ITHEA

International Journal INFORMATION TECHNOLOGIES & KNOWLEDGE

2010 Volume 4 Number 1

International Journal
INFORMATION TECHNOLOGIES & KNOWLEDGE
Volume 4 / 2010, Number 1

International Journal "INFORMATION TECHNOLOGIES & KNOWLEDGE" Vol.4, Number 1, 2010

Edited by the Institute of Information Theories and Applications FOI ITHEA, Bulgaria, in collaboration with:
V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
Institute of Mathematics and Informatics, BAS, Bulgaria,
Universidad Politécnica de Madrid, Spain.

Publisher ITHEA
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, www.foibg.com, e-mail: info@foibg.com
Printed in Bulgaria

ISSN 1313-0455 (printed) ISSN 1313-048X (online) ISSN 1313-0501 (CD/DVD)

IMPROVING ACTIVE RULES PERFORMANCE IN NEW P SYSTEM COMMUNICATION ARCHITECTURES

Juan Alberto de Frutos, Luis Fernández, Carmen Luengo, Alberto Arteta

***Abstract**: Membrane systems are models of computation which are inspired by some basic features of biological membranes. Transition P systems are very simple models. Many hardware and software architectures have been proposed for implementing them. In particular, there are implementations in cluster of processors, in microcontrollers and in specialized hardware. This work proposes an analysis of the P system in order to be able to reduce the execution time of a given evolution step.*

We present a solution for improving the time of working out the active rules subset of a membrane. This task is critical for the entire evolution process efficiency because it is performed inside each membrane in every evolution step. Therefore, we propose to carry out a static analysis over the P system. The collected information is used for obtaining a decision tree for each membrane. During the execution time of the P system, active rules of a membrane will be determined as a result of a classification problem from the corresponding decision tree. By incorporating decision trees for this task, we will notice some improvements.

***Keywords**: Decision Tree, ID3, Active Rules, Transition P System*

***ACM Classification Keywords**: I.2.6 Learning – Decision Tree; D.1.m Miscellaneous – Natural Computing*

Introduction

Membrane Computing was introduced by Gh. Păun in [Păun, 2000a], as a new branch of natural computing, inspired on living cells. Membrane systems establish a formal framework in which a simplified model of cells constitutes a computational device. Starting from a basic model, for Transition P systems many different variants have been considered; and many of them have been proven to be equivalent to the Turing Machine in computational complexity terms.

Membrane Computing has a massively parallel character at two levels: each region applies its rules in a parallel way and every region performs this work simultaneously. In such a way that there are some variants of P systems which can solve complete NP problems in polynomial time. Regarding implementations of Transition P systems, there are several challenges for researches in order to get real implementations of such systems. Next section shows an overview of some implementations for which the present work is going to propose an optimization.

A Transition P System evolves through transitions between two consecutive configurations that are determined by the membrane structure and multisets present inside membranes. It can be considered two sequential phases in every transition step: application of evolution rules inside membranes, and communication among membranes in the system. The present work tries to contribute with improvements in the first phase implementation.

The first task in application of evolution rules inside a membrane phase is to determine the activeness for every rule. A rule is active if it can be applied in the current evolution step; that depends on the P system state in that moment. Then, the subset of active rules will be applied in a maximal parallel and non deterministic way. There are many papers which main goal is to improve the internal parallelism of the membrane in such a way that several active rules can be applied simultaneously in a implementation. However, about optimization of the process that obtains active rules we can only mention the work done in [Fernández, 2006]. Specifically, Fernández uses decision trees in applicability of evolution rules. One of the goals of the present work is to extend those decision trees in such a way that not only applicability is considered, but also all conditions for a rule to be applied in an evolution step.

The structure of this work is as follows: firstly, related works about architectures for implementing P systems are presented; next, it will be focused on the conditions for an evolution rule to be active; in the following section we will show how a decision tree for obtaining active rules is built; then, a software implementation directed to the proposed architectures; and finally, an efficiency analysis will be carried out, together with our conclusions.

Proposed implementations for P Systems

They can be grouped into three researching lines. First of them tries to achieve a hardware specialized in membrane processing. Second line makes use of a cluster of processors in which membranes are allocated. And the last line is based on microcontrollers PIC16F88 adapted to membrane processing. As we are going to propose a software implementation for obtaining active rules in this paper, we will focus on the last two lines in the following.

Architectures based on a cluster of computers

Computers are connected by a local net. Each one houses several membranes, reaching a certain degree of parallelism. In this line we can mention the works carried out in [Syropoulos, 2003], [Ciobanu, 2004], [Tejedor, 2007], [Bravo, 2007a], [Bravo, 2007b] and [Bravo, 2008].

Syropoulos et al. propose a distributed implementation by using Java Remote Method Invocation (RMI), while Ciobanu et al. use Message Passing Interface (MPI). These authors do not carry out a detailed analysis of the communication phase; however Ciobanu noticed the possibility of network congestion when the number of membranes grows in the system. That is why the works presented by Tejedor et al. [Tejedor, 2007] and Bravo et al. [Bravo, 2007a], [Bravo, 2007b] and [Bravo, 2008] try particularly to tackle the bottleneck communication problem.

The architecture proposed in [Tejedor, 2007] avoids communication collisions and reduces the number and length of external communications. It is based on four pillars: several membranes are placed at each processor which evolve, at worst sequentially; in every processor a proxy is in charge of communications; processors are placed in a tree topology in order to minimize the number of communications; and it is established a communication order through a token. As conclusion, Tejedor et al. states that "if it is possible to make that application time be N faster times [...] the number of membranes that would be run in a processor would be multiplied by $\sqrt{N}$, the number of required processors would be divided by the same factor and the time required to perform an evolution step would improve approximately with the same factor $\sqrt{N}$. The goal of the present work fits in this context that is to reduce the application time inside a membrane.

More recently, Bravo et al. in [[Bravo, 2007a], [Bravo, 2007b] and [Bravo, 2008], have proposed three variants of the architecture proposed by Tejedor et al, in which the total evolution time has been progressively reduced.

Architectures based on microcontrollers

This line of implementing P systems has been proposed by Gutierrez et al. in [Gutierrez, 2006], [Gutierrez, 2007] and [Gutierrez, 2008]. It consists in a low cost hardware based on microcontrollers PIC16F88 that making use of external memory modules is able to solve the problem of small capacity of storage in these devices. It means a flexible solution due to microcontrollers allow to be software programmed. Figure 1 contains a picture with a real implementation. The represented microcontroller has been adapted to perform membrane execution. Besides, it has been designed to be connected up to with 254 additional microcontrollers.

Fig. 1. A microcontroller PIC16F88 for implementing P systems.

Conditions for a rule to be active

Formally, a transition P system of degree *m* is a construct of the form

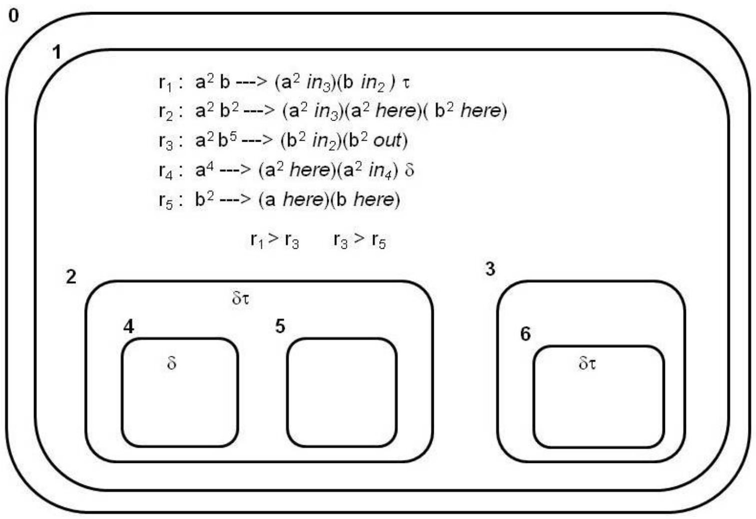
$$\Pi = (U, \mu, \omega_1, \ldots, \omega_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), i_0), \text{ where:}$$

- *U* is the alphabet of objects
- μ is a membrane structure, consisting of *m* membranes, labelled with *1,2,...., m*. It is a hierarchically arranged set of membranes, contained in a distinguished external membrane, called skin membrane. Several membranes can be placed inside a parent membrane; and finally, a membrane without any other membrane inside is said to be elementary.
- $\omega_i \mid 1 <= i <= m$ are strings over *U*, representing multisets of objects placed inside the membrane with label *i*.
- $R_i \mid 1 <= i <= m$ are finite sets of evolution rules associated to the membrane with label *i*. Rules have the form $u \rightarrow v$, $u \rightarrow v\delta$ or $u \rightarrow v\tau$, with $u \in U^+$ and $v \in (U^+ \times TAR)^*$, where $TAR=\{here, out\} \cup \{in_j \mid 1 <= i <= m\}$. Symbol δ represents membrane dissolution, while symbol τ represents membrane inhibition. ρ_i, *1 <= i <= m*, are priority relations defined over R_i, the set of rules of membrane *i*.
- i_0 represents the label of the membrane considered as output membrane.

Evolution rules able to be applied in any evolution step of the P system must accomplish three requisites at that moment: useful, applicable and active. A rule r_j in membrane *i* is useful in a evolution step if all targets are adjacent to membrane *i* and not inhibited. For example in figure 2, evolution rule r_4 in membrane 1 has membrane 4 as a target, and then it is not useful at the beginning. But if membrane 2 is dissolved then membrane 4 and 5 become adjacent to membrane 1, and rule r_4 useful. On the other hand, a useful rule r_j is applicable in membrane *i* if its antecedent is included in the membrane multiset. Finally, an applicable rule r_j is active in membrane *i* if there is no other applicable rule with higher priority in membrane *i*. Active rules will be applied in a parallel and non-deterministic way in the current evolution step.

Obtaining useful rules from usefulness states

The usefulness state concept at membranes of a P System was introduced in [Frutos, 2007]. This state allows any membrane to know the set of child membranes with which communication is feasible, that is to say, adjacent and not inhibited membranes. This set of child membranes constitute the membrane context, which changes dynamically as membranes are dissolved or inhibited in the P system.



0
1
r1 : a2 b ---> (a2 in3)(b in2) τ
r2 : a2 b2 ---> (a2 in3)(a2 here)(b2 here)
r3 : a2 b5 ---> (b2 in2)(b2 out)
r4 : a4 ---> (a2 here)(a2 in4) δ
r5 : b2 ---> (a here)(b here)
r1 > r3 r3 > r5
2
δτ
4
δ
5
3
6
δτ

Fig.2. Example of transition P System

The set of usefulness states for a membrane *i* in a Transition P system can be obtained statically at analysis time, as it is shown in [Frutos, 2007]. A usefulness state *j* in membrane *i*, q_j^i, represents a valid context for that membrane, $C(q_j^i)$, which means a context that can be reached in any evolution step. Figure 2 represents an example of Transition P system. Only rules associated to membrane 1 are detailed. Symbol δ in membranes 2, 4 and 6 represents the possibility of these membranes to be dissolved by application of some rules inside them. The symbol τ represents the possibility of inhibition for membranes 2, 3 and 6 by the same cause. Usefulness states for membrane 1, together with their corresponding contexts, are depicted in first and second column of table 1.

Usefulness states	Context	Useful rules	Useful rules when permeable
q_0^1	{2,3}	r_1, r_2, r_5	r_3
q_1^1	{4,5,3}	r_2, r_4, r_5	
q_2^1	{5,3}	r_2, r_5	
q_3^1	{3}	r_2, r_5	

Table 1. Usefulness states for membrane 1

From a given usefulness state q_j^i, it can be obtained statically the set of useful rules in the following way:

- An evolution rule $u \to v\xi, where\ \xi \in \{\delta, \tau, \lambda\}$ is useful in q_j^i if $\forall\ target\ in_k \in v, k \in C(q_j^i)$, what it means that all child targets are included in the context.
- In addition, If target *out* belongs to *v* then the evolution rule will be useful only if membrane *i* is permeable, otherwise communication with its father membrane is not feasible.

Third and fourth column in table 1 represents useful rules for every usefulness states. Rule r_3 will be useful in q_0^1 only if membrane 1 is not inhibited. To sum up, useful rules can be obtained from the usefulness state and the permeability state of the membrane.

Tables defining transitions among usefulness states can be also defined at analysis time, as it is explained in [Frutos, 2007]. During system execution, a membrane makes a transition when its context changes, i.e. a child membrane changes its permeability. In such a way that useful rules for any membrane are always available from its usefulness and permeability states.

Regarding implementations, problems arise when membranes have a high number of states, which causes transition tables to grow up. That is why in [Frutos, 2007] transition tables are avoided by encoding usefulness states; hence transitions are carried out directly in the code. According to this idea, an important definition is introduced:

Total Context for membrane *i*. It is the set made up of all membranes that eventually can become children of membrane i. Therefore, all contexts are included in the total context.

$$TC(i) = Children(i) \bigcup_{i_k \in ChildrenDis(i)} TC(i_k)$$

Children of a membrane *i* is the set of all children in μ, the initial structure; and *ChildrenDis* for a membrane *i* is the set of all children in μ that can be dissolved. For instance, in our example of figure 2, TC(1) = {2, 4, 5, 3}.

Each one of usefulness states for a membrane *i* is encoded by *TC(i)*, depending on its context, with binary logic. The value 1 represents that the membrane belongs to the state context. Thus, the usefulness state q_0^1 in our example, which represents the context {2, 3}, is encoded as 1001. Transitional logic among usefulness states is described in [Frutos, 2007] without making use of transition tables. Besides, we want to emphasize the work carried out in [Frutos, 2008], which describes in an exhaustive way an implementation for usefulness states updating in some architectures based on a cluster of processors.

Decision trees for active rules

A decision tree is a tool that allows determining the class which one element belongs to, depending on the values of some attributes or properties of the element. Each non-leaf node of a decision tree corresponds to an input attribute, and each arc to a possible value of that attribute. A leaf node corresponds to the expected value of the output attribute, that is to say, the element classification. An element is classified by starting at the root node of the decision tree, testing the attribute specified by this node and moving down the tree branch corresponding to the value of the attribute. This process is repeated until a leaf node is reached.

There are a lot of algorithms to generate decision trees. Specifically ID3 is an outstanding algorithm belonging to TDIDT family (Top-Down Induction of Decision Trees). It is based on entropy information, a measure of uncertainty. Entropy is used to determine how informative a particular input attribute is about the output attribute. By using this, algorithm ID3 generates an optimum decision tree from a non incremental set of instances and without repetitions.

E	a_1	.....	a_j	C
e_1	v_{11}	.....	v_{1j}	c_1
⋮	⋮		⋮	⋮
e_i	v_{i1}	.....	v_{ij}	c_i

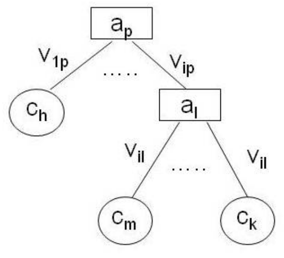


Fig. 3. Example of instances for ID3 algorithm, together with the obtained tree

Algorithm ID3 uses as input a set of data about instances such as figure 3. Each row represents an instance (e_i), which includes a value (v_{ij}) for each attribute (a_j), and the decision or classification for that instance (c_i). Then ID3 algorithm obtains a decision tree making use of entropy property of attributes. Fernandez et al. in [Fernández, 2006] proposed incorporating decision trees in the calculus of evolution rules applicability. A decision tree for a membrane is able to classify the current multiset of objects in the membrane, obtaining the subset of applicable rules for the current evolution step. In this way, it is reduced the number of checks necessary for determining the subset of applicable rules. Nevertheless, Fernandez et al. do not take into account conditions for rules to be useful and active, which are also necessary to determine the subset of rules that can be applied in an evolution step. The present work, supporting in usefulness states analysis, tries to extend the advantage of decision trees. In such a way that a decision tree for a membrane obtained at analysis time can determine the subset of active rules at execution time. With this aim we make the following considerations:

- The elements to be classified are all different states or situations of the membrane in the P system at any time. These situations are characterized by:
  - The membrane usefulness state, due to the set of useful rules depends on it.
  - The membrane permeability state, due to the set of useful rules depends on it. However, only information about inhibition is needed.
  - The membrane multiset in order to determine the set of applicable rules.

  Active rules subset is determined from applicable rules bearing in mind priority relations among them. Nevertheless, these relations are static, that is to say, they do not depend on the state of the P system.
- The set of attributes A_i is established as properties necessary to define a situation of the membrane *i* in the P System. Specifically, the set A_i consist of the following attributes:
  - Attributes necessary to set up the usefulness state of membrane *i*. Thus, there will be one attribute for each membrane belonging to membrane *i* total context. The associated value will be true if the represented child membrane belongs to the current usefulness state.

    $$A_i \supset \{a \equiv m_j \mid j \in TC(i)\}$$
  - One attribute more to determine inhibition in the permeability state of membrane *i*. The value true will be used when membrane *i* is inhibited.

    $$A_i \supset \{a \equiv I\}$$
  - Furthermore, as proposed in [Fernández, 2006], we consider another set of attributes for applicability of rules. These attributes represents the set of weight checks between objects from the membrane multiset and objects from antecedent of every evolution rule. Neither repetitions nor checks with zero are considered. An attribute value will be true in case of the multiset weight is greater or equal to antecedent weight in the evolution rule.

    $$A_i \supset \{a \equiv |\omega|_u \geq k \;\big|\; |input(r)|_u = k \wedge k \neq 0 \;\; \forall r \in R \; \forall u \in U\}$$

  $|\omega|_u$ represents the weight of a symbol *u* in ω, and *input(r)* represents the antecedent multiset of rule *r*.
- Finally, classification will be carried out into subsets of evolution rules, i.e. a membrane *i* state will be classified into the corresponding subset of active rules. Hence, classes to be considered are:

  $$C = \{c \equiv R_{AC} \big| \; R_{AC} \in \mathrm{P}(R)\}$$

To obtain a decision tree in analysis time under these conditions, we have to start from a set of data, representing instances. Every instance consists of the corresponding attributes and classification. Besides, training data will be complete, that is, every possible membrane situation will be represented as an instance. Thus, the decision tree will be used for classifying the current membrane state, and the resulting class will be correct anyway.

As every possible membrane state has to be considered, and taking into account the set of attributes A_i, the amount of instances in the training data for membrane *i* is the following:

$$|E_i| = |Q_i| * \prod_{u \in U} \left(|C_i^u| + 1 \right)$$

Where $|Q_i|$ is the number of usefulness states for membrane *i*, and $|C_i^u|$ is the number of different checks with symbol *u* in any rule antecedent of membrane *i*, that is, attributes with the form $|\omega|_u >= k$. Besides, if membrane *i* has inhibiting capability, this value has to be multiplied by 2 in order to consider attribute *I*. Anyway, what it is important to note is that the needed information for every instance is available at analysis time.

Coming back to the example of P system introduced in figure 2, attributes for membrane 1 are the following:

- As *TC(1)* = {2,4,5,3}, four attributes are needed: m_2, m_4, m_5, m_3.
- As membrane 1 has inhibiting capability by means of r_1, an attribute *I* has to be included.
- Finally, the set of different checks for applicability in evolution rules are: $|\omega|_a >= 4$, $|\omega|_a >= 2$, $|\omega|_b >= 5$, $|\omega|_b >= 2$ and $|\omega|_b >= 1$. Therefore, we consider another five attributes.

The resulting training data are shown in table 2. Each instance is named with the corresponding usefulness state, followed with *I* (inhibited) or *NI* (not inhibited), and finally the applicability state. As example, *1001NIa²b⁵*, represents the instance with the following properties for membrane 1: the usefulness state is *1001* (membranes 2 and 3 as context), the permeability state is permeable, the amount of objects *a* in *ω* is greater than 2, but not greater than 4; and finally, there are 5 or more objects *b* in *ω*.

E	m_2	m_4	m_5	m_3	I	$\|\omega\|_a>=4$	$\|\omega\|_a>=2$	$\|\omega\|_b>=5$	$\|\omega\|_b>=2$	$\|\omega\|_b>=1$	C
$1001NIa^0b^0$	Yes	No	No	Yes	No	No	No	No	No	No	∅
$1001NIa^0b^1$	Yes	No	No	Yes	No	No	No	No	No	Yes	∅
$1001NIa^0b^2$	Yes	No	No	Yes	No	No	No	No	Yes	Yes	$\{r_5\}$
$1001NIa^0b^5$	Yes	No	No	Yes	No	No	No	Yes	Yes	Yes	$\{r_5\}$
$1001NIa^2b^0$	Yes	No	No	Yes	No	No	Yes	No	No	No	∅
$1001NIa^2b^1$	Yes	No	No	Yes	No	No	Yes	No	No	Yes	$\{r_1\}$
$1001NIa^2b^2$	Yes	No	No	Yes	No	No	Yes	No	Yes	Yes	$\{r_1,r_2\}$
$1001NIa^2b^5$	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	$\{r_1,r_2\}$
$1001NIa^4b^0$	Yes	No	No	Yes	No	Yes	Yes	No	No	No	∅
$1001NIa^4b^1$	Yes	No	No	Yes	No	Yes	Yes	No	No	Yes	$\{r_1\}$
$1001NIa^4b^2$	Yes	No	No	Yes	No	Yes	Yes	No	Yes	Yes	$\{r_1,r_2\}$
$1001NIa^4b^5$	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	$\{r_1,r_2\}$
…………	……..	……..	…….	…….	……	…….	……	……	……	……	…….

Table 2. Instances of membrane 1 for ID3 algorithm

In table 2 only a few number of instances for membrane 1 are represented. The total amount can be worked out as it is shown below:

$$|E_1| = 2 \cdot |Q_1| \cdot \prod_{u \in U} \left(|C_1^u| + 1 \right) = 2 \cdot |Q_1| \cdot (|C_1^a| + 1) \cdot (|C_1^b| + 1) = 2 \cdot 4 \cdot 3 \cdot 4 = 96 \text{ instances}$$

Instance classification

Every instance is classified into the corresponding set of active rules. This task is carried out at analysis time as:

$$C = Max\ (Useful\ Rules \cap Applicable\ Rules)$$

- Firstly, useful rules are obtained from the current usefulness state and the current permeability state, as it is shown in table 1.
- Secondly, attributes related to checks of objects weights in multiset determine the applicability property of rules, as it is shown in [Fernandez, 2006]. The intersection with the set of useful rules gets the set of useful and applicable rules (*C’*).
- Lastly, priorities among rules have to be considered in order to get the active rules subset, which implies to work out the maximal over the priority relation of *C’*. With this aim, we have to work out a transitivity matrix (*M*) expressing the priority relation. Then *C’* has to be multiplied by *M*, obtaining the set of rules with less priority that any rule in *C’*. Therefore, the maximal of *C’* is obtained by removing those rules:

$$C = Max(C') = C' \wedge \neg\ (C' \times M)$$

For example, we are going to classify instance *1001NIa²b⁵*. First, as usefulness state is *1001* the corresponding state in table 1 is q_0^1. With normal permeability, the set of useful rules is $\{r_1, r_2, r_3, r_5\}$. Second, with two or more symbols *a* (but not four or more) and five or more symbols *b*, the set of applicable rules is $\{r_1, r_2, r_3, r_5\}$. Therefore, intersection between useful an applicable rules is $C' = \{r_1, r_2, r_3, r_5\}$.

With regard to priorities, membrane 1 has two relations: $r_1 > r_3$ and $r_3 > r_5$, obtaining the following transitivity matrix *M*:

$$\begin{array}{c|ccccc} & r_1 & r_2 & r_3 & r_4 & r_5 \\ \hline r_1 & 0 & 0 & 1 & 0 & 1 \\ r_2 & 0 & 0 & 0 & 0 & 0 \\ r_3 & 0 & 0 & 0 & 0 & 1 \\ r_4 & 0 & 0 & 0 & 0 & 0 \\ r_5 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Now we represent $C' = \{r_1, r_2, r_3, r_5\}$ with binary logic, obtaining *11101*. Then C is obtained as follows:

$$C = Max(11101) = (11101) \wedge \neg((11101) \times M) = 11000 \quad \text{---->} \quad \{r_1, r_2\}.$$

As conclusion, all necessary data for every instance are available at analysis time in the P system. Thus ID3 algorithm can be applied obtaining a decision tree, which determines classification of any situation of the P system into an active rules subset. Specifically, decision tree corresponding to membrane 1 in our example is depicted in figure 4.

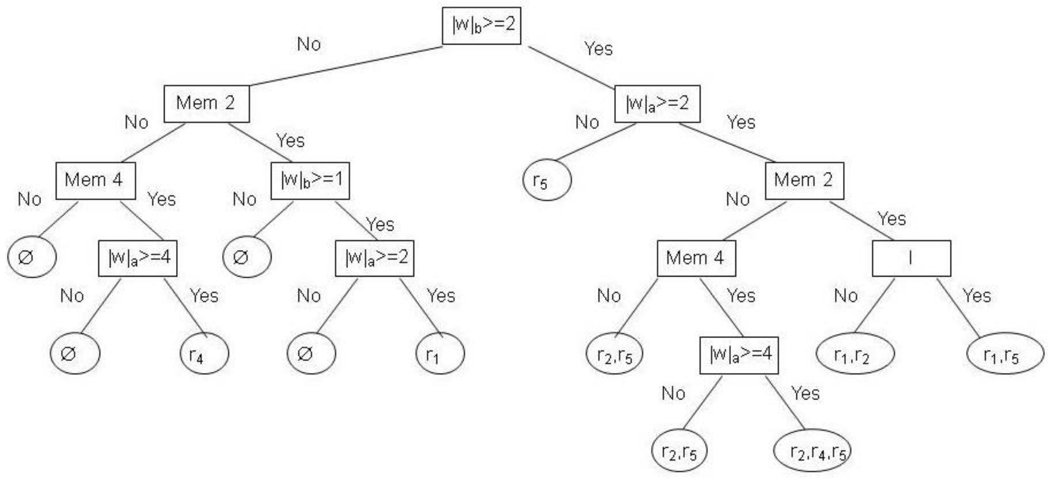


Fig.4. Decision tree obtained by ID3 algorithm for membrane 1

A software implementation for decision trees

Computation of active evolution rules subset is critical for the whole evolution process efficiency, because it is performed inside each membrane in every evolution step. That is why software implementation should be carried out in a low level with an assembly language. Efficiency is considerably increased, mainly by using registers in a convenient way. PC assembly language utilizes some registers with general purpose, such as EAX, EBX, ECX, EDX, with 32 bits. Below we show how these registers are used in our implementation.

EAX: it will be used to return the subset of active rules with binary logic. Coming back to membrane 1 of our example in figure 1, only five lower bits will be used for representing rules r_5, r_4, r_3, r_2 and r_1, respectively.

EBX: usefulness state and permeability state will be stored in this register. In this way, only six lower bits of EBX will be used for membrane 1 in our example. Four of them for storing the current usefulness state; and the last two bits for membrane inhibition and dissolution respectively, as it is represented in figure 5.

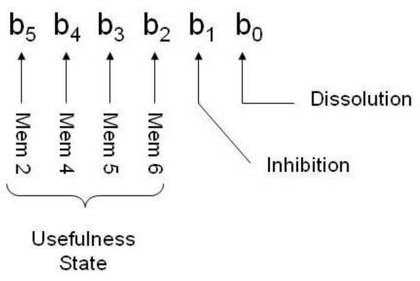


Fig. 5. Usefulness and permeability states in EBX register.

ECX: it will be used for storing ω, the membrane multiset. Specifically, in our example:

- CH (higher 16 bits) will contain $|\omega|_a$, the amount of symbols *a* in ω.
- CL (lower 16 bits) will contain $|\omega|_b$, the amount of symbols *b* in ω.

```
    CMP CL, 2          ; |w|b>=2 ?
    JAE et1
    MOV EAX, 0000H     ; ∅
    TEST EBX, 0020H    ; Mem 2 ?
    JNZ et2
    TEST EBX, 0010H    ; Mem 4 ?
    JZ et3
    CMP CH, 4          ; |w|a>=4 ?
    JB et3
    MOV EAX, 0008H     ; {r4}
et3 RET
et2 CMP CL, 1          ; |w|b>=1 ?
    JB et4
    CMP CH, 2          ; |w|a>=1 ?
    JB et4
    MOV EAX, 0001H     ; {r1}
et4 RET

et1 MOV EAX, 0010H     ; {r5}
    CMP CH, 2          ; |w|a>=2 ?
    JB et5
    TEST EBX, 0020H    ; Mem 2 ?
    JNZ et6
    MOV EAX, 0012H     ; {r2, r5}
    TEST EBX, 0010H    ; Mem 4 ?
    JZ et7
    CMP CH, 4          ; |w|a>=4 ?
    JB et7
    MOV EAX, 001AH     ;{r2,r4,r5}
et7 RET
et6 MOV EAX, 0003H     ; {r1, r2}
    TEST EBX, 0002H    ; I ?
    JZ et5
    MOV EBX, 0011H;    ; {r1, r5}
et5 RET
```

Fig. 6. Assembly code for the membrane 1 decision tree

Taking into account these conditions, an algorithm can be generated automatically for a given decision tree at analysis time. Going on with the decision tree represented in figure 4, the corresponding code which obtains the active rules subset for membrane 1 can be the represented in figure 6.

In general, for implementing a leaf node of the decision tree, assembly code stores previously the suitable set of active rules in EAX. However, optimization in the code is possible. For instance, we can observe in the decision tree of figure 4 four leaves which subset is ∅, whereas only one storing instruction for that subset in EAX can be appreciated in the assembly code. This improvement can be achieved by carrying out a previous analysis into the decision tree. The goal is to determine the best points in decision tree for loading an active rules subset in EAX. This analysis is performed with a two-way algorithm. Firstly, the decision tree is analyzed upward, carrying out the process represented in figure 7 in every non-leaf node *i*.

```
j <--- Child_Left (i)
k <--- Child_Right (i)
IF Is_Leaf(j) OR Is_Leaf(k) THEN BEGIN
        IF Is_Leaf(j) THEN EAX(i) <--- Active_Set(j)
        IF Is_Leaf(k) THEN EAX(i) <--- EAX(i) +
Active_Set(k)
    END
ELSE EAX(i) <--- Common_Sets (EAX(j), EAX(k))
```

Fig. 7. Bottom up algorithm for determining subsets to be preloaded in EAX

After performing this process over the decision tree in figure 4, results are shown in figure 8. In every non-leaf node it is represented subsets of active rules that could be stored in EAX when implementing that node. Now, decision tree is analyzed from root to leaves choosing a subset in every non-leaf node. When a non-leaf node is analyzed, it is important to take into account the subset already loaded in the upper node. If this subset is considered as a possible subset to be loaded in the current node, then no loading is performed. Results are depicted in figure 9, corresponding with the assembly code in figure 6.

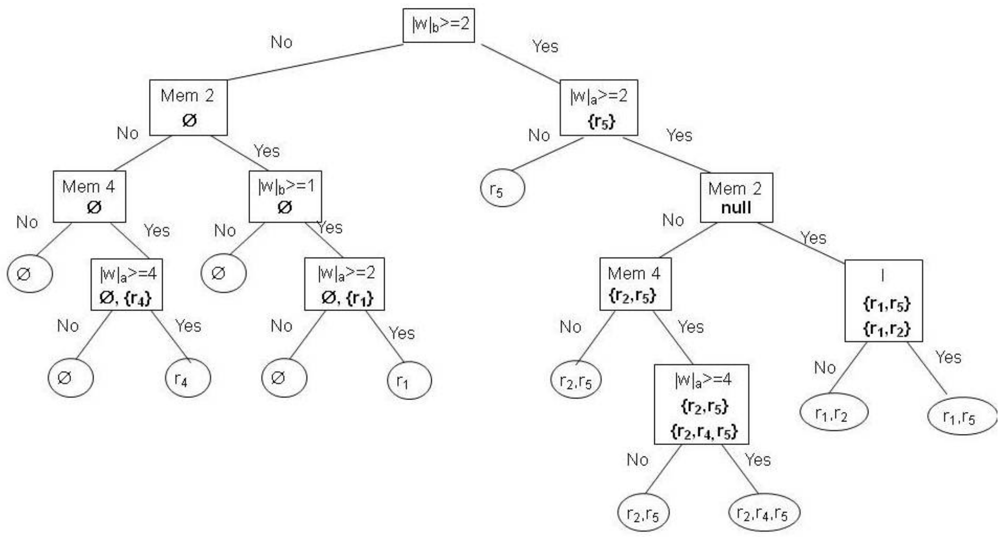


Fig. 8. Bottom up analysis in decision tree for membrane 1

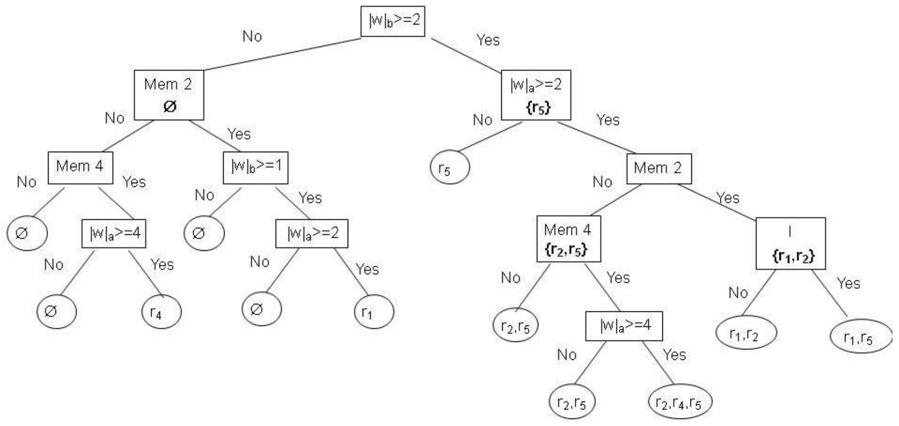


Fig. 9. Top down analysis in decision tree for membrane 1

To sum up, a P system is statically analyzed, obtaining a particular assembly code for each membrane. As result, this software is a suitable solution for architectures based on a cluster of processors, such as [Syropoulos, 2003], [Tejedor, 2007], [Bravo, 2007a], [Bravo, 2007b] and [Bravo, 2008]. In all these architectures a membrane evolves in a single process. Such process would contain the assembly code obtained as proposed in this paper; and the membrane would get active rules from the decision tree in every evolution step.

As regards architecture based on microcontrollers PIC16F88, proposed by Gutierrez et al. in [Gutierrez, 2006], [Gutierrez, 2007] and [Gutierrez, 2008], we have to bear in mind that these devices are software programmed. More precisely, in [Gutierrez, 2007] Gutierrez et al. made use of the Microchip MPLAB IDE integrated Environment, in which the tool MPASM allows to work with assembly language. Therefore, this architecture could get profit from the solution proposed in this paper. Besides, this solution could especially fit here, due to the lack of memory for data in microcontrollers. The implementation proposed in this paper does not require space for additional data; moreover, it reduces data by avoiding transitivity matrix for priorities.

Analysis of results

This section presents an analysis of efficiency in the proposed solution based on decision trees. In this way, we are going to compare it with classical solutions for obtaining active rules. This kind of solutions consists of three sequential algorithms, obtaining useful, applicable and active evolution rules respectively.

