CIRSSE - R - 94

NAGW-1333

NASA-CR-192735

GRANT
IN-63-CR
153758
P 25

CIRSSE

Center for Intelligent Robotic Systems for Space Exploration

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

(NASA-CR-192735) TASK SEQUENCE PLANNING IN A ROBOT WORKCELL USING AND/OR NETS (Rensselaer Polytechnic Inst.) 25 p

N93-21391

Unclas

G3/63 0153758

TASK SEQUENCE PLANNING IN A ROBOT WORKCELL USING AND/OR NETS

by

Tiehua Cao and Arthur C. Sanderson

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering
Troy, New York 12180-3590

June 1991

CIRSSE REPORT #94

Published in: Proceedings of the 1991 IEEE Intelligent Control Symposium

Task Sequence Planning in a Robot Workcell Using AND/OR Nets

Tiehua Cao and Arthur C. Sanderson

Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590, USA
June 4, 1991

Abstract

This paper describes an approach to task sequence planning for a generalized robotic manufacturing or material handling workcell. Given the descriptions of the objects in this system and all feasible geometric relationships among these objects, an AND/OR net which describes the relationships of all feasible geometric states and associated feasibility criteria for net transitions is generated. This AND/OR net is mapped into a Petri net which incorporates all feasible sequences of operations. The resulting Petri net is shown to be bounded and have guaranteed properties of liveness, safeness and reversibility. Sequences are found from the reachability tree of the Petri net. Feasibility criteria for net transitions may be used to generate an extended Petri net representation of lower level command sequences. The resulting Petri net representation may be used for on-line scheduling and control of the system of feasible sequences. A simulation example of the sequences is described.

Keywords

Task sequence planning, Petri net, AND/OR network, reachability, coverability, markings.

1 Introduction

An automatic planning system is an essential part of a complete robotic system. A typical planning system is a hierarchical top-down structure. Before the mechanisms and the robot start to work, the host computer in the system obtains the initial states and the final states. The planning system will represent and be able to generate all possible sequences of operations. Then a certain sequence will be chosen based on some evaluation criterion. Each operation which is included in this specific sequence will be considered as a sub-goal at that time. For each sub-goal, the lower level planning work will depend on the feasibility criteria of the AND/OR net. A lower level command sequence corresponding to each high level operation will be generated. Each lower level command sequence will be fired and each command could directly be used to control some mechanisms and the robot. During the execution, new information might be fed back to the computer, and the control commands may be updated to coordinate the whole system.

The whole robotic system as well as its environment is in the initial state before the system starts working. The intelligent robot should have the ability to analyze the information it has from the outside world such as the descriptions of the current or initial states, the knowledge base that represents the set of conditional statements which defines the knowledge in the problem domain, and also the final states the system is expected to reach, and then reason so that it could automatically obtain all possible operation procedures to reach the final states.

Task-level planning[1] provides a high-level sequencing of actions which satisfy domain constraints and ordering relations to accomplish a given task goal. Many domain-independent planning techniques and systems have been developed[2-4]. Most of these planning methods use search techniques developed in the artificial intelligence area. In robot workcells, and particularly assembly, domain-dependent planning methods are often more effective since they represent and reason about domain-related constraints[5-11]. The aim of task level planning is to efficiently generate all possible task sequences and to be able to choose among them. Some sequences may contain fewer operation steps, and each different kind of operation may have a different cost function. Some sequences may also contain parallel operations to save total time needed to fulfill the task. Depending on the task domain and application, we may have alternative cost functions, such as economic cost, minimum time, or maximum reliability, or we may require only that any feasible solution be found.

2 AND/OR Net Representation of the System

The representation of assembly plans in our previous work is based on an AND/OR graph[5,8]. AND/OR graphs have *and-arcs* connecting one initial node to k terminal nodes. We review the basic definition of AND/OR graphs as follows:

Definition 1 *An AND/OR graph* is a pair of sets *(V, H)* in which V is a finite set, and H is a subset of the cartesian product $V \times (\prod(V) - \{\emptyset\})$, where $\prod(V)$ is the set of all subsets of V.

The elements of V are called *nodes*, and the elements of H are called *and-arcs*. For an and-arc (λ, Λ), the node λ is the *initial* node, and the nodes in Λ are the *terminal* nodes. The and-arc (λ, Λ) is *incident from* λ and is *incident to* the nodes in Λ. The and-arc (λ, Λ) is said to connect node λ to the nodes in Λ.

In this section, we introduce the AND/OR net representation as a means to effectively represent geometric relations and constraints among objects and devices in the system. We are given a complete geometric description of objects and object relations, and have extended the AND/OR graph described above to represent more general relationships among objects as system substates. The AND/OR net is defined as follows:

Definition 2 *Pair-match set* Γ*:* Given two finite sets $\Sigma_1 = \{a_1, a_2, \ldots, a_m\}$, $\Sigma_2 = \{b_1, b_2, \ldots, b_n\}$,

$$\Gamma(\Sigma_1, \Sigma_2) = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} \{\{a_i, b_j\}\}$$

Definition 3 *An AND/OR net* is a three-tuple *(S, A, N)* where S is a finite set of states $(s_1, s_2, \ldots, s_t)$, $A \subseteq \Gamma(S, \prod(S) - \{\emptyset\})$, $N \subseteq \Gamma(S, S)$, *and* $A \bigcap N = \emptyset$, where $\prod(S)$ is the set of all subsets of S.

The elements of S are called *nodes*, the elements of A are called *AND-arcs*, and the elements of N are called *IST-arcs*. The AND-arcs $\{\lambda, \psi\}$ is said to connect node λ to the nodes in ψ, $\psi \subseteq$ S. The IST-arc $\{\lambda_1, \lambda_2\}$ is said to connect node λ_1 to the node λ_2. IST represents *Internal State Transition.*

The definition of the *system geometric states diagram* is:

Definition 4 *System geometric states diagram* lists all objects O_i in the system and all possible states O_i^k corresponding to each object. All possible assemblies of n objects, and all possible subassemblies of m objects are special cases of objects, and subassemblies may be derived directly from assemblies.

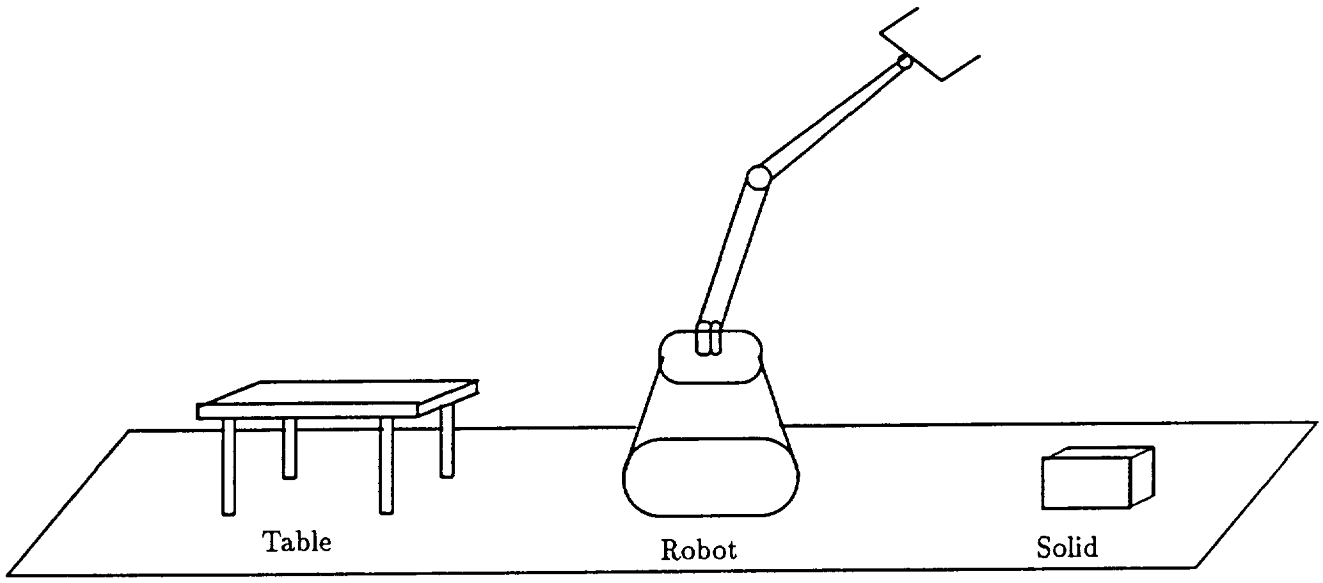
Note that in this discussion, a *subassembly* reflects any geometric configuration of contacting parts, including, for example, a robot holding a part. The system states diagram orders the representation of subassemblies by number of parts n. Each subassembly of order n may be further decomposed into $\prod(n)$ subassemblies. Only those decompositions which are geometrically feasible are included in the AND/OR net. In addition, objects with internal state changes are indicated by links.

