The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

Implementation of real-time distributed control for discrete event robotic manufacturing systems using Petri nets

G. Yasuda

Nagasaki Institute of Applied Science, Nagasaki 851-0193, JAPAN
(Tel & fax : 81-95-839-8973)
(yasuda_genichi@pilot.nias.ac.jp)

Abstract: The paper deals with a systematic method of modeling and real-time control for discrete event robotic manufacturing systems using Petri nets. Because, in the complex manufacturing systems, the controllers are distributed according to their physical structure, it is desirable to realize real-time distributed control. In this paper, the task specification of discrete event manufacturing processes is represented using a global Petri net. Then it is decomposed and distributed into the machine controllers, which are coordinated through communication between the coordinator and machine controllers. The coordination algorithm was derived using formal expressions of transition firing and state change so that the decomposed transitions fire simultaneously. Implementation of real-time distributed control using a microcomputer network is described for an example robotic manufacturing cell. The demonstrations show that the proposed method can be used as an effective tool for consistent modeling, simulation and real-time control of large and complex manufacturing systems.

Keywords: Discrete event robotic manufacturing systems, modeling, real-time control, implementation, Petri nets.

I. INTRODUCTION

As the factory automation systems have been becoming larger and more complex. The system architecture has shifted to distributed and parallel processing from centralized processing in order to reduce the development cost and to improve the reliability. In the field of factory automation systems, a demand for the automatic control has diversified and the control logic has become extremely complicated, because the combinative complexity of the control requirement, which comes from the non-deterministic features of event driven systems such as manufacturing control systems, is inevitable. To deal with the complexity, a new methodology on control system design based on the concept of event driven system is necessary. However, appropriate representation methods and analysis methods for control mechanism have not sufficiently been established.

Programming paradigm modeled by a network, such as Petri net, has been considered to be useful, because the network model can describe the execution order of parallel/sequential processes directly without ambiguity. The Petri net is excellent in expression and analysis of the dynamic behavior of event driven systems, because it can model the system that consists of simultaneous process elements that interfere to each other. The programming technique makes it possible to realize systematic and high-level description of system specification. Therefore, it has been applied to a variety of system developments such as real-time systems, production systems, communication systems, and so on.

However, in case of factory automation systems, the network model becomes complicated and it lacks for the readability and comprehensibility. Besides, only specification analysis stage has been supported, and the support for the control software coding stage is insufficient [1]. Therefore, the flexibility and expandability are not satisfactory in order to deal with the specification change of the system.

Due to its complexity, a large and complex manufacturing system is commonly structured into a hierarchy; factory, line, station, machine, and actuators. The manufacturing system handles complicated tasks by dividing a task hierarchically in this structure, which is expected to be effective in managing cooperation tasks executed by great many machines or robots. Conventional Petri net based control systems were implemented based on an overall system model. Since in the large and complex systems, the controllers are geographically distributed according to their physical (hardware) structure, it is desirable to realize the hierarchical and distributed control. If it can be realized by Petri nets, the modeling, simulation and control of large and complex discrete event manufacturing systems can be consistently realized by Petri nets. In this paper, the author presents a methodology for hierarchical and distributed control of large and

complex robotic manufacturing systems using extended Petri nets, to construct the control system where the cooperation of each controller is implemented so that the aggregated behavior of the distributed system is the same as that of the original system and the task specification is completely satisfied.

II. MODELING OF DISCRETE EVENT SYSTEMS USING EXTENDED PETRI NETS

Discrete event systems such as robotic manufacturing systems have the properties of asynchronism, ordering, concurrency and conflict, so that deadlock will be apt to occur in the systems. The Petri net is one of the effective means to represent such systems. For applying it to the design, analysis, and control of the systems the guarantee of safeness and the notation of input/output of signals from/to machines should be required. A kind of graph deduced from the Petri net was proposed so as to satisfy the above requirements [2].