Classical algorithm for useful rules (Figure 10). Complexity order is *O(n)*, where *n* represents the amount of checks. In the worst case $n = |R| * (|TC| + 1)$. For every evolution rule algorithm must check every target in the membrane usefulness state. Additionally, it has to be checked for every evolution rule if there is a target *out* when the membrane is inhibited. As usefulness state length is equal to total context length, it has to be performed $|TC| + 1$ checks for every rule.

```
$R_U$ <--- ∅
FOR EACH $r_i$ in R DO BEGIN
    j <--- 1
    useful <--- true
    WHILE j <= length (TC) AND useful DO BEGIN
        IF (TC(j) $\subset$ TAR($r_i$) ) AND UsefulnessState(j) = 0        (* Impossible target *)
                                THEN useful <--- false
        j <--- j + 1
    END
    IF useful THEN
        IF (out $\not\subset$ TAR($r_i$) ) OR (PermeabilityState $\neq$ Inhibited)
            THEN $R_U$ <--- $R_U \cup \{ r_i \}$
END
```

Fig 10. Algorithm for obtaining useful rules

Classical algorithm for applicable rules (Figure 11). It is also an algorithm with *O(n)* complexity. In this case, the amount of checks is equal to $|R|*|U|$ in the worst case. For every rule, the algorithm must check the weight of every symbol in the multiset.

```
$R_A$ <--- $\emptyset$
FOR EACH $r_i$ in $R_U$ DO BEGIN
    applicable <--- true
    FOR EACH u in U DO
        IF (| input($r_i$) $|_u$ > | $\omega$ $|_u$ ) THEN BEGIN      (* Not enough symbols in $\omega$ *)
            applicable <--- false
            break
        END
    END
IF applicable THEN $R_A$ <--- $R_A \cup \{ r_i \}$
END
```

Fig. 11. Algorithm for obtaining applicable rules

Classical algorithm for active rules. As it is detailed in a previous section, this algorithm calculates the maximal over the priority relation of useful and applicable rules. With this aim, it firstly multiplies a vector with these rules by the transitivity matrix expressing priorities. Then, the resulting subset of rules has to be eliminated, obtaining the corresponding maximal. Multiplication complexity is *O(n^2)*, where *n* is equal to $|R|$; meanwhile complexity for subtracting rules is *O(n)*, where *n* is also $|R|$.

As regards solution based on decision trees proposed in this paper, the implemented algorithm starts in the root and finishes in a leaf. Therefore computational complexity of the algorithm is *O(n); n* represents the amount of check operations, that in the worst case it is the longest branch of the decision tree. At the most, a branch length will be the amount of attributes considered in ID3 algorithm, due to each step of ID3 algorithm chooses an attribute from the subset of not chosen attributes. Then, the greatest amount of checks is equal to $|TC|+1+\sum_{u\in U}|C_i^u|$ i.e. the amount of attributes. Table 3 summarizes these results.

	Useful rules	Applicable rules	Active rules
Classical algorithms	*O(n)* $n=\|R\|*(\|TC\|+1)$	*O(n)* $n=\|R\|*\|U\|$	*O(n^2)* + *O(n)* $n=\|R\|$
Decision tree	*O(n)* $n=\|TC\|+1+\sum_{u\in U}\|C_i^u\|$		

Table 3. Complexity order of algorithms.

From these results, we can conclude that decision tree solution performs a fewer number of operations than classical solutions, as it is shown in the following reasoning:

- Decision tree solution performs $|TC|+1$ check operations for obtaining the useful rules subset, meanwhile classical algorithm requires $|R|*(|TC|+1)$ check operations.

- On the other hand, for applicability of rules, decision tree performs $\sum_{u \in U} |C_i^u|$ checks in the worst case, avoiding repeated and redundant checks. For instance, decision tree will never check if $|\omega|_a \geq 3$ after determining that $|\omega|_a \geq 5$. So, the amount of check operations in decision trees will never be greater than $|R| * |U|$.
- There is an obvious difference between both solutions in the analysis of priorities for obtaining active rules. Decision tree does not need any operation, whereas classical algorithms must multiply by the transitivity matrix.

In the following, we are going to apply the previous comparative to a set of P systems published in [Păun, 2000a] and [Păun, 2000b], which are listed in table 4. For every P system, table 4 shows the amount of operations to be performed by classical algorithms and decision tree respectively. Values of the table refer to the addition of the values for every membrane of the P system; and considering always the worst case.

	Classical Algorithms			Decision Tree
P-System	Amount of checks for useful rules	Amount of checks for applicable rules	Amount of operations for active rules	Amount of checks for active rules.
First example in [Păun, 2000a]	24	24	20	10 (15'62%)
Decidability (*n mod k = 0*) in [Păun, 2000a]	6	15	12	7 (21,21%)
Generating n^2 | *n>=1* in [Păun, 2000a]	15	18	20	6 (11,32%)
Generating n^2 | *n>=1* in [Păun, 2000b]	14	19	20	7 (13,20%)

Table 4. Amount of operations in classical algorithms versus decision trees

From table 4 we conclude that decision trees carry out a fewer amount of operations. Even if we consider only check operations, classical algorithms need a greater amount of them. If we also consider operations that classical algorithm perform for applying priorities, the resulting value is significantly lower in decision trees. Specifically, the total amount of operations in decision trees vary from 11,32% to 21,21% of the total amount of operations in classical algorithms. Besides, as the code for decision trees has been specifically optimized for each membrane, we can conclude that decision trees allow a significant reduction in the required time for applying rules inside membranes.

Finally, we are going to add some remarks about memory space required by both types of solutions.

- A decision tree handicap is the need to keep a different code for each membrane; meanwhile classical algorithms make use of a common code. Thus, in architectures based on a cluster of processors, each processor must house as many decision trees as membranes are allocated on it.
- On the other hand, a decision tree advantage is that it does not require space for the transitivity matrix expressing priorities.

- Also, we have to mention that decision tree could grow up significantly. For instance, a membrane with a great deal of rules could mean a great deal of attributes in table for ID3 algorithm; and consequently, a big decision tree. In this case, it may happen that, depending on the architecture used for implementing the P system, memory space could be insufficient; and then, classical algorithms had to be chosen. Anyway, as decision tree code is obtained at analysis time, the best solution for a concrete P system can be also determined at analysis time.

Conclusions

We have proposed a static analysis of transition P systems in order to get a specific code for each membrane. This code can determine the subset of active rules in every evolution step in an optimal way. The analysis we propose consists mainly of determining the set of usefulness states for each membrane. These states are shown in [Frutos, 2007] and [Frutos, 2008]; Furthermore, the analysis considers all situations of applicability for rules, depending on antecedents, as it is explained in [Fernández, 2006]; The analysis also takes into account priorities among rules. As a result, a decision tree is obtained that gathers all information collected during the analysis. In the end, the decision tree is transformed into an optimized assembly code. This also occurs during the analysis.

Some architecture for implementing transition P systems can take advantage of this implementation such as most of the architectures based on a cluster of processors and architectures based on microcontrollers.

An analysis has been carried out in this paper, in which we conclude that decision tree is significantly more efficient than classical solutions for determining active rules. According to the work carried out by Tejedor et al. in [Tejedor, 2007], it is possible to obtain improvements in some architectures based on a cluster of processors. Improvements such as reduction of the total time for an evolution step increase of the number of membranes that could run in a processor and reduction of the required processors.

Bibliography

[Bravo, 2007a] G. Bravo, L. Fernández, F. Arroyo. J.A. Frutos, A Hierarchical Architecture with Parallel Communication for Implementing P systems, Fifth International Conference Information Research and Applications (i. TECH 2007), June 2007, Varna, Bulgary, 168-174.

[Bravo, 2007b] G. Bravo, L. Fernández, F. Arroyo. J. Tejedor, Master Slave Distributed Architecture for Membrane Systems Implementation, 8th WSEAS Int. Conf. on Evolutionary Computing (EC'07), June 2007, Vancouver, Canada.

[Bravo, 2008] G. Bravo, L. Fernández, F. Arroyo, M. A. Pea, Hierarchical Master-Slave Architecture for Membrane Systems Implementation, 13th Int. Symposium on Artificial Life and Robotics (AROB '08), Feb 2008, Beppu, Oitia (Japan).

[Ciobanu, 2004] G.Ciobanu, W. Guo, P Systems Running on a Cluster of Computers. Workshop on Membrane Computing (Gh. Păun, G. Rozemberg, A. Salomaa Eds.), 2004, LNCS 2933, Springer, 123-139

[Fernández, 2006] L. Fernández, F. Arroyo, I. García, G. Bravo, Decision Trees for Applicability of Evolution Rules in Transition P Systems, Fourth International Conference Information Research and Applications (i. TECH 2006) June 2006, Varna, Bulgary.

[Frutos, 2007] J. A. Frutos, L. Fernández, F. Arroyo, G. Bravo, Static Analysis of Usefulness States in Transition P Systems, Fifth International Conference, Information Research and Applications (I.TECH 2007), June 2007, Varna, Bulgary. 174-182.

[Frutos, 2008] J. A. Frutos, F. Arroyo, A. Arteta. Usefulness States in New P System Communication Architectures, Ninth Workshop on Membrane Computing (WMC9), July 2008, Edinburgh, Scotland.

[Gutierrez, 2006] A. Gutierrez, L. Fernández, F. Arroyo, V. Martínez, A Design of a Hardware Architecture Based on Microcontrollers for the Implementation of Membrane Systems, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, 2006

[Gutierrez, 2007] A. Gutierrez, L. Fernández, F. Arroyo, S. Alonso, Hardware and Software Architecture for Implementing Membrane Systems: A Case of Study to Transition P Systems, The DNA International Meeting on DNA Computing (DNA13), June 2007, Memphis, USA.

[Gutierrez, 2008] A. Gutierrez, L. Fernández, F. Arroyo, Suitability of Using Microcontrollers in Implementing New P System Communication Architectures, 13th Int. Symposium on Artificial Life and Robotics (AROB '08), Feb. 2008, Beppu, Oitia (Japan).

[Păun, 2000a] Gh. Păun, Computing with Membranes, Journal of Computer and System Sciences, 61(1), 2000, 108-143.

[Păun, 2000b] Gh. Păun, G. Rozemberg, A Guide to Membrane Computing, Theoretical Computer Science, vol 287, 2000, 73-100.

[Syropoulos, 2003] A.Syropoulos, E.G. Mamatas, P.C. Alliomes, K.T. Sotiriades, A Distributed Simulation of P Systems. Workshop on Membrane Computing, Tarragona, Spain, 455-460.

[Tejedor, 2007] J. Tejedor, L. Fernández, F. Arroyo, G.Bravo, An Architecture for Attacking the Bottleneck Communication in P Systems, 12th Int. Symposium on Artificial Life and Robotics, Jan 2007, Beppu, Oita, Japan, 500-505.

Authors' Information

Juan Alberto de Frutos Velasco – *Dpto. Lenguajes, Proyectos y Sistemas Informáticos (LPSI) de la Escuela Universitaria de Informática (EUI) de la Universidad Politécnica de Madrid (UPM); Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: jafrutos@eui.upm.es*

Luis Fernández Muñoz – *Dpto. Lenguajes, Proyectos y Sistemas Informáticos (LPSI) de la Escuela Universitaria de Informática (EUI) de la Universidad Politécnica de Madrid (UPM); Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: setillo@eui.upm.es*

Carmen Luengo Velasco – *Dpto. Lenguajes, Proyectos y Sistemas Informáticos (LPSI) de la Escuela Universitaria de Informática (EUI) de la Universidad Politécnica de Madrid (UPM); Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: cluengo@eui.upm.es*

Alberto Arteta Albert – *Dpto. Lenguajes, Proyectos y Sistemas Informáticos (LPSI) de la Escuela Universitaria de Informática (EUI) de la Universidad Politécnica de Madrid (UPM); Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: aarteta@eui.upm.es*

IMPLEMENTING TRANSITION P SYSTEMS

Santiago Alonso, Luis Fernández, Víctor Martínez

***Abstract**: Natural computing is a whole area where biological processes are simulated to get their advantages for designing new computation models. Among all the different fields that are being developed, membrane computing and, more specifically, P Systems, try to get the most out of the biological cell characteristics and of the chemical processes that take place inside them to model a new computation system.*

There have been great advances in this field, and there have been developed a lot of works that improved the original one, developing new ideas to get the most of the different algorithms and architectures that could be used for this new model. One of the most difficult areas is the actual implementation of these systems. There are some works that try to implement P Systems by software simulations and there are some more that design systems that implement them by using computer networks or specific hardware like microcontrollers. All these implementations demonstrate their validity but many of them had the lack of some main characteristics for P Systems.

As continuation for some earlier published works, present work pretends to be the exposition of the design for a complete new hardware circuit that may be used to develop a P System for general purpose, complying with the two main characteristics that we consider more important: a high level of parallelism (which does these systems specially indicated to solve NP problems) and the fact that they should be non deterministic.

***Keywords**: Transition P System, membrane computing, circuit design.*

***ACM Classification Keywords**: D.1.m Miscellaneous – Natural Computing*

Introduction

Membrane computing was first introduced by Georghe Păun in 1998 in his article "Computing with Membranes" ([Păun, 1998]). In it, Păun proposes a new computation model based on the structure and behavior of the biological cells existing on nature. He takes advantage of their characteristics to define the possibility of having a structure in which take place multiple processes in a non deterministic way. Each process “fights” for consuming the resources that are present inside that structure and may generate some new elements. These variations along the time may be considered as computation and the final result may be represented by the final stat of the structure.

Membrane computation does not intend the modeling of the cells natural behavior. This would be more appropriated for "bioinformatics" and should have as main goal, the understanding of such natural system for biological investigations. Such as we have just previously exposed, membrane computation does pretend to use the idea of computation by the means of "state changes" that a cell may have, allowing the final state to be considered as the "result".

To use these ideas, a hierarchy for membranes is defined. A membrane will be, not only the "separator" for a specific region, but also its content, that may be composed by:

- A set of different elements, which represent the different chemical materials that are located inside it and that may react among themselves to produce a change in their state.

- A set of rules that define how previously mentioned elements may evolve. These rules should be the equivalent to biochemical rules that govern reactions inside biological cells.
- One or more "inside" membranes that separate regions with the same structure than current.

There are some more cell characteristics but for this presentation we will just consider one more (that will not conform part of our final system). This characteristic is "selective permeability". It is well known that cell's membrane are not completely impermeable and that they may allow the way in, through themselves, for specific elements that pass from the exterior to inside and vice versa, either by active transportation (with energy consumption) or by passive one (for example, by osmosis).

This capacity to create "pores" is represented in the model by the indication of the target membrane for the result of a rule application (written in parenthesis).

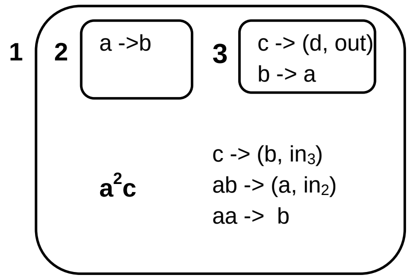


Fig. 1: Simple structure for a membrane

In this figure we represent the fact that a reaction among two elements "a" and "b" from membrane numbered as one would result in the production of an element for class "a" that would be sent to membrane "two", and the existence of an element "c" would result in the generation of an element "b" that would be passed to membrane numbered as three. Inside membrane three we may see that the reaction for an element "c" will result in the generation of an element "d" that should be passed to the membrane just superior in the hierarchy.

This representation is the one considered by so denominated "P Systems" that are, therefore, defined by a membrane structure that contains a multiset of objects, evolution rules that determine how the existing objects in the region change and are transformed, and a series of input/output indications.

Now, it is important to underline two main characteristics for these systems: rules application must be done in a non deterministic manner in the sense that reactions that take place among different elements may have the same priority and in coming scenarios could be executed in a different order. The second important characteristic is the fact that all the rules are applied in parallel, and so, combined with the non-determinism, may result in the execution of any of all the possible evolutions in that scenario. But not only all the rules are applied in parallel, but it is important to realize that all the membrane evolve together, in such a way that if we consider the passing of elements from one membrane to another, this could influence highly in the different states that a membrane could go through. This determines the existence of a "communication" step after the "evolution" step, in which membranes pass their "products" to the another ones indicated by their own rules.

Of course, computation comes from the different "configurations" that the system is going to take, understanding as such each of the resultant states from applying as many times as possible each of the evolution rules over the elements inside the membrane. Final result may be represented by, for example, remainder elements once that all the possible evolutions have been done, or by the elements that the superior membrane puts outside in the last evolution step.

In this work we have to notice that, without forgetting that a membrane may need multiple evolution steps and as many as communication steps with the other membranes, the study is focused on how rules application may be done in a specific evolution step and the final goal for this document is to obtain a design for a circuit that solves those rules application, without worrying about the communication phase among them.

Some background

Since Păun introduced P Systems, there has been done a lot of work in different approaches: the development of the model, the theoretical study of the different possible solutions or the direct application of this "know-how" through its implementation either using software or hardware. In all these fields, there are a lot of very important references that concern P Systems, as the one in which Păun defines P Systems [Păun, 1998] and, of course, others where the theoretical model is developed. There are, also, a lot of very good works from Fernández ([Fernandez et al., 2006]), Tejedor ([Tejedor et al, 2007]), or Gil ([Gil et al, 2008]) that cover the area which tries to find better algorithms and architectures to be implemented in P Systems. Concerning practical investigation, we may talk about "software simulations", finding very good works as the ones from Ciobanu [Ciobanu & Parashiv, 2001] or the ones from Nepomuceno [Nepomuceno, 2004].

As we may see, there are many more works that we could refer here, but, as far as this work is concerned, we need to focus on the hardware implementations, that would be our background. Solutions for these hardware implementations are approached from different points of view:

First we may consider the use of computer networks, with a good representation by [Syropoulos et al., 2003]. Second, we may see the developing of these systems using microcontrollers ([Gutiérrez et al., 2006]). The last working area about implementations is the one that focuses on the same goal than the ones already related but using FPGAs as processing devices instead of microcontrollers. A FPGA is a device which main characteristic is that it may be configured by the customer using a hardware description language (HDL). Its logic and components allows it to be used in complex circuits. Is in this specific field of Transition P System implementation where we are going to focus from now on.

The first outstanding work was done by Petreska and Teuscher in [Petreska & Teuscher, 2003], in which they designed what they called a “universal component", that allows membrane implementation in specialized hardware such as FPGAs. Martínez in [Martínez et al., 2006] presents the design for a new circuit that finds the set of active rules. Nguyen also presents in an excellent work ([Nguyen et al, 2007] and [Nguyen et al, 2008]) a design for a hardware system (Reconfig-P) implemented using FPGAs that develops all the evolution steps in membranes. Finally, in this specific field of designs using FPGAs, the authors of current work have exposed in [Alonso et al., 2008] a design that works out a circuit that solves rule application inside a membrane. This last work enhances the two already cited characteristics: indeterminism and the highest level of parallelism that may be brought. The result is a design that executes a very high number of operations over the active rules but also the amount of required resources is very high.

Goal

As we said before, the two main characteristics that we want in an implementation for Transition P Systems are:

- The indeterminism with which evolution rules are applied. We will understand as indeterminism the fact that different executions over the same system may not provide the same set of applied rules. If there are rules with the same priority to be applied, any of them will be chosen in a random manner.
- The high level of parallelism that these systems must have. This parallelism occurs at two different levels: the first refers to the fact that all the membranes evolve at the same time. This, which is obvious in "biological world”, is very important for P Systems power. The second level refers to the parallelism

that takes place inside each one of the membranes. In a P System membrane, all evolution rules intend to be applied in parallel and as many times as possible, consuming as many resources as they can.

Once we have emphasized this, we may establish the final goal for current work: to design a circuit using FPGAs that brings a solution for a Transition P System trying to accomplish these two characteristics. However, this circuit won't pretend to cover all the phases nor all the characteristics for these systems. We are going to focus on the evolution phase, that is that one in which the evolution rules are applied until available multiset is empty, without worrying about communication phase. So, we will design a circuit that implements an algorithm that does the active rules application over a given multiset, with a basic characteristic: it must have an universal nature and so, the circuit won't be dependent on initial conditions for the P System and will apply any set of active rules over any multiset (with the limitations imposed by hardware and necessary preloading). On the other hand, the design will simplify the model avoiding characteristics as membrane inhibition or new membrane creation. All the application rules present at a membrane will have the same execution priority.

The method

With the objective of keeping as high as possible the parallelism level during evolution rule application, power set of active rules is used. Power set of a specific set of elements is the one that contains all the possible sets formed by its elements. So, if R is the set of initial rules:

$R = \{R_1, R_2, \dots, R_n\}$

Its power set is:

$P(R) = \{\varnothing, R_1, R_2, \dots, R_n, R_1 R_2, \dots, R_1 R_n, \dots, R_{n-1} R_n, \dots, R_1 R_2 \dots R_{n-1} R_n\}$

As we may see, each of the elements from the power set is the result of one of the possible elements combinations from R. Knowing that $\{\varnothing\}$ is an element that presents no interest for our work, let's consider our power set as the one conformed by all the combinations except the empty one:

$P'(R) = \{ R_1, R_2, \dots, R_n, R_1 R_2, \dots, R_1 R_n, \dots, R_{n-1} R_n, \dots, R_1 R_2 \dots R_{n-1} R_n \} = P(R) - \{\varnothing\}$

Doing so, considering the power set as the initial set of active rules, what we have is the chance that, at the moment of choosing a rule that is formed by a combination of any others, its application will be the application of all the rules that compose it.

Once we have set which will be the active rules over which we are going to work, we have to say that one of the main problems in algorithms for active rules application is to establish the search space for a solution. There are algorithms that do the application for rules that do not represent a possible solution and so, they have to throw it away to get another one that could be valid. This could be the case for an algorithm that first randomly selects a rule among active ones and after that, checks if there are or not enough elements in the multiset to apply it. Solution for this problem goes through the search for rules only among the space where there are only valid.

One of the possible ways of implementing this vision is to turn around the traditional approach: instead of looking for an active rule and determine if it may be applicable or not, we will proceed to determine, for any possible multiset in our process, which are the rules that may be applicable. This is possible because each superscript for each element in the antecedent shows the requirements for that specific rule to be applied. If we establish intervals with these superscripts for all the elements that are needed in the antecedent and in those intervals we represent the set of rules that may be applied with those elements, what we have are intervals from which what we can get, at each moment, the set of active rules in each iteration. Now, only lasts that from this set of active rules, one of them should be randomly chosen and such rule should be applied a random number of times (between 1 and its maximum applicability factor).

As we are going to use it, we shall remember that we call *“maximum applicability”* for a rule, and we denoted it for “MAX”, the maximum number of times that a specific rule may be applied in such a way that it consumes all the resources in the multiset and taking into account that it could not be applied any more due to the fact that wouldn’t be enough elements in the multiset to do it.

Let’s see the whole proposed algorithm by an example in which we may see the interval creation and the initial preloading for the rules array: let’s assume that the initial system has just one membrane, being the multiset the following:

$W = a^3 b^4$

And let the active rules in the membrane be:

$R = \{a^3b \rightarrow c; a^2b^2 \rightarrow d\}$

As we already said, with the goal of getting the higher level of parallelism, the first thing that is done is to obtain the power set of R (with the exception of the empty set), that will be constructed by the same rules that are in R, plus the possible combinations among themselves (in our example we will have just one more rule):

$P'(R) = \{r_1, r_2, r_3\} = \{r_1 = a^3b \rightarrow c; r_2 = a^2b^2 \rightarrow d; r_3 = a^5b^3 \rightarrow cd\}$

Once that the initial set of active rules is established, for each element from all the antecedents in the rules, a set of values is formed and it will represent, in ascendant order, the different superscripts that appear for the element in each of the rules, that, after all, are the needed occurrences for each element for the application of each rule:

$M(a) = \{2, 3, 5\}$

$M(b) = \{1, 2, 3\}$

In our case, there are as many superscripts for “a” than for “b”, but it is possible not to be so, because they depend on the number of rules in which they appear and on the possibility for those superscript to be repeated in the rules.

Starting from zero, with each of the superscript we will set intervals that will be closed on their inferior limit and opened in their superior limit, in such a way that each of the superscripts will be exactly the limit. In our example we may write down four intervals for each element, having infinite as superior limit for the last of the intervals:

For element "a":

Superscript	0	2	3	5
Interval	[0,2)	[2,3)	[3,5)	[5, -)

For element "b":

Superscript	0	1	2	3
Interval	[0,1)	[1,2)	[2,3)	[3, -)

These intervals determine "regions" where we may locate the rules depending on how many units of a specific element they need to be applied. So, if a rule needs three "a" elements, it would be in the interval [3,5) for that element and if the same rule (or another) needs just one "b" element, then it would be in the interval [1,2) corresponding to the "b" element.

To complete properly the regions, taking into account that a rule needs the combination of different elements for its application, what we do is to build an array that in each dimension represents one of the existing elements with the intervals shown before, in such a way that any array cell is the region where we can find the rules that may be applied with the specific amount of those elements (indicated by the interval). This may be seen clearly by building the array for our example, where we can see that each element is a dimension (just two elements means two dimensions: rows and columns) with their intervals:

b\a	[0,2)	[2,3)	[3,5)	[5, -)
[0,1)	-	-	-	-
[1,2)	-	-	r_1	r_1
[2,3)	-	r_2	r_1 r_2	r_1 r_2
[3, -)	-	r_2	r_1 r_2	r_1 r_2 r_3

This array is filled up in its cells with the set of active rules that would be applicable if the multiset of elements has as many elements as the intervals for each element ("a" and "b") indicate. Of course, once a rule appears in a cell, it must appear in all the higher intervals in the corresponding row or column for the cell, because those intervals indicate that there are more elements than the needed ones. As well, the cell corresponding to the intersection of higher intervals for each element should contain all the existing rules because it indicates the existence of as many elements occurrences as needed to apply any of the rules.

The array is an "application map" that indicates us the set of rules that may be applied depending on the number of occurrences for each element that are in the multiset. So, if the multiset has only one occurrence of the "a" element, we can see that there would be no applicable rule because in the column under interval [0,2) there are no rules. This is easily verifiable because the three rules we have need three, two and five occurrences of "a" respectively.

If current multiset has, for example, four occurrences for "a" and two occurrences for "b", looking the cell corresponding to the column [3,5] for "a" and the row [2,3) for "b", we may check that applicable rules would be r_1 or r_2.

Knowing that the creation and initialization process for this array may be done before any evolution step that takes place inside the membrane, we may consider it as input data for the circuit we intend to design.

Once the prior phase has concluded, what we get is that, knowing a specific multiset, we have to find the cell that corresponds with the crossing for intervals indicated by the superscripts of each of the elements from that multiset. Once the cell is located, applicable rules are gotten from inside it. After this and with the goal of guarantying the indeterminism, one of them is randomly chosen and will be applicable a random number of times. Besides it, to avoid problems if this number is greater than the times that could be applicable with the elements in the multiset, this number will be limited to the maximum number of times that the rule may be applicable over the multiset. This is equivalent to calculating MAX (maximum applicability) and randomly generate the number inside the interval [1,MAX].

We could represent the algorithm as follows:

```
Phase 1
1. Power set P' creation for the set of rules R
2. Array loading with the applicable rules in each interval
3. Interval creation for each of the elements in the rules antecedents
4. Let W = {$w_1$, $w_2$,…$w_n$} be the initial multiset
Phase 2
5. REPEAT
6.      R ← $r_i$   $\forall r_i \in$ Array[$w_1$][$w_2$]…[$w_n$]
7.      r ← Aleatory(R)
8.      MAX ← Applicability(r)
9.      K ← Aleatory(1, MAX)
```

```
10.     W ← W – {K * input(r)}
11.     count (K, P'(R))
12. UNTIL | R | = 0
```

As we can see in phase two, its operating is similar to others already shown in previous works but with the advantage of knowing that the selected rule is always applicable (step 7). During step 8, its maximum applicability is computed to go on generating the number of times that really will be applied (step 9). The only thing left to do is the subtraction of the spent elements from the multiset (step 10) and register the rule for ulterior queries (we have to notice that we are working with the power set but what we want to know is the set of applied rules from the original set). This process will be iterative until the selected array cell will bring an empty set of rules.

The circuit

Data structure and its representation

Knowing that the goal for this work is a circuit design, we cannot forget that hardware imposes a series of limitations that in theory may not exist. So, for example, the number of elements in a multiset must be limited, and so occurs with the number of rules, etc. To be able to deal with these specifications, we impose the following restrictions and representations:

- Each element from the multiset and each of the elements from the evolution rules will be represented by a set of symbols corresponding to a finite alphabet. Cardinality for this set, just for this work, will be eight, in such way that we will be able to address them with three bits:

  $O = \{a, b, c, d, e, f, g, h\}$
- Multiset will be represented, then, by an array with eight elements
- Correspondence between elements and the alphabet symbols will take place through the position that occupy in the alphabet and in the array, being the same for both.

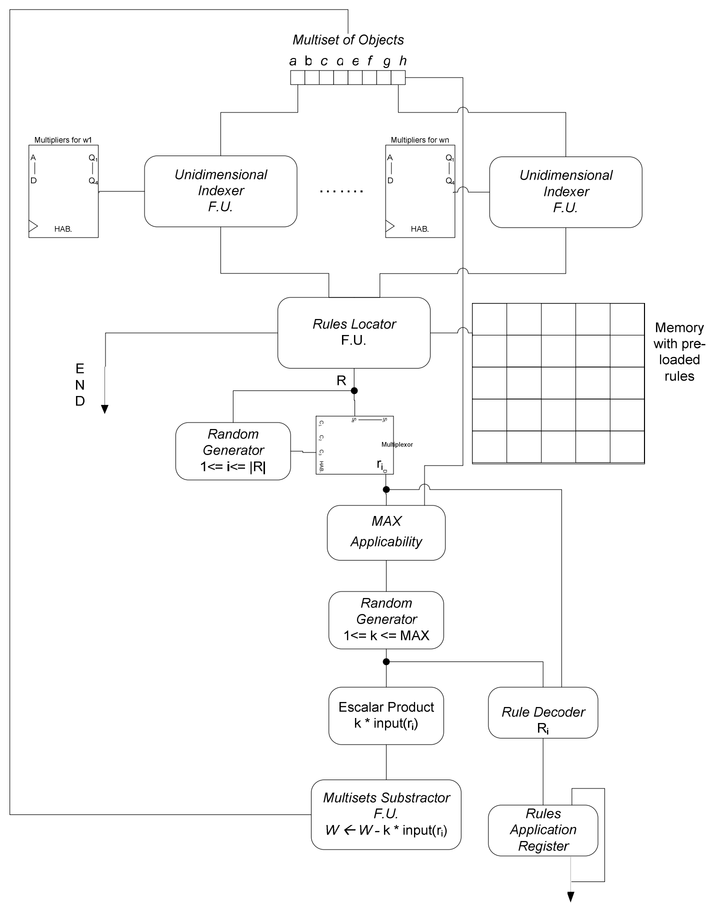
The array, in each position, will contain a number that refers to the amount of occurrences that exist in the multiset for the corresponding element.

The problem associated with the requirement of using a multidimensional array does not require more than a formula that translates any position in such array to another position in a one-dimensional array.

General design for the circuit

The general design of the circuit basically matches with the development, step by step, of the mentioned algorithm, with some adaptations due to the hardware limitations. Of course, all the first phase, including the array preloading must be done by software. As we may see, the available multiset is represented by a register with the number of occurrences for each element. From here, the different functional units that solve each of the steps in the algorithm are represented in the figure.

In a first stage, there are some functional units called "unidimensional indexer" that are in charge of locating, for each of the elements in the multiset, the interval where they belong. To do this, these units have a register with the superscripts already in ascendant order and that have been loaded at the same time that the rules vector. These units will get the indexes for each of the interval to which the superscripts belong.


Multiset of Objects
a b c d e f g h
Multipliers for w1
HAB.
Unidimensional Indexer F.U.
....
Multipliers for wn
HAB.
Unidimensional Indexer F.U.
Rules Locator F.U.
Memory with pre-loaded rules
E N D
R
Random Generator 1<= i<= |R|
Multiplexor
MAX Applicability
Random Generator 1<= k <= MAX
Escalar Product k * input(ri)
Rule Decoder Ri
Multisets Substractor F.U. W ← W - k * input(ri)
Rules Application Register

Once all the indexes are obtained, a second functional unit ("rules locator") is the one that locates, with those indexes, the set of rules that may be applicable inside the preloaded array and, in a next stage, by a random number generator (between 1 and the cardinality of the set of rules), one of them will be chosen to be applied.

Going on with the same algorithm, in the next stage, the maximum applicability for the selected rule is computed and a random number k is generated that will indicate the number of times that the rule will be actually applied. Of course there are just two more things to do: on one hand, to multiply k by the rule antecedents and subtract the result from the multiset, to get the new resultant multiset. We have to underline that data that is preloaded in the array does not change, and when the current multiset changes, the functional units will locate another intervals to which their superscripts belong.

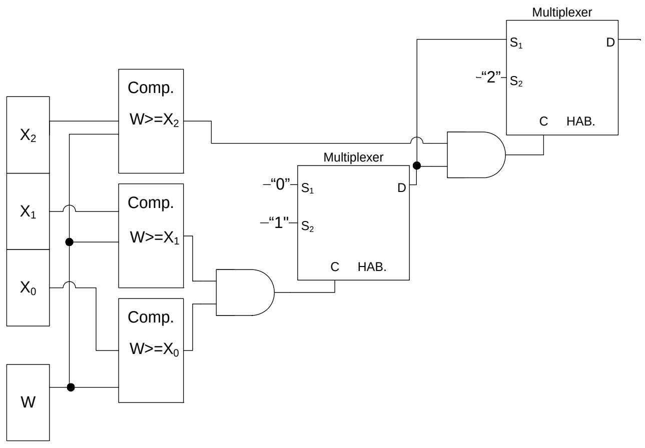
We can see that, along with the multiplication and subtraction of the consumed resources there is a functional unit called "rule decoder" that allows, starting from the applied rule (that belongs to the power set P') to know the rules from the starting set (a transformation from P' to R).

The circuit must keep going on until the functional unit that locates the applicable rules activates the "end signal", that will be when obtained set of rules from the array is empty.

One-dimensional Indexer functional unit

To be able to get the applicable rules set from the multidimensional array we need to know the value for the indexes that indicate the intervals to which the elements from the multiset belong. The goal for this functional unit is precisely to obtain the interval to which belongs the elements superscript. Knowing that one of these units will be in charge of one element, the circuit will have as many units as many elements may have the multiset (its maximum in this work will be eight, as long as we have limited the alphabet to eight symbols).

Input data for the unit are, from the register, the interval limits, while from the multiset, the number that indicates the number of occurrences for the specified element. During first iteration, solution for this module could come from a simple use of comparators, due to the fact that the index value has to be, necessarily, the limit for one of the intervals. Afterwards, during subsequent iterations, the value will be the result of the subtraction of the consumed elements, and this means that the indicated values could be inside the interval, not just in its limits. That's why, to spend a reasonable time, we use a "two by two" elements comparator, that will allow us to decide if the value in the multiset is greater or not than the interval limits.



Unidimensional Indexer for three intervals

In the figure we may see the design for one element from the multiset (W) and just three intervals $[X_0, X_1)$, $[X_1, X_2)$ and $[X_2, \infty)$. Using comparators and multiplexers, the obtained output is the number for the interval to which W belongs, taking into account that the first interval is numbered as zero.

Complexity referring to the number of necessary execution steps for this functional unit is order of the number of existing intervals, that, in the worst case, is a number order of the number of rules in P', that is multiple of $2^{|R|}$.

Naturally, if this functional unit brings the interval number for one element, we will need as many units as elements are, but with the advantage that the process will take place in parallel for all of them.

Memory and rules locator

The proposed design needs less hardware resource than others [Alonso et al., 2008] and, in return, uses a memory module where active rules are stored. This memory must be proportional, on one hand, to the number of elements in the alphabet or number of elements that a rule may contain, because each of them will be represented in one array dimension, that is |W|, and, on the other hand, to the number of intervals that each of those elements has, because in each array dimension we have to have as many columns as intervals has the specified element. Taking into account that the worst case is the one in which a specific element has a different number of occurrences for each of the rules, it seems clear that the maximum number of intervals, for that element, will be equal to the number of rules plus one. The amount of needed memory will be, thus, order of $|W|*|R'|$, always knowing that the number R' of rules is referred to the power set P' and so, is order of $2^{|R|}$, being |R| the number of rules in the original set. Referring to this original set, complexity will be, then, order of $|W|*2^{|R|}$.

To store the rules that have to appear inside each memory position, we may choose among a binary codification in just a word with the necessary length (one rule is represented in each position) and so, with eight bits we could represent eight rules and it is easy to increment the word's length if necessary. In this codification, a 1 means that the rule is an active one and a zero means "non applicable rule". This system would not increase by itself the amount of needed memory, but would do very much complex the generation of random numbers because they would have to be generated just for the positions that contain a 1 in the word.

Another way of doing this is the selected representation that causes the needed memory to be bigger, because for each of the cells in the array there will be as many words as the maximum number of existent rules and so, we will have all the number of the rules directly from it. So, for example, we will have eight words with three bits each, which will bring us directly the number of the applicable rule. Knowing that the empty element has been deleted from our power set, the zero will mean a non active rule. We have, finally, that the amount of memory is order of $|W|*2^{|R|}*2^{|R|}$, because this is the maximum amount of rules to be stored, or what is the same, order of $|W|*2^{2|R|}$.