The AND/OR net is derived from the system geometric states diagram. The nodes in the AND/OR net correspond to all objects, which may be subassemblies and assemblies, appearing in the geometric states diagram. The AND-arcs represent the feasible decompositions from subassemblies(assemblies) to a corresponding set of subassemblies. The IST-arcs represent the feasible internal state transitions from a subassembly(assembly) to another subassembly(assembly). These two subassemblies(assemblies) are listed in the same column in the geometric states diagram and contain the same number of original objects.

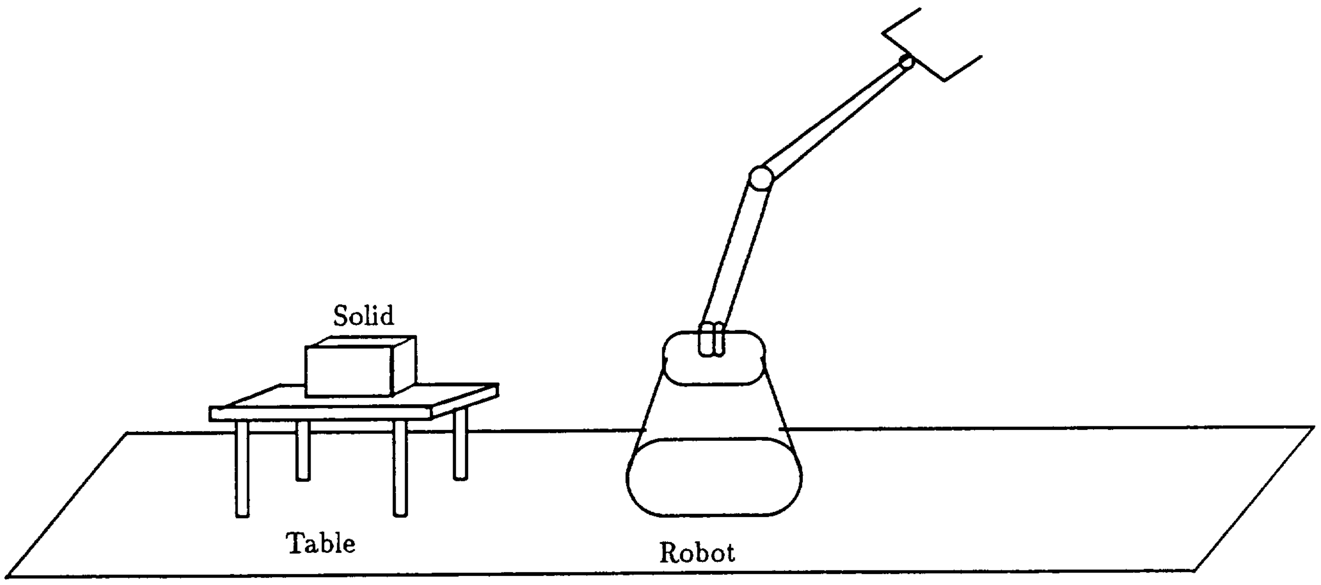
An example of an AND/OR net definition is shown in figures 1 to 3. Figure 1 shows a robot which transfers an object to the surface of a table. The initial states and final states are presented in figure 1. All feasible geometric relationships among these objects are shown in the geometric states diagram in figure 2.

The maximum number of steps of generating the final diagram is $C_n^1 + C_n^2 + \ldots + C_n^n = 2^n - 1$, where n is the number of objects in the system and C_n^j indicates the combinations of n. Within each step, we may obtain more than one configuration of contacting objects. The resulting AND/OR net state diagram is shown in figure 3 and is based on the feasible decompositions of subassemblies of order n to subassemblies of order no more than $n-1$. In the next sections, we will describe the mapping of this AND/OR net to a Petri net.

Methods for extracting all possible sequences from the AND/OR graph or AND/OR tree representation of the system will not work in this case. First, because cycles may appear in an AND/OR net, more difficulties may be met when we use the methods developed in AND/OR graphs for automatically searching task sequences. Secondly, depending on the definition, an AND/OR net possesses the special characteristic of reversibility. Last, we may consider AND/OR graphs or AND/OR trees as subsets of AND/OR nets. Therefore, it is necessary to introduce a searching algorithm for AND/OR nets.



(a)



(b)

