



ST*

1384 OF 1

BEBR

FACULTY WORKING
PAPER NO. 1384

THE LIBRARY OF THE
JAN 13 1993
UNIVERSITY OF ILLINOIS
URBANA-CHAMPAIGN

A Distributed Knowledge-Based Approach to Flexible Automation: The Contract-Net Framework

Michael J. Shaw

Andrew B. Whinston

BEBR

FACULTY WORKING PAPER NO. 1384

College of Commerce and Business Administration

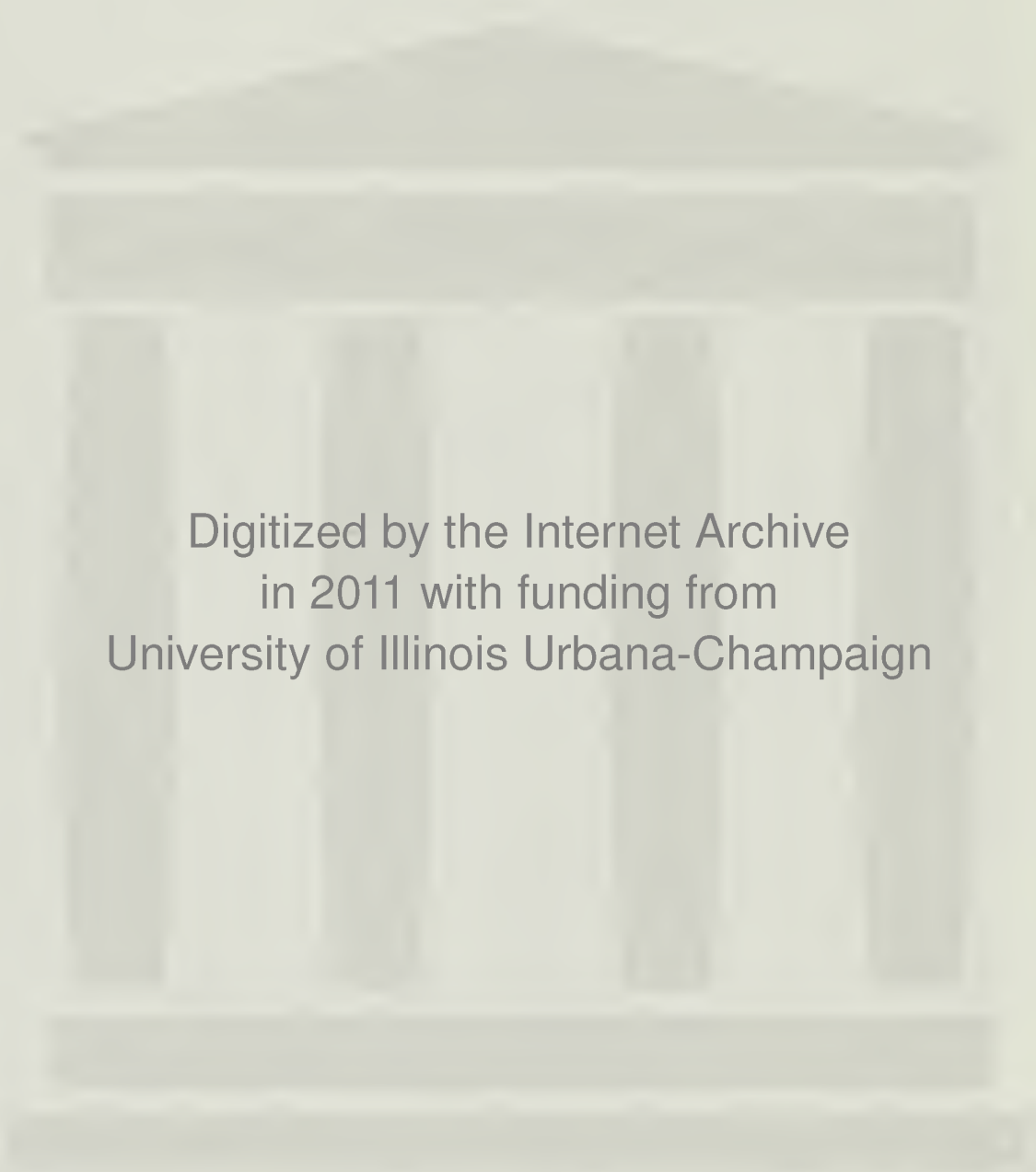
University of Illinois at Urbana-Champaign

August 1987

A Distributed Knowledge-Based Approach
to Flexible Automation:
The Contract-Net Framework

Michael J. Shaw, Assistant Professor
Department of Business Administration

Andrew B. Whinston
Purdue University



Digitized by the Internet Archive
in 2011 with funding from
University of Illinois Urbana-Champaign

ABSTRACT

This paper applies distributed artificial intelligence to the real-time planning and control of flexible manufacturing systems (FMS) consisting of asynchronous manufacturing cells. A knowledge-based approach is used to determine the course of action, resource sharing, and processor assignments. Within each cell there is an embedded automatic planning system that executes dynamic scheduling and supervises manufacturing operations. Because of the decentralized control, real-time task assignments are carried out by a negotiation process among cell hosts. The negotiation process is modeled by augmented Petri nets--the combination of production rules and Petri nets--and is executed by a distributed, rule-based algorithm.

Keywords: FMS Scheduling and Control; Distributed Artificial Intelligence; Networking FMS.

1. Introduction

Flexible automation--automation that can handle a large and constantly changing variety of produced items--has played an increasingly important role in the efforts to improve the productivity of the manufacturing industry (Hutchinson 1984, Merchant 1983). The recent progress in computer technologies has accelerated the realization of flexible automation. The use of computers in manufacturing, such as the computer numerical controlled (CNC) machines, adds programmability and thus versatility into manufacturing systems. More important, computers also provide on-line execution of manufacturing planning and decision making. These two capabilities, computerized control and on-line planning, and decision making are integrated into a well-orchestrated, automated manufacturing system--referred to as the Flexible Manufacturing System (FMS)--that can produce wide-ranging items efficiently.

In implementing FMS, the cellular architecture has emerged as the most effective and economical organization for the system (Cutkosky 1983, McLean et al. 1983, Ranky 1986, Shaw 1987). Such a system consists of a collection of "manufacturing cells." As shown in Figures 1 and 2, each cell is controlled by a host computer supervising the activities in the cell. Any cell can communicate with other cells through a local area network. The corresponding information system is highly distributed. For instance, the host computers used in the National Bureau of Standard's research facility range from a VAX 11/780 to 8086- or 68000-based single board computers and multiple

processor systems. It is thus important to take into account the networking features in the planning and control of FMS.

Insert Figures 1 and 2 Here

Intelligent manufacturing planning and control in an FMS requires knowledge bases that contain information about current tasks, manufacturing procedures, and the production environment (McLean et al. 1983). Furthermore, the evolution of the FMS into distributed architectures has complicated the information processing requirements. Issues that must be considered include the effective planning and problem solving in the distributed environment; the structuring of knowledge bases to facilitate knowledge sharing between system components; and the coordination and communication among manufacturing cells. To achieve these, this paper develops a distributed knowledge-based approach for manufacturing planning and control. In it, an embedded nonlinear planning system in each cell generates production procedures, supervises resource sharing, and guides manufacturing executions. This planning system also keeps track of the manufacturing environment, directing adjustments on the production procedures when necessary.

