Task Translation and Integration Specification in Intelligent Machines

Fei-Yue Wang, *Member, IEEE*, and George N. Saridis, *Fellow, IEEE*

Abstract— **Intelligent machines are defined to be hierarchically intelligent control systems composed of three levels: the organization level, the coordination level, and the execution level. This paper presents an analytical model for the coordination level of intelligent machines, which, together with the established mathematical formulation for the organization level and the well developed control theory for the execution level, completes the first step toward a mathematical theory for intelligent machines. The framework of the coordination level is a tree structure consisting of a dispatcher and a number of coordinators. A new type of transducers, Petri net transducers (PNTs), has been introduced to serve as the basic module in our analytical model. PNTs provide a formal description for the individual processes within the dispatcher and coordinators. The concurrence and conflict among these processes can be represented by PNTs conveniently. Coordination structures are introduced as a formalism for the specification of integration in the coordination level. The task precedence relationship in the coordination process is presented by the Petri nets derived from the coordination structures. These Petri nets also provide us a formal approach of using the concepts and analysis methods in the Petri net theory to investigate the properties of the coordination structures. A case study of modeling an intelligent assembly robotic system has been conducted for the purpose of illustration.**

I. Introduction

THE QUEST to build machines that perform anthropomorphic tasks autonomously or interactively in structured or unstructured environments has a long tradition in the history of human beings. The effort along this direction has been intensified tremendously by the new advancements in modern technology during the past two decades. Such machines, called *intelligent machines* in engineering, will play the key role in various modern and future industries, such as space exploration, robotic systems, and computer integrated manufacturing.

The design of intelligent machines has brought many new challenges to the scientific community. Among them, the control problem is one of the most important issues. Since theories and technologies in conventional automatic control had been known to be inadequate to deal with the diversified aspects in the control of intelligent machines, a new discipline, called *intelligent control*, has emerged for this purpose during the past decade [21].

Manuscript received September 10, 1990; revised May 27, 1992. This work was supported by NASA through the NASA Center for Intelligent Robotic Systems for Space Exploration under Grant #NAGW-1333.

F.-Y. Wang is with the Department of Systems and Industrial Engineering, The University of Arizona, Tucson, Arizona, 85721.

G. N. Saridis is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180.

IEEE Log Number 9208250.

The theory of intelligent controls is still in its early stage of development. Methodological issues are both open and central. Since Fu [8] coined the name of intelligent controls in 1971 as the field of interaction of artificial intelligence and automatic control systems, different ideas for the formalization of the definitions and the structure of intelligent machines have been proposed by and debated among various researchers [2], [4], [13], [15]. An analytical approach has been proposed and pursued by Saridis since the 1970s [15]–[22], which expanded the field of intelligent controls to include three components: artificial intelligence, operations research, and control theory.

The structure of intelligent machines has been defined by Saridis [15]–[22] to be the structure of hierarchically intelligent control systems, composed of three levels hierarchically ordered according to the principle of increasing precision with decreasing intelligence (IPDI) [16], [23], namely:

1) *The organization level* represents the brain of the system with functions dominated by artificial intelligence to reason, to plan, and to make decisions about the organization of tasks;
2) *The coordination level* defines the interface between high and low levels of intelligence with functions dominated by operations research that coordinate the activities of the hardware; and
3) *The execution level* is the lowest level with high requirement in precision with functions dominated by Control Theory to execute the specified tasks. Fig. 1 presents the structure of intelligent machines.

A mathematical formulation for the organization level has been developed by Saridis and Valavanis [22], [24]. The focus of this paper is on a formal theory for the coordination level with a tree topology consisting of a *dispatcher* as the root and a set of *coordinators* as the subnodes (see Fig. 1). The basic requirement for such an analytical model is the establishment of an *information structure* that specifies the necessary precedence relationship of the relevant information processing for the coordination of the diversified activities in this Level. Specifically, the following features should be accomplished: 1) a formal description of each individual process within each system unit (i.e., dispatcher or coordinator) in the Level; 2) a formal specification of the cooperation and connection among system units; 3) a mechanism of control and communication for task processes in system units.

We will address those issues based on the linguistic decision approach developed by Saridis and Graham [9], [19]. A knowledge of basic Petri net theory and formal languages has been assumed throughout this paper. The first part of the

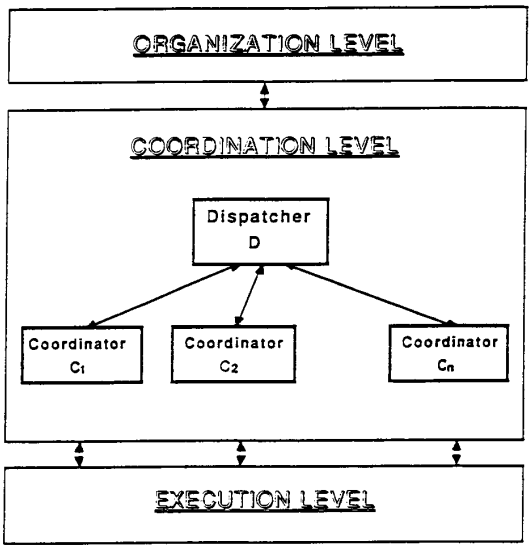


Fig. 1. The structure of intelligent machines.

paper (Sections II–III) presents a new type of language transducers, Petri net transducers (PNTs), for individual process description within a system unit. The second part (Sections IV–VII) introduces the theory of coordination structure for specification of integration among system units. Since both PNTs and coordination structures are Petri net-based models, the execution rule of Petri nets therefore provides naturally a control and communication mechanism for task processes in the coordination level.

In the linguistic decision approach, task processes of the dispatcher and coordinators have been considered as the process of translating the higher level task plans to the lower level control actions by using *vocabulary decision schemata* [19]. Since the decision schemata are grammars, PNTs have been introduced to implement them. There are several classic automata available as the implementation model for the vocabulary decision schemata, such as finite state transducers and pushdown transducers [1]. The reason for choosing PNTs over the other automata is mainly due to the following facts: *First*, the connection between the dispatcher and coordinators is quite an important issue in the modeling of the coordination level; however, it is difficult to use the classic automata to specify these kinds of connections. *Second*, it is inadequate to describe the concurrence of activities in the coordination level by using classic automata. The reason is that the primitive notion of *states* in these automata is intended to represent the status of the entire system modeled at a certain instance, whereas the notion of *places* in Petri nets describes only the status of some components of the system modeled. *Third*, there are no convenient methods currently available to conduct either qualitative or quantitative process analysis for systems modeled by these automata. All these problems can be overcome by Petri net-based models, since it has been shown in both theoretic works and applications, especially in the modeling of manufacturing processes [3], [10], [11], [14], [33]–[36], that Petri nets can be used to specify connections among the system units, to describe concurrence and conflict in the system processes, and to perform qualitative and quantitative process analysis.

The specification for the integration of the dispatcher and coordinators has been achieved by *coordination structures*. A coordination structure is obtained by connecting a set of PNTs with a set of connection points through the connection mappings. Specifically, the use of those coordination structures can enable us to: 1) describe the task translation of the dispatcher and coordinators (through PNTs); 2) represent the individual process within the dispatcher and coordinators (through PNTs); 3) specify the cooperation and connection among the dispatcher and coordinators; 4) perform the process analysis and evaluation; and 5) provide a control and communication mechanism for the real-time monitor or simulation of coordination process. A case study for a primitive intelligent robotic system has been conducted in Section VIII to demonstrate those concepts.

The results presented in this paper are further development of our previous studies on this topic [25], [27]–[31]. They also can be used for the integration specification of other kinds of distributed systems, for example, the software system for the manufacturing message specification (MMS) [25], [32]. The major difference between our linguistic and analytical approach and the traditional programming approach [7] for coordination problem is that our model provides a formal framework for task verification, analysis, synthesis and performance evaluation. The work of using Petri nets to formalize the decision structure of human organizations by Levis and his colleagues [6], [12] is similar to ours, however, their focus is on representation and performance evaluation whereas ours on integration and process analysis.

II. Task Translation and Petri Net Transducers

The task process in the coordination level of intelligent machines is the process of task translation: $L_O \rightarrow L_C \rightarrow L_E$, where L_O represents the set of task plans generated by the task organizer in the organization level, L_C describes the set of control actions in the coordination level, and L_E gives the set of real-time operations to be executed in the execution level. The dispatcher receives the task plans from the organizer, decomposes the plans into coordinator-oriented control actions and dispatches them to the corresponding coordinators. Upon obtaining the control commands from the dispatcher, a coordinator then translates the commands into the operation instructions and sends them down to the appropriate execution devices in the lowest level for real-time execution.

Those translation processes performed by the dispatcher and coordinators can be described by using PNTs as their models. A PNT is a language translator that translates an input task plan into an output plan. Formally,

Definition 1: A Petri net transducer (PNT), M, is a 6-tuple,

$$M = (N, \Sigma, \Delta, \sigma, \mu, F) \text{ where}$$

1) $N = (P, T, I, O)$ is a Petri net with an initial marking μ;
2) Σ is a finite input alphabet;

3) Δ is a finite output alphabet;
4) σ is a translation mapping from $T \times (\Sigma \cup \{\lambda\})$ to finite sets of Δ^*;
5) $F \subseteq R(\mu)$ is a set of final markings.

where λ is the empty string and Δ^* the set of all possible strings over Δ. In applications, Σ may represent the set of primitive tasks for task planning, and Δ the set of primitive operations for task execution. A PNT has three parts: an *input tape* to read input plans (strings over Σ) in, a *Petri net controller* to control the translation process, and an *output tape* to write output plans (strings over Δ) on. The behavior of a PNT can be conveniently described in terms of configurations of the PNT. A *configuration* of PNT M is defined as a triple (m, x, y) where $m \in R(\mu)$ (i.e., the reachability set of N with initial marking μ) is the current state (or marking) of the net N; $x \in \Sigma^*$ is the input string remaining on the input tape with the leftmost character of x under the input head; $y \in \Delta^*$ is the output string emitted up to this point.

A *move* by PNT M is reflected by a binary relation $\Rightarrow_M$ (or $\Rightarrow$, when M is clear) on configurations. Specifically, for all $m \in R(\mu), t \in T, a \in \Sigma \cup \{\lambda\}, x \in \Sigma^*$ and $y \in \Delta^*$ such that $\delta(m, t)$ is defined and $\sigma(t, a)$ contains $z \in \Delta^*$, we write

$$(m, ax, y) \Rightarrow (\delta(m, t), x, yz)$$

where $\delta(m, t)$ is the state transition of Petri nets. We will use $\Rightarrow^*$ to denote the *transitive* and *reflexive* closure of $\Rightarrow$. The *translation* of M is defined to be the set:

$$\tau(M) = \{(x, y) \mid (\mu, x, \lambda) \Rightarrow^* (m, \lambda, y) \text{ for some } m \in F\}$$

$\tau(M)$ will be called a *Petri net translation* or *a Petri transducer mapping*. A string y is said to be an *output* of a string x or x is the *input* of y iff $(x, y) \in \tau(M)$. The *input language* and the *output language* of M are defined to be the following two sets:

$$\alpha(M) = \{x \mid \exists y \in \Delta^*, (x, y) \in \tau(M) \text{ and } \omega(M) = \{y \mid \exists x \in \Sigma^*, (x, y) \in \tau(M)\}$$

Note that in Definition 1, translation mapping σ is allowed to generate string instead of only single alphabet at one instance of translation. However, it is easy to show that string and alphabet translations are mathematically equivalent, and in general, the restriction of single alphabet translation would lead to a much larger Petri net controller.