The rules locator is the functional unit in charge of locating in the memory the active rules that may be applicable with the current multiset. As input data has the interval numbers to which the element's superscripts belong. With these numbers we may refer each of the active rules array's dimensions. To do so, we have into account that a multidimensional array is just a unidimensional one in which any position may be obtained by doing a simple transformation, multiplying the indexes by the number of elements in each dimension. So, if we are trying to locate a position (i, j, k) in an array with three dimensions which number of elements for each dimension are d_1, d_2, and d_3 respectively, we may do the transformation:

$$Pos(i, j, k) = k*d_1*d_2 + i * d_1 + j$$

What we get is a set of rules that is the input for a multiplexer which output will be selected by a random number generator between 1 and the number of rules obtained in the step just before. With this process the result is a rule that is always applicable.

We have to say that when the rules selector gets an empty set of rules activates the "end" signal and the process is finished.

Maximum applicability functional unit

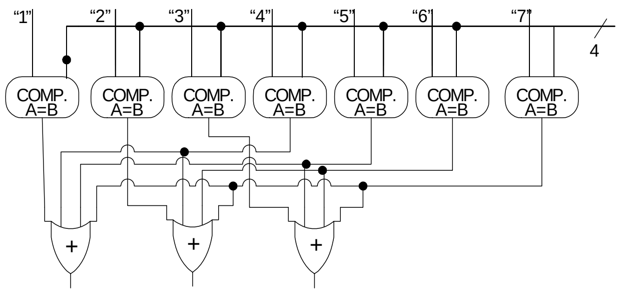
Next step, as we may check in the circuit diagram, is to get the maximum applicability value for the selected rule. To do this, we got the functional unit designed in [Martínez et al., 2006]. Its functioning is based upon the division of the superscript for each element in the rule antecedent by the value indicated in its position in the multiset. Once the results of the divisions are got, the smallest of them will be the one that indicates the number of times the rule may be applied.

Naturally, with the goal of keeping safe the non-determinism, after that, a random number is generated between 1 and that maximum applicability value, and it will indicate how many times the rule will be applied actually.

Other functional units

Once at this point, there is still some work to do and that is why there are some more functional units: the first one gets the result of multiplying the random number by the superscripts of the rule antecedent, and this is the number of consumed elements. Naturally, this number is passed to another functional unit to subtract them from the multiset to be ready for next iteration.

At the same time, the circuit must have a functional unit called "rule decoder" in charge of mapping the applied rule that belongs to the power set to the corresponding ones from the initial set of rules (R). The design for this unit is based upon the one from [Alonso et al., 2008], although this case is simpler because we have already the number of the rule. In the figure we may see the design for a power set (P') of seven rules.



The signals that are activated in this unit are brought to the following unit with the intention of being accumulatively registered and so, to have at the end of all the iterations, the number of times that each rule has been applied.

Conclusion

Until now, the circuit is still in its design phase and although some of the functional units have been tested with simulators, we do not have still real measurements over the circuit and so, we cannot get conclusions upon its real implementation. Besides this, we may establish some guidelines about necessary resources and its behavior.

The design for this circuit pretends to be a balance among the very high cost for hardware resources that the preceding circuit had and the goal for a very high parallelism level during rules application. On the other hand, we may not forget that the circuit represents a universal membrane, in such a way that, after a preloading stage, its use to implement any Transition P System may be possible (naturally, with the limitations imposed for this prototype).

Through the reading of this work, we may easily see that we have achieved a lowering for the number of hardware resources that we needed before and that was exponentially proportional to the number of rules. This enhancement is done, although, with the use of a higher amount of memory associated to the circuit (in which active rules for each interval are stored).

On the other hand, in this circuit a preloading is included and it needs to be executed before the first iteration and this brings some more complexity in terms of processing time. Complexity for this software is order of O (|R|*|W|) because it locates, for each rule, the intervals that has each multiset element.

It also results remarkable the fact that complexity orders in this design goes exponentially high because of the goal to increase parallelism level to the maximum, and so, consider the power set of the rules as the initial set of active rules. In case we want to give up this parallelism increment, we may apply the same circuit to the initial set R and, of course, complexity would be much smaller. In this case, functional unit "rule decoder" would be of no sense because the rule number could be stored directly through a decoder, in the register.

In the following table we do a small theoretical comparison among the circuit in [Alonso et al., 2008], the one designed in this work and the same but without considering the power set:

	Circ. in [Alonso et al., 2008]	|P'| rules	|R| rules
Modules/gates	$O(2^{\|R\|})$	$O(\|W\|)$	$O(\|W\|)$
Memory	$O(\|R\|)$	$O(\|W\|2^{2\|R\|})$	$O(\|W\|2^{\|R\|})$
Software	-	$O(\|W\|2^{\|R\|})$	$O(\|W\|2^{\|R\|})$

As a final conclusion for this work, we have to underline that the design, in the absence of real tests, seems right towards its implementation as a universal membrane with the exposed characteristics in a FPGA. As we have mentioned, limitations would come from hardware.

Bibliography

[Alonso et al., 2008] S. Alonso, L. Fernández, F. Arroyo, J. Gil. *Main Modules design for a HW Implementation of Massive Parallelism in Transition P-Systems*. Thirteenth Internacional Sysmposium on Artificial Life and Robotics 2008 (AROB 13th). Beppu, Oita, Japan, 2008. Article in "Artifical Life and Robotics" (Springer), Volume 13, pp 107-111, 2008.

[Ciobanu & Parashiv, 2001] G.Ciobanu, D.Paraschiv. *Membrane Software. A P System Simulator*. Workshop on Membrane Computing, Curtea de Arges, Romania, August 2001, Technical Report 17/01 of Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona, Spain, 2001, 45-50 and Fundamenta Informaticae 49, 1-3 (2002), 61-66.

[Fernandez et al., 2006] Fernández L., Arroyo F., Tejedor J.A., Castellanos J., *Massively Parallel Algorithm for Evolution Rules Application in Transition P System*. International Workshop (WMC7), Pp: 337-343, Leiden (Netherlands) 2006.

[Gil et al, 2008] Gil F.J., L. Fernández, F. Arroyo, A. Frutos. *Parallel Algorithm for P Systems implementation in Multiprocessors*. International Sysmposium on Artificial Life and Robotics 2008 (AROB 13th). Beppu, Oita, Japan. 2008

[Gutiérrez et al.,2006] A. Gutiérrez, L. Fernández, F. Arroyo, V. Martínez. *Design of a hardware architecture based on microcontrollers for the implementation of membrane systems*, SYNASC 2006, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing Timisoara, Romania. September 2006.

[Martínez et al., 2006] V. Martinez, L. Fernández, F. Arroyo, I. García, A. Gutierrez. *A Hardware Circuit for the application of active rules in a Transition P System region*. Fourth International Conference Information Research and Applications (i.TECH-2006). ISBN-10: 954-16-0036-0. Varna (Bulgary) Junio, 2006. Pp. 147-154.

[Nepomuceno, 2004] Nepomuceno I., *A Java Simulator for Membrane Computing*. Journal of Universal Computer Science 10, pp. 620-629. 2004

[Nguyen et al, 2007] V. Nguyen, D. Kearney, G. Gioiosa. *Balancing Performance, Flexibility and Scalability in a Parallel Computing Platform for Membrane Computing Applications*. Workshop on Membrane Computing 2007. Thessaloniki, Greece, June 25-28, 2007

[Nguyen et al., 2008] V. Nguyen, D. Kearney, G. Gioiosa. *An algorithm for non deterministic object distribution in P Systems and its implementation in hardware*. Membrane Computing: 9th Workshop, WMC 2008, Edinburgh, UK, July 2008

[Păun, 1998] Gh. Păun, *Computing with Membranes*, Journal of Computer and System Sciences, 61(2000) and Turku Center of Computer Science-TUCS Report nº 208, 1998.

[Petreska & Teuscher, 2003] B.Petreska, C.Teuscher, *A reconfigurable hardware membrane system*. Workshop on Membrane Computing (A.Alhazov, C.Martín-Vide and Gh.Paun, eds) Tarragona, July 17-22 2003, 343-355.

[Syropoulos et al., 2003] A.Syropoulos, E.G.Mamatas, P.C.Allilomes, K.T.Sotiriades. *A distributed simulation of P systems*. Workshop on Membrane Computing (A.Alhazov, C.Martin-Vide and Gh.Paun, eds); Tarragona 2003, 455-460.

[Tejedor et al, 2007] J.A. Tejedor, L.Fernández, Arroyo, G. Bravo. *An Architecture for Attacking the Communication Bottleneck in P Systems*. International Sysmposium on Artificial Life and Robotics 2007 (AROB 12th). Beppu, Oita, Japan. 2007

Authors' Information

Santiago Alonso *– Natural Computing Group of Universidad Politécnica de Madrid. - Dpto. Organización y Estructura de la Información de la Escuela Universitaria de Informática, Ctra. de Valencia, km. 7, 28031 Madrid (Spain); e-mail: salonso@eui.upm.es*

Luis Fernández *– Natural Computing Group of Universidad Politécnica de Madrid. - Dpto. Lenguajes, Proyectos y Sistemas Informáticos de la Escuela Universitaria de Informática, Ctra. de Valencia, km. 7, 28031 Madrid (Spain); e-mail: setillo@eui.upm.es*

Víctor Martínez *– Natural Computing Group of Universidad Politécnica de Madrid. - Dpto. Arquitectura y Tecnología de Computadores de la Escuela Universitaria de Informática, Ctra. de Valencia, km. 7, 28031 Madrid (Spain); e-mail: victormh@eui.upm.es*

MEMBRANE COMPUTING: NON DETERMINISTIC TECHNIQUE TO CALCULATE EXTINGUISHED MULTISETS OF OBJECTS.

Alberto Arteta, Angel Castellanos, Ana Martinez

Abstract: *Within the membrane computing research field, there are many papers about software simulations and a few about hardware implementations. In both cases, algorithms are implemented. These algorithms implement membrane systems in software and hardware that try to take advantages of massive parallelism. P-systems are parallel and non deterministic systems which simulate membranes behavior when processing information.*

This papers describes the evolution rules application process and it presents software techniques for calculating maximal multisets on every evolutionary step.

These techniques improve the best performance achieved by the p-systems when applying evolution rules.

Algorithms could stop being useful when the number of objects “n” in which they depends on, increases. By using this technique, that specific problem can be overcome. The output can be given under a constant complexity order. The complexity order might be constant under certain conditions, regardless the value “n”. In order to do this, the proper use of memory is essential. This work will provide the details for building a structure. This structure will allow us to improve performance in terms of time. Moreover this structure can be allocated in the random access memory and/or the virtual memory

Keywords: *P-systems, Parallel systems, Natural Computing, evolution rules application, set of patterns, structure.*

ACM Classification Keywords: *D.1.m Miscellaneous – Natural Computing*

Introduction

Membrane computing is a parallel and distributed computational model based on the membrane structure of living cells [Păun 2000]. This model has become, during these last years, a powerful framework for developing new ideas in theoretical computation. The main idea was settled in the base of connecting the Biology with Computer Science in order to develop new computational paradigms. P-systems are the structures that reproduce the information processing occurring in living cells. Nowadays, it can be found that several P Systems simulators are much elaborated” [Păun 2005] .At this point, there is a problem with the parallelism synthesized in [Păun 2005] by: “parallelism a dream of computer science, a common sense in biology“. This is the reason why, Păun avoids “to plainly saying that we have ‘implementations of P systems’, because of the inherent non-determinism and the massive parallelism of the basic model, features which cannot be implemented, at least in principle, on the usual electronic computer. […]

There are several uses of the p-systems. P-systems can be used to increase performance when dealing with known problems; for example, the knapsack problem [Pan, Martin 2005].

Also, Membrane computing is currently used to model other parallel and distributed systems as Grid [Qi, Li, Fu, Shi, You 2006].

An overview of membrane computing software can be found in [Ciobanu, Pérez-Jiménez, Ciobanu, Păun 2006], or tentative for hardware implementations [Fernández, Martínez, Arroyo, Mingo 2005] is enough to understand how difficult it is to implement membrane systems on digital devices. This does not mean that simulations of P systems on common computers are not useful; actually, such programs were used in biological applications, and can also have important didactic and research applications". Every algorithm that implements the evolution rules application, for making evolving the systems is sequential. This is due mainly to the fact that technologies are inherently sequential or, when concurrent technology is used, evolution rules are applied to the multiset under mutual exclusion. In this paper, by using a linear structure and allocating it in the physical and/or virtual memory of a given computer, response timing might be reduced considerably. In scenarios where traditional algorithms do not obtain good performance, the technique explained in this paper can guarantee an excellent performance as long as a few requirements are fulfilled. Respecting the inherent non-determinism within the p-systems, the technique respects it as it is a nondeterministic technique itself.

This paper is structured as follows:

- Introduction to membrane computing
    - Definition of concepts as: *P-systems*, *Extinguished multisets*, *multisets of objects, multiplicity of an object and evolution rules.*
    - We focus on the evolution rule application phase. A brief description of this phase is provided
- Creation of a *n-dimensional structure*. That *structure* is created throughout an application that establishes a link between the *initial multisets*, and the number of times that each *evolution rule* should be applied in order to obtain an *extinguished multiset.* The dimension of the structure will be equal to the number of objects we are working with. The number of entries this structure has will be determined by a prefixed value called *"benchmark".*
- Analysis of the *technique* described in this work covering:
- Finally, some conclusions will be established. These conclusions will be the result of doing a study in detail throughout the entire work presented here.

Membrane computing

Transition P Systems

A Transition P System of degree n $n > 1$ is a construct $\Pi = (V, \mu, \omega_1, .., \omega_n, (R_{1,}\rho_1), ..(R_{n,}\rho_n), i_0)$

Where:

V is an alphabet; its elements are called objects;

μ is a membrane structure of degree n, with the membranes and the regions labeled in a one-to-one manner with elements in a given set ; in this section we always use the labels 1,2,..,*n*;

ω_i $1 \le i \le n$, are strings from V^* representing multisets over *V* associated with the regions 1,2,..,*n* of μ

R_i $1 \le i \le n$, are finite set of evolution rules over V associated with the regions 1,2,..,*n* of μ; ρ_i is a partial order over R_i $1 \le i \le n$, specifying a priority relation among rules of R_i . An evolution rule is a pair (*u*,*v*) which we will usually write in the form $u \to v$ where *u* is a string over *V* and *v*=*v'* or *v*=*v'*δ where *v*' is a string over $(V \times \{here, out\}) \cup (V \times \{in_j \; 1 \le j \le n\})$, and δ is a special symbol not in. The length of u is called the radius of the rule $u \to v$

i_o is a number between 1 and *n* which specifies the output membrane of Π

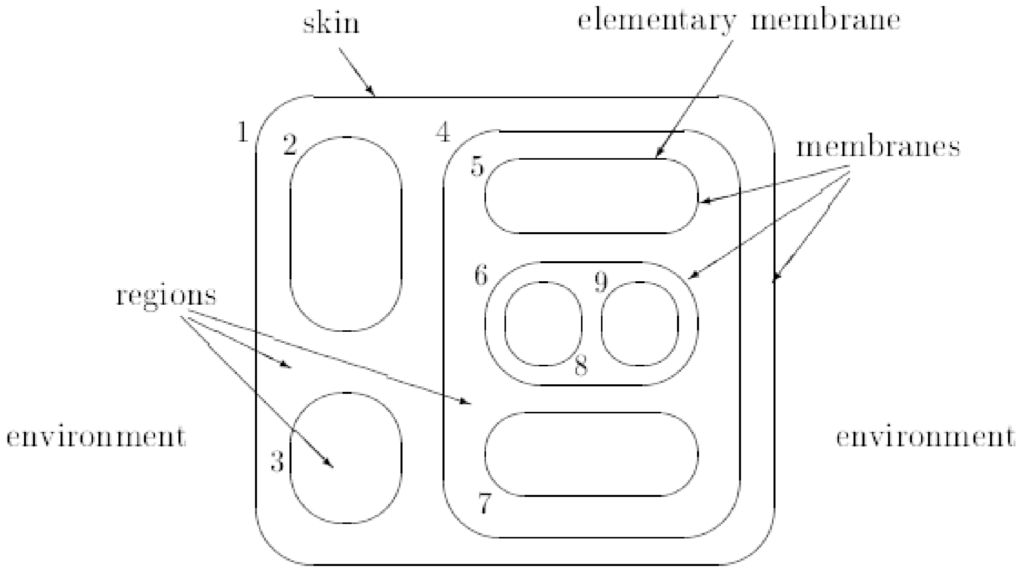


Fig 1 P-system structure

Definition Multiset of objects

Let *U* be a finite and not empty set of objects and *N* the set of natural numbers. A *multiset of objects* is defined as a mapping:

$M : U \rightarrow N$

$a_i \rightarrow u_1$

Where a_i is an object and u_i its multiplicity

As it is well known, there are several representations for multisets of objects.

$M = \{(a_1,u_1),(a_2,u_2),(a_3,u_3)...\} = a_1^{u_1} \cdot a_2^{u_2} \cdot a_n^{u_n}$ ......

Note: *Initial Multiset* is the multiset existing within a given region in where no application of evolution rules has occurred yet.

Definition *Evolution rule* with objects in *U* and targets in *T*

Evolution rule with objects in *U* and targets in *T* is defined by $r = (m,c,\delta)$ where $m \in M(U), c \in M(UxT)$ *and* $\delta \in \{$*to dissolve, not to dissolve*$\}$

From now on *'c'* will be referred a s the consequent of the evolution rule *'r'.*

Note The *set of evolution rules* with *objects* in *U* and targets in *T* is represented by *R (U, T).*

Definition Multiplicity of an object in a multiset of objects *M(U)*

Let $a_i \in U$ be an object and let $m \in M(U)$ be a multiset of objects. The multiplicity of an object is defined over a multiset of objects such as:

$$| \; |_{a_i} : U \times M(U) \rightarrow N$$

$$(a_i, m) \rightarrow |m|_{a_i} = n \mid (a_i, n) \in m$$

Definition Multiplicity of an object in an evolution rule *r*

Let $a_i \in U$ be an object and let $R(U,T)$ be a multiset of evolution rules. Let $r = (m,c,\delta) \in R(U,T)$ where $m \in M(U), c \in M(UxT)$ *and* $\delta \in \{\textit{to dissolve, not to dissolve}\}$

The multiplicity of an object is defined over an evolution rules such as:

$$| \ |_{a_i} : U \times R(U,T) \to N$$

$$(a_i, r) \to |m|_{a_i} = n \mid (a_i, n) \in m$$

P-system evolution

Let C_i be the *consequent* of *the evolution rule* r_i. Thus,

the representation of the evolution rules are:

$r_1 : a_1^{u11} a_2^{u12} .. a_n^{u1n} \to C_1$

$r_2 : a_1^{u21} a_2^{u22} .. a_n^{u2n} \to C_2$

.... $\to$ ....

$r_m : a_1^{um1} a_2^{um2} .. a_n^{umn} \to C_m$

P-systems evolve, which makes it change upon time; therefore it is a dynamic system. Every time that there is a change on the p-system we will say that the p-system is in a new transition. The step from one transition to another one will be referred to as an evolutionary step, and the set of all evolutionary steps will be named computation. Processes within the p-system will be acting in a massively parallel and non-deterministic manner. (Similar to the way the living cells process and combine information).

We will say that the computation has been successful if:

- The halt status is reached.
- No more evolution rules can be applied.
- Skin membrane still exists after the computation finishes.

Evolution rule application phase

This phase is the one that has been implemented following different techniques.

In every region within a p-system, the evolution rules application phase is described as follows:

Rules application to a multiset of object in a region is a transforming process of information which has input, output and conditions for making the transformation.

Given a region within a p-system, let $U = \{a_i \mid 1 \leq i \leq n\}$ be the alphabet of objects, *m* a multiset of objects over *U* and *R(U,T)* a multiset of evolution rules with antecedents in *U* and targets in *T*.

The input in the region is the initial multiset *m.*

The output is a maximal multiset *m*'.

The transformations have been made based on the application of the evolution rules over m until m' is obtained.

Application of evolution rules in each region of P systems involves subtracting objects from the initial multiset by using rules antecedents. Rules used are chosen in a non-deterministic manner. This phase ends when no rule is applicable anymore.

The transformation only needs rules antecedents as the consequents are part of the communication phase.

Observation

Let $k_i \in N$ be the number of times that the rule r_i is applied. Therefore, the number of symbols a_j which have been consumed after applying the evolution rules a specific number of times will be:

$$\sum_{i=1}^{m} k_i \cdot u_{ij}$$

Definition Maximal Multisets

Given a region *R*, let *U* be an alphabet of objects $U = \{a_i \mid 0 < i \leq n\}$.

Let $M(U) = \{(a_i, u_i) \mid a_i \in U\ u_i \in \mathrm{N}\ 0 < i \leq n\ \}$ the set of all the multisets over *U*. Let $x \in M(U)$ be a multiset of objects over *U* within *R*. Let $R(U,T) = \{r_i \mid \exists m \in \mathrm{N}\ i \leq m,\ i \in \mathrm{N}\}$ be a set of evolution rules. $r_j = (x_j, c_j, \delta) \in R(U,T)$ and $x_j = \{(a_{ji}, u_{ji}) \mid \exists n,m \in \mathrm{N}\ i \leq n,\ j \leq m\ \ i,j \in \mathrm{N}\}$

Let $k_j \in \mathrm{N}$ the number of times that the rule $r_j \in R(U,T)$ is applied over x. we say x is an extinguished multiset if and only if:

$$\bigcap_{j=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m}(k_j \cdot u_{ji}) \leq u_{ji}\right)\right] \text{ [1]}$$

Observation

In other words, *m* is a maximal or extinguished multiset if and only if it is not possible to apply any more evolution rules over it.

Definition

Given a region *R* and alphabet of objects *U*, and *R (U, T)* set of evolution rules over *U* and targets in *T*.

$$r_1 : a_1^{u_{11}} a_2^{u_{12}} . a_n^{u_{1n}} \rightarrow C_1$$
$$r_2 : a_1^{u_{21}} a_2^{u_{22}} . a_n^{u_{2n}} \rightarrow C_2$$
$$\ldots \quad \rightarrow \ldots$$
$$r_m : a_1^{u_{m1}} a_2^{u_{m2}} . a_n^{u_{mn}} \rightarrow C_m$$

Maximal multiset is that one that complies with:

$$\bigcap_{l=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m}(k_j \cdot u_{ji}) \leq u_{li}\right)\right] \quad (1)$$

Within a given membrane, Let $U = \{a_i \mid i = 1,..n\}$ be a set of objects. Let T be a set of targets. Let $\omega = a_1 a_{2..} a_n$ be a multiset of objects and let u_i be the multiplicity of a_i. Let *R(U,T)* be a multiset of evolution rules with objects in *U* and targets in *T*. Let *m* be the number of evolution rules within *R(U,T)*. Let k_i be the number of times that the rule $r_i \in R(U,T)$ is applied. We define:

$$\varphi_1 : N^m \quad \rightarrow \quad N^n$$
$$(k_1, k_2, .., k_m) \quad \rightarrow \quad (u_1, u_2, .., u_n)$$

“*m*” is the number of the evolution rules within *R(U,T)*, “*n*” is the number of symbols within the given membrane.

$$\varphi_1(k_1,k_2,..,k_m,) \equiv \begin{pmatrix} u_{11} & u_{21} & ...u_{m1} \\ u_{12} & u_{22} & ...u_{m2} \\ ... & ... & ... \\ u_{1n} & u_{2n} & ...u_{mn} \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ ... \\ k_m \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ ... \\ u_n \end{pmatrix}$$

Based on definition we are interested on $\{k=(k_1,k_2,..,k_m) \in N^m / \varphi_1(k)=u=(u_1,u_2,..,u_n)\}$ and $\bigcap_{l=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m}(k_j \cdot u_{ji}) \le u_{li}\right)\right]$

$$\begin{aligned} \Phi_1 : \mathrm{N}^n \quad & \to \quad \mathrm{P}(\mathrm{N}^m) \\ u=(u_1,u_2,..,u_n) & \to k=(k_1,k_2,..,k_m) \end{aligned}$$

This function returns a random value $k=(k_1,k_2,..,k_m) \in N^m / \varphi_1(k)=u$. We know that there might be more than one "k". We are just interested on one value 'k' that complies with $\varphi_1(k)=x$. This is possible as the set $K=\{(k_1,k_2,..,k_m) \in N^m / \varphi_1(k_1,k_2,..,k_m)=x\}$ is always included in a feasible region [1]. This feasible region is an area which contains the set K. There are several methods to select one element of K within a feasible region. These methods are studied on [Arteta, Fernandez, Gil 2008].

Because any of these methods always returns a value, we can state that Φ_1 is a computable function. For example, we can compute this function by implementing the algorithm for Application of Evolution Rules based on linear Diophantine equations. [1]

Patterns function Φ_2

Φ_2 is called pattern function because has as a set of set of patterns as input . This function transforms this set of set of patterns into a set of numbers we call $A=\{(u_1,u_2,..,u_n) \in \mathrm{N}^n\}$. The elements of this resulting set are all the combinations of all the possible values which are included in the Set of set of patterns SP

$$\Phi_2 : \quad \mathrm{P}(\mathrm{N}^m) \quad \to \quad \mathrm{N}^n$$

$$\begin{pmatrix} [[0,u_{11}],...[0,u_{m1}]],...,[[0,u_{11}],..[0,u_{mn}]], \\ ... \\ [[0,u_{12}],..[0,u_{m1}]],...,[[0,u_{12}],..[0,u_{mn}]], \\ \end{pmatrix} \to \begin{bmatrix} u_1,u_2,..,u_n \\ u_1',u_2',..,u_n' \\ ... \\ u_1'^p,u_2'^p,..,u_n'^p \end{bmatrix}$$

Bulding the Structure

Creation of the structure L_Φ

Within a region, let $U=\{a_i \mid i=1,..n\}$ be a multiset of objects and let $R\ (U,\ T)$ be a multiset of evolution rules. Given the set $\{k_i \in N\}$ the number of times that the evolution rule r_i is applied over the initial multiset}, we build the evolution rules function. Φ

Once the function has been constructed, the new structure must be built this way:

- The output of Φ_2 , which is a set of numbers we call $A=\{(u_1,u_2,..,u_n)\in \mathrm{N}^n\}$.
- The output of Φ is a unique random value $k=(k_1,k_2,..,k_m)\in N^m$
- For any element, $u=(u_1,u_2,..,u_n)\in A$ we store the value $k=(k_1,k_2,..,k_m)\in N^m \;/\; \varphi_1(k)=u$ .Also we store this value in all the positions that are included in [$(u_1,u_2,..,u_n)$] of the structure. Thus, $L_\Phi\ [(u_1,u_2,..,u_n)] = k_1k_2...k_m$
- If the position $(u_1,u_2,..,u_n)$ of the structure already has information, then we do not store anything.

The expression to build the structure is as follows:

$$L_\Phi\left[\Phi_2\left(SP_{R(U,T)}\right)\right]=\Phi\left(SP_{R(U,T)}\right)$$

Note: $SP_{R(U,T)}$ is the set of set of patterns associated to *our R(U,T)*

Each entry of the structure contains the values: $k_1,...,k_m$. These values will indicate the number of times that an evolution rule should be applied to an initial multiset in order to obtain an extinguished multiset. The number of objects determines the coordinates of a given position in the structure; thus, as the number of objects increases so it does the dimension of the structure; i.e. working with an alphabet *V* of 2 symbols $V=(a_1,a_2)$ means that our structure is two-dimensional. The numbers of entries existing in the structure depends on the dimension, (number of symbols of our alphabet) and the benchmark of every object. This benchmark is a value associated to every object of our alphabet.

The benchmark fixes a limit for the multiplicity of the objects existing in our alphabet. In order to make the structure useful, we need to fix reasonably high benchmarks. Our structure does not store information about the values that are higher that the prefixed benchmarks.

Non determinism

Let us *B* be the set $\{k_1,..,k_m \mid$ after applying the evolution rule r_i a number of k_i times to a given multiset $M=\{(a_1,u_1),(a_2,u_2),..,(a_n,u_n)\}$, $\forall\ i\in N\ \ i\le m$, we obtain an extinguished multiset}

In order to preserve the non-determinism, we must assure that the value returned by our technique is $k\in B$.*k* is chosen randomly between all the elements within *B.*

So far the structure we are building complies with that.

As per [1] we know that there might be more than one element in *B.*

Thus, we must build a dynamic structure which updates itself after the structure returns a solution $(k_1,..,k_m)\in N^m$.

The way to do this is:

For any entry of $L_\Phi\ [u_1,u_2,..,u_n]$:

Let us *B* be the set $\{k_1,..,k_m \mid$ after applying the evolution rule r_i a number of k_i times to a given multiset $M=\{(a_1,u_1),(a_2,u_2),..,(a_n,u_n)\}$, $\forall\ i\in N\ \ i\le m$, we obtain an extinguished multiset}.

We store in that entry $L_\Phi\ [u_1,u_2,..,u_n]$ an element K= $(k_1,..,k_m)\in B$

- When an initial multiset $M = \{(a_1,u_1),(a_2,u_2),..,(a_n,u_n)\}$ is provided as input, we return K.
- We update the entry $L_\Phi\ [u_1,u_2,..,u_n]$ by allocating a value $k' = (k_1' ,..,k_m') \in B$

$(k_1,..,k_m)$ is returning in constant time. The value within the structure is updated once $(k_1,..,k_m)$ is returned. Following this procedure, our technique is ready to return a different output every time we have the same input. The output is chosen in a non deterministic manner.

Technique's Analysis

Building the structures guarantees, that any input value $(u_1,..,u_n) \in N^n$ has a direct relation with $(k_1,..,k_m) \in N^m$. This implies that the structure can be built from a given set of evolution rules. In this way, any algorithm that intends to calculate an extinguished multiset from an initial multiset of objects can generate the structure L_Φ during the compilation time.

During the compiling stage of an algorithm, a new structure is created. When the multiplicities of the initial multisets are provided as input values $u_1,..,u_n$ of a given algorithm, this algorithm will search the entry $[u_1,..,u_n]$ of the structure. Thus, the algorithm returns a combination of values $(k_1,..,k_m) \in N^m$ which are stored in that entry. Any entry or cell stores a set of combinations $(k_1,..,k_m) \in N^m$. Any of these combinations complies with:

$\varphi_1(k_1,k_2,..,k_m) = \{u_1,u_2,..,u_n\}$

One of the combinations stored in a cell is selected randomly. The dynamic way that the structure is created, assures that for each input values, different outputs might be generated. Thus, we are in conditions to state that the use of the structure preserves the non-determinism, which is a characteristic of the Paun's biological model.

In both structures, the number of cells that L_Φ needs, are: $\prod_{i=1}^{n} benchmark\ (a_i)$.

a_i is an object of the alphabet of our p-system and n is the number of objects of our alphabet. The combination of the number of objects and the selected benchmarks of each object will determine the number of cells that the structure must have. Thus, the combination of these two factors will have a major influence in the amount of memory used by L_Φ .

Within the last years, the algorithms that calculate extinguished multisets have had a complexity Order $\Theta(m)$ where m is the number of evolution rules involved. When the number of evolution rules is high, the execution time can take too long. On the contrary, the best scenario for this technique is when the combination between number of symbols and their benchmarks is low, regardless the number of evolution rules and the maximum applicability benchmark [Fernández, Castellanos, Arroyo, tejedor, Garcia 2006] of any evolution rule.

For instance, we consider a hypothetical scenario where the alphabet of objects $S = \{a_1,a_2\}$, and the number of evolution rules is considerably high. Let us take a number extremely high as 10^{12} .Let us also consider that the maximum value that an evolution rule can be applied is 10^{24} . Under these conditions, an algorithm based on the maximal applicability benchmark [Fernández, Castellanos, Arroyo, tejedor, Garcia 2006] would be inefficient as its complexity is *O(m)*, where *m* is the number of evolution rules. On the contrary, an algorithm that uses this technique can find extinguished multisets with a constant complexity order.

The amount of memory needed to create the structure is explained as follows:

Using this technique we will have to reserve 124 bits for each structure's cell as per: $10^{12} \cdot 10^{24} = 10^{36} \leq 2^{4^{36}} = 2^{124}$. This means 15 bytes per cell .Let us set a benchmark of 30000 for each symbol a_i of the multiset. Considering that there are two objects within our current Multiset: $S = \{a_1, a_2\}$, then we will have $30000 \cdot 30000 = 9 \cdot 10^8$ number of cells of L_Φ . Thus, the amount of memory needed to create L_Φ is $9 \cdot 10^8 \; cells \cdot 15 (bytes/cell) = 135 \cdot 10^8 \; bytes = 13.5 Gb$

Thus, we can assure that when having 13.5 Gb any algorithm using this technique can find extinguished multisets with complexity order *O(1)* as long as Input values for $a_1 \; and \; a_2 \quad multiplici \; ties$ are <30000. This will occur regardless the number of evolution rules.

Once this is done, the structure can be built. This structure relates the times that each evolution rule should be applied to the multiplicity input values of the initial multiset. This occurs as long as these values are always either less than or equal to their benchmark.

An algorithm, as the one based on the maximum applicability benchmark, could create the structures during compilation stage. If the number of evolution rules is high, it would be worth considering creating the structure. This will find the corresponding values to the number of times that each rule should be applied in order to obtain an extinguished multiset. This will happen as long as the input values of the object's multiplicities are lower than the previously fixed benchmarks.

The algorithm

The algorithm we present here is very simple. Once the structure has been built based on the indications given in this paper, it searches on the structure.

The multiplicities of a given initial multiset are provided as input for the algorithm.

The algorithm then searches in the corresponding entry. The only thing it does it is asking for the input numbers which correspond to the initial multiplicity of a given multiset. After that, it just look it up in the entry that has the coordinates equal to the input numbers. It just returns the stored values.

```
(1)   $u_1, u_2, .., u_n \leftarrow Multiplici\,ty(m)$
(2)   BEGIN
(3)   $output(L_\Phi(u_1, u_2, .., u_n))$
(4)   END
(5)   $REPLACE(L_\Phi(u_1, u_2, .., u_n))$
```

$(u_1, u_{2,} ...u_n)$ is the input value corresponding to the multiplicities of the initial multiset m. The algorithm searches in the structure L_Φ the position $(u_1, u_{2,} ...u_n)$. After the output is given, another value replaces the entry $(u_1, u_{2,} ...u_n)$.

A traditional algorithm can take advantage of the technique as follows:

```
(1)  $u_1, u_2, .., u_n \leftarrow Multiplici\ ty(m)$
(2)  $IF\ (u_1, u_2, .., u_n) < BENCHMARK\ (u_1, u_2, .., u_n)\ THEN$
        (3)  BEGIN
        (4)  $output\left(L_\Phi(u_1, u_2, .., u_n)\right)$
        (5)  END
        (6)  $REPLACE\left(L_\Phi(u_1, u_2, .., u_n)\right)$

(7)  ELSE
        (8)  {TRADITIONA L ALGORITHM }
```

If the multiplicities of the initial multiset are lower than the benchmark then we use the structure; If not, then we use the traditional algorithm.

Conclusions

This work contributes with a new technique in finding extinguished multisets from initial multisets. The technique offers a complement to be used by the traditional algorithms which calculate extinguished multisets. These algorithms find the extinguished multisets after applying a certain number of evolution rules a certain number of times. The technique presents the idea of going "backwards" from the initial multisets to the number of times that each rule should be applied. This process automatically obtains the times that each rule should be applied and no extra calculation is necessary. From there, calculating the extinguished multisets is immediate.

New and old algorithms can take advantage of this technique. Differential factor is that this technique is not affected by the number of evolution rules, which is the major problem traditional algorithms face. Here we return a solution with constant complexity as long as a few requirements are fulfilled.

We've been able to prove that, using this technique, the number of rules has little effect when reserving space for the structures we use. Moreover the use of this technique guarantees that the resulting multisets are obtained in a non deterministic manner.

In order to generalize this technique, a deeper analysis would be necessary to estimate the memory needed when certain rules are provided. This study will focus on four factors when building the structures:

- Number of evolution rules
- Applicability Benchmark of an evolution rule
- Number of symbols
- Benchmark associated to the symbols

This rigorous study will be explained in successive papers.

