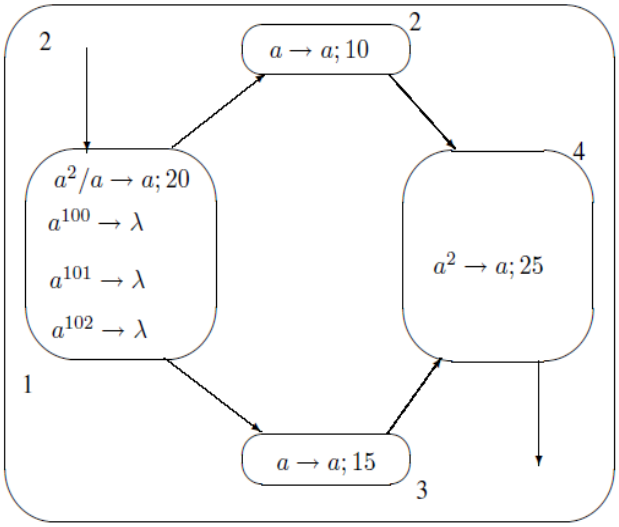

Figure 2. Module 2: An Illustration

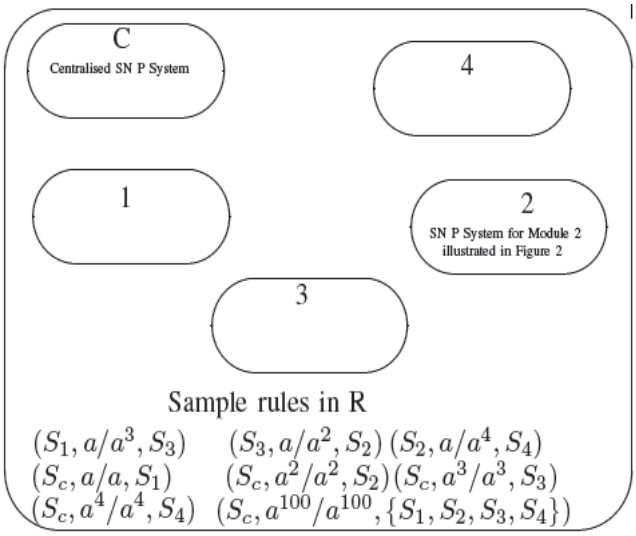

Figure 3. DSN P System: An Illustration

Consider an e-course with 4 modules. Let the order of presentation be 1,3,2 and 4. All the modules can be converted into *SNP systems* using *Algorithm 1*. Figure 2 is an illustration of the *SNP System* that represents module 2. In this illustration Module 2 has been designed to have 4 sub-modules. After the first sub-module is presented the sub-modules 2 and 3 commence their presentation and finally sub-module 4 is triggered when 2 and 3 complete. The delay d expressed in the rule of the form $a^r \rightarrow a; d$ represents the duration of presentation of the associated e-learning sub-module. Figure 3 is an illustration of the *DNSP System* where all the modules are composed into a course by grouping them and a centralized *SNP* systems created to control the order of execution, Skin rules are written to communicate from one *SNP* system to another *SNP* system. Now let us assume that module 3 in the *DNSP System* is currently being presented to a learner.

After the execution of module 3 the module 2 will be executed if the presentation is uninterrupted by either a user interaction or by dynamic need for adaptation. To change the predefined order of presentation, for instance to present module 4 instead of 2, a fixed number of *a's* say *100 a's* can be sent to all *SNP* systems from the central *SNP* system. Each *SNP* system can then use their forgetting rule and the presentation can be *paused*. Following this *4 a's* can be sent to *SNP* system 4 so that it can be trigged and presented.