Example 1: Consider a PNT $M = (N, \Sigma, \Delta, \sigma, \mu, F)$ with the following specification:

$$\Sigma = \{e\}; \Delta = \{a, b, c\}; \sigma(t_1, e) = \{a\}, \sigma(t_2, \lambda) = \{b\}$$
$$\sigma(t_3, \lambda) = \{c\}; \mu = (1, 0, 0); F = \{\mu_f = (0, 0, 1)\}.$$

The Petri net N is given in Fig. 2. When an input string $e^2 = e.e$ is issued to M, it will be translated into an output string a^2cb^2. The detailed translation process is,

$$\begin{aligned}(\mu, e^2, \lambda) &\Rightarrow (\mu_1 = \delta(\mu, t_1) = (1, 1, 0), e, a) \\ &\Rightarrow (\mu_2 = \delta(\mu_1, t_1) = (1, 2, 0), \lambda, a^2) \\ &\Rightarrow (\mu_3 = \delta(\mu_2, t_3) = (0, 2, 1), \lambda, a^2c) \\ &\Rightarrow (\mu_4 = \delta(\mu_3, t_2) = (0, 1, 1), \lambda, a^2cb) \\ &\Rightarrow (\mu_5 = \delta(\mu_4, t_2) = (0, 0, 1), \lambda, a^2cb^2)\end{aligned}$$

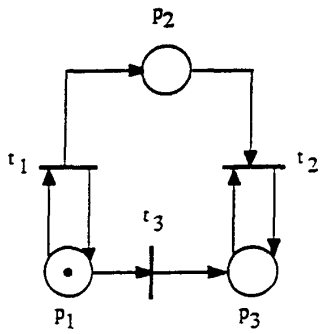


Fig. 2. An example of PNT.

that is,

$$(\mu, e^2, \lambda) \Rightarrow^* (\mu_5, \lambda, a^2cb^2).$$

It is easy to show that in this case

$$\alpha(M) = \{e^n \mid n \geq 0\}, \omega(M) = \{a^ncb^n \mid n \geq 0\}$$

and

$$\tau(M) = \{(e^n, a^ncb^n) \mid n \geq 0\}.$$

A PNT *halts* at configuration (m, ax, y) when no transitions, for which $\sigma(t, a)$ is defined, are enabled at the current marking m. We call m the *deadlock marking* of the PNT. When a deadlock marking occurs, the input string will be rejected. Note that a deadlock marking of a PNT is not necessarily a deadlock marking of its Petri net.

As in the Petri net language literature, PNTs can be classified according to the following three kinds of translation mappings and four kinds of final marking sets:

Definition 2: For a PNT $M = (N, \Sigma, \Delta, \sigma, \mu, F)$, its translation mapping σ is

1) *free translation mapping* if $\sigma(t, a)$ is defined then $a \neq \lambda$. If both $\sigma(t_1, a)$ and $\sigma(t_2, b)$ are defined, then $a \neq b$ if $t_1 \neq t_2$;
2) *λ-free translation mapping* if $\sigma(t, a)$ is defined then $a \neq \lambda$;
3) *λ-transition translation mapping* if no restriction on σ.

Its final marking set F is classified as

1) *L-type* if F is a finite set of markings in $R(\mu)$;
2) *G-type* if $F = \{m \in R(\mu) | m \geq m_i$ for some $i, i = 1, \cdots, n\}$;
3) *T-type* if $F = \{m \in R(\mu) | m$ is a deadlock marking of $N\}$;
4) *P-type* if $F = R(\mu)$.

The PNT M in Example 1 has a L-type final marking set with a λ-transition translation mapping. Assume now that F is a G-type set, say, $F = \{m \in R(\mu) | m \geq (0, 0, 1)\}$, and the translation mapping is unchanged, then, $\tau(M) = \{(e^n, a^ncb^k) | n \geq k \geq 0\}$. When F is T-type, the result is the same as that for L-type, since (0, 0, 1) is the only deadlock marking of N. When F is P-type, we have, $\tau(M) = \{(e^n, a^n), (e^n, a^ncb^k) | n \geq k \geq 0\}$.

A PNT with a free translation mapping will be called a *free* PNT; a PNT with a L-type final marking set will be called a L-type PNT, and so on. There exist 12 classes of PNTs resulting from the cross product of the four types of the final markings and the three types of translation mappings. Since the language properties of *L-type* PNTs can be easily determined and their expression power is sufficient for task translation in intelligent machines, we will consider only *L-type* PNTs in this paper, and from now on by PNTs we mean *L-type* PNTs.

To discuss the language properties of PNTs, i.e., the problem of what kinds of task plans may be translated or produced by PNTs, we begin with the following special class of PNTs.

Definition 3: A PNT $M = (N, \Sigma, \Delta, \sigma, \mu, F)$ with $N = (P, T, I, O)$ is called a *simple* PNT (SPNT) iff:

1) $\forall t \in T$ there exists one and only one $a \in \Sigma \cup \{\lambda\}$ such that $\sigma(t, a)$ is defined;
2) $\forall t \in T$ and $a \in \Sigma \cup \{\lambda\}$, if $\sigma(t, a)$ is defined, then $|\sigma(t, a)| = 1$.

Condition 1) implies that in a SPNT, a transition of its Petri net is responsible for translating one and only one input character; and condition 2) implies that each transition has only one output string to translate its input character. Therefore, once a transition is located, a SPNT is deterministic in the emitting of the output string. The following theorem indicates the importance of this type of PNTs.

Theorem 1: For any PNT M, there exists a SPNT M' such that $\tau(M') = \tau(M)$.

All the proofs in this paper are given in the Appendix at the end of the paper.

This theorem indicates that as far as language property is concerned, SPNTs and PNTs are equivalent. However, like the colored Petri nets versus the ordinary Petri nets, a general PNT sometimes offers us a more compact description than a SPNT does.

Based on Theorem 1, it is easy to show that the language property of PNTs can be characterized by the following theorem:

Theorem 2: The input and output languages of a PNT are both Petri net languages.

This theorem guarantees that PNTs can be used as the consistent models for the dispatcher and coordinators, i.e., it is always possible to construct coordinators to process the tasks issued by a dispatcher, if they are to be described by PNTs. Since Petri net language takes regular language as its proper sub-language, the theorem also indicates that the set of PNTs include the set of finite state transducers as its proper subset.

III. Synchronization of Petri Net Transducers

The synchronous composition operator is a mechanism used to coordinate the operations of several PNTs. The basic idea is to let PNTs read a common input tape and to synchronize their translation processes by using common input characters (or input tasks). The need for such synchronization rises in various situations where system units have to cooperate in order to accomplish some common tasks. For example, consider an assembly robotic system consisting of two arms with independent controllers. The tasks for the robotic system are to move parts from one position to the other. To move small parts, the two arms can perform the tasks independently. However, to move large parts, the two arms have to perform the tasks together, which requires the coordination of their individual actions. Thus, when a PNT is used as the motion coordinator for each of the two arms, the operations of the two PNTs have to be synchronized.

Definition 4: The *synchronous composition* of two PNTs:

$$M_i = (N_i, \Sigma_i, \Delta_i, \sigma_i, \mu_i, F_i),$$
$$N_i = (P_i, T_i, I_i, O_i), P_1 \cap P_2 = \emptyset, T_1 \cap T_2 = \emptyset, i = 1, 2,$$

is a PNT, denoted by $M_1 || M_2$, specified as the followings,

$$M_1 | M_2 = (N, \Sigma, \Delta, \sigma, \mu, F), N = (P, T, I, O)$$

with

$$P = P_1 \cup P_2,$$
$$T = T_{11} \cup T_{12} \cup T_{22},$$
$$\Sigma = \Sigma_1 \cup \Sigma_2,$$
$$\Delta = \Delta_1 \cup \Delta_2$$

where

$$T_{11} = \{t_i \mid t_i \in T_1 \& \text{there exist} a \in \Sigma_1 - \Sigma_2 \text{ such that } \sigma_1(t_i, a) \text{ is defined}\}$$
$$T_{22} = \{t_i \mid t_i \in T_2 \& \text{there exists} a \in \Sigma_2 - \Sigma_1 \text{ such that } \sigma_2(t_i, a) \text{ is defined}\}$$
$$T_{12} = \{t_{ij} \mid \text{there exists} t_i \in T_1 \& t_j \in T_2 \& a \in \Sigma_1 \cap \Sigma_2 \text{ such that } \sigma_1(t_i, a) \& \sigma_2(t_j, a) \text{ is defined}\}$$

$$I(t) = \begin{cases} I_1(t) & \text{if } t \in T_{11} \\ I_2(t) & \text{if } t \in T_{22} \\ I_1(t_i) + I_2(t_j) & \text{if } t = t_{ij} \in T_{12} \end{cases},$$

$$O(t) = \begin{cases} O_1(t) & \text{if } t \in T_{11} \\ O_2(t) & \text{if } t \in T_{22} \\ O_1(t_i) + O_2(t_j) & \text{if } t = t_{ij} \in T_{12} \end{cases},$$

the initial marking is

$$\mu(p) = \begin{cases} \mu_1(p) & \text{if } p \in P_1 \\ \mu_2(p) & \text{if } p \in P_2 \end{cases}$$

the set of final markings is

$$F = \{m \in R(\mu) \mid \exists m_1 \in F_1 \text{ and } m_2 \in F_2 \text{ such that } \forall p \in P, m(p) = m_1(p) \text{ for } p \in P_1, \; m(p) = m_2(p) \text{ for } p \in P_2\}$$

finally, the translation mapping is

$$\sigma(t, a) = \begin{cases} \sigma_1(t, a) & \text{if } t \in T_{11} \& a \in \Sigma_1 - \Sigma_2 \\ \sigma_2(t, a) & \text{if } t \in T_{22} \& a \in \Sigma_2 - \Sigma_1 \\ \{z_i z_j, z_j z_i \mid z_i \in \sigma_1(t_i, a) \; \& z_j \in \sigma_2(t_j, a)\} & \text{if } t = t_{ij} \in T_{12} \; \& a \in \Sigma_1 \cap \Sigma_2 \\ \text{undefined} & \text{otherwise} \end{cases}.$$

With this definition, it is not difficult to show that a move by $M = M_1 \parallel M_2$ can be represented as

$$((m_1, m_2), ax, y) \Rightarrow \begin{cases} ((\delta(m_1, t_i), m_2), x, yz_i) & \text{if } a \in \Sigma_1 - \Sigma_2 \\ ((m_1, \delta(m_2, t_j)), x, yz_j) & \text{if } a \in \Sigma_2 - \Sigma_1 \\ ((\delta(m_1, t_i), \delta(m_2, t_j)), x, yz_iz_j) & \text{or} \\ ((\delta(m_1, t_i), \delta(m_2, t_j)), x, yz_jz_i) & \text{if } a \in \Sigma_1 \cap \Sigma_2 \end{cases}.$$

where $t_i \in T_1$ and $t_j \in T_2$, $z_i \in \sigma_1(t_i, a)$ and $z_j \in \sigma_2(t_j, a)$. That is, the input characters in the alphabet of one PNT, but not the alphabet of the other, are translated by that PNT alone, and the input characters in the alphabet of two PNTs are translated by both the PNTs simultaneously in an arbitrary order.

