Petri Net Model Based Implementation of Hierarchical and Distributed Control for Discrete Event Robotic Manufacturing Cells

Gen'ichi Yasuda
Nagasaki Institute of Applied Science
Japan

1. Introduction

Manufacturing systems, where the materials which are handled are mainly composed of discrete entities, for example parts that are machined and/or assembled, are called discrete manufacturing systems. The rapid development of industrial techniques makes the manufacturing systems larger and more complex, in which system control is divided into a hierarchy of control levels: planning, scheduling, coordination and local control (Silva, 1990). At the upper level, the global system is considered and its representation is strongly aggregated. Each low level is a disaggregation of the upper one; the portion of the system considered is smaller, but more details are taken into account. The time horizon of the low level is shorter, and real-time constraints are progressively hard. At the coordination level (shop, cell), the function is to update the state representation of the workshop in real-time, to supervise it, and make real-time decisions. The decision making system has to schedule and synchronize the machine utilizations to decide at each instant what has to be done and on which machine.
Frequently, a flexible manufacturing system is structured into manufacturing cells. A cell is an elementary manufacturing system consisting of flexible machine tools (or assembly devices), some local storage facilities for tools and parts and some handling devices such as robots in order to transfer parts and tools between the cell and the global transport system. Elementary manufacturing cells are called workstations. In a manufacturing cell, some devices operate concurrently and cooperatively or interfere to each other. Conventional representation methods such as time chart and state transition graph can not cope with the system like this. So a powerful method which can represent a discrete event system including nondeterministic parallel and concurrent action is desired.
As computer technology has been continuing to be upgraded to higher level year by year, the control system architecture has changed from centralized processing to distributed processing in order to reduce the development cost and to improve reliability. In developing distributed processing systems, major difficulties are that adequate expression methods and analyzing techniques for control mechanisms have not sufficiently been established. In the field of manufacturing systems that are typical examples of the event driven system, demands for the automatic control has diversified and the control logic has become extremely complicated. To deal with the complexity, a new methodology on control system design based on the discrete event driven system is necessary.

For the flexibility and expandability of the event driven system software, a programming paradigm based on a network model is considered to be useful, because the network model can describe the execution order of sequential/parallel processes or works directly without ambiguity. For this class of problems, Petri nets have intrinsic favorable qualities and it is very easy to model sequences, choices between alternatives, rendezvous and concurrent activities by means of Petri nets (Thuriot et al., 1983). Moreover, the formalism allowing a validation of the main properties of the Petri net control structure (liveness, boundedness, etc.) guarantees that the control system will not fall in a deadlocked situation. Petri net based programming technique makes it possible to realize systematic and high-level description of system specification (Gentina & Corbeel, 1987), (Jockovic et al., 1990), (Wang & Saridis, 1990). Furthermore, a real-time implementation of the Petri net specification by a software called the token game player can avoid implementation errors, because the specification is directly executed by the token game player and the implementation of these control sequences preserves the properties of the model.
The Petri net has been applied to a variety of system developments such as real-time systems, manufacturing systems, communication systems, and so on, as an effective tool for describing control specifications and realizing the control system in a uniform manner. However, there are some problems to be solved as follows.

1. Although the description capability of the Petri net is very high, in case of flexible manufacturing systems the network model becomes very complicated and it lacks for the readability and comprehensibility.
2. Although the specification analysis stage has been supported, the support for the control software coding stage is insufficient. The flexibility and expandability are not satisfactory in order to deal with the specification change of the control system.

This paper discusses problems of the system realization by using Petri nets and proposes a design method for hierarchical and distributed manufacturing control systems. The Petri net model can be used from the conceptual, the global design of the system to the detailed design of its partial systems so that the modeling, simulation and control of large and complex discrete event manufacturing systems can be consistently realized by Petri nets. The complex control system can be easily designed and realized by using the global Petri net as an upper controller which controls lower controllers designed by detailed Petri nets. 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. At this point of view, this paper shows the efficiency of the proposed hierarchical and distributed control method by the simulation experiments of cooperative transferring tasks with mechanical arm robots.

2. Manufacturing systems and Petri nets

At each level of discrete manufacturing system control, any modeling has to be based on the concepts of events and states (Fogel & Sebestyenova, 1992), (Holt & Rodd, 1994), (Kasturia & Dicesare, 1988), (Rogers & Williams, 1988). An event corresponds to a state change. When using Petri nets, events are associated with transitions. Activities are associated to the firing of transitions and to the marking of places which represents the state of the system. In addition to its graphic representation differentiating events and states, Petri nets allows the modeling of true parallelism and the possibility of progressive modeling by using stepwise refinements or modular composition. Libraries of well-tested subnets allow components reusability leading to significant reductions in the modeling effort. The possibility of progressive modeling is absolutely necessary for flexible manufacturing systems because they are usually large and complex systems. The refinement mechanism allows the building of hierarchically structured net models which leads to the implementation of hierarchical and distributed control.

Formally, a Petri net has two types of nodes, called places and transitions. A place is represented by a circle and a transition by a bar. The places and transitions are connected by arcs. The number of places and transitions are finite and not zero. An arc is connected directly from one place to a transition or a transition to a place. In other words a Petri net is a bipartite graph, i.e. places and transitions alternate on a path made up of consecutive arcs.

