Dynamics of Randomly Constructed Computational Systems

Miguel A. Peña[1], Pierluigi Frisco[2]

[1] Dpto. Inteligencia Artificial, Facultad de Informática
Universitad Politécnica de Madrid
Campus de Montegancedo, 28660 Madrid, Spain
m.pena@upm.es
[2] School of Mathematical and Computer Sciences
Heriot-Watt University, EH14 4AS Edinburgh, UK
P.Frisco@hw.ac.uk

Summary. We studied Petri nets with five places constructed in a pseudo-random way: their underlying net is composed of *join* and *fork*. We report initial results linking the dynamical properties of these systems to the topology of their underlying net.

The obtained results can be easily related to the computational power of some abstract models of computation.

1 Introduction

Recently [4, 5, 1, 7], several abstract models of computation operating with multisets of objects have been related to Petri nets. This study let to define new ways to prove the computational power of these abstract models of computations. Moreover, it also let a hierarchy of computational process to be defined. This hierarchy is based on *building blocks* (small Petri nets used to construct more complex Petri nets), the way the building blocks are combined, and the way the Petri net runs.

The results related to this hierarchy have been obtained using systems created *ad hoc*: the Petri nets were engineered in specific ways so to be able to generate specific languages. The languages generated by 'pseudo-random' Petri nets, remain to be investigated. Here 'random' refers to the fact that Petri nets are created composing building blocks in a random way, while 'pseudo' refers to the fact that some limitations to this randomness or to the way the Petri nets runs, are imposed. This direction of research was raised in [3] (suggestion for research 4).

In this paper we report our initial results on these investigations. The overall aims of this research is to be able to predict the behaviour of a computing system just looking at what has been called *topology of information flow* [6], that is, at the way the several parts of the system interact.

2 Basic Definitions

The model of Petri nets considered by us are known either as *elementary net systems* (*EN systems*), as *1-bounded place-transitions systems* (*1-bounded P/T systems*), or *safe Petri nets* [8].

An *elementary net system* (or *EN system*) is a tuple $N = (P, T, F, C_{in})$, where:

i) (P, T, F) is a *net*, that is:
   1. P and T are sets with $P \cap T = \emptyset$;
   2. $F \subseteq (P \times T) \cup (T \times P)$;
   3. for every $t \in T$ there exist $p, q \in P$ such that $(p, t), (t, q) \in F$;
   4. for every $t \in T$ and $p, q, \in P$, if $(p, t), (t, q) \in F$, then $p \neq q$;

ii) $C_{in} \subseteq P$ is the *initial configuration* (or *initial marking*).

Elements of P are called *places* (graphically represented with circles), elements of T are called *transitions* (graphically represented with rectangles). We use the common Petri net terminology and notation [8] with the exception of using the term *configuration* instead of *marking*.

We consider *maximal strategy* as running mode (i.e., the way transitions fire): in each configuration all transitions that can fire do so. Moreover, if in a configuration there is a conflict (two different transitions with a not empty intersection of input sets can fire), then all the transitions in the conflict fire. If after a firing a place should receive more than one token (from the firing of two different transitions), then only one token is assumed to be present in that place. This ensures that in every configuration places have at most one token and that the behaviour (sequence of configurations) of an EN system is deterministic.

This rather restrictive firing strategy (similar to the ones present in random Boolean networks [2]) has been mainly dictated by efficiency during these initial simulations. In section 5 we note that the firing strategy should be definitely changed in order to obtain results of a more general use.

We considered EN systems composed of only two building blocks: *join* and *fork* depicted in Figure 1.

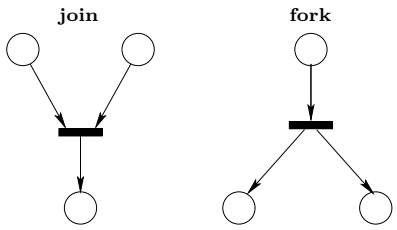

Fig. 1. Building blocks: *join* and *fork*.

