An Application of LLP Supervisory Control with Petri Net Models in Mobile Robots

Keigo Kobayashi and Toshimistu Ushio
Department of Systems and Human Science , Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan

Abstract

This paper is concerned with high level control of robot systems. We abstract the robot system as a discrete event system and give a high level controller which consists of an LLP supervisor and a Petri nets model. We give its simple application to a mobile robot system.

1 Introduction

Recently, supervisory control theory has been proposed for controlling logical discrete events systems by Ramadge and Wonham[1], and its applications to robotics have been studied[2]–[6]. In supervisory control, behaviors of controlled systems are described by formal languages or automata, and assumed to be time-invariant. But in robot systems, unexpected events may occur, and interactions with environments are time-varying. So, limited lookahead policies(LLPs)[7] are very useful strategies for robot systems.

In this paper, we consider searching problems in mobile robots. Petri nets are used as unified models to represent maps, procedures of tasks, the qualitative state of the robot, and so on. Since a model of the interactions with environments changes dynamically according to what the mobile robot is conscious of. We describe the consciousness by stepwise refinement[8] of Petri nets.

2 Petri Nets

2.1 Mathematical Definition

We review a definition of Petri nets. Let P be a set of places and T a set of transitions. The marking $m : P \to \mathrm{N}$ is the number of tokens in a place, where N denotes the set of all nonnegative integers, and $M = \{m \mid m : P \to \mathrm{N}\}$ is a set of markings. The connection $A : (P \times T) \cup (T \times P) \to \mathrm{N}$ describes input and output arcs of a transition. Then the Petri net is defined as $G = \{P,\ T,\ A,\ m_0\}$, where $m_0 \in M$ is an initial marking.

Graphical representations of Petri nets are as follows: places, transitions, connections, and tokens are represented by circles "◯", bars "|", arcs "→", and bullets "•", respectively. The arcs connected to a transition show how many tokens move when the transition fires.

Let $m \in M$ and $t \in T$. If $m(p) \geq A(p,t)\ (\forall p \in P)$, then the transition t is said to be enabled and we write $m(t >$. If the enabled transition t fires, the marking changes to m' where

$$m'(p) = m(p) - A(p,t) + A(t,p) \quad (\forall p \in P) \quad , \quad (1)$$

and we write $m(t > m'$.

2.2 Composing Petri Nets

Let $G_1 = \{P_1,\ T_1,\ A_1,\ m_{10}\}$ and $G_2 = \{P_2,\ T_2,\ A_2,\ m_{20}\}$ be Petri nets where $P_1 \cap P_2 = \emptyset$ and $T_1 \cap T_2 \neq \emptyset$. We can get the composed Petri net as follows.

Let $P = P_1 \cup P_2$ and $T = T_1 \cup T_2$. To make the notation brief, we extend the domain of a marking naturally to get $m_{10}(p_2) = 0\ (\forall p_2 \in P_2)$ and $m_{20}(p_1) = 0\ (\forall p_1 \in P_1)$. Then we define a marking $m_0 : P \to \mathrm{N}$ such that

$$m_0(p) = m_{10}(p) + m_{20}(p) \quad (\forall p \in P) \quad . \quad (2)$$

We also define $A : (P \times T) \cup (T \times P) \to \mathrm{N}$ such that

$$A(p,t) = A_1(p,t) + A_2(p,t) \quad (\forall p \in P,\ \forall t \in T) \quad (3)$$

and

$$A(t,p) = A_1(t,p) + A_2(t,p) \quad (\forall p \in P,\ \forall t \in T) \quad (4)$$

with similar extensions of the domains. We express above relations as $m_0 = m_{10} + m_{20}$ and $A = A_1 + A_2$.

Proc. of the 2000 IEEE International Conference on Systems, Man & Cybernetics, 3015/3020 (2000)

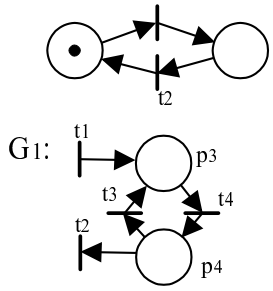

Figure 1: Petri nets

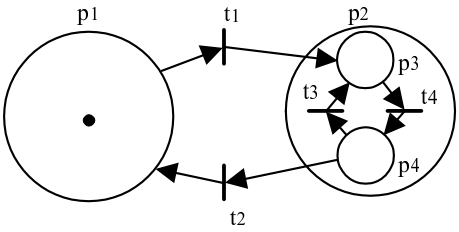

Figure 2: Refinement of place p_2

Now we can get a composition $G_1+G_2 = \{P_1 \cup P_2,\ T_1 \cup T_2,\ A_1 + A_2,\ m_{10} + m_{20}\}$.