Example 2: Consider the example mentioned in the beginning of this section. Petri nets for motion processes of the two arms are given in Fig. 3(a). The corresponding PNTs are specified as

$$\Sigma_i = \{\mathrm{G}_i, \mathrm{GB}_i, \mathrm{GR}_i, \mathrm{MSP}_i, \mathrm{RSP}_i, \mathrm{MLP}, \mathrm{RLP}\},$$
$$\Sigma_1 \cap \Sigma_2 = \{\mathrm{MLP}, \mathrm{RLP}\}$$
$$\Delta_i = \{\text{motion control algorithms and grasping algorithms}\}, i = 1, 2;$$

where G means go to the grasp position; GB means go back to home position; GR means grasp a part; MSP means move with a small part; RSP means release a small part; MLP: move with a large part; RLP means release a large part. The translation mapping is

$$\sigma_i(t_{i1}, G_i) = \sigma_i(t_{i2}, \mathrm{GR}_i) = \sigma_i(t_{i3}, \mathrm{MSP}_i) = \sigma_i(t_{i3}, \mathrm{MLP}) = \sigma_i(t_{i4}, \mathrm{RSP}_i) = \sigma_i(t_{i4}, \mathrm{RLP}) = \sigma_i(t_{i5}, \mathrm{GB}_i) = \Delta_i$$

That is, a general task cycle for an arm is to go to the grasp position; to grasp a part; to move with the part; to release the part; and to go back to home position.

Fig. 3(b) gives the PNT that resulted from the synchronous composition of two PNTs for the two arms. Two new transitions are introduced in the synchronous composition and their translation mappings are specified as

$$\sigma(t_{33}, \mathrm{MLP}) = \sigma(t_{44}, \mathrm{RLP}) = \Delta_1 \times \Delta_2$$

From the synchronous composition, we see that when a task for moving a small part is issued, say, a task G_1G_2 GR_2 MSP_2 GR_1 MSP_1 RSP_2 RSP_1 GB_2 GB_1, the composition will be exactly the same as if the two arms work independently. For example, arm two can move with its part before arm one has grasped its part. However, when a task for moving large parts is issued, say a task, $G_1G_2\mathrm{GR}_2\mathrm{GR}_1\mathrm{MLP}$ RLP $\mathrm{GB}_2\mathrm{GB}_1$, then the composition will force two arms to work synchronously for the common actions. For example, transition t_{33} for task MLP cannot be enabled until both of its input places have a token, i.e., moving with the part cannot be executed until both arms have grasped the same large part. A similar situation happens to the transition t_{44} for task RLP.

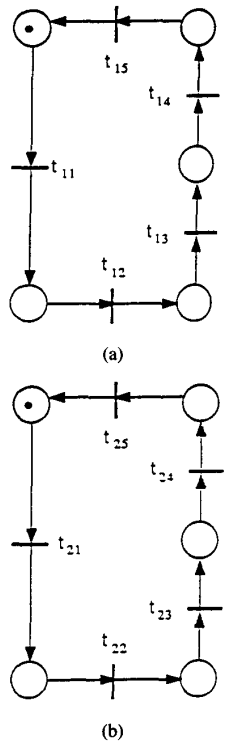


Fig. 3. (a) Individual PNTs for two arms. (b) The synchronous composition of two PNTs for two arms.

Since the synchronous composition of two PNTs is still a PNT, the synchronous composition can be directly generalized to more than two PNTs by defining

$$M_1 \| M_2 \| \bullet\bullet\bullet \| M_{k-1} \| M_k \equiv (M_1 \| M_2 \| \bullet\bullet\bullet \| M_{k-1}) \| M_k.$$

IV. Integration Specification and Coordination Structures

The Petri net transducers describe the individual process of task translating in the dispatcher and coordinators. To specify the connection among them, we introduce the model of coordination structures in this section. A coordination structure consists mainly of three parts: a set of system units, a set of connection points, and a set of connection mappings from the units to the connection points. The connection points represent places where the units exchange their information. The connection mappings describe the links between the units and the connection points. Various types of coordination structures can be defined by proposing different forms of connection points or by designing different patterns of connections mappings.

For the integration specification in the coordination level of intelligent machines, PNTs have been used to describe the system units, i.e., the dispatcher and coordinators, and the connection mappings are classified into two classes: receiving mappings for information acquisition and sending mappings for information distribution. Starting from the general cases to the more specific cases, we specify four types of coordination structures in the sequel.

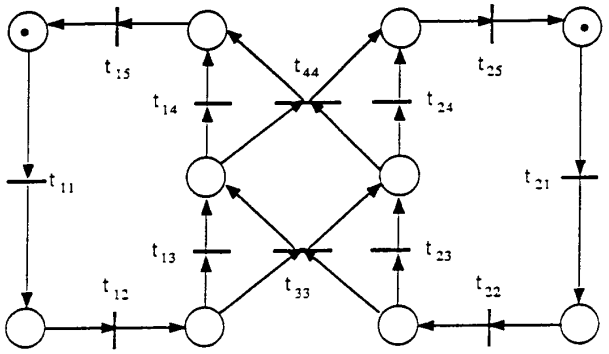


Fig. 4. The configuration of the coordination structures.

Definition 5: A coordination structure CS with an arbitrary connection is defined to be a 7-tuple,

$$\text{CS} = (D, C, F, R_D, S_D, R_C, S_C) \text{ where}$$

1) *Dispatcher*: $D = (N_d, \Sigma_o, \Delta_o, \sigma_d, \mu_d, F_d)$ is a PNT with a Petri net $N_d = (P_d, T_d, I_d, O_d)$;
2) *Coordinators*: $C = \{C_1, C_2, \cdots, C_n\}$ is a set of coordinators, n$\geq$ 1,

   $C_i = (N_c^i, \Sigma_c^i, \Delta_c^i, \sigma_c^i, \mu_c^i, F_c^i)$ is a PNT with a Petri net $N_c^i = (P_c^i, T_c^i, I_c^i, O_c^i), i = 1, \cdots, n$;
3) $F = \{f_1, \cdots, f_s\}$ is a set of *connection points*, $s \geq 1$;
4) R_D and S_D are mappings from T_d to finite subsets of F, i.e.,

   $$R_D : T_d \to 2^F, \quad S_D : T_d \to 2^F$$

   R_D and S_D are called the *dispatcher receiving* and *sending mappings*, respectively. The R_D indicates how information can be received by the dispatcher from the connection points. The S_D indicates how information can be sent from the dispatcher to the connection points.
5) $R_C = \{R_C^1, \cdots, R_C^n\}$ and $S_C = \{S_C^1, \cdots, S_C^n\}$. R_C^i and S_C^i are mappings from T_C^i to finite subsets of F, i.e.,

   $$R_C^i : T_C^i \to 2^F, \quad S_C^i : T_C^i \to 2^F$$

   R_C and S_C are called the *coordinator receiving* and *sending mappings*, respectively. The meanings of R_C^i and S_C^i are similar to that of R_D and S_D for $i = 1, \cdots, n$.

The configuration of the coordination structures is shown in Fig. 4. A coordination structure CS with an arbitrary connection is called a *general coordination structure.*

In Definition 5, no restriction has been imposed on the receiving and sending mappings; therefore, arbitrary connections between the dispatcher and coordinators may be assigned. Although this gives the coordination structures the greatest power for connection description, it makes any kind of analysis for system properties extremely difficult. Another fact is that the role of the dispatcher is no different than that of the coordinators in the configuration of connections. They all have equal positions. The only way in which the dispatcher distinguishes itself from the coordinators is through the process of language translation, i.e., only the dispatcher can receive input task strings from the organizer in the organization level and the coordinators get their input task strings from the dispatcher. An application of the general coordination structures has been presented by Wang and Saridis [27].

We now consider two important special connection patterns: the ring connection and the tree (or star) connection.

Definition 6: A coordination structure CS with a *ring* connection is a general coordination structure satisfying the conditions:

1) $F = F_0 \cup F_1 \cup \cdots \cup F_n, \quad F_i \cap F_j = \emptyset, i \neq j;$
2) $R_D : T_d \to 2^{F_n}, \quad S_D : T_d \to 2^{F_0};$
3) $R_C^i : T_C^i \to 2^{F_{i-1}}, \quad S_C^i : T_C^i \to 2^{F_i}, i = 1, \cdots, n;$

where F_0 is called the *output connection points* of the dispatcher D; F_n is called the *input connection points* of the dispatcher D; F_{i-1} is the *input connection points* of coordinator C_i; and, F_i the *output connection points* of coordinator C_i.

Definition 7: A coordination structure CS with a *tree* connection is a general coordination structure satisfying the conditions:

1) $F = F_1 \cup \cdots \cup F_n, \quad F_i \cap F_j = \emptyset, i \neq j;$
2) $R_D : T_d \to 2^F, \quad S_D : T_d \to 2^F;$
3) $R_C^i : T_C^i \to 2^{F_i}, \quad S_C^i : T_C^i \to 2^{F_i}, i = 1, \cdots, n;$

where F_i is the connection points of coordinator C_i.

The following simple examples provide illustrations for the coordination structures with ring connection and tree connection, respectively.

Example 3: Fig. 5 presents an example of the coordination structures with ring connection, where an arc with arrow from a transition to a connection point indicates a sending relationship between the transition and the point, while an arc with arrow from a connection point to a transition indicates a receiving relationship. The formal specification for connection is,

$$\begin{aligned} F_i &= \{f_{i1}, f_{i2}\}, & i &= 0, 1, 2; \\ R_D(t_{d2}) &= F_2, & S_D(t_{d1}) &= F_0; \\ R_C^1(t_{c1k}) &= \{f_{0k}\}, & S_C^1(t_{c1k}) &= \{f_{1k}\}, & k &= 1, 2; \\ R_C^2(t_{c2k}) &= \{f_{1k}\}, & S_C^2(t_{c2k}) &= \{f_{2k}\}, & k &= 1, 2. \end{aligned}$$

Consider this coordination structure as a Petri net: it is easy to see that, in order to take the dispatcher from the initial marking back to the initial marking (a *complete cycle* of execution), the transitions in the coordination structure have to be fired according to one of the following firing sequences:

$$t_{d1} \cdot t_{c11} \cdot t_{c12} \cdot t_{c21} \cdot t_{c22} \cdot t_{d2},$$
$$t_{d1} \cdot t_{c11} \cdot t_{c21} \cdot t_{c12} \cdot t_{c22} \cdot t_{d2}$$

Therefore, the coordination structure defines the *task precedence* relationship (the execution order) among the tasks in the dispatcher and coordinators.

