

Title	A Petri-Net-Based Model for the Mathematical Analysis of Multi-Agent Systems
Author(s)	HIRAISHI, Kunihiko
Citation	IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, E84-A(11): 2829-2837
Issue Date	2001-11-01
Type	Journal Article
Text version	publisher
URL	<a href="http://hdl.handle.net/10119/4688">http://hdl.handle.net/10119/4688</a>
Rights	Copyright (C)2001 IEICE. Kunihiko Hiraishi, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, E84-A(11), 2001, 2829-2837. <a href="http://www.ieice.org/jpn/trans_online/">http://www.ieice.org/jpn/trans_online/</a>
Description	

# A Petri-Net-Based Model for the Mathematical Analysis of Multi-Agent Systems

Kunihiko HIRAISHI<sup>†a)</sup>, *Regular Member*

**SUMMARY** Agent technology is widely recognized as a new paradigm for the design of concurrent software and systems. The aim of this paper is to give a mathematical foundation for the design and the analysis of multi-agent systems by means of a Petri-net-based model. The proposed model, called  $PN^2$ , is based on place/transition nets (P/T nets), which is one of the simplest classes of Petri nets. The main difference of  $PN^2$ 's from P/T nets is that each token, representing an agent, is also a P/T net.  $PN^2$ 's are sufficiently simple for the mathematical analysis, such as invariant analysis, but have enough modeling power.

**key words:** *Petri nets, multi-agent systems, object orientation, agent orientation, mobile agents*

## 1. Introduction

Agent technology is widely recognized as a new paradigm for the design of concurrent software and systems. The aim of this paper is to give a mathematical foundation for the design and the analysis of multi-agent systems by means of a Petri-net-based model.

Petri nets are a well-known model for concurrent and distributed systems, and there have been many results on the theory, and also on practical applications. To represent an agent as a token in Petri nets, it is necessary to introduce notion of *objects*, i.e., a collection of data types with methods that give access to the data, into attributes of tokens. There have been various Petri-net-based models introducing this concept [1]–[8]. Since most of these classes of object Petri nets are based on high-level Petri nets, i.e., they allow arbitrary transformation functions on tokens, they are too complex to be analyzed. These high-level object Petri nets aim to describe real applications in an object-oriented manner, and their research direction is different from that of this paper. We aim to propose a model that is sufficiently simple for the mathematical analysis, but has enough modeling power.

Recently, an elementary class of object Petri nets, called *elementary object systems* (EOS), has been proposed [9] to model systems in an object oriented manner keeping the model as simple as possible. The approach of this paper is similar to that of EOS. The proposed model, called  $PN^2$  (“Petri Nets in a Petri Net”), is

based on place/transition nets. Intuitively, a  $PN^2$  is a Petri net such that each token, representing an agent, is also a place/transition net. This feature is essential to represent autonomous agents, i.e., an agent is a single process that can make decisions by itself. Each token describing an agent is available for transitions in the upper-level net when the corresponding transition of the token is enabled, and changes its state by occurrences of its own transitions. Each token can move between places, and can make its copy.

Main features of  $PN^2$ 's are summarized as follows:

- (i) A class of  $PN^2$ 's, called infinite-sort  $PN^2$ 's, has the same modeling power as Turing machines.
- (ii)  $PN^2$ 's allow dynamic bindings of transitions. This may cause a combinatorial number of transition bindings, but it is avoidable in the invariant analysis. In addition, more than two agents can synchronize.
- (iii)  $PN^2$ 's are based on *the value semantics* [10], whereas many of object Petri nets are based on *the reference semantics*. In the reference semantics, each net as a token is a reference to the object. Therefore, copying tokens does not mean copying objects. In the value semantics, each net as a token represents an independent instance of objects, and therefore duplication and vanishing of objects are allowed.

## 2. Preliminaries

A multi-set over a non-empty set  $S$  is a function  $M : S \rightarrow \mathbb{N}$ , i.e., for each  $s \in S$ ,  $M(s)$  denotes the number of occurrences of  $s$  in  $M$ . Let  $\mathcal{M}_S$  denote the family of all multi-sets over  $S$ . For  $M \in \mathcal{M}_S$ , let  $|M| = \sum_{s \in S} M(s)$ . We usually denote a multi-set  $M$  by a formal sum  $\sum_{s \in S} M(s) \# s$ . For two multi-sets  $M_1, M_2 \in \mathcal{M}_S$ , we write  $M_1 \subseteq M_2$  if  $M_1(s) \leq M_2(s)$  for every  $s \in S$ . Moreover, the difference  $M_2 - M_1$  is defined only when  $M_1 \subseteq M_2$ , and is a multi-set  $\sum_{s \in S} (M_2(s) - M_1(s)) \# s$ . We simply write  $s$  to denote  $1 \# s$ , and write  $\emptyset$  to denote the empty multi-set.

A *place/transition net* (P/T net) is a triple  $\pi = (P, T, A)$ , where  $P$  is a finite set of places,  $T$  is a finite set of transitions, and  $A : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$  is a function representing arcs. A *marking* of  $\pi$  is a multi-set over  $P$ . Let  $\Sigma$  be a finite set of symbols. A P/T

Manuscript received March 19, 2001.

Manuscript revised June 4, 2001.

<sup>†</sup>The author is with School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.

a) E-mail: hira@jaist.ac.jp

net  $\pi = (P, T, A)$  with a labeling function  $\ell : T \rightarrow \Sigma$  is called a *labeled-P/T net* ( $\ell$ -P/T net, for short), and is denoted by  $\pi^\ell$ . As usual, the domain of the labeling function  $\ell$  is extended to  $T^*$ . We can identify an  $\ell$ -P/T net with a P/T net if the labeling function  $\ell$  is an injection. A pair  $\mu = (\pi^\ell, m)$  of an  $\ell$ -P/T net and a marking is called a *marked  $\ell$ -P/T net*.