It is important to stress the point that, when input values are higher than the benchmark previously set, algorithms as *maximum applicability benchmark* [Fernández, Castellanos, Arroyo, tejedor, Garcia 2006] or algorithm based on Diophantine solution [1] should be used. A combination of either one of these algorithms with this technique would be ideal, as the performance will increase depending on the case we are working on.

In a real scenario, the possible amount of memory will determine the optimum number of evolution rules, the applicability benchmark of each rule, the number of required symbols and the benchmarks associated to these symbols.

Within the researching area: "*membranes computing",* this work proves that not only using parallelism is beneficial in terms of performance but also memory is really helpful under certain conditions. In this case, a proper utilization of memory let us build a structure which will reside in the physical and/or the virtual memory. This structure will have the information that relates multisets to evolution rules.

It will always be a smart option to combine the technique with traditional algorithms.

Bibliography

[Arteta, Fernandez, Gil 2008] A. Arteta, L.Fernandez, J.Gil. Algorithm for Application of Evolution Rules based on linear diophantine equations. Synasc 2008. Timisoara Romania September 2008.

[Ciobanu, Pérez-Jiménez, Ciobanu, Păun 2006] M. Pérez-Jiménez, G. Ciobanu, Gh. Păun. Applications of Membrane Computing, Springer Verlag. Natural Computing Series, Berlin, October, 2006.

[Fernández, Castellanos, Arroyo, tejedor, Garcia 2006] L. Fernández, J.Castellanos, F. Arroyo, J. tejedor, I.Garcia. New algorithm for application of evolution rules. Proceedings of the 2006 International Conferenceon Bioinformatics and Computational Biology, BIOCOMP'06, Las Vegas, Nevada, USA, 2006.

[Fernández, Martínez, Arroyo, Mingo 2005] L. Fernández, V.J. Martínez, F. Arroyo, L.F. Mingo. A Hardware Circuit for Selecting Active Rules in Transition P Systems. Proceedings of International Workshop on Theory and Applications of P Systems.Timisoara (Romania), September, 2005.

[Pan, Martin 2005] L. Pan, C. Martin-Vi de. Solving multidimensional 0-1 knapsack problem by P systems with input and active membranesl. Journal of Parallel and Distributed Computing Volume 65 , Issue 12 (December 2005)

[Păun 2000] Gh. Păun. Computing with Membranes. Journal ofComputer and System Sciences, 61(2000), and Turku

Center of Computer Science-TUCS Report nº 208, 1998.

[Păun 2005] Gh. Păun. Membrane computing. Basic ideas, results, applications. Pre-Proceedings of First International

Workshop on Theory and Application of P Systems,Timisoara (Romania), pp. 1-8, September , 2005.

[Qi, Li, Fu, Shi, You 2006] Zhengwei Qi, Minglu Li, Cheng Fu, Dongyu Shi, Jinyuan You. Membrane calculus: A formal . method for grid transactions. Concurrency and Computation: Practice and Experience Volume18,Issue14 , Pages1799-1809. Copyright © 2006 John Wiley & Sons, Ltd.

Authors' Information

Alberto Arteta *–Associate ProfessorTechnical University of Madrid.aarteta@eui.upm.es*

Angel Castellanos *–Departamento de Ciencias Basicas aplicadas a la Ingeniería Forestal. Escuela de Ingeniería Técnica Forestal. Technical University of Madrid, Avda. de Ramiro de Maeztu s/n 28040 Madrid, Spain. angel.castellanos@upm.es*

Ana MartinezCastellanos *–Departamento de Ciencias Basicas aplicadas a la Ingeniería Forestal. Escuela de Ingeniería Técnica Forestal. Technical University of Madrid, Avda. de Ramiro de Maeztu s/n 28040 Madrid, Spain.* ana.martinez@upm.es

LINEAR PROGRAM FORM FOR RAY DIFFERENT DISCRETE TOMOGRAPHY

Hasmik Sahakyan, Levon Aslanyan

***Abstract**: A special quality of discrete tomography problem solutions that requires the ray difference is considered. Two classes of reconstruction tasks of $(0,1)$-matrices with different rows are studied: matrices with prescribed column and row sums and matrices with prescribed column sums only. Both cases are known as algorithmically open problems. We reformulate them as integer programming problems. Depending on parameters obtained, the Lagrangean relaxation model and then variable splitting technique, or a greedy heuristics approaches are applied for getting approximate solutions. In later case an optimization version is considered, where the objective is to maximize the number of pair-wise different row rays, which in case of existence of a matrix, is equivalent to the requirement of row differences.*

***Keywords**: discrete tomography, (0,1)-matrices, integer programming*

***ACM Classification Keywords**: F.2.2 Nonnumerical Algorithms and Problems: Computations on discrete structures*

Introduction

Reconstruction algorithms that solve the so called inverse structural problems, have many applications in image processing, medicine, computer tomograph assisted engineering and design, electron microscopy, etc. There are a number of well known medical problems that require discrete reconstruction. For example, Onnasch and Prause [PrauseOnnasch, 1996] described an application of the discrete tomography technique, to routinely reconstruct the acquired biplane cardiac angiograms. Their model based reconstruction approach aims to recover the three-dimensional shape of the left or right chambers of the heart. In [SlumpGerbrands, 1982] Slump and Gerbrands presented a method based on a network flow approach that reconstructs the left ventricle of the heart from two projections.

Reconstruction of discrete sets (finite subsets of the two-dimensional integer lattice) from given projections, - is one of the main tasks of Discrete Tomography. Discrete sets can be presented as binary images. The line sum of a line through the image is the sum of the values of the points on this line. The projection of the image in a certain direction consists of all the line sums of the lines through the image in this direction. Any binary image with exactly the same projections as the original image is *a reconstruction of the image*.

Opposite to methods of Computerized Tomography which use several hundreds of projections in Discrete Tomography a few projections are available. The main problem arising here is that the reconstruction task is usually extremely underdetermined, i.e. there may be many different binary images with the same projections.

On the other hand for any set of more than two directions, the problem of reconstructing a binary image from its projections in those directions is NP-complete. For exactly two directions, the horizontal and vertical ones, it is possible to reconstruct an image in polynomial time. Already in 1957, Ryser found a necessary and sufficient condition for a pair of vectors being the horizontal and vertical projections of a discrete set ([Ryser, 1957]). In the proof of his theorem, Ryser also described a reconstruction algorithm. Another result of Ryser is the definition of the switching operation by which discrete sets having the same projections can be transformed into each other.

So, the problem of reconstructing a binary image from a small number of projections generally leads to a large number of solutions. To reduce the number of possible solutions, a priori information on the image being reconstructed is used. Two special geometrical properties/constraints are often imposed, - convexity and connectivity:

A matrix is horizontal convex (h-convex) if in every row the 1's form an interval, similarly, vertical convex (v-convex), - when in every column the 1's form an interval; and a matrix is hv-convex if it is h-convex and v-convex. A matrix is connected if the set of 1's is connected with respect to the adjacency relation, where every pixel is adjacent to its two vertical neighbours and to its two horizontal neighbours.

It is proven ([BarcucciDelLungoNivatPinzani,1996], [Woeginger,2001], [DurrChrobak,1999] that the existence problems of h-convex, v-convex, hv-convex matrices and the existence problem for connected matrices (polyominoes) are NP-complete; and the reconstruction problem for horizontal and vertical convex polyominoes can be solved in polynomial time.

In [DahlFlatberg, 2002] G. Dahl and T. Flatberg consider a variant of reconstructing hv-convex (0,1)-matrices, where instead of requiring the ones to occur consecutively in each row and column, they maximize the number of neighboring ones. Then the problem is reformulated as an integer programming problem and a solution method based on variable splitting is proposed.

We will consider another relevant concept in this context, - the requirement of different rows on matrix to be reconstructed. Two cases are studied with the requirement of row differences. First - reconstruction of $(0,1)$-matrices with prescribed row and column sums and different rows, and the second - reconstruction with prescribed column sums and different rows. Both are known as algorithmically open problems. The first problem we reformulate as an integer programming problem and use the Lagrangean relaxation and variable splitting technique for an approximate solution. Solving the second problem, we try to find in a constructive way a matrix with given parameters. Additionally we consider an optimization problem - instead of requiring all different rows, we maximize the number of different pairs of rows, which in case of existence of a matrix, is equivalent to the requirement of different rows. Then an approximation greedy algorithm is applied for constructing matrices with the maximum number of different pairs of rows.

$(0,1)$ matrices with different rows

In this section we consider a special case of discrete tomography, - to reconstruct the $(0,1)$-matrix with prescribed row and/or column sums and with all different rows.

Consider a $(0,1)$-matrix of size $m \times n$. Let $R = (r_1, \cdots, r_m)$ and $S = (s_1, \cdots, s_n)$ denote the row and column sums of the matrix respectively, and let $U(R,S)$ be the class of all $(0,1)$-matrices with row sums R and column sums S. A necessary and sufficient condition for the existence of a $(0,1)$ matrix of the class $U(R,S)$ was found by Ryser. Now we formulate the two basic problems:

P1. Existence of a $(0,1)$ matrix with the given row and column sums and with different rows

Given integer vectors $R = (r_1, \cdots, r_m)$ and $S = (s_1, \cdots, s_n)$. Is there a binary matrix $X = \{x_{i,j}\}$ in the class $U(R,S)$ with different rows?

P2. Existence of a $(0,1)$ matrix with the given column sums and with different rows

Given an integer vector $S = (s_1, \cdots, s_n)$. Is there a binary matrix $X = \{x_{i,j}\}$ with different rows whose column sums are given by $S = (s_1, \cdots, s_n)$?

No polynomial algorithms are known for solving P1 or P2, and they are known as open problems. The combinatorial origin of P1 is the hypergraph degree sequence problem, where the complexity is open even for the case $r_i = 3, i = 1, \cdots, m$. P2 comes from the n -dimensional unit cube subsets partitioning.

Below in this section we reformulate P1 as an integer programming problem and show the way of use of integer programming techniques to find the approximate solutions.

Let X be a $(0,1)$ -matrix of size $m \times n$. Obviously $\sum_{j=1}^{n} x_{i,j} = r_i, i = 1, \cdots, m$ and $\sum_{i=1}^{m} x_{i,j} = s_j, j = 1, \cdots, n$ are the row and column sums of the matrix.

The requirement of row difference in X which initially is a combinatorial property, will be presented algebraically, using the intersections of pairs of binary rows. Consider two rows i' and i'' that have the same row sum r . If these rows are different, then they intersect (by 1s) in less than r places. For determining the intersection size of this pair of rows we introduce additional n binary variables $y_{p(i',i''),j}$, so that these variable satisfy requirements $(y_{p(i',i''),j} = 1) \Leftrightarrow (x_{i',j} = 1) \,\&\, (x_{i'',j} = 1)$, where $p(i',i'')$ indicates the number (enumeration) of pair (i',i'') . Obviously this summarized picture can be provided by the following conditions:

$$\begin{cases} y_{p(i',i''),j} \le x_{i',j} \\ y_{p(i',i''),j} \le x_{i'',j} \\ y_{p(i',i''),j} \ge x_{i',j} + x_{i'',j} - 1 \end{cases}$$

In these terms the row pair intersection size is presented by formula $\sum_{j=1}^{n} y_{p(i',i''),j}$, and $\sum_{j=1}^{n} y_{p(i',i''),j} < r$ is the condition of distinction of these rows.

Indeed such conditions are necessary for only pairs of rows with the same sum. It is satisfactory to separate the row pairs with equal sums imposing these restrictions only for such rows. Below, in fact, we prefer to compose a system of inequalities for all pairs of rows.

Enumerate pairs of rows and let $p(i',i'')$ is the number of the pair (i',i'') , for $1 \le i' < i'' \le m$. Assuming that $r_1 \le \cdots \le r_m$, the requirement of different rows has been easily replaced with the following property: intersection size for each pair (i' , i''), $1 \le i' < i'' \le m$ is less than row sum of i'' -th row.

Now P1 can be formulated as follows: given the following system

$$(\text{P1}) \begin{cases} (1) \sum_{i=1}^{m} x_{i,j} = s_j, j = 1, \cdots, n \\ (2) \sum_{j=1}^{n} x_{i,j} = r_i, i = 1, \cdots, m \\ (3) \begin{cases} y_{p(i',i''),j} \le x_{i',j} \\ y_{p(i',i''),j} \le x_{i'',j} \\ y_{p(i',i''),j} \ge x_{i',j} + x_{i''j} - 1 \end{cases} \qquad 1 \le i' < i'' \le m, \; j = 1, \cdots, n \\ (4) \sum_{j=1}^{n} y_{p(i',i''),j} < r_{i''} \qquad 1 \le i' < i'' \le m \\ (5) x_{i,j} \in \{0,1\}, \; y_{i,j} \in \{0,1\} \end{cases}$$

where $R=(r_1,\cdots,r_m)$ and $S=(s_1,\cdots,s_n)$ are integer vectors and $r_1\le\cdots\le r_m$. $X=\{x_{i,j}\}_{m\times n}$ and $Y=\{y_{i,j}\}_{C_m^2\times n}$ are unknown variables. Is there a $(0,1)$ solution to this system?

Our immediate goal is to represent (P1) in the canonical form of a liner program instance: $Az\le b$.

First we explain the structure of vector z . z is introduced as concatenation of two parts: x and y :

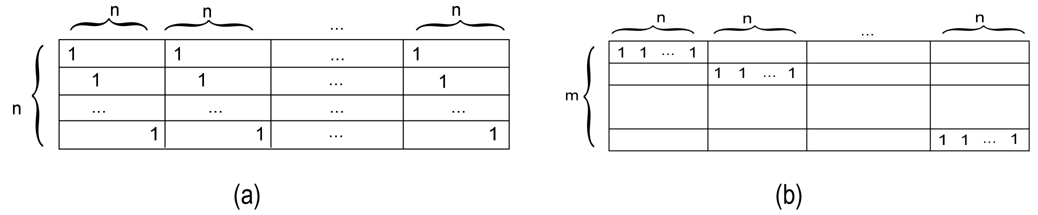
$$x=\begin{pmatrix} x_{1,1}\\ \vdots \\ x_{1,n}\\ \vdots \\ x_{m,1}\\ \vdots \\ x_{m,n}\end{pmatrix} \text{ and } y=\begin{pmatrix} y_{1,1}\\ \vdots \\ y_{1,n}\\ \\ \vdots \\ \\ y_{C_m^2,1}\\ \vdots \\ y_{C_m^2,n}\end{pmatrix},\quad z=\begin{pmatrix} x\\ y\end{pmatrix}, \text{ with } m\cdot n+C_m^2\cdot n \text{ coordinates.}$$

Vector b is a column vector consisting of several fragments: b_1 , b_2 , b_3 , b_4 , b_5 , where b_1 and b_2 are transposed forms of S and R respectively, b_3 is $2n\cdot C_m^2$-length column vector consisting of all 0s and b_4 is $n\cdot C_m^2$-length column vector consisting of all 1s. C_m^2 components of b_5 are composed by the pairs $p(i',i'')$, as the maximum values of $(r_{i'},r_{i''})$. $n+m+(3n+1)\cdot C_m^2$ is the total length of vector b .

Now compose the matrix A. It will have $n+m+(3n+1)\cdot C_m^2$ rows and $m\cdot n+C_m^2\cdot n$ columns and can be presented in form:

$$\begin{pmatrix} A_1 & B_1\\ A_2 & B_2\\ A_3 & B_3\\ A_4 & B_4\\ A_5 & B_5\end{pmatrix}$$

The structures of $(A_1)_{n\times(m\cdot n)}$ and $(A_2)_{m\times(m\cdot n)}$ is given below in figure 1



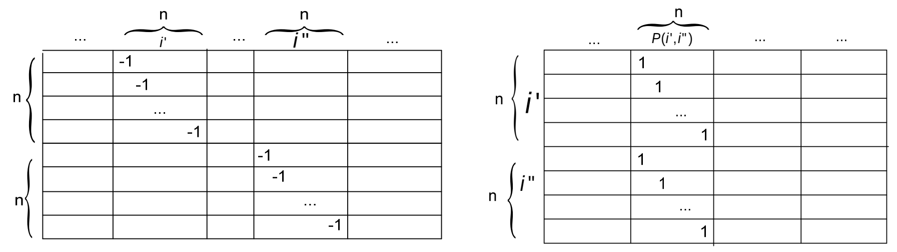
(a) (b)

Fig.1

(a) shows the placement of 1s in $(A_1)_{n\times(m\cdot n)}$ and (b) - in $(A_2)_{m\times(m\cdot n)}$.

$(B_1)_{n\times(C_m^2\cdot n)}$ and $(B_2)_{m\times(C_m^2\cdot n)}$ consist of all 0 components.

$(A_3)_{2n\cdot C_m^2\times(m\cdot n)}$ and $(B_3)_{2n\cdot C_m^2\times(C_m^2\cdot n)}$ consist of C_m^2 submatrices (vertically) of the following structures (one for each pair of rows (i',i'')) given in figure 2.

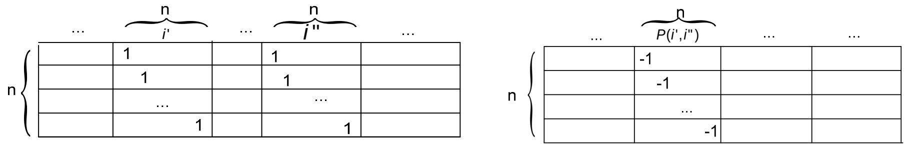


(a) (b)

Fig.2

(a) gives the placement of 1s in $(A_3)_{2n\cdot C_m^2\times(m\cdot n)}$ and (b) - in $(B_3)_{2n\cdot C_m^2\times(C_m^2\cdot n)}$.

Similarly, $(A_4)_{n\cdot C_m^2\times(m\cdot n)}$ and $(B_4)_{n\cdot C_m^2\times(C_m^2\cdot n)}$ are the vertical concatenation of the following structures given below in figure 3 that are constructed for each pair of rows (i',i'') .



(a) (b)

Fig.3

(a) corresponds to $(A_4)_{n\cdot C_m^2\times(m\cdot n)}$ and (b) - to $(B_4)_{n\cdot C_m^2\times(C_m^2\cdot n)}$.

$(A_5)_{C_m^2\times(m\cdot n)}$ consists of all 0 components and $(B_5)_{C_m^2\times(C_m^2\cdot n)}$ has the form given in figure 4:

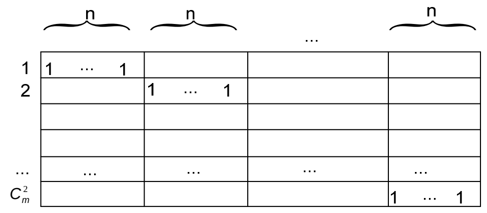


Fig.4

Now A_1 and A_2 (with b_1 and b_2) provide column and row sums respectively, - (1) and (2) conditions in (P1). A_3, B_3 and A_4, B_4 (with b_3 , b_4) are composed for presenting intersections of row pairs, - (3). And finally, B_5 with b_5 is to provide the condition (4), - row differences.

Below in figure 5 we present the whole scheme.

This sparse matrix codes the discrete tomography P1 problem in terms of integer linear programming. Several classes of integer programming are known solvable in polynomial time – problems with total unimodular matrices, set network problem, etc. Matrix A is not as simple but the given form helps to see and construct the appropriate relaxations to receive and apply approximate solution algorithms.

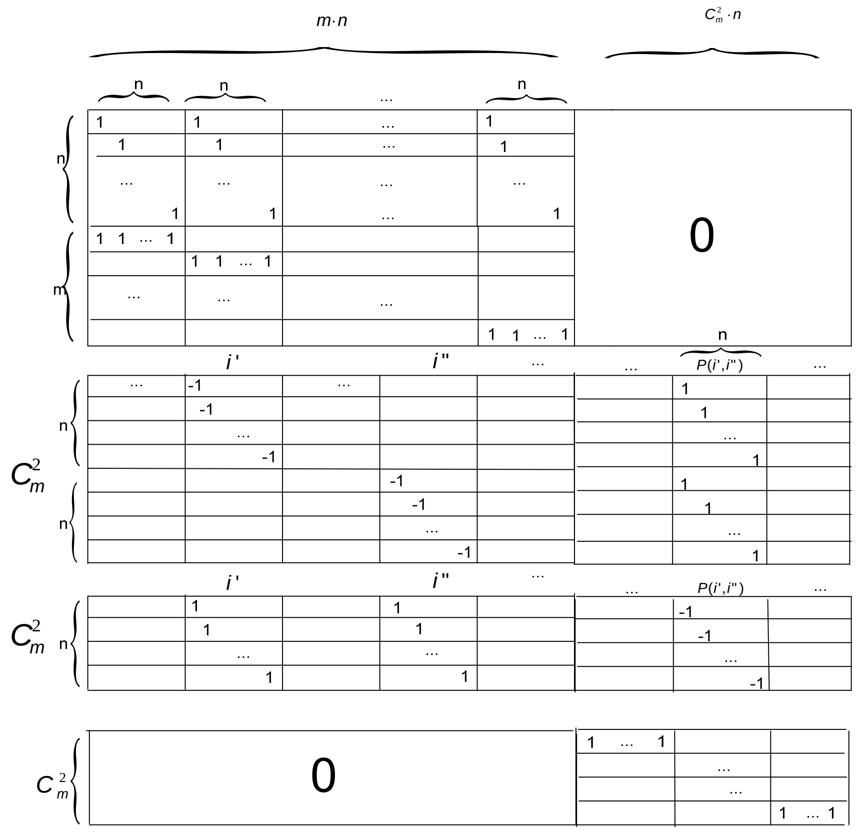


Fig.5

Let us consider now also the problem P2 and the problem of representing it in a linear program system.

Let X be a $(0,1)$-matrix of size $m \times n$ for problem P2. $\sum_{i=1}^{m} x_{i,j} = s_j, j = 1, \cdots, n$ provides the column sums of the matrix. For presenting the condition of different rows of X, we are considering the following representation of rows. i-th row of X, as a binary vector of size n can be represented in form $2^{n-1}x_{i,1} + 2^{n-2}x_{i,2} + \cdots + 2^0 x_{i,n}$. If two rows are different then the numerical values of the corresponding representations are different. We get the following system:

$$
(\mathrm{P2}) \begin{cases} (1) \sum_{i=1}^{m} x_{i,j} = s_j, j = 1, \cdots, n \\ (2) \sum_{j=1}^{n} 2^{n-j} x_{i',j} \neq \sum_{j=1}^{n} 2^{n-j} x_{i'',j} \quad 1 \le i' < i'' \le m \\ (3) x_{i,j} \in \{0,1\} \end{cases}
$$

Enumerate pairs of rows and let $p(i', i'')$ be the number of the pair (i', i''). We introduce integer variables $y_1, \cdots, y_{C_m^2}$, where $y_{p(i',i'')} = \sum_{j=1}^{n} 2^{n-j} x_{i',j} - \sum_{j=1}^{n} 2^{n-j} x_{i'',j}$.

Now P2 can be formulated in the following way: given the system

$$(P2)\begin{cases}(1)\sum_{i=1}^{m} x_{i,j} = s_j, j=1,\cdots,n \\ (2) y_{p(i',i'')} = \sum_{j=1}^{n} 2^{n-j}(x_{i',j} - x_{i'',j}) \quad 1 \le i' < i'' \le m \\ (3)\ y_j \ne 0, \quad j=1,\cdots,C_m^2 \\ (4) x_{i,j} \in \{0,1\} \\ (5)\ y_j \in Z\end{cases}$$

where $S=(s_1,\cdots,s_n)$ is an integer vector and $X=\{x_{i,j}\}_{m\times n}$ and $Y=\{y_j\}_{C_m^2}$ are unknown variables. Is there a solution of this system?

In the same way as in the previous case, this system can be presented in form $Az \le b$.

Thus we get two cases of integer programming problems and now techniques of integer programming can be applied for solving them. First the linear programming relaxation is considered and experiments are done in existing linear programming software environment. The results were not adequate. Afterward an experimentation software system is created which provides an environment for treatment of combinatorial problems. In particular P1 and P2 are relaxed in Lagrangean manner and experimented for different classes of (0,1)-matrices. The results are satisfactory for a number of specific cases.

Lagrangean relaxation and variable splitting for P1 and P2

We consider Lagrangean relaxation of P1. There are many ways in which a given problem can be relaxed in a Lagrangean fashion. We will use variable splitting technique - we split our problem into separate vertical and horizontal subproblems, then the horizontal subproblem is further separated into subproblems for each pair of rows.

We duplicate variables $x_{i,j}$, getting 2 independent sets of variables $x_{i,j}^h$ and $x_{i,j}^v$, and then dualize the copy (duplication) constraint using Lagrangean multipliers $\lambda_{i,j}$.

$$(P1LR)\begin{cases}\max\{\sum \lambda_{i,j}(x_{i,j}^h - x_{i,j}^v)\} \\ (1)\sum_{i=1}^{m} x_{i,j}^v = s_j, j=1,\cdots,n \\ (2)\sum_{j=1}^{n} x_{i,j}^h = r_i, i=1,\cdots,m \\ (3)\begin{cases} y_{p(i',i''),j} \le x_{i',j}^h \\ y_{p(i',i''),j} \le x_{i'',j}^h \qquad 1 \le i' < i'' \le m,\ j=1,\cdots,n \\ y_{p(i',i''),j} \ge x_{i',j}^h + x_{i'',j}^h - 1 \end{cases} \\ (4)\sum_{j=1}^{n} y_{p(i',i''),j} < r_{i''} \qquad 1 \le i' < i'' \le m \\ (5) x_{i,j}^h, x_{i,j}^v \in \{0,1\},\ y_{i,j} \in \{0,1\}\end{cases}$$

Split the problem into sub problems – horizontal and vertical:

$$\text{(P1LR-h)} \begin{cases} \max\{\sum \alpha_{i,j} x^h_{i,j}\} \\ (1)\sum_{j=1}^{n} x^h_{i,j} = r_i, \quad i = 1,\cdots,m \\ (2)\begin{cases} y_{p(i',i''),j} \le x^h_{i',j} \\ y_{p(i',i''),j} \le x^h_{i'',j} \\ y_{p(i',i''),j} \ge x^h_{i',j} + x^h_{i'',j} - 1 \end{cases} \quad 1 \le i' < i'' \le m,\ j = 1,\cdots,n \\ (3)\sum_{j=1}^{n} y_{p(i',i''),j} < r_{i''} \quad 1 \le i' < i'' \le m \\ (4) x^h_{i,j} \in \{0,1\}, y_{i,j} \in \{0,1\} \end{cases} \qquad \text{(P1LR-v)} \begin{cases} \max\{\sum \beta_{i,j} x^v_{i,j}\} \\ (1)\sum_{i=1}^{m} x^v_{i,j} = s_j, j = 1,\cdots,n \\ (2) x^v_{i,j} \in \{0,1\} \end{cases}$$

Using similar reasons P1LR-h is then split into subproblems for each pair of rows.

Further we apply an iterative procedure to find the optimisation coefficients $\lambda_{i,j}$. On each iteration we consider C^2_m +1 separate subproblems (C^2_m horizontal and 1 vertical).

The obtained vertical and horizontal subproblems are simple and can be easily solved. Solutions of subproblems of this and similar kind are gathered in special library in the above mentioned experimentation software.

An analog reasoning is true for P2 .

Construction of a $(0,1)$ matrix with the given column sums and with different rows

In this section we consider the problem P2 from another point of view. This time we look for a possible solution - by constructing a special solution/matrix which is in a specific standard format.

Let an integer vector $S = (s_1,\cdots,s_n)$, $0 \le s_i \le m$, $i = 1,\cdots,n$ is given. If there exists an $m \times n$ $(0,1)$-matrix with all different rows and with s_i column sums, then after a finite number of row transpositions it can be transformed into the matrix with the same column sums, which has the “canonical form” – where each column consists of continuous intervals of 1’s (higher part) and 0’s (lower part) such that they split the intervals of the previous column in 2 parts. Therefore if P2 has a solution, it can be found (constructed) by algorithms which compose matrices in column-by-column fashion. We will refer to the construction version of P2 as P2-C.

The first column is being constructed by allocating s_1 1’s to the first s_1 rows-positions followed by the $m - s_1$ 0’s in others. Two intervals is the result: – the s_1 interval of 1’s, and the $m - s_1$ interval of 0’s. We denote these intervals $d_{1,1}$ and $d_{1,2}$. Hereafter the first index will indicate the number of column and the second – the number of interval within the column. Intervals with odd numbers consist of 1s, and intervals with even number consist of 0s. So for the first column we have the following system:

$$\begin{cases} d_{1,1} + d_{1,2} = m \\ d_{1,1} = s_1 \end{cases}$$

We construct the second column putting s_2 ones and $m-s_2$ zeros in $d_{2,1}$, $d_{2,2}$, $d_{2,3}$ and $d_{2,4}$ intervals such that $d_{2,1}$ and $d_{2,3}$ are filled by 1's, and $d_{2,2}$, $d_{2,4}$ - by 0's, and

$$\begin{cases} d_{2,1}+d_{2,2}+d_{2,3}+d_{2,4}=m \\ d_{2,1}+d_{2,2}=s_1 \\ d_{2,1}+d_{2,3}=s_2 \end{cases}$$

In general, k-th column consists of $d_{k,1},d_{k,2},\cdots,d_{k,2^k}$ intervals (among them can be 0-length intervals) filled by 0's and 1's accordingly, such that

$$\begin{cases} \sum_{i=1}^{2^k} d_{k,i}=m \\ \sum_{i=0}^{2^{k-(j+1)}} (d_{k,2^{j+1}\cdot i+1}+d_{k,2^{j+1}\cdot i+2}+\cdots+d_{k,2^{j+1}\cdot i+2^j})=s_{k-j}, \quad j=0,\cdots,k-1 \end{cases}$$

So we formulate the following problem:

Given a system of $d_{n,1},d_{n,2},\cdots,d_{n,2^n}$ variables

$$\text{(P2-C)} \begin{cases} \sum_{i=1}^{2^n} d_{n,i}=m \\ \sum_{i=0}^{2^{n-(j+1)}} (d_{n,2^{j+1}\cdot i+1}+d_{n,2^{j+1}\cdot i+2}+\cdots+d_{n,2^{j+1}\cdot i+2^j})=s_{n-j}, \quad j=0,\cdots,n-1 \\ d_{i,j}\in\{0,1\} \end{cases}$$

Is there a solution of the system? As in previous cases we reformulate our problem as an integer programming task. But in this case we get s a system with the exponential number of variables.

Below we will consider an optimization version of P2-C and bring an approximation greedy algorithm for its solution.

During the construction of matrices in column-by-column fashion, partitioning of the intervals in each step can be performed by different ways - following different goals. Let assume that the partitioning of intervals aims to maximize some quantitative characteristics, which leads to the matrices with different rows in case when the later exists. One of such characteristics - the number of pairs of different rows – is considered in [S, 1995]. An approximation greedy algorithm, which constructs the target (0,1)-matrices in the above described column-by-column fashion of partitioning is considered. The algorithm provides the optimal construction of each column – i.e. the construction, which provides the maximal number of new (i,j) pairs of different rows in each step. It is proven that the optimal construction of each column is provided by partitioning, which distributes the difference $s_k-(m-s_k)$ "homogeneously" on all current non atomic intervals.

Bibliography

[PrauseOnnasch, 1996] G. P. M. Prause and D. G. W. Onnasch, Binary reconstruction of the heart chambers from biplane angiographic image sequence, IEEE Transactions Medical Imaging, 15 (1996) 532-559

[SlumpGerbrands, 1982] Slump, C.H., Gerbrands, J.J., A network flow approach to reconstruction of the left ventricle from two projections. Comput. Gr. Im. Proc., 18, 18.36 (1982).

[DurrChrobak, 1999] Durr Ch., Chrobak M., Reconstructing hv-convex polyominoes from orthogonal projections, Information Processing Letters 69 (1999) pp. 283-291.

[BarcucciDel LungoNivatPinzani, 1996] E. Barcucci, A. Del Lungo, M. Nivat, and R. Pinzani, Reconstructing convex polyominoes from horizontal and vertical projections, Theoret. Comput. Sci., 155:321{347, 1996.

[Woeginger, 2001] G.J. Woeginger. The reconstruction of polyominoes from their orthogonal projections, Inform. Process. Lett., 77:225{229, 2001.

[Ryser, 1966] H. J. Ryser. Combinatorial Mathematics, 1966.

[DahlFlatberg, 2002] G. Dahl and T. Flatberg, Lagrangian decomposition for reconstructing hv-convex (0,1)-matrices, Report 303, ISBN 82-7368-255-2, ISSN 0806-3036, December 2002

Authors' Information

Hasmik Sahakyan *– Leading Researcher, Institute for Informatics and Automation Problems, NAS RA, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: hasmik@ipia.sci.am*

Levon Aslanyan *– Head of Department, Institute for Informatics and Automation Problems, NAS RA, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: lasl@.sci.am*

PIGMENTED RAT-BASED VISION FOR ARTIFICIAL INTELLIGENCE APPLICATIONS

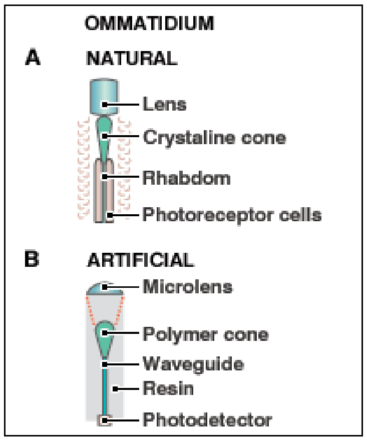
Francisco J. Cisneros de los Rios , Isabel Martín Moreno-Cid , Abel Sanchez-Jimenez , Juan Castellanos, Fivos Panetsos

Abstract: *One of the most important objectives of artificial vision is the development of bioinspired and biomimetic robot vision as well as the development of bionic eyes for the blind. Depending on the specific application different eye models can be used but the most ambitious is the development of a human-like eye. However, human eye is extremely complex and if we even design such a visual device the amount of information to process should be computationally intractable. Normally low-resolution image quality and low visual acuity would be sufficient for our purposes, so simpler biological models could represent excellent computational alternatives. In the present communication we propose to use the visual system of the rat and we justify our proposal by proving that this model fits perfectly the requirements of our applications.*

Introduction

In the last decades artificial vision has been integrated in our life, mainly in industrial applications like quality control, fault detection or robot guidance. More ambitious objectives include bio-inspired and biomimetic robot development and bionic eyes for the blind (Panetsos F., et al., 2009). Different eye models can be used depending on the desired application, spanning from insect eyes to the human retina that represent the highest goal in artificial vision (B. Dietrich 2007 y R Hendriks 2008).

Insect eyes are compound eyes usually consisting of thousands of tiny lens-capped optical units, the ommatidia. Ommatidia are formed by a lens, a crystalline cone and the rhabdom channel that contains the photoreceptors (Fig.1, left). The light is conveyed to the photoreceptors through the lens and the rhabdom channel. Photoreceptors are connected to the optical nerve cells to which transmit information. In insect eyes ommatidia are orientated in slightly different directions and generate slightly overlapping images. This way the compound eye creates a low-resolution mosaic image but with excellent performances in movement detection. Artificial insect vision reproduces this eye design (Fig.1, left) (K-H Jeong 2006); dibujar del ojo del insecto y el ojo artificial) [http://axxon.com.ar/mus/Insectos.htm]



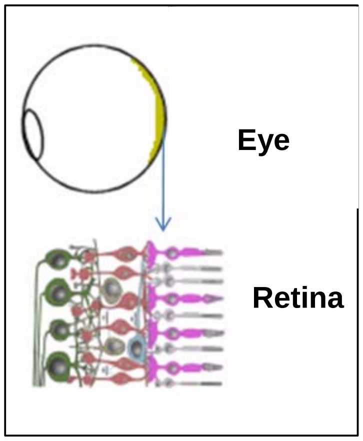


Figure 1. Left, insect vision: natural (A) and artificial (B). Right, mammal vision: natural

Mammal retina is the eyeball’s inner lining onto which images of the outside world are projected by the light that enters the eye (Fig.1, right). Images formed at the retina consist of continuous multidimensional stimuli formed by spatiotemporal and intensity patterns of different wavelengths of light. A sheet of photoreceptors transforms these images into another multidimensional pattern made by spatiotemporal and intensity pattern of electrical signals. Photoreceptors are connected to the ganglion cells to which they transmit their electrical activity. Signals from the ganglion cells are transmitted to the thalamus and then to the cortex, where visual perception takes place (P. Kara 2002).

