Agent modeling of the distributed dynamic systems on the earth's surface with using the colored and nested Petri nets

Georgy Dorrer*

Reshetnev Siberian State University of Science and Technology, 31 prospect "Krasnoyarsky rabochy" Krasnoyarsk, Russia

Abstract. The application of formalisms of multi-agent systems, color and nested Petri nets to digital modeling and control of dynamic processes on the Earth's surface is considered. A GIS-based multi-agent system has been implemented in the form of a two-level color nested Petri net. The system is currently accepted for use as a training simulator.

1 Introduction

The ongoing Fourth Industrial Revolution involves the massive introduction of cyber-physical systems in the production and service of human needs. The creation of such systems requires the use of modern information technologies. Among such technologies, an important place is occupied by technologies based on the formalisms of artificial intelligence, multi-agent systems, and various variants of Petri nets [1-3]. In recent years, the rapid development of these technologies has continued. In particular, was solved the following problems:

- the use of agent-based fuzzy Petri nets to represent knowledge, taking into account contradictions [4],
- design and analysis of software architecture models using UML and Petri nets [5],
- control of the Petri neural network for modeling technological processes [6],
- modeling of active antennas and modifications of the architecture of communication facilities [7],
- formalization of document flow processes of technical documentation [8],
- a new paradigm for representing uncertain knowledge using "plausible Petri nets" [9].

Of particular interest is the creation of cyber-physical systems for a wide class of natural processes, called dynamic processes on the Earth's surface [10]. Examples of such processes are wild fires, oil spills, the spread of plant pests, vegetation degradation, and a number of others, which, as a rule, have negative consequences for the environment and the economy.

In this paper, using the formalisms of colored and nested Petri nets, we describe the creation and operation of a multi-agent system designed to support decision-making in the fight against wild fires.

* Corresponding author: g_a_dorrer@mail.ru

2 Agents and multi-agent systems

The general definition of agent is an entity that is in a certain environment, which perceives it with the help of sensors, receives data that reflects the events occurring in the environment, interprets this data and acts on the environment using effectors [1]. From the programmer point of view, an agent is a hardware or software system that performs actions in the environment and also changes the states of other agents in the process of interaction. The main properties of agents are autonomy and purposefulness, that is, autonomous performance of certain actions in a certain environment, receiving a stream of incoming information coming from the environment or other agents, processing this stream and purposeful impact on the environment based on the results of this processing.

To fulfill their mission, software agents are combined into systems that are called multi-agent systems (MAS) [2]. These systems consist of the following components: agents, environments interacting with agents and with each other, and forming an organization that performs individual and joint actions, including communications and evolution.

This paper considers multi-agent systems designed to simulate and control spontaneous distributed dynamic processes on the Earth's surface, such as floods, mudflows, landslides, wild fires, the spread of plant pests, and others [10].

To describe these processes, a digital geoinformation model of the environment is created, and the multi-agent system includes various types of agents that model the following entities:

- a chain of sections of the front of the dynamic process, which together form the contour of the propagating process (agents of type *A*);
- objects that directly influence the process (on agents of type *A*) in order to stop or localize it (agents of type *B*);
- objects that indirectly affect the dynamic process by changing the characteristics of the environment (agents of type *D*);
- agent-manager, managing and coordinating the action of all agents of the system (agent *M*).

3 Using Petri nets for system modeling

One of the most common and effective formalisms for modeling and analyzing complex distributed systems are Petri nets (PN). The basic terminology that defines the structure and functioning of these networks is described below; a detailed description is contained in numerous works, for example, in the monograph [2]. The Petri net is a bipartite oriented multigraph consisting of vertices of two types - positions pi and transitions tj, connected by arcs aij. Vertices of the same type cannot be connected directly. Positions can host resources (tokens) that can move across the network from transitions to positions and from positions to transitions. Depending on the number of tokens types in PN, there are ordinary PN (tokens of only one type) and color CPN (there are many different tokens). The distribution of tokens by position is called network marking. An event is the firing of a transition, in which tokens from the input positions of this transition are moved to the output positions.

