Tools for P system testing

Alexandru Ciobanu[1], Raluca Lefticaru[1],
Ionuţ Mihai Niculescu[1], Florentin Ipate[1]

Department of Computer Science, University of Pitesti
Str. Targu din Vale 1, 110040 Pitesti, Romania
`florentin.ipate@ifsoft.ro`

Abstract. This paper presents some approaches on testing P systems that allow building specific testing tools. The two approaches used for automatic test case generation are based on model-checking and building the derivation tree, respectively. The classes of P systems addressed so far are transitional, with evolution-communication rules, and probabilistic P systems, with electrical charges.

Keywords: P systems testing, model checking, derivation tree, NuSMV, P-Lingua.

1 Introduction

The field of *membrane computing*, which deals with distributed and parallel computing models called *P systems*, has been a rapidly growing research area in the last ten years. Initially coined by Gheorghe Păun in [11], P systems have been intensively studied from a computational perspective, many variants being introduced and investigated and a substantial set of applications have been identified and modelled with such systems [12]. A special volume has been dedicated to applications of P systems [3] and more recently models of highly complex ecosystem phenomena have been provided [1].

Although there are promising applications and simulators developed for many variants of P systems, there are no tools available for testing such models. Several attempts have been made to use available model checking methods for verifying stochastic P system [13] or testing basic classes of P systems [6], but no generic tool has as of yet surfaced. In this paper we present several testing principles underpinning the creation of such tools. As P-Lingua [4, 10] is increasingly becoming more popular and it is extending to accept various types of P systems, we will use it for specifying and simulating P systems.

2 Previous work

A model checking-based approach for P systems testing was developed in [6] and further extended in [7]. In this approach NuSMV [8] was used to automatically produce the P system computation which is covering a certain rule or combination of rules. NuSMV is a symbolic model checker, used to operationally specify a system and then to verify the correctness of the system using temporal logic formulae [2]. Advantageously NuSMV verifies the entire state space of the model and provides counterexamples if the specified properties are false. This feature is exploited to develop test cases, corresponding to counterexamples, through the following steps:

1. The P system is transformed into a Kripke structure.
2. The Kripke structure is written in NuSMV syntax.
3. The rule coverage criteria are written using temporal logic formulae.
4. These formulae are negated (e.g. "this rule will never be applied").
5. NuSMV is run and the counterexamples provided for the negated formulae are interpreted as test cases.

Sample output of such a process would look as followed:

```
Test case:          s --> a,b --> b,c^2 --> c^3
Rules applied:       {r1}   {r2,r3}    {r4}
```

3 Probabilistic P system testing using NuSMV

In order to create a testing application for P systems, we extended the tool described in [6, 7] to *probabilistic P systems*, first considered by A. Obtulowicz in [9]. The application uses the theory presented previously: it transforms a P system into SMV, the language accepted by NuSMV, and runs *Linear Temporal Logic* (LTL) specifications against it.

The application adds LTL specifications of a second form, to check certain properties of the P system, for example if an object of the multiset will ever reach n units, or will it ever be completely exhausted. The tool also expands the capabilities of the system mentioned above by adding support for multi membrane with different permeability properties, rules with probabilities, and membrane polarization.

An important aspect is representing probabilistic transitions in SMV. Consider a rule r and c_r the probability associated with it. Let n_r be the number of times rule r is applied in a transition and N the number of times all rules, of the same membrane as r and having the same left hand side as r, are applied in the same transition. Here we introduce a probability margin ε and we require that:

$$c_r - \varepsilon \leq \frac{n_r}{N} \leq c_r + \varepsilon.$$

We gate the above by the requirement $N \geq m$, for m sufficiently large depending on the application, to eliminate impossible scenarios.

4 Derivation tree based testing for P systems

Another approach to P systems testing involves the creation of a derivation tree for the system and traversing the tree to read possible execution paths [5]. This is achieved by implementing an extension of the P-Lingua framework [10]. The tree root has the initial configuration of the P system and the children nodes contain all subsequent configuration of the system. The children are obtained by applying combination of rules in a maximal parallel mode.

At building the derivation tree, the rules probabilities are ignored, and all possible combinations of rules are stored. Each node of the derivation tree stores the multiset of the particular configuration, as well as the rules applied to acquire that particular multiset. A constant k is used to define the maximum number of tree levels. Once this constant is reached, the application stops building new nodes. This is particularly useful in non terminating P system to ensure the application halts.

Taking as an example the probabilistic P system: $\Pi = (V, \mu, w_1, w_2, R_1, R_2)$, $V = \{a, b, c, d, e\}$, $\mu = [_1[_2]_2]_1$, $w_1 = \lambda$, $w_2 = a^4$, $R_1 = \{r_1 : [b]_1^0 \xrightarrow{1} [a]_1^0\}$, $R_2 = \{r_1 : [a^2]_2^0 \xrightarrow{0,4} b[c]_2^0;\ r_2 : [a]_2^0 \xrightarrow{0,4} [d]_2^0;\ r_3 : [a]_2^0 \xrightarrow{0,2} e[]_2^0\}$, a part of the derivation tree is given in Fig. 1.