An ordinary Petri net (Murata, 1989) is represented by the 5-tuple $G = \{P, T, I, O, M_o\}$ such that:

$P = \{p_1, p_2, ..., p_n\}$ is a finite, not empty, set of places;

$T = \{t_1, t_2, ..., t_m\}$ is a finite, not empty, set of transitions;

$P \cap T = \phi$, i.e. the sets P and T are disjointed;

$I : P \times T \rightarrow \{1, 2, 3, \cdots\}$ is the input weight function;

$O : P \times T \rightarrow \{1, 2, 3, \cdots\}$ is the output weight function;

$M_o : P \rightarrow \{0, 1, 2, 3, \cdots\}$ is the initial marking;

The pre-incidence matrix of a Petri net is $C^- = [c_{ij}^-]$ where $c_{ij}^- = I(p_i, t_j)$; the post–incidence matrix is $C^+ = [c_{ij}^+]$ where $c_{ij}^+ = O(p_i, t_j)$, then the incidence matrix of the Petri net $C = C^+ - C^-$.

Each place contains some (positive or zero) marks or tokens. The number of tokens in each place is defined by the marked vector or marking $M = (m_1, m_2, ..., m_n)^T$. The number of tokens in one place p_i is called $M(p_i)$. The marking at a certain moment defines the state of the Petri net, or the state of the system described by the Petri net. The evolution of the state therefore corresponds to an evolution of the marking, caused by the firing of transitions.

In an ordinary Petri net, where the current marking is M_k, a transition t_j is enabled if $\forall p_i \in P, M_k(p_i) \geq I(p_i, t_j)$. An enabled transition can be fired reaching a new marking M_{k+1} which can be computed as $M_{k+1} = M_k + C \cdot V$; this equation is called state equation of the Petri net, where $V = (v_1, v_2, ..., v_m)^T$ is the transition vector such that v_j =1 if transition t_j fires and 0 if not. When a transition is enabled, this does not imply that it will be immediately fired; this only remains a possibility. The firing of a transition is indivisible; the firing of a transition has duration of zero.

The firing of an enabled transition will change the token distribution (marking) in a net according to the transition firability rule. A sequence of firings will result in a sequence of markings. A marking M_n is said to be reachable from a marking M_0 if there exists a sequence of firings that transforms M_0 to M_n. The set of all possible markings reachable from M_0 is denoted $R(M_0)$. A Petri net is said to be k-bounded or simply bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from M_0, i.e., $M(p_i) \leq k$ for every place p_i and every marking $M \in R(M_0)$. A Petri net is said to be safe if it is 1-bounded. Bumping occurs when despite the holding of a condition, the preceding event occurs. This can result in the multiple holding of that condition. From the viewpoint of discrete event process control, bumping phenomena should be excluded.

The Petri net was modified so that the system is safe. The weight of the input and output weight functions I,O is 1 if the arc exists and 0 if not. Because the modified Petri net is safe, for each place p_i , $m_i = 0$ or 1, and a transition t_j is enabled if the two following conditions are met:

1. For each place p_k , such that $I(p_k,t_j)=1$, $m_k=1$
2. For each place p_l such that $O(p_l,t_j)=1$, $m_l=0$

Besides the guarantee of safeness, considering not only the modeling of the systems but also the actual manufacturing system control, the additional capability of input/output signals from/to the machines are required (Hasegawa et al., 1984). So gate arcs are classified as permissive or inhibitive, and internal or external. An external gate 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. An internal signal arc sends the signal from a place to a machine. Thus a transition is enabled if and only if it satisfies all the following conditions:

1. It does not have any output place filled with a token.
2. It does not have any empty input place.
3. It does not have any internal permissive arc signaling 0.
4. It does not have any internal inhibitive arc signaling 1.

An enabled transition may fire when it does not have any external permissive arc signaling 0 nor any external inhibitive arc signaling 1.

When an enabled transition t_j fires, the marking M is changed to M' , where

1. For each place p_k , such that $I(p_k,t_j)=1$, $m'_k=0$
2. For each place p_l such that $O(p_l,t_j)=1$, $m'_l=1$

To summarize it, the firing of a transition removes a token from each input place and put a token in each output place connected to it. As a natural requirement, 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. The assignment of token into the places of a Petri net is called marking and it represents the system state.

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. By the representation of the activity contents and control strategies in detail, features of discrete event manufacturing systems such as ordering, parallelism, asynchronism, concurrency and conflict can be concretely described through the extended Petri net.

The extended Petri net is a tool for the study of condition-event systems and used to model discrete event manufacturing systems through its graphical representation. Analysis of the net reveals important information about the structure and the dynamic behavior of a modeled manufacturing system. This information can then be used to evaluate the manufacturing system and suggest improvements or changes. The flow chart of simulation to check the dynamic behavior of the system quantitatively is shown in Fig. 1.

Implementation methods to do the real-time control are classified into two kinds (hardware and software). Hardware implementation realizes the places and transitions by hardware in a modular form, which are mutually connected in the same way as the Petri net model.