We give a simple example of step refinement of a place. There are two Petri nets G and G_1 in Fig.1. Then the composed Petri nets $G + G_1$ can be represented as Fig.2. Figure 2 shows that adding G_1 to G makes the place p_2 refined. On the other hand, removing G_1 from $G + G_1$ is called reduction. We note that transitions $t_1,\ t_2 \in T \cap T_1$ are only enabled in $G + G_1$ when they are enabled in G and in G_1 simultaneously. For valid refinement, It is assumed that

$$\sum_{p \in P_1} m_{10}(p) = m_0(p_2) \quad . \tag{5}$$

3 Supervisory Control

Supervisory control theory has been proposed for logical control of discrete event systems[1]. We assume that transitions are classified into controllable and uncontrollable transitions: firing of controllable transitions are prohibited by external controllers called supervisors while uncontrollable transitions can fire whenever they are enabled. Let $T_c \subset T$ be a set of controllable transitions. We introduce a mapping

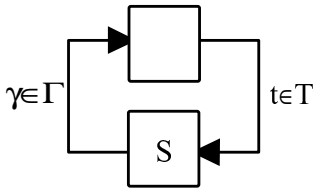

Figure 3: Supervisor

$\gamma : T_c \to \{0,\ 1\}$ called a control pattern. Each controllable transition t can fire if it is enabled and $\gamma(t) = 1$. It is often said that t is control enabled if $\gamma(t) = 1$.

For a Petri net G, a language generated by G denotes $L(G) \in T^*$, where T^* is the Kleene closure of T. A logical controller called a supervisor S is formally defined by $S : L(G) \to \Gamma$. A supervisor selects a control pattern according to firing of transitions so that every *trace*(a sequence of transitions) in the controlled Petri net is restricted in a set of desirable traces. Shown in Fig. 3 is a diagram of the Petri net G controlled by the supervisor S.

Ramadge and Wonham have proposed off-line design of a supervisor where a reachability graph of G is used. But the off-line design is not suitable for a large reachable set and/or time varying systems. Since we will apply supervisory control in robot motion planning under time-varying environments, we adopt another design approach called the limited lookahead policy(LLP). An LLP supervisor predicts all possible traces from the current state, which forms a tree, and selects a control pattern. When it observes firing of a transition, it updates the tree and reselects a control pattern.

4 Controlled Systems

In this section, we propose a high level controller using an LLP supervisor with Petri net based modeling.

4.1 Robot System

In this paper, we propose a high level controller of a robot system. The robot system is supposed to have low level controllers, which are assumed to have corresponding sub tasks and to control the actuators of the robot appropriately in order to achieve the sub tasks.

The robot system is also supposed to have an event detector which uses the sensor information and detects firing of events such as the completion/failure of a sub

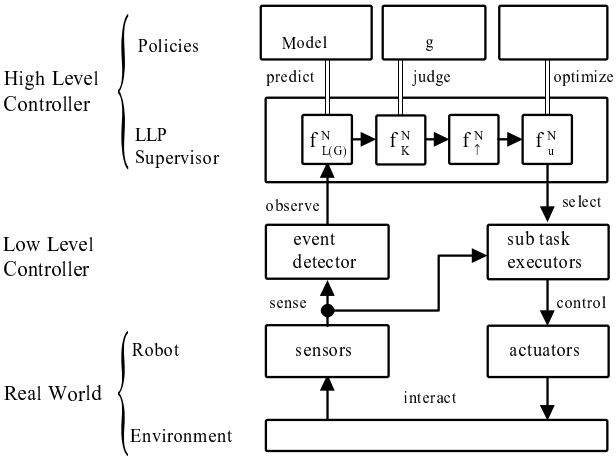

Figure 4: System

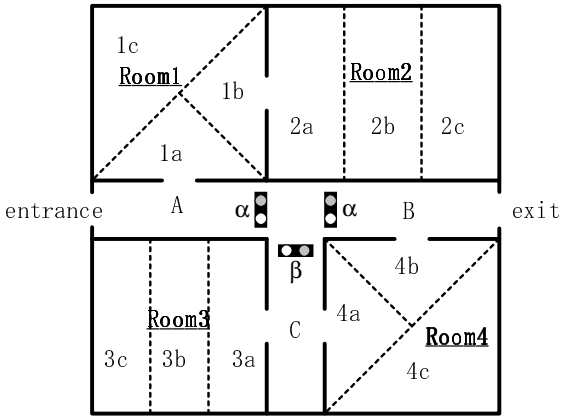

Figure 5: Map of a building

task or changing of discrete state of the system.

