Timed Transition System Models for Programming Physarum Machines
Extended Abstract

Andrew Schumann[1] and Krzysztof Pancerz[1,2]

[1] University of Information Technology and Management
Sucharskiego Str. 2, 35-225 Rzeszów, Poland
aschumann@wsiz.rzeszow.pl
[2] University of Management and Administration
Akademicka Str. 4, 22-400 Zamość, Poland
kpancerz@wszia.edu.pl

Abstract. In the paper, we show that timed transition system models can be used as a high-level model of behavior of Physarum machines. A Physarum machine is a programmable amorphous biological computer experimentally implemented in the vegetative state of Physarum polycephalum. Timed transition system models have been used in our new object-oriented programming language for Physarum polycephalum computing.

Keywords: Physarum polycephalum, unconventional computing, object-oriented programming language, timed transition systems.

1 Timed Transition System Models

A Physarum machine is a programmable amorphous biological computer, experimentally implemented in the vegetative state of Physarum polycephalum (also called slime mould) [2] that is a one-cell organism belonging to the species of order *Physarales*, subclass *Myxogastromycetidae*, class *Myxomycetes*, and division *Myxostelida*. The plasmodium of Physarum polycephalum spread by networks can be programmable. The ability to perform useful computational tasks, in propagating and foraging behaviour of the plasmodium, was firstly emphasized by T. Nakagaki et al. (cf. [7]). In *Physarum Chip Project: Growing Computers from Slime Mould* [3] funded by the Seventh Framework Programme (FP7), we are going to construct an unconventional computer on programmable behavior of Physarum polycephalum. The Physarum machine comprises an amorphous yellowish mass with networks of protoplasmic veins, programmed by spatial configurations of attracting and repelling stimuli. The plasmodium looks for attractants, propagates protoplasmic veins towards them, feeds on them and goes on. As a result, a transition system is built up. Therefore, Physarum motions can be treated as a kind of natural transition systems with states presented by attractants and events presented by plasmodium transitions between attractants

[2]. We can define the following three basic forms of Physarum transitions (motions): *direct* (direction: a movement from one place, where the plasmodium is located, towards another place, where there is a neighbouring attractant), $fuse$ (fusion of two plasmodia at the place, where they meet the same attractant), *split* (splitting plasmodium from one active place into two active places, where two neighbouring attractants with a similar power of intensity are located).

Formally, a transition system is a quadruple $TS = (S, E, T, s_0)$ [8], where S is the non-empty set of states, E is the set of events, $T \subseteq S \times E \times S$ is the transition relation, s_0 is the initial state. Usually transition systems are based on actions which may be viewed as labelled events. If $(s, e, s') \in T$ then the idea is that TS can go from s to s' as a result of the event e occurring at s. A transition system can be presented as a graph structure with nodes corresponding to states and edges corresponding to transitions.

To program computation tasks for amorphous biological computers, we are developing a new object-oriented programming language (cf. [10]). The proposed language can be used for developing programs for Physarum polycephalum by the spatial configuration of stimuli. This configuration of stimuli can be identified with a low-level programming language for Physarum machines. To program behavior of Physarum polycephalum, we have also proposed to use some high-level models, e.g., ladder diagrams, Petri nets, and transition systems [9].

We can consider some other operations (instructions) in Physarum machines like: *add node*, *remove node*, *add edge*, *remove edge* [2]. Adding and removing nodes can be implemented through activation and deactivation of attractants, respectively. Adding and removing edges can be implemented by means of repellents put in proper places in the space. An activated repellent can avoid a plasmodium transition between attractants. Adding and removing edges can change dynamically over time. To model such behavior, we propose to add another high-level model, based on timed transition systems [4], to our language. It is assumed, in transition systems mentioned earlier, that all events happen instantaneously. In timed transition systems, timing constraints restrict the times at which events may occur. The timing constraints are classified into two categories: lower-bound and upper-bound requirements.

Let N be a set of nonnegative integers. Formally, a timed transition system $TTS = (S, E, T, s_0, l, u)$ consists of an underlying transition system $TS = (S, E, T, s_0)$ as well as a minimal delay function (a lower bound) $l : E \to N$ assigning a nonnegative integer to each event and a maximal delay function (an upper bound) $u : E \to N \cup \{\infty\}$ assigning a nonnegative integer or infinity to each event.