The extended Petri net consists of the following six elements: place, transition, directed arc, token, gate arc, output signal arc. A place represents a condition of a system element or action. A transition represents an event of the system. A directed arc connects a place to a transition, and its direction shows the input and output relation between them. Places and transitions are alternately connected using directed arcs. The number of directed arcs connected with places or transitions is not restricted. A token is placed in a place to indicate that the condition corresponding to the place is holding.

A gate arc connects a transition with a signal source, and depending on the signal, it either permits or inhibits the occurrence of the event which corresponds to the connected transition. Gate arcs are classified as permissive or inhibitive, and internal or external. An output signal arc sends the signal from a place to an external machine.

Formally, the firability condition and external gate condition of a transition j are described as follows:

$$t_j(k) = \bigcap_{m=1}^{M} p_{j,m}^{I}(k) \wedge \bigcap_{n=1}^{N} \overline{p_{j,n}^{O}(k)} \wedge \bigcap_{q=1}^{Q} g_{j,q}^{IP}(k) \wedge \bigcap_{r=1}^{R} \overline{g_{j,r}^{II}(k)} \quad (1)$$

$$g_j^{E}(k) = \bigcap_{u=1}^{U} g_{j,u}^{EP}(k) \wedge \bigcap_{v=1}^{V} \overline{g_{j,v}^{EI}(k)} \quad (2)$$

where,

M : input place set of transition j
$p_{j,m}^{I}(k)$: state of input place m of transition j at time sequence k
N : output place set of transition j
$p_{j,n}^{O}(k)$: state of output place n of transition j at time sequence k
Q : internal permissive gate signal set of transition j
$g_{j,q}^{IP}(k)$: internal permissive gate signal variable q of transition j at time sequence k
R : internal inhibitive gate signal set of transition j
$g_{j,r}^{II}(k)$: internal inhibitive gate signal variable r of transition j at time sequence k
U : external permissive gate signal set of transition j
$g_{j,u}^{EP}(k)$: external permissive gate signal variable u of transition j at time sequence k
V : external inhibitive gate signal set of transition j
$g_{j,v}^{EI}(k)$: external inhibitive gate signal variable v of transition j at time sequence k

The state (marking) change, that is, the addition or removal of a token of a place, is described as follows:

$$p_{j,m}^{I}(k+1) = p_{j,m}^{I}(k) \wedge \overline{(t_j(k) \wedge g_j^{E}(k))} \quad (3)$$

$$p_{j,n}^{O}(k+1) = p_{j,n}^{O}(k) \vee (t_j(k) \wedge g_j^{E}(k)) \quad (4)$$

An enabled transition may fire when it does not have any external permissive arc signaling 0 nor any external inhibitive arc signaling 1. The firing of a transition removes tokens from all its input places and put a token in each output place connected to it. The assignment of tokens into the places of a Petri net is called marking and it represents the system state. In any initial marking, there must not exist more than one token in a place. According to these rules, the number of tokens in a place never exceeds one, thus the Petri net is essentially a safe graph; the system is free from the bumping phenomenon. If a place has two or more input transitions or output transitions, these transitions may be in conflict for firing. When two or more transitions are firable only one transition should fire using some arbitration rule.

In the proposed procedure of modeling of robotic manufacturing systems, a global, conceptual Petri net model is first chosen which describes the aggregate manufacturing process. The places which represent the subtasks indicated as the task specification are connected by arcs via transitions in the specified order corresponding to the flow of subtasks and a workpiece.

For the example manufacturing cell with two robots,

one machining center, and two conveyors, where one is for carrying in and the other is for carrying out, the main execution of the system is indicated as the following task specification:

(1) A workpiece is carried in by the conveyor CV1.
(2) The workpiece is loaded to the machining center MC by the Robot R1.
(3) The workpiece is processed by the machining center MC.
(4) The workpiece is unloaded to the conveyor CV2 by the Robot R2.
(5) The workpiece is carried out by the conveyor CV2.

The places representing the existence of machines are also added to connect transitions which correspond to the beginning and ending of their subtasks. Thus at the conceptual level the manufacturing process is represented as shown in Fig. 1 for the cell with two robots.