Our aim in this paper is to give a high level controller which observes the trace and selects an appropriate controllable event to complete a task successfully.

4.2 High Level Controller

Figure 4 shows the block diagram of the system consists of controllers and a controlled robot system. The proposed controller consists of an event detector, a high level controller, a low level controller, and Petri nets which model information on the robot system such as maps, qualitative states of robots, and so on. The high level controller (HLC) based on an LLP supervisor consists of four parts: generating a prediction tree, detecting illegal regions, calculating the supremal controllable sub language, and selecting the optimal command. The detail procedure in the HLC is a follows:

1) If the event detector observes the firing of a transition, the HLC makes the transition fire on the Petri nets model, and predicts the behavior of the composed Petri net N steps ahead where one step means the firing of one transition. This is the function block $f^N_{L(G)}$ in Fig. 4 and it results an N-tree.

2) The HLC determines which traces in the N-tree are illegal with the policy. This is the function block of the f^N_K.

3) The HLC calculates the supremal controllable sublanguage. It trims all nodes in the illegal regions and if there is an uncontrollable transitions from a node into an illegal region, the node is also trimmed. This is the function block of the $f^N_\uparrow$.

4) The HLC chooses the optimal trace with a cost function. If the first transition in the trace is controllable, the transition is made to be enabled. This is the function block of the f^N_u.

In order to design the HLC, we must model the behavior of the robot system and give policies to detect the illegal region and to evaluate a trace in the prediction tree. We give an example in the next section to show how to design the Petri nets model and the policies.

5 Application to a Mobile Robot

We show an application to a mobile robot. The robot moves in a building depicted in Fig. 5. There are an entrance, an exit, rooms 1 to 4, hallways A to C, traffic signals α and β in the building. When the signal α is green and β is red, if the robot is in hallway A or C, it can go to any hallways, but if it is in hallway B, it cannot go to any other hallways, and vice versa. The robot has vision sensors and is supposed to be able to see the traffic signals. It is also supposed that the robot has a map of the building in advance and can get where it is on the map with its vision sensors.

An object is supposed to be in the building, but the robot doesn't know where in the building the object is. The task for the robot is to enter the building, to search an object in the building, and to exit from the building after finding the object.

