Motion Planning for Humanoid Robots Using Timed Petri Net and Modular State Net

Keigo KOBAYASHI, Atsuhito NAKATANI, Hideyuki TAKAHASHI, and Toshimitsu USHIO

Osaka University
Machikaneyama 1-3, Toyonaka, Osaka, 560-0054, JAPAN
{kobayasi,ushio}@sys.es.osaka-u.ac.jp, {nakatani,takahasi}@hopf.sys.es.osaka-u.ac.jp

Abstract— **In this paper, we propose a supervisory control system for motion planning of humanoid robots. The proposed system is hierarchically structured into two levels. The lower level controls and monitors the robots using modular state nets. The upper level generates an optimal sequence of motion for user's requirements using timed Petri nets.**

Keywords— **humanoid robot, modular state net, timed Petri net, supervisory control**

I. Introduction

Recently, there have been many researches for motion planning of Humanoid robots. In order to control behaviors of humanoid robots, a state net architecture has been proposed by Kanehiro et al.[1]. Since the action space is represented as a state transition graph which is embedded in a sensory space, the architecture has two advantages: easiness of incremental extensions and integrations, and an autonomous error recovery. But, in humanoid robots, there are many sensors so that the dimension of the state space becomes large.

In this paper, we introduce a modular state net which is a state net representing behaviors of a part of the robots such as arms, legs, and so on, and behaviors of the robots are represented by a combination of modular state nets for those parts. To have a feasible path of the whole body, we introduce a timed Petri net as an abstracted model of the set of all modular state nets. We give an example to model a humanoid robot HOAP-1 and show simulation and experiment results of signaling with flags.

II. Modular State Net

In order to control a humanoid robot, a state net architecture is proposed as a state transition graph which is embedded in a sensory space as shown in Fig. 1. A point p in the sensory space has a coordinate $p = [s_1\ s_2\ \cdots\ s_N]$ where each s_i is sensory information such as a joint angle. So a node in the state net shows a stationary state of the humanoid robot. An action is defined as a timed trajectory $p(t)$ in the sensory space which starts at a node and ends at another node. In the state net architecture, we can update the network by adding new nodes or arcs. We can also detect an occurrence of an error by comparing the current sensory information and a coordinate of an executing action. But some humanoid robots have many sensors and the number of the dimension of the state becomes large and it becomes difficult to prepare enough actions to give various motions. So we decide the state vector $p = [s_1\ s_2\ \cdots\ s_N]$ into sub vectors such as $p = [p_1\ p_2\ \cdots p_M] = [s_1 \cdots s_{i_1} | s_{i_1+1} \cdots s_{i_2} | \cdots | s_{i_{M-1}+1} \cdots s_N]$. The order of elements of p and the division is designed such that each p_j shows a part of the body of the humanoid robot, such as arms, legs, and so on. Thus we introduce a state net for each sub space, which will be called a modular state net in the following.

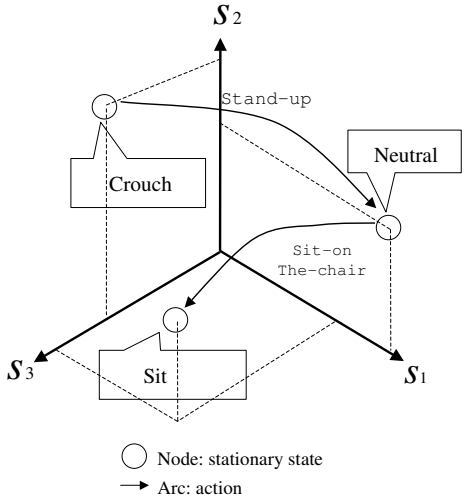

Fig. 1. StateNet

In a modular state net architecture, we can design motions of a part of the humanoid robot individually, and a motion of the whole body is given as a combination of actions of modular state nets. Since the number of the combinations is large, various actions are expected to be generated. But all of the combinations are not feasible since some of them may cause a conflict or may not satisfy a dynamic constraint. In order to find a feasible path, we introduce an abstracted model of the set of all modular state nets, where no sensory information appears explicitly.

