LC-Petri Net expanded to be suitable for making FMS model *

Katsumi WASAKI** Yasushi FUWA*** Masayoshi EGUCHI****
Yatsuka NAKAMURA ***

In this paper, we propose an extension of Petri nets (LC-Petri net : LCPN) suitable for FMS design and discuss its methods of evaluation. This LCPN is a marked net with addition of the following features: data assignment of marks, representation of firing conditions as logic equations, coupling of output procedures with transition firing, etc. Also, since the concept of transition firing evaluation orders is omitted from the analysis of conventional Petri nets, we introduce this concept formally in our proposed net. Finally, in order to study the behavior of a system modelled with this net, we provide a means for searching the reachability tree of markings. This LCPN is a extended Petri nets which solve a problem of description from Place/Transition Petri nets and Colored Petri nets before.

Keyword : coloured Petri net, extended Petri net, control software, system stability, CASE tools

1 Introduction

There has been a rapid expansion recently in factory automation (FA), making it necessary to organize flexible manufacturing systems (FMSs), composed of robots, transfer machines, automatic storage systems, and image processing systems, for quality control. In general, FMS controllers have to control multiple manufacturing units which work together concurrently and asynchronously while maintaining operation synchronization. Programmable manufacturing controllers (PMCs) have been used as FMS controllers for advanced FA systems and are programmed using sequence ladder languages/diagrams, state charts or decision tables. However, these method have the following problems[5]:

(1) A variety of programming errors and bugs are easily introduced.
(2) Verification of the correctness of specifications is difficult.
(3) Tracing control flow/software is difficult by anyone other than the original programmers.

A descriptive model of sequence control by Petri nets has become attractive due to its simplicity [1][2]. Petri net is a graphical model which makes understanding the control flows easy. The mathematical nature of the Petri net can be used to obtain information about the behavior of systems which operate in a dynamic environment. Especially when time or safety factors make actual simulations of a system unfeasible, a study of the relationships between the mathematical objects of a Petri net can reveal such conditions as deadlocks, traps, reachability of marking states, etc. which aid in the verification of system operations.

In the design of parallel systems, many modelling techniques adopting the concept of Petri net have been announced. The reason for this is that it is suitable for describing the characteristics of a parallel system and once a Petri net model has been created, it can be analyzed easily. When designing systems whose characteristics do not depend on time, an effective approach is the use of place/transition nets (PNs). When the degree of complexity of the target system increases, colored Petri nets (CPNs) [3], or high-level Petri nets, are used in design.

Examples of applying old PNs and CPNs to system design and analysis have been given in such areas as hardware design by Jensen [3], deadlock verification of Ada programs by Murata et al [4], and FA control by Nagao et al [5].

The following problem exists when PNs and CPNs are actually applied to the work of describing control systems. The operation of these nets (firing conditions of transition and movement of tokens) is

* Retouch correction of ICARCV'96 ; December 3-6,1996
** Department of Electronics and Control Engineering
*** Faculty of Engineering, Shinshu University
**** Tokyo University of Mercantile Marine
A part of this research was accomlished by the education research special expenditure of 1996 term.
Received October 31, 1997

uniquely fixed. It is necessary to use many transition and place elements to represent the branching of conditions in the processing and as a result, the net scale increases.

This paper proposes the concept of a logical colored Petri net (LCPN) [6]. The LCPN is a PN with the following improvements. Tokens have data (colors) and firing conditions are given by an arbitrary logical expression which is written in terms of the presence of tokens in input places and the data values of tokens. The token output at firing is decided by using a function based on the data values of tokens in input places. Therefore, transitions and places which were added as a result of the above-mentioned condition branching are unnecessary and the net scale can be reduced [7].

Software which controls an actual FMS is modeled by an LCPN. Next, the designed net model is analyzed by using a reachability tree. The net model made for the system is implemented in hardware and software, the operation is confirmed, and the perpetual change of the LCPN is verified.