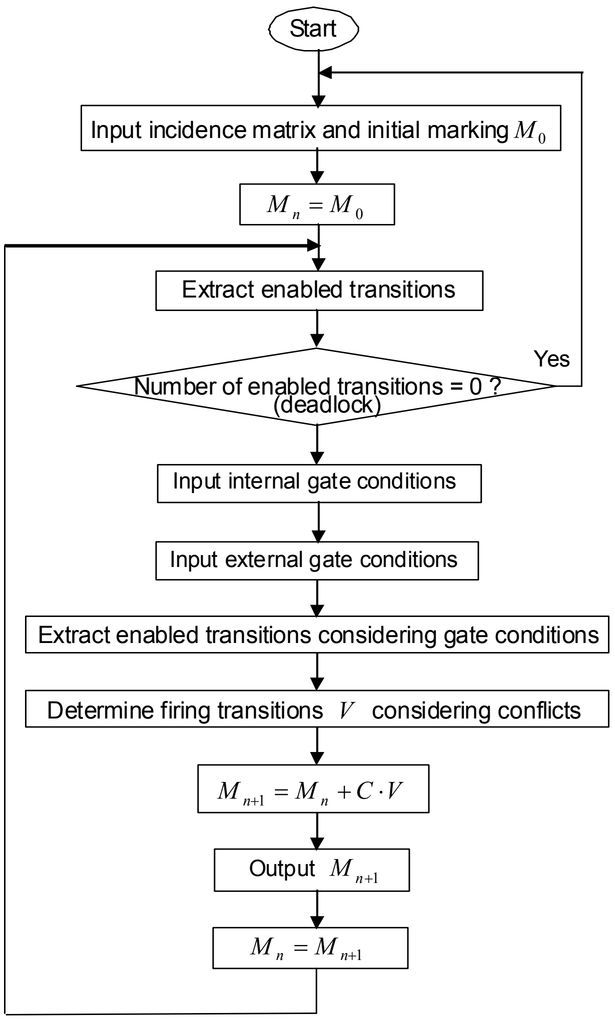

Start
Input incidence matrix and initial marking M_0
$M_n = M_0$
Extract enabled transitions
Number of enabled transitions = 0 ? (deadlock)
Yes
Input internal gate conditions
Input external gate conditions
Extract enabled transitions considering gate conditions
Determine firing transitions V considering conflicts
$M_{n+1} = M_n + C \cdot V$
Output M_{n+1}
$M_n = M_{n+1}$

Fig. 1. Flow chart of extended Petri net simulator

Signals from controlled objects (machines and external sensors) are connected to the transitions as external gate conditions. When a token exists in a place, the control information assigned to the place is transmitted to the machine. The simulation is performed by monitoring the marking. Hardware implementation by decomposing the net model into several state transition diagrams, which are easily realized using conventional sequential controllers, and combining them with interlock signals, can be effective in many industrial applications.
On the other hand software implementation is performed by developing control software based on the logical structure of the Petri net model (Taubner, 1988). Simulation is performed by program called a token game player with graphic display, and real-time control is performed through the computer interface. Centralized schemes can be

sequentially or concurrently implemented (Silva, 1990). Practically centralized sequential schemes are employed at the local control level frequently in special purpose real-time computers named Programmable Logic Controller (PLC) (Atabakhche et al., 1986) by means of converting the Petri net directly into a Boolean equation. Centralized concurrent implementations are basically composed of many specific tasks (possibly one per transition) and a coordinator. The coordinator plays the token game on the net model, initiating the execution of the tasks attached to the fired transitions. When a task ends its activity, it informs the coordinator to proceed with the next activations. The coordinator is an active task with a high priority that acts as the kernel of the application multitasking level. It provides indirect synchronization between the activities of specific tasks associated with the firing of transitions. The simplicity and easy modification of the synchronization/communication scheme are among the advantages of centralized implementation. The basic problems with this kind of solution are relatively inefficient execution (memory occupation and execution time) and weakness for safety. Decentralized implementation schemes are usually built by decomposing the net into a set of sequential processes and the required primitives concerning synchronization/communication between these processes are directly inserted, where required, in the body of the sequential processes (Georgeff, 1983), (Yasuda, 1999). The classical synchronization/communication mechanisms are based on synchronous rendezvous between tasks and asynchronous message passing implemented by means of buffers or mailboxes. In decentralized implementations, synchronization and communication between sequential processes are direct; there is no intermediate active element. In this paper, a hierarchical and distributed implementation is proposed, where a system controller with the global Petri net model coordinates several local controllers with detailed Petri net models.

3. Specification of manufacturing tasks using Petri nets

A Petri net model based consistent method for specification and implementation of hierarchical and distributed control of robotic manufacturing systems is proposed. A specification procedure for discrete event manufacturing systems based on Petri nets is as follows. First, the conceptual level activities of the manufacturing system are defined through a Petri net model considering the task specification corresponding to the aggregate manufacturing process. To deal with the problems related to the description of discrete event manufacturing systems using Petri nets, the system specification can be decomposed into two aspects: manufacturing sequence and resource allocation. A Petri net is associated to each aspect, and these two nets are strongly synchronized with the mechanism similar to transitions merging between the nets.

First, activities of manufacturing sequence must be executed to reach a given goal. For example a part is carried from a machine to another machine because the next operation has been scheduled on the machine. This activity modifies the physical state of the part according to the scheduled fabrication. In the Petri net specification, the location of the token shows the fabrication step of the part. An evolution of the process can be represented by distinct activities (starting event, effects and finishing event). Second, with respect to resource allocation, to execute the required activity, the process must be in a special state. For example, before executing a machining activity, on a given part, a machine must be free, and the required tool must be available, etc. These constraints depend on the layout and rules of operation of the process. They must be respected whatever the manufacturing sequence. In the Petri net specification, the location of the tokens models the state of the resources. The general representation of conceptual Petri net model by the above procedure is shown in Fig. 2, where the activity of each equipment as well as each subtask composing the task specification is represented as a place of the Petri net.

