STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 3, 2011

DP AUTOMATA AND PETRI NETS

MONICA KERE

Abstract. At the first view there are no similarities between P systems and Petri nets, but this paper describes a connection between them. Petri nets can describe a very large domain of systems. P systems represent a computational model that describe the interaction between chemical processes and their membranes. The dP automata are a class of membrane systems that tries to combine a number of P systems and makes them to communicate. The behavior of such an automaton is made in an non-deterministically maximally parallel way. The objects are moving inside each P system and also between the P systems from the automaton.

This paper will use on the modeling process a place transitions network (PTN). Graphical symbols from PTN will be used to describe the behavior of a dP automaton(dPA). The properties from the PTN will be define for the dP automata providing that an automaton is having the same expressing power as a Petri net.

1. Introduction

The domain of P systems and membrane computing was very much investigated from the beginning. A class of P systems, called dP automata, was introduced in the recent paper [6]. This automaton tries to combine n P systems and makes them to communicate. The P systems from inside a dPA are called components and the rules through which they communicate are called communicating rules. Each P system behaves according to its rules. Those rules are symport/antiport rules. All the P systems from the dPA are interchanging objects from their skin membranes, using the communicating rules, that are also symport/antiport rules. So a dPA will have the rules that are applied for each P system and the rules applied for all the P systems. All the rules are applied in a non-deterministic parallel manner. The environment

Received by the editors: April 10, 2011.

2010 *Mathematics Subject Classification.* 68Q45, 68Q60.

1998 *CR Categories and Descriptors.* D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Petri nets*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation – *Alternation and nondeterminism.*

Key words and phrases. dP automata, Petri nets, P systems.

is common for all the P systems and it contains objects available in arbitrarily many copies.

The formalization of the operations of the symport/antiport rules, as in [5], can be realized as:

- *(ab, in)* or *(ab, out)* are symport rules, which means that a and b pass together through a membrane entering in the former case and exiting in the latter. This rule doesn't allow a or b to pass separately in/out the membrane. For this case it is necessarily another rule like *(a, in)* or *(a,out)* named uniport.

- *(a, in; b, out)* represents an antiport rule, which means that a enters and at the same time b exists the membrane.

Each component can take an input, work on it, communicate with other components, and provide the answer to the problem in the end of a halting computation. For a dPA a configuration is represented through a vector with n components $C = \{L_1, ..., L_n\}$, the component i is the multiset of objects from the P system $i, i = 1, .., n$. A halting computation will accept the string $x_1...x_n$ over O and starting from the initial configuration and applying the internal rules, but also the communicating rules in a non-deterministically maximally parallel manner, take from the environment the substrings $x_1, ..., x_n$, respectively, and eventually halts.

2. dP Automata Properties

To provide some connections between Petri nets(PNs) and dP automata we represent such an automaton using a Place Transition Petri net. The objects will be represented as places with tokens corresponding to the number of copies and the rules will be drawn as transitions. The places are drawn as cycles, the transitions as squares and the connections between them by arrows. The example below shows how a dPA can be modeled using a PTN.

Below we analyze the behavioral properties of a dPA modeled by a Petri net. Important properties for the PTN are described in [4]. These properties can be investigated also for the dP automata. The description of the behavioral properties like *terminating, deadlock-free, boundedness* and *liveness* for a P system is done in [7]. Because a dPA is also a P system, and because a P system modeled with a PTN is still a Petri net, we can introduce these properties as follows below.

Definition 1: For a given dP automaton, we define the following properties:

i) A dPA is *terminating* if the sequence of transitions between configurations is finite.

ii) A configuration C_n in a dPA is said to be *reachable* if there exists a sequence of transactions that transforms C_0 in C_n.