We used this technique to develop software for a PMC for image processing controller in a FMS factory with LCPNs in order to estimate their effectiveness. A desktop personal computer was used as a development tool for LCPNs in laboratory/office settings and a laptop personal computer was used in the manufacturing rooms.

The use of LCPNs reduced the cost of design, programming and testing on site to one third of what development would have cost using conventional procedural language-based methods.

2 Logical Colored Petri Net (LCPN)

In this chapter, we give the definition of a logical colored Petri net (LCPN) which can be used to improve the description capabilities of old PNs and CPNs.

2–1 Definition of LCPN

Definition 1 *LCPN : A logical colored Petri net is a set* $N_E = (S_E, T_E, F_E, M_E)$ *which fulfills each of the following conditions.*

(i) $S_E = \{s_1, s_2, \cdots, s_n\}$ *and* $T_E = \{t_1, t_2, \cdots, t_m\}$ *are sets of place and transition elements, respectively.* $F_E = \{f_1, f_2, \cdots, f_l\} \subseteq (S_E \times T_E) \cup (T_E \times S_E)$ *is a set of arcs from places to transitions and transitions to places.*

(ii) *The mark of each place* $s_i \in S_E$ $(i = 1, 2, \cdots, n;$ $n = \mid S_E \mid)$ *can take the value of the natural number from 1 to N. This is written* $\mu(s_i)$. *We assume* $\mu(s_i) = 0$ *if there is no mark in* s_i.

$$\mu : S_E \longrightarrow \{0, 1, 2, \cdots, N\}$$

(iii) *The capacity (maximum number of marks) of each place* s_i *is 1. The set of all possible marks from (1) and (2) is represented as a mapping from* S_E *to* $\{0, 1, 2, \cdots, N\}$.

$$M_E \equiv \{0, 1, 2, \cdots, N\}^{S_E}$$

(iv) ${}^*t_j$ *represents the set of all places (input places) which have an arc extending to* $t_j \in T_E$ $(j = 1, 2, \cdots, m; m = \mid T_E \mid)$. *Similarly,* $t_j{}^*$ *represents the set of all places (output places) with an arc extending from* t_j.

$$\begin{aligned} {}^*t_j &= \{s \in S_E : (\exists f)(f \in F_E, f = (s, t_j))\} \\ t_j{}^* &= \{s \in S_E : (\exists f)(f \in F_E, f = (t_j, s))\} \end{aligned}$$

(v) *The firing evaluation of a transition* t_j *for an arbitrary marking* $\mu \in M_E$ *examines the firing condition* Φ^j. $\Phi^j(\mu \mid {}^*t_j)$ *is described by a logical expression in terms of the state* $\mu \mid {}^*t_j$ *of the places which belong to* ${}^*t_j$ *and is used to determine the next marking* $\mu' \in M_E$.

t_j *is called firable if* Φ^j *is evaluated to be true. If* t_j *is fired, a mark is removed from each place in* ${}^*t_j - t_j{}^*$. *Places in* $t_j{}^*$ *depend on the state of* ${}^*t_j$ *and are modified by the following* C^j.

$$C^j : \{0, 1, 2, \cdots, N\}^{{}^*t_j} \longrightarrow \{0, 1, 2, \cdots, N\}^{t_j{}^*}$$

if $\Phi^j(\mu \mid {}^*t_j)$ *is true then*

$$\mu' = \begin{cases} 0 & : on \ {}^*t_j - t_j{}^* \\ C^j(\mu \mid {}^*t_j) & : on \ t_j{}^* \\ \mu & : otherwise \end{cases}$$

On the other hand, t_j *is not firable if* Φ^j *is false. At this time, the state* $\mu' = \mu$ *is unchanged.*

We can show the next marking resulting from a transition evaluation as a mapping f^j from state $\mu \in M_E$ of the places to $\mu' \in M_E$. □

The notable characteristics of this firing evaluation are found in two points. First, the conditions of firing can be given by an arbitrary logical expression and second, the next marking can be decided freely according to the previous state of marks. This greatly simplifies the work of expressing diverging conditions.