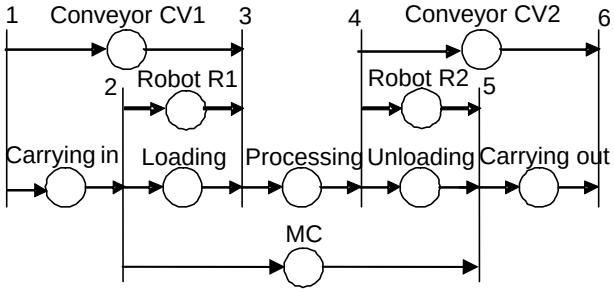

Fig.1 Petri net representation of example system with two robots at the conceptual level

Next, based on the hierarchical approach, the Petri net is translated into detailed subnets by stepwise refinements from the highest system control level to the lowest machine control level. At each step of detailed specification, a place of the Petri net is substituted by a subnet in a manner which maintains the structural properties. The detailed Petri net representation of the loading operation in the example system, which is a typical task at the subsystem coordination level in the factory automation system, is shown in Fig.2. Loading a workpiece to the machining center necessitates the cooperative or synchronized activities among the input conveyor, the machining center, and the robot. Similarly, unloading a workpiece from the machining center, necessitates the cooperative or synchronized activities among the output conveyor, the machining center, and the robot.

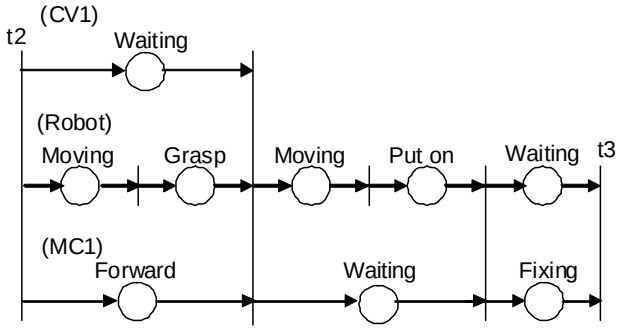

Fig.2 Detailed Petri net representation of loading operation

III. DECOMPOSITION AND COORDINATION ALGORITHM

For the manufacturing system, an example structure of hierarchical and distributed control is composed of one station controller and three machine controllers. The detailed Petri net is decomposed into subnets, which are executed by each machine controller.

In the decomposition procedure, a transition may be divided and distributed into different machine controllers as shown in Fig.3. The machine controllers should be coordinated so that these transitions fire simultaneously, that is, the aggregate behavior of decomposed subnets should be the same as that of the original Petri net. Decomposed transitions are called global transitions, and other transitions are called local transitions.

By the Petri net model, the state of the discrete event system is represented as the marking of tokens, and firing of any transition brings about change to the next state. So the firing condition and state (marking) change before decomposition should be the same as those after decomposition. If transition j is divided into s transitions $j1$, $j2$, , , js, the firability condition of a transition after decomposition is described as follows:

$$t_{jsub}(k) = \bigcap_{m=1}^{Msub} p^{I}_{jsub,m}(k) \wedge \overline{\bigcap_{n=1}^{Nsub} p^{O}_{jsub,n}(k)} \wedge \bigcap_{q=1}^{Qsub} g^{IP}_{jsub,q}(k) \wedge \overline{\bigcap_{r=1}^{Rsub} g^{II}_{jsub,r}(k)} \quad (5)$$

$$g^{E}_{jsub}(k) = \bigcap_{u=1}^{Usub} g^{EP}_{jsub,u}(k) \wedge \overline{\bigcap_{v=1}^{Vsub} g^{EI}_{jsub,v}(k)} \quad (6)$$