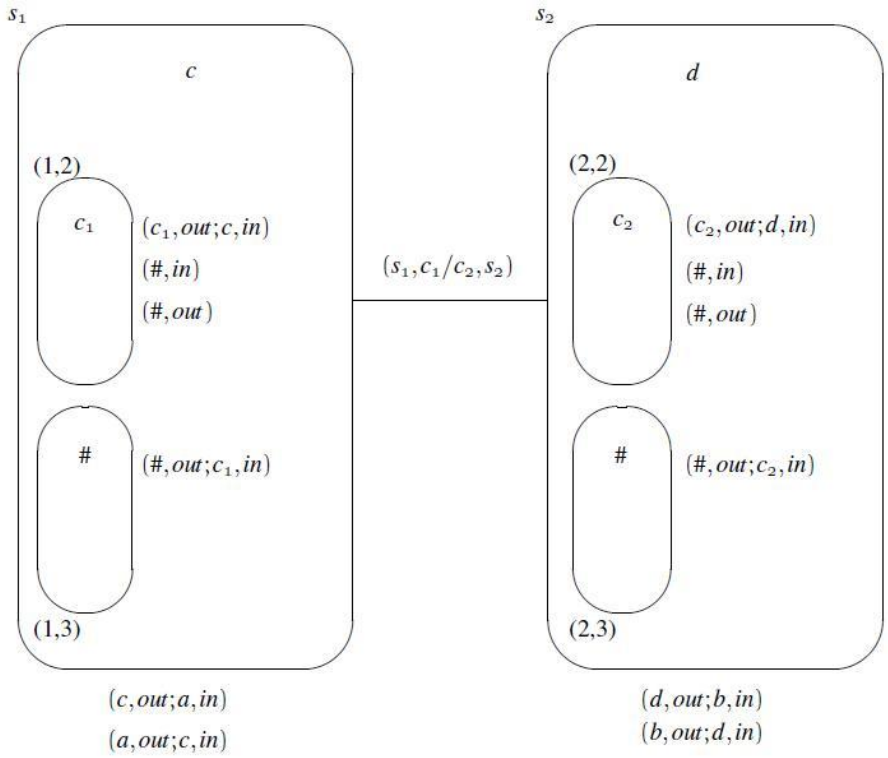

FIGURE 1. The dP automaton [1]

iii) *Boundedness* can be seen in two ways: local and global. *Local boundedness* suppose that the number of copies of each object in each region to be less than a given integer k. For a given integer k, the dPA is said to be *global bounded* if the number of copies of each object in each component is less than k for every configuration reachable from C_0. A dPA is said to be safe if $k = 1$.

iv) A rule is said to be dead in the configuration C if it cannot be executed in any configuration reachable from C. A rule is said to be live if it is not dead in any configuration reachable from C_0. A configuration C is dead if there is no rule which can be executed in C. A dPA is said to be *deadlock-free* if there are no reachable dead configuration. A dPA is *living* if each rule is living.

v) A dPA is said to be *reversible* if for each configuration reachable from C_0 we can reach again C_0.

vi) Giving a configuration C minimum needed to apply a rule r, then r is potentially realized (r can be applied at least one time in some sequences of executions) if and only if C is *coverable*.

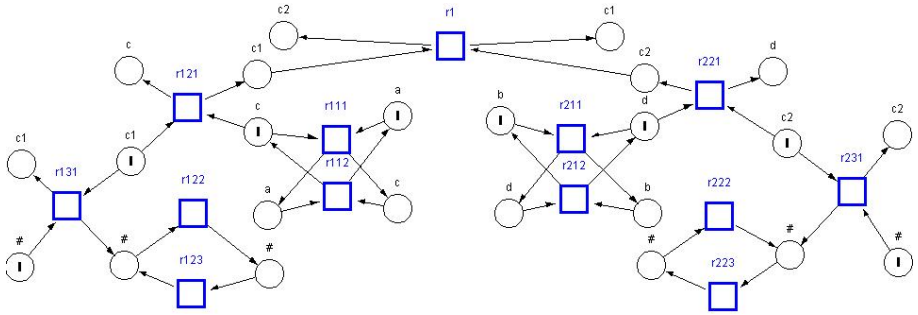

FIGURE 2. The representation using Petri nets for the dP automaton from Fig. 1

vii) A dPA is said to be *persistent* if, for any two rules that can be realized, the execution of one will not disable the other one.

Because a dPA modeled with a PTN is also a PTN and because those properties are given for the PTN and using the definitions from Def. 1 we get the following theorems:

Theorem 1 If the PTN for a given dPA terminates, then the dPA terminates.

Proof: If the dPA doesn't terminate, then there exists an infinite sequence. When the dPA is modeled using the PN, then there also exists an infinite sequence. Every sequence consists of some transaction-steps and send transactions and each of this is a one-to-one mapped to a transition in the PN. So the sequence of transitions in the PN is not finite. Thus the PN doesn't terminate.

Theorem 2 If the configurations in the PTN for a given dPA are reachable, then the configurations in the dPA are reachable.

Theorem 3 If the PTN for a given dPA is bounded, then the dPA is bounded.

Theorem 4 If the PTN for a given dPA has liveness, then the dPA has liveness.

Theorem 5 If the PTN for a given dPA is reversible, then the dPA is reversible.