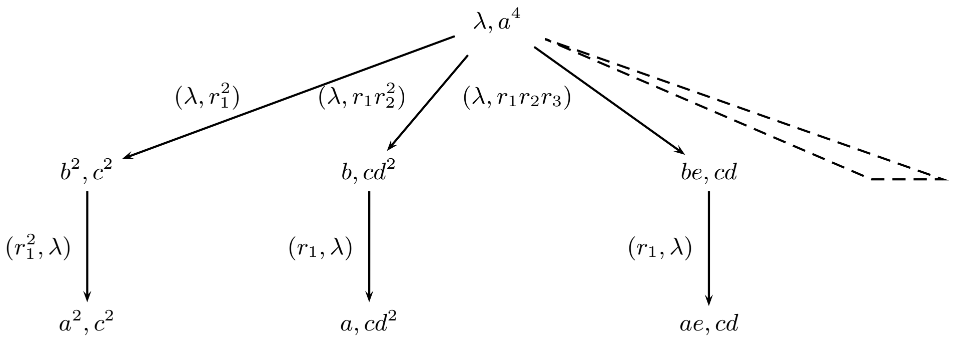

Fig. 1. Derivation tree for Π and $k = 2$

On the derivation tree, depth first searches are used to find paths which (when executed) would satisfy different rule coverage criteria. All partial paths which cover new rules are stored. The search yields, as in the first application, a path through the P system, as well as the oracle for the particular path. More details regarding the testing tools and some case studies are available at `http://fmi.upit.ro/evomt/psys/psys_tools.html`.

5 Conclusions and future work

The tools developed work well on small and medium size P systems (for which the state space of the corresponding NuSMV models, as described in [6, 7] have the state space in the order of 10^{12}). Unfortunately, as the complexity of the P system expands, the models become extremely complex and the testing utilities face a state explosion problem. Further work needs to be put forth in an attempt to reduce the memory burden put on the modeling tools to allow more complex P systems to be tested. Research is also ongoing into other simulation and model checking tools that might have better tolerance to very large data sets and aid the test generation.

Acknowledgments. This work was supported by CNCSIS - UEFISCSU, project number PNII - IDEI 643/2008.

References

1. Cardona, M., Colomer, M.A., Margalida, A., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Sanuy, D.: A P system based model of an ecosystem of some scavenger birds. In: Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing - 10th International Workshop, WMC 2009, Revised Selected and Invited Papers. LNCS, vol. 5957, pp. 182–195. Springer (2010)
2. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: NuSMV: A new symbolic model checker. International Journal on Software Tools for Technology Transfer 2(4), 410–425 (2000)
3. Ciobanu, G., Păun, G., Pérez-Jiménez, M.J. (eds.): Applications of Membrane Computing. Natural Computing Series, Springer (2006)
4. García-Quismondo, M., Gutiérrez-Escudero, R., Martínez-del-Amor, M.A., Orejuela-Pinedo, E., Pérez-Hurtado, I.: P-Lingua 2.0: A software framework for cell-like P systems. International Journal of Computers, Comunication and Control IV(3), 234–243 (2009)
5. Gheorghe, M., Ipate, F.: On testing P systems. In: Corne, D.W., Frisco, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing - 9th International Workshop, WMC 2008, Revised Selected and Invited Papers. LNCS, vol. 5391, pp. 204–216. Springer (2009)
6. Ipate, F., Gheorghe, M., Lefticaru, R.: Test generation from P systems using model checking. Journal of Logic and Algebraic Programming. DOI:10.1016/j.jlap.2010.03.007, in press (2010)
7. Lefticaru, R., Ipate, F., Gheorghe, M.: Model checking based test generation from P systems using P-Lingua. In: Brainstorming Week on Membrane Computing (2010, in press)
8. NuSMV Web page : `http://nusmv.irst.itc.it/` (last visited, June 2010)
9. Obtulowicz, A.: Probabilistic P systems. In: Păun, G., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing - International Workshop, WMC 2002, Revised Papers. LNCS, vol. 2597, pp. 377–387. Springer (2003)
10. P-Lingua Web page : `http://www.p-lingua.org` (last visited, June 2010)
11. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
12. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
13. Romero-Campero, F.J., Gheorghe, M., Bianco, L., Pescini, D., Pérez-Jiménez, M.J., Ceterchi, R.: Towards probabilistic model checking on P systems using PRISM. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing - 7th International Workshop, WMC 2006, Revised, Selected, and Invited Papers. LNCS, vol. 4361, pp. 477–495. Springer (2006)