P Systems with Control Nuclei

Gheorghe Ştefănescu[1], Traian Şerbănuţă[2], Camelia Chira[3], and Grigore Roşu[2]

[1] University of Bucharest
gheorghe@funinf.cs.unibuc.ro
[2] University of Illinois at Urbana-Champaign
{tserban2,grosu}@cs.uiuc.edu
[3] Babes-Bolyai University Cluj-Napoca
cchira@cs.ubbcluj.ro

Summary. We describe an extension of P-systems where each membrane has an associated control nucleus responsible with the generation of the rules to be applied in that membrane. The nucleus exports a set of rules which are applied in the membrane region (only for one step, but in the usual maximal-parallel way), then the rules are removed and a new iteration of this process takes place. This way, powerful control mechanisms may be included in P-systems themselves, as opposed to using the level of "strategies" previously exploited for simulating P-systems. The nuclei may contain general programs for generating rules, ranging from those using information on the full system, to more restricted programs where only local information in the nuclei themselves is used. The latter approach mixed with a particular mechanism for the representation of the control programs, the rules, and the export procedure is engaged to develop a model for cell growth and division in normal and abnormal (tumoral) evolution of biological systems.

1 Control nuclei

The relation between P-systems and the K rewrite-based framework is thoroughly exploited in [4] where an extension of P-systems with structural data has been introduced, accompanied by an implementation using K and Maude rewriting engine. In this paper, we describe a further extension of P-systems, briefly mentioned in [4], obtained by integrating powerful mechanisms (called "control nuclei") to control the activity in P-systems. With these two extensions, we foreseen the development of a high level modeling and programming language on top of P-systems, powerful enough to simulate the behavior of complex, real biological systems.

A *P-system with control nuclei* (PCN for short) is a P-system [2] where each membrane has an associated nucleus with a program for generating the rules to be used within the membrane region. The semantics of a PCN is simple: repeatedly, at each running step, a set of rules is generated by the nucleus program in each membrane and the usual maximal-parallel rule for P-systems is applied (for one step, only).

```
Pa::
while(true){
  export R1;
}
```

```
Pb::
while(true){
  export prim(R1);
  ⋮
  export prim(Rn);
  export Unprim;
}
```

```
Pc::
(code of membrane i)
x = i mod n;
goto x;
while(true){
  0: export R0;
  ⋮
  n-1: export R(n-1);
}
```

Fig. 1. Examples of nucleus programs

Figure 1 presents a sample of nucleus programs, written in a conventional programming language, but enriched with an `export` statement. A program is executed from its current control point until it reaches an `export` statement. In such a point, the program is stopped and the exported rules are applied in the membrane region. When this transforming process is finished and the rules are discarded from the membrane, the nucleus program in reactivated, starting from its last control point, until a new `export` statement is reached and the process is repeated.

In the first example in Figure 1 (labelled `Pa`), the nucleus constantly produces the same set of rules `R1`, and therefore a PCN using such nucleus programs is actually a standard P-system.

The second example `Pb` illustrates a program dealing with priority strategies for applying the rules. Before going into details, some explanations on the notation are necessary: by `prim(R)` we denote the set of rules obtained from `R` by a decoration with $'$ (prim) of its right-hand side terms; `Unprim` is the rule which strip out the prim decoration of all terms. The program `Pb` acts as follows: it first generates and applies the rules in `R1`; when no such rules may be applied, the rules in `R2` are applied, and so on; by using prim-unprim decorations we constrain the rules `R1, R2, ...` to be applied to the original elements in the membrane region, prohibiting the use of the newly produced values in subsequent rules.

The third program `Pc` illustrates a kind of pipelined synchronous execution in a P-system. Let us suppose that the P-system has `m` membranes (denoted from 0 to `m-1`) and `n` sets of rules `R0,...,R(n-1)`. Each membrane infinitely repeats a cyclic execution of the rules `R0,...,R(n-1)`, but starting with a different rule. For instance, if `m=4` and `n=3`, the system uses the following rules in its membranes `0,...,3`: `(R0,R1,R2,R0), (R1,R2,R0,R1), (R2,R0,R1,R2), ...`

The sample programs in Figure 1 are simple examples used to illustrate the concept of control nuclei. Actually, any kind of nucleus programs may be used in PCNs. A particular benefit of the implementation of P-systems in K, described in [4], is that it can be easily adapted to use such powerful nucleus programs - just mix the P-systems implementation in [4] with the known representation of several common programming languages in K.

A particular technical problem, hidden by the informal presentation above, is the way to handle the application of a rule which dissolves the membrane. The unanswered question is the following: what happens with the nucleus program of the dissolved membrane? A few options can be sketched here, grouped in two classes: (1) a further extension of PCNs to have more nuclei (and nucleus programs) associated to a membrane; (2) the application of certain rules to combine the nucleus program of the dissolved membrane with the nucleus program of the parent membrane. Subsequently, the latter option opens a full range of possibilities to combine nucleus programs, which are not detailed here.