In a coordination structure with ring connection, a system unit receives information from only one of the remaining units, and transfers the information to only one of the other units. The coordinators can communicate among themselves in a sequential fashion. The dispatcher and coordinators still

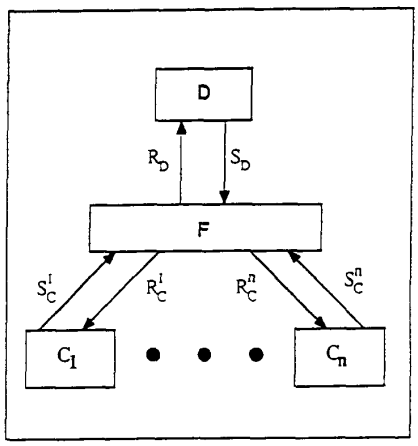


Fig. 5. A coordination structure with ring connection.

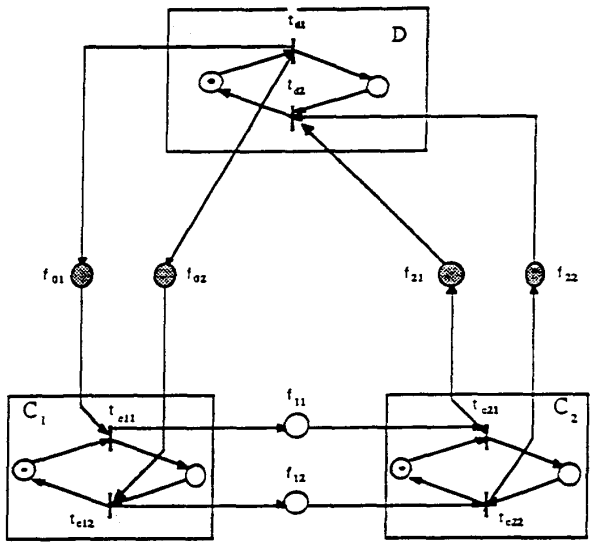


Fig. 6. A coordination structure with tree connection.

hold equal position in the connection configuration as in the case of the general coordination structures. As described in the begining of Section II, however, only the dispatcher can receive input tasks from the higher level, the task process will then start from and end at the dispatcher.

Example 4: Fig. 6 gives a coordination structure with tree connection. The formal specification for the connection is

$$F_i = \{f_i\}, \quad i = 1, 2; \qquad S_D(t_{d1}) = R_D(t_{d2}) = F;$$

$$R_C^1(t_{c11}) = S_C^1(t_{c12}) = F_1; \quad R_C^1(t_{c21}) = S_C^1(t_{c22}) = F_2.$$

Again, consider the coordination structure as a Petri net. By listing all possible firing sequences, we can find that, in one complete cycle of execution, the transitions in the coordination structure have to be fired in according to one of the following firing sequences:

$$t_{d1} \cdot t_{c11} \cdot t_{c12} \cdot t_{c21} \cdot t_{c22} \cdot t_{d2},$$
$$t_{d1} \cdot t_{c11} \cdot t_{c21} \cdot t_{c12} \cdot t_{c22} \cdot t_{d2},$$
$$t_{d1} \cdot t_{c11} \cdot t_{c21} \cdot t_{c22} \cdot t_{c12} \cdot t_{d2},$$
$$t_{d1} \cdot t_{c21} \cdot t_{c22} \cdot t_{c11} \cdot t_{c12} \cdot t_{d2},$$
$$t_{d1} \cdot t_{c21} \cdot t_{c11} \cdot t_{c12} \cdot t_{c22} \cdot t_{d2},$$
$$t_{d1} \cdot t_{c21} \cdot t_{c11} \cdot t_{c22} \cdot t_{c12} \cdot t_{d2}.$$

In a coordination structure with tree connection, the dispatcher occupies a dominant position in the connection configuration. No direct communication among the coordinators is allowed, and the coordinators have to exchange information with each other through the dispatcher. Considering the role played by the dispatcher in the task translation, we see that the dispatcher serves as both a task control center and an information communication center. When direct connctions are permitted only between the dispatcher and a coordinator, the coordination structures with the tree connection are the appropriate model for the coordination level.

The coordination structures with tree connection defined by 4.3 are still too complex to allow us to perform system analysis easily for the coordination level. To compensate the analysis aspect, we impose some further restrictions on the coordination structures with tree connection and introduce the model of *simple coordination structures.*

Definition 8: A simple coordination structure CS with a tree connection (simple coordination structure, for short) is a coordination structure with a tree connection satisfying the conditions:

1) $F_i = \{f_I^i, f_{SI}^i, f_O^i, f_{SO}^i\}$: f_I^i is called the input point of f_{SI}^i the input semaphore of f_O^i the output point, and f_{SO}^i the output semaphore of C_i, respectively;
2) R_D and S_D satisfy the following connection constraints:
   a) $(t, f_I^i) \in S_D \Leftrightarrow (t, f_{SI}^i) \in R_D, (t, f_O^i) \in R_D \Leftrightarrow (t, f_{SO}^i) \in S_D$;
   b) $(t, f_I^i) \notin R_D, (t, f_{SO}^i) \notin R_D, (t, f_O^i) \notin S_D, (t, f_{SI}^i) \notin S_D$;
   c) Each C_i has an integer $n_i \geq 1$, called the *task buffer capacity*, such that $\forall$ firing sequence s of transitions in N_d:

$$0 \leq \#\{t | t \text{ in } s \& (t, f_I^i) \in S_D\} - \#\{t | t \text{ in } s \& (t, f_O^i) \in R_D\} \leq n_i + 1.$$

3) R_C^i and S_C^i satisfy the following connection constraints:
   a) $(t, f_O^i) \in S_C^i \Leftrightarrow (t, f_{SO}^i) \in R_C^i \& (t, f_{SI}^i) \in S_C^i$;
   b) $(t, f_I^i) \notin S_C^i, (t, f_{SO}^i) \notin S_C^i, (t, f_O^i) \notin R_C^i, (t, f_{SI}^i) \notin R_C^i$;
   c) only one initially enabled transition t_s^i in C_i with $(t_s^i, f_I^i) \in R_C^i$;
   d) only one transition t_f^i in C_i with $(t_f^i, f_O^i) \in S_C^i$;
   e) only transition t_f^i has its output places being the input places of t_s^i.

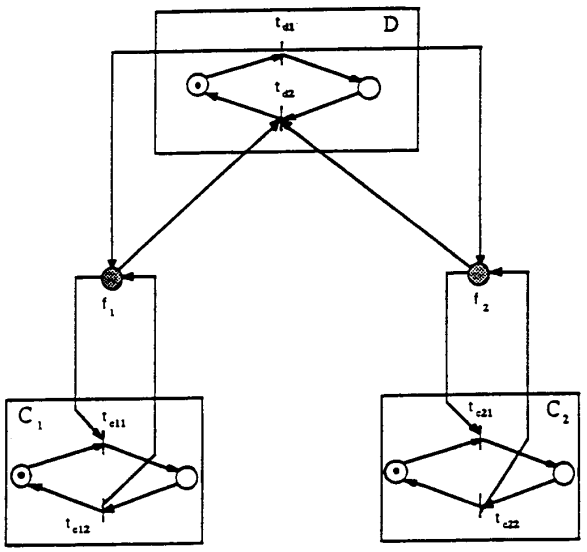


Fig. 7. A simple coordination structure.

The conditions in 1)–3) of the simple coordination structures represent a set of formal specifications for a simple control protocol, i.e., before the dispatcher sends a new task to a coordinator, the dispatcher must check whether the coordinator is capable of receiving more tasks (through the number of tokens left in the input semaphore), whereas before a coordinator reports the result of a task execution back to the dispatcher, the coordinator must check whether its output buffer is capable of holding more task reports (through the number of tokens left in the output semaphore). Physically, this protocol is designed based upon the assumption that only limited memory space and communication network access are available to a coordinator. Specifically, the connection constraints on R_D and S_D indicate that: a) the dispatcher must check the input semaphore f_{SI}^i before it sends information to C_i, and reset the output semaphore f_{SO}^i after information has been received from C_i; b) the dispatcher cannot receive information from the coordinators through f_I^i or f_{SO}^i, or send information to the coordinators through f_O^i or f_{SI}^i; c) before a transition receives the execution result from a coordinator, there must be other transitions which activate the coordinator a sufficient number of times (but bounded by the buffer capacity n_i); d) every coordinator is connected with the dispatcher bidirectionally. The connection constraints on S_C^i and R_C^i imply that: a) C_i must check the output semaphore f_{SO}^i before it sends information to others, and reset the input semaphore f_{SI}^i after information has been sent; b) C_i cannot receive information through f_O^i or f_{SI}^i, or send information through f_I^i or f_{SO}^i; c) only one initially enabled transition, t_s^i, in C_i can receive tasks from the dispatcher; d) only one transition, t_f^i, in C_i can send information to the dispatcher; e) C_i can receive a new task only after it reports the result of the previous task execution.

Fig. 7 presents a simple coordination structure. In the following sections, we will concentrate on the analysis of simple coordination structures.

V. From Coordination Structures to Petri Nets

From the examples given in the previous section, we have already seen that Petri nets can be naturally obtained from the coordination structures by considering the connection points as places, and the receiving and sending mappings as the input and output functions, respectively. We formally define the Petri nets which underlie the simple coordination structures and show how they can be used to describe the operations of the coordination structures. Similar procedures can be applied to derive the Petri nets which underlie the other types of coordination structures. It should be pointed out that once these Petri nets have been obtained, then all the concepts, methods, and tools developed in Petri net theory for system analysis and synthesis [14] can be used to address various analysis and synthesis issues of the coordination structures.

Definition 9: The Petri net N underlying a simple coordination structure CS is a Petri net, specified as $N = (P, T, I, O)$, where

$$P = P_d \cup P_C \cup F, \quad T = T_d \cup T_C;$$
$$P_C = \cup_{i=1}^n P_c^i, \quad T_C = \cup_{i=1}^n T_c^i;$$
$$I(t) = \begin{cases} I_d(t) \cup \{f \mid (t,f) \in R_D\} \text{ if } t \in T_d \\ I_c^i(t) \cup \{f \mid (t,f) \in R_C\} \text{ if } t \in T_c^i \end{cases},$$
$$O(t) = \begin{cases} O_d(t) \cup \{f \mid (t,f) \in S_D\} \text{ if } t \in T_d \\ O_c^i(t) \cup \{f \mid (t,f) \in S_C\} \text{ if } t \in T_c^i \end{cases}.$$

The initial marking of N is

$$\mu(p) = \begin{cases} \mu_d(p) \text{ or } \mu_c^i(p) & \text{if } p \in P_d \text{ or } p \in P_c^i \\ n_i & \text{if } p = f_{SI}^i \text{ or } f_{SO}^i \\ 0 & \text{otherwise} \end{cases}.$$

In terms of the Petri net N, a token in the input point of a coordinator indicates that a task has been issued to the coordinator. The dispatcher can send a task command to a coordinator when there is a token in the input semaphore of the coordinator indicating the coordinator is available for task execution. A token in the output point of a coordinator indicates that a task has been completed by that coordinator, and the feedback information of the task execution has been contained in that token. A coordinator can send feedback of task execution to its output point when there is a token in its output semaphore, which implies that the communication facility is ready for information transferring between the dispatcher and coordinator. Once a transition in the dispatcher takes the feedback information from the output point, it will reset the output semaphore of the coordinator. The overall operation of a coordination structure can be described below.