Moreover, since a job may consist of a set of tasks to be assigned to several manufacturing cells, the system needs to supply a coordinating mechanism that, through the communications network, permits the matching of tasks to cells. To achieve such coordination by decentralized control, three issues need to be addressed:

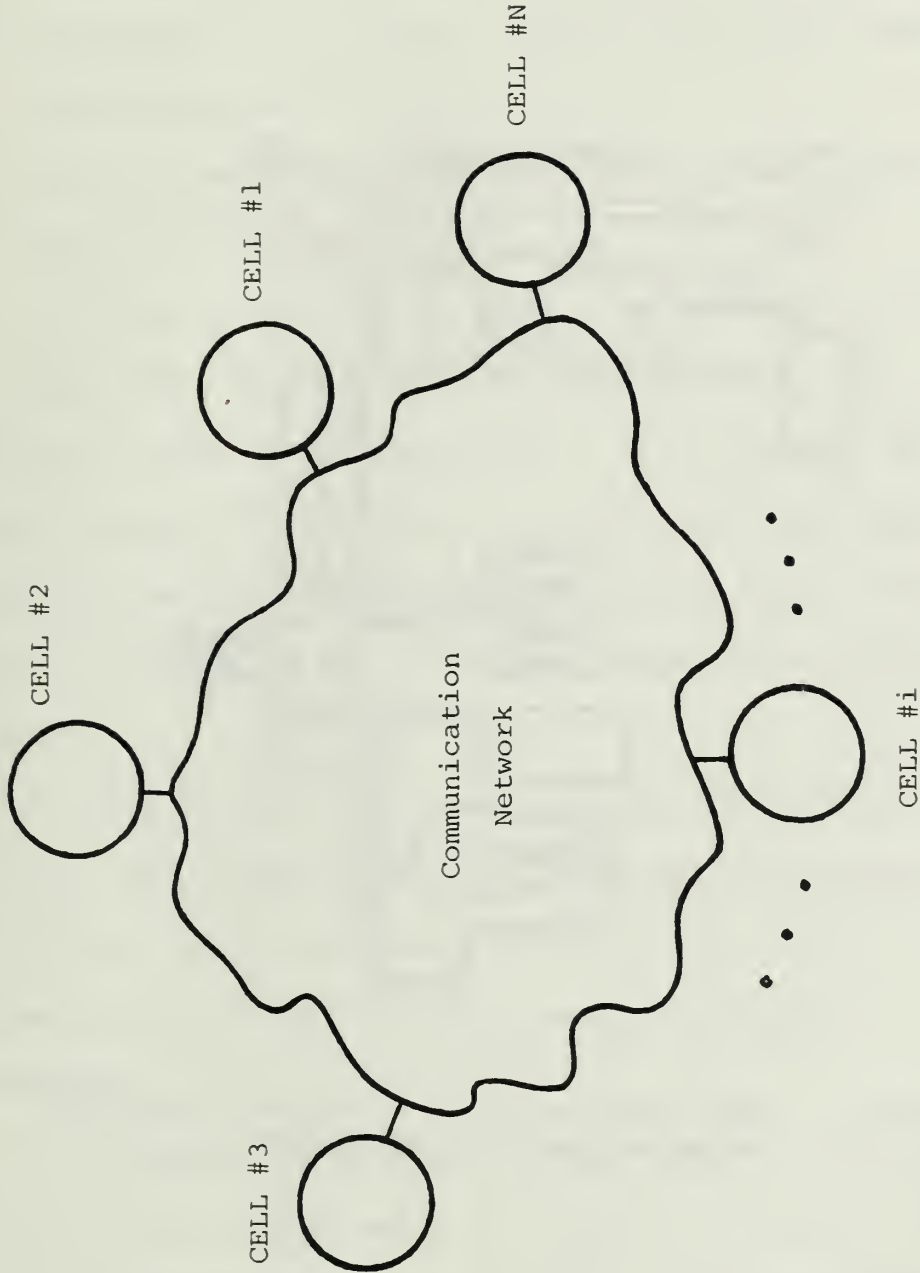


Figure 1. A Network of Manufacturing Cells

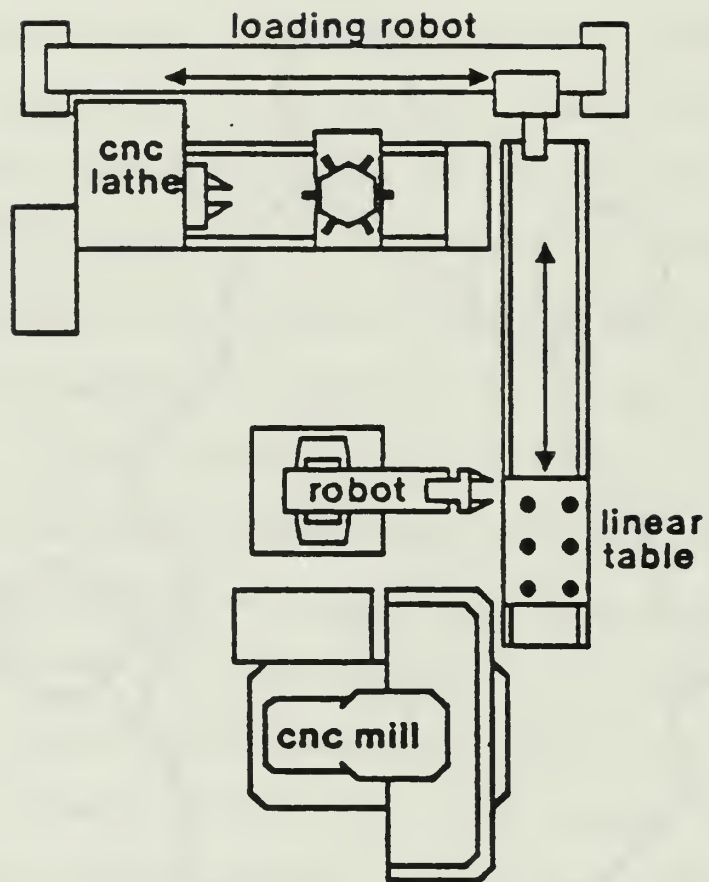


Figure 2 The Organization of a Manufacturing Cell
(Adapted from Cutkosky 1983)

- (1) the model of the problem-solving process which, through the communications network, dynamically distributes tasks among the cells;
- (2) the design of an interface language that enables effective communication among cell hosts; and
- (3) the programming and execution of this problem-solving process at each cell in a decentralized manner.

Following the distributed artificial intelligence framework (Davis and Smith 1983, Smith 1980), this paper uses the "contract net" approach to meet these requirements. A network-wide negotiation procedure is used to ensure orderly information transformation and events sequencing between asynchronous, cooperating cells. The underlying idea is to structure the interactions among cells by tasks negotiation.

In order to carry out the process systematically, it is important to have a good representation of the negotiation process and to capture the dynamic, concurrent nature of the process. Also, this representation should be integrated into executable programs to coordinate task execution.

The augmented Petri net, an integration of production rules and Petri nets, is used to model the negotiation process. The automation of this augmented Petri nets leads to a distributed algorithm for task allocation. Since the procedure is designed to be implemented by a variant of a rule-based system--i.e., the control production system, the knowledge-based system in a sense contains two kinds of program knowledge: the knowledge for task planning and the knowledge for intercell cooperation.

The remainder of this paper is organized as follows. Section II characterizes the distributed artificial intelligence system in the FMS environment and discusses the planning approach for flexible automation. Section III shows the use of negotiation process for coordinating the tasks; this process is then represented by the augmented Petri net model. Section IV illustrates the implementation of the negotiation process. The final section summarizes the paper.

II. Knowledge-Based Planning

In general, a knowledge-based planning system develops a course of action for the agents to reach to goals desired. In an FMS, the agents may be robots, computerized machine-centers, or the host computer of a cell; they usually can carry out a variety of operations, including various types of machining, workpiece routing, loading, and unloading operations (Costa and Garetti 1985). As shown in Figure 3, a knowledge-based planning system for the FMS has three basic components:

Insert Figure 3 Here

- (1) the database which serves as the working memory. It contains a symbolic description of the real world, referred to as the world model. This world model is represented by the collection of first-order predicates in the database.
- (2) The action model, which describes the transformational effects of actions that map states to other states. Such transformations are usually modeled by operators similar to the STRIPS operators defined in Fikes and Nilsson (1971).