Let  $\mu = (\pi^\ell = (P, T, A)^\ell, m)$  be a marked  $\ell$ -P/T net. A transition  $t$  is *enabled* in a marking  $m$  if for every  $p \in P$ :  $A(p, t) \leq m(p)$ . An enabled transition can *occur*. An occurrence of a transition  $t$  changes the marking to  $m'$  such that  $m'(p) = m(p) + A(t, p) - A(p, t)$  ( $p \in P$ ). We write  $m \xrightarrow{t} m'$  to denote this situation. Moreover, this notation is extended to finite sequences of transitions, i.e., we write  $m \xrightarrow{\sigma} m'$  to denote that an occurrence of a sequence of transitions  $\sigma \in T^*$  changes the marking from  $m$  to  $m'$ . We say that a marking  $m'$  is *reachable* from a marking  $m$  if  $m \xrightarrow{\sigma} m'$  for some  $\sigma \in T^*$ . Let  $\mathcal{R}(\mu)$  denote the set of all markings reachable from  $m$ . Since marked  $\ell$ -P/T nets will be treated as tokens in the proposed model, we introduce a state transition function  $\delta$  on the set of all marked  $\ell$ -P/T nets by  $\delta(\mu, t) = \mu'$ , where  $\mu' = (\pi^\ell, m')$ . Let  $PT_\Sigma$  denote the class of all marked  $\ell$ -P/T nets with labeling functions to  $\Sigma$ .

### 3. $PN^2$ 's

In this section, we show the definition of the model with some examples.

#### 3.1 Definition

A  $PN^2$  is a 9-tuple  $PN^2 = (P, \Sigma, T, E, \bullet, \cdot, \cdot, \cdot, s_0)$ , where

1.  $P$  is a finite set of *places*;
2.  $\Sigma$  is a finite set of *symbols*;
3.  $T$  is a finite set of *transitions*;
4.  $E$  is a finite set of *transition components*;
5.  $\bullet : E \rightarrow \mathcal{M}_P \setminus \{\emptyset\}$ ;
6.  $\cdot : E \rightarrow \mathcal{M}_P$ ;
7.  $\cdot_\Sigma : E \rightarrow \Sigma$ ;
8.  $\cdot_T : E \rightarrow T$ ;
9.  $s_0$  is the *initial configuration*, where a configuration is a mapping  $s : P \rightarrow \mathcal{M}_{PT_\Sigma}$ .

**Example 1:** Figure 1 is a graphical representation of a  $PN^2 = (P, \Sigma, T, E, \bullet, \cdot, \cdot, \cdot, s_0)$  defined as follows:

1.  $P = \{p_1, p_2, p_3\}$ ;
2.  $\Sigma = \{a, b, c, d\}$ ;
3.  $T = \{t_1, t_2, t_3\}$ ;
4.  $E = \{e_1, e_2, e_3, e_4, e_5\}$ ;
5.  $\bullet e_1 = p_1$ ,  $\bullet e_2 = p_3$ ,  $\bullet e_3 = p_2 + p_3$ ,  $\bullet e_4 = p_2$ ,  $\bullet e_5 = p_2$ ;
6.  $e_1^\cdot = p_2$ ,  $e_2^\cdot = p_2 + p_3$ ,  $e_3^\cdot = p_2 + p_3$ ,  $e_4^\cdot = p_1$ ,  $e_5^\cdot = \emptyset$ ;
7.  $e_{1\Sigma} = a$ ,  $e_{2\Sigma} = c$ ,  $e_{3\Sigma} = d$ ,  $e_{4\Sigma} = b$ ,  $e_{5\Sigma} = c$ ;

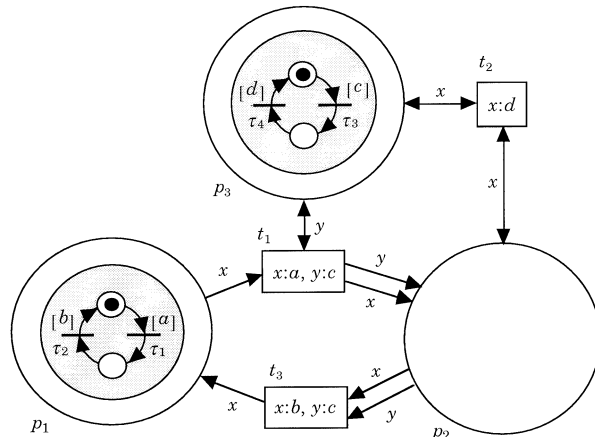


Fig. 1 An example of  $PN^2$ 's.

8.  $e_{1T} = t_1$ ,  $e_{2T} = t_1$ ,  $e_{3T} = t_2$ ,  $e_{4T} = t_3$ ;  $e_{5T} = t_3$ ;
9.  $s_0 : p_1 \mapsto \mu_1$ ;  $p_2 \mapsto \emptyset$ ;  $p_3 \mapsto \mu_2$ , where  $\mu_1$  and  $\mu_2$  are the marked  $\ell$ -P/T nets shown in the figure.

Each transition  $t \in T$  consists of a set of transition components  $E_t := \{e \in E \mid e_T = t\}$ . In the graphical representation of  $PN^2$ 's, we associate labels  $x, y, z, \dots$  with each arc to distinguish transition components, and write  $x : a, y : b, \dots$  to indicate that  $x_\Sigma = a$ ,  $y_\Sigma = b, \dots$  for transition components  $x, y, \dots$ .

In the definition of  $PN^2$ , the 6-tuple  $(P, T, E, \bullet, \cdot, \cdot, \cdot, \cdot_T)$  defines a P/T net. We call this *the upper-level net*. On the other hand, marked  $\ell$ -P/T nets in the configuration is called *lower-level nets*.

We also define a subclass of  $PN^2$ 's, called *finite state agent nets* (FSAN's), which will be useful in describing real applications. An FSAN is a  $PN^2$  such that each marked  $\ell$ -P/T net in the lower-level is a deterministic finite automaton (DFA), or a one-safe state machine in Petri net terminology.