Let us consider a simple timed transition system shown as a graph structure in Figure 1 with the following timing constraints: $l(e_1) = 0$, $u(e_1) = \infty$, $l(e_2) = 0$, $u(e_2) = \infty$, $l(e_3) = 5$, $u(e_3) = 10$. The code in our created language has the following form:

```
#TRANSITION_SYSTEM
s1=new TS.State("s1");
s1.setAsInitial;
```

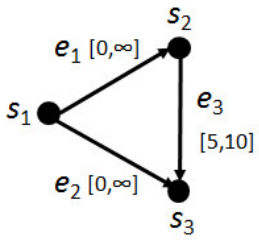

Fig. 1. A timed transition system.

```
s2=new TS.State("s2");
s3=new TS.State("s3");
e1=new TS.Event("e1");
t1=new TS.Transition(s1,e1,s2);
e2=new TS.Event("e2");
t2=new TS.Transition(s1,e2,s3);
e3=new TS.Event("e3");
e3.setTimingConstraints(5,10);
t3=new TS.Transition(s2,e3,s3);
```

The default timing constraints are 0 as a lower bound and ∞ as an upper bound.

As a result of programming the Physarum machine, we obtain spatial configurations of stimuli presented in Figure 2, (a) for the time instant $\tau = 4$, (b) for the time instant $\tau = 8$, where P is Physarum, A_{s_1}, A_{s_2}, A_{s_3} are attractants, and R is a repellent. It is easy to see that the event e_3 is allowed only if actual time $t \in \{5, 6, \ldots, 10\}$. Therefore, in the model in Figure 2 (a), a repellent, avoiding the transition between states s_2 and s_3 as a result of the event e_3, is present, i.e., it is activated.

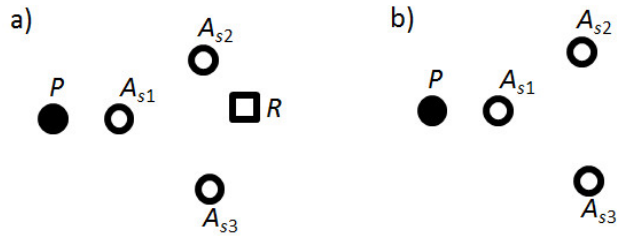

Fig. 2. Spatial configurations of stimuli for the Physarum machine.

In a real-life implementation of Physarum machines, attractants are sources of nutrients or pheromones, on which the plasmodium feeds whereas in case of repellents the fact that plasmodium of Physarum avoids light and some thermo-

and salt-based conditions is used. Repellents can be activated or deactivated for proper time periods, especially in case of light.

Modelling dynamically changed structures over time is important in some cases of abstract machines used in non-classical computations. One of them is a Kolmogorov-Uspensky machine. In [1], there was shown that the plasmodium of Physarum polycephalum is a biological substrate that implements a Kolmogorov-Uspensky machine (at the beginning called a Kolmogorov complex) - a concept of an abstract machine defined on a dynamically changing graph-based structure outlined by A.N. Kolmogorov and V.A. Uspensky [5], [6]. The Kolmogorov-Uspensky machine is a process on a finite undirected connected graph with distinctly labelled nodes. A computational process travels on the graph, activates nodes and removes and adds edges. There is only one active node at any step of development.

Acknowledgments

This research is being fulfilled by the support of FP7-ICT-2011-8.

References

1. Adamatzky, A.: Physarum machine: Implementation of a Kolmogorov-Uspensky machine on a biological substrate. Parallel Processing Letters 17(4), 455–467 (2007)
2. Adamatzky, A.: Physarum Machines: Computers from Slime Mould. World Scientific (2010)
3. Adamatzky, A., Erokhin, V., Grube, M., Schubert, T., Schumann, A.: Physarum Chip Project: Growing computers from slime mould. International Journal of Unconventional Computing 8(4), 319–323 (2012)
4. Henzinger, T., Manna, Z., Pnueli, A.: Timed transition systems. In: de Bakker, J., Huizing, C., de Roever, W., Rozenberg, G. (eds.) Real-Time: Theory in Practice, Lecture Notes in Computer Science, vol. 600, pp. 226–251. Springer Berlin Heidelberg (1992)
5. Kolmogorov, A., Uspensky, V.: On the definition of an algorithm. Uspekhi Mat. Nauk 13(4), 3–28 (1958), in Russian
6. Kolmogorov, A.N.: On the concept of algorithm. Uspekhi Mat. Nauk 8(4), 175–176 (1953), in Russian
7. Nakagaki, T., Yamada, H., Toth, A.: Maze-solving by an amoeboid organism. Nature 407, 470–470 (2000)
8. Nielsen, M., Rozenberg, G., Thiagarajan, P.: Elementary transition systems. Theoretical Computer Science 96(1), 3–33 (1992)
9. Pancerz, K., Schumann, A.: Principles of an object-oriented programming language for Physarum polycephalum computing. In: Proceedings of the 10th International Conference on Digital Technologies (DT'2014). pp. 273–280. Zilina, Slovak Republic (2014)
10. Schumann, A., Pancerz, K.: Towards an object-oriented programming language for Physarum polycephalum computing. In: Szczuka, M., Czaja, L., Kacprzak, M. (eds.) Proceedings of the Workshop on Concurrency, Specification and Programming (CS&P'2013). pp. 389–397. Warsaw, Poland (2013)