An artificial retina connected to the higher stations of the visual pathway should generate artificial visual percepts (sensations of spots of light) similar to the natural ones (G. A. Horridge 2006). Similarly, an artificial retina used by a bioinspired robot should provide eye-like images to the robot’s brain. However, human eye is extremely complex and if we even design such a visual device, the amount of information to process should be computationally intractable. Fortunately, the eyes of most mammals, including humans, are very similar their main differences found in the distribution of the photoreceptors (H. Perry 1981). So simpler mammal eyes could be proposed as experimental paradigms for mammal-like artificial vision. In particular, the similarities in the effects of visual deprivation on visual acuity between rats and other mammals confirm that rats are a good model system for our purposes (Prusky et al. 2002).

Biomimetic robot vision does not normally need high acuity and visual neuroprostheses will be implanted to blind subjects through some hundred or at maximum thousand of microelectrodes. In both cases, a low-resolution image quality or low visual acuity would be sufficient for our purposes. The visual system of the rat fits perfectly our requirements and it could be used as a model for our study.

Visual acuity for robotics and neuroprosthetics

Visual acuity is measured in cycles per degree (cpd), a measurement of the number of lines that can be seen as distinct within a degree of the visual field. It reaches 30.0 cpd in humans, 1.0-1.2 cpd in pigmented rats and 0.3-0.5 in albino rats (Hughes, 1979; Prusky et al. 2002). The maximal spatial resolution of dark-and light-adapted retinal ganglion cells recorded from optic tract of hooded rat is, however, only about 1.2 cycles/degree or about 25 min of visual angle (A. Hughes 1979). Ganglion cells are relatively evenly distributed across the rat retina, the density varying from 3,000 in the central area to 600 cells/mm2 in the periphery (5:1, (G. Paxinos, 2004).The denser the ganglion cells the higher the acuity at that point of the retina, it reaches 6,774 cells/mm2 in rats and 38,000 cells/mm2 in humans (Curcio and Allen 1990). A theoretical estimation of rat’s maximum acuity indicates 1.5 cpd consistent with experimental data (Heffner and Heffner 1992). In Table 1 are showm the physical characteristics of the three eyes.

Table 1. Comparison of visual properties of the human and rat eye

	Humans	Pigmented rat	Albino rat
Best visual acuity	20/20	20/600	20/1200
Resolution	~ 30 cpd	~ 1.0 cpd	~ 0.5 cpd
Maximum density of photoreceptors (fovea)	10.000 cells/mm2	3,000 cells/mm2	3,000 cells/mm2
Depht vision	2.3 m - ∞	0.07 m - ∞	0.07 m - ∞
Maximum monocular visual field (normal vision)	60º	max 60º	max 60º
Minimum monocular visual field (low vision)	>20º	>20º	>20º

To establish the visual acuity several high-contrast tests with letters of different sizes (optotypes) are presented to the observer at a fixed distance. The smallest letter the observer is able to detect or recognize (depending on the task assigned to the subject) is taken as threshold (Baily, 1976). In Fig. 2, in the first line we can observe the perception of two optotypes by humans and the two rats where it is clear that pigmented rats vision could substitute the human one with good performances. The first character of this optotype is 9.0 cm at 110 cm from the rat. The next character 4,5 cm placed at the same distance. If we translate Prusky's cpd measurements into vision chart measurements we obtain about 20/600 for the pigmented rat and 20/1200 for the albino rat. In the middle line a more intuitive example corroborates the suitability of the pigmented rats model while in the bottom a real scene is shown as perceived by the three eyes. Pigmented rat is able to distinguish the same objects as a human observer while an albino rat cannot.

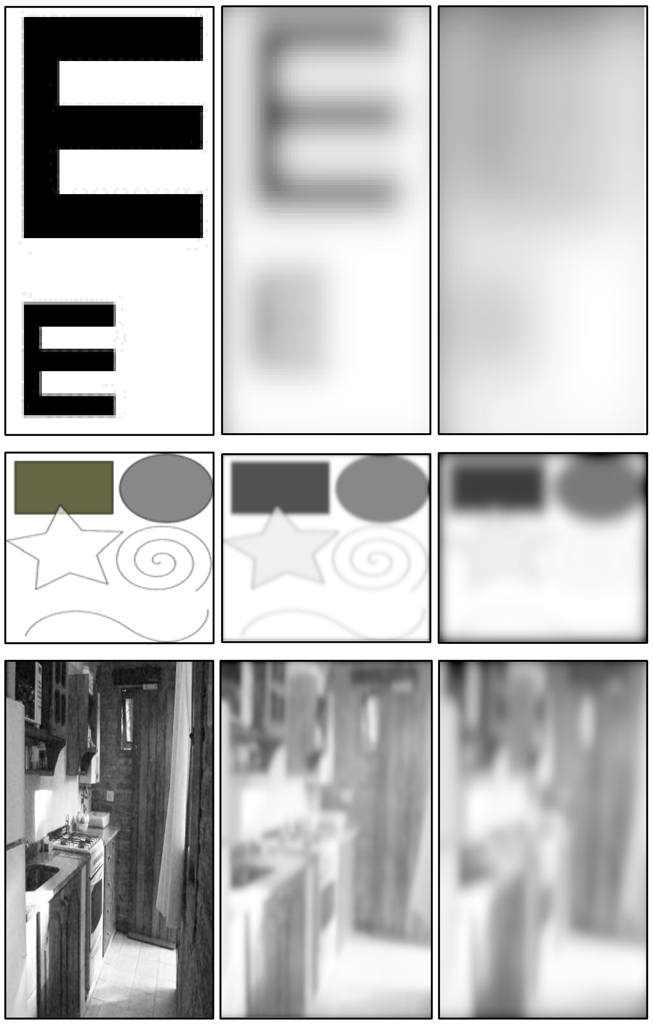


Figure 2. Top: A seeing optotype e as perceived by a human (left), a pigmented rat with visual acuity of 1.2 cpd (center) and an albino rat with visual acuity 0.5 cpd (right). Middle: Different drawings as perceived by humans and the two types of rats prove the suitability of the pigmented rat model for artificial vision applications. Bottom: Areal scene as perceived by the three eyes.

Given their visual resolution, pigmented rats should be able to distinguish a 9.0 x 9.0 cm optotype at a distance of 1.1 m and an albino rat the same optotype at 4.5 cm. The minimun resolution to define an optotype is 5 x 5 pixels. Taking into account the loss of information when we reduce the size of an image and the different visual angles, the following dimensions of an artificial retina should be necessary (Fig. 3). In Fig. 4 we present the vision capabilities of the rats at XXº in two real environments, one close and the other open.

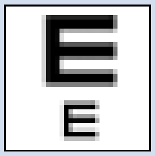
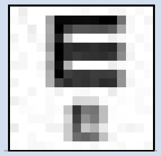
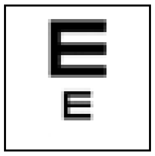
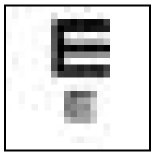
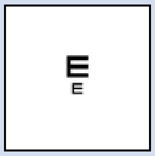
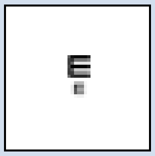
Degrees	Pigmented rat (1.2 cpd)	Albino rat (0.5 cpd)
20	32 x 32 px 	16 x 16 px 
30	44 x 44 px 	22 x 22 px 
60	88 x 88 px 	44 x 44 px 

Figure 3. Different optotypes as perceived by the two types of rats using different vision angles.

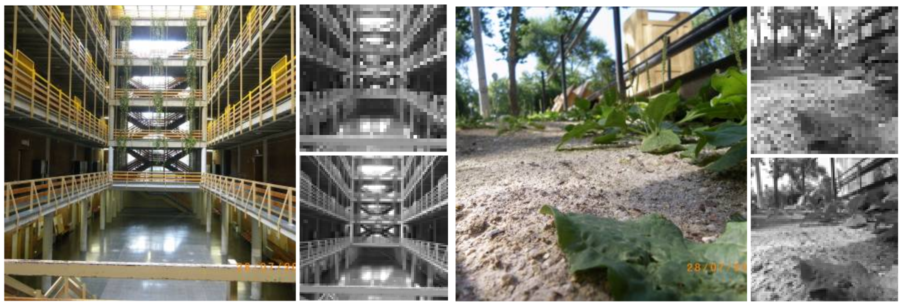


Figure 4. A close (left) and an open environment (right) as perceived by the two types of rats using XXº angle.

Conclusions

Any visual neuroprosthetic design have to consider the less amount of information to inject to the Central Nervous System, specially for the number of implanted electrodes. "Ad hoc" information processing and coding strategies will reduce the dimension of the thalamic input. On the other hand, an important research objective of artificial vision for robotics is to reduce the amount of information that is obtained by the visual system and needs very time and memory consuming processing. According to our study, relatively good performances can be achieved by artificial systems using visual acuity characteristics as low as those of the pigmented rats. Our proposal has the additional advantage that rats can behave well with their visual system although their optical apparatus is bad, the neuronal organization of the retina is very rough and their visual acuity is low. Therefore with our model we could use obtain good performances in either biomimetic robotics or visual neuroprostheses using a rat-mimetic visual system.

References

[A. Hughes 1979] Hughes A, Wässle H.An estimate of image quality in the rat eye.Hughes A, Wässle H. Invest Ophthalmol Vis Sci. 1979 Aug;18(8):878-81

[B. Dietrich] B. Dietrich. On the vision of insects Current Biology.17, 329–335, February 20, 2007

[C.A.Curcio, 1990] C.A. Curcio , K.A. Allen. Topography of ganglion cells in human retina. J Comp Neurol. 1990 Oct 1; 300(1):5-25

[G. A. Horridge 2006] El ojo compuesto de los insectos. *Science* 28 April 2006: Vol. 312. no. 5773, pp. 557 - 561 DOI: 10.1126/science.1123053

[G. Prusky, 2002] G.T. Prusky, K.T. Harker, R. M. Douglas, I. Whishaw. Variation in visual acuity within pigmented, and between pigmented and albino rat strains, Behav Brain Res. 2002 Nov 15;136(2):339-48

[F. Panetsos, 2009] F. Panetsos, E.Diaz-de Cerio, A.Sanchez-Jimenez, C.Herrera-Rincon. Thalamic visual neuroprostheses: Comparison of visual percepts generated by natural stimulation of the eye and electrical stimulation of the thalamus. NER2009

[H. Perry 1981] Spatial contrast sensitivity of cells in the lateral geniculate nucleus of the rat. J. Physiol. (1981), 315, 69-79 69

[G. Paxinos, 2004] G. Paxinos. The Rat Nervous System, Cap.32 Visual System, 3th edition 2004. Elsevier.

[K-H Jeong 2006] Ki-Hun Jeong, Jaeyoun Kim, Luke P. Lee Biologically Inspired Artificial Compound Eyes. *Science* 28 April 2006: Vol. 312. no. 5773, pp. 557 - 561 DOI: 10.1126/science.1123053

[P. Kara 2002] Prakash Kara, John S. Pezaris, Sergey Yurgenson, and R. Clay Reid. The spatial receptive field of thalamic inputs to single cortical simple cells revealed by the interaction of visual and electrical stimulation. October, 2002, Harvard Medical School, Boston, MA,

[R Hendriks 2008] Retinal Implants, Still in Their Infancy, Provide a New Vision of the Future. Richard Hendriks. March 2008 Bionic Eyes. Medical Device Marketplace Review

[R.S. Heffner, 1990] RS, Heffner, H.E. Heffner. Visual factors in sound localization in mammals. J Comp Neurol. 1992 Mar 15;317(3):219-32.PMID: 1577997 [PubMed - indexed for MEDLINE.

Authors' Information

Francisco J. Cisneros de los Rios *– Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: kikocisneros@gmail.com*

Isabel Martín Moreno-Cid *– Neurocomputing and Neurorobotics Research Group, Universidad Complutense de Madrid, 28037 Madrid, Spain. School of Optics, Universidad Complutense de Madrid, 28037 Madrid, Spain: email: fivos.panetsos@opt.ucm.es*

Abel Sanchez-Jimenez *– Neurocomputing and Neurorobotics Research Group, Universidad Complutense de Madrid, 28037 Madrid, Spain. School of Optics, Universidad Complutense de Madrid, 28037 Madrid, Spain: email: fivos.panetsos@opt.ucm.es*

Juan Castellanos *– Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: jcastellanos@fi.upm.es*

Fivos Panetsos *– Neurocomputing and Neurorobotics Research Group, Universidad Complutense de Madrid, 28037 Madrid, Spain. School of Optics, Universidad Complutense de Madrid, 28037 Madrid, Spain: email: fivos.panetsos@opt.ucm.es*

VARIETIES OF BIOLOGICAL INFORMATION: A MOLECULAR RECOGNITION APPROACH TO SYSTEMS BIOLOGY AND BIOINFORMATICS

Jorge Navarro, Ángel Goñi-Moreno & Pedro C. Marijuán

Abstract: *Bioinformatic and systems biology developments should be accompanied not only by a plethora of computer tools, but also by an in-depth reflection on the distinctive nature of biological information. In this work we attempt a consistent approach to the multiple varieties of information in the living cell by starting out from the conceptualization of molecular recognition phenomena. Subsequently, an elementary approach to the "informational architectures" behind cellular complexity may be chartered. In the interplay of the different informational architectures two cellular subsystems should be highlighted: on the one side the transcriptional regulatory network, and on the other, the cellular signaling system that is in charge of the interrelationship with the environment. The embodiment of functional agents and the peculiar handling of DNA sequences along the evolutionary process will suggest a parallel with the von Neumann scheme of modern computers, including the cellular capability to "rewrite the DNA rules" along ontogenetic development.*

Keywords*: Molecular recognition, Informational architectures, DNA addresses, Transcriptional regulatory network, Cellular signaling system, von Neumann scheme.*

ACM Classification Keywords*: D. Software. D.1 Programming Techniques*

Introduction: molecular recognition as a key to biological organization

The approach followed here is based on a bottom-up strategy, linking informational architectures, and in general the structures of cellular organization, to molecular recognition modalities [Conrad, 1996], [Marijuán, 2003, 2009].

Molecular recognition is but one of the fundamental territories of chemistry. Paraphrasing Shaik's words [Weinhold and Landis, 2007], it is the central element from which an entire bio-chemical and evolutionary universe is constructed. Actually, molecular *specificity* and molecular *affinity*, which provide the ground for any molecular recognition phenomena are amongst the most essential concepts of classical chemistry and molecular sciences —as all chemical reactions are based on the relative specificity of the intervening molecular partners and on their mutual affinity or free energy availability. It means, in other words, that the "making and breaking of bonds" is what makes possible the mutual recognition and the formation of complexes between biomolecular partners.

By far, it is in the highly heterogeneous molecules that constitute living matter where the chemical phenomenon of molecular recognition reaches its maximal universality, ubiquity, and combinatory capabilities. The myriad of molecular recognition encounters at the cytoplasm of a bacterium are taking place in a highly organized and systematic way: no "insulating wires" are needed. And this is one of the most remarkable information processing resources of the living cell (a wired cell would be unthinkable of at the molecular scale!).

Apparently, all the multitude of specific molecular matching events in the cell are occurring on a case by base basis, beyond any useful molecular taxonomy, but this is not the case, as we are going to discuss. Precisely we will distinguish classes of informational architectures based on molecular recognition considerations. The most important functional interrelationship in the living cell concerns the population of protein biomolecular agents which are built by means of transcriptional and translational processes performed upon the sequential arrangement of nucleic acids. This means the class of “diluted” architectures based on networks of enzymes and proteins versus the class of “sequential” architectures.

We will particularly discuss how a network of gene expression relationships is organized within a concrete living cell (*Mycobacterium tuberculosis*) and how the topological governing of this network is deployed by the cellular signaling system in charge of the interrelationship with the environment. The embodiment of functional agents and the peculiar handling of DNA sequences along the evolutionary process will suggest a parallel with the von Neumann scheme of modern computers, including the cellular capability to “rewrite the DNA rules” along ontogenetic development.

Looking for a unitary molecular recognition background

Regarding the question of how many specific recognition encounters may be distinguished within the biomolecular “soup” of any living cell [Goodsell, 1991], it is surprising that in spite of the ubiquity and universality of biomolecular recognition phenomena, they are not well focused in their general categorization yet.

Molecular recognition, like any other specific chemical reaction, simply implies the “making and breaking of bonds”. The problem with biomolecular recognition instances is that they involve an amazing variety and combinatorics of almost any type of chemical bond (and particularly of Coulombian motifs), which together provide specificity and affinity to the intermolecular encounters: covalent bonds, hydrogen bonds, hydrophobic / hydrophilic forces, dipole forces, van der Waals forces, ionic Coulombian forces, etc. Dozens or even hundreds of weak bonds may participate, for instance, in the formation of a protein-protein specific complex.

Quite probably, measuring molecular recognition and establishing its crucial parameters and variables can only be realized biologically on a case-by-case basis. At least this is the current trend in most molecular biological and molecular dynamic approaches.

A few references, however, could provide some interesting insights about molecular-recognition generalities. First, [Meggs,1998] about “biological homing”, contemplated particularly from a Coulombian “lock and key” combinatory point of view; then [Lin, 2001] about the changes in thermodynamic entropy and entropy of mixing derived from molecular similarity changes; and finally [Carlton, 2002], with original proposals for measuring the information content of any complex molecular system.

Symmetry considerations

The usefulness and depth of symmetry considerations in molecular recognition phenomena, as emphasized by [Lin, 2001], are self-evident. Symmetry allows a direct classification of biomolecular recognition occurrences by means of three ordering categories: *identity, complementarity,* and *supplementarity.* They respectively mean: recognition by sharing identical molecular properties (e.g., self-organization of phospholipids in membranes), recognition by means of complementary properties of the molecular partners (e.g., moieties, or the nucleic acids’ double helix), plus recognition through a quasi-universal capability to wrap or envelop any molecular shape by building a complex molecular scaffold of weak bonds around the target (e.g., enzymic active sites, protein complexes).

In the *supplementarity* case (not contemplated by Lin’s approach), the partial surfaces involved are inherently sloppy in their specificity and have a very variable affinity. Possibly we could keep calling *complementarity* to this facultative and highly variable interrelationship, but at the cost of leaving in the dark a very interesting distinction with respect to the very clean and holistic matching between complementary moieties (molecular fractions). Let us illustrate the additional difference introduced by means of a literary metaphor. Daniel Defoe’s character, Mr. Gulliver, could be matched by *identity* with his seaman fellows (dressing the same uniform, for instance), or by *complementarity* in the relationships with his loving wife Mrs. Gulliver; but throughout *supplementarity* he was matched by a motley crew of Lilliputians who had built many kinds of small 'bonds' around his bodily parts.

Informational architectures

From an organizational point of view, the previous categories based on symmetry considerations would be reflecting the global distribution of molecular functions within the cell, the different classes of informational architectures (see Table 1):

- *identity* in the structural self-organization of membrane and cytoskeleton support systems (the structural & support architecture),

- *complementarity* in the informational memory-banks of nucleic acids (the sequential architecture),

- *supplementarity* in the active sites and recognition-surfaces of enzymic molecular machinery (the diluted, processing architecture).

Identity	Complementarity	Supplementarity
nucleotides/RNA	RNA/RNA pairing	enzymes/substrates
nucleotides/DNA	RNA/DNA pairing	enzymes/effectors
amino acids/protein chains	RNA/ribozymes	enzymes/cofactors
phospholipids/membranes	RNA/ribosomes,	enzymes/proteins
tubulins/microtubules	RNA/amino acids	antibodies/antigens
actins/microfilaments	RNA/ribonucleoproteins	receptors/peptides
clathryn/vesicles	DNA/DNA pairing	receptors/transmitters
carbohidrates/glycoproteins	DNA/polimerases	receptors/ligands
lipids/lipoproteins	DNA/promoters	receptors/hormones
	DNA/histones	channels/ions
	DNA/transcription factors	channels/nucleotides
	DNA/repressors	channels/ligands
		proteins/chaperons
		proteins/protein kinases
		proteins/proteinphosphatases
		proteins/proteases
		proteins/proteasomes
		proteins/converter enzymes
		proteins/protein multimers
		proteins/protein complexes
		proteins/protein machines

Table 1. Basic categories of molecular recognition in the living cell.

Together these three architectural classes integrate a "universal processing and constructing system" capable of exploiting an endless variety of boundary conditions at the molecular scale. In the astonishing matching games performed by the molecular crews of the living cell, so to speak, the "Lilliputian populations" made out from amino acids are coded into a sequential genome, are altered systematically, and are evolutionarily selected. This cellular "society" becomes organized in a very sophisticate way so that both the *functions* and the circumstantial *addresses* of the multifarious molecular crews —as we will discuss— are put together into the same information bank.

Embodiment of the functional agents: the diluted architecture

Enzymes and proteins, the agential stuff of the "diluted architecture" coded onto the DNA, appear as flexi-molecular machines with a life cycle of their own [Ho, 1995]. Their constitutive structure of linked amino acids is permanently caught into a state of flow, from birth at ribosomes to final degradation at proteasomes. In actuality, it is in the enigmatic folding process taking place at chaperons (in itself a computational NP-problem) where enzymes and proteins acquire their machine-like characteristics, which enable them to perform a regular function within the cell.

Enzyme (and protein) function is but a continuation of the folding process. Apparently it implies a clear and regular succession of enzymic states: specific molecular recognition of the substrate, mutual coupling, lowering of the activation energy, interconversion between forms of energy, exit of the substrate transformed into product, and culmination of a regular work cycle [Marijuán and Westley, 1992], [Urry, 1995]. As a matter of fact, classical biochemical approaches have described this regular functioning of the enzyme through deterministic rate equations, non-linear ones that are often analyzed in a linear simplified way by means of control theory.

Nevertheless, this functioning may also be approached probabilistically. A stochastic dynamics –*molecular automata*– where enzymes "fire" their state transitions according to probabilities derived from the free energy differences in between states, can be more realistic than classical equations of control theory [Marijuán, 1994]. Moreover, such probabilistic dynamics would be closer to the stochastic nature of transitions in the "post-folding" energy landscape from which the different states of the enzyme cycle are derived [Frauenfelder, 1991], [Shimizu and Bray, 2001].

Apart from the classical discussion about determinism versus stochasticity in the enzyme's function, we have to pay attention to the role that the *embodiment* of the function plays in the way such functionality is deployed cellularly. For instance, the whole organization of degradation processes or *degradomics* (traditionally forgotten) nowadays appears almost as complex as the transcription process itself [Marijuán, 1996, 2002]. The very duration of the biomolecular agent depends on this planned process of degradation. Besides, some of the striking ecological regularities found in living organisms —so to speak, depending on their "efficient" biomass— might be related to the commonality of processes or stages in the life cycle of each molecular agent. Concretely, animals in their metabolic rates and life spans [Atasanov, 2005], and plants in their photosynthetic surfaces and life spans [Wright, 2004], deploy an amazing constancy that can be explained only by taking into account the strict coupling at the molecular level between stochastic dynamics and degradation process, for any enzyme or protein functionally active.

Primary versus secondary addresses

A parsimonious approach to the function of the biomolecular agent has to pay attention not only to the functional *"what"* dictated in the active site of the enzyme, and to its global *duration*, but also to a series of accompanying processes distributed over different parts of the molecular structure, which may include: modulation by effectors, intracellular transportation, permanent (post-translational) modification, formation of complexes, time-frames derived from transcription and translation, and as already said the final (or partial) degradation.

Thus, the *"what"* of the functional clause should be accompanied by many other circumstances such as: *how fast, where, which way, with whom, when,* and *how long*. In general, the functionalities of the active site and the retinue of accompanying processes are independently defined onto the DNA sequences, constituting *addresses* which are separately coding for function (*"primary address"* coding the active site), and also for the other operation of control, transportation, splicing, modification, complexes, transcription-translation, degradation, etc. (each one implying some specific *"secondary address"* in the DNA coding, irrespective that they may be functionally operative in the DNA, RNA, or in the protein stages).

In prokaryotes, the global arrangement of embodiment processes is simpler than in eukaryotes, in correspondence their protein components are smaller and contain fewer domains comparatively. The possibility of systematic tinkering upon multiple modules and domains becomes one of the most distinctive evolutionary strategies of eukaryotes, the tool-box of their multicellularity. A serial-combinatoric arrangement of exons and introns (which usually constitute folding domains), tailored for each tissue by differential splicing, allows eukaryotes a far bigger proteome than prokaryotes (around one or two orders of magnitude) without multiplying the number of genes involved [Claverie, 2001].

By tinkering and playing combinatory games upon exons and introns containing a vast array of secondary addresses, eukaryotic cells may systematically explore and change the whole boundary conditions surrounding the triggering of each biomolecular function —mastering all those circumstances of *when, where, how fast, which way, for how long, with whom,* etc., which together co-determine the functional action of any eukaryotic enzyme or protein [Marijuán, 2003].

The generation of variety within biological genetic algorithms is surprisingly complex in most eukaryotic genomes, potentially involving occurrences such as: SNPs, repetitive DNA, mobile elements, transposons, retrotransposons, telomere shortening, gene and segmental duplications, chromosome fissions and fusions, whole genome duplications, symbiosis, etc. The striking complexity of eukaryotic bauplans and organismic physiologies has been achieved only by the combined action of all these engines of variation impinging upon the set of different addresses involved in the embodiment of the biomolecular functional agents.

Subsequently, in the evolutionary problem-solving strategies of prokaryotic and eukaryotic cells, their very different DNA grammars would imply crucial differences. Their respective universality in evolutionary problem solving is addressed directly towards the solution of *molecular "assimilation"* phenomena in one case, while in the other case it is addressed towards harnessing *molecular "organization"* phenomena (morphology and differentiation).

Controling gene expression: the sequential architecture

The elements of the diluted architecture are all of them coded into the sequential architecture, thus the control of gene expression by the elements of the former will give an overall picture of the self-modification capabilities of the cellular system. Traditionally most studies have focused in the expression of individual genes and not in the texture of the overall network or in the internal/external instances of control concerning the guidance of gene expression. Currently, however, transcriptional regulatory networks are built for different prokaryotic microorganisms and eukaryotic specialized cell-types or cellular functions.

As an instance of such networks, the authors have compiled a large-scale *M. tuberculosis* transcriptional regulatory network, which has been built upon a previously published TR network [Balázsi, 2008] the largest to date, with further addition of different kinds of resources pertaining to publicly available sources: DNA microarrays, operons, orthology approaches, and synthetic biology experiments [Navarro & Marijuán, 2010]. See Figure 1. Our compilation forms part of ongoing studies tending to the development of a *new vaccine* based on the mutant strain SO2 [Martín, 2006]. The objective is to contribute to a better understanding of both the transcriptional control by the system and the re-organization of the cell cycle that takes place in the different environments, as well as gauging the impact of the SO2 strain on physiological and immune systems of the body.

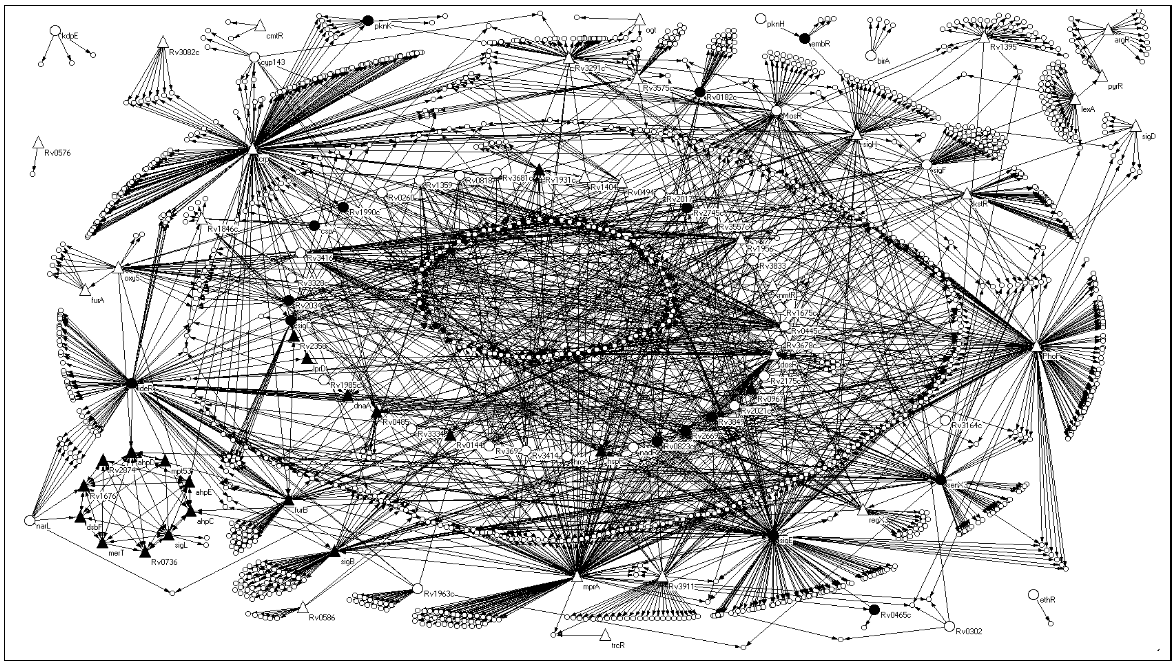


Figure 1. The Transcriptional-Regulatory (ETR) Network of *M. tuberculosis*. Nodes represent *Mt*'s genes, and links represent their regulatory interactions. Transcription factors appear either green or blue, depending on whether they have known transcriptional regulator or not. The white nodes represent output elements without transcriptional activity. The triangle nodes represent protein transcription factors that auto-regulate their own expression. Approximately 35% of the genome is covered by this network. (Modified from: Navarro, 2010).

The 1,400 network nodes represented in Figure 1 correspond all of them to specific genes of *M. tuberculosis* and their protein products, while the 2,304 links correspond to gene expression regulatory interactions by 94 transcription factors.

The network shows a clear organization in structural levels that correspond with the complex functions and life-cycle stages of this highly sophisticate pathogen. Although the functions are relatively well defined in modules or communities, they can change dramatically by simply "rewiring" some connections of the genetic network. This has already been made in other bacteria: it has been demonstrated experimentally that it is possible to make that a bacterium synthesizes (or not) a green fluorescent protein simply by exchanging the regulatory regions of genes, *lacR*, *tetR* y *lambda cI*, not changing sequence in these genes [Guet et al., 2002].

The genome of the bacillus contains more than 4,000 genes, and close to 190 transcription factors. Of this entire repertoire, the new ERT network represents 94 transcription factors and 1,400 genes. So there is plenty of room for future improvement of the network, as new laboratory works will describe new links derived from other transcription factors not worked out yet. In general, the number of transcription factors per genome translates into greater genetic network connectivity, which is correlated with increased complexity of the microorganism structures and life cycle [Levine and Tjian, 2003].

Cellular signaling systems: topological governance

By itself the transcription network is “blind”. In other words, the coupling between the sequential and the diluted architectures needs adaptive capability to respond to environmental demands. This is done by means of signaling guidance, so to partially deploy the genetic circuits in response to relevant happenstances of the environment or from within the cell. The *topological governance* of the transcription regulatory network, the decision of what parts should be activated or what particular circuits should be inhibited, is achieved thus by the cellular signaling system or *signalome.*

Prokaryotic signaling systems

In prokaryotes, a variety of molecular systems are involved in the *signalome*, ranging from simple transcription-sensory regulators (a single protein comprising two domains), such as the well-known *embR, alkA* or *furB*, to those systems of multiple components and interconnected pathways that regulate key stages of the cell cycle, such as latency, pathogenesis, replication, and dispersion. A basic taxonomy of bacterial signaling systems was proposed by the authors somewhere else [Marijuán, 2010], which was centered on “the 1-2-3 scheme” (see Figure 2):

The first level of signaling complexity corresponds to simple regulators, "the one-component systems (OCS)." In fact, most cellular proteins involved in cellular adaptation to changing environments, in a general sense, could be included as participants in this primary category [Galperin, 2005]. Around one hundred OCS elements may be present in a moderately complex prokaryotic cell.

Increasing the scale of complexity, the "two-component systems (TCS)” appear, which include a histidine kinase protein receptor and an independent regulatory response; conventionally they are considered as the central paradigm in prokaryotic signaling systems, and in fact, a number of intercellular communication processes among different species are carried out by these specialized systems. A few dozen TCS pathways may be present in the prokaryote.

To maintain conceptual coherence, an additional category, the "three-component system (ThCS)" should apply to two-component systems that incorporate additional non-kinase receptor for activating the protein kinase (eg, methylated receptors described for the chemotaxis.) Very few pathways are showing the ThCS arrangement but they are very important ones (e.g., chemotactic guidance).

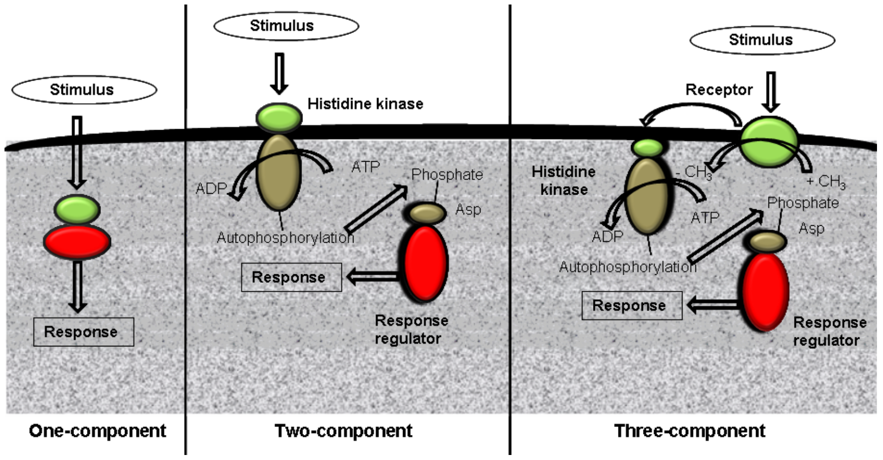


Figure 2. The three characteristic signaling pathways developed by prokaryotes. The external stimulus is perceived either by an internal receptor–transducer (left), or by a transmembrane histidine kinase that connects with a response regulator (center), or by an independent receptor associated to the histidine kinase (right). (Modified from: [Marijuán, 2010]).

Eukaryotic signaling systems

In eukaryotes the signaling system comprises many hundreds of different classes of dedicated molecular agents (receptors, ion channels, transducers, amplification cascades, second messengers, intermediate effectors, final effectors) that can be arranged differently in each tissue. See Fig. 3 for a very simplified scheme. It is very important that, in multicellular organisms, every cell-type has tailored its *specialized signalome* along its developmental trajectory, in dependence of its own history of received signals and self-modifying processes [Marijuán, 2002].

In eukaryotes, rather than a simple taxonomy like the 1-2-3 seen in prokaryotes, the scheme of signaling pathways becomes an interconnected network. Some of the main pathways are shown in a linear way in Figure 2. Here, the general "detection, measurement, and intervention" character of the signalome has to be emphasized. The second messengers (cAMP, cGMP, Ca, InsP3, diacylglicerol, ceramide...) are dramatically modified in their concentrations by the different signaling paths that have been transiently activated, within a generalized cross-talking among all activated paths. Particularly throughout the very fast changes in second messenger concentrations, an integrated perspective (measurement) of the different internal and external influences at play is obtained within the cell, and is subsequently passed towards intermediate chains and the final effectors.