Figure 1. Example of a Moving Task for a Robot

(a) Initial states (b) Final states

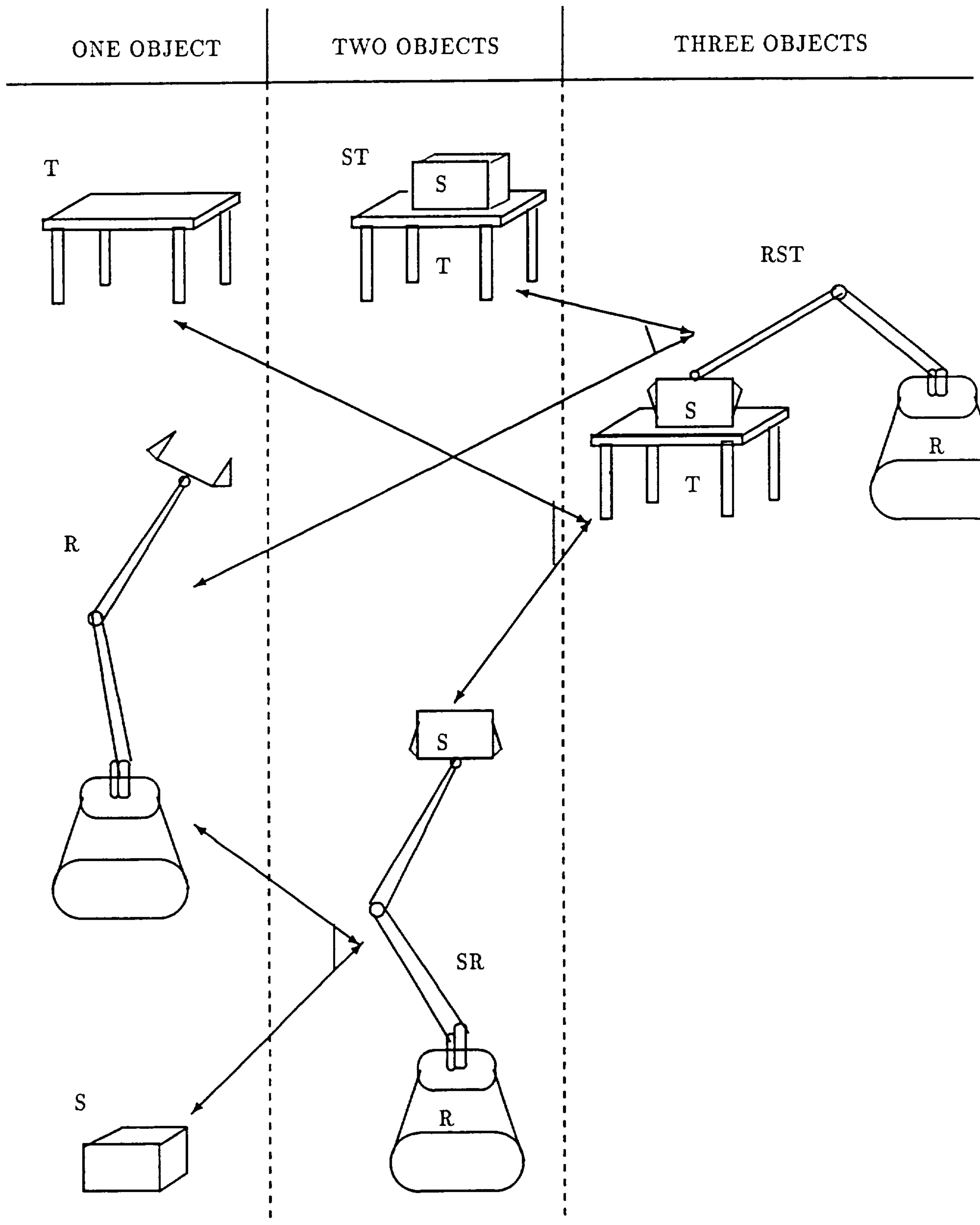


Figure 2. System Geometic States Diagram

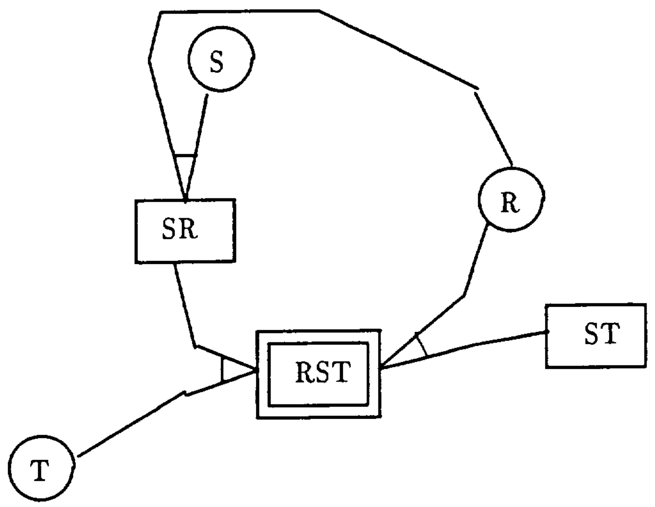


Figure 3. The AND/OR net representation for the example

3 Mapping AND/OR Net to Petri Net

The Petri net has been widely used in modeling manufacturing systems, computer systems, and robot assembly planning systems as well as other kinds of engineering applications[12-19]. This is an efficient abstract and formal information flow model. The Petri net is characterized by its flexibility and efficiency in modeling and analysis of complex discrete-event systems. The definition and some relating terminologies are shown as follows.

Definition 5 *A Petri net structure, N,* is a four-tuple, $N = (P, T, \alpha, \beta)$. $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of *places*, $n \geq 0$. $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of *transitions*, $m \geq 0$; $P \cap T = \emptyset$. $\alpha \subseteq \{P \times T\}$ and $\beta \subseteq \{T \times P\}$ are sets of directed arcs.

Definition 6 *Marking μ:* Marking μ of Petri net N is a mapping from set P to set $\Lambda = \{0, 1, 2, \ldots, L\}$ which is a finite set, i.e.,

$$\mu : P \rightarrow \Lambda,$$

where μ sets tokens to every place in N, $\mu_i = \mu(p_i) \in \Lambda$ indicates the number of tokens in place p_i, μ can be in the form:

$$\mu = (\mu_1, \mu_2, \ldots, \mu_n)^T; \qquad \mu_i = \mu(p_i), \quad p_i \in P.$$

Definition 7 *Marked Petri net M:* A Petri net structure N containing a marking μ is *a marked Petri net* which is the following five-tuple,

$$M = (P, T, \alpha, \beta, \mu)$$

Sometimes, for the sake of simplicity, we refer to a *marked Petri net* as a *Petri net* as shown later in this paper.

Definition 8 *Petri net graph:* The Petri net graph consists of directed arcs and two kinds of nodes. In the graph, circle node and bar node represent place and transition respectively. Directed arc, which links circle node and bar node, indicates the relation between place and transition. Marking μ is shaped by solid dots in a circle node.

One important property of a Petri net is the representation of serial and concurrent events and resource constraints. For this system, we use Generalized Stochastic Petri Nets(GSPN) software[20,21] to represent the system and carry out some simulations as well as verifying the task sequences.

Each node in the AND/OR net becomes a place in the Petri net. All transitions in the system AND/OR network can occur in either direction. Based on this property, we can map each AND/OR net transition into two directed transitions in the Petri net. One difference between the AND/OR net and Petri net is that the same state in the AND/OR net could appear more than one time. Thus the Petri net is a less efficient representation of the system states, but offers advantages in modeling the generation of sequences.

For a transition $\{\lambda_1, \lambda_2\} \in N$ in an AND/OR net, the *mapping* to elements in a Petri net is defined as a function $\mathcal{F}_1$, where

$$\mathcal{F}_1(\{\lambda_1, \lambda_2\}) = (\lambda_1, t_1) \bigcup (t_1, \lambda_2) \bigcup (\lambda_2, t_2) \bigcup (t_2, \lambda_1).$$

For a Transition $\{\lambda, \psi\} \in A$ in an AND/OR net, the *mapping* to elements in a Petri net is defined as a function $\mathcal{F}_2$, where

$$\mathcal{F}_2(\{\lambda, \psi\}) = (\lambda, t_1) \bigcup \bigcup_{i=1}^{k} (t_1, \lambda_i) \bigcup \bigcup_{i=1}^{k} (\lambda_i, t_2) \bigcup (t_2, \lambda), \qquad \lambda_i \in \psi.$$

The algorithm for converting the AND/OR net to Petri net is shown as follows:

Algorithm 1 *Mapping from AND/OR net to Petri net.*
Input: AND-OR net $N_A = (S, A, N)$.
Output: Petri net $N_P = (P, T, \alpha, \beta)$.

```
1. initialize $P = T = \alpha = \beta = \emptyset, n_P = n_T = 0$;
2. for each set $n_i \in N, n_i = \{n_{i1}, n_{i2}\}$
   begin
       add 2 transitions $t_{n_P+1}, t_{n_P+2}$,
```

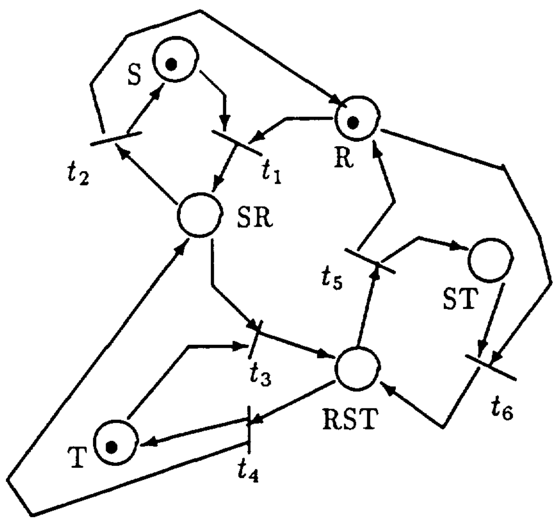
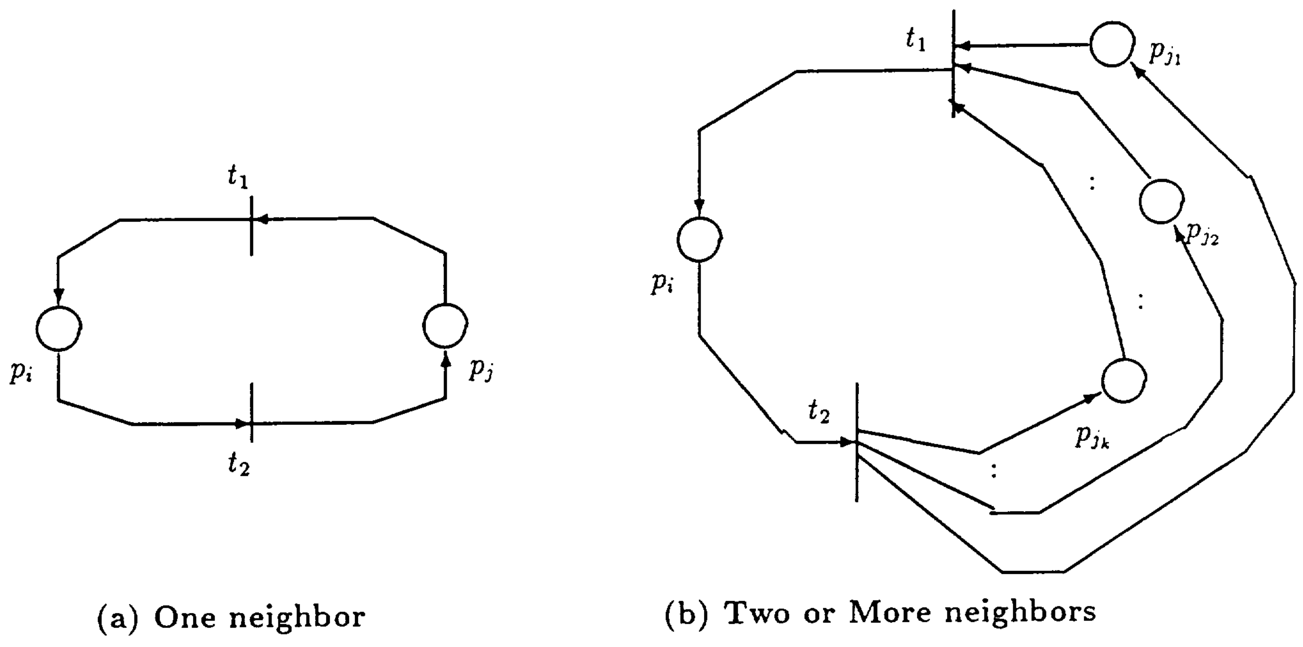