Definition 1. *Let $x, y \in \{join, fork\}$ be building blocks and let $\bar{t}_x$ and $\hat{t}_y$ be the transitions present in x and y respectively.*

We say that y comes after x *(or* x is followed by y*, or* x comes before y *or* x and y are in sequence*) if $\bar{t}_x^{\bullet} \cap {}^{\bullet}\hat{t}_y \neq \emptyset$ and ${}^{\bullet}\bar{t}_x \cap {}^{\bullet}\hat{t}_y = \emptyset$. We say that* x and y are in parallel *if ${}^{\bullet}\bar{t}_x \cap {}^{\bullet}\hat{t}_y \neq \emptyset$ and $\bar{t}_x^{\bullet} \cap {}^{\bullet}\hat{t}_y = \emptyset$.*

We say that a net is composed of *building blocks (it is* composed of x*) if it can be defined by building blocks (it is defined by* x*) sharing places but not transitions. So, for instance, to say that a net is* composed of joins *means that the only building blocks present in the net are* join*.*

3 The Simulator and Its Complexity

A computer program (in the following called *simulator*) able to create and run EN systems composed of *join* and *fork* has been written and it can be downloaded from `http://www.macs.hw.ac.uk/~pier/download.html`.

In the following j denotes the number of *join*, f denotes the number of *forks* and p denotes the number of places in a Petri net.

The maximum number of *join* (or *fork*) that can be present in a Petri net with p places is $\frac{p(p-1)(p-2)}{2}$. Moreover, when the Petri net is connected, $j + f \geq \frac{p}{2}$.

Given the number of places, the number of Petri nets with j *join* and f *forks* is

$$\frac{(\frac{p(p-1)(p-2)}{2})!}{j(\frac{p(p-1)(p-2)}{2} - j)!} * \frac{(\frac{p(p-1)(p-2)}{2})!}{f(\frac{p(p-1)(p-2)}{2} - f)!}$$

This number is definitely high even if one considers that it includes isomorphic nets. Due to this high number, we could only generate and run nets with 5 places. This means that the number of *join* and *fork* in these nets ranged from 1 to 30.

For each different triple of p, j and f, only 1% of the possible nets has been created and run for all its possible initial configurations.

The simulator created these nets in a random way.

The set of all possible configurations of such a net is called *configuration space*. The dynamics of an EN system is such that it will start from its initial configuration and it will reach an *attractor*. With *attractor* we define both a configuration from which no firing is possible or a set of configurations that are cyclically repeated. We name the configurations in the following way: With *isolated configuration* we refer to a configuration which is not reachable from any configuration and from which no transition is possible; with *final configuration* we refer to a configuration from which no transition is possible. Clearly, any isolated configuration is also a final configuration but a configuration can be final but not isolated.

The tests run on a computer with a single CPU of 2.4 GHz and with 1.5 GB of 800 MHz RAM. The simulation took 70 hours and the output files occupy 5 GB.

In Figure 2 a net and its configuration spaces are depicted. In this figure the configuration with no tokens is not shown (and in the following we do not consider this configuration). Each configuration in the configuration space is represented as

a number in a circle. The number encodes the configuration of the EN system (as a conversion from binary to decimal). For instance, the encoding of configuration $\{0, 1, 0, 1, 1\}$ (nodes 0 and 2 have no token while the remaining nodes have 1 token) in the net depicted in Figure 2 is 11 (place 0 is the leftmost and place 4 is the rightmost).

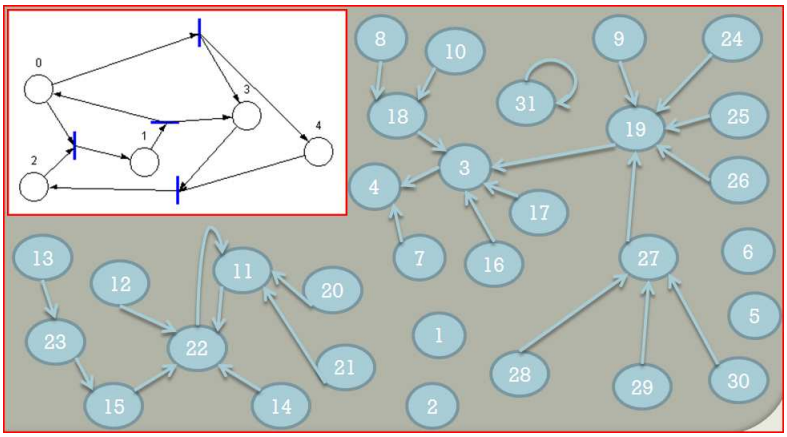

Fig. 2. A net and its configuration space

The attractors in Figure 2 are configurations 1, 2, 4, 5, 6, (11, 22) and 31, where (11, 22) define a *cycle* (i.e., an attractor with more than one configuration) in the configuration space. The configurations 1, 2, 5 and 6 are isolated (and final). Configuration 31 is not isolated as a transition (to itself) indeed is possible. Configurations 1, 2, 4, 5 and 6 are final.