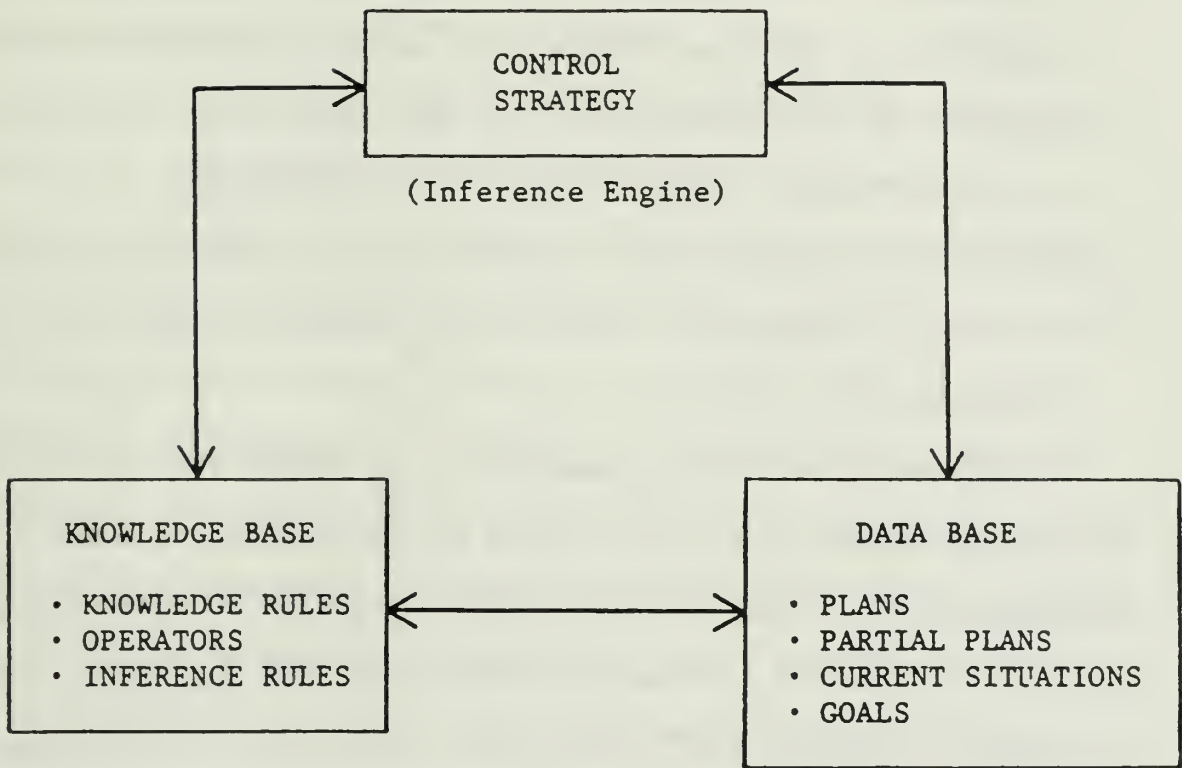


Figure 3. Basic Structure of a Planning System

In addition to the standard STRIPS formalism--which specifies an action by the add list, delete list, and preconditions--we have also included two more descriptions for each action--the "resource" used during the action, and the "duration" of the action.

(3) The inference engine, which directs the plan generation process.

It selects a sequence of operators to achieve the goal state from a given initial state.

The linear plans can be generated by any STRIPS-like plan generation system (Fikes et al. 1972, Fikes and Nilsson 1971). Such a planning system can use a backward-chaining method in searching for the best actions--i.e., it works backward from the goal state and find a sequence of actions that could produce this goal state from the initial state. The process of plan generation, then, can be viewed as finding the solution path in a search tree. The root of the tree is the goal state, and instances of operators define the branches. The solution path starts with the root (the goal state) and leads to the leaves (the initial state), thereby defining the plan.

Within this planning framework, the manufacturing process corresponding to each task is modeled by state-changing transformations, represented by operators. If the manufacturing processes for different tasks are independent, then, in principle, they can be executed in parallel. In reality, however, different tasks usually are competing for machines, tools, and other resources; therefore, the planning system must take into account the interactions among the processes. Because the plans generated by these methods are partially ordered, these methods are called nonlinear planning (Sacardoti 1977, Shaw and Whinston 1985, Vere 1983, Wilkins 1984).

This approach displays some desirable characteristics for real-time planning and control in the FMS environment. First, it is goal-directed, i.e., users only need to specify the goals and the planning system would, accordingly, derive the necessary steps on-line. Second, it is dynamically adjustable. New goals can be accommodated while the current production plan is still being executed; also, plans can be modified when unexpected events occur (e.g., tool or machine breakdowns). A "plan revision scheme" is initiated when bottlenecks are detected; the scheme, in turn, seeks to use alternative resources to improve the throughput. In addition, travel paths taken by guided carts or the arm movements of neighboring robots should be analyzed so that any potential conflicts or interferences are avoided (Bourne and Fussell 1982, Lozano-Perez 1982).

Formally, the problem to be solved in the planning system can be defined as a quadruple

$$PR = \langle S, O, IS, G \rangle$$

where S is the set of states in the database, O is the set of operators, defined as functions $S \rightarrow S$. Both IS and G belong to S ; IS denotes the initial state, and G denotes the goal state. The inference engine selects the sequence of operators in the search space based on predefined control strategies.

To perform planning by a distributed system with a group of intelligent agents, the problem PR is decomposed into subproblems. The final plan consists of a collection of plans for these subproblems, coordinated to be applicable to all initial states IS and to achieve

the goal state G. The coordination between the planning agents may be accomplished through messages passed between agents. The key issue, then, is how to automate this coordination to result in orderly interactions. The strategy we shall use throughout this paper for this type of multi-agent planning consists of four phases:

- (1) Goal decomposition. A job is decomposed into tasks to be executed by different cells.
- (2) Tasks distribution. Tasks are distributed among the cells so that each task is performed by the most appropriate cell available.
- (3) Tasks execution. Cells perform the assigned tasks as required.
- (4) Tasks synthesis. Finished tasks of the same job are assembled.

This strategy is referred to as the task-sharing strategy. Another widely used strategy is result-sharing--i.e., each agent will independently initiate its problem solving activities and subsequently aggregates its results. In the foregoing task-sharing scheme, the task distribution problem needs special attention because of the loosely coupled, decentralized control structure. The following sections discuss the use of a negotiation process to achieve it.

Task-sharing is carried out based on the job hierarchy where each job is decomposed into a set of tasks. Each task may correspond to the operations for a part family, thus can be performed in a single set-up. The job hierarchy is shown in Figure 4.

Insert Figure 4 Here

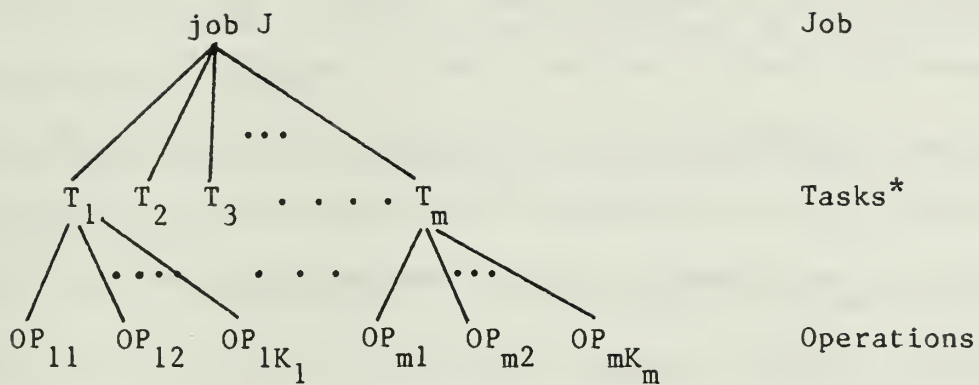


Figure 4. The Job Hierarchy

*Tasks T_1, T_2, \dots, T_m are decompositions of J . Each task, T_i , consists of one or more operations to be performed. There may be precedence orders imposed among the tasks.

III. The Contract Net Framework

A. The Negotiation Procedure

In an FMS, several cells may be assigned to a single job if its operations belong to different part families. To utilize the system resources efficiently, each task, which consists of a group of operations according to the job hierarchy, must be assigned to the most appropriate cell--referred to as the task-sharing process. Task sharing also occurs when a cell is overloaded and needs to distribute some of its tasks.

A widely used metaphor for distributed artificial intelligence is the expert-team metaphor, in which the system under study is viewed as a group of experts that solve problems cooperatively (Chandrasekaran 1981, Lesser and Corkill 1983, Minsky 1979, Yang et al. 1985). Applying this metaphor to the FMS environment, we shall treat each cell's host computer as an agent specialized in planning and managing its local tasks. The whole FMS, then, is a network of such agents with different areas of specialties, and task sharing can be carried out by proper cooperation among the cells. Every cell in the network can play the role of the contractor or the manager for tasks; a cell should "negotiate" with other cells to determine how to share tasks. Specifically, the announcement-bid-award cycle is used to distribute tasks among cells.

The announcement-bid-award cycle is initiated when a manufacturing cell has a task it is not capable of handling. The cell announces the task to other cells to seek assistance. The announcement messages contain three types of task-dependent information:

- a) The eligibility specification: listing the qualification for a cell to submit a bid.
- b) The task abstraction: providing a brief description of the task to allow interested cells to evaluate the task by comparing it with other announced tasks.
- c) The bid specification: specifying the expected format of the bid to be submitted.

A cell in the network keeps an "active-task announcement list" for every machine in the cell and ranks each announced task in the list according to its type. When a machine becomes idle, the cell selects a task at the top of the list and submits a bid to the cell which announced the task (the manager cell). A manager cell may receive several such bids in response to a single task announcement. The award decision is made based on all the bids received, and the manager cell selects the most 'preferable cell based on a ranking function. The successful bidders are informed of the award through award messages from the manager cells and the task will be transferred accordingly. The ranking function used in submitting bids can correspond to various types of scheduling heuristics. For example, when the ranking function is calculated by the total processing time of the operation in a task, assigning task to the lowest bidder is precisely the shortest-processing time (SPT) dispatching rule used in dynamic scheduling.

