8th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 25 – 27, 2006

# INTELLIGENT CONTROL OF ARTIFICIAL SOCIAL SYSTEMS

**Călin CIUFUDEAN[1], Alexandru LARIONESCU[1]**
**George DILINTAS[2], Panagiotis SINIOROS[2]**
[1]*"Ştefan cel Mare" University of Suceava*
*str. Universităţii nr.13, RO-720229 Suceava*
*calin@eed.usv.ro, lari@eed.usv.ro*
[2]*Technological Education Institute of Piraeus, Greece*
*pasin@teipir.gr*

***Abstract****. This paper proposes a new model for Artificial Social Systems (ASS) behaviors. ASS exists in practically every multi-agent system, and play a major role in the performance and effectiveness of the agents. This is the reason why we introduce a more suggestive model for ASS. To model these systems, a class of Petri nets is adopted and briefly introduced in the paper. This class allows representing the flow of physical resources and control information data of the ASS's components.*
*The issue of the verification of the interconnections or interfaces among the Petri nets of primary components is realized with the help of the functional abstractions of the Petri net model of these components.*
***Keywords****: Artificial Social Systems, Petri nets, Social Laws, Multi-agent Systems, formal verification, eventuality.*

## Introduction

An Artificial Social System (ASS) is a set of restrictions on agent's behavior in a multi-agent environment [1].
ASS allows agents to coexist in a shared environment and pursue their respective goals in the presence of other agents. A multi-agent system consists of several agents, where at given point, each agent is in one of several states.
In each of its states, an agent can perform several actions. The actions an agent performs at a given point may affect the way the state of this agent and the state of other agents will change.
A system of dependent automata consists of two or more agents, each of which may be in one of a finite number of different local states. We denote the set of local states of an agent *i* by $P_i$. The set ($P_1$, $P_2$, ..., $P_n$) of states of the different agents is called system's configuration.
The set of possible actions an agent *i* can perform is a function of the local state. For every state $p \in P_i$ there is a set $A_i(p)$ of action that *i* can perform when in local state p.
The row actions ($a_1$, ..., $a_n$) denote the actions the different agents perform at a given point and is called their joint action there. An agent's next state is a function of the system's current configuration and the joint action performed by the agents. A goal for an agent is identified with one of its states. That is the reason why an agent has plans how to attain its goal.
A plan for agent *i* in a dependent automata is a function U(p) that associates with every state p of agent *i* a particular action $a \in A_i(p)$. A plan [2] is said to guarantee the attainment of a particular goal starting from an initial state, in a given dependent automata system, if by following this plan the agent will attain the goal, regardless of what the other agent will do, and what are the initial states of the other agents. A dependent automata system is said to be social if, for every initial state $p_o$ and goal state $p_g$, it is computationally feasible for an agent to devise, on-line, an efficient plan that guarantees to attain the goal $p_g$ state when starting in the initial state $p_o$. For a proper behavior, a dependent automata system is modeled with a social law. Formally, a social law Q for a given dependent automata

system consists of functions ( $A_1^{'}$, $A_2^{'}$, ..., $A_n^{'}$), satisfying $A_i^{'}(p) \subset A_i(p)$ for every agent *i* and state $p \in P_i$. Intuitively, a social law will restrict the set of actions an agent is "allowed" to perform at any given state. Given a dependent automata system S and a social law Q for S, if we replace the functions $A_i$ of S by the restricted functions $A_i^{'}$, we obtain new dependent automata system. We denote this new system by $S^Q$. In $S^Q$ the agents can behave only in a manner compatible with the social law T [3], [4].

In controlling the actions, or strategies, available to an agent, the social law plays a dual role. By reducing the set of strategies available to a given agent, the social system may limit the number of goals the agent is able to attain. By restricting the behaviors of the other agents, however, the social system may make it possible for the agent to attain more goals and in some cases these goals will be attainable using more efficient plans than in the absence of the social system.

A semantic definition of artificial social systems gives us the ability to reason about such systems. For example, the manufacturer of the agents (e.g., robots) that are to function in the social system will need to reason about whether its creation will indeed be equipped with the hardware and the software necessary to follow the rules. In order to be able to reason properly, we need a mathematical model and a description language [8], [9]. We chose the Petri nets model and a prepositional language.

