LLP Supervisory Control with Timed Petri Net Models in Mobile Robots

Keigo Kobayashi, Kengo Inoue 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 timed Petri net and give a high level controller which consists of an LLP supervisor. 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]–[4]. 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)[5] 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 have described the consciousness by stepwise refinement[6] of Petri nets in [7]. We introduce timed Petri nets[8], which is also colored Petri net[9]. The tokens have color to shows which transitions fired to put them to the places, and the frozen times of the tokens depend on the color. We can exploit the rule to get more proper supervisory controllers.

Proc. of the 2001 IEEE International Conference on Systems, Man & Cybernetics, 3229/3234 (2001)

2 Petri Nets

2.1 Mathematical Definition

We give a definition of timed colored Petri nets in this paper. In order to simplify the notation, we assume the Petri net is safe, namely, the number of tokens in one place is guaranteed to be either zero or one. We shall use $\mathcal{B}$ to denote the set $\{0, 1\}$ and $\Re$ the set of all real numbers. Let P be a set of places, T a set of transitions, C a set of colors. We assume that a color of a token depends on a transition whose firing put the token in a place. carries the token and define $F : T \to C$ is a color function. The marking $m : P \to \mathcal{B} \times C \times \Re$ is denoted by a triple $m = (m_n, m_c, m_t)$, where $m_n : P \to \mathcal{B}$ is the number of tokens in a place, $m_c : P \to C$ is a color of the token in the place, $m_t : P \to \Re$ is the time when the token is put at the place. If there are no tokens in a place, m_c and m_t are undefined for the place. The set of all markings is denoted by M. The connection $A : (P \times T) \cup (T \times P) \to \mathcal{B}$ describes input and output arcs of a transition. We introduce a delay function $\theta : C \times P \times T \to \Re^+$, where $\Re^+$ denotes the set of all non-negative real numbers. Then the timed colored Petri net is defined as $G = (C,\ P,\ T,\ A,\ F,\ \theta,\ m_0)$, where $m_0 \in M$ is an initial marking.

Let $m \in M$ and $t \in T$. A transition t is said to be enabled at time τ if

$$m_n(p)\phi(\tau - m_t(p) - \theta(m_c(p), p, t)) \geq A(p, t) \quad (\forall p \in P) \tag{1}$$

where

$$\phi(x) = \begin{cases} 0 & (x < 0) \\ 1 & (x \geq 0) \end{cases} \tag{2}$$

and we write $m[t, \tau\rangle$. If the enabled transition t fires, the marking changes to $m' = (m'_n, m'_c, m'_t)$ given by

$$m'_n(p) \quad = \quad m_n(p) - A(p, t) + A(t, p) \tag{3}$$

$$m'_c(p) \quad = \quad \begin{cases} F(t) & (A(t, p) > A(p, t)) \\ m_c(p) & (A(t, p) = A(p, t)) \end{cases} \tag{4}$$

$$m'_t(p) \quad = \quad \begin{cases} \tau & (A(t,p) > A(p,t)) \\ m_t(p) & (A(t,p) = A(p,t)) \end{cases} \qquad (5)$$

and we write $m[t,\tau\rangle m'$.

Equation (1) shows that the token in place p is not available for firing of a transition t until $\theta(m_c(p),p,t)$ time passes from its entering time $m_t(p)$, and the time when the token becomes available again depends on the color of the token.

Equation (4) implies that the color of the token is assigned by the transition, namely, if the color of the token $m_{0c}(p) = a$, the transition t_p such that $F(t_p) = a$ fired. So the maps θ and F can model how long it takes from firing of t_p to that of t through p.

If $A(t,p) = A(p,t) = 1$, there exists a pair of arcs $t \to p$ and $p \to t$ that is equivalent to a permission arc from p to t. Since the token in p doesn't move by firing of t, we don't update m_c and m_t in this case. When $A(t,p) < A(p,t)$, $m'_n(p)$ becomes 0 and $m'_c(p)$ and $m'_t(p)$ are undefined.

When $m_n(p) \geq A(p,t)$ $(\forall p \in P)$, there exists a $\tau_N \geq 0$ such that $m[t,\tau_N\rangle$. Then we write $m[t\rangle$. Assume $m_p[t_p,\tau_p\rangle m_0$, then $\tau_N \geq \tau_p$ is given by

$$\tau_N = \max_{A(p,t)>0} \{m_t(p) + \theta(m_c(p),p,t)\} \quad . \qquad (6)$$

2.2 Composing Petri Nets

Let $G_1 = (C_1,\ P_1,\ T_1,\ A_1,\ F_1,\ \theta_1,\ m_{10})$ and $G_2 = (C_2,\ P_2,\ T_2,\ A_2,\ F_2,\ \theta_2,\ m_{20})$ 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 $C_{12} = C_1 \cup C_2$, $P_{12} = P_1 \cup P_2$, and $T_{12} = T_1 \cup T_2$. Then we define a marking $m_0 : P_{12} \to \mathcal{B} \times C_{12} \times \Re$ such that

$$m_{120}(p) = \begin{cases} m_{10}(p) & (p \in P_1) \\ m_{20}(p) & (p \in P_2) \end{cases} \quad . \qquad (7)$$

