Modeling and distributed implementation of synchronization and coordination in multi-robot systems

Gen’ichi Yasuda

Nagasaki Institute of Applied Science, 536 Aba-machi, Nagasaki 851-0193, Japan

Abstract

This paper deals with control system design and implementation problems encountered in multiple robot systems. A systematic method of constructing hierarchical net models is described for their direct implementation such that the net is translated into the detailed net by stepwise refinements from the highest task specification level to the lowest machine control level. Basic Petri nets are extended as a prototyping tool for expressing real-time control of robotic systems based on command response concept. The hierarchical and distributed coordinators are introduced to perform the synchronization and coordination of the robots and other machines in the system. The proposed method allows a direct coding of the inter-task cooperation from the net specification and can be implemented using off-the-shelf real-time executives.

© 2012 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of the Centre of Humanoid Robots and Bio-Sensor (HuRoBs), Faculty of Mechanical Engineering, Universiti Teknologi MARA.

Keywords: Control system design; multi-robot systems; discrete event systems; modeling; implementation; synchronization.

1. Introduction

Controlling complex robotic systems such as flexible manufacturing cells with multiple robots, requires sophisticated real-time distributed control systems. A major problem concerns the definition and control system design for the cooperation between the subsystems. Petri nets have been successfully introduced as an effective tool for describing process specifications and developing control algorithms of discrete event systems [1]. However, in the field of multiple robotic systems, the network model becomes complicated and it lacks the readability and comprehensibility [2], [3]. Therefore, the synthesis, correction, updating, etc., of the system model and programming of the controllers are not simple tasks. The author presents a Petri net based specification and real-time control method for large-scale and complex robotic systems. According to the hierarchical and distributed structure of the system, the conceptual net representation of a robotic task is broken down to generate the detailed nets at the local machine control level. An algorithm is proposed for coordination of machine controllers so that robots and other machines can synchronize activities and avoid harmful conflicts. By the proposed method, the whole cycle of control system design for multiple robotic systems from specification to automatic code generation, can be performed consistently using Petri nets.

2. Extended Petri Nets for Robotic Applications

A Petri net comprises two types of nodes, places representing conditions (or states) and transitions representing events, which are interconnected by directed arcs. Tokens, which reside at the places, are used to indicate the instantiation of a state. The current state of a net is represented by the distribution of tokens, or the marking, in the net. In a Petri net, the places, transitions and tokens are represented by the circles, bars and dots respectively. The extended Petri net adopts the following elements as input and output interfaces which connect the net to its environment: gate arcs and output arcs. A gate arc connects a transition with a status signal source, and depending on the signal, it either permits or inhibits the occurrence of
the event. External gate arcs from the machines are connected to the transitions of the net when needed, for example, to synchronize and coordinate operations. An output arc connects a place with an external machine and sends a command signal to the machine. When a token enters a place that represents a subtask, the machine defined by the machine code is informed to execute a subtask with specified parameters. For multiple sequence control of discrete event robotic systems, the axioms of nets are as follows:

1) A transition is enabled, if and only if, each of its input places has one token, each of its output places has no token and it does not have any internal permissive gate arc signalling 0 nor any internal inhibitive arc signalling 1;

2) An enabled transition is fired when it does not have any external permissive arc signalling 0 nor any external inhibitive arc signalling 1.

3) When an enabled transition fires, the marking is changed to the new one, where each of its input places has no token and each of output places has one token.

A transition without any input place is called a source transition, and one without any output place is called a sink transition. A source transition is unconditionally enabled, and the firing of a sink transition consumes a token in each input place but does not produce any. According to these axioms, the number of tokens in each place never exceeds one, thus, the net is essentially 1-bounded and said to be a safe graph. 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 enabled only one transition should be fired using some arbitration rule.


A specification procedure for discrete event robotic systems based on net models is as follows. The net based methodology addressed in the paper regards the tasks to be performed by the system for being primal. So, first, a robotic task is specified using the sequence of positional states representing the flow of workpieces or parts processed during the task. A positional state represents a workplace where workpieces can be temporally held for local handling or processing. The workplaces are associated to machines or resources in the system such as robotic hands, machining centers, conveyors, and buffer tables. Shared resources include protective information for mutual exclusion. By the occurrence of an event of global handling such as carrying, picking, placing, a positional state of a workpiece is changed to the next positional state. By the occurrence of an event of processing such as machining or assembling, a physical state of the workpiece is changed to the next state, while the positional state can be not changed. For robotic operation, a workplace is composed of a set positions closely located for local handling such as approaching, retreating, hand opening and closing which are defined with respect to the workplace.