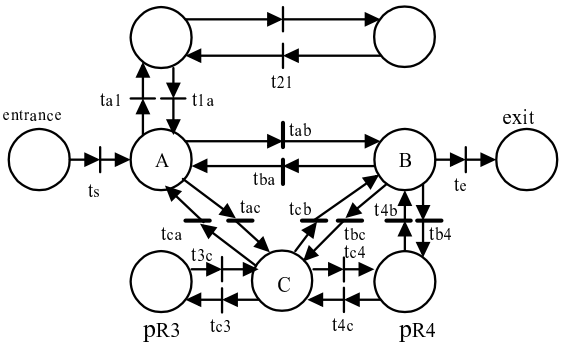

Figure 6: Map of the building

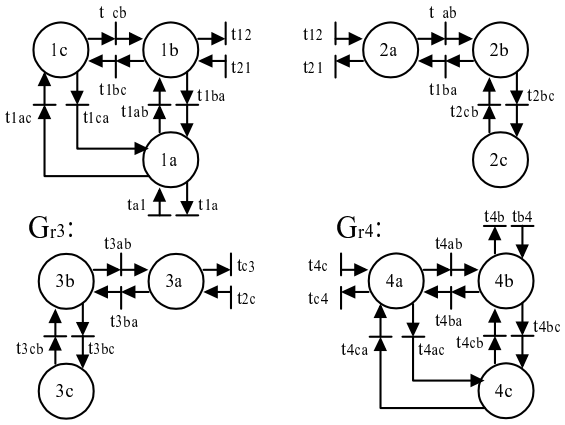

Figure 7: Maps of the rooms

5.1 Petri Nets Model

We model the mobile robot system as a Petri net. The Petri net consists of four sub Petri nets. The first Petri net $G_M = \{P_M,\ T_M,\ A_M,\ m_{M0}\}$ shows where the robot is (Fig. 6). The places mean the rooms, the hallways, the entrance and the exit. One token moves from a place to another, to show where the robot is. The place p_{Ri} $(i = 1, 2, 3, 4)$ means the room i and is refined by adding Petri nets $G_{ri} = \{P_{ri},\ T_{ri},\ A_{ri},\ m_{ri0}\}$ shown in Fig. 7. Let $P_R = \{p_{Ri}\}$, $P_r = \cup_i P_{ri}$, $T_r = \cup_i T_{ri}$, $A_r = \sum_i A_{ri}$.

Stepwise refinements are done when the robot enters a room. Suppose the event detector observes the firing of the transition $t \in T_M$. If there is a place $p \in P_R$ such that $A_M(t, p) > 0$, the place p is refined while, if there is a place $p \in P_R$ such that $A_M(p, t) > 0$, the place p is reduced. With these stepwise refinements, we can concentrate on the planning in the current room, and we can make a rough plan after exiting the room. Thus limited lookahead calculations become efficient.

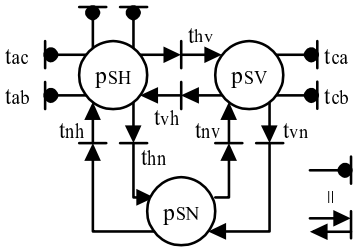

Figure 8: Signal Information

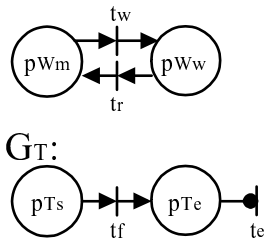

Figure 9: Task selection

The second Petri net G_S shows the information of the traffic signals. There are three places p_{SH}, p_{SV} and p_{SN}. The place p_{SH} means that the signal α is green and β is red, while the place p_{SV} means that the signal α is red and the signal β is green. The place p_{SN} means that the robot is not in the hallway and cannot watch the traffic signals (Fig. 8). There are permission arcs which are pairs of both input and output arcs from p_{SV} and p_{SH} to transitions in T_M in order to model the green signal.

The third Petri net G_W shows whether the robot is moving or stopping. The robot is allowed to wait for the traffic signals to change, and the transition t_w fires when the robot starts waiting (Fig. 9).

The last Petri net G_T shows the current task. There are two places p_{Ts} and p_{Te}. The place p_{Ts} means the robot has not found the object yet and is now searching, while the other place P_{Te} means the robot has found the object and is going to the exit. The transition t_f fires when the robot finds the object (Fig. 9).

The set of all controllable transitions is $T_M \cup T_r \cup T_W$, while $\{t \mid t \in T_S \cup T_T,\ t \notin T_M\}$ is the set of the

uncontrollable transitions. Thus the HLC selects a optimal transitions in $T_M \cup T_r \cup T_W$ that is executed by a low level controller.

5.2 Illegal Regions

When a room turns out to be a dead end, the robot need not enter the room again to find the object. Thus the HLC should give an illegal region to prevent the robot from entering the room again.

In order to detect a dead end, we prepare a map $E_0 : P_r \to \mathrm{N}$. We give initial values $E_0(p) = 0$ $(\forall p \in P_r)$ in advance, and update the value while the robot is moving. Suppose the event detector observes the firing of a transition $t_a \in T_r$. If there is a place $p_a \in P_r$ such that $A_r(p_a, t_a) > 0$, we define a set $\tilde{P}_a = \{p \mid p \in P_r,\ t \in T_r \setminus \{t_a\},\ A_r(p_a, t) > 0,\ A_r(t, p) > 0\}$. Then we update the value $E_0(p_a) = 1$ when $\tilde{P}_a = \emptyset$ or $E_0(p) = 1$ $(\forall p \in \tilde{P}_a)$. We define

$$E_{0R} : P_R \to \mathrm{N} : p_{Ri} \mapsto \min_{p \in P_{ri}} E_0(p) \tag{6}$$

and an illegal region $\{m \mid m \in M,\ \ p \in P_r,\ m(p) > 0,\ E_0(p) = 1\} \cup \{m \mid m \in M,\ p \in P_R,\ m(p) > 0,\ E_{0R}(p) = 1\}$.