To start operation, a CS receives a task plan from the organizer, puts it on the input tape of the dispatcher D, and the D begins the process of dispatching. Once a transition t of D, with $f_I^{i_1}, \cdots, f_I^{i_s}$ as its output places in F, has been fired with respect to the current marking of N to execute a primitive task a, it will send the selected control string $z \in \sigma_d(t, a)$ to the coordinators $C_{i_1}, \cdots, C_{i_s}$, and activate the synchronous composition

$$C_{i_1} \| \bullet \bullet \bullet \| C_{i_s}$$

for the processing of z. Upon to the completion of a task by a C_{i_k}, transition $t_f^{i_k}$ will put a token (feedback) to $f_O^{i_k}$, if it was enabled with respect to the current marking of N. The feedback will be read by the dispatcher to continue the task process, and C_{i_k} will become idle again. Once the dispatcher reaches its final markings and the coordinators are either in the initial marking (i.e., no task processing for a while) or the final marking, the entire task process is completed, and the requested job is accomplished successfully.

Clearly, the synchronous composition provides the dispatcher a mechanism of synchronizing the task execution of the coordinators, and the Petri net N specifies the precedence relation of the activities in the dispatcher and coordinators, and therefore, defines the *information structure* of the CS. From the point of view of the execution of N, a task plan issued by the organization level can be considered as a path specification in the N_d, and, in turn, the control actions selected by the transitions of D can be thought of as the path specifications in the Petri nets of the coordinators. This fact demonstrates again that a PNT provides a mechanism to control the executions of Petri nets.

VI. Some Requirements for Task Processing

In the model of coordinate structures, the input alphabet $\sum_o$ of the dispatcher represents the set of primitive events in the organization level [22]; the input alphabets, $\sum_C^i$ of the coordinators, $i = 1, \cdots, n$, are the set of primitive control actions in the coordination level; and the output alphabets Δ_C^i of the coordinators, $i = 1, \cdots, n$, give the set of primitive operations in the execution level. To ensure the continuity of the task translating process, the following relationship must exist between the output alphabet of the dispatcher and the input alphabet of the coordinators,

$$\Delta_0 = \Sigma_C \equiv \bigcup_{i=1}^{n} \Sigma_c^i.$$

The behavior of the dispatcher and coordinators are specified by the transition sequence sets $L(N_d, \mu_d)$ and $L(N_C^i, \mu_C^i), i = 1, \cdots, n$, respectively. As in the theory of program verification, where the behavior (i.e., all possible routine sequences) of a program is used to prove the correctness of the program and to guide the implementation, the transition sequence sets $L(N_d, \mu_d)$ and $L(N_C^i, \mu_C^i)$ can also be employed for the design , analysis, implementation and simulation of the models for the dispatcher and coordinators.

A coordinator must have the capability to process (or translate) all the possible tasks issued by the dispatcher. In terms of input language of the coordinator, these task processing capability requirements can be specified formally as,

$$\alpha(C_i) \supseteq \bigcup \{\sigma_d(t, a) \uparrow_{(T_d^i)^*} | \ t \in T_d^i \text{ and } a \in \Sigma_O\}, i = 1, \cdots, n, \text{ where}$$
$$T_d^i = \{t \mid t \in T_d \text{ and } (t, f_I^i) \in S_D\}$$

By the closure property of Petri net languages under the union operation and Theorem 2, this requirement is guaranteed to be satisfiable.

A task plan $s \in \Delta_0^*$ is said to be *executable* by a coordination structure CS if the following final configurations can be reached by the dispatcher and coordinators:

$$(\mu_d, s, \lambda) \Rightarrow^* (m, \lambda, y) \quad \text{for some } m \in F_d, \text{ and}$$
$$(\mu_c^i, \lambda, y_c^i), \text{ or } (m_c^i, \lambda, y_c^i), \quad \text{for some } m_c^i \in F_c^i, i = 1, \cdots, n$$

It should be pointed out that the λ-moves (the firings of transitions caused by $\sigma(t, \lambda)$) have a special interpretation. They may represent the *internal operations* occurring in the dispatcher or the coordinators which are activated to provide the necessary information or resource for the continuity of the coordination process.

VII. Process Properties of the Coordination Structures

One of the merits offered by modeling the coordination level with the coordination structures is that the underlying Petri nets can provide us a way to use Petri net concepts and analysis methods to investigate the properties of the processes in the coordination level, such as *liveness, boundedness, reversibility, consistency, repetitiveness*, etc. In this section, we present three theorems on the boundedness, liveness, and reversibility of the coordination structures.

Theorem 3: The Petri net N underlying a simple coordination structure CS is bounded if all the Petri nets $N_d, N_c^i, i = 1, \cdots, n$ are bounded.

It is also easy to show that when $n_i = 1$, the safeness of $N_d, N_c^i, i = 1, \cdots, n$, will guarantee the safeness of N. The boundedness of the underlying Petri net guarantees the structural stability of the coordination structure CS. It can be shown, however, that for a PNT M with a bounded Petri net, we can define an equivalent finite state transducer M' using the reachability set of the bounded net such that $\tau(M') = \tau(M)$. This implies that the *input and output languages of the bounded PNT are regular languages*. This fact indicates that the language complexity of PNTs with the bounded Petri nets is very simple. Therefore, for some cases, the unbounded PNTs in the coordination structures are required if the express power of regular languages is inadequate.

Theorem 4: The Petri net N underlying a simple coordination structure CS is live if all the Petri nets $N_d, N_c^i, i = 1, \cdots, n$ are live.

It is clear from the proof that a transition in T_d can be enabled through the same number of firings of transitions in both Petri nets N and N_d from the same marking. However, the firing sequences in N and N_d may be different. Especially, two transitions in T_d that are parallel in N_d may no longer be parallel in N, since they may require inputs from the same coordinator with the task buffer capacity equals one.

The liveness of N ensures the absence of deadlock in the coordination structures. The following example shows that the connection constraint c) 2) is necessary to guarantee the liveness of N.

Example 5: Consider the coordination structure CS shown in Fig. 8. Obviously, the Petri nets for the dispatcher and coordinator are live nets. The possible firing sequences of

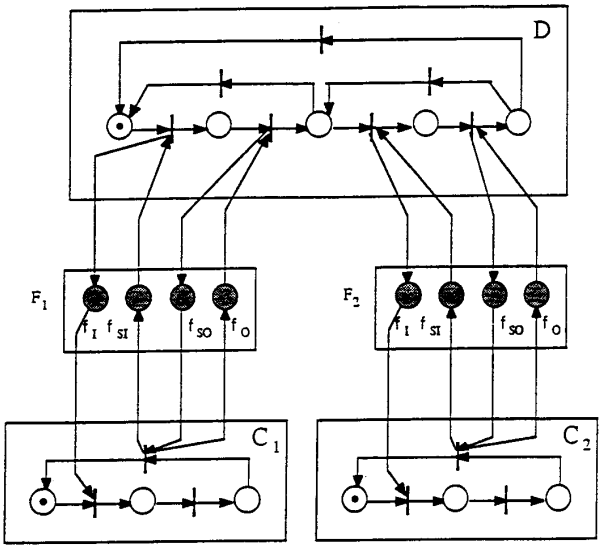


Fig. 8. A coordination structure with deadlocks.

transitions in N_d is,

$$(t_{d1}t_{d2}t_{d3})^*t_{d1}, (t_{d1}t_{d2}t_{d3})^*t_{d1}t_{d2}, (t_{d1}t_{d2}t_{d3})^*t_{d1}t_{d2}t_{d3}$$

Clearly for any finite number K, we have

$$\#\{t' \mid t' \text{ in } (t_{d1}t_{d2}t_{d3})^M \& \ (t', f_I^i) \in S_D\} - \#\{t'' \mid t'' \text{ in } (t_{d1}t_{d2}t_{d3})^M \& (t'', f_O^i) \in R_D\} > K$$

$\forall M \geq \left[\frac{K}{2}\right] + 1$. Therefore, 2c) is violated here when the task buffer capacity $n_1 = 1$. It is easy to check that after the firing sequence

$$t_{d1} \quad t_s \quad t_f \quad t_{d2} \quad t_{d3} \quad t_s \quad t_f \quad t_{d1} \quad t_s$$

the net N will be in deadlock. Thus, the underlying Petri net N is not a live net. Actually, for any finite n_1, we can find an firing sequence which leads N to a deadlock state.

Finally, the following theorem gives the reversibility of the underlying Petri net N with respect to its component nets.

Theorem 5: If all the Petri nets $N_d, N_c^i, i = 1, \cdots, n$, are individually reversible, then N_d, and $N_c^i, i = 1, \cdots, n$, as the subnets of the Petri net N underlying the simple coordination structure CS, are still reversible.

Therefore, the dispatcher and coordinators in a simple coordination structure are always capable of re-initializing themselves.

Unlike in the previous two theorems, no conclusion has been made about the state of the connection points in this theorem. It is not clear whether the underlying Petri net N is still reversible when the connection points are included. However, this is not an important issue since the theorem already guarantees the reversibility of the states of the dispatcher and coordinators.

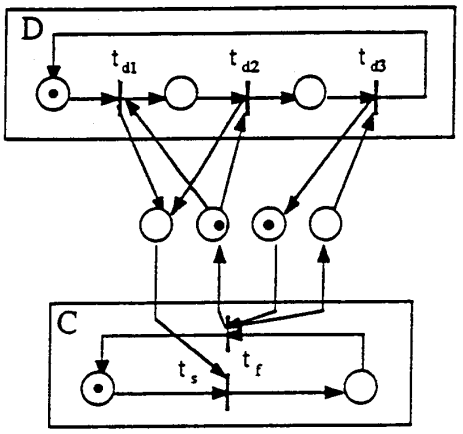


Fig. 9. The layout of the assembly workcell.

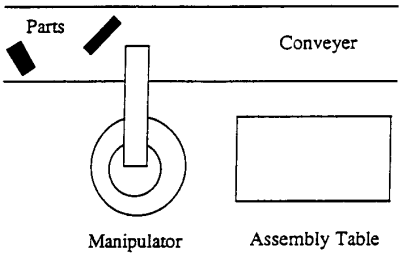


Fig. 10. The Petri net model for the dispatcher.