Next, from the net model of task specification, the coordination level activities of the system are defined as a net model corresponding to the aggregate discrete event process considering machine operations. The places which represent the subtasks or operations indicated as the task specification are connected by arcs via transitions in the specified order corresponding to the flow of subtasks and a workpiece. The places representing the machines or other resources used for the subtasks are also added to connect transitions which correspond to the beginning and ending of their subtasks.

The example robotic cell has one robot, one machining center, and two conveyors, where one is for carrying in and the other one is for carrying out, as shown in Fig. 1. The system is structured into a set of physical workplaces where operations are performed on the workpieces. The discrete assignment of positions is also shown in Fig. 1.

Fig. 1. Configuration of example robotic cell.
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 R.
3) The workpiece is processed by the machining center MC.
4) The workpiece is unloaded by the robot R.
5) The workpiece is carried out by the conveyor CV2.

The net representation of the task specification is shown in Fig. 2.

Fig. 2. Net representation of task specification.

In the net, the arcs which connect the adjoining nodes represent the operations needed to transfer the workpieces from one place to another one. An operation corresponds to an event whose pre- and post-conditions are positional states, and the detailed representation is shown in Fig. 3(a), (b). First, the event indicating an operation is represented by a transition, then the transition is translated into a subnet with the beginning and ending events and the place “In progressive”. After the ending event, the completed state is indicated as place “Completed”. For actual net based control, the resource place is added in parallel with the place “In progressive” for resource assignment. Furthermore, a dummy place is added in parallel with the two places; “In progressive” and “Completed”, for safeness of the net, as shown in Fig. 3(b). Generally, the compact representation (Fig. 3(a)) is used for net representation of machine control, which means that if the action in progressive is completed, the ending transition is enabled. The resource place, which is essential for any place representing robotic action in progressive, can take the role of a dummy place for safeness.

The example robotic system works in the following way: Raw pieces of material come on the incoming conveyor CV1 up to the take up position “S12”. Robot R waits in front of the conveyor CV1, and on stopping approaches in the position, grips the object and returns to the position for waiting. Then it turns to position “S31” in front of the machining center MC, goes to position “S32” in the working space of the machining center and there it leaves the workpiece. After directed gripping of the workpiece is effected by a vice, the robot R draws back to the position “S31” and there it waits for the machining center to complete object processing. After object processing, the robot goes to the machining center in the position “S32”, takes the processed object from an opened vice, and carries it over to the position in front of the outgoing conveyor CV2, and then to position “S41” where the workpiece is put on. Finally, the workpiece is carried out by conveyor CV2.

The system controller level net model of the task specification is shown in Fig. 4. The net representation of the task specification is sequentially defined according to the flow of workpieces. In parallel with each place representing a subtask, resource places representing associated machines are connected. For loading, first, a workpiece is transferred from the conveyor to the robot, then from the robot to the machining center. So, these three machines are associated to the subtask to guarantee the efficient system coordination. The resource place associated to loading and unloading is the same place representing the state of the robot; the place has two input and output transitions. So, the system controller should arbitrate any conflicts with order, logical priority, or some other arbitration rule.
The net model executed for coordination by system controller is shown in Fig. 5. Initially, a token is placed in the resource places of the conveyor and the machining center, which indicate that these resources are not working, waiting for a command request. Because the robot is initially idle, the resource place has no token. A transition is enabled if its pre-condition action is completed, which is informed with a corresponding gate signal as shown in Fig. 3. The resource places are also used in order to guarantee the safeness of the operations.

The transition place connection, which defines the labels of input (negative) and output (positive) places for each transition, is indicated in Table 1. Net simulation based on transition enabling and firing test is executed using the transition place connection table, the transition gate connection table, and the place transition connection table for arbitration as well as the transition enabling table and the marking table.