Figure 4. The Petri net representation for the example

```
            $T = T \bigcup \{t_{n_P+1}, t_{n_P+2}\}$;
        $n_P = n_P + 2$;
        check whether $n_{i1}, n_{i2}$ is in P,
            if $n_{i1}$ not in P, $P = P \bigcup \{n_{i1}\}$;
            if $n_{i2}$ not in P, $P = P \bigcup \{n_{i2}\}$;
        $\alpha = \alpha \bigcup \{(n_{i1}, t_{n_P+1}), (n_{i2}, t_{n_P+2})\}$;
        $\beta = \beta \bigcup \{(t_{n_P+1}, n_{i2}), (t_{n_P+2}, n_{i1})\}$;
    end
3. for each set $a_i \in A, a_i = \{a_\lambda, \psi_a\}$
    begin
        add 2 transitions $t_{n_P+1}, t_{n_P+2}$,
            $T = T \bigcup \{t_{n_P+1}, t_{n_P+2}\}$;
        $n_P = n_P + 2$;
        for every $e_j \in \{\{a_\lambda\} \bigcup \psi_a\}$ and $\{e_j\} \bigcap P = \emptyset$,
            $P = P \bigcup \{e_j\}$;
        $\alpha = \alpha \bigcup \bigcup_j \{(a_\lambda, t_{n_P+1}), (e_j, t_{n_P+2})\}$, for all $e_j$;
        $\beta = \beta \bigcup \bigcup_j \{(t_{n_P+1}, a_\lambda), (t_{n_P+2}, e_j)\}$, for all $e_j$;
    end
```

Such a Petri net representation for the example in figure 1 is shown in figure 4. The initial marking of one token in places S, R and T represents the initial states, i.e., there are one robot, one table and one solid available and they are geometrically independent for the time being. The following properties of the resulting Petri net may be shown as follows:



(a) One neighbor (b) Two or More neighbors

Figure 5. The Connectedness with P_i and its neighboring Places

Theorem 1 *The Petri net mapped from an AND/OR net is safe.*
Proof: Because the Petri net is mapped from an AND/OR net, we should clarify the meanings of *AND_arc* and *IST_arc* again. The IST_arc in the AND/OR net corresponds to the internal geometric state change inside an object. All possible assemblies of n objects, and all possible subassemblies of m objects are special cases of objects. The AND_arc in the AND/OR net corresponds to combining geometric state changes among two or more objects.

Suppose $C = (P, T, \alpha, \beta)$ with initial marking μ. Choose any place $p_i \in P$, $\mu' \in R(C, \mu)$, if we could verify that $\mu'(p_i) \leq 1$, then the proof is complete.

p_i might connect with neighboring places in two ways(figure 5).

Case 1: Corresponding to the internal change of geometric state of an object or a relating set of objects in the system(figure 5(a)).

Suppose $\mu'_t(p_i) \geq 2$. For simplicity, we assume $\mu'_t(p_i) = 2$. There should exist $t' < t$ such that $\mu'_{t'}(p_i) = 1$ and $\mu'_{t'+1}(p_i) = 2$. Therefore at time t', $\mu'_{t'}(p_i) = \mu'_{t'}(p_j) = 1$, p_j and p_i are neighboring places in the Petri net. Because the Petri net is mapped from the AND/OR net and this case concerns the internal geometric change, we conclude that at the same time t', the two possible internal feasible states of a single object or a relating set of objects could appear simultaneously. The contradiction is thus obtained.

Case 2: Corresponding to the change of interrelationship among two or more objects in the system.

(i) p_i is as shown in Figure 5(b).

Also suppose $\mu'_t(p_i) = 2$. There should exist $t' < t$ such that $\mu'_{t'}(p_i) = 1$ and $\mu'_{t'+1}(p_i) = 2$. Therefore at time t', $\mu'_{t'}(p_i) = \mu'_{t'}(p_{j_1}) = \mu'_{t'}(p_{j_2}) = \cdots = \mu'_{t'}(p_{j_k}) = 1$, $\mu'_{t'}(p_{j_1}), \cdots, \mu'_{t'}(p_{j_u})$, $\cdots$, $\mu'_{t'}(p_{j_k})$ are combining neighboring geometric states in the AND/OR net, $1 \leq u \leq k$. We conclude that at the same time t', the two possible combining geometric states, which include exactly the same objects, could appear simultaneously. A contradiction is thus obtained again.

(ii) p_i is in the place of p_{j_u} as shown in Figure 5(b).

Follow the same procedure as in case 1. We could conclude that at a time t', two possible combining geometric states, which contain at least one common object, would appear simultaneously. A contradiction is obtained.

This statement of safeness concerning the Petri net is therefore assured.

Corollary 1 *The Petri net mapped from an AND/OR net is bounded.*

The corollary is directly derived from Theorem 1 because the number of tokens in any place cannot exceed 1.

Theorem 2 *The Petri net mapped from an AND/OR net is live.*
Prove: The Petri net mapped from AND/OR net consists of two kinds of subnet which are actually loops as shown in figure 5. Therefore, for any transition t_i in this Petri net and any $\mu' \in R(C, \mu)$, where μ is the initial marking, there exists a firing sequence σ such that t_i is enabled in $\delta(\mu', \sigma)$. We thus conclude that the transition t_i is live at level 4 and therefore *live*. Because t_i is an arbitrary transition, the Petri net is thus live.

Theorem 3 *The Petri net mapped from an AND/OR net is reversible.*

Prove: As shown in the proof of Theorem 2, the Petri net is a set of loops according to the *mapping* definitions. Therefore, the initial marking is reachable from all reachable markings. The Petri net is thus reversible.

The live Petri net guarantees a deadlock-free system. The boundedness property ensures that the capacity is not exceeded. And the reversibility implies that the system can re-initialize itself, and it is important for the automatic recovery from errors and failures. Therefore, if a robotic assembly or handling system is represented as an AND/OR net, it is not only convenient for the system to generate task plans, but also the controller will supervise and coordinate the system more efficiently.

This mapped Petri net does not satisfy the property of conservation because of the geometric characteristics of the system.

4 Data Structure for Searching Sequences in Petri Net

The task planning problem is concerned with reachable markings when the system has been modeled with a Petri net. The reachability problem is the most basic Petri net analysis problem[13]. Many other analysis problems can be stated in terms of the reachability problem.

Definition 9 The Reachability Problem: *Given a Petri net C with marking μ and a marking μ', is $\mu' \in R(C,\mu)$?*

Another problem which is similar to reachability but is slightly different is called the coverability problem. A marking μ'' covers a marking μ' if $\mu'' \geq \mu'$.

Definition 10 The Coverability Problem: *Given a Petri net C with initial marking μ and a marking μ', is there a reachable marking $\mu'' \in R(C,\mu)$ such that $\mu'' \geq \mu'$?*

Briefly speaking, finding a possible sequence is concerned with the reachability problem and finding all possible sequences is concerned with the coverability problem.

Before we describe the algorithm for obtaining the reachability tree for the system Petri net, we state some useful definitions, i.e., *frontier nodes, terminal nodes and duplicate nodes*[13]. The following algorithm is different from general reachability algorithms in the sense of the generation of leaves and the stopping criteria for the developing of the tree because our resultant Petri net will have no *dead marking* as we have shown before.

Definition 11 *Frontier nodes, terminal nodes and duplicate nodes* When we develop the reachability tree, all current new markings are called *frontier nodes.* The markings, in which no transition is enabled or depending on some criterion, no further transitions are necessary, are called *terminal nodes.* The markings which have previously appeared in the tree are called *duplicate nodes.*

In the system Petri net, we define the states of the system as *markings.* The algorithm therefore begins by defining the initial marking, which corresponds to the initial state in the system, to be the root of the tree. The final state of the system is defined as the final marking in the Petri net. All leaves in the reachability tree are defined to be the final marking. The algorithm is based on a breadth-first forward chaining criterion.