V. Experimental Results

The time taken to modify the learning path is the *response time* of the system. In the proposed model, adaption is carried out by using a fixed number of communication rules and hence the time to adapt will be resonably low. We simulated the *Distributed SNP System* and calculated the mean response time required by the system for adaptation. Experiments were conducted using Fedora OS, CPU: Intel(R) Core(TM) 2, RAM: 4 Giga Byte using the programming language C. The program was compiled and run in the Open MPI environment. As the number of modules used in the course composition was increased in a linear manner the response time also increased linearly. The results are presented in the form of a graph in Figure 4. By using the proposed distributed model the Adaptive e-learning environments will appear to have a seamless presentation of personalised content.

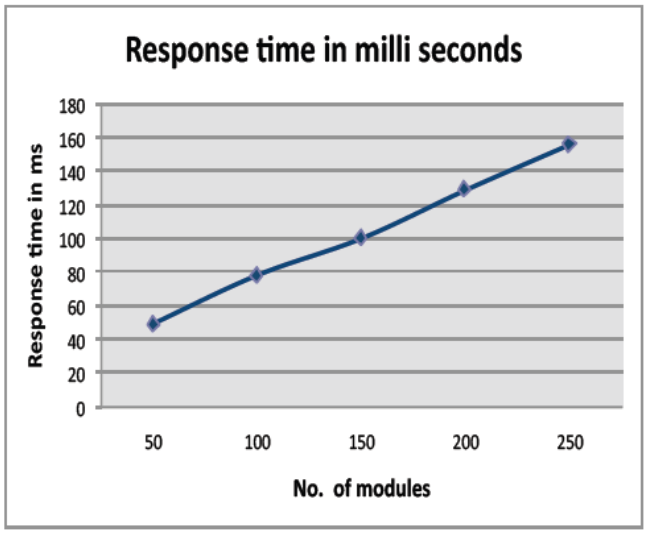

Figure 4. Effect of varying the number of learning materials

VI. Conclusion

This paper presents a membrane computing model referred to as *Distributed Spiking Neural P Systems* that can be used to represent dynamic and distributed systems effectively. The basic component of this distributed model is a variant of the existing *Spiking Neural P Systems*. The model includes a central monitoring feature and local environments for each component in the distributed system. These additional features incorporated in the proposed membrane computing model makes it functionally suitable for applications in real time and dynamic environments, as illustrated by the challenging application of adaptive e-learning.

References

[1] P. Alexandros and L.-R. Susanne. Adaptive learning environments and e-learning standards. *Electronic Journal on e-Learning*, 2:181–194, February 2004.

[2] M. Alian and M. AL-Akhras. Adalearn: an adaptive e-learning environment. In *Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications*, ISWSA '10, pages 21:1–21:7, 2010.

[3] P. Brusilovsky. Knowledgetree: a distributed architecture for adaptive e-learning. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 104–113, 2004.

[4] D. E. Dekson and E. S. M. Suresh. Adaptive e-learning techniques in the development of teaching electronic portfolios - a survey. *International Journal of Engineering Science and Technology*, 2:4175–4181, 2010.

[5] P. Gheorghe and P.-j. Mario J. Solving problems in a distributed way in membrane computing : dp systems. *International Journal of Computers, Communication and Control*, 5:238–250, 2010.

[6] M. Ionescu, G. Păun, and T. Yokomori. Spiking neural p systems. *Fundam. Inf.*, 71:279–308, February 2006.

[7] C. Mulwa, S. Lawless, M. Sharp, I. Arnedillo-Sanchez, and V. Wade. Adaptive educational hypermedia systems in technology enhanced learning: a literature review. In *Proceedings of the 2010 ACM conference on Information technology education*, SIGITE '10, pages 73–84, 2010.

[8] G. Păun. Computing with membranes. *J. Comput. Syst. Sci.*, 61:108–143, August 2000.

[9] L. Razzaq and N. T. Heffernan. Towards designing a user-adaptive web-based e-learning system. In *CHI '08 extended abstracts on Human factors in computing systems*, pages 3525–3530, New York, NY, USA, 2008. ACM.