III. Timed Petri Net

We introduce a timed Petri net to model the set of all modular state nets. The timed Petri net is given by 6-tuple $G = (P, T, A, F, \theta, m_0)$. We decompose the place

Proc. of the 2002 IEEE International Conference on Systems, Man & Cybernetics, 334/339 (2002)

set P into two subsets P_s and P_d, where $P = P_s \cup P_d$ and $P_s \cap P_d = \emptyset$. A place in P_s means a node in one of the modular state net, a stationary state, while a place in P_d means an arc in one of the modular state net, an executing of an action. We also decompose the transition set T into two subsets T_s and T_e, where $T = T_s \cup T_e$ and $T_s \cap T_e = \emptyset$. Firing of a transition in T_s is a start of an action, while firing of a transition in T_e means completion of an action. All transitions in T_s are controllable since we can prevent them from firing. But all transitions in T_e are uncontrollable since the actions are timed trajectories and they must reach ending state in a certain time. $A : (P \times T) \cup (T \times P) \rightarrow N$ is a set of arcs. $F : P \times T \rightarrow N$ is a set of inhibitor arcs. The time from start to completion of an action is modeled by $\theta : P_d \rightarrow \Re$. The initial marking is given by $m_0 : P \rightarrow N$.

A transition $t \in T$ is said to be enabled if

$$\forall p \in P : m(p) \geq A(p,t) \tag{1}$$

and

$$\forall p \in P : F(p,t) = 0 \vee m(p) = 0 \quad , \tag{2}$$

and we write $m[t >$. A trace is defined as a sequence of transitions. when $\sigma = t_1 t_2 \cdots t_n$ satisfies $m_0[t_1 > m_1[t_2 > \cdots m_{n-1}[t_n >$, we write $m_0[\sigma >$.

If the enabled transition t fires, the marking changes to m' given by

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

and we write $m[t > m'$. If a token comes into a place $p \in P_d$ at time τ, the token is not available for firing until $\tau + \theta(p)$, and usually an uncontrollable transition $t \in T_e$ that have an input arc from p fires at $\tau + \theta(p)$.

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

Inhibitor arcs are used to avoid sequential or parallel combinations of actions in different modular state nets. There are two kinds of connections of inhibitor arcs. One is from a static place in P_s to a transition in T_s that implies an infeasible sequential combination. In the example in Fig. 2, the left arm cannot start moving until the right arm starts moving. The other is from a dynamic place in P_d to a transition in T_s that implies an infeasible parallel combination. In the example in Fig. 3, the left arm cannot start moving until the right arm finishes moving. Some transition may have inhibitor arcs from both a static place and a following dynamic place.

However a trace $\sigma = t^l_{s12} t^r_{s12} t^r_{e12} t^l_{e12}$ may satisfy $m_0[\sigma >$, σ is not feasible when $\theta(p^l_{d12}) < \theta(p^r_{d12})$. Because if we let the timed trace as (t^l_{s12}, τ_1) (t^r_{s12}, τ_2) (t^r_{e12}, τ_3) (t^l_{e12}, τ_4), then $\tau_4 = \tau_1 + \theta(p^l_{d12})$ and $\tau_3 = \tau_2 + \theta(p^r_{d12})$ contradict $\tau_1 \leq \tau_2 \leq \tau_3 \leq \tau_4$. If a trace is feasible in this sense, the minimum time to execute the transition is defined. When a timed Petri net $G = (P, T, A, F, \theta, m_0)$ is safe, namely, $m(p) \leq 1$ $(\forall p \in P, m_0[\forall \sigma > m)$, feasibility and minimum time are calculated in the following way. First we define a timed marking with $m : P \rightarrow \mathcal{B}$ and $\psi : P \rightarrow \Re^+$, where $\mathcal{B} = \{0, 1\}$ and $\Re^+$ is the set of all nonnegative real numbers. If $m(p) = 1$, $\psi(p)$ means when the token comes into p. Thus if a transition t fires at time τ, ψ is updated as follows:

$$\psi'(p) = \begin{cases} \tau & (A(t,p) > A(p,t)) \\ \psi(p) & (A(t,p) = A(p,t)) \end{cases} \quad . \tag{4}$$