There may be three types of cells in an FMS where the contract-net framework is applicable: (1) flexible machining cells, where

general-purpose machines are used and the set-up is flexible for performing a wide-ranging family of operations; (2) product-oriented cells, where a certain type of product is manufactured, e.g., gear cell for producing gears; and (3) robot assembly cells, where robots are used for putting sub-assemblies together. Depending on the set-up of a flexible cell or a robot assembly cell, the cell's control unit would give different performance estimates at different moments. The product-oriented cells, on the other hand, have relatively more static functions in terms of the set of operations they perform. For a job requesting an operation that can be performed in these product-oriented cells, the task-announcement message can be directly addressed to the destination cell. The scheduling of jobs can be accelerated by such "focused addressing."

Under the distributed control scheme, the dynamic system information such as cell status, location of parts, position of tools, progress of jobs, etc., is managed by a distributed database. The bidding scheme can be executed dynamically by message passing. Essentially, the bid submitted by each cell reflects the "price" for the cell to embark on the task. The same bidding scheme can be applied to resource allocation as well. That is, if some manufacturing resources--such as cutting tools, fixtures, part programs, and pallets--are not readily available, then the bid should include the price for getting the resources (i.e., the time it will take for the supporting resources to arrive).

B. An Application Example

We shall use an example in this section to illustrate the task negotiation procedure for real-time scheduling. Suppose an FMS consists of five flexible cells, with operation loading shown in Figure 5. For the sake of simplicity, let us assume that the travel time between any two cells is a constant T_M . The bidding duration is T_b . The communication delay for transmitting messages is assumed to be insignificant comparing to T_M , T_b and the processing times. T_a is time taken for awarding a task.

Insert Figure 5 Here

Suppose that a job, Job #003, arrives in the system at time 0. Job #3 has a job hierarchy as shown in Figure 6(a). Figure 6(b) shows that Task 3 cannot begin until both Task 1 and Task 2 are finished. This may happen because Task 1 and Task 2 are performed on two separate components; they are assembled together when finished and then Task 3 is performed. Thus, Tasks 1 and 2 can be announced simultaneously.

Insert Figure 6 Here

Initially, an idle cell-host is selected randomly as the dispatcher of the newly arrived job. A manager cell is designated later after the first round of negotiation (Cell #2 is the manager cell in this case since it is the awardee with longer processing time).

The information flows passed through the communication network to carry out the task-negotiation procedure is shown in Figure 7. Note that the bidding function, f_i , used by Cell i in example is based on

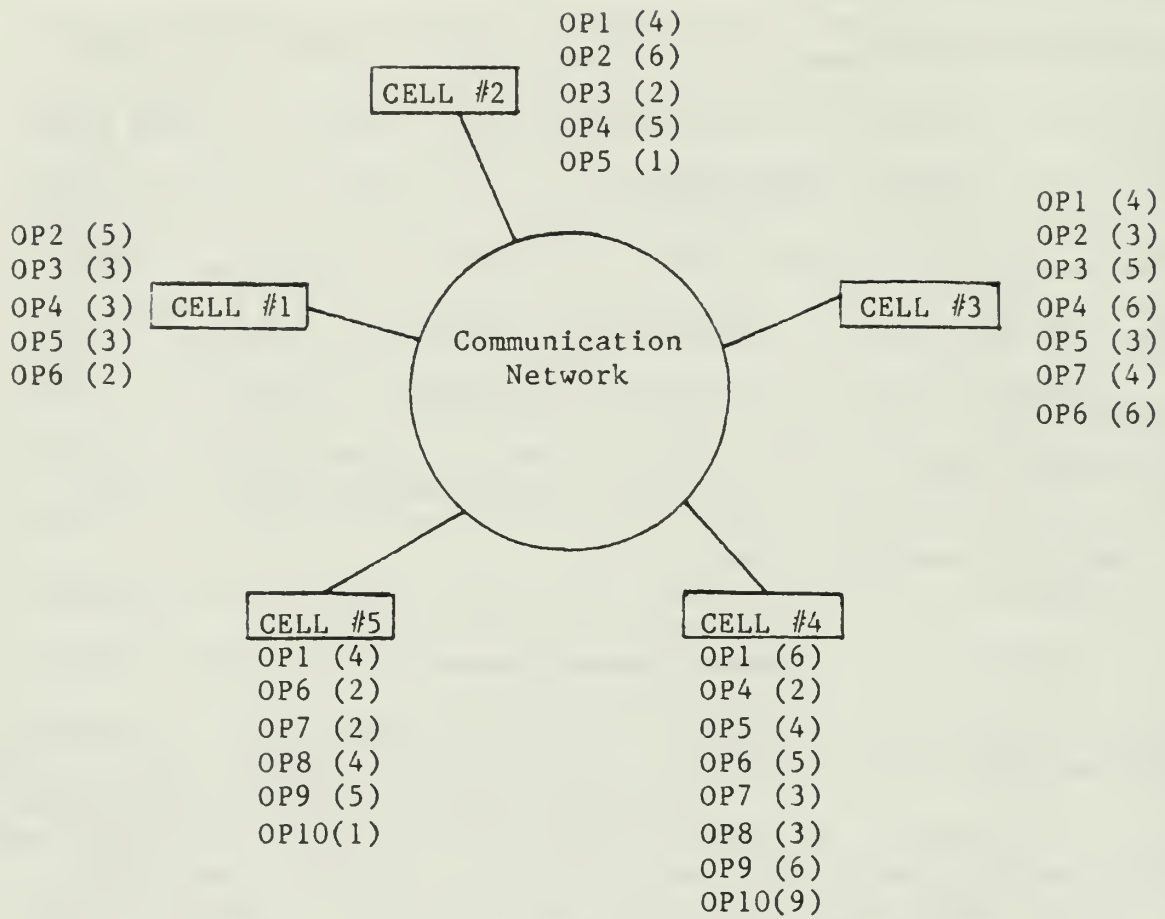
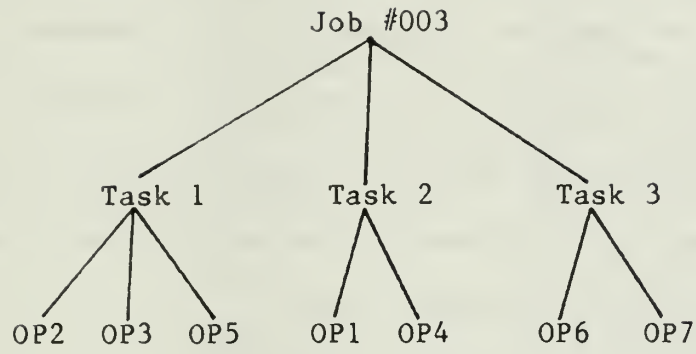


Figure 5. The example system and the operation loading of each Cell (the number in each parenthesis represents the corresponding processing time).



(a)



(b)

Figure 6(a). The job hierarchy in the example.

(b). The precedence relations between the tasks.

the SPT rule (Baker 1974); this rule is used here because of the assumption that the time for transporting parts between any pair of cells is a constant. Shaw (1987) showed that in general situations, the earliest-finishing-time (EFT) rule performs better than the SPT rule. The EFT rule is calculated by the sum of travel time, processing time and expected queueing time.

Insert Figure 7 Here

C. The Negotiation Protocol

To execute the negotiation process by way of communications activities, a set of communication rules--referred to as the protocol--must be established to regulate the interactions between the cells so that they proceed in an orderly fashion. Furthermore, the protocol also has to carry out the task sharing process--the process of distributing tasks among cells. A formal model for the protocol should contain three components:

1. A formalism for the message content. This task-dependent language describes the task information in messages.
2. A formalism for the message format. This formalism, independent of the task domain, is used to format all the messages, so that the message content can be properly interpreted.
3. A formalism for the negotiation process. This formalism is used to describe the proper sequencing of actions to carry out the negotiation procedures among cells. Since there may be several manager cells at the same time, this formalism needs to describe asynchronous, parallel processes.