#### 3.2 State Transition Rule

Let  $T_\Sigma$  denote the set of all transitions of marked  $\ell$ -P/T nets in  $PT_\Sigma$ . A *transition binding* for a transition  $t \in T$  is a pair of functions  $b : E_t \rightarrow PT_\Sigma$  and  $w : E_t \rightarrow T_\Sigma$  such that for each  $e \in E_t$ :

- (i)  $w(e)$  is a transition of  $b(e)$ ;
- (ii)  $w(e)$  has a label  $e_\Sigma$ ;
- (iii)  $w(e)$  is enabled in  $b(e)$ .

A transition binding  $(b, w)$  is *enabled* in a configuration  $s$  if for each  $p \in P$ ,

$$\sum_{e \in E_t} \bullet e(p) \# b(e) \subseteq s(p).$$

A transition binding can *occur* if it is enabled. An occurrence of a transition binding  $(b, w)$  changes the configuration to  $s'$  such that for each  $p \in P$ ,

$$s'(p) = s(p) - \sum_{e \in E_t} \bullet e(p) \# b(e) + \sum_{e \in E_t} e \bullet (p) \# \delta(b(e), w(e)).$$

Since  $\bullet e \neq \emptyset$  for every transition component  $e$  (see 5. in the definition of  $PN^2$ ), no new  $\ell$ -P/T nets are introduced by any occurring of transition bindings. Let  $B_t$  denote the set of all transition bindings for a transition  $t$  and let  $B = \cup_{t \in T} B_t$ .

We write  $s \xrightarrow{(b,w)} s'$  to denote that an occurrence of a transition binding  $(b, w)$  changes the configuration from  $s$  to  $s'$ , and this notation is extended to finite sequences of transition bindings. We say that a configuration  $s'$  is *reachable* from a configuration  $s$  if  $s \xrightarrow{\sigma} s'$  for some  $\sigma \in B^*$ . Let  $\mathcal{R}(PN^2)$  denote the set of all configurations reachable from the initial configuration.

**Example 2:** We consider a  $PN^2$  in Fig. 1. Denoting each configuration  $s$  by a vector  $[s(p_1), s(p_2), s(p_3)]$ , the configuration changes as follows.

$$\begin{aligned} & [\mu_1, \emptyset, \mu_2] \xrightarrow{(b_1, w_1)} [\emptyset, \mu'_1 + \mu'_2, \mu'_2] \xrightarrow{(b_2, w_2)} \\ & [\emptyset, \mu'_1 + \mu_2, \mu_2] \xrightarrow{(b_3, w_3)} [\mu_1, \emptyset, \mu_2], \end{aligned}$$

where  $\mu'_1 = \delta(\mu_1, \tau_1)$ ,  $\mu'_2 = \delta(\mu_2, \tau_3)$ ,  $\mu_1 = \delta(\mu'_1, \tau_2)$ ,  $\mu_2 = \delta(\mu'_2, \tau_4)$ , and  $(b_i, w_i)$  ( $i = 1, 2, 3$ ) are unique transition bindings for transition  $t_i$  such that

$$\begin{aligned} b_1 : e_1 \mapsto \mu_1, e_2 \mapsto \mu_2; \quad w_1 : e_1 \mapsto \tau_1, e_2 \mapsto \tau_3, \\ b_2 : e_3 \mapsto \mu'_2; \quad w_2 : e_3 \mapsto \tau_4, \\ b_3 : e_4 \mapsto \mu'_1, e_5 \mapsto \mu_2; \quad w_3 : e_4 \mapsto \tau_2, e_5 \mapsto \tau_3. \end{aligned}$$

**Example 3:** We consider an AGV (Automated Guided Vehicle) system consisting of a graph representing tracks on which two AGV's are moving (Fig. 2).  $AGV_1$  visits  $A$  and  $D$  alternately, and  $AGV_2$  visits  $C$  and  $F$  alternately. The track graph is partitioned into three zones in which at most one AGV is allowed to exist at any moment.

We model this system by an FSAN. Figure 3 is the upper-level net representing the track.  $AGV_1$  and  $AGV_2$  are modeled by DFA's shown in Fig. 4, which are initially put on place  $A$  and  $C$ , respectively. Each controller that supervises a zone is modeled by a DFA in

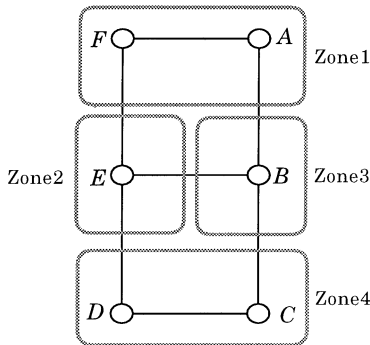


Fig. 2 Track graph and zones.

Fig. 5, which is put on place  $Z_i$  ( $i = 1, \dots, 4$ ). When an AGV moves from one zone to another, it communicates with zone controllers of both zones. The state of each zone controller is changed by this movement.

Behavior of each AGV is determined by interaction with *the environment* consisting of the track graph and other agents. In this system,  $AGV_1$  and  $AGV_2$  will behave as follows:

$$\begin{aligned} AGV_1 : & A \rightarrow B \rightarrow (C \text{ or } E) \rightarrow D \rightarrow C \rightarrow B \\ & \rightarrow (A \text{ or } E \rightarrow F \rightarrow A) \rightarrow \dots, \\ AGV_2 : & C \rightarrow B \rightarrow (A \text{ or } E) \rightarrow F \rightarrow A \rightarrow B \\ & \rightarrow (C \text{ or } E \rightarrow D \rightarrow C) \rightarrow \dots. \end{aligned}$$

Reduction of states in lower-level nets is an interesting problem, and is also practically important. Be-

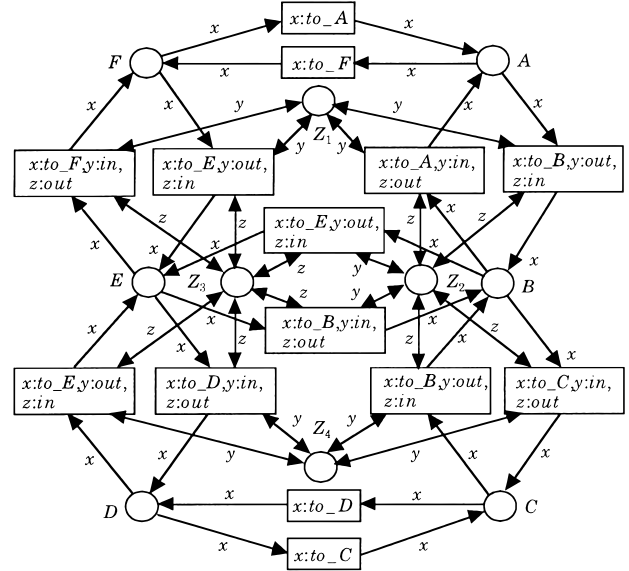


Fig. 3 The upper-level net.