Algorithm 2 *The generation of the reachability tree.*
Input: Petri net $N_P = (P, T, \alpha, \beta, \mu_0)$, final marking μ_f.
Output: The corresponding reachability tree.
Let x be a frontier node to be processed.

1. If there exists another node y in the tree which is not a frontier node, and has the same marking associated with it, $\mu[x] = \mu[y]$, then classify node x as a *duplicate node.*

2. If the marking $\mu[x] = \mu_f$, then classify node x as a *terminal node.*

3. For all transitions $t_j \in T$ which are enabled in μ[x], [i.e., $\delta(\mu[x], t_j)$ is defined], create a new node z in the reachability tree. The marking $\mu[z]$ associated with this new node is, for each place p_i, $\mu[z]_i = \delta(\mu[x], t_j)_i$.
An arc, labeled t_j, is directed from node x to node z. Node x is redefined as an *interior node*; node z becomes a *frontier node.*

4. When all nodes have been classified as terminal, duplicate, or interior, duplicate nodes will copy all descendants from its corresponding nodes. Then duplicate nodes will be reclassified as interior nodes.

5. When all nodes have been classified as terminal or interior, the algorithm stops.

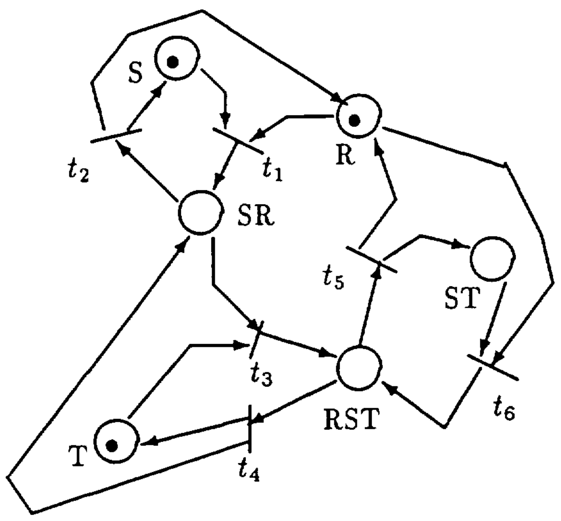
Following the algorithm shown as above, we could generate a reachability tree for the sample Petri net shown in Figure 4 and finally we could obtain the assembly sequences required. In this case, only one shortest sequence is available. The sequence of transition firings and the corresponding sequence of markings are shown in figure 6. Note that the sequence of places in markings are: R, S, T, SR, ST and RST, respectively.

5 Task Sequence planning for a Practical Problem

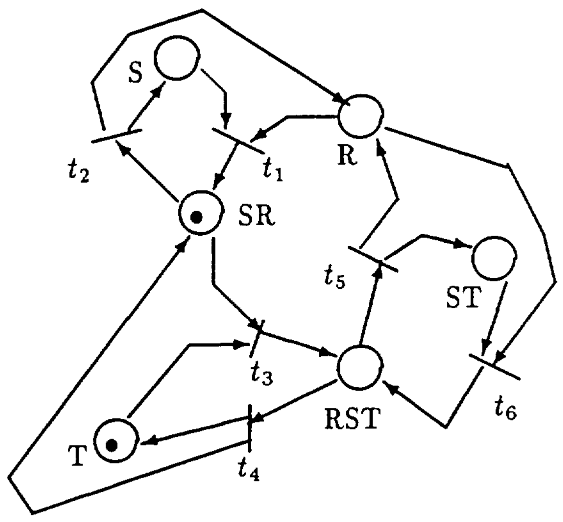
In this section we consider a practical problem using the methods described above. Our solutions to this problem will still be based on the task geometric descriptions. The states of the system is completely decided by the geometric relationships amongst the objects and their internal geometric states. Each *efficient* operation which will be represented as a corresponding transition in the Petri net acting inside the system will cause the changes of feasible geometric states.

The system consists of a robot, two tables and a book. Figure 7 shows an initial state and the desired final state which could be mapped to the geometric descriptions. The only difference between these two states is where the book is. Therefore, we consider all feasible geometric states for one object, within two objects and among three objects(figure 8). For this problem, we assume the maximum size of the robot gripper is not large enough for the robot to grasp the book when the book is fully lying on the table. It is thus necessary to first move the book to the edge of the table and try to pick up the book from one side of the book. This is the reason why we distinguish two cases for the connectedness of the book and a table. For the connectedness of three objects, the cases are a little bit more complex because of the relative geometric relations between the book and table. When the book is on the edge of the table and the robot is touching one side of the book from our view, we ignore the place of the robot relative to the table, i.e., $(T_1BR)_4$ and $(T_2BR)_4$ may include two kinds of geometric relations.

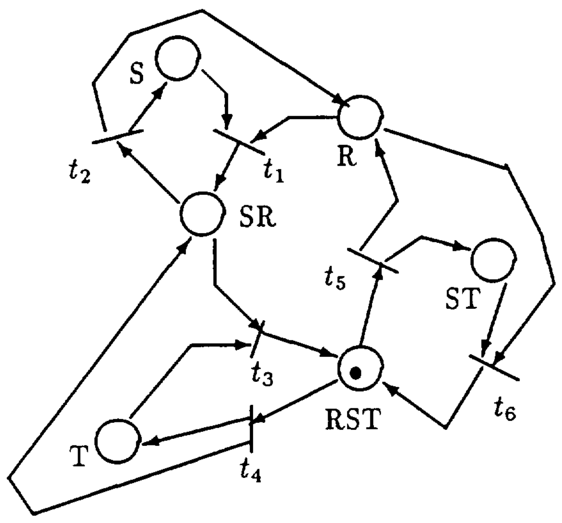
From the system geometric states diagram, we obtain the AND/OR net representation for the moving task(figure 9). For simplicity of the figure, the net is shown as separate subnets, but because of the common nodes in each subnet, it is really a connected net. We map this AND/OR net to the Petri net(figure 10) following Algorithm 1 described before. The description for each operation could be deduced from the Petri net and the corresponding system geometric states descriptions. By searching the reachability tree of this Petri net and following the Algorithm 2, we obtain all possible task sequences and the optimal sequence or the fewest-number-step sequence in this case is shown as follows.



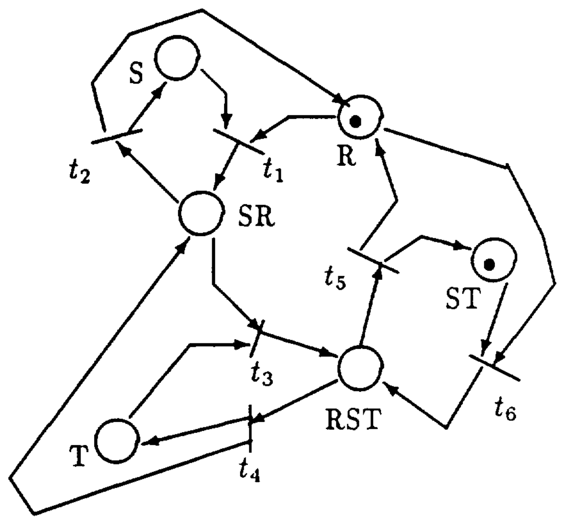
(a) Transition fireable: t_1
Marking: 111000
Operation: None