We also define $A_{12} : (P_{12} \times T_{12}) \cup (T_{12} \times P_{12}) \to \mathcal{B}$ such that

$$A_{12}(p,t) = \begin{cases} A_1(p,t) & (p \in P_1 \wedge t \in T_1) \\ A_2(p,t) & (p \in P_2 \wedge t \in T_2) \\ 0 & (\text{otherwise}) \end{cases} \qquad (8)$$

and

$$A_{12}(t,p) = \begin{cases} A_1(t,p) & (p \in P_1 \wedge t \in T_1) \\ A_2(t,p) & (p \in P_2 \wedge t \in T_2) \\ 0 & (\text{otherwise}) \end{cases} \quad . \qquad (9)$$

Now we can get a composition $G_1 + G_2 = (C_{12},\ P_{12},\ T_{12},\ A_{12},\ F_{12},\ \theta_{12},\ m_{120})$ where

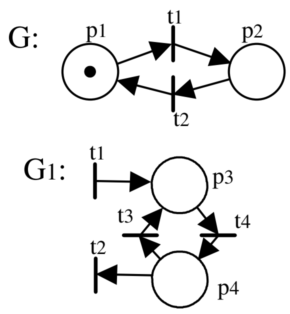

Figure 1: Petri nets

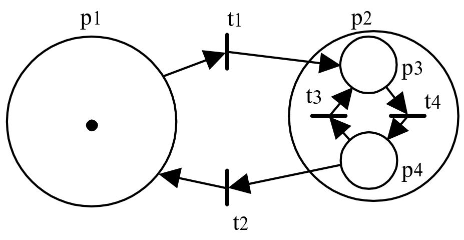

Figure 2: Refinement of place p_2

$$\theta_{12}(c,p,t) = \begin{cases} \theta_1(c,p,t) & (c \in C_1 \wedge p \in P_1 \wedge t \in T_1) \\ \theta_2(c,p,t) & (c \in C_2 \wedge p \in P_2 \wedge t \in T_2) \\ 0 & (\text{otherwise}) \end{cases}$$

and

$$F_{12}(p) = \begin{cases} F_1(p) & (p \in P_1) \\ F_2(p) & (p \in P_2) \end{cases} \quad . \qquad (10)$$

Let $t_a \in T_1 \cap T_2$. We note that $m_{120}[t_a\rangle$ if and only if $m_{10}[t_a\rangle$ and $m_{20}[t_a\rangle$.

We give a simple example of composing two Petri nets. There are two Petri nets G and G_1 in Fig.1. 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 tokens move when the transition fires.

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.

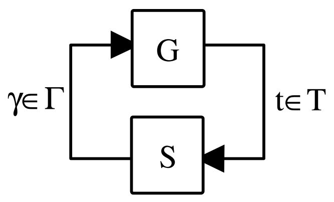

Figure 3: Supervisor

For valid refinement, It is assumed that

$$\sum_{p \in P_1} m_{10n}(p) = m_{0n}(p_2) \quad . \tag{11}$$

3 Supervisory Control

3.1 Supervisor for DES

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 $\gamma : T_c \rightarrow \{0,\ 1\}$ 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 discrete event system 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 $S : L(G) \rightarrow \Gamma$. A supervisor selects a control pattern according to firing of transitions so that every *trace*(a sequence of transitions) in the controlled discrete event system is restricted in a set of desirable traces. Shown in Fig. 3 is a diagram of the discrete event system 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.

3.2 LLP Supervisor for Timed Petri Nets

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)[5]. An LLP supervisor for timed Petri net predicts all possible traces with limited length and calculate minimum times to fire all transitions in the traces sequentially. These traces form a tree and each node has a time to reach them. Then the LLP supervisor evaluates each branch and finds the optimal traces. The control pattern given by the supervisor consists of the controllable transitions which are the first transitions in the optimal traces. When the supervisor observes firing of a transition, it updates the tree and reselects a control pattern.

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.

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. Each node in the tree has the marking and the minimum time to reach there.

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.

4 Application to a Mobile Robot

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.