From eq.(1) and eq.(5),

$$t_j(k) = \bigcap_{sub=1}^{S} t_{jsub}(k) \quad (7)$$

From eq.(2) and eq.(6),

$$g_j^E(k) = \bigcap_{sub=1}^{S} g_{jsub}^E(k) \quad (8)$$

where,

S : total number of subnets

$Msub$: input place set of transition $jsub$ of subnet sub

$p_{jsub,m}^I(k)$: state of input place m of transition $jsub$ of subnet sub at time sequence k

$Nsub$: output place set of transition $jsub$ of subnet sub

$p_{jsub,n}^O(k)$: state of output place n of transition $jsub$ of subnet sub at time sequence k

$Qsub$: internal permissive gate signal set of transition $jsub$ of subnet sub

$Rsub$: internal inhibitive gate signal set of transition $jsub$ of subnet sub

$Usub$: external permissive gate signal set of transition $jsub$ of subnet sub

$Vsub$: external permissive gate signal set of transition $jsub$ of subnet sub

The addition or removal of a token of a place connected to a decomposed transition is described as follows:

$$p_{jsub,m}^I(k+1) = p_{jsub,m}^I(k) \wedge \overline{(t_j(k) \wedge g_j^E(k))} \quad (9)$$

$$p_{jsub,n}^O(k+1) = p_{jsub,n}^O(k) \vee (t_j(k) \wedge g_j^E(k)) \quad (10)$$

From the logical formulation of firability condition and marking before and after decomposition, it is proved that the firability condition of the original transition is equal to AND operation of firability conditions of decomposed transitions. In case that a transition in conflict with other transitions is decomposed, these transitions should be coordinated by the station controller. The coordinator was introduced to execute the coordination algorithm [2].

IV. PETRI NET MODEL BASED CONTROL EXPERIMENTS

The control software is distributed into the station controller and machine controllers. The station controller is composed of the Petri net based controller and the coordinator. The conceptual Petri net model is allocated to the Petri net based controller for management of the overall system. The detailed Petri net models are allocated to the Petri net based controllers in the machine controllers. The control of the overall system is achieved by coordinating these Petri net based controllers such that decomposed transitions fire at the same time and the task specification is completely satisfied. The overall control structure of the example manufacturing system was implemented on a local area network of computers as shown in Fig.3.

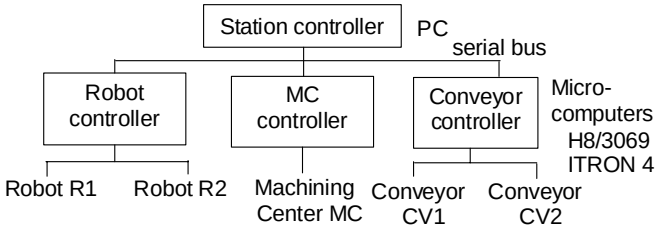

Fig.3 Implementation of real-time control system on microcomputer network

Each Petri net based machine controller on a dedicated microcomputer (Renesas H8/3069, real-time OS ITRON 4.0) executes real-time machine control, by initiating the execution of the unit actions attached to the fired transitions. The machine controller in charge of robot control executes robot motion control through the transmission of command and the reception of status report with serial interface to the real robot controller. The station controller implemented on another computer is also used to support modeling, simulation, and debugging of Petri nets and to load the Petri net based control program into the machine controllers. Communications among the controllers are performed using TCP/IP protocol. The coordinator sends the commands based on the conceptual, global Petri net model and coordinates the global transitions, which are accessed by the controllers as a shared file, such that decomposed transitions fire simultaneously.

V. CONCLUSIONS

An implementation of real-time distributed control for discrete event robotic manufacturing systems was described to realize consistent modeling, simulation and real-time control of large and complex robotic systems.

REFERENCES

[1] Hasegawa, K., Takahashi, K., and Miyagi, P. E., (1988) Application of the Mark Flow Graph to represent discrete event production systems and system control, Trans. of SICE, 24, 69-75

[2] Yasuda, G., (2009) Implementation of distributed cooperative control for industrial robot systems using Petri nets, Preprints of the 9th IFAC Symposium on Robot Control (SYROCO '09), 433-438