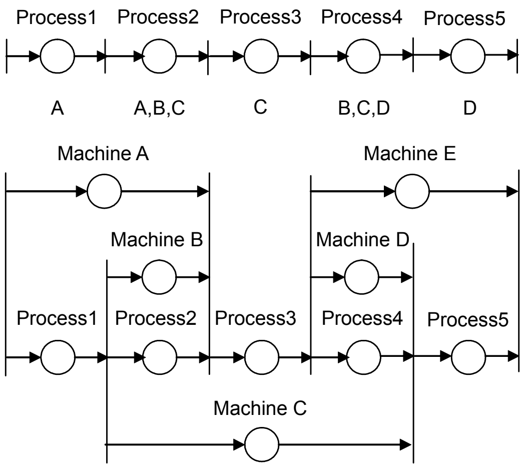


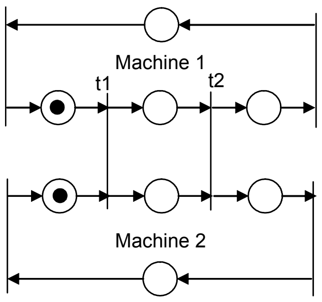
Fig. 2. General representation of conceptual Petri net model

4. Hierarchical representation of Petri nets

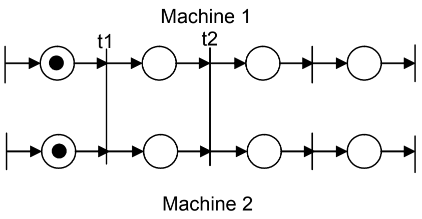
The detailed Petri nets are deduced to represent the control activities of associated machines. The macro representation of the manufacturing process is effectively used for achieving a top-down interpretation down to the concrete lower level activities using Petri nets. This procedure is repeated up to an appropriate level corresponding to the control level of the equipment responsible for the activity execution. At each step of detailed representation, a place of the Petri net are substituted by a subnet in a manner which maintains the structural properties such as liveness.

Petri nets for machine activity control are the loop structures because they eventually return to each home position. As required control strategies between equipment, the following two are frequently used: cooperative control and selective control. In cooperative control as shown in Fig. 3, transitions t1 and t2 indicate that two machines 1 and 2 should be controlled to start their actions synchronously. In selective control as shown in Fig. 4, where transitions t1and t2 are in conflict, one of two manufacturing processes is selected by an arbiter program which is substantially executed in the “Test” place.

For an example system of transferring task (Perez & Koutsourelis, 1987) by two robots (work transfer from the robot R1 to the robot R2), the conceptual Petri net model is shown in Fig. 5. Then Fig. 6 shows the detailed Petri net model, where places of work reception and work placement in the conceptual model are made detailed. The “Hand over” operation is accomplished through cooperative synchronized actions by two robots. The procedure of machine activity control is as follows. When a workpiece comes up to the initial position, which is represented as an external gate signal, the robot R1 takes the object, transfers it to the working space of the robot R2, and there waits for the robot R2 to taking the object. When the robot R1 stops, the robot R2 grasps it, then the robot R1 opens the hand. Then the robot R2 transfers it to the buffer and there releases it, and finally returns to the initial position.



(a) Loop structure



(b) Parallel structure

Fig. 3. Example representation of cooperative control between two machines

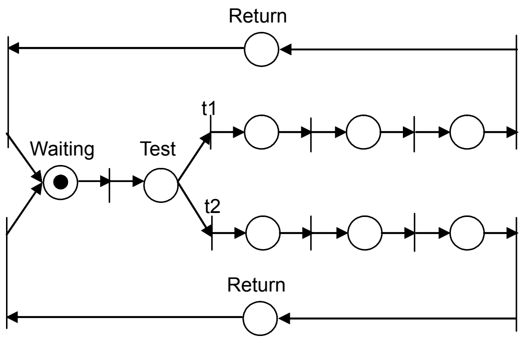


Fig. 4. Example representation of selective control between two manufacturing processes