The color Petri net CPN, in accordance with the formalism of K. Jensen [4], contains the following elements: a color set (a set of types of resources); a set of positions; a set of transitions; a set of arcs; a color function that determines the types of resources available for each position; expressions on arcs defining the rules for transferring resources along arcs; an initialization function that determines the number of resources in positions at the initial moment. The network operates in discrete time.

3.1 Formal description of agents

Let us consider a graphical representation of Petri nets that describe the functioning of the above types of agents and their interaction.

Type A agent description - PROCESS.

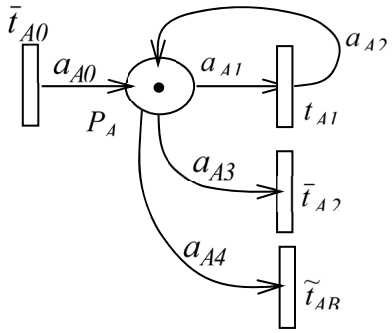

Fig. 1. Petri net simulating an agent of type A.

The spatial position of an agent of type A is determined by the system network in accordance with the dynamics of the process front based on the particular model used. The state of this agent is determined by marking the PA position: if there is a load A token in this position, the agent is considered active. At the beginning of work, the resource is activated by triggering the transition $\bar{t}_{A0}$ along the arc a_{A0} at the signal of the system network. The active state of the agent is maintained along the arcs $a_{A1,}$ a_{A2} when the transition $t_{A1.}$ is triggered. If there is no resource for distribution in the area occupied by the agent, the resource is reset through the transition $\bar{t}_{A2.}$When agent of type A interacting with an agent of type B, the resource load A decreases at each simulation step by the value delta load A, which is transmitted along the arc $a_{A4}$when the transition $\tilde{t}_{AB.}$is triggered. When load A=0, an agent of type A becomes passive.

Type B agent description - FIGHTER

The spatial position of an agent of type B is determined by the system network in accordance with the algorithm for the movement of opposing forces.

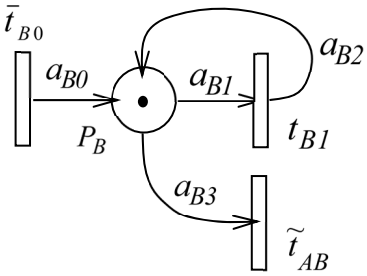

Fig. 2. Petri net simulating an agent of type B.

The state of this agent is a Boolean variable and is determined by marking the position PB: if there is a resource in this position, the agent is considered active. At the beginning of work, the resource is activated through the transition $\bar{t}_{B0}$ along the a_{B0} from the system

network. The active state of the agent is maintained along the arc $a_{B1}$when the transition $t_{B1.}$is triggered. When the transition $\tilde{t}_{AB}$ is activated, the agent of type A - decrease the value of the token load_A in position PA, as described above.

3.2 Integration of agents of types A and B

The integration model for agents of types *A* and *B* is shown in Figure 3.

This process is controlled by the system network *SN*, which performs the functions of an agent-manager M gives permission to the agents and supplies them with resources. In Figure 3, as an example, a fragment of the satellite part of the colored Petri net is shown, showing the interaction of agents of type A and B, controlled by the system network, which is represented by agent M.

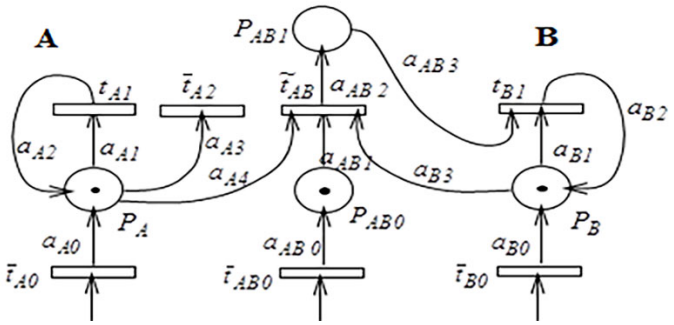

Fig. 3. A fragment of the satellite Petri net describing the integration of agents A and B under the control of a system network modeled by agent M.