## A class of Petri nets for modeling artificial social systems

Petri nets as a mathematical and graphical tool, provide a uniform environment for modeling, formal analysis and design of discrete-event systems. The resemblance between social systems and discrete-event systems gives us the opportunity to model the artificial social systems with Petri nets. One of the major advantages of using Petri nets is that the same model can be used for the analysis of behavioral properties as well as performance evaluation. Petri nets as a graphical tool provide a powerful communication medium between the user, typically the engineer requirements, and the customer ones. Complex requirement specifications can be represented graphically using Petri nets instead of using ambiguous textual descriptions or mathematical notations difficult to understand by the customer. The interactive graphical simulation of Petri nets models of systems allowed one to study the dynamics of the modeled systems [5].

As a mathematical tool, Petri nets allow the formal analysis of the modeled systems. Petri nets pass a number of properties. These, when interpreted in the context of the modeled system (e.g., social system) allow the system designer to identify the presence or absence of the specific functional properties (e.g., social laws) of the system under design. The recognition of these factors allowed us to model and analyze social systems such as production lines, job shops, robotic assembly cells, flexible manufacturing systems etc.

We may notice that, even if design model can be constructed fairly rapidly by using predefined objects, checking the model for its logical correctness may still be a hard task. Although, logical correctness of an object can be established separately from other objects, the need to check the correctness of the interactions (social laws) between objects requires the whole model of the system to be considered. So, because the social systems are usually complex, with large dimensions, their complexity may be such that the model is no longer amenable to analysis by classical methods provided for Petri nets. One possible solution to the complexity problem is to replace the "mechanism" which realizes the functionality of an object by a simplified "mechanism" which retains the required functionality [6]. This functionality defines the way an object (e.g., agent) responds to its inputs (e.g., stimuli). The replacement objects are called functional abstractions.

We notice that the mechanism of an abstraction no longer bears resemblance to that of the actual object. However, the fact that functional

abstractions retain the functionality of the actual objects is sufficient to study the correctness of the interactions among components of the system (e.g., the behavior of an artificial social system), without paying attention to the correctness of the components themselves. An additional advantage of using Petri nets with functional abstractions is the compact representation of the model of the system.

This helps to comprehend architecture and functionality of complex social systems.

We define functional abstractions of Petri nets models in general term [4]:
Let PN = (P, T, IN, OUT) be a Petri net model of an object.
Let $P_I = \{P_{I1}, P_{I2}, ..., P_{Im}\}$ be a set of interface places of the model. The interface places are the places via which the model interacts with its environment.
Let $P_i = \{P_{i1}, P_{i2}, ..., P_{in}\}$ be a set of internal places of the model. We have, $P = P_I \cup P_i$ and $P_I \cap P_i = \emptyset$.
Let $T = \{t_1, t_2, ..., t_k\}$ be a set of transitions of the model where $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.
Let f be the functionality of the object represented by PN.
Let $P_{FAI} = \{P_{FAI1}, P_{FAI2}, ..., P_{FAIu}\}$ be a set of interface places of the functional abstraction.
Let $P_{FAi} = \{P_{FAi1}, P_{FAi2}, ..., P_{FAiw}\}$ be a set of internal places of the functional abstraction. Thus $P_{FA} = P_{FAI} \cup P_{FAi}$ and $P_{FAI} \cap P_{FAi} = \emptyset$.
Let $f_{FA}$ be the functionality of the functional abstraction of PN. Functional abstraction $PN_{FA}$ of a Petri net model PN of an object is a Petri net, which has:
$P_I = P_{FAI}$ Interface places of the Petri net model, PN, and its functional abstraction, $P_{FA}$, are identical.
$P_i \cap P_{FAi} = \emptyset$. Different internal places, and $|P_i| > |P_{FAi}|$; PN involves more internal places than $PN_{FA}$.
$T \cap T_{FA} = \emptyset$. Different transitions, and $|T| > |T_{FA}|$; PN involves more transitions then $T_{FA}$.
$f = f_{FA}$ Functionality of PN and $PN_{FA}$ are identical.

The functionality of a $PN_{FA}$ can be described by the following semantic [5], [7]: ($PN_{FA}$, f); where f is a formula having the following syntax:
1) propositions: p, $t_{fir}$, and t, where $p \in P$ and $t \in T$, are atomic propositions;
2) atomic propositions are formula;
3) if f and g are formulas, then so are -g, f + g, f · g, $f \Rightarrow g$, of, [ ] f, < > f.