VIII. Case Study: A Prototype Intelligent Robotic System

To demonstrate the application of the coordination structures for the integration specification in intelligent machines, a case study of modeling an intelligent assembly robotic system (IARS) for assembly tasks has been conducted. The major components of the IARS include: 1) a general-purpose manipulator with a gripper; 2) a vision system with several cameras; 3) a sensor system with touch, crossfire, and force/torque sensors; 4) a communication network for information exchange; 5) a high-level digital computer for control and communication activities. The particular assembly job considered here is to move parts from a conveyer into the slots on an assembly table. Fig. 9 presents the layout of the assembly workcell.

The coordination structure for the coordination level of the IARS consists of a dispatcher (D), a vision coordinator (VC), a sensor coordinator (SC), a path planning coordinator (PC), a motion coordinator (MC), and a gripper coordinator (GC). Individual PNTs for the dispatcher and coordinators are presented in the sequel.

A. The Dispatcher (D)

The input alphabet is $\Sigma_o = \{e_1, e_2, e_3, e_4\}$, where e_1, e_2, e_3 are task primitives involved with the vision, motion, and sensor coordinators, respectively; e_4 is a task primitive of grasping or releasing parts. The task plans from the organization level have been assumed to be generated by the grammar

$$G = (N, \Sigma_o, P, S)$$

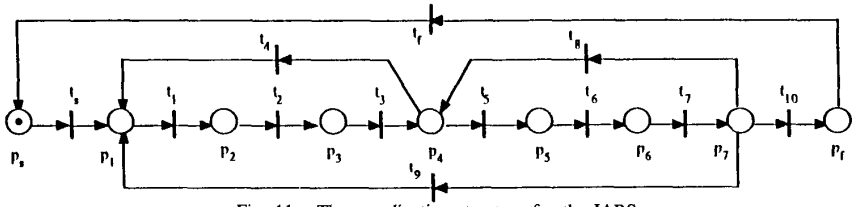


Fig. 11. The coordination structure for the IARS.

where $N = \{S, M, Q, H\}$ is the set of non-terminal symbols, $\Sigma_o = \{e_1, e_2, e_3, e_4\}$ the set of terminal symbols, and $P = \{S \rightarrow e_1M,\ M \rightarrow e_2S|e_2Q,\ Q \rightarrow e_3H,\ H \rightarrow e_4Q|e_4S|e_4\}$ the set of production rules.

The output alphabet is $\Delta_o = \Sigma_v \cup \Sigma_s \cup \Sigma_p \cup \Sigma_a \cup \Sigma_g$, where Σ_v, Σ_s, Σ_p, Σ_a, and Σ_g are, respectively, the input alphabets for the VC, SC, PC, MC, and GC:

$$\Sigma_v = \{contr_v,\ proc_v,\ analy_v,\ send_{\rm vp},\ send_{\rm pvm},\ finish\}$$
$$\Sigma_s = \{contr_s,\ proc_s,\ analy_s,\ send_{\rm sg},\ move\_send_{\rm gs},\ finish\}$$
$$\Sigma_p = \{path, send_{\rm vp}, send_{\rm pvm}, finish\}$$
$$\Sigma_a = \{move, send_{\rm pvm}, finish\} \text{ and}$$
$$\Sigma_g = \{send_{\rm sg}, move\_send_{\rm gs}, finish\}.$$

The meanings of these control instructions are given in the descriptions for the corresponding coordinators.

Fig. 10 gives the Petri net model for the dispatcher. The translation mapping σ_d for the D is specified as:

$$\sigma_d(t_1, e_1) = \{\lambda, contr_v.proc_v.analy_v.finish\}$$
$$\sigma_d(t_2, e_2) = \{(contr_v.proc_v.analy_v.send_{\rm vp}.path\ send_{\rm pvm}.move)^+.finish\}$$
$$\sigma_d(t_5, e_3) = \{\lambda, contr_s.proc_s.analy_s.finish\},$$
$$\sigma_d(t_6, e_4) = \{(contr_s.proc_s.analy_s.send_{\rm sg}\ .move\_send_{\rm gs})^+.finish\}$$

where $s^+ = \{s^n, n \geq 1\}$. All other transitions of the dispatcher are internal operations.

The input language of the dispatcher is exactly the task plans issued by the organizer (i.e., grammar G). The empty string λ in the translation mapping σ_d means no action since the required visual or sensory information is already available. The interactive motion control strings, $(contr_s.\ proc_s.\ analy_s.\ send_{\rm sg}.\ move\_send_{\rm gs})^+.finish$,or $(contr_v.\ proc_v.\ analy_v.send_{\rm vp}.path.send_{\rm pvm}.move)^+finish$, of the dispatcher involve the synchronous composition of two or three coordinators, i.e., SC||GC or VC||PC||MC. When these control strings are issued for the motion or grasping tasks, the corresponding coordinators have to work cooperatively to achieve the required task. The additional information scripts or data files associated with control instructions will not be discussed here.

B. The Vision Coordinator (VC)

The subtask plans to be processed by the VC are $\{\lambda, contr_v.\ proc_v.\ analy_v.finish,\ (contr_v.\ proc_v.\ analy_v.\ send_{\rm vp}.\ send_{\rm pvm})^+.finish\}$.

The output alphabet Δ_v of the VC consists of the hardware operations for the cameras. Since we are not going to be involved with the execution level, we do not specify Δ_v in detail and only describe the function of the translation mapping σ_v for the VC. The Petri net model for the VC is given in Fig. 11. The translation mapping σ_v is specified as:

$$\sigma_v(t_1, contr_v) = \{\text{instructions to control the camera devices and to take pictures}\},$$
$$\sigma_v(t_2, proc_v) = \{\text{algorithms for image processing}\},$$
$$\sigma_v(t_3, analy_v) = \{\text{algorithms for image analysis and fusion}\},$$
$$\sigma_v(t_5, send_{\rm vp}) = \{\text{procedures for sending the information to the PC}\},$$
$$\sigma_v(t_7, finish) = \sigma_d(t_8, finish) = \{\text{procedures for formulating the feedback information}\},$$
$$\sigma_v(t_{\rm com}, send_{\rm pvm}) = \{\text{procedures for receiving the information from the PC}\}.$$

C. The Sensor Coordinator (SC)

The subtask plans to be processed by the SC are $\{\lambda, contr_s.\ proc_s.analy_s.finish,\ (contr_s.proc_s.analy_s.send_{\rm sp}.\ move\_send_{\rm gs})^+.\ finish\}$.

The Petri net model for the SC is the same as that of the VC. The places and transitions also have the similar meaning. The output alphabet Δ_s of the SC consists of the hardware operations for the sensor devices. The translation mapping σ_s is similar to the mapping σ_v.

D. The Path Planning Coordinator (PC)

The subtask plans to be processed by the PC are $\{(send_{\rm vp}\ .path.send_{\rm pvm})^+.finish\}$. The PC is based on the obstacle avoidance path planner developed in [5]. Fig. 11 gives the Petri net model for this coordinator.

The translation mapping σ_p for the path planning coordinator can be specified as:

$$\sigma_p(t_s, send_{\rm vp}) = \{\text{procedures for receiving the information from the VC}\}$$
$$\sigma_p(t_1, path) = \{\text{procedures for searching the path from memory}\}$$
$$\sigma_p(t_2, path) = \{\text{algorithms for constructing the geometrical path}\}$$

$$\sigma_p(t_5, send_{\text{pvm}}) = \{\text{procedures for sending the information to the VC and MC}\}$$
$$\sigma_p(t_7, send_{\text{vp}}) = \{\text{procedures for receiving the information from the VC}\}$$
$$\sigma_p(t_7, finish) = \{\text{procedures for formulating the feedback information}\}.$$

E. The Motion Coordinator (MC)

The subtask plans to be processed by the MC are $\{(send_{\text{pvm}}.move)^+.finish\}$. Fig. 11 gives the Petri net model for the coordinator.

The translation mapping σ_m for the motion coordinator can be specified as:

$$\sigma_m(t_1, move) = \{\text{procedures for evaluating the trajectory}\},$$
$$\sigma_m(t_5, finish) = \{\text{procedures for formulating the feedback information}\}.$$
$$\sigma_m(t_{\text{com}}, send_{\text{pvm}}) = \{\text{procedures for receiving the information from the PC}\}.$$

F. The Gripper Coordinator (GC)

The subtask plans to be processed by this coordinator are $\{(send_{\text{sg}}.move\_send_{\text{gs}})^+.finish\}$.

The Petri net model for the coordinator is same as that of the MC. The places and transitions also have the similar meaning, except that the transition t_1 here has to determine the fine path to approach the desired object. The translation mapping σ_g for the GC can be specified as

$$\sigma_g(t_1, move\_send_{\text{gs}}) = \{\text{procedures for determining the fine motion for hand and sending information to the SC}\}.$$
$$\sigma_g(t_5, finish) = \{\text{procedures for formulating feedback information}\}.$$
$$\sigma_g(t_{\text{com}}, send_{\text{sg}}) = \{\text{procedure for receiving information from the SC}\}.$$

All the unspecified transitions in the coordinators are internal operations.

The coordination structure (CS) for the coordination level of the IARS now can be constructed from the individual PNTs by introducing the connection points, and the receiving and sending mappings (see Fig. 11). Note that the coordination structure constructed here is not a simple coordination structure since it allows direct connection between coordinators, and although the individual PNTs are unbounded, the CS obtained by integration, viewed as a Petri net, is bounded.

The CS is operated according to the execution procedures described in Section V. When control $(contr_v.proc_v.analy_v.send_{\text{vp}}.path.send_{\text{pvm}}.move)^+.finish$ is issued by the dispatcher, the actual number of iterations is determined by the PC as it finds that the desired arm motion is completed. In this case, the PC will send a message to the VC and MC to indicate the termination of execution and to fire the transition t_7 for the $finish$ task. Note that this control action guarantees that the PC can always received the new information from the VC for its path planning. Similarly, the number of iterations of the control action $(contr_v.proc_v.analy_v.send_{\text{vp}}.path.send_{\text{pvm}}.move)^+$ $finish$ is determined by the GC as it finds that the required task is accomplished.

IX. Conclusion

An analytical theory of coordination for intelligent machines has been established in this paper by establishing a formal model for the coordination level. It has been demonstrated that this model can enable the establishment of an information structure in the coordination level. The information structure specifies the necessary task precedence relationship in the coordination of the diversified activities.

A new type of transducers, Petri net transducers (PNTs), has been introduced to serve as the basic module in our analytical model. PNTs provide a formal description for the individual processes within the dispatcher and coordinators. The concurrence and conflict among these processes can be represented by PNTs conveniently. Another application of PNTs is to control and to synchronize the operations of Petri nets.

Several coordination structures have been defined as a formal tool for specification of the connection and cooperation between the dispatcher and the coordinators. The relation between the coordination structures and Petri nets is investigated in detail. The results of using the concepts and the analysis methods in Petri net theory to investigate the process properties of the coordination structures present one aspect of their advantages as the model for the coordination level. More important, the coordination structures provide a structural formulation for a mechanism of control and communication for task processes in the dispatcher and the coordinators. The use of colored Petri nets for modeling the coordination structures is a possible direction for future research.