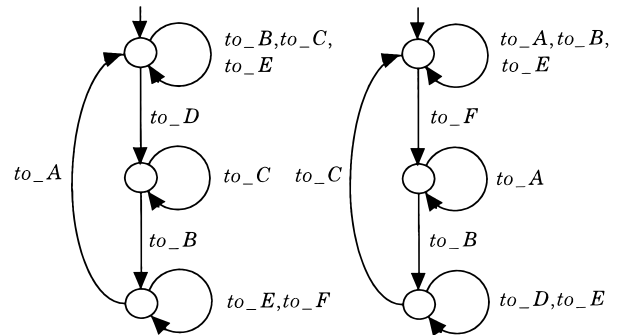


Fig. 4 DFA's representing  $AGV_1$  and  $AGV_2$ .

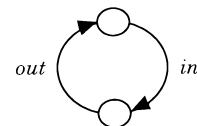


Fig. 5 A DFA for zone control.

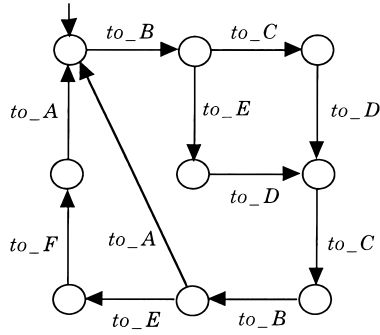


Fig. 6 Another representation of  $AGV_1$ .

havior of AGV in Fig. 6 is the same as that in Fig. 4, but it has more states. This problem is closely related to *decentralized supervisory control problem* of discrete event systems [12].

#### 4. Modeling Power of $PN^2$ 's

In this section, we show some results on the modeling power of  $PN^2$ 's.

##### 4.1 Simulating Marked P/T Nets

Marked P/T nets can be considered as a special class of  $PN^2$ 's by treating each token as a marked  $\ell$ -P/T net having one transition and no places, i.e., it takes only one state and the transition is always enabled. As usual, we draw “•” to denote such a token (e.g., Fig. 9).

##### 4.2 Bounded/Finite-Sort $PN^2$ 's

A  $PN^2$  is called *bounded* if there exists a nonnegative integer  $k$  such that for every  $s \in \mathcal{R}(PN^2)$  and every  $p \in P$ :  $|s(p)| \leq k$ , and is called *unbounded* otherwise. A  $PN^2$  is called *finite-sort* if all marked  $\ell$ -P/T nets contained in the initial configuration  $s_0$  are bounded<sup>†</sup>, and is called *infinite-sort* otherwise. When a  $PN^2$  is finite-sort, there exists a marked P/T net  $\widetilde{PN}^2 = (\widetilde{P}, \widetilde{T}, \widetilde{A}, \widetilde{m}_0)$  that simulates the behavior of the  $PN^2$ . We show the construction.

First we define the following sets.

- $\overline{PT}_\Sigma$  is the set of all marked  $\ell$ -P/T net  $(\pi^\ell, m)$  such that  $m \in \mathcal{R}(\mu_0)$  for some  $\mu_0 = (\pi^\ell, m_0)$  in the initial configuration.
- $\overline{B}_t$  is the set of all transition binding  $(b, w)$  for transition  $t$  that is valid, i.e., for all  $e \in E_t$ :  $b(e) \subseteq \overline{PT}_\Sigma$ . Let  $\overline{B} = \overline{B}_t$ .

By the assumption, both  $\overline{PT}_\Sigma$  and  $\overline{B}$  are finite and effectively computable. Note that in general not all marked  $\ell$ -P/T nets in  $\overline{PT}_\Sigma$  appear in reachable configurations.

For each place  $p \in P$  and each marked  $\ell$ -P/T net

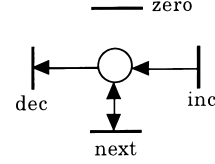


Fig. 7 Construction of a  $PN^2$  representing the register machine (1).

$\mu \in \overline{PT}_\Sigma$ , we prepare a place  $p_\mu$ , and for each transition  $t \in T$  and each transition binding  $(b, w) \in \overline{B}_t$ , we prepare a transition  $t_{(b,w)}$ , where arcs are connected as follows:

- $\widetilde{A}(p_{b(e)}, t_{(b,w)}) = \bullet e(p)$  and  $\widetilde{A}(t_{(b,w)}, p_{\delta(b(e), w(e))}) = e \bullet(p)$  ( $e \in E_t$ );
- Otherwise,  $\widetilde{A}(\cdot, t_{(b,w)}) = \widetilde{A}(t_{(b,w)}, \cdot) = 0$ .

The initial marking  $\widetilde{m}_0$  is defined by  $\widetilde{m}_0(p_\mu) = s_0(p)(\mu)$ . Then the state transition diagrams of  $PN^2$  and  $\widetilde{PN}^2$  are isomorphic to each other.

Note that even if a  $PN^2$  is bounded, it is not always possible to construct a marked P/T net that simulates the behavior of the  $PN^2$ . An example will be shown in the next subsection.