The atomic propositions *p*, $t_{fir}$, and *t*, mean that there is at least one token in place *p* in the current marking, that transition *i* can fire in the current marking, and that transition *t* fires in the current marking, respectively.
Symbols -, +, · , and $\Rightarrow$ represent the Boolean connectives. The formula of, "next", means that *f* becomes true in the next marking. The formula [ ] f, "henceforth" means that *f* becomes true in every marking reached from the current marking.
The formula $\langle\,\rangle$ f, "eventually", means that *f* becomes true at some marking reachable from the current marking. Let S be dependent automata system. Let, also, $\alpha, \beta$ be sequences of elements of S. $|\alpha|$, $|\beta|$ denote the length of $\alpha, \beta \in S$. Let L(PN,M), where M is a marking of a finite PN, be a set of all firing sequences from M. For a formula f, $\langle M, \alpha \rangle \models$ means that f is satisfied by the pair of M and $\alpha$, where $\models$ denotes a valid formula.
The following properties, which were proved in [5], [8], hold:

a) $\langle M, \alpha \rangle \models f + of$ *implies* $\langle M, \alpha \rangle \models \langle\,\rangle f$ (1)

b) $\langle M, \alpha \rangle \models [\,]\ (f_1 \Rightarrow \langle\,\rangle f_2) \cdot [\,]\ (f_2 \Rightarrow \langle\,\rangle f_3$ *implies* $\langle M, \alpha \rangle \models [\,]\ (f_1 \Rightarrow \langle\,\rangle f_3)$ (2)

c) $\langle M, \alpha \rangle \models \langle\,\rangle (\langle\,\rangle f)$
*implies* $\langle M, \alpha \rangle \models \langle\,\rangle f$ (3)

d) $\langle M, \alpha \rangle \models [\,]\ (t_{i\,fir} \Rightarrow \langle\,\rangle t_i)$ (4)

**An example**

To demonstrate how this class of Petri nets models can be used to represent the artificial social systems, we consider a shunting operation [9].

The railway convoy initially is in front of the reception railway group (a token present on place $P_1$ in Fig.1).

The convoy moves (firing transition $t_1$) to the reception group (a token present on place $P_2$) from the kicking horse pass.

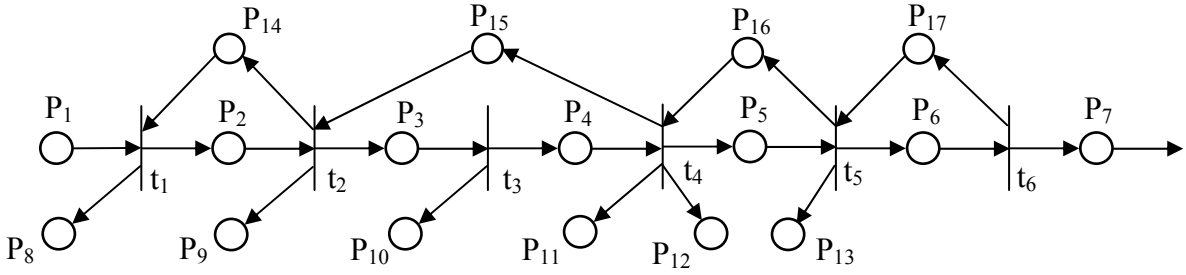


**Figure 1. Petri nets model of a shunting.**

The condition in which the convoy moves is that the reception railway is free (a token present on place $P_{14}$). The drive railway engine is sent, afterwards, from the reception group (a token present on place $P_8$) back to the station.

If the kicking horse pass is free (a token present on place $P_{15}$) the convoy moves there (firing transition $t_2$). The presence of the convoy on the kicking horse pass (a token present on place $P_3$) means, also, that the convoy needs free shunting railways (a token present on place $P_9$). When the shunting operation is allowed (that means that the shunting railways are free, and a token is present on place $P_{10}$), the wagons are shunted (firing transition $t_4$). After that, the wagons for the same route form another convoy in the compound railway group (a token present on place $P_4$), if and only if, the compound group is free (a token present on place $P_{16}$). A railway engine will be attached to the convoy (firing transition $t_4$ when a token is present on place $P_{12}$) and the shunting railways are free (a token present on place $P_{11}$) and the new convoy is ready to go (a token present on place $P_5$). The railway engine is attached to the convoy (firing transition $t_5$, when a token is present on place $P_{13}$) if there is available an engine (a token present on place $P_{17}$). After that, the convoy is tugged to the dispatch railway group (firing transition $t_6$) if the dispatch group is free (a token on place $P_6$). The following production rules and corresponding logic formulas can describe the functionality, or external functional behavior, of the shunting operation. To provide an explicit link between descriptive and formal representation of functionality, in the production rules, labels of the corresponding logic formulas are included in the brackets. The same labels are used to tag interface places in Petri nets models, as well as to represent the corresponding logic formulas. This is to provide an explicit link between functionality and the formal representation expressed by these two environments.