The spatial position of agent A is determined in accordance with the propagation process model. The state of this agent is determined by marking the position PA, which includes a Boolean variable status A that defines the status of the agent, and the intensity of energy release to propagate intensity A in the local area occupied by the agent. If there is a resource in this position, the agent is considered active. The implementation of the described algorithm in the form of an agent-based model made it possible to effectively manage the population of A-agents (add and remove agents). At the beginning of work, the resource is activated by a signal from the system network by triggering the transition $\bar{t}_{A0}$along the arc $a_{A0.}$ The active state of the agent is maintained along the a_{A1}, a_{A2} when the transition t_{A1} is triggered. In the absence of conditions for the start of the process propagation in the area occupied by the agent, the resource is reset to zero through the transition $\bar{t}_{A2}$.

When interacting agent A with an agent of type B, the intensity A resource decreases at each modeling step by the value delta_intensity_A, which is transmitted along the arc a_{A4} when the transition $\bar{t}_{AB}$fires. With intensity A=0, agent A becomes passive and goes into the "stay" state. The spatial position of the B-agent is determined by the system network in accordance with the algorithm of the movement grids.

The state of the agent is a Boolean variable *status B* and is determined by marking the position P_B: if there is a resource in this position, the agent is considered active. At the beginning of work, the resource is activated through the transition $\bar{t}_{B0}$along the arc *a_B0* from the system network *SN* (agent *M*). The active state of the agent is maintained along the a_{B0} when the transition $\bar{t}_{B1}$ fires.

4 Two-level multi-agent system using nested Petri nets

The formalism of Petri nets is convenient to use to create multi-agent systems that model the propagation and localization of dynamic processes on the Earth's surface. One of the tools for solving this problem is the so-called Nested Petri Nets (NPN) [3].

Let us consider a formal description of a nested Petri net that models the interaction of these agents.

A two-level GIS-oriented nested Petri net NPN contains the following elements: a system net, which is a colored Petri net that controls the operation of satellite nets; satellite colored Petri nets that model the behavior of type A agents; satellite colored Petri nets that model the behavior of a type B agent; a function labels of transitions, with the help of which vertical and horizontal synchronization of elements of the nested network is carried out; the function of displaying the state of the network on the map of the area.

An example of an NPN network is shown in Figure 4.