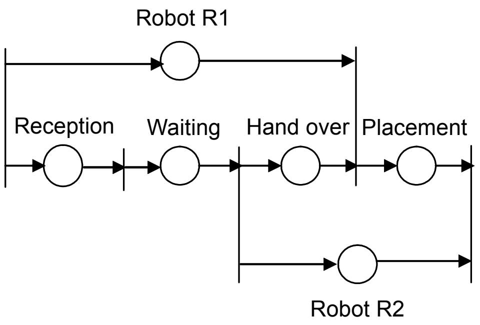


Fig. 5. Conceptual Petri net model of work transfer task by two robots

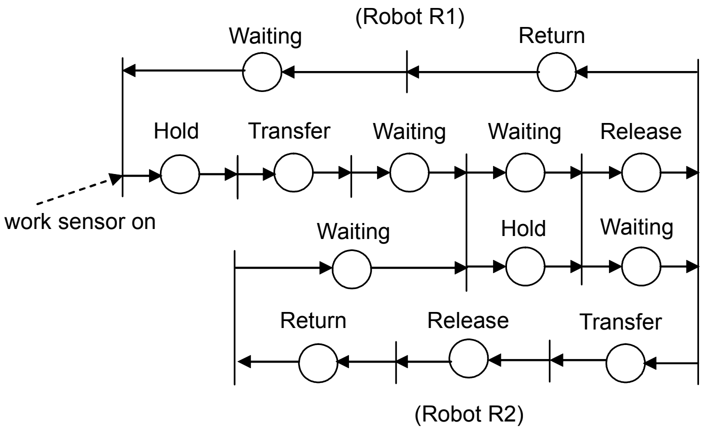


Fig. 6. Detailed Petri net model of work transfer task by two robots

The more detailed Petri net model at the actuator level of "Hold" and "Transfer" action, which is composed of opening, grasping and movement of each motor axis to the specified position, is shown in Fig. 7. In this figure, the "Dummy" place represents that its input and output transitions are start and end of unit action, respectively. This means that macro representation of a Petri net model can be achieved by replacing a detailed net with a dummy place.

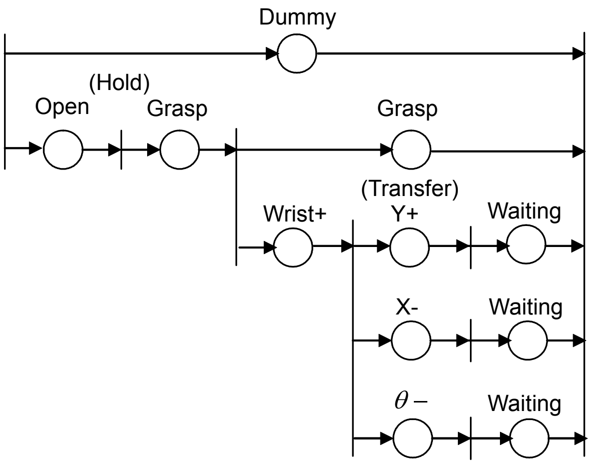


Fig. 7. Detailed Petri net model of "Hold" and "Transfer" action

The basic procedure of coordination by the communication between the coordinator and the local controllers is explained in detail as follows. As shown in Fig. 8, when a place in a Petri net model represents an "action", then the coordinator sends a "start" command with the parameter pointing to the detailed Petri net. The local controllers receive the command and the corresponding local controller executes the detailed subnet. When the execution is finished, the local controller returns the "end" signal to the coordinator. When a place represents the “waiting” state, then the coordinator sends a “ready” command with the parameter pointing to the next detailed Petri net. The local controller receives the command and executes the corresponding initialization. When the execution is finished, the local controller returns the “ack” signal to the coordinator.

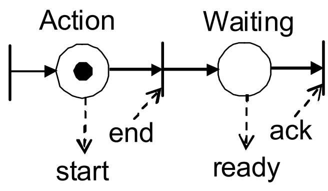


Fig. 8. Communication control between coordinator and local controllers

For the actual control, the operations of each machine are broken down into a series of unit motions, which is also represented by mutual connection between places and transitions. A place means a concrete unit motion of a machine. From these places, output signal arcs are connected to the machines, and external gate arcs from the machines are connected to the transitions of the Petri net when needed, for example, to synchronize and coordinate operations. When a token enters a place that represents a unit motion, the machine defined by the machine code is informed to execute a determined motion with a determined data; these are defined as the place parameters.

5. Implementation of hierarchical and distributed control

Transferring task by four robots (work transfer from the robot R1 to the robot R2, then transfer from the robot R2 to the robot R3 or R4 according to the type of workpiece) has been modeled and implemented as an experimental system. The configuration of the system contains four robots, one incoming and two outgoing conveyors, and a set of sensors. The sensor’s set consists of touch sensors on robots, which indicate whether a robot has or has not grasped a workpiece, and limit switch sensors for robot arm positioning. The layout of the experimental system is shown in Fig. 9.
The cell works in the following way: Workpieces come on the incoming conveyor CV1 up to the take up position 1-1. The robot R1 waits in front of the conveyor CV1 in position 1-2 which is defined by the limit switch for positioning the arm, and on conveyor stopping the robot R1 approaches in position 1-1, grips the workpiece with an energized magnetic hand and a touch sensor and returns to position 1-2. Then the robot R1 turns to position 1-3, goes into the working space of the robot R2 and there waits for the robot R2 to taking the workpiece. When the robot R2 comes to position 2-1, the robot R1 hands over the workpiece to the robot R2 by deenergizing the hand. The robot R1 detects it with the touch sensor. The robot R2 tests the workpiece with the touch sensor, transfers it to the robot R3 or R4 according to the result. Then, the robot R2 hands over the workpiece to the robot R3 or R4 in position 2-2 or 2-3, while the robot R3 or R4 takes it in position 3-1 or 4-1 by energizing the hand. Finally, the robot R3 or R4 carries the workpiece over to position 3-2 or 4-2 and there leaves it, while the robot R2 returns to position 2-1. The conceptual Petri net model and the detailed model are shown in Fig. 10 and Fig. 11, respectively.