1) If the railway convoy is in front of the reception railway group ($P_1$), after the convoy overtakes the kicking horse pass, the engine is sent back to the station ($P_8$) and a demand for free shunting railways is initiated ($P_9$).

$$MS_1 = [\,]\ (P_1 \Rightarrow \langle\,\rangle\ (P_8 P_9)) \qquad (5)$$

2) If the demand for free shunting railways is initiated ($P_9$), then the shunting operation is allowed ($P_{10}$).

$$MS_2 = [\,]\ (P_9 \Rightarrow \langle\,\rangle\ P_{10}) \qquad (6)$$

3) After the shunting operation, the railways are set free and ($P_{11}$) and a demand for an engine is initiated ($P_{12}$).

$$MS_3 = [\,]\ (P_{10} \Rightarrow \langle\,\rangle\ (P_{11} P_{12})) \qquad (7)$$

4) If the demand for an engine is initiated ($P_{12}$), then the engine is attached to the new convoy ($P_{13}$).

$$MS_4 = [\,]\,(P_{12} \Rightarrow \langle\,\rangle\, P_{13}) \quad (8)$$

5) If the engine is attached to the new convoy ($P_{13}$), then the convoy goes to the dispatch group ($P_7$).

$$MS_5 = [\,]\,(P_{13} \Rightarrow \langle\,\rangle\, P_7) \quad (9)$$

This functionality can be easily extracted from the graphical representation of the Petri net shown in Fig.2. This model contains all the interface places, which appear in the original Petri net model shown in Fig.1. The net structure connecting these places does realize the required functionality, without any concern for the actual structure and dynamics of the original Petri net model reflecting the "inner" working of the modeled object.

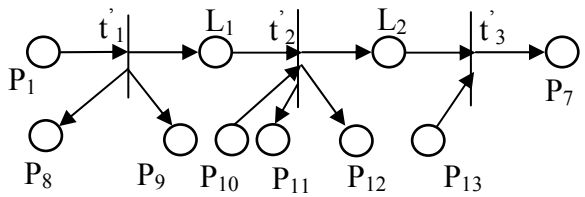


**Figure 2. Functional abstraction of the PN in Figure 1.**

## Conclusions

In this paper, an approach to the design of artificial social systems using Petri nets and their functional abstractions was presented. To model artificial social systems, a class of Petri nets was briefly recalled. This class allows representing unidirectional flow of physical resources and information control data. The issue of the verification of the design model was addressed by the introduction of functional abstraction of the PN. By using functional abstractions during the verification stage, the graphical complexity of the design model can be reduced substantially. This is useful especially when large-scale systems are designed. It allows the designer to establish correct flow of resources and information/data in the system before the level of detail of actual components is increased to result in desired structural properties. We exemplify our approach on a technical operation (e.g., shunting) from railway systems. Interesting studies can be made on large human social systems; therefore, future work will deal with this stringent problem.

## References


[1] G. Gaspar, "Communication and Belief Changes in a Society of Agents: Towards a Formal Model of an Automated Agent". *In J. Demazeau and J. P. Muller, editors, Decentralised AI2*, pp.245-255, North-Holland/Elsevier, 1991

[2] Y. Moses, M. Tennenholtz, "Artificial Social Sysems", *www.Kluivert.dl., Sistem Automato*, 2002.

[3] R. Zurawski, M. Zhon, "Petri nets and industrial applications; A tutorial*", IEEE Trans. Ind. Electron.*, vol.41, pp.567-583, Dec. 1994.

[4] R. Zurawski, "Systematic construction of functional abstraction of Petri nets model of flexible manufacturing systems", *IEEE Trans. Ind. Electron.*, vol.41, pp.584-592, Dec. 1994.

[5] L. Lamport, "Sometime is sometime not ever", *In Proc. 7th ACM Syrup., Las Vegas, N.V.*, pp.174-185, Jan.1980

[6] Ciufudean C., "Modeling the reliability of the interaction man-machine in railway transport"*, The Annals of the "Stefan cel Mare" University of Suceava*, Year VII, No.13, pp.80-84, 2000.

[7] C. Ciufudean, A.B. Larionescu, "Safety criteria for production lines modeled with Petri nets", *Advances in Electrical and Computer Engineering*, vol. 2(9), no.2(18), pp.15-20, 2002.

[8] C. Ciufudean, D. Popescu, "Modeling Digital Signal Perturbation with Stochastic Petri Nets", *Advances in Electrical and Computer Engineering,* vol. 4(II), no. 1 (21), pp. 71-75, Suceava, Romania, 2004.

[9] C. Ciufudean, G. Mahalu, G. Anastasiu, *Petri net representation of shunting operations*, Advances in Electrical and Computer Engineering, vol. 1(8), no. 2(16), pp. 25-28, 2001.