We write (3) and (4) as $(m, \psi)[(t, \tau) > (m', \psi')$. Then a transition $t \in T_e$ is feasible when $m[t >$ and for p_a such that $A(p_a, t) > 0$,

$$\max_{p \in P} \psi(p) < \psi(p_a) + \theta(p_a) \tag{5}$$

and the minimum time for firing is given as

$$\tau_{\min}(m, \psi, t) = \psi(p_a) + \theta(p_a) \quad (A(p_a, t) > 0). \tag{6}$$

If $t \in T_s$, $\tau_{\min}(m, \psi, t) = \max_{p \in P} \psi(p)$. Repetition of (4)–(6) gives a feasibility check and the minimum executing time for a trace $\sigma = t_1 t_2 \cdots t_n$. When it holds that $\tau_i = \tau_{\min}(m_{i-1}, \psi_{i-1}, t_i)$ and $(m_{i-1}, \psi_{i-1})[(t_i, \tau_i) > (m_i, \psi_i)$ $(i = 1, 2, \cdots, n)$, σ is said to be feasible at (m_0, ψ_0). If σ is feasible, $\tau_{\min}(m_0, \psi_0, \sigma) = \tau_n$ is defined. Assuming $\psi_0(p) = 0$ $(\forall p \in P)$, we also define $\tau_{\min}(m_0, \sigma) = \tau_n$. The set of all feasible trace is denoted by $L_F(m_0) \subset T^*$, where T^* is the Kleene closure of T.

If a timed marking (m, ψ) has a place $p_a \in P_d$ such that $m(p_a) = 1$ and

$$\max_{p \in P} \psi(p) > \psi(p_a) + \theta(p_a) \quad , \tag{7}$$

then (m, ψ) is an error state and a supervisory controller must avoid such states.

IV. Optimal Path Planning

We use a kind of limited lookahead policy supervisory controller[4] to find an optimal path from the current state to any given desired state. Calculating a tree consists of all possible path with finite length, we search for a path from the current state to the desired state. We enlarge the length of the tree one by one to find optimal paths that have shortest time to execute.

Let a goal marking m_G. We search for an optimal trace $\sigma_o \in L_F(m_0)$ such that $m_0[\sigma_o > m_G$ and

$$\tau_{\min}(m_0, \sigma_o) \leq \tau_{\min}(m_0, \sigma) \quad (m_0[\forall \sigma > m_G) \quad . \tag{8}$$

A set of feasible traces with length N is denoted by S_N. Searching the goal marking m_G, we have $R_N \subset S_N$ as a set of σ such that $m_0[\sigma > m_G$ and $\sigma \in S_N$. The set $\cup_{i \leq N} S_i$ gives N-length tree and we let G_N be the set of optimal traces to reach goal in the N-length tree. The minimum time to reach goal in the tree is also denoted by τ_N. Then the following inductive algorithm gives the set of all optimal paths.