5.3 Optimal Transition

We evaluate a trace $\sigma = t_{i_1} t_{i_2} \cdots t_{i_n}$ in order to find the optimal transition. Let $\bar{\sigma}_k = t_{i_1} \cdots t_{i_k}$ $(1 \le k \le n)$ be a prefix of σ and m the current marking. In order to calculate the possibility of finding the object, we define a map $E_1 : P_r \to \Re$, where $\Re$ is the set of all real numbers. We give initial values $E_1(p) = 1$ $(\forall p \in P_r)$ in advance, and update the value while the robot is moving. When the event detector observes the firing of a transition $t_b \in T_r$, if for each place $p_b \in P_r$ such that $A_r(p_b, t_b) > 0$, we update the value $E_1(p_b) = 0$. This implies that the possibility of the room which the robot has entered becomes zero.

If there is an integer k such that $m(\bar{\sigma}_k > m'$ and $m'(p) > 0$, we define $f_\sigma(p) = \min\{k \mid m(\bar{\sigma}_k > m',\ m'(p) > 0\}$, otherwise we define $f_\sigma(p) = \infty$. As an exception, we define $f_\sigma(p) = \infty$ if the place p is refined. We give a design parameter $0 < \alpha < 1$, which is called a discount rate. Then we evaluate the possibility of finding the object along σ as

$$J_1(\sigma) = \sum_{p \in P_r} E_1(p)\alpha^{f_\sigma(p)-1} + \sum_{p \in P_R} E_{1R}(p)\alpha^{f_\sigma(p)-1} \tag{7}$$

where

$$E_{1R}(p_{Ri}) = \frac{(1 + \alpha + \alpha^2 + \cdots + \alpha^{N_i - 1})}{N_i} \sum_{p \in P_{ri}} E_1(p) \tag{8}$$

and N_i is the number of the elements of P_{ri}. With discount rate α, we can give a better value if the robot enters a room sooner where the object is expected to be found. The possibility of a reduced room is not the simple sum of the refined parts but with average discount (8).

When the robot is far from the object and the possibilities E_1 around the robot becomes zero, it may need much time for the robot to leave there without other policies. In order to avoid such cases, we define a map $E_2 : P_r \to \Re$. We give initial values $E_2(p) = 0$ $(\forall p \in P_r)$ in advance, and update the value while the robot is moving. Let $p_0 \in P_R$ be the room where the robot is. If there are no pairs of a place $p \in P_r$ and a trace σ in the prediction tree such that $f_\sigma(p) < \infty$ and $E_1(p) > 0$, we update $E_2(p_0)$ as $E_2(p_0) - 1$ to show that there are no rooms with possibility around p_0. We give

$$J_2(\sigma) = \sum_{p \in P_r, f_\sigma(p) < \infty} E_2(p) + \sum_{p \in P_R, f_\sigma(p) < \infty} E_{2R}(p) \tag{9}$$

where $E_{2R}(p_{Ri}) = \sum_{p \in P_{ri}} E_2(p)$.

Thus we give a cost function of σ as

$$J(\sigma) = \begin{cases} J_1(\sigma) & (J_1(\sigma) > 0) \\ J_2(\sigma) & (J_1(\sigma) = 0) \end{cases} \tag{10}$$

The HLC searches for the optimal traces σ_M with (10) in the prediction tree, and if $\bar{\sigma}_{M1}$ is controllable, $\bar{\sigma}_{M1}$ is set to be enabled. If there are more than one optimal transition, choose one at random.

5.4 Numerical Simulation

We give a result of a numerical simulation. Let the discount rate $\alpha = 0.7$, the length of prediction tree $N = 4$. Suppose that the object is in the room 4 at the area 4_c.

We show a prediction tree at the initial state (Fig. 10). The robot first entered the room 1 because the trace $t_s t_{a1} t_{12}$ contained many rooms that had possibility to find the object. But the robot could not find the object along the trace, and left the rooms 1 and 2 by detecting the dead end. Then the robot tried to enter another room to find the object. The robot just happend to enter the room 4 while the both the rooms 3 and 4 were evaluated at the same score. After finding the object, it went to the exit. The whole trajectory is shown in Fig. 11.