2 Modeling cell normal and abnormal development

In this section, we briefly describe how PCNs (P-systems with control nuclei) can be used to model cell growth and division in biological systems, both in normal and abnormal (tumoral) developments. The reader is directed to [1] for further explanation and details about the terms, concepts, and phenomena used in this section.

The abstract development of PCNs in the previous section, based on programs written in conventional programming languages, is better suited for "in silico" models. To tackle "in vivo" systems, we present a particular low-level DNA-based biological representation of the nucleus programs and of the transformation rules. The resulted systems are named *biological P-systems with control nuclei* (BPCNs for short).

The transformation rules associated to membranes in BPCNs include rules with the following format

$$a \rightarrow p(c) \text{ if } c$$

Such a rule represents an abstract formulation of a part of the transcription process where the DNA/RNA code c is used to transform the aminoacids in a into $p(c)$, the protein represented by c. Notice that, in each step, a single c can be used for several transformations of a's in $p(c)$'s. To be consistent with the PCN semantics previously developed, the code c has to become inactive at the end of the transformation step, either being degraded or moved in a trash/inactive area (for instance in the nucleus). From a biological perspective, it is not clear why a code c is to be lost at the end of a transformation step, and perhaps a variation of the model where a code c is allowed to be active during several transformation steps is more appropriate.

Like in ordinary P-systems, a rule result $p(c)$ can migrate in another membrane. Unlike ordinary P-systems, in BPCNs a protein $p(c)$ can also migrate into control nuclei and can be attached to specific spots of the DNA, with an activation or inhibition result of the related gene.

The nucleus consists of a strand of DNA, a sequence of basic nucleotides separated in "genes," each gene codifying a protein. On the DNA strand, several proteins are attached at specific spots. In a current configuration, the DNA strand

has one or more control points where active genes are copied and exported into the membrane region for transcription. The specific program (or mechanism) used to get the position of the control points for the active genes used in a next transcription step is left unspecified at this moment. (A simple option could be that each control point travels along the DNA strand and stops at the first active gene. However, this is an oversimplification, as it does not take into account the dynamics and the timing of the attachment of the proteins to the DNA strand, or the insertion or deletion of new control points.)

While in P-systems one could use an expensive abstract rule to duplicate a membrane and its contents, in BPCNs one could develop a more detailed mechanism for cell growth and division, closer to the real processes seen in biological systems. The payoff for this effort of having a detailed representation of the division process is that one could also model and study abnormal (tumoral) development of the cells.

According to [1], Chapter 27, the cell cycle consists of the following phases: (G0) - a commitment is taken towards a division process; (G1) - this is a growth stage where RNA and proteins are synthesized; (S) - this phase contains DNA replication; (G2) - during this period, the cell gets two complete diploid sets of chromosomes; (M-mitosis) - here the nucleus is dissolved and the daughter cells are created.

Abstract versions of most of these processes can be modeled in BPCNs as follows: (G0) - a starting control point is inserted in a code at a point where the division is codified; (G1) - a duplication rule $x \to x\ x$, where each membrane component gets a copy, may be used for this growth stage; (S) - as transformation rules take place in membrane regions only, the DNA duplication is slightly complicated: the full DNA code is exported into the membrane region, duplicated there, and finally moved back into the nucleus; (G2) - the abstract version of this stage is not completely clear at this moment - it may have to deal with the need to have the same "control points" in both copies of the DNA strands (as in Unix processes obtained by the `fork` command); (M) - in this stage the nucleus is divided into two (this is opposite to the previously mentioned process of joining two nuclei); finally, use a rule to divide a membrane in two membranes, with an even separation of its contents into the daughter membranes.

The described BPCNs for development of the cells and their division could be easily adapted to take into account tumor attacks. The result of a DNA tumor viral infection, roughly falls into two categories: (1) permissive cells allow for multiplication of the DNA virus, then the cell dies and the viral DNA is spread into the neighboring cells; (2) nonpermissive cells may sometimes be infected by the insertion of the viral DNA into the nucleus DNA, changing the cell phenotype (in particular, after cell division, the daughter cells inherit an infected nucleus).

To conclude, P-systems with control nuclei, in both their abstract and more biologically motivated forms, promise to be a good candidate for modeling and understanding the evolution of complex (including biological) systems.

References

1. Lewin, B.: Genes VIII. Oxford University Press (2004).
2. Paun, G.: Computing with membranes. Journal of Computer and System Sciences, 61, 108–143 (2000).
3. Rosu, G.: K: A Rewriting-Based Framework for Computations – Preliminary version. Technical Report UIUCDCS-R-2007-2926, Department of Computer Science, University of Illinois (2007). `http://fsl.cs.uiuc.edu/k`.
4. Serbanuta, T., Stefanescu, G., Rosu, G.: Defining and Executing P-systems with Structured Data in K. In: Proc. Workshop on Membrane Computing 2008, LNCS 5391, pp374–393. Springer, Berlin (2009).
5. URL: The Web Page of Membrane Computing: http://ppage.psystems.eu/