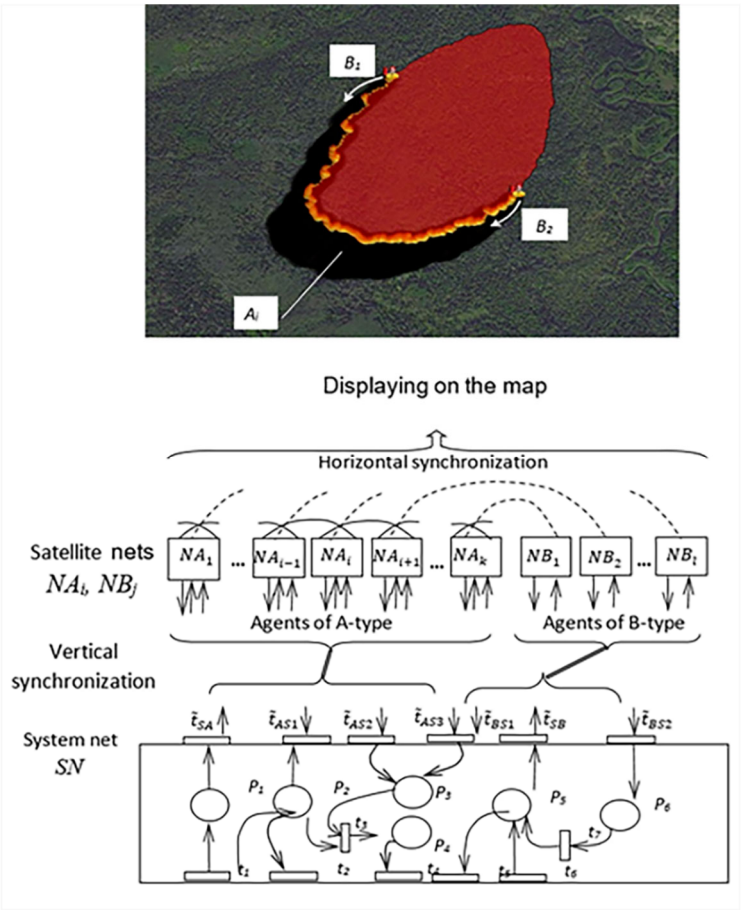

Fig. 4. Scheme of a multi-agent system that simulates the spread and elimination of a wild fire, which is implemented as a two-level nested Petri net.

The structure of networks included in NPN is similar for both system and satellite networks. Each of them is a colored Petri net PN in the form of a division 2, however, to ensure the operation of these nets as a single multi-agent system, the function of labeling

transitions $\Lambda = \Lambda_v \cup \Lambda_h$ is provided, which performs vertical and horizontal synchronization of the networks using special transitions, shown in figure 4. Vertical synchronization is provided by the transition marked with a tilde sign $\tilde{t}_{AS}$, and horizontal synchronization is provided by the transition marked with a horizontal bar $\bar{t}_{i+1}$. The described system contains two subsystems: the base (system) - *M*-agent and satellite (agents - *A, B* and *D*). The system part contains information about the simulation environment and manages the interaction of satellite agents that simulate the situation on the map. Transitions t_1 - t_6 define the user interface.

The considered system is implemented, in particular, using the CPN Tools software [2], a tool for editing, modeling and analyzing high-level Petri nets. This program supports basic Petri nets as well as temporary Petri nets and colored Petri nets. It has a simulator and a state space analysis tool. However, within the framework of this article, the issue of implementation is not considered.

5 Conclusion

The paper shows that the use of the formalisms of multi-agent systems, color and nested Petri nets allows you to create visual models of dynamics and control, including for complex dynamic processes on the Earth's surface.

The Taiga-3 software system developed on the basis of the formalisms described in present article, combined with the MOODLE distance learning process support system, served as the basis for the creation of the Taiga-analyst computer simulator, designed to teach decision-making to combat wild fires [10]. The system is currently accepted for use in the Siberian Fire and Rescue Academy as a training simulator for cadets in their preparation in the specialties 20.05.01 "Fire Safety" and 20.03.01 "Techno-spheric Safety".

References

1. S. Russell, P. Norvig, *Artificial intelligence*. *Modern approach* (Williams Publishing House, 2006)
2. K. Jensen, *Coloured Petri Nets. Basic concepts; analysis methods and practical use* **1-3** (Springer-Verlag Berlin, 1997)
3. I.A. Lomazova, *Nested Petri Nets: Modeling and Analysis of Distributed Systems with an Object Structure* (M., Scientific world, 1994)
4. Xu Xue-Guo, Applied Science **9(5)** 983 (2019). https://www.doi.org/10.3390/app9050983
5. Harald Storrle, *Software architecture models - design and analysis using UML and Petri nets* (Books on request, 2020)
6. H. Gao, Hu Xinyang, IEEE Access **8** 18420-18425 (2020). https://www.doi.org/10.1109/ACCESS.2020.2968510
7. Marion Juan, David Mailand, Phifis, Nicholas, Gregoris Guy, Techniques de l'Ingenieur. Securité des Systèmes Industriels (2021). https://www.doi.org/10.51257/a-v1-se1221
8. E.P. Bulavsky, O.K. Vaisov, Transport automation **6(4)**, 687- 692 (2018)
9. Manuel Chiacio, Juan Chiacio, Darren Presscott, John Andrews, Information sciences **453** 323-345 (2018). https://www.doi.org/10.1016/j. ins. 2018.04.029
10. G. Dorrer, A. Dorrer, and S. Yarovoy, IOP Conf. Ser.: Mater. Sci. Eng. **537** 042052 (2019)