1. Let $S_0 = \{\varepsilon\}$. $R_0 = \emptyset$, $G_0 = \emptyset$, $\tau_0 = \infty$, $N = 1$.

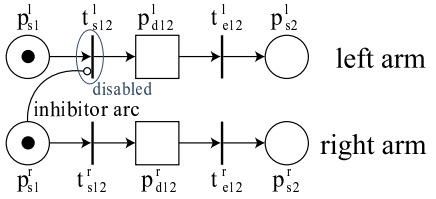

Fig. 2. Inhibitor arc from static place

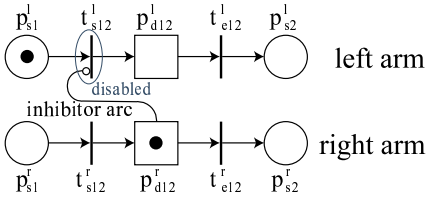

Fig. 3. Inhibitor arc from dynamic place

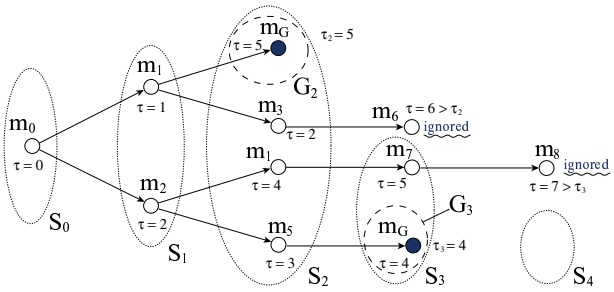

Fig. 4. Tree for finding the optimal traces

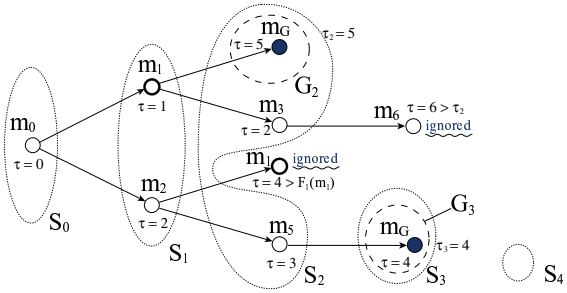

Fig. 5. Tree by improved algorithm

2. Let $S_N = \{\sigma' | \sigma' = \sigma t \in L_F(m_0), t \in T, \sigma \in S_{N-1} \backslash R_{N-1}, \tau_{\min}(m_0, \sigma') < \tau_{N-1}\}$ and $R_N = \{\sigma | \sigma \in S_N, m_0[\sigma > m_G\}$.
3. $\tau_N = \min\{\tau_{N-1}, \min_{\sigma \in R_N} \tau_{\min}(m_0, \sigma)\}$.
4. If $\tau_N < \tau_N - 1$ then $G_N = \{\sigma | \sigma \in R_N, \tau_{\min}(m_0, \sigma) = \tau_N\}$ else $G_N = G_{N-1} \cup \{\sigma | \sigma \in R_N, \tau_{\min}(m_0, \sigma) = \tau_N\}$.
5. If $S_N \neq \emptyset$ then $N = N + 1$ and go back to 2. otherwise G_N is the set of optimal traces and τ_N is the minimum time.

In 2. the tree is enlarged by one step while the goal nodes are terminated and the infeasible nodes are ignored. If an executing time is turned to be over the optimal time, the node is also ignored. In 3. the optimal time is updated if there exists a new goal node whose time is less than previous optimal time. In 4. if the optimal time is updated, G_N is replaced by the new goal nodes that have new optimal time, otherwise new goal nodes that have the same time is added to G_N. Since any transition $t \in T_e$ needs positive time for firing, this algorithm is over in finite steps. An example of this algorithm is illustrated in Fig. 4, where m_i is a marking of a node, and τ is the minimum time to reach a node.

To reduce calculating cost, we give another algorithm. Suppose there are two trace σ_1 and σ_2 such that $m_0[\sigma_1 > m_a$, $m_0[\sigma_2 > m_a$ and $\tau_{\min}(m_0, \sigma_1) < \tau_{\min}(m_0, \sigma_2)$. The branch following σ_2 is less expected to have optimal path than the branch following σ_1. Terminating the node σ_2, we can cut much cost although we may have only semi-optimal solutions.