<u>Time</u>	<u>Agent</u>	<u>Information Type</u>
0	Dispatcher	Task Announcement (Task 1)
0	Dispatcher	Task Announcement (Task 2)
T_b	CELL #1	Bid ($f_1(\text{Task 1}) = 11$)
T_b	CELL #2	Bid ($f_2(\text{Task 1}) = 9$)
T_b	CELL #3	Bid ($f_3(\text{Task 1}) = 12$)
T_b	CELL #2	Bid ($f_2(\text{Task 2}) = 9$)
T_b	CELL #3	Bid ($f_2(\text{Task 3}) = 10$)
T_b	CELL #4	Bid ($f_4(\text{Task 4}) = 8$)
T_b	Dispatcher	Award (Task 1 → CELL #2)
T_b	Dispatcher	Award (Task 2 → CELL #4)
$T_b + T_a$	CELL #2 (Manager)	Accept Task 1
$T_b + T_a$	CELL #4	Accept Task 2
$T_b + T_a + T_M$	CELL #2	(Task 1 Arrives)
$T_b + T_a + T_M$	CELL #4	(Task 2 Arrives)
$T_b + T_a + T_M + 8$	CELL #4	(Finish Task 2)
$T_b + T_a + T_M + 9$	CELL #2	(Finish Task 1)
$T_b + T_a + T_M + 9$	CELL #2	Task Announcement (Task 3)
$T_b + T_a + T_M + 9 + T_b$	CELL #3	Bid ($f_3(\text{Task 3}) = 9$)
$T_b + T_a + T_M + 9 + T_b$	CELL #4	Bid ($f_4(\text{Task 4}) = 8$)
$T_b + T_a + T_M + 9 + T_b$	CELL #5	Bid ($f_5(\text{Task 5}) = 4$)
$T_b + T_a + T_M + 9 + T_b$	CELL #2	Award (Task 3 → CELL #5)
$2x(T_b + T_a) + T_M + 9$	CELL #5	Accept Task 3
$2x(T_b + T_a + T_M) + 9$	CELL #5	(Task 3 Arrives)
$2x(T_a + T_a + T_M) + 13$	CELL #5	(Finish Task 3)
		Job #003 exits the system

Figure 7. The information flows for task negotiation in the example problem.

The first two formalisms can use the context free grammar with BNF expressions. This method is fairly standard in modeling protocols and will not be discussed here. Details of it can be found in (Tanenbaum 1981, Teng and Liu 1978). Subsequent discussions focus on the formalism for task negotiation.

IV. Modeling and Programming the Negotiation Process

A. A Model for the Negotiation Process

A good model for the negotiation process must describe two aspects of task negotiation:

1. a procedural representation of the communication and coordination mechanisms between the cells; and
2. a declarative representation of the local problem solving process within a cell.

We use the augmented Petri net model (Dubois and Steckle 1983, Nelson et al. 1983, Zisman 1978) to achieve these requirements. The augmented Petri net is an integration of two representation models: production rules are used to model the individual events (the declarative representation), and the Petri net is used to model the interactions and temporal relationships between these events (the procedural representation). The augmented Petri net model has been proven effective in modeling asynchronous, concurrent processes where the combination of state variables grows exponentially. In it, each transition corresponds to a production rule and the Petri net structure of the model can be viewed as the interactions between the productions. To understand the mechanism of an augmented Petri net, let us first review some features of the Petri net as a modeling tool.

Originally designed to model process concurrency and precedence relations, the Petri net model has been used to model, specify, and verify communication protocols (Peterson 1981). The definition of the Petri net follows:

Definition 1 (Petri Net)

A Petri net, W , is a quadruple, $W = \langle P, T, I, O \rangle$, where P is the set of places, T is the set of transitions; $I: T \rightarrow P^*$ defines the input function, and $O: T \rightarrow P^*$ defines the output function.

A place is marked if it has one or more tokens; a transition is enabled if each of its input places are marked. The firing of an enabled transition removes one token from each of its input places and adds one token to each of its output places. A token distribution among the available places in a Petri net is called a marking of the net.

Corresponding to each Petri net and an initial marking, Petri net language is defined as follows:

Definition 2 (Petri Net Language)

If there exists a Petri net, $W = \langle P, T, I, O \rangle$, a labelling function l for the transition $l: T \rightarrow Z$, and an initial marking λ , then all the possible sequences of transition firings constitute the Petri net language:

$$L(l) = \{l(\beta) \in Z^* \mid \beta \in T^* \text{ and } \delta(\lambda, \beta)\}$$

where δ is the next-state function. For a sequence of transitions $t_{j1}, t_{j2}, \dots, t_{jk}$, $\delta(\lambda, t_{j1}t_{j2}t_{j3}\dots t_{jk})$ represents that the firing of the transition sequence, t_{j1}, t_{j2} , up to t_{jk} , is legal.

We now proceed to define formally an augmented Petri net:

Definition 3 (Augmented Petri Nets)

An augmented Petri net is composed of seven elements:

$$APN = \langle P, T, I, O, \lambda, AP, D \rangle$$

where $\langle P, T, I, O \rangle$ is a Petri net as defined in Definition 1; λ is the initial marking of this net. The set of transitions, T , also defines the set of productions, with each transition corresponding to one production rule. D is the set of database elements in the production system and AP is the set of active productions whose conditions are satisfied by D .

A transition t in T is firable iff

- (1) $t \in AP$; and
- (2) $I(t)$ is marked; $I(t)$ represents the set of input places of the transition t .

In the augmented Petri net model, since there is a production corresponding to every transition, we can label the transition and the associated production rule with the same labelling function. The Petri net language in the augmented Petri net can thus be seen as either the set of all possible sequences of transitions or, alternatively, as the set of all allowable sequences of production rule invocations. If each transition corresponds to an activity, the Petri net language generates the correct sequence of activities.

Task negotiations for several tasks are usually executed concurrently. The manager cell may be ranking the incoming bids while the potential contractors at the same time are collecting task-announcements and deciding on whether to submit bids. Consequently, the transfer of messages (e.g., task-announcements, bids) from one cell to another requires synchronized activities among the cells involved. Augmented Petri nets can help ensure the correct implementation of these synchronized activities.

To use the augmented Petri net model, the negotiation process can be represented by two subsets: one (Figure 8(a)) models the necessary actions of the manager cell who announces a task to other cells, processing the incoming bids and awards the task to the selected cell; the other sub-net (Figure 8(b)) models the corresponding actions of the cells who receive the task-announcement (the contractor cells). This sub-net deals with the decision on submitting bids.

Insert Figure 8 Here

Each activity in the process is represented by a production rule, and the interactions among these activities are represented by the Petri net. Each transition in the Petri net (denoted by a bar in the figures) corresponds to one production rule. When a transition is enabled (i.e., all input places are marked), the corresponding rules will determine the firing condition.

Table 1 lists the set of production rules that correspond to the transitions in the two augmented Petri nets in Figures 8(a) & (b). Rules T1 to T9 correspond to the task-announcement process of the