2–2 Introduction of Firing Evaluation Orders

This section introduces the concept of order in firing evaluation in LCPNs and analyzes the movement of the modelled system. When an LCPN is implemented in software and hardware, the order of evaluating each transition is uniquely decided by distributing the clock for calling the associated control procedure or operation. Therefore, when different evaluation orders producing different system behaviors is a problem, this becomes an effective solution. In a previous work, we have discussed the issues of Petri net stability and deadlock verification when considering firing evaluation orders using a subclass of PNs called the Boolean Marking Net (BMN) [8][9].

Definition 2 *The definition of the order of firing evaluation of transitions is introduced into LCPNs below.*

(i) *G is a permutation group of $\{1, 2, \cdots, m\}$ ($m = \mid T_E \mid$). It is used to decide the order of firing evaluation $\{t_1, t_2, \cdots, t_m\}$ of transitions and each element $g \in G$ corresponds to infinite sequence $t_{g(1)}, t_{g(2)}, \cdots, t_{g(m)}, t_{g(1)}, t_{g(2)}, \cdots, t_{g(m)}, \cdots$ representing a cyclic firing evaluation order.*

(ii) *The firing evaluation of transition t_j ($j = 1,2,\cdots,m : m = \mid T_E \mid$) is shown by a mapping f^j from state $\mu \in M_E$ to $\mu' \in M_E$.*

(iii) *The order of firing evaluation concerning m transitions is based on $g \in G$.*

$$f^{g(1)g(2)\cdots g(m)} = f^{g(m)} \times \cdots \times f^{g(2)} \times f^{g(1)}$$

(iv) *When one cycle g of a firing evaluation for m transitions is given, we write is as:*

$$g : t_{g(1)} \rightarrow t_{g(2)} \rightarrow \cdots \rightarrow t_{g(m)}$$

This is abbreviated as follows.

$$g = (g(1),\ g(2),\ \cdots,\ g(m))$$

□

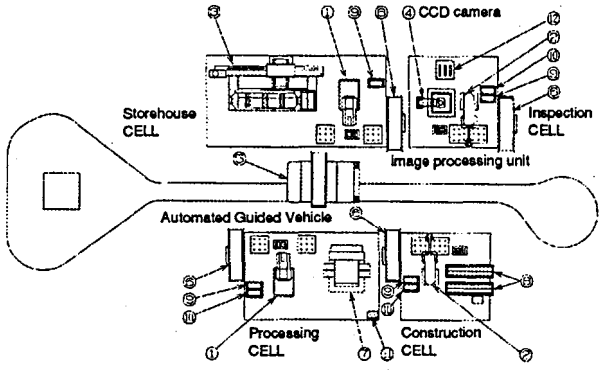
3 Modelling of PMC Control Software by LCPNs

3–1 Outline of FMS System and PMC control

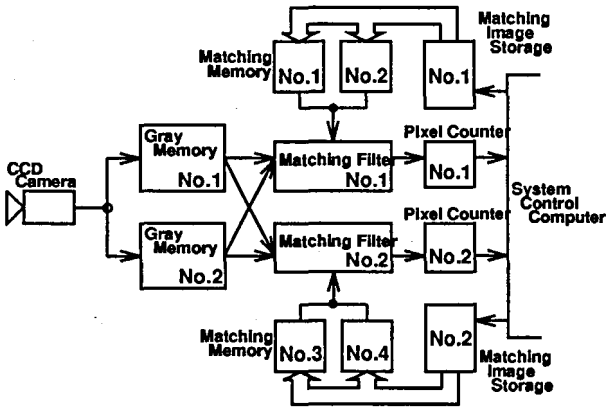
Figure 1(a) shows the equipment layout of the FMS system. This control system is composed of four process CELLs, an Automated Guided Vehicle(AGV), an image processing unit, a CCD-camera, and a system control computer.