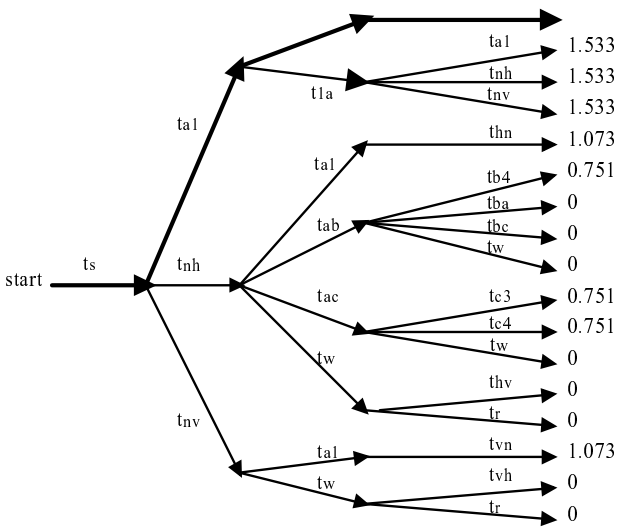

Figure 10: Tree at the initial state

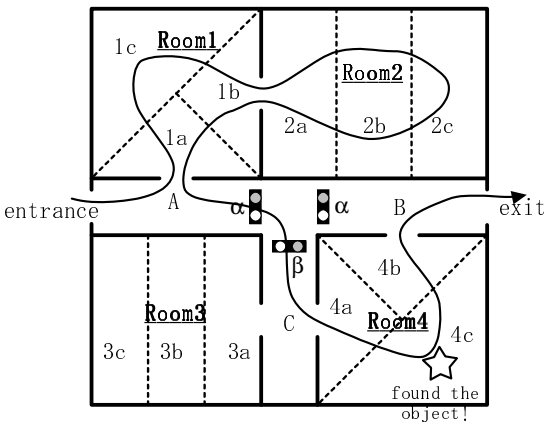

Figure 11: Trajectory

6 Conclusion

In this paper, we modeled robot systems by Petri nets, and proposed a high level controller based on an LLP supervisor. We considered a simple example of a mobile robot which searches for an object in a building. We gave several sub tasks to the robot and proposed some policies for LLP supervisory control.

We proposed a method for selecting a "qualitative" command to a mobile robot using LLP supervisory control. We calculated a prediction tree consisting of N-step projection of its qualitative behavior online. We proposed a method for dynamical stepwise refinement of the Petri net model according to the consciousness so that the effective calculation of the tree is achieved.

Acknowledgement

This work has been supported by CREST of JST (Japan Science and Technology).

References

[1] P. J. Ramadge and W. M. Wonam, "Supervisory Control of a Class of Discrete Event Processes," *SIAM Journal of Control and Optimization* vol. 25, no. 1, pp. 206–230 (1987).

[2] B. J. McCarragher and H. Asada, "The Discrete Event Modeling and Trajectory Planning of Robotic Assembly Tasks," *Trans. of the ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 117, pp. 394–400 (1995).

[3] G. E. Hovland, "Control of Sensory Perception for Discrete Event Systems," *A thesis submitted for the degree of Doctor of Philosophy of The Australian National University*, `http://www.dissertation.com/library/1120702a.htm` (1999).

[4] L. Bogoni and R. Bajcsy, "Functionality Investigation Using a Discrete Event System Approach," *Robotics and Autonomous Systems*, vol. 13, pp. 173–196 (1994).

[5] J. Košecká, H. Christensen, and R. Bajcsy "Experiments in Behaivior Composition," *Robotics and Autonomous Systems*, vol. 19, pp. 287–298 (1997).

[6] S. L. Ricker, N. Sarkar, and K. Rudie, "A Discrete-event Systems Approach to Modeling Dextrous Manipulation," *Robotica*, vol. 14, pp. 515–525 (1996).

[7] S. -L. Chung, S. Lafortune, and F. Lin, "Limited Lookahead Policies in Supervisory Control of Discrete Event Systems," *IEEE Trans. Automatic Control*, vol. 37, no. 12, pp. 1921–1935 (1992).

[8] I. Suzuki and T. Murata, "A Method for Stepwise Refinement and Abstraction of Petri Nets," *Journal of Computer and System Sciences*, vol. 27, no. 1, pp. 51–76 (1983) .