4 Results

We addressed several questions during our study. The plotted answers to these questions and brief comments are present in the following. The tables used to generate the plots can be downloaded from
`http://www.macs.hw.ac.uk/~pier/cvPublications.html`

How does the probability to have at least one cycle in the configuration space depend on the number of join and fork?

We found that *join* and **fork** equally influence the presence of cycles in the configuration space. The plot in Figure 3 shows that if the number of *fork* is bigger than 9 or the number of *join* is bigger than 19, then it is certain that the configuration space contains at least one cycle.

How does the number of not-isolated configurations depend on the number of join and fork?

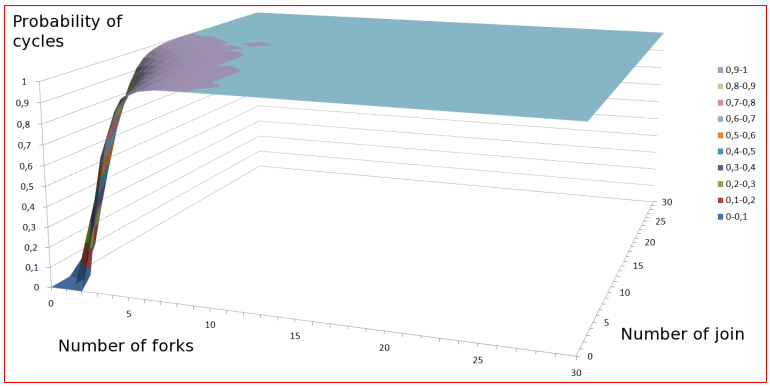

Fig. 3. Probability to have at least one cycle in the configuration space as a function of the number of *join* and *fork*

We found that the number of **fork** have a strong influence on the number of not-isolated configurations decreasing them to 0 with only 5 *forks* are present in the net. Also the increase of *join* tend to decrease the number of not-isolated configurations, but not with a marked effect as the number of *forks*. This is clearly shown by the plot in Figure 4.

Interestingly, the maximum number of not-isolated places is reached in nets with only 2 *joins*.

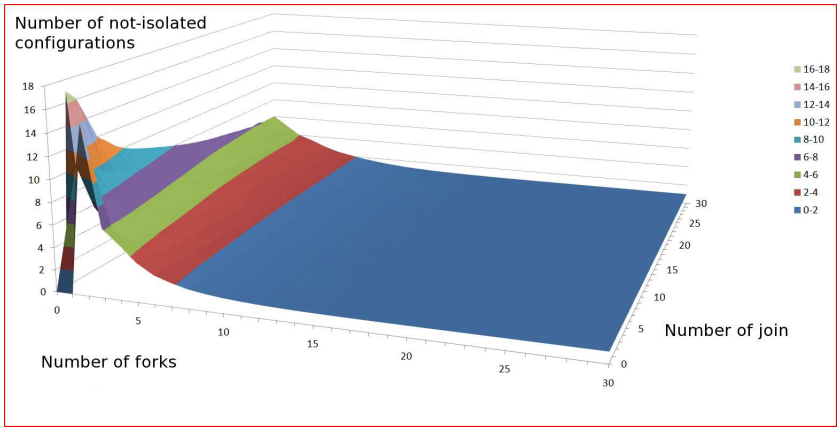

Fig. 4. Number of not-isolated configurations as a function of the number of *join* and *fork*

What is the minimum number of cycles present in the state space of Petri nets as a function of join and fork?

Also in this case the influence of *join* and *fork* is asymmetrical: a net can have up to 18 *join* and still no cycle is present in the state space of the Petri net. Differently, if the net has at least 11 *fork*, then the state space of the Petri net has at least 1 cycle. This is shown by the plot in Figure 5.

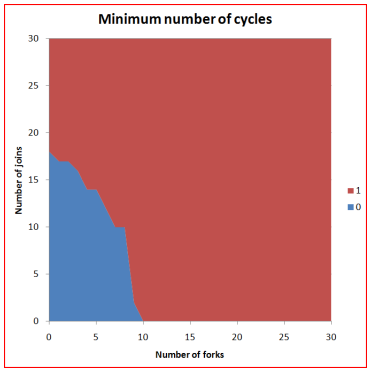

Fig. 5. Probability of the Petri net to end up in a cycle as a function of the number of *join* and *fork*

How many initial configurations will end up in a cycle depending on the number of join and fork?