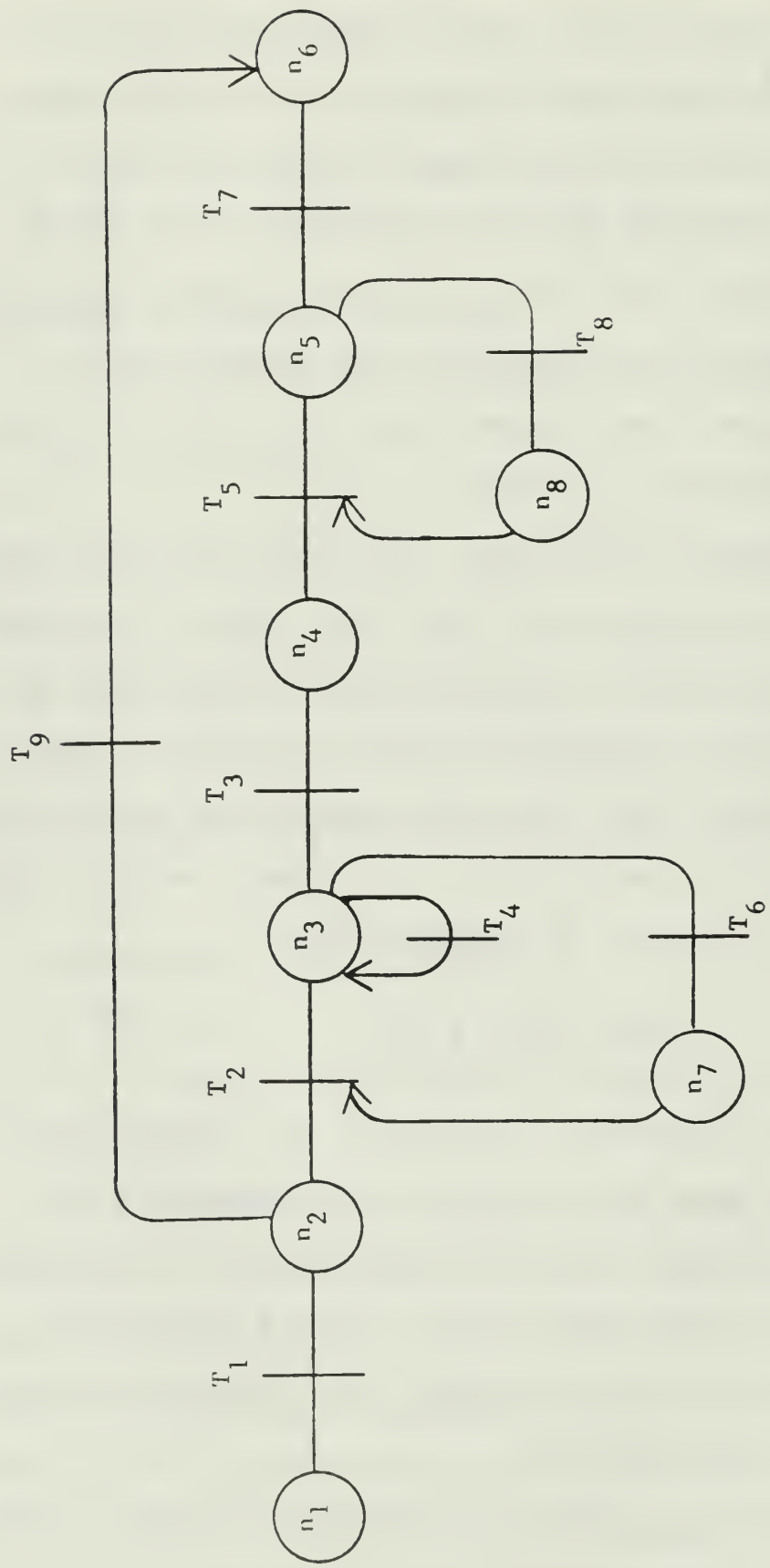


Figure 8(a) The Task-Announcement Process

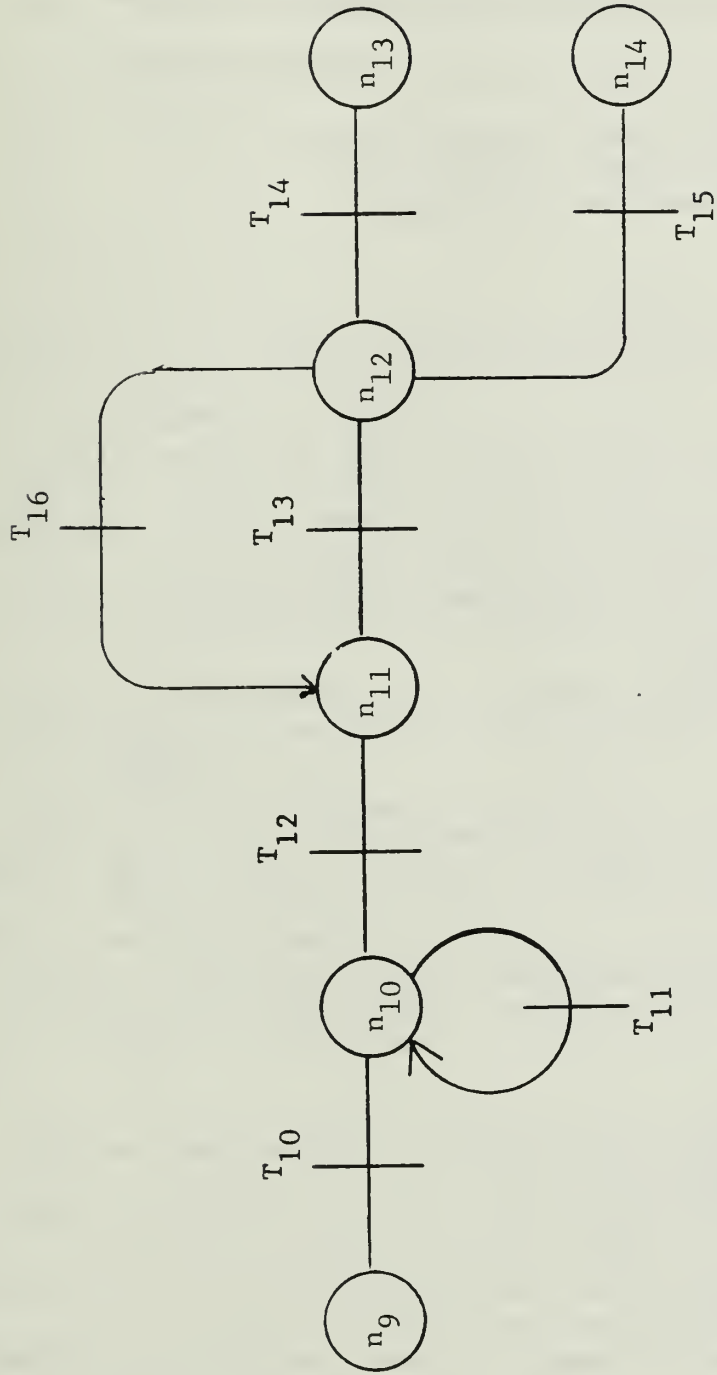


Figure 8 (b) The Bidding Process

manager cell; rules T10 to T16 correspond to the bid-submitting process of the contractor cell. At each step in the process, the augmented Petri nets guide the negotiation process of all cells so that the activities for task negotiation are correctly carried out. The definition of the places used in the nets is shown in Table 2.

Insert Tables 1 and 2 Here

B. Controlled Production Systems

The Petri net language discussed in Definition 2 can serve as the "control language" to regulate the invocation of production rules in the production system during its inference process. Such a production system whose control structure is represented explicitly is called a controlled production system.

The control language in effect guides the allowable sequences of production invocations, i.e., a production would not be considered unless it is accepted by the control language. At each stage of the execution, the control language acts to "focus" the control on a subset of the productions and prohibits the other productions from being invoked.

When the Petri net language (Definition 2) is employed as the control language, it can help ensure that the transitions are fired in the right sequence according to the corresponding Petri net. Furthermore, since each transition represents a production rule in an augmented Petri net, the Petri net language, when used as the control language, essentially dictates the legal sequence of rule invocations.

Table 1

The Production Rules

- T₁ if (NEW-TASK task) then (TASK-INITIALIZATION task)
- T₂ if (TASK-EVALUATE task) then (TASK-ANNOUNCEMENT task)
- T₃ if (BID-RETURN bid) and GEQ time-now deadline) then (BID-PROCESSING bid)
- T₄ if (LEQ time-now deadline) then null
- T₅ if (GT time-now deadline) AND (NE bid-list blank) then (BID-AWARD bid-list)
- T₆ if (GT time-now deadline) AND (EQ bid-list blank) then (REANNOUNCE task)
- T₇ if (REPLY-TO-AWARD accept) then (LIST-ASSIGNMENT task)
- T₈ if REPLY-TO-AWARD reject) then (RE-AWARD task)
- T₉ if (NOT(TASK-EVALUATE task)) then (LIST-AGENDA task)
- T₁₀ if (TASK-ANNOUNCED task) AND (BID-EVALUATE task) then (TASK-RANKING task)
- T₁₁ if (EQ(PROCESSOR-FOR-TASK task) busy) then (LIST-ACTIVE-TASK-ANNOUNCEMENT task)
- T₁₂ if (EQ(PROCESSOR-FOR-TASK task) idle) then (BID-REPLY (BID-SELECT a-t-a-1))
- T₁₃ if (GEQ time-now deadline) then (BIDDING task)
- T₁₄ if (BID-REPLY accept) AND (CELL-CONDITION normal) then (LIST-AGENDA task) AND (REPLY-TO-AWARD accept)
- T₁₅ if (BID-REPLY accept) AND (CELL-CONDITION not-normal) then (REPLY-TO-AWARD reject)
- T₁₆ if (BID-REPLY reject) then (RE-BIDDING(BID-SELECT a-t-a-1))

Table 2

The Definition of the Places Used in the Petri Nets

<u>Place</u>	<u>Description</u>	<u>Successor(s)</u>	<u>Condition for Choosing the Successor</u>
N_1	a task needs to be done	N_2	
N_2	initiating the task	N_3 N_6	when the task needs to be assigned when the task can be executed locally
N_3	broadcasting the task-announcement message	N_4 N_7 N_3	when the submission deadline is reached when no bid is submitted before the deadline
N_4	a bid is submitted	N_5	
N_5	a bidder is selected to award the task to	N_6 N_8	the task is successfully assigned when the awardee rejects the task
N_6	the task is assigned to a cell	null	
N_7	the task needs to be reannounced	N_3	
N_8	the awardee rejected the assigned task; re-award is needed	N_5	
N_9	when a task-announcement message is received	N_{10}	
N_{10}	ranking the task along with other received announcement	N_{10}	when the cell is busy

Table 2 (continued)

<u>Place</u>	<u>Description</u>	<u>Successor(s)</u>	<u>Condition for Choosing the Successor</u>
N_{11}	select a task to bid on	N_{12}	
N_{12}	submit a bid	N_{13} N_{14} N_{11}	accept the task if the cell is normal reject the award if the cell is faulty when the bid is not successful
N_{13}	the task is awarded		
N_{14}	reject the task because of faulty cell conditions		

Using control language in a rule-based production system gives two advantages: first, the execution becomes more efficient because the control language reduces the applicable set of production rules by eliminating irrelevant production rules from consideration; second, since the control structure is explicitly represented by this control language, it can be modified without having to change the contents of the production system. This separation between control and programs is an important feature of the knowledge-based programming (Georgeff 1982, Kowalski 1979). Now let us define a controlled production system and its relation with the augmented Petri net model.