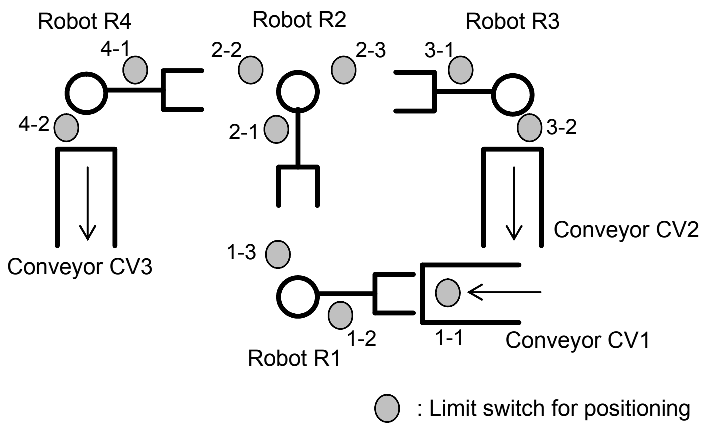


Fig. 9. Layout for work transfer task by four robots

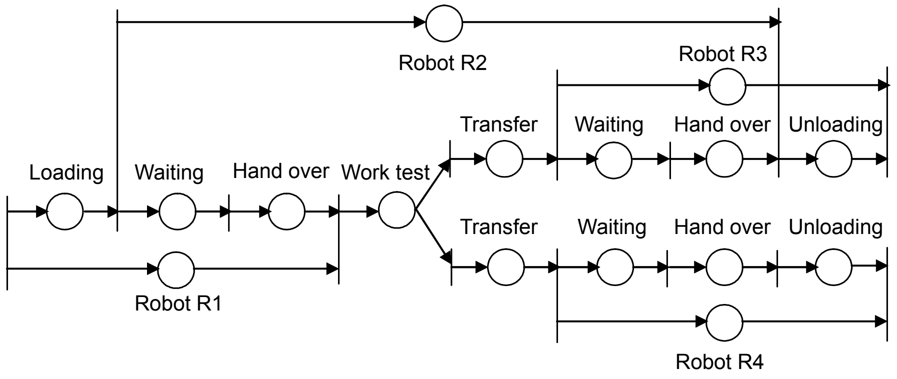


Fig. 10. Conceptual Petri net model of work transfer task by four robots

It is natural to implement a hierarchical and distributed control system, where one controller is allocated to each control layer or block. For the manufacturing system, an example structure of hierarchical and distributed control is composed of one system controller and several local controllers as shown in Fig. 12. The detailed Petri net is decomposed into subnets, which are executed by each local controller. The system 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 in the system controller for management of the overall system. The coordinator monitors the overall system using external sensors and coordinates the local controllers by sending the commands and receiving the status. The detailed Petri net models are allocated to the Petri net based controllers in the local controllers. Each local controller directly monitors and controls the sensors and actuators of its machine. In the example system, one local controller is assigned to each robot and conveyor.