##### 4.3 Infinite-Sort $PN^2$ 's

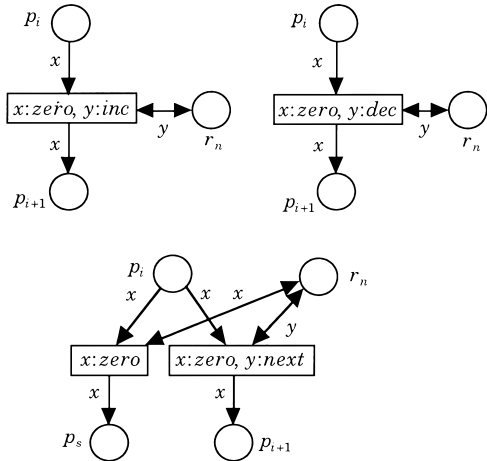
We show that the class of infinite-sort  $PN^2$ 's has the same level of modeling power as Turing machines. To prove this, *register machines* are used. A register machine is a computer-like machine with a number of registers which are used to store arbitrarily large numbers. A program is written to manipulate the registers. It was proved that a register machine with the following set of instructions is equivalent to Turing machines [11].

1.  $P(n)$ : increase register  $n$  by 1.
2.  $D(n)$ : decrease register  $n$  by 1 (only if register  $n$  is nonzero).
3.  $j(n)[s]$ : jump to statement  $s$  if register  $n$  is zero.

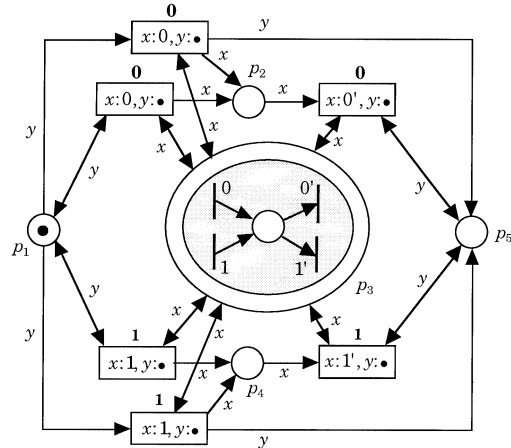
We now show that the register machine can be converted to an infinite-sort  $PN^2$ . We represents  $n$  registers used in a program by  $n$  places  $r_1, \dots, r_n$ . We also use  $m + 1$  places  $p_0, \dots, p_m$  to represent the program counter, where  $m$  is the number of statements in the program. In the initial configuration, we put a marked  $\ell$ -P/T net shown in Fig. 7 on  $p_0$  and also on each  $r_i$ ,  $i = 1, \dots, n$ . Each instruction in the program is represented by a transition shown in Fig. 8.

The most important part of this construction is the transition from  $p_i$  and  $r_n$  to  $p_s$ . Occurring of this transition are possible only when both nets in  $p_i$  and  $r_n$  are identical. Transition *zero* in the lower-level net

<sup>†</sup>A marked  $\ell$ -P/T net  $\mu$  is bounded if there exists a nonnegative integer  $k$  such that for every  $m \in \mathcal{R}(\mu)$  and every  $p \in P$ :  $m(p) \leq k$ .



**Fig. 8** Construction of a  $PN^2$  representing the register machine (2).



**Fig. 9** A  $PN^2$  accepting  $L_{ww^R}$ .

is always enabled because it has no input places, however its firing is restricted by the upper-level net. The marked  $\ell$ -P/T net in place  $p_i$  has no tokens, and the marked  $\ell$ -P/T net in place  $r_n$  has tokens corresponding to the value of the register. If the value is 0, then both nets are identical and only the left transition is enabled. Otherwise, only the right transition is enabled. This enables *zero testing*.

4.4 Accepting Languages

$PN^2$ 's can represent arbitrary large numbers with zero-testing ability. Given a marked  $\ell$ -P/T net  $\mu = (\pi^\ell, m_0)$ ,  $L(\mu) = \{\ell(\sigma) \mid \sigma \in T^*, \exists m \in F : m_0 \xrightarrow{\sigma} m\}$  is called the language accepted by  $\mu$ , where  $F$  is a finite set of accepting markings.

It was shown that  $L_{ww^R} = \{ww^R \mid w \in \Sigma^*\}$  cannot be accepted by any marked  $\ell$ -P/T nets [11]. It is a non-regular, context-free language. The reason is described as follows. Any marked  $\ell$ -P/T net cannot have  $k^r$  possible markings after firing  $r$  transitions, where  $k$  is a constant. This implies that any marked  $\ell$ -P/T net cannot simulate a pushdown stack.

We can show a  $PN^2$  that accepts  $L_{ww^R}$  (Fig. 9). Let  $\mu(i)$  denote the marked  $\ell$ -P/T net initially put on  $p_3$  with  $i$  tokens. The set of accepting markings is  $F = \{[\emptyset, \emptyset, \mu(0), \emptyset, 1]\}$ . By an input 0010, the configuration changes to

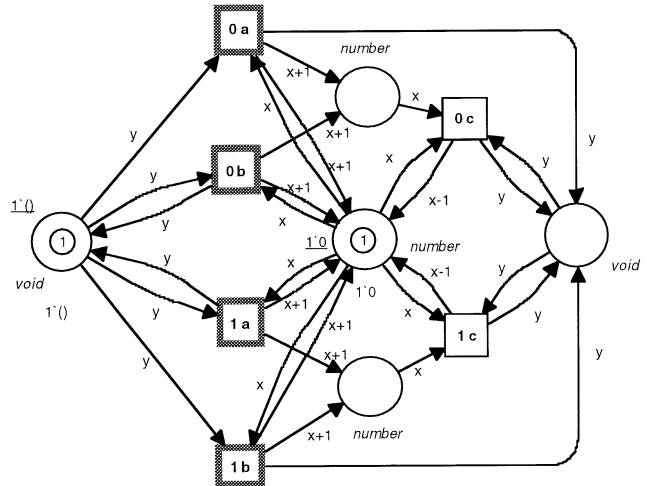
$$p_2 \mapsto \mu(1) + \mu(2) + \mu(4); p_3 \mapsto \mu(4); p_4 \mapsto \mu(3).$$

It realizes a pushdown stack.