Formally, a production system can be defined as follows:

Definition 4 (Production System)

A production system is a triple

$$PS = \langle R, D, h \rangle$$

where R is the set of production names, D is the set of database elements, and h is the interpretation of the production R , represented as:

$$h: R \rightarrow (q, r)$$

q is the set of conditions and r the set of actions corresponding to R . The state of the production system is defined by the contents of the database elements. When the conditions of a production P_1 , denoted $q(P_1)$, is satisfied by the current database, then P_1 is said to be invocable. If P_1 is invoked, then the database is transformed to a new state, denoted by $r(P_1)$.

Definition 5 (Controlled Production System)

A controlled production system is defined as a quadruple

$$M = \langle R, D, h, C \rangle.$$

The subset $\langle R, D, h \rangle$ is a production system defined in Definition 4. A state in the CPS is defined by a pair $S = \langle u, X_1 \rangle$ with $u \in C$ and $X_1 \in D$. A production rule p is said to be applicable if $up \in C$. A state $\langle up, X_2 \rangle$ is said to be derivable from the state $\langle u, X_1 \rangle$, denoted

$$\langle u, X_1 \rangle \rightarrow \langle up, X_2 \rangle$$

If p is applicable at $\langle u, X_1 \rangle$ ($up \in C$) and the database elements in X_1 satisfy the preconditions of p ($q(X_1) = \text{TRUE}$), then the actions of p change X_1 to X_2 ($r(X_1) = X_2$).

Based on Definitions 1 through 5, we can now propose a theorem which equates the augmented Petri net model to a production system controlled by the Petri net language.

Theorem 1:

For any augmented Petri net

$$\text{APN} = \langle P, T, I, O, \lambda, \text{AP}, D \rangle$$

there exists a controlled production system,

$$M = \langle R, D, h, C \rangle$$

such that APN and M generate the same sequence of production rules.

(The proof is described in the Appendix.)

This isomorphism between (1) the augmented Petri net model and (2) a production system model with a separate control language enables us to deal with the task-sharing problem by using the production system listed in Table 1, and the Petri nets can serve as the control structure. Such an algorithm is similar to the inference procedure used in production systems, as the one used in OPS5 (Forgy 1981). The only difference is that in the beginning of each cycle the algorithm picks the "applicable" rules by the control language.

Procedure - Task-sharing {executed by a manager cell}

Input: a task T, consisting of a set of decomposable subtasks (t_i)

Begin

Repeat {Based on Controlled Production System}

(Step 1) <Control> --

Determine the set of productions accepted by the control language
→ p';

(Step 2) <Selection> --

Select the productions among p' which are invocable → CF {the conflict set};

(Step 3) <Conflict resolution> --

Select a production from the CF set according to the conflict resolution strategy;

(Step 4) <Execution> --

Activate the "then" part of the selected production;

Until the goal condition appears in the database;

end {task-sharing}.

The standard production system employs just steps 2-4 as the inference cycle. Step 1 is used to select only those rules whose corresponding transitions are firable by the current marking in the Petri

net. That is, the incoming places of those transitions all have a token. This condition is guided by the Petri-net language serving as the control language. Step 2 to Step 4 is the "recognition-action cycle" used in standard production systems. A rule is invocable whenever there are database elements that satisfy the conditions of the rule. If more than one rule is invocable, then these rules are collected into the conflict set CF. A conflict resolution strategy is used in Step 3 to select one rule from the set CF. It can be as straightforward as to select the first production rule that is applicable (Nau 1983). OPS5 provides two conflict-resolution strategies called LEX and MEA that make it easy to add production to an existing set and have the new productions fired at the right time. The underlying firing rules for both LEX and MEA strategies are aimed at achieving the following: (1) preventing rules in CF from executing more than once; (2) directing the inference engine to attend to the most recent data in working memory; and (3) giving preference to rules with more specific conditions. Besides these recency rules in which the selection criterion is based on the amount of time the data elements have been in the working memory, other selection rules available include the special case rules and the distinctiveness rules (McDermott and Forgy 1978).

Step 4, in turn, makes changes in the database according to the "then" component of the selected rule. The cycle continues until either of two conditions occurs:

- (1) the goal state is derived and the inference procedure is successful; or

(2) the goal state has not been achieved, but the conflict set in Step 2 is empty.

In the second case, the inference procedure has failed; an exception handling routine will be called upon to take over.

C. Performance

The performance of the task-sharing algorithm is affected by two factors: the performance of (1) the inference procedure in the controlled production system and (2) the negotiation process among the cells. The former is determined by the organization and search strategies of the production system; the latter is determined by the parameters of the negotiation protocol. A simulation study evaluating the performance of the negotiation process with a variety of ranking functions is reported in Shaw (1987).

For a conventional rule-based production executing data-directed inference procedure, the cycle time of the recognize-act cycle is:

$$\begin{aligned} T.\text{cycle} &= T.\text{condition-match} + T.\text{action} \\ &= (P \times M \times T.\text{match}) + ((A/P) \times T.\text{act}), \end{aligned}$$

where P = size of the production system;

M = size of the database (number of predicate literals);

$T.\text{match}$ = Average time to find a match between preconditions and the database elements;

A = number of action elements of all rules;

$T.\text{act}$ = average time needed to execute the action parts of a rule; and

A/P = average number of actions of a rule.

Since the task-sharing algorithm utilizes a control language to screen the applicable production rules first, the cycle time of the production system is modified as

$$T.\text{cycle}' = T.\text{control} + (P' \times M \times T.\text{match}) + ((A/P) \times T.\text{act})$$

where $T.\text{control}$ is the time needed to locate the applicable productions by checking the control language. In our case, it is the time taken for the Petri net language to find the firable transitions based on the current markings. P' is the size of the active production rules decided by the control language; in general, $P' < P$.

V. Summary

The production system, the inference engine, and the database for executing task-sharing are integrated into a knowledge-based system implemented in every cell host. The major function of this component is to distribute tasks and to coordinate the problem-solving activities--primarily planning and scheduling--among the cells. This paper also shows a network-wide negotiation procedure for achieving task sharing.

The major thrust of using augmented Petri nets and the corresponding controlled production system to carry out the negotiation procedure is that it provides a suitably powerful modeling language for executing the process. Because of the problem-solving nature of the negotiation procedure, the approach also has flexibility in achieving task sharing intelligently. More research needs to be done in exploring the optimization behavior of the negotiation process. Nevertheless, the knowledge-based approach has proved very effective in its performing the planning and control functions. In particular: (1) It enforces

the separation of the knowledge description and the control structure. Knowledge is described by production rules and the control structure is represented by the augmented Petri nets. (2) It applies a uniform approach to both the planning within each cell and the task sharing among the cells. As such, each cell's knowledge based contains two types of knowledge: the knowledge for planning and scheduling and the knowledge for task sharing, coordination, and communication. (3) It uses a decentralized scheme to perform planning and control. The approach, therefore, enjoys the benefits associated with distributed processing systems, such as graceful degradation, modularity, extensibility, improved performance, and reliability (Enslow 1977, Cristian and Skeen 1987).

Using the knowledge-based approach for real-time planning and scheduling also permits incorporation of general heuristic knowledge. In this context the information system for an FMS becomes a system with networked knowledge sources cooperating to carry out the manufacturing process. This viewpoint is further elaborated in a separate paper (Shaw 1987).

Appendix

Proof for Theorem 1.