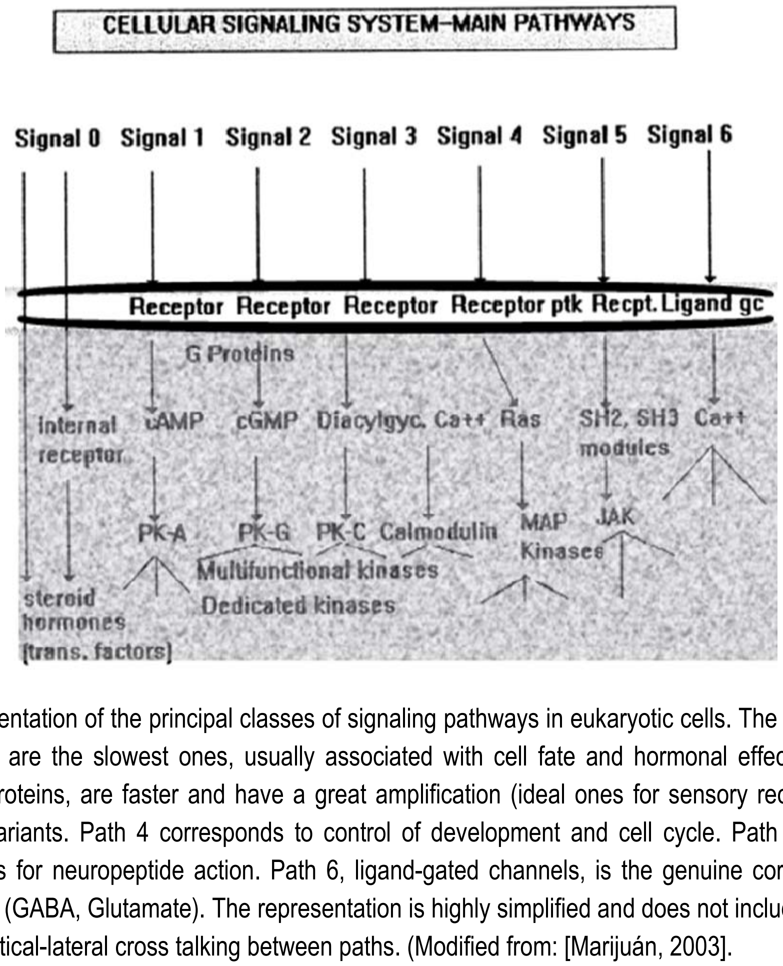


Figure 3. Representation of the principal classes of signaling pathways in eukaryotic cells. The signaling paths in the left (steroids) are the slowest ones, usually associated with cell fate and hormonal effects. Paths 1 to 3, mediated by G proteins, are faster and have a great amplification (ideal ones for sensory receptors), counting with numerous variants. Path 4 corresponds to control of development and cell cycle. Path 5 represents the customary access for neuropeptide action. Path 6, ligand-gated channels, is the genuine cortical path for fast neurotransmitters (GABA, Glutamate). The representation is highly simplified and does not include further effector cascades and vertical-lateral cross talking between paths. (Modified from: [Marijuán, 2003].

At the end of the signaling command-chain, the nuclear machinery is waiting to be fed with a combination of *ad hoc* signals in order to change the transcriptional status of the genome [Janes, 2005]. This nuclear part of the whole signalome apparatus has already been implementing the *histone code,* in order to allow a tight grip upon the euchromatin-heterochromatin states which regulate access to transcription —so that the well measured signals from the cytoplasmic signalome may be finally enacted as a new transcription program in relation with the advancement of the cell cycle or with the specialized function of the cell [Janes, 2005]. As already said, a comparison with the "direct" topological governance of the transcriptional regulatory network we have discussed in the prokaryotic case is not viable, except in the most simplified cases (e.g., steroid paths, degradome).

Everything has to converge factually on the cell cycle: signaling system, transcription, metabolism, protein synthesis, protein degradation, network organization, and control of the cell cycle itself [Marijuán, 1996]. As successive rounds of cell replication are accomplished, there occurs a functional rewriting of "DNA rules" by the signaling system. The transcriptional status of most DNA regions is systematically altered as cells advance along the totipotent, pluripotent and stem cell path, until completion of the tissular differentiation and specialization is achieved [Gasser, 2002]. The euchromatine / heterochromatine state of a number of genomic regions is irreversibly altered by signaled implementation of the previously mentioned "histone code".

From the point of view of formal systems, this unusual characteristic of "rewriting the own rules" could be significant concerning the cellular automata field [Wolfram, 2002], perhaps opening new paths towards new types of biologically inspired cellular automata capable of negotiating complex morphological / differentiation spaces.

Concluding comments: on evolutionary problem-solving strategies

With the spiraling and multiplication of cellular cycles along the eukaryotic developmental path, the hallmark for a new type of biocomplexity is set. The prokaryotic cell cycle has been loaded with a formidable complexity in eukaryotes, setting the stage for a number of *emergences* discussed in computational and philosophical fields. For instance: transition from molecular stochasticity to systemic robustness and quasi-determinism, organization of cellular signaling systems, establishment of an "informational" cell cycle (based upon molecular recognition dynamics), interplay of cellular bottom up causality with organismic top-down causality, exhibition of behavior endowed with autonomy and agency, etc.

In the extent to which this complexity growth of eukaryotes has been built by tinkering upon the scheme of *functional or primary addresses* and secondary addresses put together onto the same DNA memory, the parallel with the von Neumann scheme of modern computers seems unavoidable –for in computers, logical functions and memory addresses are also put together into the CPU memory. See some other recent interpretations in [Danchin, 2009], [Yan, 2010].

Because of this DNA scheme in eukaryotic cells, the evolutionary genetic algorithms for physiological problem-solving are largely parallelized in eukaryotes. The different components of the biomolecular solutions may be tinkered with separately, and linked together later on [Peisajovich, 2010]. Besides, every molecular stage (transcription, folding, transportation, modification, complexes, degradation), specifically coded onto DNA addresses, may be used as a new functional element of control. Solutions may be chosen, then, from an augmented set of molecular building blocks.

The so called "Central Dogma" of classical molecular biology should not be taken as a closed black-box; rather the successive stages and intermediate transcripts could participate as legitimate molecular partners, each one endowed with endogenous recognition capabilities, within a whole *transmolecular matrix* of controlling interactions [Marijuán, 2002, 2003]. As an instance, in the recently discovered phenomenon of RNA interference, scores of micro RNAs are transcribed for the only purpose of using up their molecular recognition capabilities within the context of other DNA transcription and RNA translation events, collectively known as "gene silencing."

In actuality, the evolutionary coupling between the two informational architectures of life, the sequential and the amorphous, has explored almost every conceivable cellular bauplan and organism physiology [Mojica, 2009]. Life has thrived throughout the deployment of an organization with amazing informational capabilities and systemic emergences. We might argue that prokaryotes have used those very capabilities mostly towards the direct solution of *molecular assimilation* problems (in their encounter with environmental substances), while eukaryotes have tamed a fascinating developmental complexity by evolving towards the general solution of *molecular organization* problems.

Based on molecular recognition phenomena, the dynamics of life is informational, and non-mechanical. In this regard, the following three principles may tentatively summarize various directions discussed in this paper:

1. The living cell is an open system continuously engaged in the advancement of a manifold trajectory: the life cycle of self-re-production.

2. The cellular advancement across the life cycle may be propelled (or nullified) not only by the availability of environmental affordances but also —and mostly— by signaling events.

3. The effects of signaling events in the living cell are irrespective of their material underpinning; biological information is decoupled from its mater and energy counterparts and becomes symbolic, "semiotic", though it always relates to self-production processes of the life cycle.

Bibliography

[Atanasov, 2005] Atanasov A.T. The linear alometric relationship between total metabolic energy per life span and body mass of poikilothermic animals. BioSystems, 82, 137-142 (2005).

[Carlton, 2002] Carlton M. The information paradigm. Posthumous compilation available at: http://conway.cat.org.au/~predator/paradigm.txt (2002.)

[Claverie, 2001] Claverie J.M. What If There Are Only 30,000 Human Genes? Science 291pp. 1255-1257 (2001)

[Conrad M, 1996]. Cross-scale information processing in evolution, development and intelligence. BioSystems 38 pp. 97-109. (1996)

[Danchin , 2009]. Bacteria as computers making computers. FEMS Microbiol Rev 33: 3-26 (2009).

[Frauenfelder, 2009] Sligar S.G. and Wolynes P.G. (1991) The Energy Landscapes and Motions of Proteins. Science 254 pp. 1598-1603 (2009).

[Galperin, 2005]. A census of membrane-bound and intracellular signal transduction proteins in bacteria: Bacterial IQ, extroverts and introverts. BMC Microbiol. 5: 1–19 (2005).

[Gasser, 2002]. Visualizing Chromatin Dynamics in Interphase Nuclei. Science 296, 1412-1416 (2002).

[Goodsell, 1991]. Inside a living cell. TIBS 16, 203-206 (1991).

[Guet, 2002] Elowitz M.B., Hsing W. & Leibler S. Combinatorial Synthesis of Genetic Networks. Science 296 (5572): 1466-70 (2002).

[Ho, 1995]. Bioenergetics. The Open University, London, (1995).

[Janes, 2005] Albeck J.G., Gaudet S., Sorger P.K., Lauffenburger D.A. and Yaffe M.B. A Systems Model of Signaling Identifies a Molecular Basis set for cytokine-Induced Apoptosis. Science 310, pp. 1646-53 (2005).

[Levine & Tjian , 2003]. Transcription regulation and animal diversity. Nature 424: 147-51 (2003).

[Lin, 2001].The Nature of the Chemical Process. 1. Symmetry Evolution –Revised Information Theory, Similarity Principle and Ugly Symmetry. Int. J. Mol. Sci. 2 pp. 10-39 (2001).

[Marijuán, 1994]. Enzymes, automata and artificial cells. In Computing with biological metaphors, ed. by R.C. Paton. Chapman & Hall, London, pp. 50-68. (1994)

[Marijuán, 1996]. The Cell as a Problem-solving 'Engine'. In Computation in Cellular and Molecular Biological Systems, ed. by R. Cuthberson, M. Holcombe and R. Paton. World Scientific, Singapore, pp. 183-194 (1996).

[Marijuán, 2002]. Bioinformation: untangling the networks of life. BioSystems 64, pp. 111-118 (2002)

[Marijuán, 2003]. From inanimate molecules to living cells: the informational scaffolding of life. In Energy and Information Transfer in Biological Systems, ed. by F. Musumeci, L.S. Brizhik and M.W. Ho. World Scientific, Singapore (2003).

[Marijuán, 2009]. The advancement of information science: is a new way of thinking necessary? tripleC 7(2): 369-75 (2009).

[Marijuán, 2010]. Navarro J. & del Moral R. (2010). On prokaryotic intelligence: strategies for sensing the environment. Biosystems 99: 94-103 (2010).

[Marijuán and Westley, 1992]. Enzymes as molecular automata: a reflection on some numerical and philosophical aspects of the hypothesis. BioSystems 27: 97-113 (1992)

[Martin, 2006]. Tuberculosis vaccines: past, present and future. Curr Opin Pulm Med. 12 (3): 186-9 (2006).

[Meggs, 1998] Biological homing: hypothesis for a quantum effect that leads to the existence of life. Medical Hypothesis 51, pp. 503-506 (1998).

[Mojica, 2009] Navarro J., Marijuán P.C. & Rafael Lahoz-Beltra. (2009). Cellular "bauplans": Evolving unicellular forms by means of Julia sets and Pickover biomorphs. BioSystems 98, 19–30 (2009).

[Navarro, 2010]. Transcriptional Regulatory Network of M. tuberculosis: Functional and Signaling Aspects. Master Thesis. Universidad de Zaragoza (2010).

[Peisajovich, 2010] Garbarino J.E., Wei P. and Lim W.A. Science 328: 368-72 (2010).

[Shimizu and Bray, 2001]. Computational Cell biology –The Stochastic Approach. In Foundations of Systems Biology, ed. by H. Kitano. The MIT Press, Cambridge (2001).

[Urry, 1995]. Elastic Biomolecular Machines. Scientific American January 44-49 (1995).

[Weinhold and Landis, 2007]. High Bond Orders in Metal-Metal Bonding. Science, 316,.61-3 (2007).

[Wolfram, 2002]. A New Kind of Science. Wolfram Media Inc. (2002).

[Wright, 2004] Reich P.B. et al. --33 authors total-- (2004) The worldwide leaf economics spectrum. Nature 428, 821-7 (2004).

[Yan, 2010] Fang G., Bhardwaj N., Alexander R.P. & Gerstein M. Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks. PNAS 107 (20): 9186-91 (2010).

Authors' Information

Jorge Navarro López– *Bioinformation and Systems Biology Group, Aragon Institute of Health Sciences (I+CS,) Zaragoza, Spain.*

e-mail: jnavarro.iacs@aragon.es

Angel Goñi-Moreno– *Natural Computing Group, Universidad Politécnica de Madrid, Spain; e-mail: anandgel@gmail.com*

Pedro C. Marijuán – *Bioinformation and Systems Biology Group, Aragon Institute of Health Sciences (I+CS,) Zaragoza, Spain.*

e-mail: pcmarijuan.iacs@aragon.es

SELF-ASSEMBLY PROCESS FOR INTEGRATED CIRCUITS BASED ON CARBON NANOTUBES USING MICROFLUIDIC DEVICES

David Moreno, Sandra Gómez, Paula Cordero

***Abstract**: New methods are needed to create integrated circuits which are able to overcome the inherent problems in the miniaturization process. These problems are mainly technological and economical; photolithography is limited and the expensive building respectively. This paper proposes the basis for a new manufacturing process of nanotechnological circuits based on semiconducting carbon nanotubes that work as FET (Field Effect Transistor) and metallic carbon nanotubes that work as nanowires. This process is based on the assembly of DNA tiles and lattices that guide the placement of carbon nanotubes to build electronic circuits. The process takes place in a microfluidic device within its chambers. Building blocks are created based on NAND logic gates. These building blocks are enabled to assemble AND, OR and NOT logic gates. The process of assembling a XOR logic gate is explained, demonstrating how to apply the process in a concrete case.*

***Keywords**: Carbon nanotubes, DNA lattice, FET, Microfluidic devices, Self-assembly process.*

***ACM Classification Keywords:** B.7.1 Types and Design Styles – Advanced technologies*

Introduction

With the technologies used today, Moore's law is going to touch the ceiling and will not be possible to further reduce the size of transistors and to double the number of them per unit area. This affirmation conduces at the search of new materials and paradigms for the construction of computing circuits Current methods of manufacturing integrated circuits have absolute control of the assembly of components but there are economic and technological limitations (factories are too expensive and photolithography has precision limitations, respectively).

Nanotechnology allows working on the atomic scale and the main advantage about this is the reduction of the circuit size. But nanotechnology has to deal with two issues for building reliable circuits: manufacturing defects and faults. Manufacturing defects are due to the imperfect fabrication process inherent to nanoscale properties. A fault is an incorrect state of the system due to manufacturing defects, component failures, environmental conditions etc. Faults can be permanent, due to system defects; intermittent, faults appear periodically; transient, due to temporary environmental conditions. A system with defect tolerance is able to operate correctly in the presence of manufacturing defects. A system with fault tolerance is able to operate correctly in the presence of permanent, intermittent or transient faults. The most used method to achieve defect tolerance and fault tolerance is components redundancy. Moreover, defect tolerance and fault tolerance can be achieved at several levels: physical level, architectural level and application level [Graham et al., 2004].

This paper presents a proposal of a new approach for nanotechnological circuit fabrication in a controlled manner without going into depth on technological details on nanotechnology and microfluidic devices. The proposed approach uses a microfluidic device to build an electronic circuit based on carbon nanotubes that are self-assembled through DNA tiles. The idea is based on the works about DNA lattices assembled within microfluidic devices and works about how to use DNA tiles and lattices to assemble carbon nanotubes to build circuits. In a first state of the process, building blocks are assembled based on NAND gates within the microfluidic device chambers. This NAND logic gates are supposed to be the same as the developed ones in other works. This is the base to build any circuit possible, memories, ALU etc.

This paper is structured in the next manner: section 2 provides background about using DNA lattices for manipulating and self-assembling carbon nanotubes and using microfluidic circuits for DNA lattices assembly; section 3 sets out the components and operation of the proposed micro fluidic device; section 4 to section 6 explains in detail the self-assembly process; section 7 shows an example of the process based on XOR logic gate; section 8 explains some generalization about the process and the last section shows conclusions and future work.

Background

L. Adleman was the first researcher in to demonstrate the computing capacities of DNA based on strands hybridization. In [Adleman, 1994] is showed the resolution of a simple Hamiltonian path problem, a NP-Complete problem, with DNA strands. After Adleman's works, Lipton was solved a simple SAT problem, a NP-Complete problem using DNA as [Lipton, 1998] and in [Winfree et al., 1996] was shown the universal computing capacity of self-assembled DNA strands.

This complementarity of bases also allows creating tiles of DNA strands to be assembled in a lattice, such as scaffolded DNA Origami [Rothemund, 2006]. In [Dwyer et al., 2004b] is showed how to create self-assembling computer circuits with DNA facilitating the use of other components for the computing itself, for it makes use of a decoupled array multi-processor (DAMP), similar to a single-design instruction, multiple-data (SIMD).

In these papers [Keren et al., 2003], [Dwyer, 2002], [Dwyer et al., 2004a], [Maune et al., 2009], [Patwardhan et al., 2004] is showed how to use DNA lattices in combination with carbon nanotubes as field effect transistors (FET) to build computer circuits. Carbon nanotubes can be used as FET due to the ability of these semiconducting nanotubes to be modulated applying a gate voltage and the ability to separate metallic nanotubes from semiconducting nanotubes and controlling the length of individual nanotubes.

Specifically, in [Maune et al., 2009] is observed the assembly process for a carbon nanotube into a DNA lattice, which consists of:

1. Designing a DNA strand with one end, the 5 'end, with forty T bases, which will assemble the carbon nanotube, and 20 bases with a certain combination, of which the 15 bases that follow the 40 T bases will be hybridized with their complementary base, the toe, freeing five bases at the other end, the 3 '.
2. The lattice has chains of DNA that are complementary to the 20 bases completely different from the 40 Ts. By having five free bases, the attraction of the lattice, fully complementary, is more favorable thermodynamically than the 15 bases toe, so that in the end, the chain that carries the carbon nanotube hybridizes with the lattice.

In [Dwyer et al., 2004a] and [Patwardhan et al., 2004] are proposed the build of a full adder and a SR-latch through DNA lattices and carbon nanotubes NAND gates They used two types of carbon nanotubes, attached in a perpendicular way: the metallic carbon nanotube acts as a gate, and the semiconducting carbon nanotube acts as channel in a CFNET. The nanotubes used as wires are attached to each side of each cavity in the tile. This proposed process has limitations in the number of FETs (100x100), in the size and complexity of the circuit. For this reason, these authors warn the necessity of a modular design to allow building larger circuits.

In this sense, the process of creating DNA lattices can be improved via microfluidic circuits. A microfluidic circuit is a configuration of microscale fluidic components such as microchannels, individually addressable valves and chambers through which fluid is allowed to flow [Dutta, 2008]. Microfluidic circuits can be represented as electrical circuits and have components as fluidic resistors, capacitors, inductors and transistors. Therefore, microfluidic circuits can be characterized by fluidic resistance, fluidic inductance, fluidic capacitance, fluidic transistors, fluidic amplifiers and fluidic logic. There are two main types of microfluidic circuits: fluidic circuits for pressure-driven flow and fluidic circuits for electrokinetically driven flow. Fluidic circuits for pressure-driven flow are based on fluid resistance. In fluidic circuits for electrokinetically driven flow, an ionized liquid can be forced to move in a dielectric capillary or microchannel under the action of an externally applied electric field [Dutta, 2008]. An interesting advantage is that flow rate can be controlled very precisely using an externally applied electric field.

In [Somei et al., 2006] are showed the difficulties in assembly DNA lattices due to aspects as the hardness of maintaining the concentrations of monomers or the 1% in errors of assembling . The solution proposed is the use of a microfluidic device for the assembling process of the DNA lattice. This architecture consists of:

a) A service port where the DNA sample solution is applied by pipetting.

b) A reaction chamber where the self-assembly DNA tile participates under controlled temperature.

c) A capillary pump which generates suction force to pull the next solution from the service port.

The different tiles are assembled in the chambers. The chambers are connected in a way that tiles are assembled in the desired manner in other chambers. The process is totally controlled due to the microfluidic device.

Self-assembly process using microfluidic devices: description

The start point after Somei's work would be to use a similar microfluidic circuit [Somei et al., 2006]. This proposed circuit self assembles the tiles with carbon nanotubes, within the chambers of the device. Each chamber can to assemble different logic gates and wires. The microfluidic device used is a fluidic circuit for pressure-driven flow.

In order to manufacture specific circuits, able to self-assemble, is necessary assembled different logic gates as well as create different tiles for the same logic gates, varying the type of input and output for this gates. T Based on NAND gate was proposed in [Dwyer et al., 2004] and using the logic gate NAND, our microfluidic circuit will have three basic phases in the self-assembly of logic gates:

1. NAND gates are assembled in order to build the rest of the required logic gates.
2. The logic gates based on the NAND gates are built.
3. Adapters are built to connect the assembled gates. This last phase depends on the circuit to be build.

The process depicted is incomplete because the assembled circuit is not ready to be used as an electronic device. The assembled circuit needs a procedure like the described in [Keren et al., 2003] for CFNETs.

It is assumed that the component to be built has several input signals. These signals are shared by every logic gate within the component and one or more output signals. It is considered a component: anything is assembled within a chamber.

It is necessary for the first phase, to define several and different DNA strands, which are markers of input and output signals. These markers point out the gates which are their inputs and outputs and they guide the assembly process within the chambers. The used strands are defined as follows:

- I_i is a strand that indicates the signal i is an input of the component corresponding to the logic gate.

- O_{ij} is a strand that indicates the output of the logic gate with input signal *i* and input signal *j*.

- $I_{O_{ij}}$ is a strand that indicates the input of the logic gate is the output O_{ij} of another logic gate.

The chambers where the components are assembled are denominated $C_{component}$, where *component* is the name of the component assembled within the chamber.

The adapters, components are denominated $A_{input/output_i}$, where $input$ is the name of the component whose output is the input of the adapter and $output_i$ is the name of the component whose input i is the output of the adapter.

The process to build the logic gates OR, AND and NOT through a microfluidic device from NAND gate are presented. In addition, an example of a new assembled logic gate based on the created logic gates, an XOR, is illustrated. In this process, only two input signals to the components are considered.

Phase 1: building blocks for the assembly of OR, AND and NOT logic gates.

In this first phase, in the microfluidic circuit is necessary to have the same number of chambers and components. The subject of this phase is to define an inverter that is showing in the next figure. Two types of inverters are needed: NOT_{11} and NOT_{22}. NOT_{11} has input signal 1 as input and NOT_{22} has input signal 2 as input. Both components are showed in the figure 1.

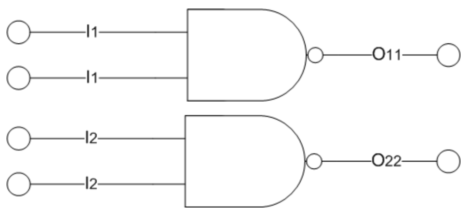


Figure 1. NOT logic gates

The building blocks for assembling OR logic gate are the components NOT_{11} and NOT_{22} and a new gate, a NAND gate, designated by $NAND_{11\_22}$ which has as inputs the outputs of NOT_{11} and NOT_{22}. This new NAND gate is showed in the figure below.

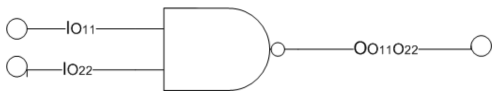


Figure 2. OR gate components

The building blocks for assembling AND logic gate are two NAND gates: $NAND_{12}$ and $NAND_{12\_12}$. The gate $NAND_{12}$ has as input signals the signal 1 and signal 2 of the circuit. $NAND_{12\_12}$. has as inputs the outputs of two $NAND_{12}$. The AND component is showed in figure 3.

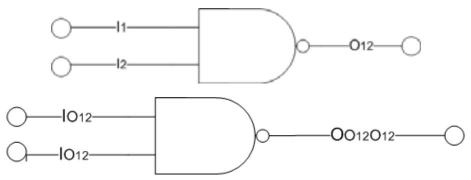


Figure 3. AND gate components

Once it is well known how to build the components, 5 chambers are necessary to assemble the 5 building blocks: $C_{NOT_{11}}$, $C_{NOT_{22}}$, $C_{NAND_{11\_22}}$, $C_{NAND_{12}}$, $C_{NAND_{11\_22}}$ and 7 different DNA strands are necessary to associate to each input and output used:

- I_1 , I_2 , O_{11} , O_{22} , O_{12} , $O_{O_{11}O_{22}}$ and $O_{O_{12}O_{12}}$ have their own DNA strand for the subsequent assembly with the other components.
  - $I_{o_{11}}$ has associated the complementary strand of O_{11} .
  - $I_{o_{22}}$ has associated the complementary strand of O_{22} .
  - $I_{o_{12}}$ has associated the complementary strand of O_{12} .

The DNA strands and the carbon nanotubes solution is applied by pipetting the corresponding service ports. The assembly process is realized within the chambers.

Phase 2: Assembling OR and AND gates.

In this phase, a new solution is applied by pipetting the corresponding service ports. This causes the new assembled components flowing from their chambers to the next chambers. Then, a new assembly process is generated. This process is responsible of the building the OR and AND gates.

To assemble the OR gates, is necessary a chamber to combine the products of $C_{NOT_{11}}$, $C_{NOT_{22}}$, $C_{NAND_{11\_22}}$ chambers. OR gate is shown in figure 4.

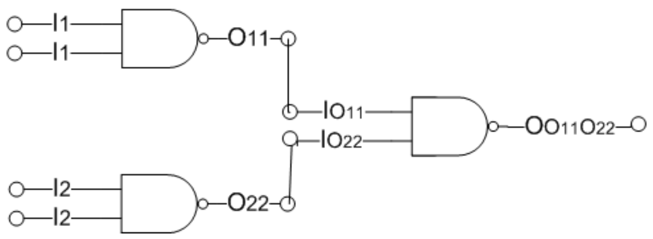


Figure 4. OR logic gate

To assemble the AND gates, is necessary a chamber to combine the products of $C_{NOT_{12}}$, $C_{NAND_{12\_12}}$ chambers. To obtain a number of AND gates, are necessary the same number of $NAND_{12\_12}$ components and the double of $NAND_{12}$ components. AND gate is shown in figure 5.

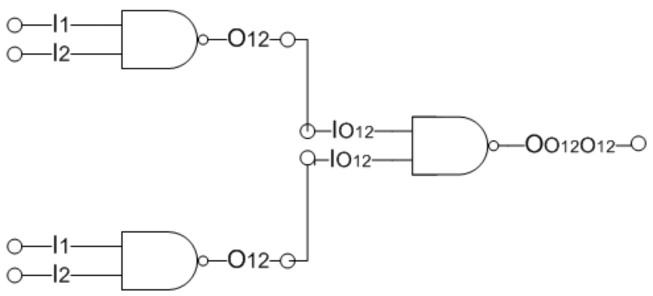


Figure 5. AND logic gate

Once it is well known how to build the components, 2 chambers are needed to assemble the 2 logic gates: C_{OR} , C_{AND} . At this moment of the assembly procedure, 7 chambers are necessary.

Phase 3: Creating adapters between logic gates

At this point, it is necessary a new type of component: the adapter. The adapter connects the outputs of the logic gates with the inputs of another logic gates. This component, which is a tile, is composed by the complementary strands of the outputs of the logic gates and the complementary strands of the inputs of the gates to be connected. This component is an interface between logic gates. It could be a signal amplifier, if carbon nanotubes as FETs are assembled within the tile. The adapters could be used to switch on or switch off parts of the circuit.

An example: XOR logic gate

Based on the components resulting from phases 1 and 2, and the adapters were described in phase 3, we build a XOR logic gate. The input signals of the component are the input signal 1 and the input signal 2.

To build a XOR gate we need two NOT gates, two AND gates, one OR gate and the following adapters:

- An adapter whose input is the complementary DNA strand of the output DNA strand of a component NOT_{11} and whose output is the complementary DNA strand of the first input of an AND gate. It is denominated A_{NOT_{11}/AND_1} .

- An adapter whose input is the complementary DNA strand of the output DNA strand of a component NOT_{22} and whose output is the complementary DNA strand of the second input of an AND gate. It is denominated A_{NOT_{22}/AND_2} .

- An adapter whose input is the complementary DNA strand of the output DNA strand of an AND gate and whose output is the complementary DNA strand of the first input of an OR gate. It is denominated A_{AND/OR_1} .

- An adapter whose input is the complementary DNA strand of the output DNA strand of an AND gate and whose output is the complementary DNA strand of the second input of an OR gate. It is denominated A_{AND/OR_2} .

For each adapter, two chambers are necessary. The first chamber is necessary to assemble the adapter with the gate whose output is the input of the adapter. The other chamber assembles the adapter with the gate whose input is the output of the adapter.. This is the process needed to avoid undesirable hybridization between logic gates whose input is the output of a logic gate of the same type. This example relies in 8 new chambers and the total of the chambers is 15.

The assembly process for the XOR logic gate is composed by the next phases:

1. Assembly of adapters of the AND gates with their inputs:

   a) Assembling adapters A_{NOT_{11}/AND_1} with NOT_{11} components, which are products of $C_{NOT_{11}}$ chamber, within C_{A_1} chamber.

   b) Assembling adapters A_{NOT_{22}/AND_2} with NOT_{22} components, which are products of $C_{NOT_{22}}$ chamber, within C_{A_2} chamber.

2. Assembly of adapters of the AND gates with their outputs:

a) Assembling adapters A_{NOT_{11}/AND_1}, which are products of C_{A_1} chamber, with input 1 of AND gates which are products of C_{AND} chamber, within C_{AND_1} chamber.

b) Assembling adapters A_{NOT_{22}/AND_2}, which are products of C_{A_2} chamber, with input 2 of AND gates, which are product s of C_{AND} chamber, within C_{AND_2} chamber.

This phase needs a dispatcher after the chamber C_{AND} having the same probability of dispatching AND gates to C_{AND_1} and C_{AND_2} chambers.

3. Assembly of adapters of the OR gates with their inputs:

a) Assembling adapters A_{AND/OR_1} with AND gates, which are products of C_{AND_1} chamber, within C_{A_3} chamber.

b) Assembling adapters A_{AND/OR_2} with AND gates, which are products of C_{AND_2} chamber, within C_{A_4} chamber.

4. Assembly of adapters A_{AND/OR_1}, which are products of C_{A_3} chamber, with input 1 of OR gates, which are products of C_{OR} chamber, within C_{OR_1} chamber.

5. Assembly of adapters A_{AND/OR_2}, which are products of C_{A_4} chamber, with input 2 of OR gates, which are products of C_{OR_1} chamber, within C_{OR_2} chamber.

The desired components, XOR logic gate, are within C_{OR_2} chamber.

Generalizations of the self-assembly process

The process proposed in this paper has only two input signals. The number of DNA strands and chambers for the process depicted are for two input signals. This process can be generalized for more input signals. Different DNA strands are used for each input and output signals of the building blocks. The next formula shows the upper bound of the number of different chains that are necessary depending on the number of signals, denominated as N.

$$2N + 3\binom{N}{2} \simeq O(N^2)$$

Figure 6. Upper bound of DNA strands

Where:

$2N$ is the number of input signals plus number of outputs of the inverters for each signal;

$\binom{N}{2}$ is the number of all the possible outputs of the OR building blocks;

$2\binom{N}{2}$ is the number of outputs of the AND building blocks.

The upper bound of the number of chambers needed for phases 1 and 2 depends on the number of input signals. The formula is illustrated below (N as the number of signals):

$$N + 3\binom{N}{2} \simeq O(N^2)$$

Figure 7. Upper bound of chambers

Where:

N is the number of inverters for each signal;

$\binom{N}{2}$ is the number of all the possible OR building blocks;

$2\binom{N}{2}$ is the number of all the possible AND building blocks.

Both, the upper bound of the number of DNA strands and the upper bound of the number of chambers have a quadratic growth respect to the number of input signals.

Conclusions and Future work

In this paper it has been shown how to build circuits with carbon nanotubes as field effect transistors (FET) in a self-assembly way using DNA strands. First, the tiles are assembled and later the lattices are compound. In this process is proposed the use of microfluidic circuits for the assembly procedure. The assembly procedure starts with the creation of building blocks based on NAND logic gates. These building blocks are enable to assemble AND, OR and NOT gates. In addition, an example of building a simple component with a XOR gate was illustrated.

In particular, for the hybridization of inputs and outputs are only needed seven different chains and their complementary DNA strands and only eight chambers to create gates AND, OR and NOT. The upper bound of the number of different DNA strands and the upper bound of the number of chambers for phases 1 and 2 of the assembly process have a quadratic growth respect to the number of input signals.

The amount of carbon nanotubes and DNA strands needed to assemble certain number of components is unknown because the results of the assembly process are probabilistic. This means that the exact number of components cannot be guaranteed but a lower bound for this amount can be obtained. This lower bound is the number of assembled components desired. Therefore, this implies to know the performance of reactions within the chambers. In this sense, simulations are needed for testing the performance of the microfluidic device proposed and the circuit manufactured within the microfluidic device. As well, is necessary the simulation of the components and circuits assembled in the process. The results can be analyzed to set the impact of the insertion of the adapters into the performance of the circuit.

Now, the limits of size and complexity of the assembled circuits can be related to the limitations in design and manufacturing of the microfluidic device.

Finally, it is possible to add more functionality to the adapters. They could be used as part of a defect and fault tolerance system, based on Multiplexing NAND for the assembled circuit.

Bibliography

[Adleman, 1994] L. Adleman. Molecular Computation of Solutions to Combinatorial Problems. In: Science 266, 1021–1024. 1994.

[Dutta et al., 2008] P. Dutta, K. Horiuchi, T. Z. Jubery. Microfluidic Circuits. In: Encyclopedia of Microfluidics and Nanofluidics. 1151. Ed. Springer. 2008

[Dwyer et al., 2002] C. Dwyer, M. Guthold, M. Falvo, S. Washburn, R. Superfine, D. Erie. DNA-functionalized single-walled carbon nanotubes. In: Nanotechnology 13, 601–604. 2002.

[Dwyer et al., 2004a] C. Dwyer, V. Johri, M. Cheung, J. Patwardhan, A. Lebeck, D. Sorin. Design tools for a DNA-guided self-assembling carbon nanotube technology. In: Nanotechnology 15, 1240–1245. 2004

[Dwyer et al., 2004b] C. Dwyer, J. Poulton, R. Taylor, L. Vicci. DNA self-assembled parallel computer architectures. In: Nanotechnology 15, 1688–1694. 2004.

[Graham et al., 2004] P. Graham, M. Gokhale. Nanocomputing in the presence of defects and faults: a survey. In: Nano, Quantum and Molecular Computing, 39-72. 2004

[Keren et al., 2003] K. Keren, R. S. Berman, E. Buchstab, U. Sivan, E. Braun. DNA-Templated Carbon Nanotube Field-Effect Transistor. In: Science 302, 1380-1382. 2003

[Lipton, 1998] R. Lipton. DNA Solution of Hard Computational Problems. In: Science 268, 542-545. 1998.

[Maune et al., 2009] H. T. Maune, S. Han, R. D. Barish, M. Bockrath,W. A. Goddard III, P. W. K. Rothemund, E. Winfree. Self-assembly of carbon nanotubes into two-dimensional geometries using DNA origami templates. In: Nature Nanotechnology 311. 2009

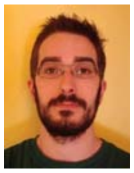
[Patwardhan et al., 2004] Jaidev P. Patwardhan, Chris Dwyer, Alvin R. Lebeck, Daniel J. Sorin, Circuit and System Architecture for DNA-Guided Self-Assembly of Nanoelectronics, Proceedings of Foundations of Nanoscience, ©ScienceTechnica, 2004

[Rothemund, 2006] P. W. K. Rothemund. Scaffolded DNA origami: from generalized multi-crossovers to polygonal networks. In: Nanotechnology: Science and Computation. Ed. Springer, 2006.

[Somei et al., 2006] K. Somei, S. Kaneda, T. Fujii, S. Murata. A Microfluidic Device for DNA Tile Self-assembly. In: DNA Computing. 325–335. Ed. Springer. 2006.