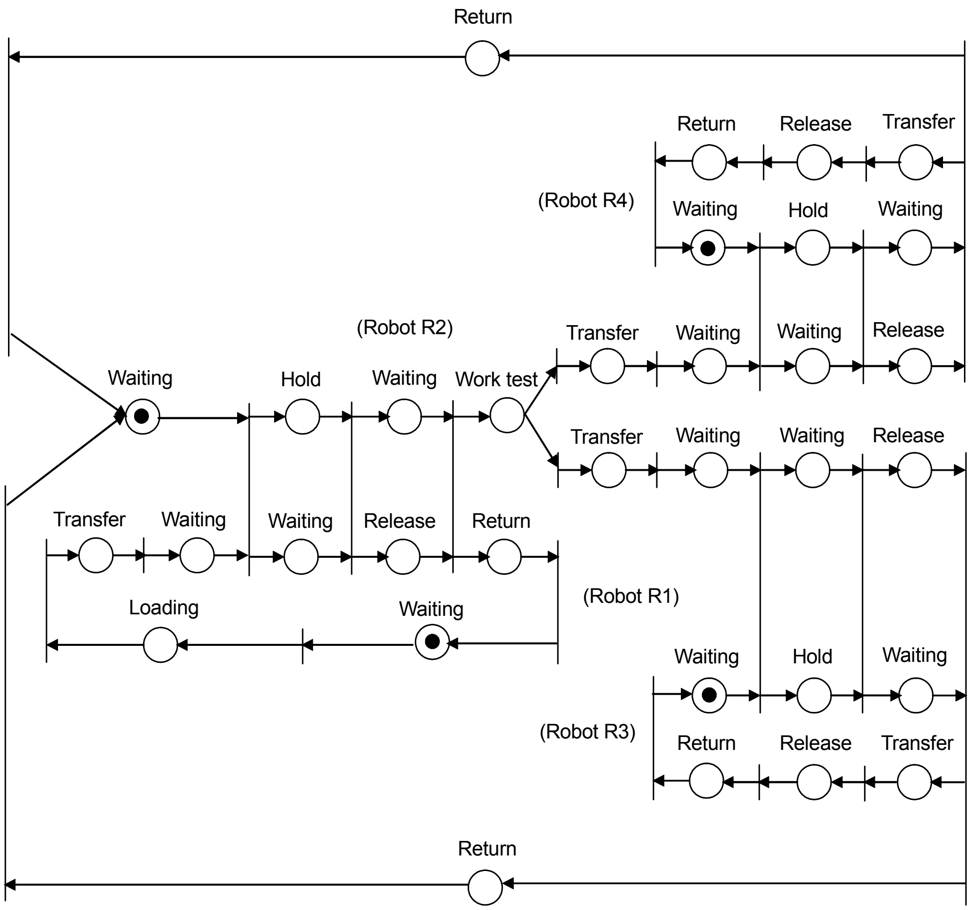


Fig. 11. Detailed Petri net model of work transfer task by four robots

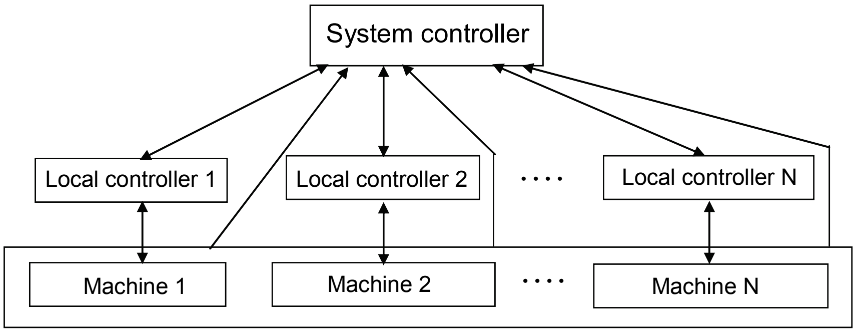


Fig. 12. Hierarchical and distributed control structure composed of system controller and local controllers

In the decomposition procedure, a transition may be divided and distributed into different local controllers as shown in Fig. 13. The local controllers should be coordinated so that these transitions fire simultaneously. Decomposed transitions are called global transitions, and other transitions are called local transitions.

As another example, loading a workpiece necessitates the cooperative or synchronized activities between the conveyor and the robot as shown in Fig. 14. First, "Carry in" operation to carry a workpiece is performed by the conveyor. At the end of the operation, when the robot is free, the "loading" operation is started. The conveyor starts waiting, the robot starts moving to grasp the workpiece. After holding the workpiece, the robot starts transferring it to another position and the conveyor is free. To exploit the above results, a coordinator program for simultaneous firing of decomposed transitions has been introduced so that the aggregate behavior of decomposed subnets is the same as that of the original Petri net.

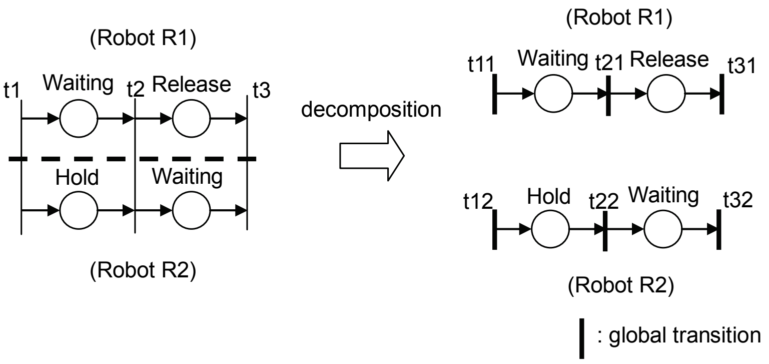


Fig. 13. Decomposition of transitions in "Hand over" operation

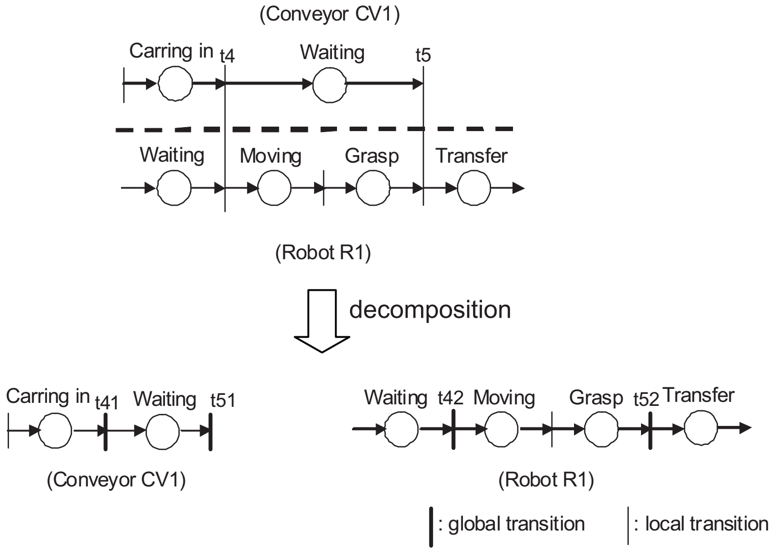


Fig. 14. Decomposition of transitions in the "Loading" operation

When some of the transitions in conflict are firable at the same time, only one of them must fire while the others become disabled. The choice for firing is done arbitrarily using an arbiter program. If arbitration of the transitions is performed independently in separate subnets, the results may be inconsistent with the original rule of arbitration. Therefore the transitions should be arbitrated together as a group by the system controller. On the other hand, arbitration of local transitions in conflict is performed by local controllers. When a robot can do several subtasks in cooperation with conveyors and machining tools in a flexible manufacturing system, transitions connected to the machine activities are found to be in conflict, in which the input place of the transitions represents the activity of the robot. In the decomposition of the detailed Petri net, these transitions also are decomposed and assigned to different machines. The global transitions should be arbitrated together as a group by the coordinator. Based on the decision of arbitration, the local controllers determine the firing transitions.

The overall control structure of an example robotic manufacturing system was implemented on a local area network of microcomputers. Each Petri net based local controller is implemented on a dedicated microcomputer (Renesas H8/3069) under the real-time OS ITRON 4.0. The local 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 upper level coordinator is implemented on another microcomputer. 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 system and local controllers as a shared file, such that decomposed transitions fire simultaneously.