Figure 4 shows the block diagram of the system consisting 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 sys-

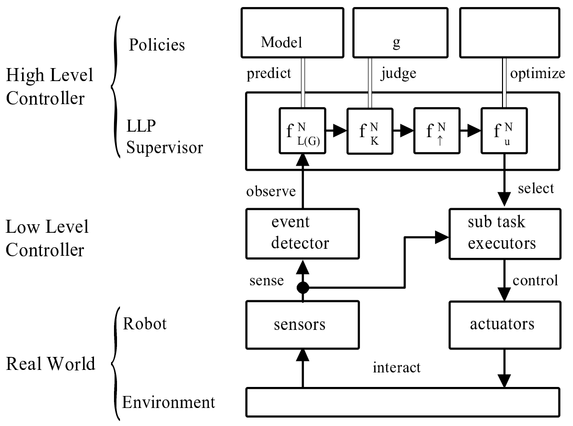

Figure 4: System

tem such as maps, qualitative states of robots, and so on. The high level controller (HLC) is based on an LLP supervisor. 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.

4.2 Mobile Robot

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

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

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 3, hallways A and B. It is 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.

A target 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 it in the building, and to exit from the building after finding it.

4.3 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 $G_M = (C_M,\ P_M,\ T_M,\ A_M,\ F_M,\ \theta_M,\ m_{M0})$

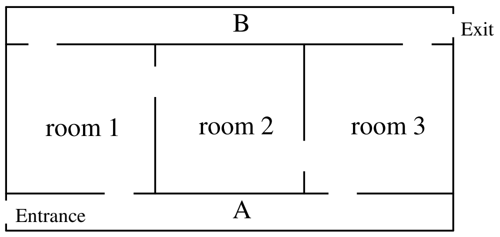

Figure 5: Map of a building

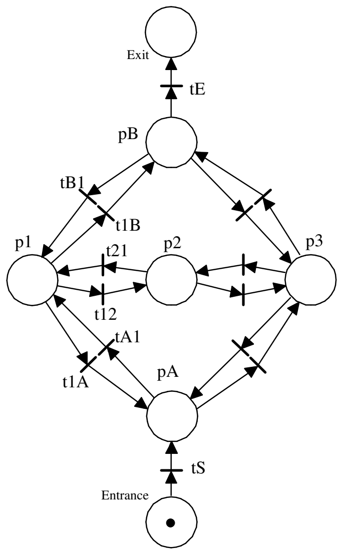

Figure 6: Map of the building

shows where the robot is (Fig. 6). We also let $C_M = T_M$ and $F(t) = t$. The delay function θ_M has geometric information of the room.

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.

Geometric information is modeled by θ_M as Fig. 7.

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. 8).

The set of all controllable transitions is T_M, while T_T is the set of the uncontrollable transitions. Thus the HLC selects an optimal transition in T_M that is executed by a low level controller.

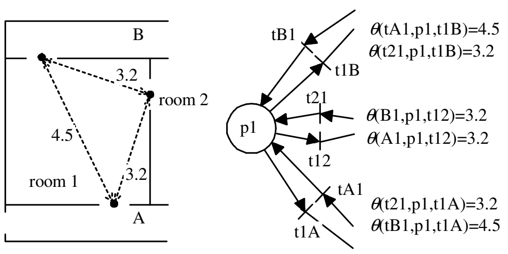

Figure 7: Delay in the room

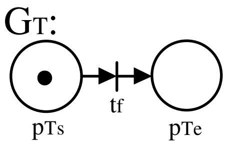

Figure 8: Task

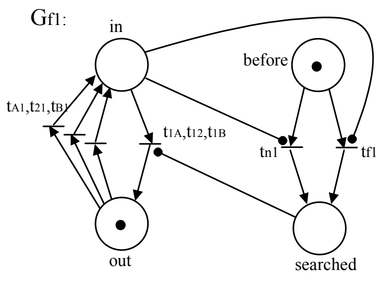

Figure 9: G_{f1}

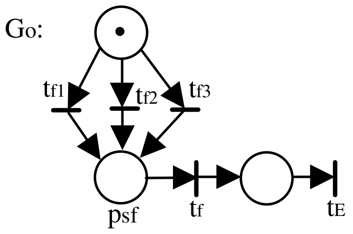

Figure 10: G_o

4.4 Getting Optimal Transition

Before giving the evaluative function, we add some Petri nets which shows when the transition t_f can occurs. The transition t_f shows the robot finds the object and is the sub goal to achieve the task.

When the robot enters the room 1 for the first time, it looks around the room in order to find the target object. If there is the object, the transition t_{f1} fires. But if there is not the object, the transition t_{n1} fires. Both transitions are uncontrollable. This is given by G_{f1} shown in Fig. 9, where a pair of input and output arcs is represented by permission arc "—•".