[Winfree et al., 1996] E. Winfree, X. Yang, and N. C. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In Proceedings of the 3rd International Meeting on DNA Based Computers, June 10--12 1996

Authors' Information



David Moreno *– Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain; e-mail: d.mnavas@alumnos.upm.es*

Sandra Gómez *– Natural Computing Group. Departamento de Lenguajes, Proyectos y Sistemas Informáticos, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Carretera de Valencia Km 7., Madrid, Spain; email: sgomez@eui.upm.es*

Paula Cordero *– Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain; e-mail: p.cordero@alumnos.upm.es*

LARGE VLSI ARRAYS – POWER AND ARCHITECTURAL PERSPECTIVES

Adam Teman, Orly Yadid-Pecht and Alexander Fish

Abstract*: A novel approach to power reduction in VLSI arrays is proposed. This approach includes recognition of the similarities in architectures and power profiles of different types of arrays, adaptation of methods developed for one on others and component sharing when several arrays are embedded in the same system and mutually operated. Two types of arrays are discussed: Image Sensor pixel arrays and SRAM bitcell arrays. For both types of arrays, architectures and major sources of power consumption are presented and several examples of power reduction techniques are discussed. Similarities between the architectures and power components of the two types of arrays are displayed. A number of peripheral sharing techniques for systems employing both Image Sensors and SRAM arrays are proposed and discussed. Finally, a practical example of a smart image sensor with an embedded memory is given, using an Adaptive Bulk Biasing Control scheme. The peripheral sharing and power saving techniques used in this system are discussed. This example was implemented in a standard 90nm CMOS process and showed a 26% leakage reduction as compared to standard systems.*

Keywords*: VLSI Arrays, SRAM, Smart Image Sensors, Low Power, AB*^{2}*C.*

ACM Classification Keywords*: B.3.1 Semiconductor Memories - SRAM, B.6 Logic Design – Memory Control and Access, B.7 Integrated Circuits – VLSI, E.1 Data Structures – Arrays, I.4.1 Digitization and Image Capture*

Introduction

The continuing persistence of Moore’s Law [Moore65] throughout recent years has led to great opportunities for embedding complex systems and extended functionality on a single die. The primary example of this trend is the modern day, high performance, multi-core microprocessor that employs large memory caches in order to achieve large bandwidth. Another popular example is the smart image sensor, which integrates additional capabilities of analog and digital signal processing into a conventional CMOS sensor array. Both microprocessors and image sensors are frequent components of various Systems-On-Chip (SOC) that also embed several additional SRAM arrays for various functionality. As a result of these trends, large VLSI arrays frequently cover a large area of various microelectronic systems, sometime well over half of the total silicon die.

One of the side effects of the integration of large VLSI arrays is, of course, power consumption. In the last decade, low-power design has ousted high-performance as the main focus of the VLSI industry. This is a result of the constant exponential rise in power density over the past three decades, coupled with the rise in popularity of mobile, battery powered devices. This power increase proved to be unacceptable in immobile, high performance systems, when the cost and complexity of heat dissipation became too high, and in mobile devices, where increased performance and functionality are required alongside the need for large spans between recharging. In today’s systems, it is very common that the main source of power consumption is the large memory arrays. In digital camera systems, the pixel array along with its periphery are obviously the main consumers of power, and likewise, in other SOCs comprising smart imagers, they tend to be close to the top of the list. These facts lead us to realization that low power solutions for embedded arrays are a necessity in modern VLSI design.

In this paper, we have chosen the two types of arrays mentioned above, embedded SRAM bitcell arrays and image sensor pixel arrays, for discussion. Through these examples, we will show that there are several similarities in the architectures and power profiles of different types of arrays. Many techniques and solutions have been developed for power reduction in each type of array, but rarely has one technique been adapted to fit another type of array. Through our discussion, we will show that such possibilities exist and provide an important direction for low power research.

The discussion will start with a review of the architectures of both types of subsystems (i.e. bitcell and pixel arrays), describing the components that compose each. We will then discuss the sources of power consumption and the related problems for each subsystem, as well as a number of existing low power solutions for each case. We will continue with a comparison of the two types of subsystems, highlighting similarities and discussing peripheral sharing opportunities. Finally, we will give an example of a system, recently developed by our group, that utilizes these similarities to achieve power reduction in a smart image sensor system with embedded memory.

SRAM Architecture and Power Considerations

Modern digital systems require the capability of storing and accessing large amounts of information at high speeds. Of the different types of memories, the Static Random Access Memory (SRAM) is the most common embedded memory, due to its high speeds and relatively high density in standard fabrication processes. SRAMs are widely used in microprocessors as caches, tag arrays, register files, branch table predictors, instruction windows, etc. and occupy a significant portion of the die area. In high-performance processors, L1 and L2 caches alone occupy over half of the die area [Mamidipaka, 2004]. Accordingly, SRAMs are one of the main sources of power dissipation in modern VLSI chips, especially high-end microprocessors and SOCs.

Figure 1 shows a typical block diagram of an SRAM, with emphasis on the main components and sub-blocks. The core of the SRAM is an array of identical bitcells, laid out in a very regular and repetitive structure, each bitcell storing either a ‘1’ or ‘0’ on a cross-coupled latch, and enabling read and write access. The bitcells are divided into rows and columns, allowing complete random access, through the use of X and Y addressing circuitry consisting of a row decoder and a column multiplexer. The addressing is propagated to the individual bitcell through a grid of horizontally wired *wordlines* and vertically wired *bitlines*. A particular bitcell is accessed (either read or written) when its row’s wordline and its column’s bitline are asserted simultaneously.

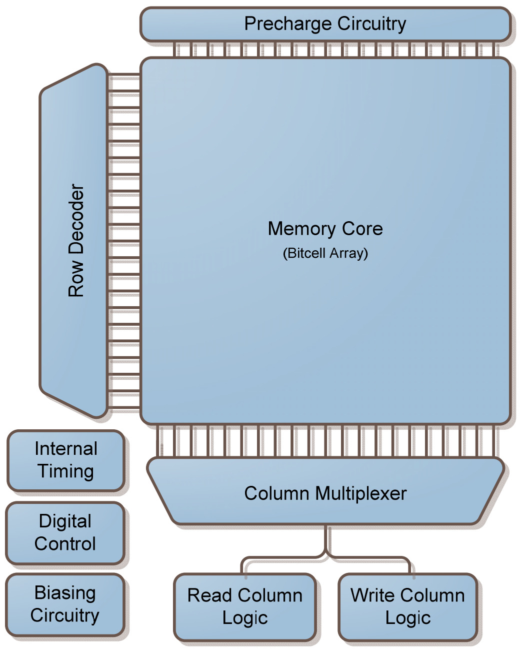


Figure 1: Typical SRAM Component Block Diagram

In order to initiate either a read or a write, the read column logic and write column logic blocks are required. The read column logic block typically consists of a low swing sense amplifier to enhance the performance and readout a digital signal from the asserted column. The write column logic block consists of a write driver that asserts the data to be written onto the relevant column. A read/write enable control signal selects which of the two blocks is activated, and the asserted wordline initiates the bitcell on the selected column to be read from or written to.

Additional blocks needed for SRAM operation include column precharge, internal timing, digital control blocks and biasing circuitry. The column precharge block prepares the read/write operation by setting the columns into a known state. The internal timing blocks sense various transitions in internal and external signals to initiate or terminate operation phases. The digital control blocks enable application of advanced error correcting, row/column redundancy, etc. The biasing circuitry is generally required for sense amplifier operation.

The power profile of SRAMs include both dynamic power, consumed during read and write operations, as well as static power, consumed during standby ("hold") periods. The dynamic power, similar to standard logic, is a function of the supply voltage and frequency, giving the standard tradeoff between power and performance/reliability. Static power in SRAMs, on the other hand, is mainly due to unwanted parasitic leakage currents. As technology scales, leakage currents become a more dominant factor, causing the static power of SRAMs to become a major issue and one of the primary static power components of many systems. A unified active power equation is given in Equation 1 [Rabaey2003] [Itoh2001]:

$$\begin{aligned} P &= V_{DD}\left(I_{array} + I_{decode} + I_{periphery}\right) = \\ &= V_{DD}\left\{\left[mi_{act} + m(n-1)i_{hld}\right] + \left[(n+m)C_{DE}V_{\text{int}}f\right] + \left[C_{PT}V_{\text{int}}f + I_{DCP}\right]\right\} \end{aligned} \tag{1}$$

where m and n are the number of columns and rows, respectively, f is the operating frequency, V_{DD} is the general supply voltage, V_{int} is the internal supply voltage, i_{act} is the effective current of the selected cells, i_{hld} is the data retention current of inactive cells, C_{DE} is the output node capacitance of each decoder, C_{PT} is the total capacitance of the digital logic and periphery circuits, and I_{DCP} is the static current of the periphery.

The dynamic power of an SRAM is mainly consumed in the following areas: address decoding, bitline charging/discharging, and readout sensing. During address decoding, power is consumed both by the switching of the decoders themselves, as well as by charging and discharging the selected wordlines, which can have high capacitances. During both read and write operations, the bitlines are precharged and subsequently discharged. This is especially power consuming during writes, when the bitline is fully discharged, or when a full discharge read scheme is chosen. Sense amplifiers typically depend on bias currents for operation, consuming constant power when they are activated.

The static power of an SRAM is primarily consumed through leakage currents inside the bitcells themselves during standby (hold) periods, i.e. when the particular cell (or the whole array) is not asserted. This includes subthreshold and gate leakages in both the inner cross coupled latch structure, as well as to/from the bitlines through the access transistors on unselected rows. Another large contributor to static power is from the precharge circuitry, when a constant charging scheme is used, i.e. a high-resistance supply or diode-connected transistor is placed on the bitlines to replenish lost precharge voltage. Other contributors to the static power are the leakage currents in the decoders and other blocks.

An in-depth analysis of the power dissipation by all SRAM components can be found in [Itoh2001].

Several standard methods have been developed over the years to reduce the power consumption of SRAMs. The standard methods are based on physical partitioning of the array in each of the axes. Banked organization of SRAMs divides the array both horizontally and vertically into sub-arrays. An external decoder raises the chip select of the selected bank, reducing the dynamic power consumption, as smaller decoders are needed, and less wordline and bitline capacitances are charged/discharged. The Divided Word Line (DWL) approach divides the array horizontally, propagating the decoder output on a global wordline, and subsequently raising the local wordline of a partition of columns, reducing the overall capacitance charged, and requiring smaller wordline drivers. Partitioning the columns using the Divided Bitline scheme, with partial multiplexing inside the array, reduces the bitline capacitance and in certain sensing schemes, will reduce the power consumption. All of these solutions come at the expense of additional area overhead, but a good tradeoff can achieve a worthwhile reduction of power consumption as well as an improvement in performance.

Using advanced timing and sensing schemes is another standard method to achieve a substantial dynamic power reduction. Using pulsed word lines and/or reduced bitline voltage swings, results in less discharge during read cycles, but is accompanied with complex design considerations and higher sensitivity to process variations. Timing the activation of sense amplifiers limits biasing currents to be present only during the exact times that the sensing is carried out. Additional low static power sense amplifiers, such as a Differential Charge Amplifiers and Self Latching Sense Amplifiers, also achieve static power reduction.

Many schemes have been proposed to reduce the bitcell leakage power, such as Supply Voltage Gating [Powell2000] [Flautner2002], Reversed Body Biasing (RBB) [Nii1998] [Hanson2003], Dynamic Voltage Scaling [Kim2002] and Negative Word Line (NWL) application [Wang 2007]. Recently, many proposals have shown minimum energy point operation of SRAMs in the subthreshold or near-subthreshold region. Examples of these include various works by Chandrakasan and Calhoun et.al. [Chandrakasan2007] [Chandrakasan2008] [Calhoun2007].

CMOS Image Sensor Architecture and Power Considerations

Traditionally, digital image sensors were fabricated in Charge Coupled Device (CCD) technology, but the integration of image sensors into more and more products, made the Active Pixel Sensor (APS) an attractive solution. This image sensor architecture is implemented in standard CMOS technology processes, and provides significant advantages over the CCD imagers in terms of power consumption, low voltage operation, and monolithic integration. With the rising popularity of portable, battery operated devices that require high-density ultra low power image sensors, the CMOS alternative has become very widespread. In addition, the CMOS technology allow for the fabrication of so called "smart" image sensors that integrate analog and/or digital signal processing onto the same substrate as the imager and its digital interface. Low power smart image sensors are very useful in a variety of applications, such as space, automotive, medical, security, industrial and others [Fish2007].

CMOS image sensors generally operate in one of two modes: rolling shutter or global shutter (snapshot) mode. When rolling shutter mode is used, each row of pixels is initiated for image capture separately in a serial fashion. This creates a slight delay between adjacent rows, resulting in image distortion in cases of relative motion between the imager and the scene. With the global shutter technique, the image is captured simultaneously by all pixels, after which the exposure is stopped, and the data is stored in-pixel while the image is read out. The operation of both techniques can be divided into three stages: *Reset, Phototransduction and Readout*. During the *Reset* stage, an initial voltage is set on the photodiode capacitance that constitutes most of the pixel area. Subsequently, the pixel enters the *Phototransduction* stage, during which the incident illumination causes the capacitance to discharge throughout a constant integration time. Readout is commenced at the end of the integration time, and the final value of the pixel is read out and converted to a digital value.

Figure 2 shows a component block diagram of a generic smart CMOS APS based image sensor. The core of the image sensor is a pixel array, generally consisting of a photodiode, in-pixel amplification, a selection scheme and a reset scheme. A full description of the operation of this pixel is given by Yadid-Pecht, et.al. [Yadid-Pecht2004].

Some smart imagers employ more complex pixels, enabling them to perform analog image processing at the pixel level, such as A/D conversion.

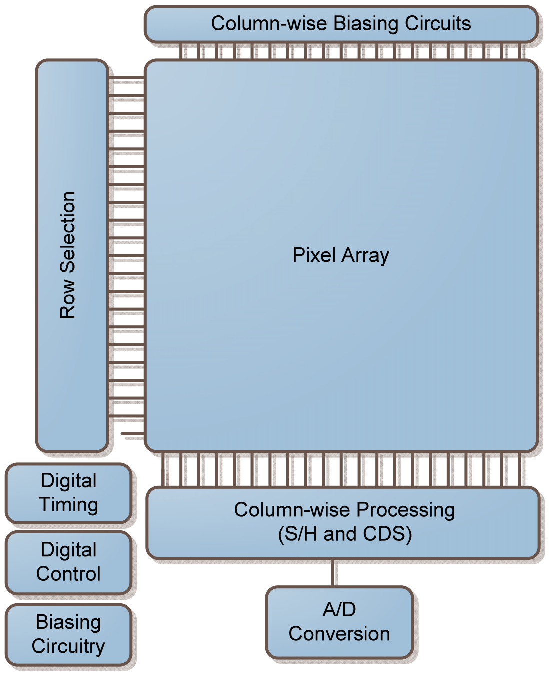


Figure 2: Generic Smart Image Sensor Component Block Diagram

Access to the pixels is carried out through the row selection block. This is usually made up of a shift register, as serial access is commonly employed, although in certain applications, a digital decoder is preferred. An entire row is generally accessed simultaneously for both reset and readout operations, except for in applications where random access is required, such as tracking window systems.

Several blocks are required at every column for the parallel operation of an entire pixel row. These include Sample and Hold (S/H) circuits, Corellated Double Sampling circuits and Analog to Digital Converters (ADC). The S/H circuitry generally measures the reset level of the pixels to enable the CDS to remove Fixed Pattern Noise (FPN). Column-wise ADCs are only one option; the others being in-pixel ADC or single ADC per imager. The selected scheme will be according to the tradeoffs of area, power, speed and precision.

Additional blocks that are required in the periphery of the imager include the general Biasing Circuitry and Bandgap References for creating biasing currents for the in-pixel signal amplifiers, usually implemented through a Source Follower (SF) scheme. and the ADCs; Digital Timing and Digital Control blocks for producing the proper sequencing of the addresses, ADC timing, etc.

The sensitivity of a digital image sensor is usually proportional to the area of the photodiode and the resolution is set by the number of pixels. This results in a relatively large area covered by the image sensor, compared to other on-chip circuits, and accordingly, a large percentage of the overall power consumption. The contribution of different image sensor components to overall power dissipation may vary significantly from system to system. For example, pixel array power dissipation can vary from a number of µWatts for a small array employing 3 transistor APS architecture to hundreds of mWatts for large format "smart" imagers employing in-pixel analog or digital processing. The power dissipation of the pixel array of a generic “smart” image sensor can be given by Equation 2:

$$P_{Array} = F_R \times N \times M \times \left(E_{reset} + E_{read\_out} + E_{analog} + E_{digital} \right) + N \times M \times P_{leakage} \qquad (2)$$

where F_R is the imager’s frame rate, N and M are the number of rows and columns, respectively, E_{reset} is the energy required for pixel reset, $E_{\mathrm{read\_out}}$ is the energy dissipated during signal readout during one frame, E_{analog} and E_{digital} are energy dissipation components dissipated by in-pixel analog and/or digital processing during one frame and P_{leakage} is the in-pixel leakage power.

The dynamic power in the above equation is proportional to the frame rate and is composed of the energy required to refill the photodiode capacitance during reset; the power dissipated through column-wise biasing currents during readout; and additional energy consumed by (optional) in-pixel functionality. The static power is due to leakage through the reset and row selection switches during integration and standby periods. These leakages also degrade the performance and precision of the imager.

The row selection block can be another major source of power dissipation, depending on the size of the array and the method of operation. In both global and rolling shutter modes, the row reset and row selection capacitances are periodically charged, proportional to the frame rate. In window tracking applications, on the other hand, the power of the row (and column) selection blocks can be dominated by leakage power, as the majority of the rows/columns may not be asserted for long periods.

The other primary source of power dissipation is the analog circuitry, including the ADCs, S/H, CDS and biasing circuitry. Optimally, these are timed to consume power only during their precise periods of operation, but they generally have a high power profile. The analog peripheral blocks present a constant tradeoff between speed, noise immunity, and precision versus power consumption and area, and for low power systems, the choice of these blocks needs to be made cautiously.

Additional power is dissipated in the digital timing and control blocks; however, the complexity and frequency of these tend to be lower than standard digital circuits, and so most common power reduction techniques can be implemented on these blocks.

An in depth description of all the contributions to power dissipation in a smart image sensor is given by Fish, et.al. [Fish2008].

Image sensors provide power reduction opportunities at all the design levels, starting with the technology and device levels, through the circuit level and all the way to the architecture and algorithm level. Standard power reduction techniques, such as supply voltage reduction and technology scaling, aren’t always applicable to CMOS image sensors, as they are frequently accompanied by unacceptable tradeoffs. Supply voltage reduction reduces both the precision and the noise immunity of image sensors, while technology scaling generally includes side effects, such as increased leakage current and dark current, as well as reduced photoresponsivity. However, at the technology level, processes can be modified for low power image sensor fabrication albeit, at an increased cost. An example of such a process is the Silicon-on-Sapphire (SOS) process that provides a very low power figure and enables backside illumination [Culuriciello2004].

The device and circuit level provide several opportunities for limiting power dissipation, depending on the options and layers provided by the chosen technology. The presence of separate wells for both nMOS and pMOS transistors enables the application of body biasing on inactive rows for leakage reduction. This technique loses its effect with scaling, as the effect on a devices threshold voltage is reduced, but image sensors are generally fabricated in technologies up to 90nm, where it is still efficient. Additional devices, such as high-VT transistors and thick oxide transistors can also be used for leakage reduction on slow busses. Another technique commonly used for leakage reduction is serial connection of “off” transistors for “stack effect” utilization [Narendra2001].

Smart image sensors provide many interesting opportunities for power reduction at the architectural and algorithm levels. Depending on the functionality of the sensor, these systems can be equipped with designated blocks for eliminating unnecessary power consumption. An example of this is the tracking sensor we proposed

[Teman2008] that used row and column shift registers for window definition and an analog winner-take-all circuit for motion tracking. In this system, the pixels outside the window of interest were deactivated and ADCs were used only for initial detection. The switching activity of the shift registers was very low, as well, further reducing the system power consumption.

Similarities between SRAMs and Image Sensors

In the previous two sections, the architectures and power profiles of two types of VLSI arrays, SRAMs and Image Sensors, were presented along with a number of examples of methods for reducing the power consumption of each. This section will deal with the similarities between these two architectures and their sources of power dissipation, arguing that low power approaches and methods developed for one type of array should be researched and adapted for the other.

Clearly, the first similarity between the two architectures is the two-dimensional array based structure of *m* rows and *n* columns of identical unit cells. SRAM bitcells have been optimized over the years to produce a dense layout to fit as much memory as possible onto a given area. This is possible due to the regular patterning of the cells, allowing many exceptions in design rules. The dense layout results in reduced capacitances, provides benefits in power and performance, as well as smaller peripheral circuits for a given memory size. In the case of pixel arrays, dense layouts provide similar benefits; however, the reduction in pixel size has a negative effect on pixel sensitivity, quantum efficiency, noise figures, etc. Various approaches for an optimal pixel layout have been proposed, such as the hexagonal shaped pixel [Staples2009].

For both SRAM bitcell and imager pixel design, leakage current during idle cycles (“hold” cycles for bitcells and “integration” cycles for pixels) has become a major focus. In some cases, smart image sensors contain memory circuits in-pixel, which further deepens the similarity. Utilization of leakage reduction methods, such as multiple threshold transistors and body biasing have been presented for both types of arrays. Modern CMOS processes include designated transistors for use in SRAM bitcells, optimized for leakage reduction. Several groups are researching low voltage operation of SRAMs in the subthreshold or near-subthreshold regions of operation. This approach could be used for image sensors, especially for operation of in-pixel or peripheral logic, due to their reduced frequency requirements.

Random or pseudo-random access to the unit cells in both types of arrays is achieved through row and column addressing. In SRAM design, the row addressing for wordline assertion is generally achieved through a row decoder, while standard image sensors, operating in the global or rolling shutter modes, use shift registers for reset and row selection. Both types of circuits are fitted to the pitch of the rows for layout and have been deeply researched for optimal operation in terms of power, area and performance, especially due to the fact that they drive large capacitances. Certain image sensors employ decoders for row addressing (such as the tracking window example, given above), while serially accessed SRAMs benefit from using a shift register. Other architectures have also been proposed, such as daisy chaining bitcells for robust digital column-wise readout [Chandrakasan2006]. SRAMs often save power and improve performance by sub-dividing the arrays into banks, local wordlines, etc. Image sensors could partially adopt similar techniques at opposing sides of the array or fully adopt them at the expense of losing several pixels that could be compensated for through signal processing.

Column addressing, which is inherent to most SRAM designs, is used in some image sensors, when random access is necessary. Column-wise operations are performed on the data of both types of arrays; SRAMs perform write-driving, precharging and readout in this fashion, while imagers perform column-wise CDS and readout. The primary noise cancellation mechanism for imagers is the CDS function, while SRAMs often employ dummy columns for timing and level comparison. Both concepts provide opportunities to be adapted to the other field.

Both fields employ analog blocks, necessary for performance, accuracy and functionality. SRAMs use sense amplifiers to speed up readout and reduce bitline swing, while imagers use ADCs to create a digital readout from the analog signal measured by the pixel. Both are done either column-wise or one-per array (or bank). Both require biasing currents for proper operation. Both are major power consumers and should be timed carefully to operate only when necessary. Smart image sensors sometimes use alternative readout blocks, such as Winner Take All (WTA) circuits, when binary decisions are required rather than precise level readout. Similar uses could be applied to SRAMs used by specific applications.

Finally, both architectures employ digital control and timing blocks to administer their respective operating modes. Image sensors require precise timing of their reset and integration signals, as well as for CDS and ADC operation. SRAMs are often asynchronously self-timed, employing Address Transition Detectors (ATD) and other circuits to initiate precharge, read and write phases. Digital control logic maneuvers the components between operation modes, and often registers are used to latch read out signals. Careful design of these timing and control blocks can provide substantial power savings.

Table 1. summarizes the architectural similarities between SRAMs and Image Sensors:

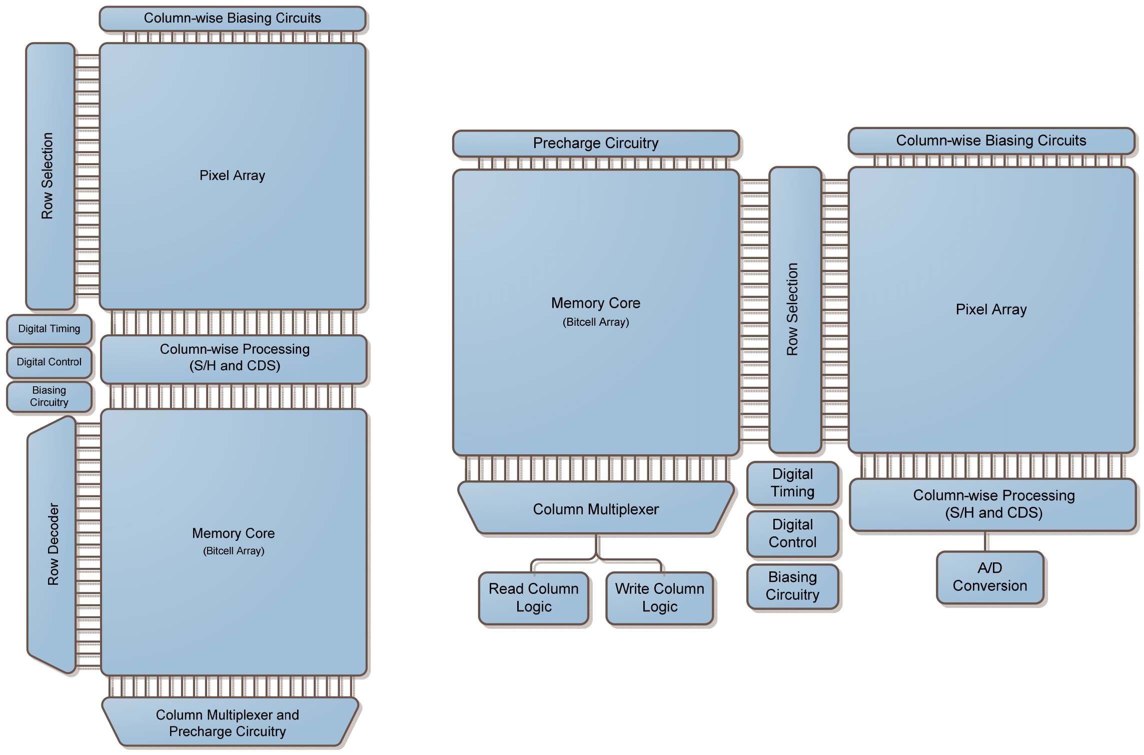
Designation	SRAM Component	Imager Component
mxn Array	Bitcells	Pixels, in-pixel memory, in-pixel ADC
Row Addressing	Decoder, Row Drivers, wordline	Shift Register, Row Drivers, Row Selection lines, Reset lines
Column Addressing	Column Multiplexer, Bitlines	Readout columns, optional column decoder/multiplexer
Column-wise Operation	Precharge circuits, Write Drivers, Bitlines, Column Sense Amplifier	Sample and Hold, CDS, Column ADC
Analog circuitry	Sense Amplifiers, Biasing Circuitry	ADC, Biasing Circuitry, Bandgap Reference
Timing and Control	Digital Control, Self Timing logic, ATD, Dummy Column, Error Correction	Digital Control, Digital Timing

Table 1: Summary of Architectural Similarities between SRAM and Image Sensors

Peripheral Sharing

In the previous section, we discussed the similarities between SRAM arrays and Image Sensors. The correlation between the two types of arrays is even more inherent in systems that employ both units, working in cohesion to achieve certain functionalities. This is often the case in smart image sensor systems that use SRAM arrays to temporarily store previously read out data or results of image processing. In such cases, the similarities provide several architectural opportunities for sharing peripherals, thus resulting in a reduction of both power and area, and often a performance improvement due to the inherent synchronization between the units.

Figure 3. shows two examples of peripheral sharing. In the Figure 3(a), a column-wise shared architecture is shown. In this case, the readout columns of the pixel array are directly connected to the vertical writing and/or reading logic of the SRAM. A possible application is a smart image sensor that periodically stores spatial data in an embedded SRAM for further use or processing. In this case, the parallel readout of the image is directly routed (through the column-wise processing, such as CDS, S/H and possibly ADC circuits) to the SRAM write drivers. SRAM operation is simplified to a one-dimensional (row) access scheme, as an entire row of data is read out from the image sensor and written in parallel. This architecture saves power and area, by simplifying or even eliminating the column addressing circuitry, integrating the timing and control signals of the two arrays, and even providing opportunity for replacing the SRAM’s row decoder with a much smaller and less power hungry shift register. Careful design can further reduce the digital and analog blocks by creating control signals and biases appropriate for both arrays.



(a) (b)

Figure 3. Two examples of peripheral sharing between SRAM and Imager arrays.

(a) Column-wise peripheral sharing. (b) Row-wise peripheral sharing

Figure 3(b) shows a possible row-wise approach to peripheral sharing. In this architecture, the two arrays are placed on a horizontal axis, enabling the distribution of row addressing signals via a mutual row selection block. One example would place a shift register in between the two arrays, asserting the reset and row selection lines of the imager in coordination with the wordlines of the SRAM, according to a predefined timing scheme. This method of SRAM addressing could be used in a serially accessed memory working in coordination with the adjacent imager. Certain applications would allow a further reduction in peripherals (saving both power and area), by integrating the column addressing blocks of the two arrays. Digital timing and control blocks could again produce common signals and analog blocks could be designed to use similar biasing levels, further integrating the two systems.

Several other peripheral sharing architectures and techniques could be proposed, depending on the application, the relationship between the smart imager and the SRAM and the operating profile of the system. Such peripheral sharing doesn't necessarily have to include complete integration between the two arrays. For example, a significant reduction in area could be achieved by using a single bandgap reference block for a standard imager and an SRAM array on the same die, even if they are independent of each other.

These architectural opportunities should be taken into consideration when developing a system that uses both types of blocks, as the saving in power and area, as well as the prospect of performance enhancement, can be considerable. The following section gives a practical implementation example of such a system.

Implementation Example: An Improved Adaptive Bulk Biasing Control (AB^{2}C) System

In the previous sections, we argued that image sensors and memory arrays have many common features and that power reduction techniques, developed for one field, could be adapted for the other. In addition, we proposed opportunities for peripheral sharing in systems, such as smart image sensors, that include both pixel arrays and embedded SRAM arrays. In this section, we will present an example of a system, developed by our group, that utilizes both approaches for power and area reduction, as well as performance optimization.

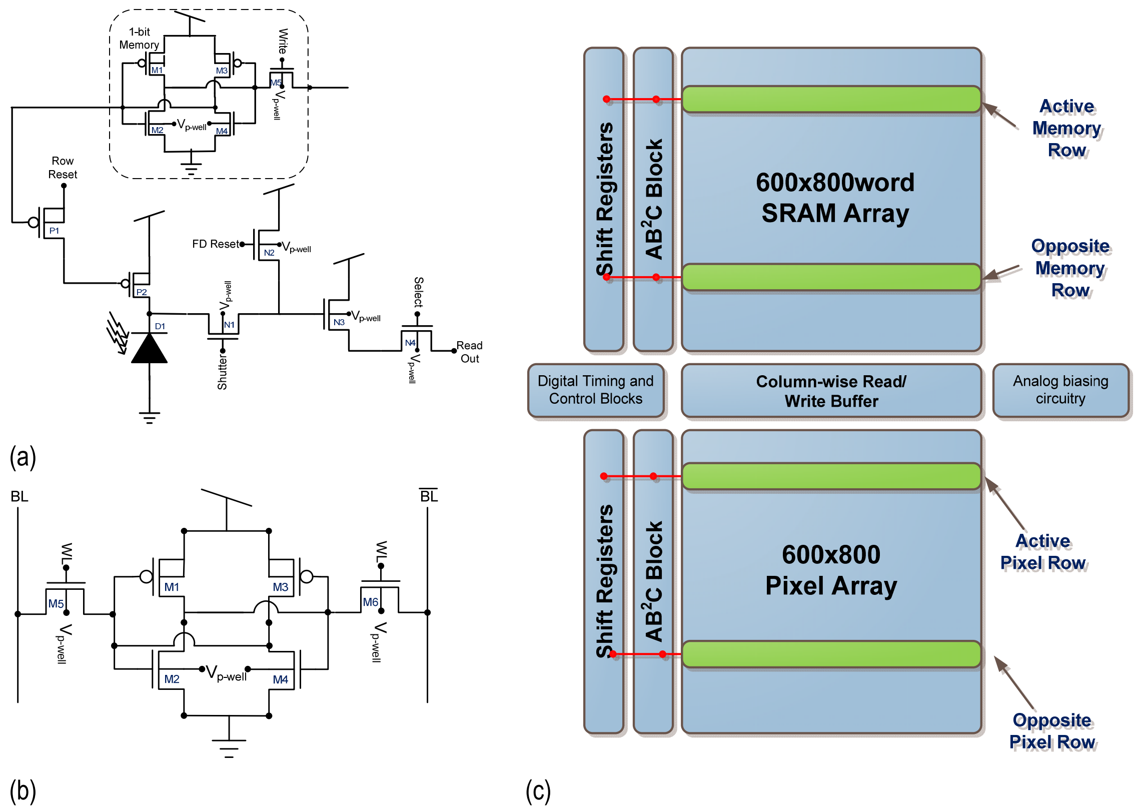


Figure 4. Architecture and basic circuits for Improved AB^{2}C System.

(a) Schematic of Smart Pixel (b) Schematic of SRAM Bitcell (c) Full system architecture

The Adaptive Bulk Biasing Control (AB^{2}C) approach to leakage reduction in image sensors was originally proposed by Fish, et. al. [Fish2007]. This system took advantage of the serial row access scheme, inherent to the majority of image sensors, for the application of a gradually changing body biasing to reduce the leakage in image sensors during the long integration periods. It can be shown that the slow voltage gradient applied to the bulk of a given row requires less power and causes less spatial noise than a standard pulsed approach. The system applies the full Reversed Body Bias (RBB) to the rows farthest away from the selected (i.e. reset or readout) row, and no RBB (or potentially a performance enhancing Forward Body Bias) to the selected row.

An improved AB^{2}C system [Teman2009] implements the original concept on a smart image sensor employing an embedded memory. Figure 4(a) shows the pixel circuit implemented in the smart image sensor employing an in-pixel memory bit. The serial access scheme of the smart imager includes a periodic partial readout of the pixel level, and according to the illumination level, data is written to both the in-pixel memory bit and the embedded SRAM array. After the full integration time, the final pixel level is read out along with the data stored at the associated SRAM address. This system provides opportunities for both row-wise and column-wise peripheral sharing, due to the synchronized serial operation of the image sensor with its associated SRAM addresses. A column-wise approach was chosen, as the parallel propagation of the column data to and from the SRAM proved to be more dense.

Implementation of the adaptive bulk biasing approach for leakage reduction in the SRAM was enabled by the serial access operation, inherent to the system. A twin-well was used to separately bias the bulks of each row of nMOS transistors, as the pMOS body biasing in deep submicron technologies (a standard 90nm TSMC process was used) is inefficient. The SRAM bitcell schematic is shown in Figure 4(b). The body nodes of the nMOS transistors was connected to the AB^{2}C circuit, driven by the row addressing shift register, used to serially access the array.

The full architecture for the Improved AB^{2}C system is shown in Figure 4(c). The column-wise setup enabled parallel writing of the image sensor readout values directly into the selected SRAM word below it, and subsequent readout of the SRAM value along with the final pixel value after integration. Similar row addressing blocks were used for both arrays, comprising shift registers for horizontally wired signals (reset, row select, wordlines) and AB^{2}C circuits. These circuits include a network of resistors with connections to the bulks of rows between them. The resistor network is biased with a voltage running between the active row and the opposite row (i.e. the row farthest away from the active row), thus creating a gradual voltage drop on the bulks of adjacent rows. The bias point is switched along with the row selection shift register, causing a small charge/discharge of the row bulk capacitance, with a minimal energy penalty. The row selection blocks (including AB^{2}C circuitry) could be shared between the two arrays, pending routing options, further saving area and power.