There exists a coloured Petri nets [13] equivalent to the above  $PN^2$  (Fig. 10). Procedures for increment/decrement of the stack pointer are described as arc expressions in the coloured Petri net, while they are represented by transitions of the lower-level nets in the  $PN^2$ .

```

color number = int;
color void = unit;
var x : number;
var y : void;
    
```



**Fig. 10** A CPN accepting  $L_{ww^R}$ .

5. State Equation and Invariant Analysis of  $PN^2$ 's

In this section, we first define state equations of  $PN^2$ 's, and after that discuss invariant analysis using incidence matrices.

5.1 Injective- $PN^2$ 's

A  $PN^2$  is called *injective* if every marked  $\ell$ -P/T net in the initial configuration has an injective labeling function. For any  $PN^2$ , we can construct an equivalent injective- $PN^2$  by

- (i) Relabeling of transitions in the lower level nets if more than one transitions have the same label;

- (ii) Duplication of transitions in the upper level net corresponding to the relabeling.

Therefore, there are no differences in the modeling power between  $PN^2$ 's and injective- $PN^2$ 's.

In injective- $PN^2$ 's, we can omit the second part of transition bindings, therefore a transition binding for a transition  $t$  is defined by a function  $b : E_t \rightarrow PT_\Sigma$  such that for each  $e \in E_t$ , the unique transition having label  $e_\Sigma$  is enabled in  $b(e)$ . In what follows, we consider injective- $PN^2$ 's only to simplify the discussion. Moreover, we identify a label with the transition having the label.

### 5.2 State Equation

We show how the state equation of an injective- $PN^2$  is defined. We first define *firing count vectors*. Suppose that  $T = \{t_1, \dots, t_n\}$  and  $P = \{p_1, \dots, p_m\}$ . In what follows, we will use an index  $i$  to denote transitions, and  $j$  to denote places. In addition, we will write just  $i$  ( $j$ ) in subscripts to indicate the transition  $t_i$  (place  $p_j$ ), e.g., we will write  $E_i$  instead of  $E_{t_i}$ , and  $B_i$  instead of  $B_{t_i}$ .

A *firing count vector* is a  $|T|$ -dimensional column vector  $x = [x_1, \dots, x_n]^t$ , where each  $x_i$  is a multi-set over  $B_i$ , and it indicates how many times each transition binding to occur. In this formulation, it is necessary to consider all elements in  $\bar{B}$ , and  $|\bar{B}|$  may increase exponentially in the size of the net. We show this problem is avoidable by introducing a different representation of firing count vectors.

Let  $E_i = \{e_i^1, \dots, e_i^{|E_i|}\}$ . We represent each  $x_i$  by a  $|E_i|$  dimensional vector  $\hat{x}_i = [x_i^1, \dots, x_i^{|E_i|}]$  such that

- (i) for each  $k = 1, \dots, |E_i|$ :

$$x_i^k = \sum_{b \in B_i} x_i(b) \# b(e_i^k)$$

- (ii) for every  $k, k' \in \{1, \dots, |E_i|\}$ ,  $|x_i^k| = |x_i^{k'}|$ .

The second requirement is necessary for the vector to be decomposed into a multi-set over  $B_i$ . The reason why this compact representation of firing count vectors is possible is that there is no interference among transition components of each transition.

The following example shows how we can avoid to deal with the combinatorial number of transition bindings.

**Example 4:** Suppose that  $E_i = \{e_i^1, \dots, e_i^h\}$ ,  $e_{i\Sigma}^k = a$  ( $k = 1, \dots, h$ ), and the input place of  $t_i$  contains  $r$  marked  $\ell$ -P/T nets having an enabled transition with label  $a$  (Fig. 11). Then the number of possible transition bindings for  $t_i$  is  $O(r^h)$ . In the above representation, however, the space necessary to represent  $x_i$  is  $O(rh)$ .

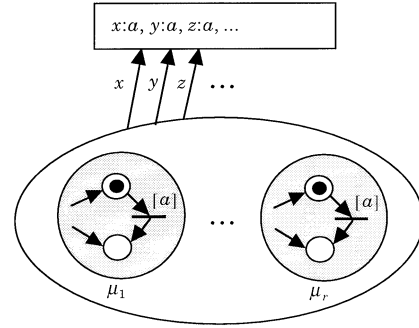


Fig. 11 Combinatorial number of transition bindings.

For a sequence of transition bindings  $\sigma \in B^*$ , let  $\psi(\sigma) = [x_1, \dots, x_n]^t$  denotes a firing count vector such that for each transition  $t_i$ ,  $x_i$  is the sum of transition bindings for  $t_i$  occurring in  $\sigma$ , and let  $\hat{\psi}(\sigma) = [\hat{x}_1, \dots, \hat{x}_n]^t$  denotes the different representation of  $\psi(\sigma)$  described above. The vector  $\hat{\psi}(\sigma)$  keeps sufficient information to determine the final marking.

**Lemma 1:** Suppose that  $s \xrightarrow{\sigma_1} s'$ ,  $s \xrightarrow{\sigma_2} s''$ , and  $\hat{\psi}(\sigma_1) = \hat{\psi}(\sigma_2)$ , then  $s' = s''$ .

We now define two matrices  $I^-$  and  $I^+$ , called the *input incidence matrix* and the *output incidence matrix*. Each of these contains a row for each place and a column for each transition. Each component  $I^-(p_j, t_i)$  ( $I^+(p_j, t_i)$ ) corresponds to  $\bullet e_i^k$  ( $e_i^k \bullet$ ) of  $e_i^k \in E_i$ , and is defined as follows:

- $I^-(p_j, t_i)$  is a row vector

$$[w_{j,i,1}^- \# Id, \dots, w_{j,i,|E_i|}^- \# Id],$$

where  $w_{j,i,k}^- \in \mathbb{N}$ , and  $Id$  is the identity function.