By G_o, we indicate that t_{fi} $(i = 1, 2, 3)$ must fire before t_f fires. We also indicate the t_E never fire before t_f fire. Before t_f fires, we let p_{sf} sub goal state and terminate the tree there. Since we cannot make t_{fi} fire, a purpose of the high level controller is to find the trace along which the transition t_{fi} may fire. Thus the trace which has many t_{ni}-s should be evaluated much and the shorter is the time to fire, the more the evaluation. For example, trace $\sigma = t_{A1}t_{n1}t_{12}t_{n2}$ in Fig. 11 has the opportunity for firing of t_{f1} and t_{f2}. With the current marking and θ, we can also calculate the minimum times τ_{f1} and τ_{f2} for t_{f1} and t_{f2} to be enabled, respectively. Then we give a cost function of σ as

$$J(\sigma) = \sum_{t \in \{t_{fi}, t_{ni}\}} \exp(-\lambda f_\sigma(t)) \quad , \qquad (12)$$

where λ is proper positive discounted rate and $f_\sigma(t)$ is the shortest time for t to fire. If there is no t in σ, $f_\sigma(t) = \infty$. The HLC searches for the optimal traces σ_M by (12) in the prediction tree, and if the first transition of σ_M is controllable, the transition is set to be enabled. If there are more than one optimal transition, choose one at random.

4.5 Numerical Simulation

We give a result of a numerical simulation. Let the discounted rate $\alpha = 0.1$, the length of prediction tree $N = 5$. Suppose that the target object is in the room 3. To simulate the case that the given map is not correct, when the robot enters the room 2 or 3, the transitions t_{23} and t_{32} are removed from G_M to model the door between the rooms is not available.

We show a prediction tree at the initial state (Fig. 12). The robot first entered the room 1 because the trace $t_A t_{A1} t_{n1} t_{12} t_{f2}$ was the shortest to fire and contained many rooms that had possibility to find the object. But the robot could not find the object along the trace, and found the rooms 2 and 3 were not connected directly. The robot selected the hallway A because the path was the shortest. After finding the object, it went to the exit. The whole trajectory is shown in Fig. 13.

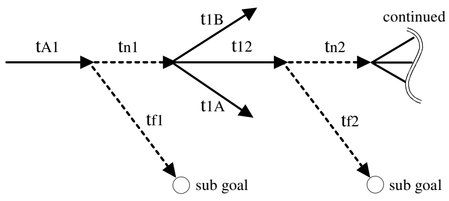

Figure 11: Evaluation of trace

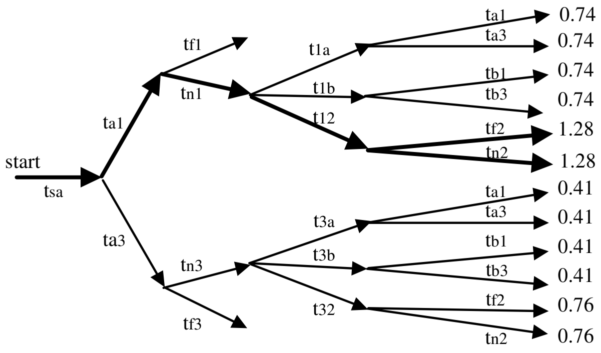

Figure 12: Tree at the initial state

5 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 to use timed Petri net which can model the minimum time to execute the sequence of the transition. We gave an example to model geometrically information. This method is also useful with stepwise refinement of the Petri net since we can distinguish the time cost of the transition in a refined place and the one among the refined places. We gave an example of updating the Petri net model on line, and our method was shown to be effective in such cases.

Acknowledgement

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

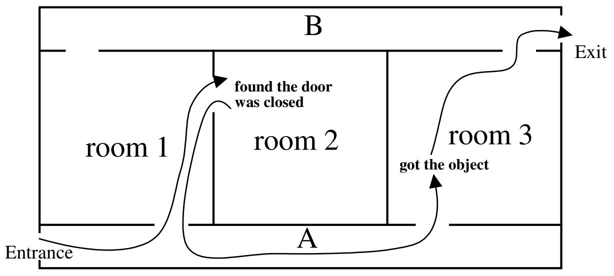

Figure 13: Trajectory

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

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

[5] 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).

[6] 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) .

[7] K. Kobayashi, T. Ushio, "An Application of LLP Supervisory Control with Petri Net Models in Mobile Robots", *Proc. of the 2000 IEEE International Conference on Systems, Man & Cybernetics*, 3015–3020 (2000).

[8] J. Wang, "Timed Petri Nets Theory and Application",*Kluwer Academic Pub.* (1998)

[9] K. Jensen, "Coloured Petri Nets", vol. 1,2,3, *Springer* (1997)