(b) Transition fireable: t_2, t_3
Marking: 001100
Operation: Robot grasps solid

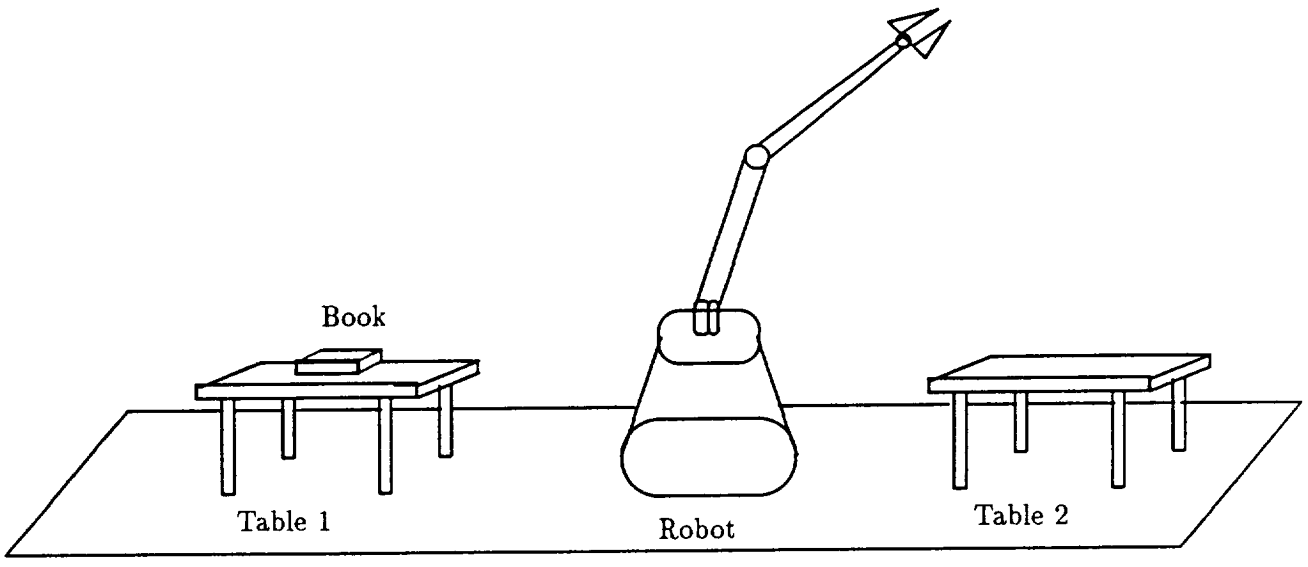


(c) Transition fireable: t_4, t_5
Marking: 000001
Operation: Robot reaches table

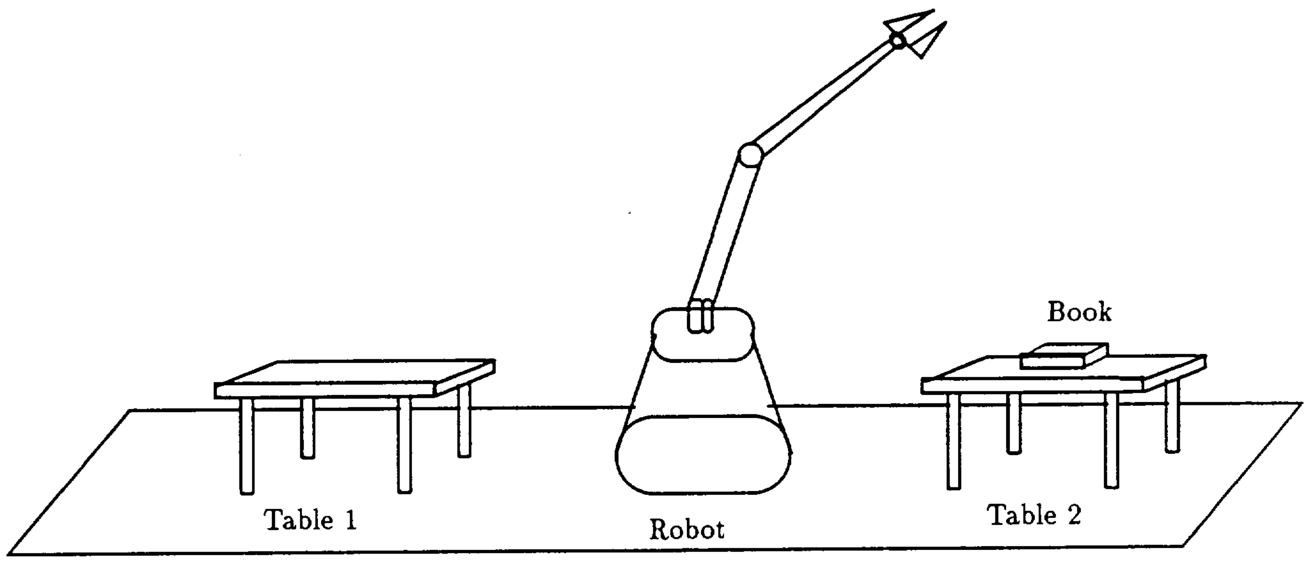


(d) Transition fireable: t_6
Marking: 100010
Operation: Robot leaves table

Figure 6. The sequence of markings and corresponding operations



(a)



(b)

Figure 7. A Robot Moves a Book from Table 1 to Table 2
(a) Initial states (b) Final states