Theorem 6 If the configurations in the PTN for a given dPA are coverable, then the configurations in the dPA are coverable.

Theorem 7 If the PTN for a given dPA is persistent, then the dPA is persistent.

The proofs for the Theorem 2, 3, 4, 5, 6, 7 are the same like for the Theorem 1.

To define some properties, we classify the dP automata as described below.

Definition 2: Subclasses of dP automata:

1) A dPA is said to be a *state machine* (SM) if in each rule there is exactly one object going out and exactly one object coming inside a region.

2) A dPA is said to be a *marked graph* (MG) if each object in each region is introduced by only one rule and is taken out through exactly one rule.

3) A dPA is said to be a *free-choice* (FC) if every object can either go out or can come inside the region through a unique rule.

4) A dPA is said to be an *extended free-choice* if two objects that are going out from a region through some common rules then all their rules for going out are common.

5) A dPA is said to be an *asymmetric choice* (AC) if two objects that have some rules for going out in common, then one of them has all the rules for going out of the other (and possibly more).

A dPA is said to be *ordinary* if the number of copies of any object in any rule is 1.

A nonempty subset of objects S in an ordinary dPA N is called a *siphon* if every rule that is having an object from S going out from a region is having an object from S entering in a region. If S is not contained under some configuration (doesn't has any copy of its objects), it remains like that under its successors.

A nonempty subset of objects Q in an ordinary dPA N is called a *trap* if every rule that is having an object from Q going inside a region is having an object from Q coming out a region. If Q is contained under some configuration (it has copies of its objects), it remains contained under its successors.

Definition 3: For a SM we define the following properties:

1) A SM is *living* iff is strongly connected and in the initial configuration contains at least one copy of an object.

2) A SM is *safe* iff the initial configuration has at most one copy of an object.

Definition 4: For a MG we define the following properties:

1) A MG is *living* iff in the initial configuration there is at least one copy of an object for each circuit.

2) A live MG is *safe* iff in the initial configuration there is exactly one copy of at least one object from each circuit.

Those properties have been defined for dP automata and they inherit them from Petri nets which describe the automata. Petri nets were more investigated than dP automata so we tried here to make the connection between Petri

nets and P systems stronger and to add more features for the new domain of automata.

3. Conclusions

P systems and Petri nets look like having nothing in common, but there are many things that make a connection between them. Some other authors in their papers, [2], [3], [7] emphasized this connection before. This paper is trying to make stronger the connection between P systems and Petri nets. A class of P systems, called dP automata, is used to be modeled using a Place Transition Network. Petri nets is a tool that can describe graphically distributed, concurrent systems. Graphical symbols from Petri nets are used to describe the dPA. Then some properties from PN are described for dPA providing this connection.

There is a lot of work to be done in this domain. Some other properties can be defined such as rechability criteria, coverability criteria. The analyze can be done using the coverability tree so that the properties can be checked just looking at the loops and dead-ends of this tree. Model Checking can be used to verify the properties defined here for a dPA. So a tool can be build for checking if an automaton can provide a property or not.

References

[1] Freund, R., Kogler, M., Păun, G., Pérez-Jimenéz, M.: *On the power of P and dP Automata*, Annals of Bucharest Univ. Mathematical-Informatics Series, 2010, in press.

[2] Frisco, F.: *P Systems, Petri Nets, and Program Machines*. In: LNCS 3850, pp. 209-223, 2006.

[3] Kleijn, J. and Koutny, M.: *Synchrony and Asynchrony in Membrane Systems*. In: LNCS 4361, pp. 66-85, 2006.

[4] Murata, T.: *Petri Nets: Properties, Analysis and Applications*, vol 77, pp. 541 - 580, IEEE Computer Society (1989)

[5] Păun, A., Păun, Gh.: *The power of communication: P systems with symport/ antiport*. New Generation Computing, 20 (2002), 295-305

[6] Păun, Gh., Pérez-Jimnéz, M.J.: *Solving problems in a distributed way in membrane computing: dP systems*. Int. J. of Computers, Communication and Control, 5, 2 (2010), 238-252.

[7] Zhengwei Qi, Jinyuan You, Hongyan Mao: *P Systems and Petri Nets*. In: LNCS, vol. 2933, pp. 819-847. Springer, Heidelberg (2004)

University of Piteşti, Faculty of Mathematics and Computer Science, Department of Computer Science,, 110040 Piteşti, Str. Târgu din Vale, No.1, Argeş, Romania

E-mail address: monicakere@yahoo.com