For an augmented Petri net APN, the first five elements $\langle P, T, I, O, \lambda \rangle$ can define a Petri net language $L(1)$. (Definition 2)

Since the active set of productions, AP, in APN is generated by matching preconditions of the productions in T against the database elements in D, AP is derivable from the production system $\langle R, D, h \rangle$ in M. (Definition 4)

Now, if we let the Petri net language $L(1)$ in APN be the control language C in M, then:

(1) (\rightarrow) If a transition t is firable in APN, t must satisfy (a) $t \in AP$ and (b) $I(t)$ is marked. These are equivalent to the conditions (a) the production t is invocable in the production system $\langle R, D, h \rangle$ and (b) t is accepted by the language C. Therefore, t is applicable in M. (Definition 5)

(2) (\leftarrow) If a production p is applicable in M, p must satisfy (a) p is invocable in $\langle R, D, h \rangle$ and (b) p is accepted by the control language C; these conditions are equivalent to the conditions (a) $p \in AP$ and (b) $p \in L(1)$; therefore the transition corresponding to p is firable in $\langle P, T, I, O, \lambda \rangle$. Thus, p is also firable in APN.

(Definition 3)

Q.E.D.

References

- [1] Baker, K., Introduction to Sequencing and Scheduling (New York: John Wiley & Sons), (1974).
- [2] Bochman, G. V. and Sunshine, C. A., "Formal methods in communication protocol design," IEEE Transactions on Communications, Vol. COM-28, No. 4, pp. 624-631, (April, 1980).
- [3] Bourne, D. and Fussell, P., "Designing programming languages for manufacturing cells," The Robotics Institute, CMU-R1-Tr-82-5, Carnegie-Mellon University, (1982).
- [4] Buchanan B. and Duda, R., "Principles of rule-based expert systems," Computer Science Department, HPP-82-14, Stanford University, (1982).
- [5] Chandrasekaran, B., "Natural and social system metaphors for distributed problem solving," IEEE Trans. on Systems, Man and Cybernetics, Vol. 11, No. 1, pp. 1-5, (January 1981).
- [6] Costa, A. and Garetti, M., "Design of a Control System for a Flexible Manufacturing Cell," Journal of Manufacturing Systems, Vol. 4, No. 1, p. 65, (July 1985).
- [7] Cristian, F. and Skeen, D., "Forward: Special Issue on Distributed Systems," IEEE Transactions on Software Engineering, Vol. SE-13, No. 1, pp. 1-2, (Jan. 1987).
- [7] Cutkosky, M., "Precision machining cells within a manufacturing system," The Robotics Institute, Carnegie-Mellon University, (1983).
- [8] Davis, R. and Smith, R., "Negotiation as a metaphor for distributed problem solving," Artificial Intelligence, Vol. 20, pp. 63-109, (1983).
- [9] Dubois, D. and Stecke, K., "Using Petri Nets to represent production processes," in Proc. of the 22nd IEEE Conference on Decision and Control, San Antonio, TX, pp. 1062-1067, (Dec. 1983).
- [10] Enslow, P. H., "What is a Distributed Data Processing System?" Computer, pp. 13-21, (Jan. 1978).
- [11] Fikes, R. E., Hart, P. E. and Nilson, N. J., "Learning and executing generalized robot plans," Artificial Intelligence, 3(4), pp. 251-288, (April 1972).
- [12] Fikes, R. E. and Nilson, N. J., "STRIPS: A new approach to the application of theorem proving to problem solving," Artificial Intelligence, 2(3/4), pp. 189-208, (1971).

- [13] Forgy, C., OPS5 User's Manual. Department of Computer Science, CMU-CS-81-135, Carnegie-Mellon University, (1981).
- [14] Georgeff, M., "Procedural control in production systems," Artificial Intelligence, Vol. 18, pp. 175-201, (Jan, 1982).
- [15] Hewitt, C., "Viewing control structures as patterns of passing messages," Artificial Intelligence, Vol. 8(3), pp. 323-364, (March 1977).
- [16] Hutchinson, G. K., "Flexibility is key to economic feasibility of automating small batch manufacturing," Industrial Engineering, pp. 77-86, (June 1984).
- [17] Kiebertz, R., "A hierarchical multicomputer for problem-solving by decomposition," Proceedings of the IEEE Distributed Computing Systems, pp. 631-71, (1979).
- [18] Kowalski, R., "Algorithm = logic + control," Communications of the ACM, Vol. 22, pp. 424-436, (August 1979).
- [19] Lesser, V. and Corkill, D., "The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks," The A.I. Magazine, Vol. 4, No. 3, pp. 15-33, (July 1983).
- [20] Lozano-Perez, T., "Robot programming," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, A.I. Memo No. 698, (1982).
- [21] McDermott, D. and Forgy, C., "Production system conflict resolution strategies," Pattern Directed Inference Systems, D. Waterman and F. Hayes-Roth, Eds., Academic Press, (1978).
- [22] McLean, C., Mitchell, M. and Barkmeyer, E., "A computer architecture for small-batch manufacturing," IEEE Spectrum, pp. 59-64, (May 1983).
- [23] Merchant, M. E., "Production: a dynamic challenge," IEEE Spectrum, Vol. 20, No. 5, pp. 36-39, (May 1983).
- [24] Minsky, M., "The society theory of thinking," Artificial Intelligence - An MIT Perspective, P. Winston, Ed., MIT Press, (1979).
- [25] Nau, D., "Expert Computer Systems," IEEE Computer, Vol. 16, No. 2, pp. 63-85, (Feb. 1983).
- [26] Nau, D. and Chang, T., "Prospects for Process selection using artificial intelligence," Computers in Industry, Vol. 4, pp. 253-263, (1983).

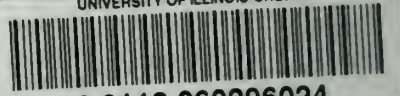
- [27] Nelson, R., Haitb, L. and Sheridan, P., "Casting Petri nets into programs," IEEE Trans. on Software Engineering, Vol. SE-9, No. 5, pp. 590-602, (September 1983).
- [28] Nelson, R., Principles of Artificial Intelligence. Palo Alto: Tioga, (1980).
- [29] Peterson, J., Petri Net and the Modeling of Systems. Englewood Cliffs, NJ: Prentice-Hall, (1981).
- [30] Ranky, P., Computer Integrated Manufacturing, Englewood Cliffs, NJ: Prentice Hall, (1986).
- [31] Sacerdoti, E. D., A Structure for Plans and Behavior. New York: North-Holland, (1977).
- [32] Shaw, M. J., "A Two-Level Approach to Scheduling in Computer Integrated Manufacturing," Proceedings of the NBS Symposium on Real-Time Optimization in Automated Manufacturing Facility, National Bureau of Standards, Gaithersberg, MD, (January 1985).
- [33] Shaw, M. J., "FMS Scheduling as Cooperative Problem Solving," Faculty Working Paper No. 1326, Department of Business Administration, University of Illinois, (1987).
- [34] Shaw, M. J. and Whinston, A. B., "Automatic Planning and Flexible Scheduling: A Knowledge-Based Approach," Proceedings IEEE International Conference on Automation and Robotics, St. Louis, (March 1985).
- [35] Simpson, J., Hocken, R., and Albus, J., "The automated manufacturing research faculty of the National Bureau of Standards, Journal of Manufacturing Systems, Vol. 1, No. 1, pp. 17-32, (Jan. 1982).
- [36] Smith, R. G., "The contract net protocol: high level communication and control in a distributed problem solver," IEEE Transactions on Computer, Vol. 29, pp. 1104-1113, (1980).
- [37] Tanenbaum, A., Computer Networks. New Jersey: Prentice-Hall, (1981).
- [38] Teng, A. and Liu, M., "A formal approach to the design and implementation of network communication protocol," Proceedings of COMPSAC, pp. 114-128, (1978).
- [39] Vere, S., "Planning in time: windows and duration for activities and goals," Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 3, pp. 246-266, (May 1983).

- [40] Wilkins, D. E., "Domain independent planning: representation and plan generation," *Artificial Intelligence*, Vol. 22, No. 3, pp. 269-301, (April 1984).

- [41] Yang, J. D., Huhns, M. N., and Stephens, L. M., "An Architecture for Control and Communications in Distributed Artificial Intelligence Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, V. SMC-15, No. 3, pp. 316, (May/June 1985).

- [42] Zisman, M., "Use of production systems for modelling asynchronous concurrent processes," in *Pattern Directed Inference Systems*, D. Waterman and F. Hayes-Roth, Eds., Academic Press, (1978).

UNIVERSITY OF ILLINOIS-URBANA



3 0112 060296024