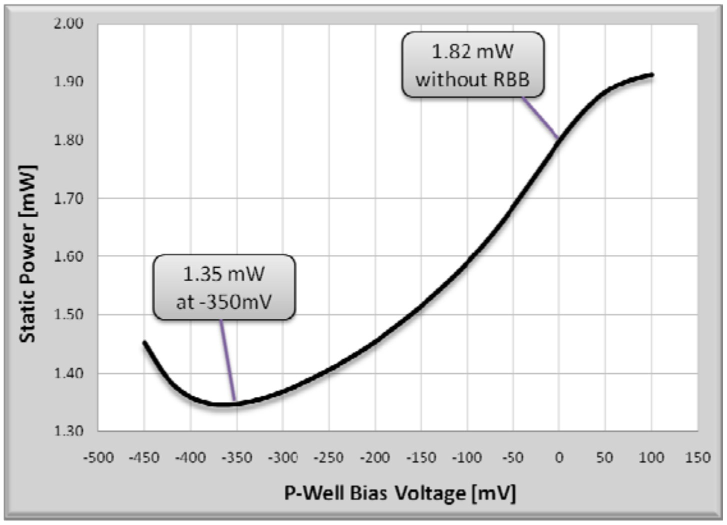


Figure 5: Static power consumption at various body biasing levels for the presented smart image sensor with embedded SRAM employing an AB^{2}C biasing scheme.

The static power reduction achieved with the application of the AB^{2}C architecture is plotted in Figure 5 for the presented system implemented in a standard TSMC 90nm CMOS process. The minimum energy point was achieved with a reverse biasing voltage of 350mV. A higher RBB results in higher power dissipation of the AB^{2}C blocks, while a lower RBB results in more pixel/bitcell leakage power. This results in a 26% power reduction as compared to the same system without the biasing voltage or the AB^{2}C power, as seen at the 0V point on the figure. This reduction improves with array sizes, and is even more effective at older technologies with a higher supply voltage, often used for image sensor implementation.

Conclusions and Further Research

A novel approach to power reduction in VLSI arrays was presented. The architectures of two types of arrays, image sensors and SRAM arrays were described. Sources of power consumption were noted for each array type, and some common techniques for power reduction were shown. It was contended that the similarities between the array types provide many opportunities for adaptation of methods and techniques for power reduction and optimization between the two. A number of architectural concepts based on peripheral sharing were suggested for systems employing both types of arrays. Finally, an example of a system that implements both approaches (method adaptation and peripheral sharing) was presented. The example showed an AB^{2}C scheme that was originally developed for image sensors and was implemented on an SRAM array, as well, providing a substantial static power reduction for the entire system. The image sensor and SRAM array were connected in a column-wise scheme, further saving both area and power, while optimizing the operation process.

Bibliography

[Rabaey2003] J.M. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2-nd Edition, Prentice Hall, 2003

[Itoh2001] K.Itoh, VLSI Memory Chip Design, Springer-Verlag, 2001

[Mamidipaka, 2004] M. Mamidipaka, K.I Khouri, N. Dutt , M. Abadir, Analytical models for leakage power estimation of memory array structures, In: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, Stockholm, Sweden , Sep. 2004.

[Flautner2002] K. Flautner, N.S. Kim, S. Martin, D. Blaauw, T. Mudge, Drowsy caches: simple techniques for reducing leakage power, In: Proceedings of the 29th annual international symposium on Computer architecture, Anchorage, Alaska, 2002.

[Powell 2000] M. Powell, S.H. Yang, B. Falsafi, K. Roy, T.N. Vijaykumar, Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories, Proceedings of the 2000 international symposium on Low power electronics and design, pg. 90-95, Rapallo, Italy, 2000

[Nii1998] K. Nii, H. Makino, Y.Tujihashi, C. Morishima, Y. Hayakawa, H. Nunogami, T. Arakawa, H. Hamano, A low power SRAM using auto-backgate-controlled MT-CMOS, Proceedings of the 1998 international symposium on Low power electronics and design, pg. 293-298, Monterey, California, 1998

[Wang2007] C.C. Wang, C.L. Lee, W.J. Lin, A 4-kb Low-Power SRAM Design with Negative Word-Line Scheme, IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, Vol. 54, No. 5, pp. 1069-1076, May 2007

[Hanson2003] H. Hanson, M.S. Hrishkesh, V. Agarwal, S.W.Keckler, D. Burger, Static energy reduction techniques for microprocessor caches, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, Issue 3, pg. 303-313, June 2003

[Kim2002] N.S. Kim, K. Flautner, D. Blaauw, T. Mudge, Drowsy instruction caches: leakage power reduction using dynamic voltage scaling and cache sub-bank prediction, Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture, pg. 219-130, Istanbul, Turkey, 2002

[Chandrakasan 2007] B.H. Calhoun, A.P. Chandrakasan, A 256-kb 65-nm Sub-threshold SRAM Design for Ultra-Low-Voltage Operation, IEEE Journal of Solid-State Circuits, vol. 42, no. 3, pp. 680-688, March 2007

[Chandrakasan 2008] N. Verma, A.P. Chandrakasan, A 256 kb 65 nm 8T Subthreshold SRAM Employing Sense-Amplifier Redundancy, IEEE Journal of Solid-State Circuits, pp. 141-149, January 2008

[Calhoun2008] J.Wang, B.H. Calhoun, Techniques to Extend Canary-based Standby VDD Scaling for SRAMs to 45nm and Beyond, IEEE Journal of Solid-State Circuits, Vol. 43, No. 11, pages 2514-2523, November 2008

[Fish2008] A. Fish, O. Yadid-Pecht, "Considerations for Power Reduction in "Smart" CMOS Image Sensors",,by Kris Iniewski, CRC Press, 2008

[Yadid-Pecht2004] O. Yadid-Pecht and R. Etienne-Cummings, CMOS imagers: from phototransduction to image processing, Kluwer Academic Publishers, 2004

[Culuriciello2004] E. Culurciello and A. G. Andreou, A 16x16 Silicon on Sapphire CMS Photosensor Array With a Digital Interface For Adaptive Wavefront Correnction, Proc. ISCAS, Vancouver, May, 2004.

[Teman2008] A. Teman, S. Fisher, L. Sudakov, A. Fish, O. Yadid-Pecht, Autonomous CMOS image sensor for real time target detection and tracking. Proc. of ISCAS 2008, pg. 2138-2141, 2008

[Narendra2001] S. Narendra, et. al., "Scaling of Stack Effect and its Application for Leakage Reduction," Proc. of ISLPED 2001, pp. 195-200, 2001

[Staples2009] C. J.Stapels, P. Barton, E. B. Johnson, D. K. Wehe, P. Dokhale, K. Shah, F. L. Augustine, J. F. Christian, Recent developments with CMOS SSPM photodetectors, Proceedings of the Fifth International Conference on New Developments in Photodetection, Pg. 145-149, October 2009

[Chandrakasan2006] A. Wang, B.H. Calhoun, A.B. Chandrakasan, Sub Threshold Design For Ultra Low Power Systems, Springer 2006

[Fish2007] A. Fish, T. Rothschild, A. Hodes, Y. Shoshan and O. Yadid-Pecht, Low Power CMOS Image Sensors Employing Adaptive Bulk Biasing Control (AB2C) Approach, Proc. IEEE International Symposium on Circuits and Systems, pp. 2834-2837, New-Orleans, USA, May 2007.

Authors' Information

Adam Teman *– Masters Student, The VLSI Systems Center, Ben Gurion University of the Negev, P.O. Box 653 Be'er Sheva 84105, Israel; e-mail: teman@ee.bgu.ac.il*

Major Fields of Scientific Research: VLSI, Digital circuit design, Low power memories and CMOS image sensors.

Dr. Orly Yadid-Pecht *– iCore Professor, Director of Integrated Sensors, Intelligent Systems (ISIS), University of Calgary, 2500 University Drive N.W. Calgary, Alberta, Canada T2N1N4, e-mail: orly@atips.ca*

Major Fields of Scientific Research: VLSI, CMOS Image Sensors, Neural Networks and Image Processing.

Dr. Alexander Fish *– Senior Lecturer, Head of Ultra Low Power Circuits and Systems Lab, The VLSI Systems Center, Ben Gurion University of the Negev, P.O. Box 653 Be'er Sheva 84105, Israel; e-mail: afish@ee.bgu.ac.il*

Major Fields of Scientific Research: Ultra-low power VLSI, Low-power image sensors, Mixed signal design.

RESEARCH PORTAL "REGIONS' INNOVATIVE DEVELOPMENT"

Lyudmila Lyadova, Zhanna Mingaleva, Natalia Frolova

Abstract: *This paper presents a project, aimed to create an information analytic system to solve the problem of organizing collective work of researchers, supporting their efficient cooperation on one of the topical problems in the sphere of economy – the problem of region's innovative development. The project supposes a creation of a portal that provides possibilities of publication, search, analysis and cataloging data on stated subject matter, as well as information exchange. In the system there should be presented not only publications, received from different sources, but also work results of the researchers, participating in the project, particularly, suggested models of innovative development of enterprises, economic sectors, regions, quantitative and qualitative assessment of their innovational development level in conditions of, on the one part, integration and on the other – intensification of competition. Special attention in the project is paid to the usage of up-to-date information technologies in conducting researches. The software of the portal includes means of information search in different sources, analytic processing of the information in accordance to developed methods. Access to the portal will be provided for users of different categories (scientists, lecturers, students, specialists in public authorities). The first stage is a creation of a research prototype of the system. Initial filling is expected to be executed on the base of data, issued by project participants (particularly, method of complex assessment of region's innovative development, which is based on the economic and mathematical methods and models; model of knowledge domain, built on the base of ontology and used for searching and analyzing papers and data; etc.).*

Keywords: *Innovations; Models of innovative development; Ontology; Intellectual search; Data analytic processing; Web-technologies.*

ACM Classification Keywords*: H. Information Systems. H.3 Information storage and retrieval: H.3.5 Online Information Services – Web-based services; H.3.6 Library Automation – Large text archives.*

Introduction

Slow paces of development of the innovation activity in the Russian economy and connected with this failures in introduction of innovation arrangements are mostly caused by the drawbacks of the system of analyzing innovation processes both in the whole sectors of the economy and within the bounds of particular economic subjects.

Significant aspect of competitiveness management in any economic system is safe and highly-qualified tools of evaluating main indicators of system development and achieved results, possibility of comparing them with other subjects, as well as possibility of a constant control of dynamics and directions of changes in key indicators of development. Special urgency and complexity is attached to the evaluation of innovation competitiveness level.

In such conditions the vital task is integration in common information space of available models and methods, their systematization, highlighting the main indicators of innovative development, as well as determination of integral indicator that characterizes the level of innovations.

One of the tasks that demand solution to achieve a goal is a creation of a set of instruments and tools for conducting researches on regions' innovative development, approbation of workable models and methods. The complexity of the task is determined by the necessity of research integration in the sphere of innovative development theory and usage of up-to-date information technologies. This will provide the possibility of new research approaches to investigation.

The project of creating the research portal “Regions' innovative development” is aimed to develop and test the prototype of the information analytic system of collecting and processing data on regions' innovation activities to support efficient managerial decision-making. Data for analysis are gained from heterogeneous unstructured or semistructured resources, particularly, Internet-resources, as well as on-line data bases. The system should provide integration, matching, aggregation and maintenance of previously uncoordinated data. In the workable system there should be different forms of data and research results visualization that will meet users' needs.

The research system is a set of instruments for economic analysis of innovation activities of particular departments, organizations, integrations, economic branches, regions. The portal should provide the possibility of collective work of researchers, the possibility of testing available models and methods of evaluating innovative development and innovational activity.

Researches on Innovational Activity

Investigation of domestic and foreign methods of evaluating innovation potential and innovation competitiveness has shown a significant divergence in approaches, as well as essential drawbacks of statistic account base [1-4].

First of all, it should be noted that there is an absence of methodological approach and a method of innovation competitiveness evaluation of economic systems in the Russian theory and practice. A level of innovations in business is considered through a level of enterprises' innovation activity, which, by turn, is determined by an indicator of innovation-active enterprises percentage. And here innovation-active enterprises are organizations that carry out development and introduction of new or improved products, technological processes and other kinds of innovation activity. Depending on a percentage of innovation-active enterprises in a certain economic branch (sector) Russian researchers speak about a level of innovation activity of a branch or of a whole region.

For a quantitative and qualitative assessment in domestic investigations on innovations researchers use such methods as: percentage of innovative products in an aggregate output; dimension of research and development costs; percentage of technological innovations costs in dimension of aggregate output; a number of introduced technologies; a number of created up-to-date technologies; percentage of innovation-active enterprises and others. However, all these indicators are examined independently and their interconnection and correlation are evaluated only through qualitative categories.

Thereby, it is hard to speak about integrity and common methodology of evaluating innovation potential and innovation competitiveness of economic systems (including enterprises) in domestic science.

Investigation of similar foreign literature has shown that the majority of methods, which are put in practice in developed countries, is based on a set of similar indicators. For instance, within the bounds of the method of determining economic branches' and sectors' innovation competitiveness, which is common in the European Union (EU), instead of direct quantitative assessment of each innovation activity factor (method of State statistic committee) an integral indicator is calculated. This indicator is a general outcome of innovation activity. As a result, with the help of UE method we can analyze the level of innovation activity not only of a particular branch, for example, cable branch, and compare it with a level of development in similar branches in other countries, but also evaluate a position of a particular enterprise in the branch, determine the difference between the innovation levels of different enterprises; point out those factors that are essential for a growth of innovation competitiveness, as well as factors that hinder enterprises' development.

One of the project's tasks is to develop a method of a complex assessment of region's innovation potential that will use economic and mathematical and statistic methods and models based on up-to-date information technologies and will help to determine a region's position among other subjects of innovation activity, mark out strong and weak points, compare advantages and disadvantages with main competitors (foreign and Russian), formulate main directions of innovative development.

Researches should go through the approbation on-line. It is necessary to provide the possibility of conducting analytic processing of received data, visualizing and comparing results, obtained from heterogeneous sources. Researchers should be able to have an access to the information, means of search and initial processing. The portal should provide these possibilities.

Information Analytic System in Innovational Activity Investigation

One of the main results of the project is to create a research system, which is aimed to integrate workable models, previously uncoordinated data, received from different sources in different formats (operational system data, accessible commercial data and others) into one storage, data coordination and further possibility of analytic processing.

The system should include statistic indicators, reports on evaluating innovations with the help of different methods, publications of different kinds on the matter, etc. the system should be used for accumulation, integration and maintenance of innovative development information. The research system should serve users' complicated requests both on search of necessary information and selection and processing. Different ways of forming requests, which will be available and handy for a user who is a specialist in a particular sphere, should be also provided. For introduction of research results there should be presenting information means in different forms (tables, graphs, diagrams, cartographical presentation, three-dimensional representation, as well as visualizing of models of different kinds).

Information system is carried out as a research portal and the main tasks of it are:

1. Computer-aided search of data sources among Internet-resources, devoted to innovation activity of particular departments and whole organizations, integrations, economic branches and regions, based on users' requests considering specificity of knowledge domain and its models.
2. Intellectual analysis, classification and cataloging of papers, gained from different sources, provision with easy-to-use navigation means for work.
3. Data gaining from received sources with the following matching and placing in one data storage.
4. Analytic processing of date placed in the storage.
5. Introduction of an access to the data storage with the help of convenient Windows- and Web-interface with a developed system of data visualizing and a handy set of instruments for creating reports.
6. Introduction of "portfolio" of economic and mathematical models of innovation activity with possibilities or editing available models of portfolio, development of new models by users and experimentation with the help of new models.

Users can exploit the system for analysis of available data, experimentation and approbation of built models and offered methods. Managers, analytics and researchers could use the system to analyze archival and operational data. The possibility of interpretation the most significant information is vital as well. The final goal of system introduction is to ease the users' access to the necessary information with the view of using it in decision-making, as well as the possibility of users' interaction in the process of investigations.

Realization of the Research Portal

Creation of such a system presupposes the necessity of using technologies that allow creating flexible, dynamically adaptable system with a high degree of feedback. This will allow carrying out its operational adjustment in changing conditions and according to specific needs of different users [5-6].

Different researches on the mattered are conducted in the sphere of information technologies, results of these researches are widely introduced in proceeding of conferences of different levels. Researches are devoted both to theoretical questions of creating adaptable systems and technologies, instruments (adaptability is investigated quite broadly: from the possibility of user’s interface adjustment to the possibility of whole restructing, reengineering of a system). However, nowadays there are only few program products that meet listed requirements (particularly, MetaCASE). In Russia today there is an absence of industrial systems of such level.

Technologies of data storage, operational analytic processing and intellectual analysis, search and analysis of text and semistructured data sources, visual analysis should be used to develop the system [7-8]. Data analysis presupposes usage of economic and mathematical, statistic and econometric methods. It is planned to use statistic, ontological methods, as well as methods of recognition, structural and semantic text analysis for search and gaining data from text and semistructured data sources.

For carrying out the project experience of early conducted works is used, particularly projects of creating means of developing dynamically adjusted Web-oriented systems [9].

Fig. 1 demonstrates the structure of the research system.

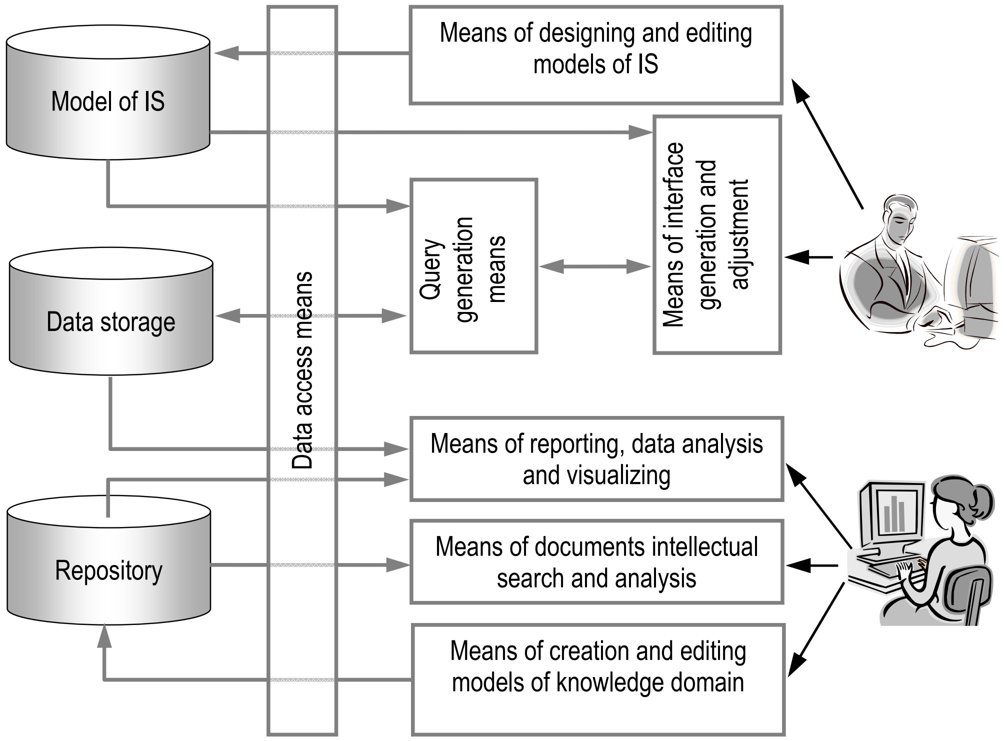

Model of IS
Data storage
Repository
Data access means
Means of designing and editing models of IS
Query generation means
Means of interface generation and adjustment
Means of reporting, data analysis and visualizing
Means of documents intellectual search and analysis
Means of creation and editing models of knowledge domain

Fig. 1. The structure of the information analytic research system

System functioning is based on interpretation of multilevel models, metadata, describing the informational system (IS). Means of modeling, creating and editing IS models allow IS adjusting in changing conditions and in accordance to users’ needs. Metadata of different levels describe data structures, user’s interface and main IS functions. Changes in models lead to changes in system functioning. Adjustment can be carried out dynamically in the process of system exploiting [5-6].

Means of documents intellectual search and cataloging are based on the usage of ontology [7]. Each user can create his or her own knowledge domain model and save it in the repository or use already designed models to search information responding user’s needs. These means provide not only information search but also information cataloging in the storage, easy-to-use means of paper navigation, data gaining for analysis.

Data analysis complex is oriented towards the work with multimentional data [8]. Users have an opportunity of creating their own data analysis models and generating reports, based on models designed by users. Each user has an opportunity to develop his or her own report models or use already developed ones.

Conclusion

Usage of research portal’s opportunities will allow not only publication of innovation economics theory papers but also their approbation with the usage of accumulated information on-line. In addition to this, portal’s means will provide users with a set of instruments for intellectual search and initial automation of information processing in accordance to users’ requests. Offered means allow significant reduction of work content, automation of operations that demand time costs. Open architecture of the portal and technologies used for project implementation allow expansion of portal’s opportunities, its functionality, adjustment of available means to meet the needs of users’ that work in different knowledge spheres.

Models and methods of innovative development evaluation, which are developed in the process of project implementation, have their own science and practical importance and can be used on different levels of management.

The research portal, which provides on-line access to data on theory of innovative development of enterprises, economic branches, regions, approbation of evaluation models and methods offered by users, possibility of collective work, will provide activization of collaboration in this sphere.

Offered technological solutions allow system adaptation in changing conditions and in accordance to users’ needs and these, by-turn, guarantee its vitality and possibility of development. These solutions also could be used to create information systems of different purpose in other knowledge domains.

Acknowledgements

The paper is carried out with the financial support of RSSF (project № 09-02-00373В/И).

Bibliography

[1] Инновационное развитие – основа модернизации экономики России: Национальный доклад. – М.: ИМЭМО РАН, ГУ-ВШЭ, 2008. – 168 с.

[2] Мингалева Ж.А., Гайфутдинова О.С. Основные методологические подходы к оценке уровня инновационной конкурентоспособности экономических систем // Методология планирования инновационного развития экономических систем. СПб, 2008. С. 648-681.

[3] Мингалева Ж.А. Оценка инновационного и научно-технического потенциала и инновационной конкурентоспособности регионов // Регион в новой парадигме пространственной организации России. – М.: Экономика, 2007. С. 556-576.

[4] Мингалева Ж.А. Развитие научно-технического и инновационного потенциалов региона: Монография. – Пермь: Пермский университет, 2006. 226 с.

[5] Лядова Л.Н. Метамоделирование и многоуровневые метаданные как основа технологии создания адаптируемых информационных систем // Advanced Studies in Software and Knowledge Engineering. International Book Series “Information Science & Computing”, Number 4. Supplement to the International Journal “Information Technologies & Knowledge”. Volume 2, 2008. Institute of Information Theories and Applications FOI ITHEA, Sofia, Bulgaria.

[6] Лядова Л.Н. Технология создания динамически адаптируемых информационных систем // Труды международных научно-технических конференций «Интеллектуальные системы» (AIS’07) и «Интеллектуальные САПР» (CAD-2007). Научное издание в 4-х томах. Т. 2. – М.: Физматлит, 2007. С.350 357.

[7] Ланин В.В., Лядова Л.Н., Чуприна С.И. Система интеллектуального поиска и автоматической каталогизации документов на основе онтологий // The XII th International Conference “Knowledge-Dialogue-Solution” (KDS’2006). Proceedings of conference / Varna (Bulgaria), June 20-25, 2006. Pp.139-145.

[8] Мальцев П.А. Моделирование многомерных данных в системе METAS BI-PLATFORM // International Book Series / Advanced Studies in Software and Knowledge Engineering. Sofia. 2008. С. 173-180.

[9] Хлызов А.В., Чичагова М.В. Создание динамически настраиваемых Web-ориентированных информационных систем // Научно-технический журнал «Инженерный вестник 1(21)/1 ‘2006» / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2006. С. 189-192.

Authors’ Information

Lyudmila Lyadova *– Perm Branch of the State University – Higher School of Economics, associate professor of Business IT Department; 38 Studencheskaya Street , Perm, Russia, 614070; e-mail: LNLyadova@mail.ru.*

Zhanna Mingaleva *– Perm State University, head of the National Economy and Economic Safety Department, 15 Bukyreva Street, Perm, Russia, 614990; e-mail: mingal1@psu.ru.*

Natalia Frolova *– Perm State University, associate professor of a Department of Information Systems and Mathematical Methods in Economics, 15 Bukyreva Street, Perm, Russia, 614990; e-mail: nvf_psu@mail.ru.*

ITHEA INTERNATIONAL SCIENTIFIC SOCIETY

The ITHEA International Scientific Society (ITHEA ISS) a successor of the international scientific co-operation organized within 1986-1992 by international workgroups (IWG) researching the problems of data bases and artificial intelligence. As a result of tight relation between these problems in 1990 in Budapest appeared the international scientific group of Data Base Intellectualization (IWGDBI) integrating the possibilities of databases with the creative process support tools. The leaders of the IWGDBI were Prof. Victor Gladun (Ukraine) and Prof. Rumyana Kirkova (Bulgaria). Starting from 1992 till now the international scientific co-operation has been organized by the Association of Developers and Users of Intellectualized Systems (ADUIS), Ukraine. It has played a significant role for uniting the scientific community working in the area of the artificial intelligence.

To extend the possibilities for international scientific collaboration in all directions of informatics by wide range of concrete activities, in 2002 year, the Institute for Information Theories and Applications FOI ITHEA (ITHEA) has been established as an international nongovernmental organization.

ITHEA Scientific Council includes:

Honorable Chair Persons: Victor Gladun (Ukraine) and Rumyana Kirkova (Bulgaria);

Chairman: Levon Aslanyan (Armenia);

Members: Adil Timofeev (Russia), Alexander Kleshchev (Russia), Alexander Kuzemin (Ukraine), Alexander Palagin (Ukraine), Alexey Voloshyn (Ukraine), Arkadij Zakrevskij (Belarus), Constantin Gaindric (Moldova), Hasmik Sahakyan (Armenia), Ilia Mitov (Bulgaria), Juan Castellanos (Spain), Koen Vanhoof (Belgie), Krassimir Markov (Bulgaria), Larisa Zainutdinova (Russia), Laura Ciocoiu (Romania), Luis Fernando de Mingo (Spain), Lyudmila Lyadova (Russia), Martin Mintchev (Canada), Nikolay Zagoruiko (Russia), Peter Stanchev (USA), Stefan Karastanev (Bulgaria), Tatyana Gavrilova (Russia), Vladimir Lovitskii (GB), Vladimir Ryazanov (Russia).

ITHEA Scientific Sections: Business Informatics (Koen Vanhoof), Cooperative Scientific Projects (Hasmik Sahakyan), Decision Support (Alexey Voloshyn), Information Interaction (Stefan Karastanev). Intelligent Engineering Systems (Martin Mintchev), Logical Inference and Design (Arkadij Zakrevskij), Modern (e-) Learning (Larisa Zaynutdinova), Natural Computing (Juan Castellanos), Pattern Recognition (Vladimir Ryazanov), Philosophy and Methodology of Informatics (Krassimir Markov), Scientific Innovation Management (Luis Fernando de Mingo), Software Engineering (Ilia Mitov), Theoretical Informatics (Levon Aslanyan).

ITHEA is aimed to support international scientific research through international scientific projects, workshops, conferences, journals, book series, etc. The achieved results are remarkable. The ITHEA became world-wide known scientific organization. One of the main activities of the ITHEA is building an International Scientific Society aimed to unite researches from all over the world who are working in the area of informatics.

Now, the ITHEA International Scientific Society is joined by more than 2200 members from 44 countries all over the world: Armenia, Azerbaijan, Belarus, Brazil, Belgium, Bulgaria, Canada, Czech Republic, Denmark, Egypt, Estonia, Finland, France, Germany, Greece, Hungary, India, Iran, Ireland, Israel, Italy, Japan, Jordan, Kyrgyz Republic, Latvia, Lithuania, Malaysia, Malta, Mexico, Moldova, Netherlands, Poland, Portugal, Romania, Russia, Scotland, Senegal, Serbia and Montenegro, Spain, Sultanate of Oman, Turkey, UK, Ukraine, and USA.

ITHEA IJ AND IBS SAMPLE SHEET FOR PREPARING THE MANUSCRIPTS

Krassimir Markov, Ilia Mitov

***Abstract**: The rules for preparing the manuscripts for the International Journals (IJ) and International Book Series (IBS) of the ITHEA International Scientific Society (ITHEA ISS) are outlined. The form for the papers is shown by this sheet. An extended up to one page abstract in the same form needs to be submitted separately.*

***Keywords**: formatting rules (Keywords are your own designated keywords).*

***ACM Classification Keywords**: A.0 General Literature - Conference proceedings (This is just an example, please use the correct category and subject descriptors for your submission. The ACM Computing Classification Scheme: http://www.acm.org/class/1998/. For instance see http://wiki.ithea.org/)*

***Conference topic**: Every paper submitted to be published by ITHEA ISS need to be presented at any of the ITHEA International Conferences. Please point here the right topic of the chosen ITHEA International Conference where the paper will be presented (topics are available on the corresponded conference web-page accessible via http://www.ithea.org or http://www.foibg.com).*

Introduction

We ask authors to follow some simple guidelines.

In essence, we ask authors to make papers look exactly like this document.

This text is a sample for preparing the manuscripts for publishing in ITHEA ISS International Journals and Book Series. All styles needed for formatting the papers are included.

The easiest way to prepare your manuscript in accordance of these rules is simply to replace the content of this sample sheet with your own material.

Responsibility for papers published in ITHEA International Journals and Book Series belongs to authors.

Please *get permission to reprint* any copyrighted material.

The camera-ready copy of the paper should be received by the ITHEA Journal Submission System (http://ij.ithea.org) or respectively by the ITHEA Conference Submission System (http://ita.ithea.org); e-mail for questions: *info@foibg.com*.

Instructions for Preparation of Manuscripts

The authors are hoped to prepare manuscripts in close accordance with the instructions given below.

This text is a sample for preparing the articles. All styles needed for formatting the papers are included. Do not include any new styles. Please, *do not use automatic numbering anyway*, because of losing the information during the assembling the journals or books.

Name the file of the manuscript beginning with the journal or conference name, following with the family names of the authors or if they are more than 2 authors – name of the first author, followed by "_et_al".

For instance if the manuscript will be submitted to:

- IJ ITK 2010 from Markov and Mitov, than the file needs to be named:
  "IJITK10-Markov_Mitov.doc" ;
- IJ ITA 2010 from Markov, Ivanova, and Mitov, than the file needs to be named:
  "IJITA10-Markov_et_al.doc";
- i.TECH 2010 from Markov and Mitov, than the file needs to be named:
  "iTECH10-Markov_Mitov.doc";
- i.TECH 2010 from Markov, Ivanova, and Mitov, than the file needs to be named:
  " iTECH10-Markov_et_al.doc".

Accepted manuscripts will be published as follow:

- surveys from 12 up to 20 pages will be published in the International Journal "Information Theories and Applications"® (IJ ITA) or International Journal "Information Technologies and Knowledge"® (IJ ITK);

- regular papers from 4 up to 12 pages will be published in specialized thematically organized collections in the International Book Series "Information Science and Computing" (IBS ISC);

- papers less than 4 pages will be assumed as extended abstracts and will be published in the ITHEA ISS International Review Journal “Information Research and Engineering”;

- books or specialized thematically organized collections will be published in the International Book Series "Information Science and Computing" (IBS ISC).

Manuscripts must be submitted within the stipulated time and electronic submission in DOC, DOCX or RTF formats is required.

Manuscripts will be evaluated for originality, significance, clarity, and soundness, and will be reviewed by at least two independent reviewers.

The authors of the accepted manuscripts will be allowed to make a correction in accordance with the suggestions of the reviewers and to submit final camera-ready manuscripts within the stipulated deadline.

The papers should be organized so as to accommodate abstract, introduction, state-of-the-art, objective, used methodology, obtained results and comparing them with similar results in the world, and references.

Format of the pages is A4 paper (210 x 297mm).

Margins of the paper sheet are: top - 30mm; bottom, left, right - 25mm.

Typing styles: The paper should begin with the title of the paper (use the style "Title") and the name(s) of the author(s), (use the style "Authors"). Please, write the whole ***first name and family*** of the authors.

After that apply style "Abstract" for: "Abstract", "Keywords" and "ACM Classification Keywords". Papers should contain up to 5 keywords.

The abstract needs to be from 200 to 500 words long. An extended up to one page abstract needs to be submitted separately.

Note that the abstract is very important for:

- directing the paper to the right reviewers;
- including the paper in the right section of the journals or collection in the book series;
- representing the paper in the ITHEA International Review Journal.

Use "Normal" style - Arial Narrow; 11pt; 1.2-spaced text; 3pt before each paragraph; without special indents; left and right justification.

Use style "Subtitle" for the titles of the separated parts of the article.

The papers need to be well structured. This means that "Introduction", "Conclusion", "Acknowledgements", "Bibliography" and "Author's Information" need to be separated clearly. The body of the paper needs to be organized in different parts named using style "Subtitle".

Figures and tables should be positioned in the body of the text, as close as possible to the relevant text. ***Number manually*** all figures and tables. Use these numbers to point them in the text. Note that the position of figures and tables may be changed during the assembling the journals or books. Color figures are good for electronic variant but they will be printed in grayscale and some colors may look as equal.

Formulas should be positioned in the body of the text, as close as possible to the relevant text. Put formula and its number in a table row without borders. Align the formula to the center and its number to the right as follow:

$$D = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

The MathType INI v1: Equation Preferences (given in ITHEA.eqp) are:

```
[Styles]
Text=Arial
Function=Arial
Variable=Arial,I
LCGreek=Symbol,I
UCGreek=Symbol
Symbol=Symbol
Vector=Times New Roman,B
Number=Times New Roman
User1=Arial Narrow
User2=Arial Narrow
MTExtra=MT Extra

[Sizes]
Full=11 pt
Script=58 %
ScriptScript=42 %
Symbol=150 %
SubSymbol=100 %
User1=75 %
User2=150 %
SmallLargeIncr=1 pt

[Spacing]
LineSpacing=150 %
MatrixRowSpacing=150 %
MatrixColSpacing=100 %
SuperscriptHeight=45 %
SubscriptDepth=25 %
SubSupGap=8 %
LimHeight=25 %
LimDepth=100 %
LimLineSpacing=100 %
NumerHeight=35 %
DenomDepth=100 %
FractBarOver=8 %
FractBarThick=5 %
SubFractBarThick=2.5 %
FractGap=8 %
FenceOver=8 %
OperSpacing=100 %
NonOperSpacing=100 %
CharWidth=0 %
MinGap=8 %
VertRadGap=17 %
HorizRadGap=8 %
RadWidth=100 %
EmbellGap=12.5 %
PrimeHeight=45 %
BoxStrokeThick=5 %
StikeThruThick=5 %
MatrixLineThick=5 %
RadStrokeThick=5 %
HorizFenceGap=10 %
```

References in the text should be keyed with the name(s) and year of the referred material - for instance [Shannon, 1949].

Put list of bibliography after the text of the article using the style "Bibliography".

Author's Information: Finish the article with the personal information for every author separately: photo, name of the author, position, organization(s), post and e-mail address(es), major fields of scientific research (keywords). For this information use style "Normal-Authors".

Note that the only way to contact the authors is pointed e-mail address in the author's information. Be sure that the addresses are written correctly. If you (or your internet provider) use anti-spam protector write the way to access the e-mail address.

Conclusion

This exemplar is meant to be a model for manuscript format. Please make your manuscript look as much like this exemplar as possible. In case of serious deviations from the format, the paper will be returned for reformatting.

NOTE: Every manuscript submitted to be published by ITHEA ISS need to be presented by author(s) personally at any of the ITHEA International Conferences.

Papers from members of the ITHEA International Scientific Society (ITHEA ISS) will be published preferably. Membership of ITHEA ISS is free and may be done by registration at the *www.ithea.org* and *www.foibg.com*. Submitted manuscripts should be original and should contain contributions of theoretical, experimental or application nature, or be unique experience reports.

Bibliography

[Shannon, 1949] C.E.Shannon. The Mathematical theory of communication. In: The Mathematical Theory of Communication. Ed. C.E.Shannon and W.Weaver. University of Illinois Press, Urbana, 1949.

Authors' Information

Krassimir Markov *– ITHEA ISS IJ, IBS and IRJ Editor in chief, P.O. Box: 775, Sofia-1090, Bulgaria; e-mail: markov@foibg.com*

Major Fields of Scientific Research: General theoretical information research, Multi-dimensional information systems

Ilia Mitov *–Vice-president, Institute of Information Theories and Applications FOI ITHEA, P.O. Box: 775, Sofia-1090, Bulgaria; e-mail: mitov@foibg.com*

Major Fields of Scientific Research: Business informatics, Software technologies, Multi-dimensional information systems

TABLE OF CONTENTS