The coordination theory for the coordination level presented here, together with the mathematical formulation for the organization level and the well developed control theory for the execution level, completes the first step toward a mathematical theory for intelligent machines. Such a mathematical theory will provide a solid scientific and engineering foundation for the design, simulation, verification, and implementation of intelligent machines such as the machines in the Intelligent Robotic Systems for space exploration or in the Computer Integrated Manufacturing Systems for future factories.

Appendix
Proofs of the Theorems

Proof of Theorem 1: Let N be the Petri net of $M = (N, \Sigma, \Delta, \sigma, \mu, F)$, $N = (P, T, I, O)$. For any $t \in T$, if there exist two different a_1 and $a_2 \in \Sigma \cup \{\lambda\}$ such that both $\sigma(t, a_1)$ and $\sigma(t, a_2)$ are defined, we introduce a new transition t' with $I(t') = I(t)$, $O(t') = O(t)$ and modify σ in such a way that $\sigma(t', a_2) = \sigma(t, a_2)$, $\sigma(t', a)$ is undefined for all other

$a \in \Sigma \cup \{\lambda\}$, and $\sigma(t, a_2)$ becomes undefined. Similarly, if $|\sigma(t, a)| > 1$ for an $a \in \Sigma$, we can introduce new transitions $t's$ such that the transitions t and $t's$ will have a translation mapping with $|\sigma(t, a)| = 1$ or $|\sigma(t', a)| = 1$. Continuing this procedure until no transitions have more than one input characters in Σ for which σ are defined and $|\sigma(t, a)| = 1$ for all $t \in T$ and $a \in \Sigma$, we will finally get a SPNT $M' = (N', \Sigma, \Delta, \sigma', \mu, F)$ with $N' = (P, T', I', O')$. The construction of M' clearly shows that $\tau(M') = \tau(M)$. Q.E.D.

Proof of Theorem 2: For a SPNT $M = (N, \Sigma, \Delta, \sigma, , \mu, F)$, a labeling function β associated with σ can be defined as:

$$\beta : T \to \Sigma \cup \{\lambda\}, \text{ and } \beta(t) = a \text{ if } \sigma(t, a) \text{ is defined}$$

and we the labeled Petri net $\gamma = (N, \Sigma, \beta, \mu, F)$ [14] as *labeled Petri net underlying* M.

By Theorem 1, we only need to consider the case for SPNTs. Let $M = (N, \Sigma, \Delta, \sigma, \mu, F)$ be a SPNT and $\gamma = (N, \Sigma, \beta, \mu, F)$ be the labelled Petri net underlying M. The fact that input language is a Petri net language follows immediately from the relation $\alpha(M) = \beta(L(N, \mu)) \equiv L(\gamma)$.

Let $\gamma' = (N, \Sigma, \beta', \mu, F)$ be a labelled Petri net with a free labeling function β', and $L(\gamma')$ be its Petri net language. It is clear that $\omega(M)$ can be derived from $L(\gamma')$ by replacing character $\beta'(t)$ in $L(\gamma')$ with the character $\sigma(t, \beta'(t))$. Since Petri net languages are closed under finite substitution, it follows that $\omega(M)$ is a Petri net language. Q.E.D.

Proof of Theorem 3: The proof is very simple. First of all, by the connection constraints 2a) and 2b) in Definition 8, for any $m \in R(N, \mu)$,

$$m(p) \leq n_i \text{ for some } i, \quad \text{if } p \in F$$

Let $R_d(N, \mu)$ be the set of all markings on the net N_d which are part of the markings of $R(N, \mu)$. In other words, $R_d(N, \mu)$ is the restriction of $R(N, \mu)$ on P_d. Since N_d is a closed subnet of the net N, it follows immediately that $R_d(N, \mu)$ is a subset of $R(N_d, \mu_d)$. Similarly, one can show that the restriction of $R(N, \mu)$ on P_c^i is a subset of $R(N_c^i, \mu_c^i), i = 1, \cdots, n$. Therefore, the boundedness of $N_d, N_c^i, i = 1, \cdots, n$ guarantees the boundedness of the net N. Q.E.D.

Proof of Theorem 4: By the connection constraint 2d) for simple coordination structures, for each C_i there exist transitions in N_d which take f_I^i and f_O^i as their input and output places in F, respectively. From connection constraint 3c), t_s^i takes both f_O^i and f_{SI}^i as its output places. Since only t_f^i has output places being the input places of t_s^i by 3d), it is guaranteed that if a number of tokens is displaced into f_I^i, the same number of tokens will appear in f_O^i when N_c^i is live. Therefore, in order to show N is live, we only need to show that N_d as a subnet of N is a live Petri net.

Let $m \in R(N, \mu)$ be an arbitrary marking; $R(N, m, k)$ be the set of markings reached from m by firing at most k transitions in T_d. Let m_d and $R_d(N, m, k)$ be the restrictions of m and $R(N, m, k)$ on P_d, respectively; and $R(N_d, m_d, k)$ be the set of markings reached from m_d by firing at most k transitions in T_d when N_d is considered as an independent Petri net. Let $T(k)$ be the set of transitions in T_d which are enabled under $R(N, m, k)$ and $T'(k)$ be the set of transitions in T_d which are enabled under $R(N_d, m_d, k)$ when N_d is considered to be independent. We first prove that for any $k \geq 0$,

$$R_d(N, m, k) = R(N_d, m_d, k) \text{ and } T(k) = T'(k)$$

When $k = 0$, $R_d(N, m, 0) = m_d = R(N_d, m_d, 0)$, and, obviously, $T'(k) \supseteq T(k)$. Let $t \in T'(0)$ be an enabled transition with respect to N_d. If $(t, f_I^i) \notin S_D$ and $(t, f_O^i) \notin R_D$ for all i, then it is clear that t is also enabled by m in N in this case, so $t \in T(0)$. When $(t, f_I^i) \in S_D$, let s be the firing sequence of transitions from m, k_S^i be the number of transitions t' in s such that $(t', f_I^i) \in S_D, k_R^i$ be the number of transitions t'' in s such that $(t, f_O^i) \in R_D, p_{IS}^i$ be the number of tokens in f_{IS}^i, p_O^i be the number of tokens in f_O^i, and k_f^i be the number of firings by transition t_f^i. Since there are n_i initial tokens in f_{IS}^i and f_{OS}^i, respectively, we have,

$$p_{IS}^i = n_i - k_S^i + k_f^i$$
$$k_R^i \leq k_f^i \leq \min\{k_S^i, n_i + k_R^i\}.$$

However, by the connection constraint 2c) in Definition 8, in this case,

$$0 \leq k_S^i - k_R^i \leq n_i$$

which implies $\min\{k_S^i, n_i + k_R^i\} = k_S^i$. Therefore, t_f^i can be fired enough times such that $p_{IS}^i > 0$, which indicates that t may be enabled under m in N, i.e., $t \in T(0)$. Similarly, when $(t, f_O^i) \in R_D$, we have,

$$p_O^i = k_f^i - k_R^i$$

the constraint 2c) indicates in this case,

$$1 \leq k_S^i - k_R^i \leq n_i + 1$$

which implies $\min\{k_S^i, n_i + k_R^i\} \geq k_R^i + 1$. Therefore, t_f^i can be fired enough times such that $p_O^i > 0$, hence $t \in T(0)$. In all the cases, $t \in T'(0) \Rightarrow t \in T(0)$, therefore $T'(0) = T(0)$.

Assume that $R_d(N, m, k) = R(N_d, m_d, k), T'(k) = T(k)$, for $k \leq q$. Clearly, $R_d(N, m, q+1) = R(N_d, m_d, q+1)$ follows immediately from $T'(q) = T(q)$. Since $T'(q+1) \supseteq T(q+1)$, by the same procedure used the proof of $T(0) = T'(0)$, we can show that $T(q+1) \supseteq T'(q+1)$. Therefore, $T(q+1) = T'(q+1)$. So $R_d(N, m, k) = R(N_d, m_d, k)$ and $T(k) = T'(k)$ for any $k \geq 0$.

Since N_d is live, it is possible to fire every transition t from m_d in N_d. However, $R_d(N, m, k) = R(N_d, m_d, k), T'(k) = T(k)$ for any k, we see that the same transition t is also possible to be fired from m in N through the same number of firings of transitions. Therefore, N is live. Q.E.D.

Proof of Theorem 5: The proof is based on the analysis in the proof for liveness. First of all, it is easy to see that the reversibility of the Petri nets for the coordinators is still held when considering them as the subnets of N. To show the reversibility of the Petri net N_d for the dispatcher, let us use the following result from Theorem 4, i.e.,

$$\forall m \in R(N, \mu), k \geq 0,$$
$$R_d(N, m, k) = R(N_d, m_d, k)$$

Since N_d is reversible, and $m_d \in R(N_d, \mu_d)$, where μ_d is the restriction of μ on P_d, then, $\mu_d \in R(N_d, m_d)$, i.e., $\mu_d \in R(N_d, m_d, q)$ for some $q \geq 0$. From the above equation it follows that

$$\mu_d \in R_d(N, m, q)$$

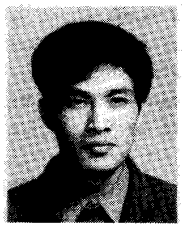
for the same q. This shows that N_d as a subnet of N is reversible. Q.E.D.

ACKNOWLEDGMENT

The authors wish to thank Professors Alan Desrochers and Robert McNaughton for their valuable comments and suggestions.