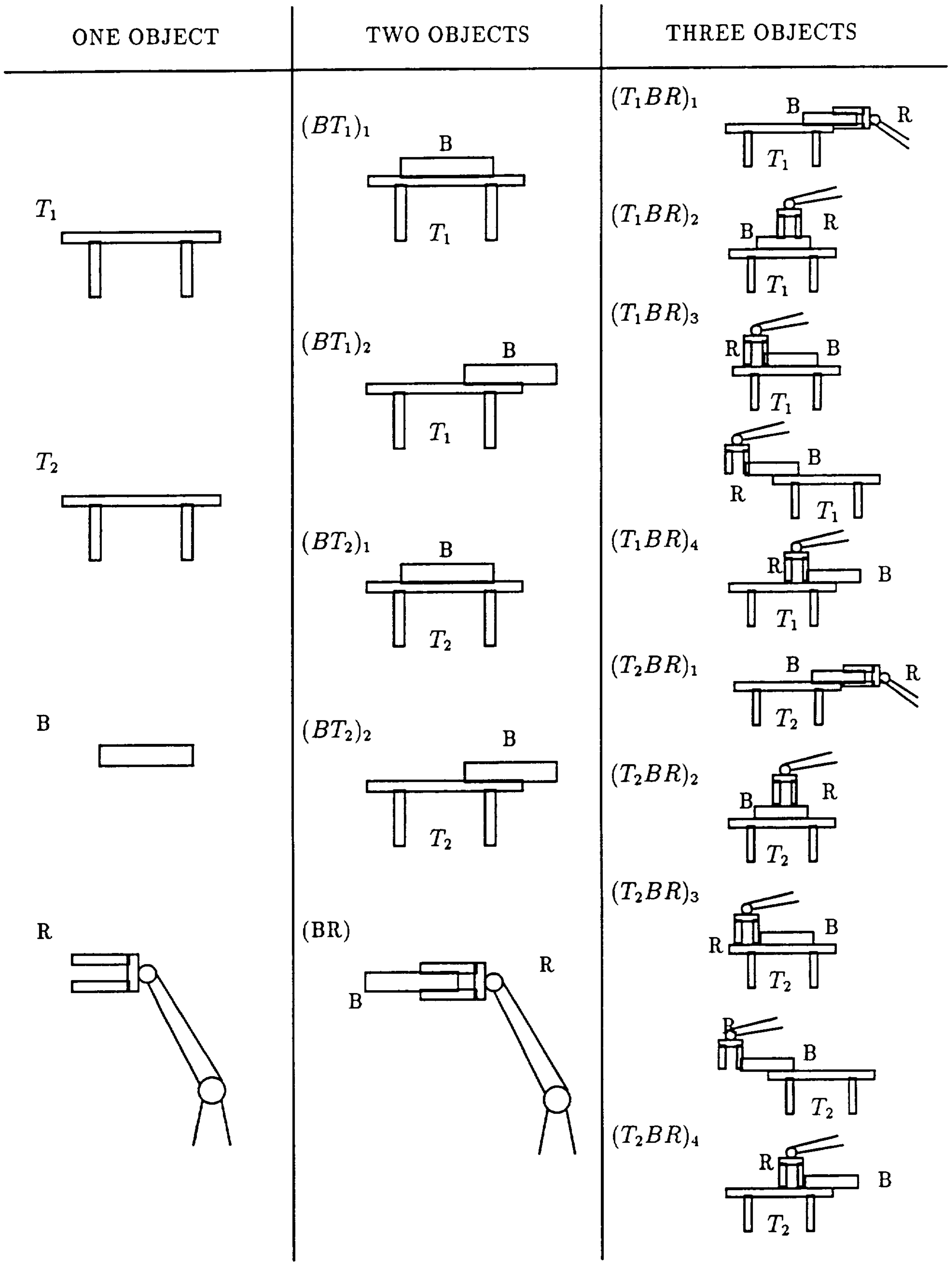


Figure 8. System Geometric States Diagram for Moving Book

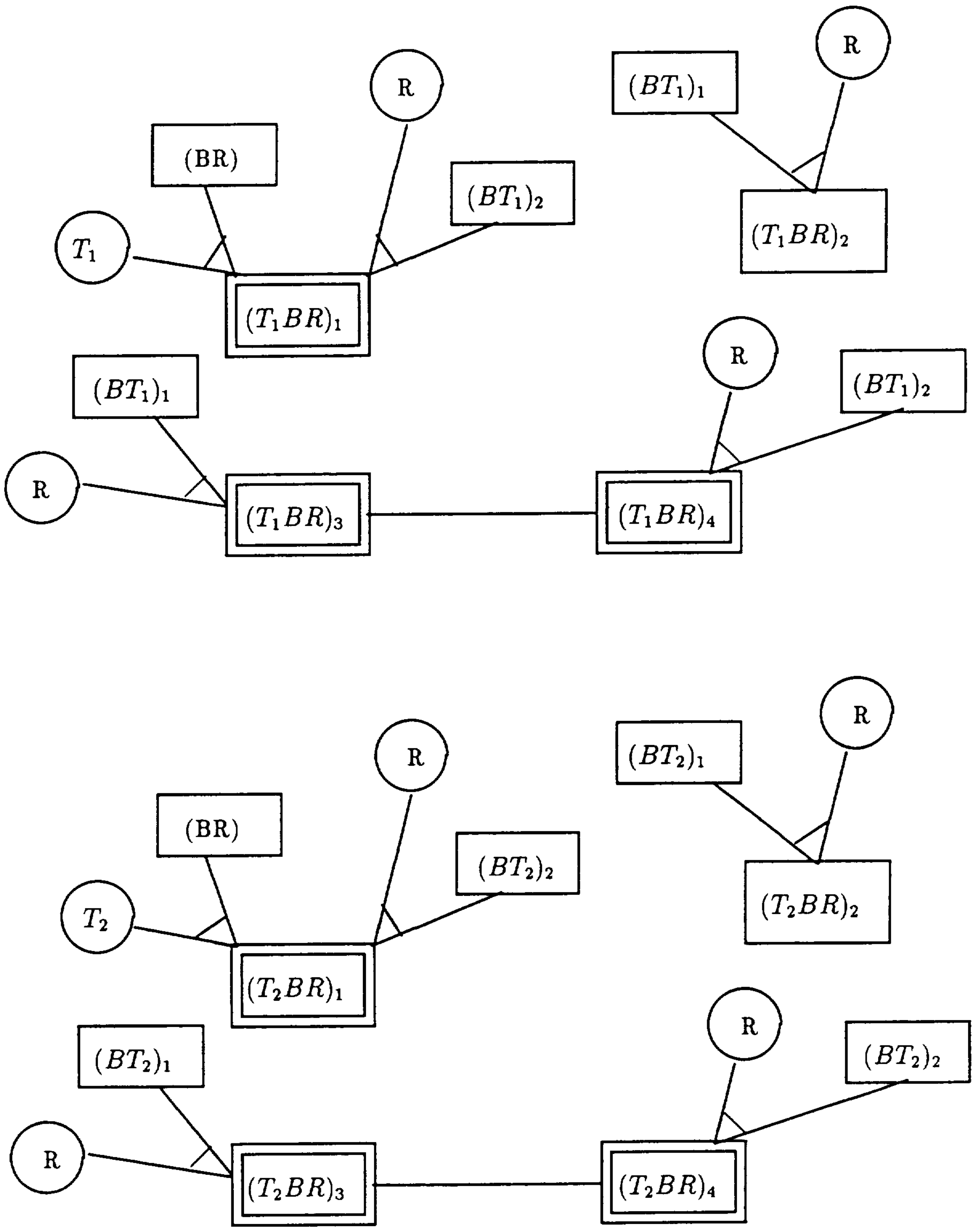


Figure 9. The AND/OR net representation for Moving Book

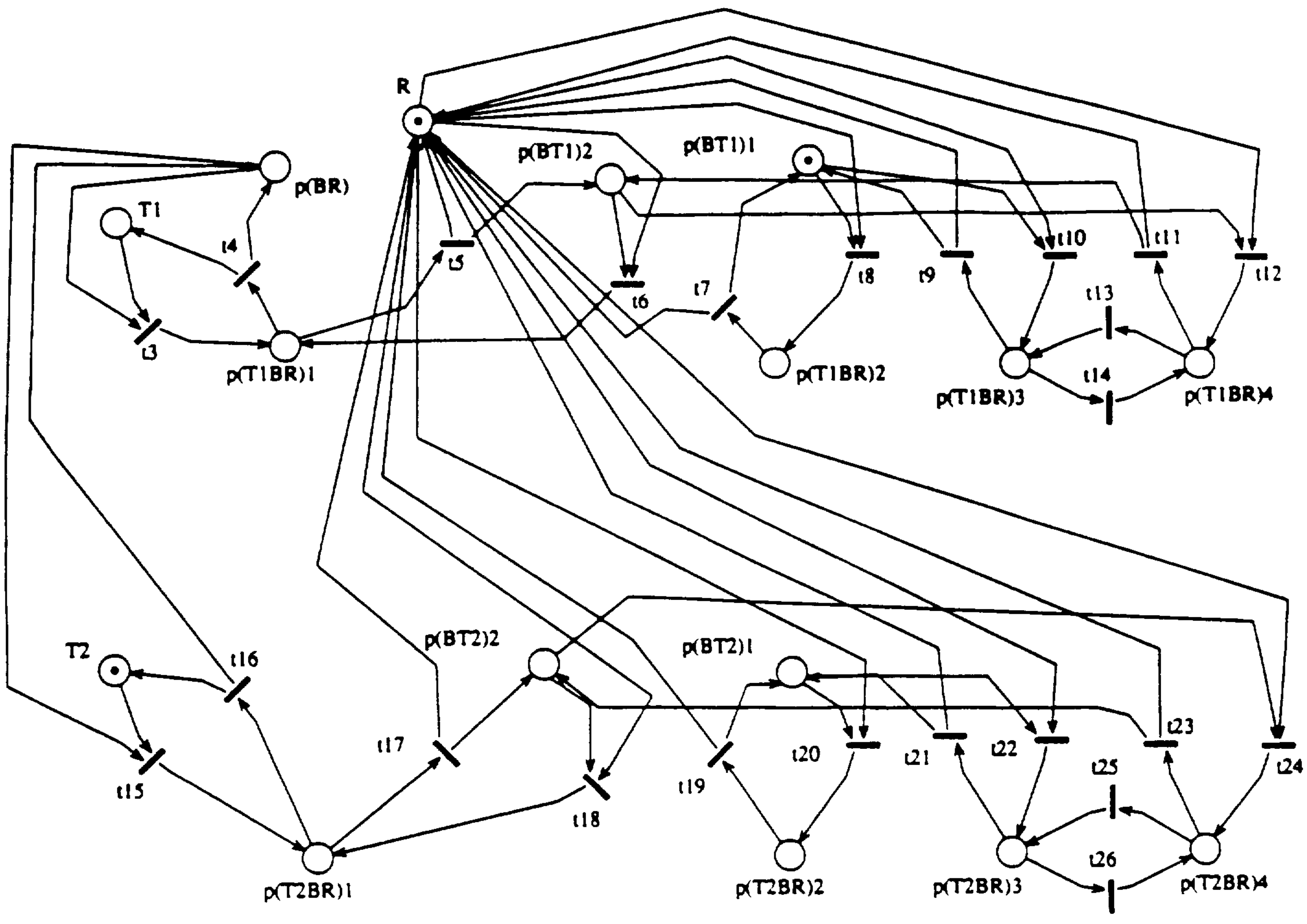


Figure 10. The Petri net representation for Moving Book

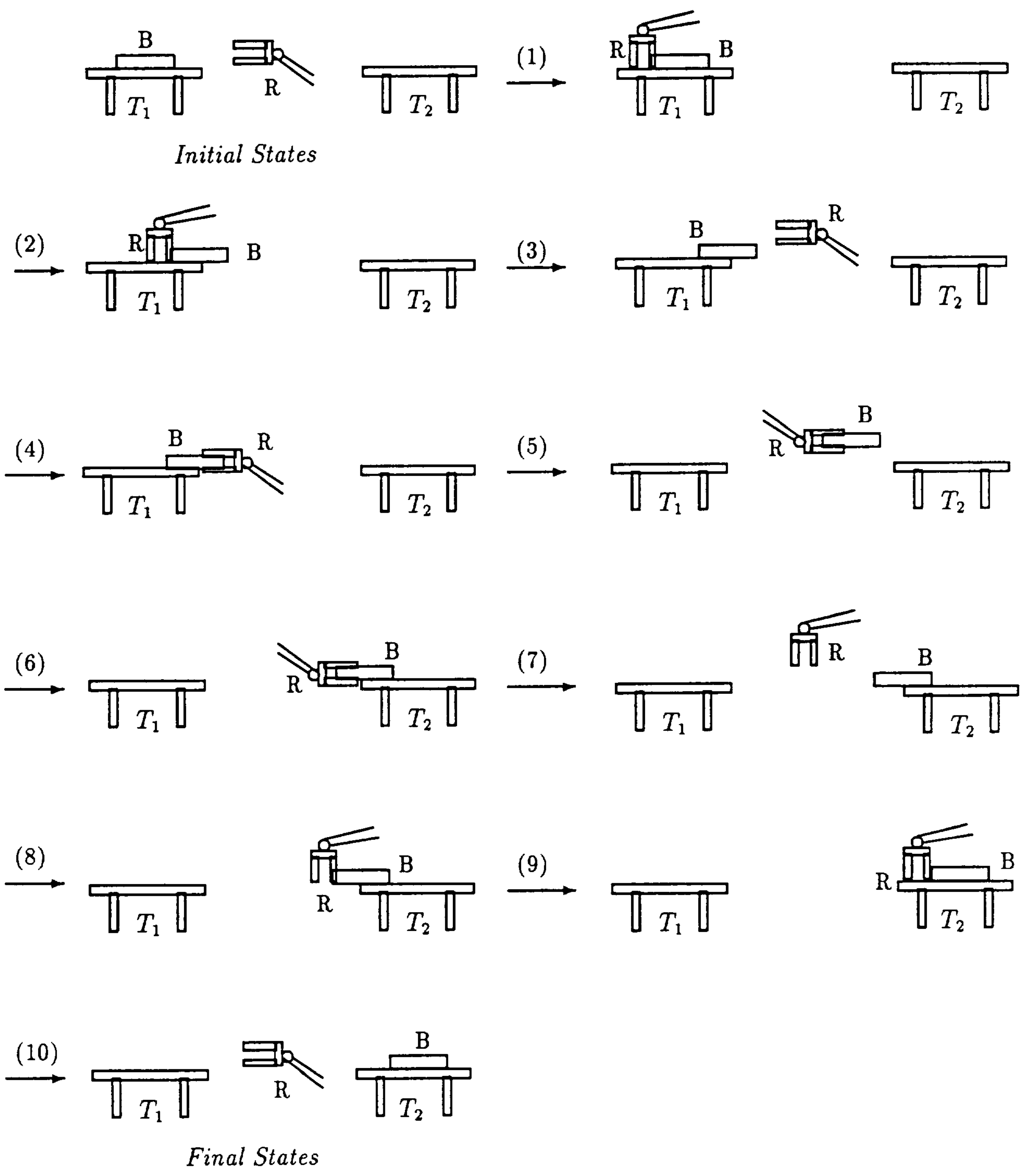


Figure 11. Task Sequences for Moving Book

1. t_{10}: the robot moves towards table 1 and touches the book which is lying on the surface of table 1.
2. t_{14}: the robot forces the book on table 1 to move to the edge of the table.
3. t_{11}: the robot leaves the book and table 1.
4. t_6: the robot reaches table 1 again and grasp the book towards the edge of table 1.
5. t_4: the robot which has grasped the book leaves table 1.
6. t_{15}: the robot moves towards table 2 and make the book touch the surface and lie on the edge of table 2.
7. t_{17}: the robot leaves the book and table 2.
8. t_{24}: the robot touches the book again but the direction of the gripper has already been changed.
9. t_{25}: the robot forces the book to move to the center of the surface of table 2.
10. t_{21}: the robot leaves the book and table 2 and then go back to its original place.

This is the only optimal solution which could be found for this problem. This optimal task sequence is directly perceived through Figure 11. This sequence is obviously reversible.

6 Conclusions

The AND/OR net is introduced as a tool for representation and reasoning about geometric constraints in a robotic workcell system. A method for mapping the AND/OR network to a Petri net is provided. We could obtain all possible task sequences by constructing the reachability tree from the Petri net. This off-line planning system has been implemented. The ideas presented here can be applied to other robotic planning problems in manufacturing and non-manufacturing domains.

Acknowledgment

This research is supported by the Defense Logistics Agency and the NASA Center for Intelligent Robotic Systems for Space Exploration at Rensselaer Polytechnic Institute.

References

[1] T. Lozano-Perez, "Task planning," in *Robot Motion: Planning and Control*, J. M. Brady *et al.*, Eds. Cambridge, MA: MIT Press, 1982, pp. 473-498.

[2] E. Sacerdoti, *A Structure for Plans and Behavior.* New York: North-Holland, 1977.

[3] R. Fikes *et al.*, "Learning and executing generalized robot plans," in *Readings in Artificial Intelligence*, N. Nilsson and B. Webber, Eds. Palo Alto, CA: Tioga, 1981, pp. 231-249.

[4] D. E. Wilkins, "Domain-independent planning: Representation and plan generation," *Artificial Intell*, vol. 22, no. 3, pp. 269-301, 1984.

[5] L. S. Homem de Mello and A. C. Sanderson, "AN AND/OR Graph Representation of Assembly Plans," *AAAI-86 Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 1113-1119, 1986.