The dynamic behavior of the system represented by a net model is simulated using the enabling and firing rules. For efficient simulation combined with real-time control of a robotic system, the following steps are executed only when some gate condition is changed.

1) Calculate the logical variable of the transition associated with the new gate condition.
2) If the transition is in conflict with other transitions the arbitration rule is executed.
3) If the transition is fired, calculate the logical variables of its input and output places.
4) Then the marking is changed and a new command is sent to the corresponding robot.

An example of transition firing and the resultant marking change is shown in Table 2. Some important net properties in robotic systems, such as boundedness or no capacity overflow, liveness or freedom from deadlock, conservativeness of resources and reversibility, are validated using reachability analysis, invariant analysis, and simulation. If capacity control in a subsystem is required, a direct path with a capacity control place is supplemented between the beginning and ending transitions of the subsystem.

![Fig. 5. Conceptual net model for coordination by system controller.](image)

Table 1. Transition connection table

<table>
<thead>
<tr>
<th>Transition No.</th>
<th>Place No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,6</td>
</tr>
<tr>
<td>2</td>
<td>2,1,-7,-8</td>
</tr>
<tr>
<td>3</td>
<td>2,-3,6,7</td>
</tr>
<tr>
<td>4</td>
<td>-3,4,7,9</td>
</tr>
<tr>
<td>5</td>
<td>-4,5,-7,8</td>
</tr>
<tr>
<td>6</td>
<td>-5,9</td>
</tr>
</tbody>
</table>

Table 2. Transition firing sequence

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fired transition</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Place with token after transition firing</td>
<td>1,6</td>
<td>2,6,7,8</td>
<td>3,8</td>
<td>1,3,6,8</td>
<td>1,4,6,7,8</td>
<td>1,5,6,9</td>
<td>2,5,6,7,8</td>
<td>2,6,7,8</td>
</tr>
</tbody>
</table>

![Fig. 4. Conceptual net representation of robotic task with subtask and resource places.](image)
Fig. 6 shows the detailed net representation of subtasks: loading and unloading in Fig. 4. For the example system, detailed net models can be automatically generated using the database of net models of general robotic subtasks and operations. In the net, the subtask by several machines is represented by a shared transition.

4. Design of Distributed Control Architecture

The detailed nets describing the subtasks are deduced for the required control command level of the hardware controller. At each step of detailed specification, places of the net are substituted by a subnet in a manner which maintains the structural properties such as liveness and safeness. The detailed subtask models are decomposed into local net models performed by subsystems by a partition of the detailed net. Several strategies for obtaining local net models may be adopted according to the number of processors and the geographical distribution of the machines in the system. In this work the strategy to create the maximal number of local net models is adopted; each local net model involves one active component such as robot, active device, or external sensor.

The overall control architecture, which is composed of one system controller and several machine controllers, is shown in Fig. 7. The system control receives the states of all the machines, executes the conceptual net model, decides the next subtask, and dispatches it to the machine controllers. The machine controllers perform the mutual cooperation with other machine controllers in order to carry out the subtask called for by the system controller. They provide a device dependent interface to the actual hardware controller by translating the commands and error messages of the corresponding machine.
The primary function of the system controller is to schedule the start of a subtask, and waits for its completion in order to command the next subtask. In the conceptual net model subtasks may involve several constituent machines, called composite subtasks. As an example, loading necessitates concerted operations involving the conveyor, the robot, and the machining center. Subtasks involving a single machine are called simple subtasks; carrying in and out by the conveyor are simple subtasks. The system state is given at any instant by the collection of states of its constituents. For initiation of a subtask, a composite state consisting of a subset of the system state needs to exist. These composite states are specified as the coordination information. A transition associated to two or more machines and corresponding transitions in the detailed nets for the machine controllers are called global transitions, and other transitions are called local transitions. The global transitions should be synchronized because they represent the same transition, so that the original transition is enabled if all transitions deriving for the distribution are enabled. The information of global transitions in the machine controllers is sufficient to coordinate the overall system. Using the names of transitions in the subsystems, global transitions for system coordination are defined; for example, G2: t0-2, t1-2, t2-2, t3-2 indicates that global transition G2 representing the end of carrying in and the start of loading from CV1 is composed of transition no.2 of System controller (subsystem no.0), transition no.2 of CV1 controller (no.1), transition no.2 of Robot controller (no.2), and transition no.2 of MC controller (no.3). Similarly, G5: t0-5, t2-5, t4-5 indicates that global transition G5 representing the end of unloading from MC and the start of carrying out is composed of transition no.5 of System controller, transition no.5 of Robot controller, transition no.5 of MC controller (no.3). Then, the coordinator information for the example control system is as follows.