REFERENCES

[1] A. V. Aho, and J. D. Ullman, *The Theory of Parsing, Translation and Compling*, Vol. 1. Englewood Cliffs, NJ: Prentice-Hall, 1972.
[2] J. S. Albus, "A new approach to manipulation control: The cerebellar model articulation controller," *Trans. ASME. J. Dynamic Syst., Meas. Control,* pp. 220–227, 1975.
[3] R. Y. Al-Jaar and A. A. Desrochers, "Petri nets in automation and manufacturing," in *Advances in Automation and Robotics,* G. N. Saridis, Ed., vol. 2. Greenwich, CT: JAI, 1990.
[4] A. Bejczy, "Task driven control," in *Proc. IEEE Workshop Intelligent Contr.* Rensselaer Polytechnic Institute, Troy, NY, 1985, p. 38.
[5] C. H. Chung, and G. N. Saridis, "Obstacle avoidance path planning by extended Vgraph Algorithm," Tech. Rep. # 12, NASA CIRSSE, Rensselaer Polytechnic Institute, Troy, NY, 1989.
[6] J. J. Demael, and A. H. Levis, "On the generation of a variable structure airport surface traffic control system," in *Proc. 4th IEEE Int. Intelligent Contr. Symp.*, 1989, pp. 74–81.
[7] E. H. Durfee, *Coordination of Distributed Problem Solvers.* Boston, MA: Kluwer Academic, 1988.
[8] K. S. Fu, "Learning control systems and intelligent control systems: An Intersection of artificial intelligence and automatic control," *IEEE Trans. Automat. Contr.*, Vol. AC-16, p. 70, 1971.
[9] J. H. Graham and G. N. Saridis, "Linguistic decision structures for hierarchical systems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-12, pp. 323–333, 1982.
[10] A. W. Holt, "Coordination technology and Petri nets," *Advances in Petri Nets*, G. Rozenberg, Ed. New York: Springer-Verlag, 1984, pp. 278–296.
[11] N. Komoda, K. Kera, and T. Kubo, "An automated decentralized control system for factory automation," *IEEE Comput.*, vol. 17, pp. 73–83, 1984.
[12] A. H. Levis, "Human organizations as distributed intelligence systems," in *Proc. IFAC/IMACS Int. Symp. Distributed Intelligence Systems: Methods and Applications*, Varna, Bulgaria, 1988, pp. 13–19.
[13] A. Meystel, "Nested hierarchical control: Theory of team control applied to autonomous robots," Lab. Appl. Machine Intell. and Robotics, Dept. ECE, Drexel Univ., 1986.
[14] J. L. Peterson, *Petri Net Theory ans The Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall International, 1981.
[15] G. N. Saridis, *Self-Organization Controls of Stochastic Systems.* New York: Marcel Dekker, 1977.
[16] G. N. Saridis and H. E. Stephanou, "A hierarchical approach to the control of a prosthetic arm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 407–420, 1977.
[17] G. N. Saridis, "Toward realization of intelligent control," *Proc. IEEE*, vol. 67, 1979.
[18] ——G. N. Saridis, "Intelligent robotic control," *IEEE Trans. Automat. Contr.*, vol. AC-28, no. 5, pp. 547–557, 1983.
[19] G. N. Saridis and J. H. Graham, "Linguistic decision schemata for intelligent robots," *IFAC J. Automatica*, vol. 20, no. 1, pp. 121–126, 1984.
[20] G. N. Saridis, "Foundations of intelligent controls," in *Proc. IEEE Workshop on Intelligent Contr.*, Rensselaer Polytechnic Institute, Troy, NY, 1985, pp. 23–27.
[21] G. N. Saridis, "Intelligent control," *IEEE Contr. Syst. Mag./*, vol. 7, no. 3, pp. 48–49. 1986.
[22] G. N. Saridis and K. P. Valavanis, "Analytic design of intelligent machines," *IFAC J. Automatica*, vol. 24, no. 2, pp 123–133, 1988.
[23] G. N. Saridis, "Analytic formulation of the principle of increasing precision with decreasing intelligence for intelligent machines," *IFAC J. Automatica*, Vol. 25, no. 3, pp. 461–467, 1989.
[24] K. P. Valavanis, *A Mathematical Formulation for the Analytical Design of Intelligent Machines*, RAL Rep. #85, Rensselaer Polytechnic Institute, Troy, NY, 294 pp., 1986.
[25] F. Y. Wang and G. N. Saridis, "A formal model for coordination of intelligent machines using Petri nets," in *Proc. 3rd IEEE Int. Intelligent Contr. Symp.*, Arlington, VA, 1988.
[26] F. Y. Wang and K. Gildea, "A colored Petri net model for connection management services in mms," *Computer Commun. Rev.*, vol. 19, no. 3., pp. 76–98, 1989.
[27] F. Y. Wang and G. N. Saridis, "The coordination of intelligent robots: A case study," in *Proc. Fourth IEEE Int. Intelligent Contr. Symp.*, Albany, NY, 1989, pp. 506–512.
[28] F. Y. Wang and G. N. Saridis, "A coordination theory for intelligent machines," *IFAC J. Automatica*, vol. 26, no. 9, 1990.
[29] F. Y. Wang, K. Kyriakopoulos, A. Tsolkas, and G. N. Saridis, "A Petri net coordination model for an intelligent mobile robot," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, pp. 777–789, 1991.
[30] F. Y. Wang, M. Mittmann, and G. N. Saridis, "Coordination specification for CIRSSE robotic platform using Petri net transducers," to be appear in *J. Intelligent Robotic Syst.*, 1992.
[31] F. Y. Wang and G. N. Saridis, *Coordination Theory of Intelligent Machines: Aplications in Intelligent Robotic Systems and CIM Systems*, To be published by Kluwer Academic Publishers, Boston, MA, 1993.
[32] F. Y. Wang and K. Gildea, "MMS design and implementation using Petri nets," in *Proc. First Int. Workshop on Formal Methods in Engi. Design, Manuf., and Assembly*, pp. 184–201, 1990.
[33] M. C. Zhou and F. DiCesare, "Adaptive design of Petri Net controllers for error recovery in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 963–973, 1989.
[34] M. C. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of Petri net for manufacturing systems," *IEEE Trans. Robotics Automat.*, vol. 8, no. 3, 1992.
[35] M. C. Zhou and F. DiCesare, "Parallel and sequential mutual exclusions for Petri net modeling for manufacturing systems with shared resources," *IEEE Trans. Robotics Automat.*, vol. 7, pp. 515–527, 1991.
[36] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems.* Boston, MA: Kluwer Academic, 1992.



Fei-Yue Wang (S'89–M'90) was born in Qingdao, China, in November 2, 1961. He received the B.E. degree in chemical engineering 1981 from Shandong Institute of Chemical Technology, Qingdao, China, the M.S. degree in mechanics from Zhejiang University, Hangzhou, China, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, New York, in 1984 and 1990, respectively.

From 1984 to 1986 he was an instructor at the Department of Mechanics, Zhejiang University. In 1986, he was awarded Pao Zao-Kong and Pao Zao-Long Scholarship for Chinese Students for his academic achievements. From 1988 to 1990 he was with the NASA Center for Intelligent Robotic Systems for Space Exploration at Rensselaer Polytechnic Institute. He jointed the University of Arizona, Tucson, Arizona, in 1990, where he is presently an Assistant Professor of the Systems and Industrial Engineering. In his previous work in mechanics and applied mathematics, he contributed significantly to the theoretical development of shells, plates, planes, three-dimensional elasticity, and micropolar elasticity for both isotropic and anisotropic materials. He is the co- author of the forthcoming book *Coordination Theory of Intelligent Machines: Applications in Intelligent Robotic Systems and CIM Systems*, and the translator of the book *Buckling of Elastic Structures* by J. Roorda. He has published more than 40 refereed journal articles and book chapters, and about 50 conference papers and technical reports in the areas of applied mathematics, mechanics, mechatronics, artificial intelligence, control theory, communication, robotics and automation, computer-integrated manufacturing, fuzzy logic, neural networks, and intelligent machines. He is the co-editor of the special issue on fuzzy logic and neural networks for *Journal of Intelligent* and *Fuzzy Systems*. He is the founder of Synergetic Systems, Inc. His fields of interest include mechatronics, robotics and automation, computer vision, computer-integrated manufacturing, intelligent controls and intelligent systems.

Dr. Wang is a member of Sigma Xi, Chinese Society of Mathematics and Mechanics, Association for Computing Machinery (ACM), and American Society of Mechanical Engineers (ASME).

George N. Saridis (M'62–SM'72–F'78) was born in Athens, Greece. He received the Diploma in mechanical and electrical engineering in 1955 from the National Technical University of Athens, Athens, Greece, the MSEE and Ph.D. degrees from Purdue University, West Lafayette, IN, in 1962 and 1965, respectively. In 1988, he was certified as Manufacturing Engineer for Machine Vision by the Society of Manufacturing Engineers. From 1955 to 1963 he was an instructor in the Department of Mechanical and Electrical Engineering of the National Technical University of Athens, Greece. From 1963 until 1981 he was with the School of Electrical Engineering of Purdue University. He was an Instructor until 1965, Assistant Professor until 1970, Associate Professor until 1975 and Professor of Electrical Engineering until 1981. Since September 1981, he has been Professor of the Electrical, Computer and Systems Engineering Department and Director of the Robotics and Automation Laboratories at the Rensselaer Polytechnic Institute, in Troy NY. In 1973 he served as Program Director of System Theory and Applications of the Engineering Division of the National Science Foundation, Washington DC. From 1988 till 1992, he was the Director of the NASA Center for Intelligent Robotic Systems for Space Exploration at Rensselaer Polytechnic Institute. In 1972–1973 he served as the Associate Editor and Chairman of the Technical Committee on Adaptive and Learning Systems and Pattern Recognition of the Control Systems Society of IEEE, Chairman of the 11th Symposium of Adaptive Processes, IEEE delegate to the 1973 and 1976 JACC, and Program chairman of the 1977 JACC. In 1973 and 1979 he was elected member of the ADCOM and in 1986 he was appointed member of the Board of Governors of the IEEE Control Systems Society. In 1979-81 he was appointed chairman of the Education Committee, and in 1986-89 chairman of the Committee on intelligent controls of the same Society. He was the International Program Committee chairman of the 1982 IFAC Symposium on Identification and Parameter System Estimation, in Washington DC, and the 1985 IFAC Symposium on Robotic Control in Barcelona, Spain. In 1974 and 1981 he was appointed Vice-Chairman of the IFAC International Committee on Education and in 1981-84 the Survey Paper Editor of *Automatica*, the IFAC journal. In 1983–1984 he was the Founding President of the IEEE Council of Robotics and Automation, and was elected member of the ADCOM of the IEEE Robotics and Automation Society in 1989 and 1990. He is also chairman of the Awards Committee of the same Society. In 1989 he served as member of the Panel on Intelligent Manufacturing of the National Research Council. In 1988 he was the General Co-chairman of the International Workshop on Intelligent Robots and Systems IROS '88 in Tokyo, Japan, Honorary Chairman of IPC of the 9th IFAC/IFORS Symposium on Identification, Budapest Hungary 1991, and Organizing Committee Chairman of the IROS '92, Raleigh NC, 1992. He is the Editor of the series Annuals on Advances in Robotics and Automation of JAI Publications since 1982. He is also member of the editorial board of IEEE Press in 1988, *Journal of Robotic Systems* since 1984, *Systems Control Encyclopedia* since 1984, *Journal of Intelligent and Robotic Systems* since 1988, and the *Journal of IMPACT* of the Society of Machine Intelligence since 1987. He is the author of the book *Self-Organizing Control of Stochastic Systems*, coauthor of the book *Intelligent Robotic Systems*, editor of the book *Advances in Automation and Robotics*, volumes 1 (1985) and 2 (1990), and co-editor of the books, *Fuzzy and Decision Processes*, *Proceedings of the 6th Symposium of Identification and System Parameter Estimation*, *Proceedings of the 1985 SYROCO* and *Knowledge Based Robotic Control*. He has written more than 350 book chapters, journal articles, conference papers and technical reports. He has also presented more than 100 invited lectures. He is the recipient of the IEEE Centennial Medal Award in 1984, and the IEEE Control Systems Society's Distinguished Member Award in 1989.

Dr. Saridis is a Fellow of IEEE and a member of Sigma Xi, Eta Kappa Nu, the New York Academy of Science, the American Society of Mechanical Engineers, the Society of Photo-Optical Engineers, the American Society of Engineering Education, the American Society for the Advancement of Science, the American Assocition of University Professors and Amnesty International. He is also Senior member of the Robotics International and Charter member of Machine Vision of the Society of Manufacturing Engineers.