- $I^+(p_j, t_i)$  is a row vector

$$[w_{j,i,1}^+ \# \delta_{e_{i\Sigma}^1}, \dots, w_{j,i,|E_i|}^+ \# \delta_{e_{i\Sigma}^{|E_i|}}],$$

where  $w_{j,i,k}^+ \in \mathbb{N}$ , and  $\delta_\tau$  is a function on  $\mathcal{M}_{PT_\Sigma}$  such that  $\delta_\tau(M) = \sum_{\mu \in PT_\Sigma} M(\mu) \# \delta(\mu, \tau)$ .

Multiplication between incidence matrices and a firing count vector is defined as follows.

$$I^-(p_j, t_i) \cdot \hat{x}_i = \sum_{k=1}^{|E_i|} w_{j,i,k}^- \# Id(x_i^k) = \sum_{k=1}^{|E_i|} w_{j,i,k}^- \# x_i^k,$$

$$I^+(p_j, t_i) \cdot \hat{x}_i = \sum_{k=1}^{|E_i|} w_{j,i,k}^+ \# \delta_{e_{i\Sigma}^k}(x_i^k).$$

A configuration  $s$  is represented by a column vector  $[s_1, \dots, s_m]^t$  such that each  $s_i$  is a multi-set over  $PT_\Sigma$ . Hence, we obtain the following state equation.

**Proposition 2:** if  $s \xrightarrow{\sigma} s'$ , then  $s' = s + I^+ \hat{\psi}(\sigma) - I^- \hat{\psi}(\sigma)$ .





**Example 7:** A  $T$ -invariant of the  $PN^2$  in Fig. 1 is

$$\hat{x} = [x_1^1 = \mu_1, x_1^2 = \mu_2, x_2^1 = \mu'_2, x_2^2 = \mu'_1, x_2^3 = \mu_2]^t.$$

In the nonnegative integer matrix representation,

$$\hat{x} = [1, 1 \mid 1 \mid 1, 1]^t.$$

In P/T nets, a  $P$ -invariant is a vector of weights so that the weighted sum of the number of tokens in each place is constant by any occurrence of transitions.  $P$ -invariants of an injective- $PN^2$  are similarly defined. Let  $x_b$  denotes the firing count vector of a single occurrence of transition binding  $b$ . A *weight function*  $w$  is a function on  $\mathcal{M}_{PT_\Sigma}$  such that

$$w \left( \sum_{\mu \in PT_\Sigma} n_\mu \# \mu \right) := \sum_{\mu \in PT_\Sigma} c_\mu \cdot n_\mu \# \mu, \quad (c_\mu \in \mathbb{N}).$$

A  $P$ -invariant is a row vector  $y = [y_1, \dots, y_m]$  of weight functions such that for every  $b \in \bar{B}$ :

$$y \cdot (I^+ \cdot x_b - I^- \cdot x_b) = 0.$$

In the integer matrix representation, each  $y_i$  is represented by an integer vector  $[y_i^1, \dots, y_i^{\#all}]$ , and  $x_b$  by a nonnegative integer vector

$$[[x_1^1, \dots, x_1^{|E_i|}], \dots, [x_n^1, \dots, x_n^{|E_i|}]]^t$$

such that each  $x_i^k$  is an  $\#_{i,k}$ -dimensional unit vector, i.e., it has value 1 for one component and 0 for others.

The number of transition bindings in  $\bar{B}$  is  $\sum_{i=1}^n \prod_{k=1}^{|E_i|} \#_{i,k}$ . However, it is not necessary to solve the equation for all of the transition bindings. Let  $I_i^k$  denote the  $\#_{all} \times \#_{i,k}$  submatrix of  $I$  corresponding to  $e_i^k$ , and let  $I_i^{k,l}$  denote the  $l$ -th column vector of  $I_i^k$ . Then a vector  $y$  of weights is a  $P$ -invariant if and only if:

- (i)  $y \cdot \sum_{k=1}^{|E_i|} I_i^{k,1} = 0 \quad (i = 1, \dots, n)$ ;
- (ii)  $y \cdot I_i^{k,l} = y \cdot I_i^{k,l+1} \quad (i = 1, \dots, n, k = 1, \dots, |E_i|, l = 1, \dots, \#_{i,k} - 1)$ .

We can easily construct an integer matrix  $I_{Pinv}$  such that  $y \cdot I_{Pinv} = \mathbf{0}$  is equivalent to the conditions (i) and (ii). The size of  $I_{Pinv}$  is  $m \cdot \#_{all} \times (n + \#_{sum} - \sum_{i=1}^n |E_i|)$ .

**Example 8:** A  $P$ -invariant of the  $PN^2$  in Fig. 1 is

$$y = [1, 0, 0, 0 \mid 0, 1, 0, 0 \mid 0, 0, 1, 1],$$

because

$$y \cdot (I_1^{1,1} + I_1^{2,1}) = y \cdot I_2^{1,1} = y \cdot (I_3^{1,1} + I_3^{2,1}) = 0.$$

Note that the condition (ii) is not necessary since  $\#_{i,k} = 1$  for all  $i, k$ .

## 6. Discussion

As written in the introduction, the simplest way to represent agents in Petri-net-based models is to introduce notion of *objects*, i.e., a collection of data types with methods that give access to the data, into attributes of tokens. Representation of an agent as an object is necessary to describe the agent independently of other agents and environments where agents interact. Since most of classes of object Petri nets are based on high-level Petri nets, they are too complex to be analyzed. There exists a trade-off between expressiveness and analysis. The aim of proposing  $PN^2$  is to give a model that is sufficiently simple for the mathematical analysis, but has enough modeling power.

The purpose of *elementary object systems* (EOS) [9] is similar to that of  $PN^2$ 's, i.e., to model systems in an object oriented manner keeping the model as simple as possible. We make a comparison between  $PN^2$ 's and EOS.

**Modeling power:** Infinite-sort  $PN^2$ 's can have infinitely many states, and has the same modeling power as Turing machines, while EOS do not allow to have infinitely many states. Each lower-level nets in EOS is an elementary net system, i.e., it has finitely many reachable markings, and duplication of agents is not allowed in the upper-level net.