G1: t0-1, t1-1 ; start of carrying in
G2: t0-2, t1-2, t2-2, t3-2 ; end of carrying in and start of loading from CV1
G3: t0-3, t2-3, t3-3 ; end of loading into MC and start of processing
G4: t0-4, t2-4, t3-4, t4-4 ; end of processing and start of unloading from MC
G5: t0-5, t2-5, t4-5 ; end of unloading from MC and start of carrying out
G6: t0-6, t4-6 ; end of carrying out

Based on the coordinator information, the system coordination program is defined, such that for a global transition if all the transitions are enabled, then these transitions are eventually fired simultaneously [4]. The coordination program can be written as production rules or IF-THEN expressions used in conventional procedural languages, such that rule conditions shown by a conjunctive form would, as a consequence, have a set of actions. In the hierarchical and distributed architecture, the enabling conditions of the transitions t0-2, t1-2, t2-2, t3-2 are sent from each machine controller to the system controller, and the resulting actions are sent from the system controller to the machine controllers. The coordinator information is memorized in a table of the system controller and used to perform net execution efficiently with the other tables representing the structure of the net specifying the robotic task. The global transition is implemented in shared memory which can be mutually accessed by the coordinator in the system controller and the control program in machine controllers through centralized communication.

The coordination information for mutual synchronization among machine controllers in the detailed net is generated autonomously using the library of subtask net models as follows.

G7: t1-7, t2-7, t3-7 ; end of picking and start of placing
G8: t2-8, t3-8 ; end of placing and start of clamping
G9: t2-9, t3-9 ; end of picking and start of placing

The synchronization of placing operation onto the table in the machining center by the robot is directly performed through asynchronous communication using a net model without the coordinator as shown in Fig. 8. In asynchronous communication a controller sends a message for the destination controller then it waits for a reply. This mechanism can be easily replaced by interlock signals using gate arcs for mutual communication between controllers.

Fig. 8. Mutual communication structure for synchronization between the conveyor and the robot.
Generally, complex multiple robot systems involve tasks that are performed on single robots independent of others, and tasks that require the coordination of two or more robots. In cases where a subtask calls for the cooperation of two or more robots, the local controllers have to be involved to ensure proper execution of that subtask. A possible communication and control net structure for selection of synchronous parallel operations by two robots is shown in Fig. 9

![Fig. 9. Mutual communication and control net structure for selection of synchronous parallel operation by two robots.](image)

5. Implementation of Machine Controller

The local machine controllers execute actions or operations to carry out the subtask command called for by the system controller. The basic net model of the local machine controller is shown in Fig. 10. The completion of an action is reported using an external gate arc from the machine. After the completion, the controller sends the status report to the net model and the command to go to its home position to the machine. Then it waits for the next request command from the net model through conflicting transitions. They are designed to include a virtual device connected to the actual robot and device controller. Each virtual device monitors the state of the physical machine it represents. Each machine state is analysed and reported to the corresponding control program in the local controller and the system controller as required. They also act as conduits for commands from the local controllers to the physical machines.

![Fig. 10. Basic net model of action execution by local controller.](image)

6. Conclusions

A systematic procedure of representing the interaction between robots and machines and the interlock signals required in multiple robot systems has been developed. The system controller oversees the operation of a set of logically decentralized controllers and initiates the cooperative execution of subtasks based on the conceptual net model. Sensory signals indicating the change of robots and machines state are used to trigger or initiate subtasks. Each subtask is performed through the cooperative operations of local controllers. The system control strategies are specified by a set of net models and their accompanying execution rules. The net model allows the user to visualize graphically the conditions required so that the subtasks programmed can be initiated. By introduction of the coordinator based on hierarchical Petri nets, the coordination mechanism is implemented in each layer repeatedly and the overall control architecture can be configured employing the holonic concept, so that the distributed autonomous control architecture can be realized based on Petri nets.

References