In the simulation experiments, the names of global transitions and their conflict relations are loaded into the upper level controller. The connection structure of a decomposed Petri net model and conflict relations among local transitions are loaded into the Petri net based local controller. In the connection structure, a transition of a Petri net model is defined using the names of its input places and output places; for example, t1-1=b1-1, -b1-11, where the transition no.1 (t1-1) of controller no.1 is connected to the input place no.1 and the output place no.11. Using the names of transitions in the controllers, global transitions are defined; for example, in Fig. 13, global transitions are G1: t1-11, t2-12, G2: t1-21, t2-22, G3: t1-31, t2-32, and in Fig. 14, global transitions are G4: t5-41, t1-42, G5: t5-51, t1-52.

By executing the coordinator and Petri net based controllers algorithms based on loaded information, simulation experiments have been performed. The Petri net simulator initiates the execution of the unit actions attached to the fired transitions through the serial interface to the robots or other external machines. When the action ends its activity, it informs the simulator to proceed with the next activations by the external permissive gate arc. Experimental results show that the decomposed transitions fire at the same time as the original transition of the detailed Petri net of the whole system task. Firing transitions and marking of tokens can be directly observed on the display at each time sequence using the simulator (Yasuda, 2009).

6. Conclusions

A methodology to construct hierarchical and distributed control systems has been proposed. By introduction of the coordinator, the Petri net based controllers are arranged according to the hierarchical and distributed nature of the manufacturing system. The coordination mechanism between the upper level and the lower level controllers is based on firability test of global transitions, and its software can be easily implemented using the token game player.

An example robotic work cell containing four industrial robot arms was constructed and system control experiments including synchronization control and selective task control were successfully performed. The Petri net model in each Petri net based local controller is not so large and easily manageable. The local controller provides a common interface for programming robots and other intelligent machines made by different manufacturers, and can be implemented using conventional programmable logic controllers (PLC). The control method can be expanded to large and complex, discrete event manufacturing systems. Thus, modeling, simulation and control of large and complex manufacturing systems can be performed consistently using Petri nets. The proposed methodology is now being adapted to advanced multiple robot applications such as exploration robots, rescue robots and home assist robots, besides industrial applications.

7. References

Atabakhche, H.; Barbalho, D. S., Valette, R. & Courvoisier, M. (1986). From Petri net based PLCs to knowledge based control, *Proceedings of IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON'86)*, 817-822

Fogel, J. & Sebestyenova, J. (1992). An object oriented conception of a real-time control of FMS, *Proceedings of IFAC Workshop on Algorithms and Architectures for Real-Time Control*, 351-356

Gentina, J. C. & Corbeel, D. (1987). Coloured adaptive structured Petri net : A tool for the automatic synthesis of hierarchical control of flexible manufacturing systems (F.M.S.), *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, 1166-1173

Georgeff, M. (1983). Communication and interaction in multi-agent planning, *Proceedings of International Conference of AAAI-83*, 125-129

Hasegawa, K.; Takahashi, K., Masuda, R., & Ohno, H. (1984). Proposal of Mark Flow Graph for discrete system control. *Transactions of SICE*, Vol. 20, No. 2, 122-129

Holt, J. D. & Rodd, M. D. (1994). An architecture for real-time distributed AI-based control systems, *Proceedings of IFAC Distributed Computer Control Systems 1994*, 47-52

Jockovic, M.; Vukobratovic, M. & Ognjanovic, Z. (1990). An approach to the modeling of the highest control level of flexible manufacturing cell. *Robotica*, Vol. 8, 125-130

Kasturia, E. & Dicesare, F. (1988). Real time control of multilevel manufacturing systems using colored Petri nets, *Proceedings of 1988 IEEE International Conference on Robotics and Automation,* 1114-1119

Murata, T. (1989). Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, Vol.77, No.4, 541-580

Perez, R. A. & Koutsourelis, D. I. (1987). A command language for multiple robot arm coordination. *IEEE Transactions on Education*, Vol. E-30, No. 2, 109-112

Rogers, P. & Williams, D. (1988). A knowledge-based system linking to real-time control for manufacturing cells, *Proceedings of 1988 IEEE International Conference on Robotics and Automation*, 1291-1293

Silva, M. (1990). Petri nets and flexible manufacturing, In: *Advances in Petri Nets 1989*, Lecture Notes in Computer Science, Vol. 424, Rozenberg, G., (Ed.), 374-417, Springer-Verlag, Berlin

Taubner, D. (1988). On the implementation of Petri nets, In: *Advances in Petri Nets 1988*, Lecture Notes in Computer Science, Vol. 340, Rozenberg, G., (Ed.), 418-439, Springer-Verlag, Berlin

Thuriot, E.; Valette, R., & Courvoisier, M. (1983). Implementation of a centralized synchronization concept for production systems, *Proceedings of IEEE Real-time Systems Symposium*, 163-171

Yasuda, G. (1999). An object-oriented multitasking control environment for multirobot system programming and execution with 3D graphic simulation. *International Journal of Production Economics*, Vol. 60/61, 241-250

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

Wang, F. & Saridis, G. N. (1990). A coordination theory for intelligent machines, *Proceedings of 11th IFAC World Congress*, 235-240