First, the surface of an object placed on the XY-stage is divided into small square areas called cells. Each cell is captured by a CCD-camera one after another as raw image. The raw image data is compared with several template images prepared beforehand and the quality of the object is inspected. For example, when the object is a printed pattern, the template images consist of the correct patterns to appear in each cell. Parts not corresponding to the template are detected and reported to the control computer as the number of incorrect pixels.

Figure 1(b) shows the internal hardware composition of the image processor. This device is composed of a template image storage area for holding multiple template data; two template image memory areas (A, B); a matching filter; a pixel counter; and two raw image memory areas (A, B) for storing the image of each cell taken from the CCD-camera. By equipping the system with a dual memory configuration, one pair of raw image data and template data can be matched and the next image can be read in at the same time. Moreover, since an independent data bus connects the template image storage and the template image memory, forwarding from storage to memory can be performed independently. The control of these parallel processes is handled by software in the system control computer.



(a) The equipment layout of FMS



(b) The internal hardware composition of image processing unit

Fig. 1. The FMS system.

3–2 Example of Designing Control Systems with LCPNs

Figure 2 shows an example of designing a control system with LCPNs based on the following PMC processing sequence.

Each process in this sequence, i.e., reading of raw images, forwarding template images, and matching, is expressed with a sub-net in the figure as Subnet-1,2,3, respectively. The operation of these sub-nets starts by sending a token to s_1, s_4, and s_6 which are at the entrance of each net. The movement of each net is as follows and the number of the inspected cell is indicated by N in the explanation.

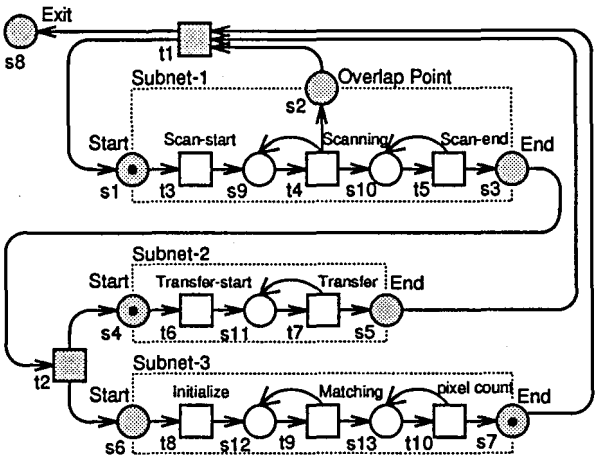


Fig. 2. An example of designing a control system with LCPNs based on the processing sequence.

3–2–1 Subnet-1: Reading of Raw Images

Subnet-1 performs the work of taking raw images from the camera into the raw image memory. Transition t_3 fires when a token with value $1 \sim N$ enters place s_1 and this places a token in s_9. The value of this token shows the position of the captured image. The actual image capture occurs when t_4 fires and a token updated with the position of the next reading is returned to s_9. Also, when the updated position reaches the overlap point, tokens are output to s_2 and s_{10}. Readings after the overlap point are performed by the firing of t_5 and when the value of the token reaches the last cell, a token is output to s_3. Finally, once the overlap point has been passed by the firing of t_5, it is possible for another token to enter s_1 again and the next cell image can be read in by the firing of t_4.

3–2–2 Subnet-2: Template Image Forwarding

Subnet-2 performs the work of forwarding template images from storage to the template image memory. Transition t_6 fires when a token with value $1 \sim N$ enters place s_4 and a token is placed in s_{11}. The value of this token shows the memory address in template image storage in the forwarding source. The actual forwarding process occurs when t_7 fires and a token updated with the value of the next mem-

ory address in the forwarding source is returned to s_9. When the forwarding process ends, it outputs a token to s_5.

3-2-3 Subnet-3: Matching Process and Pixel Counter Processing

Subnet-3 performs the work of comparing images and writing the results in the pixel counter. Transition t_8 fires when a token with value $1 \sim N$ enters place s_6 and this places a token in s_{12}. The value of this token shows the position of the image to be processed in the matching and pixel counting. When t_9 fires, the actual matching takes place and a token is returned to s_{12}. Once the matching process is complete, a token is output to s_{13} and transition t_{10} fires. At this point, the pixel counter processing is performed and a token is returned to s_{13}. Once the processing is completed, a token is output to s_7.