[6] L. S. Homem de Mello and A. C. Sanderson, "Task Sequence planning for assembly," in *IMACS World Congress '88 on Scientific Computation*, Paris, July, 1988.

[7] L. S. Homem de Mello and A. C. Sanderson, *Automatic Generation of Mechanical Assembly Sequences*, Technical Report CMU-RI-TR-88-19, The Robotics Institute, Carnegie Mellon University, December, 1988.

[8] L. S. Homem de Mello and A. C. Sanderson, "AND/OR Graph Representation of Assembly Plans," In *IEEE Trans. on Robotics and Automation*, Vol. 6, No.2, pp. 188-199, 1990.

[9] A. C. Sanderson, L. S. Homem de Mello, and H. Zhang, "Assembly Sequence Planning," *AI Magazine: Special Issue on Assembly Planning*, Vol. 11, No. 1, Spring, 1990, pp 62-81.

[10] L. S. Homem de Mello and A. C. Sanderson, "Evaluation and Selection of Assembly Plans," *Proc. 1990 International Conference on Robotics and Automation*, pp. 1588-1593.

[11] L. S. Homem de Mello and A. C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Trans. on Robotics and Automation*, Vol. 7, No. 2, pp. 228-240, 1991.

[12] T. Agerwala, "Putting Petri Net to work," *Computer*, vol. 12, no. 12, pp. 85-94, 1979.

[13] J. L. Peterson, *Petri net, Theory and the Modelling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[14] P. Alanche *et al.*, "PSI: A Petri Net Based Simulator for Flexible Manufacturing Systems," in *Advances in Petri Net* 1984, *Lecture Notes in Computer Science 188*, G. Rozenberg, Ed. New York: Springer-Verlag, pp. 1-14.

[15] K.-H. Lee and J. Favrel, "Hierarchical reduction method for analysis and decomposition of Petri nets," *IEEE Trans. Syst. Man Cybern.*, vol. 15, no. 2, pp. 272-280, 1985.

[16] R. Valette, M. Courvoisier, H. Demmou, J. M. Bigou and C. Desclaux, "Putting Petri Net to Work for Controlling Flexible Manufacturing Systems," *Proceedings of the International Symposium on Circuits and Sytems*, pp. 929-932, Kyoto, Japan, 1985.

[17] N. Vishwanadham and Y. Narahari, "Colored Petri net models for automated manufacturing systems," In *Proc. 1987 IEEE Robotics and Automation Conference*, pages 1985-1990, Raleigh, NC, 1987.

[18] R. Al-Jaar and A. Desrochers, "Petri Nets for Automation and Manufacturing," *Advances in Automation and Robotics.*, vol. 2, Ed. G.N. Saridis, JAI Press, 1988.

[19] W. Zhang, "Representation of Assembly and Automatic Robot Planning by Petri Net," *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 2, pp. 418-422, 1989.

[20] M. Ajmone Marsan, G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems," *ACM Transactions on Computer Systems* 2(1), May, 1984.

[21] M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte, "Generalized Stochastic Petri Nets Revisted: Random Switches and Priorities," in *Proc. Int. Workshop on Petri Nets and Performance Models*, IEEE-CS Press, Madison, WI, USA, August, 1987.