The fact that the state space contains at least a cycle does not imply that all initial configurations will end up in in a cycle. It is confirmed that for high numbers of *join* and *fork* the Petri net will certainly enter a cycle. This is shown by the plot in Figure 6.

5 Final Remarks

A Petri net whose attractors are all final configurations can only generate or accept finite languages. Our study proved that the presence of only *join* or many *join* and a few *forks* let Petri nets have only final configurations as attractor. This result is rather immediate: a *join* consumes tokens, so if a net has only join, then sooner or later it will run out of tokens.

Thinks become more interesting when attractors with more than one place are present. In this case the set of languages generated or accepted is infinite. The kind of languages depends on the number of these attractors and their topology. We did not study this.

As said in Section 1, these results can be easily translated to formal models of computation operating with multisets of objects. Unfortunately, the firing strategy adopted by us, does not find a counterpart in any such model. This is, for instance, due to the fact that we allow one single token to be used in the firing of more than

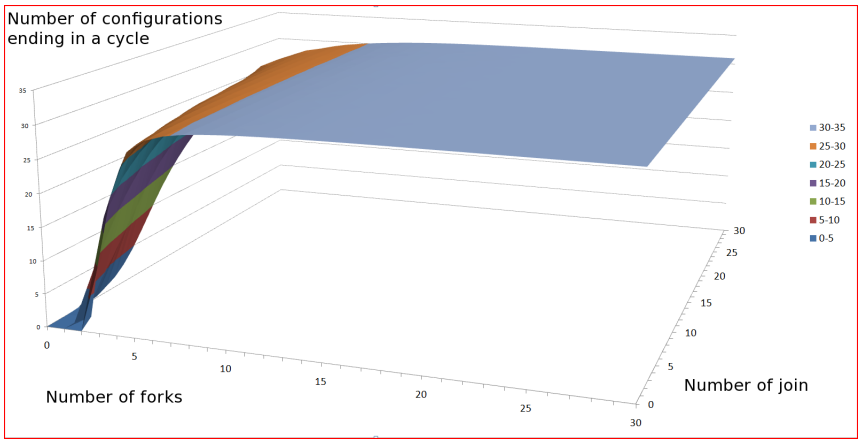

Fig. 6. Number of configurations that end up in a cycle as a function of the number of *join* and *fork*

one transition. The translation of this feature in, for instance, P systems, means that one single occurrence of an object can be used in the same configuration by different rules.

For this reason one of our future direction of research will be to implement firing strategies closer to the operational modes of existing formal models of computation.

Acknowledgment

The work of Miguel A. Peña was supported by a grant for short stays in Spain and abroad for the beneficiaries of official pre-doctoral programs for training researchers, year 2009, from the Technical University of Madrid.

References

1. G. Păun, G. Rozenberg, and A. Salomaa, editors. *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
2. B. Drossel. Random boolean networks. http://arxiv.org/pdf/0706.3351.
3. P. Frisco. P systems and topology: some suggestions for research. Seventh Brainstorming Week on Membrane Computing, 2009, pp. 123-132, Universidad de Sevilla, Technical report num. 1/2009, volume 1.
4. P. Frisco. P systems, Petri nets, and Program machines. In R. Freund, G. Lojka, M. Oswald, and G. Păun, editors, *Membrane Computing. 6^{th} International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised Selected and Invited Papers*, volume 3850 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, Berlin, Heidelberg, New York, 2006.

5. P. Frisco. *Computing with Cells. Advances in Membrane Computing.* Oxford University Press, 2009.
6. P. Frisco. Conformon P systems and topology of information flow. In G. Păun, editor, *Membrane Computing. 10^{th} International Workshop, WMC 2009, Curtea de Arges, Romania, August 24-27, 2009, Revised Selected and Invited Papers*, volume 5957 of *Lecture Notes in Computer Science*, pages 30–53. Springer-Verlag, Berlin, Heidelberg, New York, 2009.
7. Z. Qi, J. You, and H. Mao. P systems and petri nets. volume 2933 of *Lecture Notes in Computer Science*, pages 286–303. Springer-Verlag, Berlin, Heidelberg, New York, 2004.
8. G. Rozenberg and J. Engelfriet. *Elementary net systems*, volume 1491 of *Lecture Notes in Computer Science*, pages 12–121. Springer-Verlag, Berlin, Heidelberg, New York, 1998.