3-2-4 Synchronization of Subnet-1,2,3

To synchronize the operation of Subnet-1,2,3, the places at the entrance and exit points of each net are connected by synchronization transitions t_1 and t_2. The values $1 \sim N$ of the tokens stored in these places show the number of the cell processed by each net.

When the overlap point is reached during the reading of raw image data, t_1 prepares to read the next cell. It also includes processing for termination when all inspections are complete. The firing condition has two parts, $\Phi^1_{overlap}$ and Φ^1_{exit}, which have the roles of starting the reading process of the next cell and terminating operation, respectively. t_1 is defined as follows.

$$
\begin{aligned}
{}^*t_1 &= \{s_2, s_5, s_7\}\ ,\ t_1{}^* = \{s_1, s_8\} \\
\Phi^1_{overlap}(\mu|{}^*t_1) &\equiv (0 < \mu(s_2) < N) \\
&\quad and\ (\mu(s_5) \neq 0) \\
&\quad and\ (\mu(s_7) \neq 0) \\
C^1_{overlap}(\mu|{}^*t_1) &\equiv \begin{cases} \mu(s_2)+1 & : on\ s_1 \\ 0 & : on\ s_8 \end{cases} \\
\Phi^1_{exit}(\mu|{}^*t_1) &\equiv (\mu(s_2) = N)\ and\ (\mu(s_7) \neq 0) \\
C^1_{exit}(\mu|{}^*t_1) &\equiv \begin{cases} 0 & : on\ s_1 \\ \mu(s_2) & : on\ s_8 \end{cases}
\end{aligned}
$$

Transition t_2 is used to perform template image forwarding and matching after the reading of raw images is complete. If the last image has been read, no transferring of next templates is needed and only the matching process is started. The firing condition has two parts: 1) Φ^2_{normal}, which is responsible for starting template forwarding and matching operations and 2) Φ^2_{final}, which initiates matching only. t_2 is defined as follows.

$$
\begin{aligned}
{}^*t_2 &= \{s_3\}\ ,\ t_2{}^* = \{s_4, s_6\} \\
\Phi^2_{normal}(\mu|{}^*t_2) &\equiv (0 < \mu(s_3) < N) \\
C^2_{normal}(\mu|{}^*t_2) &\equiv \begin{cases} \mu(s_3)+1 & : on\ s_4 \\ \mu(s_3) & : on\ s_6 \end{cases} \\
\Phi^2_{final}(\mu|{}^*t_2) &\equiv (\mu(s_3) = N) \\
C^2_{final}(\mu|{}^*t_2) &\equiv \begin{cases} 0 & : on\ s_4 \\ \mu(s_3) & : on\ s_6 \end{cases}
\end{aligned}
$$

The scale of the net for realizing this control system covered 3 sub-nets, 10 transitions, and 13 places as shown in Figure 2.

4 Analysis and Operation Testing

4-1 Analysis of LCPN

Whenever a plate arrives, the image processing system should be able to perform processing repeatedly in a stable manner. For this, it is essential that there are no transitions which cannot fire in the system and that the state of markings is maintained when the system performs certain constant repeated operations. For this reason, we generate a reachability tree from a certain initial marking to analyze the net.

Each marking M_i $(i \in \{0, 1, 2, \cdots, 17\})$ in the reachability tree is represented in terms of the number of tokens (0/1) of each place s_i.

$$M_i = (|s_1|, |s_2|, |s_3|, |s_4|, |s_5|, |s_6|, |s_7|, |s_8|)$$

Next, we modified t_1 and added a new place s_{14} reason why detect a situation of the system falls. The abnormal situation occurs when there is a token in s_2 and no token in s_5 or s_7. The firing condition and next marking of t_1 are changed as follows.

$$\Phi^1_{error}(\mu|^* t_1) \equiv (\mu(s_2) \neq 0) \quad and \quad ((\mu(s_5) = 0)\ or\ (\mu(s_7) = 0))$$

$$C^1_{error}(\mu|^* t_1) \equiv \begin{cases} 0 & : on\ s_1, s_8 \\ \mu(s_2) & : on\ s_{14} \end{cases}$$

We generated the reachability tree for the corrected net and confirmed that processing stops when the above-mentioned abnormality occurs.