**Agent communication:** In  $PN^2$ 's, communication links are defined by the function  $\cdot_\Sigma$ , which defines links by specifying *labels* of lower-level nets. In EOS, it is necessary to specify the interaction relation, which is a binary relation between the set of transitions in the upper-level net and the set of transitions in lower-level nets. Both representations are equivalent except that communication links in EOS are restricted to those between two agents.  $PN^2$ 's allow links in which more than two agents participate.

EOS also provide another type of interaction between two objects in the same place, called object/object interaction. Communication links for this interaction is defined by a symmetric relation on the set of transitions in lower-level nets. In  $PN^2$ 's, the same type of interaction can be represented by adding transitions with self-loops to the place.

In EOS, the object/object interaction relation is universal in all places of the upper-level net, while  $PN^2$ 's can describe different interaction relation in each place.

**Semantics:**  $PN^2$ 's are based on the value semantics, whereas EOS are basically based on the reference semantics<sup>†</sup>. As a result, enabling of transitions in EOS is restrictive. It does not allow duplication of agents, but  $PN^2$  can do.

<sup>†</sup>Relationship between value semantics and reference semantics in EOS is discussed in [10]

**Mathematical analysis:** The main purpose of proposing EOS is to give formal semantics of object systems, and the research direction is different from that of  $PN^2$ 's. Therefore, no work has been done on the mathematical analysis. On the other hands,  $PN^2$ 's are designed to enable mathematical analysis on numerical matrices such as invariant analysis. In addition, allowing dynamic bindings of transitions may cause a combinatorial number of transition bindings, but it is avoidable in the invariant analysis.

Process algebra is an alternative approach to model concurrent systems. Several models based on process algebra have been proposed to represent mobility of agents, such as  $\pi$ -calculus [14], *CHOCS (Calculus of Higher Order Communicating Systems)* [15], and *ambient calculus* [16].  $\pi$ -calculus realizes the mobility of agents by movement of links between processes. Each process can send link names to other processes, and communication links between agents are dynamically constructed. CHOCS is a higher order process algebra. In CHOCS, each process can send and receive processes. This feature enables to represent mobile codes in a distributed environment. An ambient is a bounded place where agents interact. In ambient calculus, each agent can enter into or exit from an ambient, and can also dissolve an ambient. Communication between agents are described as the movement of messenger agents.

These models are oriented to expressiveness of agent-based systems. In contrast to these models,  $PN^2$ 's represent the mobility by movement of agents (lower-level nets) in a virtual environment (the upper-level net), and are oriented to mathematical analysis such as the state equation and invariant analysis. Existence of invariants are necessary for the system to be stable. Such system-theoretic view is important in the design of large and complex systems.

## 7. Conclusion

The proposed model  $PN^2$ 's are one of simplest Petri-net-based models for representing the following features of multi-agent systems: internal states of an agent, dynamic interaction between agents, and multiple environments in which agents act.

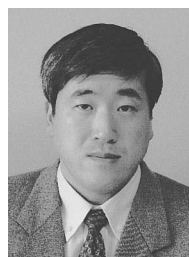
This paper is just a proposal. In the future, we will study theoretical analysis on the model and applications such as JAVA based software.

## References

- [1] M. Baldassari, "An environment for object-oriented conceptual programming based on PROT Nets," Lecture Notes in Computer Science, vol.340, pp.1–19, 1988.
- [2] E. Battiston, F. De Cindio, and G. Mauri, "OBJSA nets: A class of high-level nets having objects as domains," Lecture Notes in Computer Science, vol.340, pp.20–43, 1988.
- [3] O. Biberstein, D. Buchs, and N. Cuelfi, "CO-OPN/2—A specification language for distributed system engineering,"

Technical Report 96/167, Software Engineering Laboratory, Swiss Federal Institute of Technology, 1996.

- [4] O. Kummer and F. Wienberg, "Renew—The reference net workshop," Petri Net Newsletter, no.56, pp.12–16, 1999.
- [5] C. Lakos, "From coloured Petri nets to object Petri nets," Lecture Notes in Computer Science, vol.935, pp.278–297, 1995.
- [6] T. Miyamoto and S. Kumagai, "A multi agent net model of autonomous distributed systems," Proc. CESA'96, Symposium of Discrete Events and Manufacturing Systems, pp.619–623, 1996.
- [7] S. Philippi, "System modeling using object-oriented Pr/T-nets," Research Report no.25/97, Institute for Computer Science, University Koblenz-Landau, 1997.
- [8] C. Sibertin-Blanc, "Cooperative nets," Lecture Notes in Computer Science, vol.815, pp.471–490, 1994.
- [9] R. Valk, "Petri nets as token objects—An introduction to elementary object nets," Lecture Notes in Computer Science, vol.1420, pp.1–25, 1998.
- [10] R. Valk, "Relating different semantics for object Petri nets," Research Report FBI-HH-B-226/00, Faculty of Informatics, University of Hamburg, 2000.
- [11] J.L. Peterson, Petri net theory and the modeling of systems, Prentice-Hall, 1981.
- [12] P. Ramadge and W.M. Wonham, "The control of discrete event systems," Proc. IEEE, vol.77, no.1, pp.81–98, 1989.
- [13] K. Jensen, Coloured Petri nets: basic concepts, analysis methods and practical use, Volume I, II, III, Springer-Verlag, 1992, 1995, 1997.
- [14] R. Milner, Communicating and mobile systems: The  $\pi$ -calculus, Cambridge University Press, 1999.
- [15] B. Thomsen, "A theory of higher order communicating systems," Information and Computation, vol.116, pp.38–57, 1995.
- [16] L. Cardelli and A.D. Gordon, "Mobile ambients," Lecture Notes in Computer Science, vol.1378, pp.140–155, 1998.



**Kunihiko Hiraishi** received from the Tokyo Institute of Technology the B.E. degree in 1983, the M.E. degree in 1985, and D.E. degree in 1990. In 1985 he joined the IAS-SIS, Fujitsu Limited. Since 1993 he has been an Associate Professor at Japan Advanced Institute of Science and Technology. His current interests include discrete event systems and computational models for concurrent systems. He is a member of the IEEE, IPSJ, and SICE.