4–2 Implementation and Operation Testing of the Control Computer

We implemented the modelled control software on the control computer by using LCPNs and confirmed its operation.

The image processing sequence was specified in terms of LCPN, but the communication with XY-stage controller, which requires high-speed processing and the basic procedures for numerical computations and comparisions as well as screen display were written with conventional C programming language. We used a personal computer from development to field-testing with a Petri net tool which operated on this platform for trial purposes. This tool can consistently perform test operations on (1) LCPN modeling, (2) net verification, (3) basic procedure linking, and (4) on-site simulations.

By using the concept of LCPN, a great cost reduction was achieved in the implementation of parallel processing systems over older development methods which lack in verification capabilities. This is a result of the following four points.

(1) When designing parallel systems, the specification can be made simply with LCPN.
(2) The stability of the system can be verified sufficiently by net analysis.
(3) The control program can be united with basic customized procedures.
(4) The real-time operation of the system can be confirmed visually.

5 Conclusion

We proposed and defined the concept of a logical colored Petri net (LCPN) which improves the description capability of current PNs and CPNs. We also introduced the concept of the transition firing evaluation orders into LCPNs. We modelled software for controlling an image proessing system by using LCPNs and analyzed the net model with a reachability tree. We implemented the control software, generated from a net whose stability was verified, and on the control computer of the system and confirmed its operation.

In the future, we plan to study further the operation of LCPN and improve it as we apply it to CASE tools for developing control systems.

Acknowledgements. A part of this study was accomlished by the education research special expenditure of 1996 term in Nagano National College of Technology. They wish to thank related all of you for providing the opportunity for this research.

References

[1] T.Murata : "Petri Nets: Properties, Analysis and Applications ", Proc. IEEE, Vol.77, No.4, pp.541–580, 1989.

[2] J.L.Peterson : "Petri Net theory and the Modelling of Systems" , Prentice-Hall, Inc., 1981.

[3] K.Jensen : "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", Vol.1, Basic Concepts, Springer–Verlag, 1992.

[4] T.Murata, B.Shenker and S.M.Shatz : "Detection of Ada Static Deadlocks Using Petri Net Invariants" IEEE Trans. Softw. Eng., Vol.15 , No.3 , pp.314–326, 1989.

[5] Y.Nagao, H.Ohta, H.Urabe and S.Kumagai : "Petri Net Based Programming System for FMS" IEICE Trans. Fundamentals. , Vol.E75–A, No.10, pp.1326–1334, 1992.

[6] K.Wasaki, Y.Fuwa, M.Eguchi and Y.Nakamura : "Extended Petri Nets for Control System Software ", Technical Report of IEICE, COMP93-12, SS93-6, pp. 37-44, 1993.

[7] K.Wasaki, Y.Fuwa, M.Eguchi and Y.Nakamura : "Capacity of the Logical Colored Petri Net (LC-net) as a CASE Tool", Technical Report of IEICE, CAS93-69, pp. 101-108, 1993.

[8] P.N.Kawamoto, Y.Fuwa and Y.Nakamura: "Basic Concepts for Petri Nets with Boolean Markings" , Journal of Formalized Mathmatics, Vol.4, No.1, 1993.

[9] P.N.Kawamoto, M.Eguchi, Y.Fuwa and Y.Nakamura : "The Detection of Deadlocks in Petri Nets with Ordered Evaluation Sequences", Technical Report of IEICE, SS92-29, KBSE92-50, pp. 45–52, 1993.