We introduce a set of marking M_n which consists of all markings in the tree $\cup_{i \leq N} S_N$ and a map $F_n : M_n \rightarrow \Re^+ \cup \{0\}$ which shows the minimum time to reach a marking. Updating 2. and 5., we give improved algorithm as follows:

1. Let $S_0 = \{\varepsilon\}$. $R_0 = \emptyset$, $G_0 = \emptyset$, $\tau_0 = \infty$, $M_0 = \emptyset$, $N = 1$.
2. Let $S_N = \{\sigma' | \sigma' = \sigma t \in L_F(m_0), t \in T, \sigma \in S_{N-1} \backslash R_{N-1}, \tau_{\min}(m_0, \sigma') < \tau_{N-1}, (m_0[\sigma' > m' \in M_{N-1} \Rightarrow \tau_{\min}(m_0, \sigma') \leq F_{n-1}(m'))\}$ and $R_N = \{\sigma | \sigma \in S_N, m_0[\sigma > m_G\}$.
3. $\tau_N = \min\{\tau_{N-1}, \min_{\sigma \in R_N} \tau_{\min}(m_0, \sigma)\}$.
4. If $\tau_N < \tau_N - 1$ then $G_N = \{\sigma | \sigma \in R_N, \tau_{\min}(m_0, \sigma) = \tau_N\}$ else $G_N = G_{N-1} \cup \{\sigma | \sigma \in R_N, \tau_{\min}(m_0, \sigma) = \tau_N\}$.
5. Let $M_N = M_{N-1} \cup \{m | \sigma \in S_N, m_0[\sigma > m\}$ and for all m such that there exists $\sigma \in S_N$ and $m_0[\sigma > m$, let $F_N(m) = \tau_{\min}(m_0, \sigma)$. We also let for all $m \in M_N \setminus \{m | \sigma \in S_N, m_0[\sigma > m\}$, $F_N(m) = F_{N-1}(m)$.
6. If $S_N \neq \emptyset$ then $N = N + 1$ and go back to 2. otherwise G_N is the set of optimal traces and τ_N is the minimum time.

In 2. we ignore a node whose marking (m') is already appeared in the tree (i.e. $m' \in M_{N-1}$) and whose executing time ($\tau_{\min}(m_0, \sigma')$) is grater than the minimum reaching time of the marking ($F_{N-1}(m')$). Since M_N contains the markings of all nodes in the tree, we add the markings of all new nodes in 5. Since new node has a marking that is not an entry of M_{N-1} or that has the minimum executing time for the marking, F_N is updated for all new markings. An example of this algorithm is illustrated in Fig. 5, where a node with marking m_1 is cut to reduce calculating cost.

Eq. (4) shows the minimum executing time is not depend on a marking m but on a timed marking (m, ψ) and this is the reason why this algorithm may overlook some optimal traces. But this influence is usually small and we can expect reasonable solutions.

V. Total System

We propose a system to select an optimal motion of a humanoid robot for a given requirement. As shown in Fig. 6, the system is hierarchically structured. The

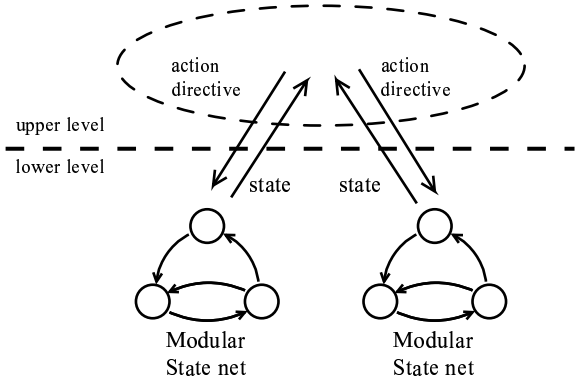

Fig. 6. Hierarchical structure of the proposed System

lower level consists of modular state nets. Each modular state net monitors and controls trajectories of parts of the humanoid robot. In the higher level, sequential or parallel combinations of the actions are planned to optimize the trajectory of the whole body.

The proposed system is shown in detail in Fig. 7. There is a timed Petri net that models the modular state nets in **path planning**. A planned path is checked by **feasibility check** and **collision check**. If they detected a collision for executing the planned path, the time Petri net is updated by adding an inhibitor arc to avoid planning the path again. Otherwise the planned path is sent to the **command queue**. The commands are sent one by one to the corresponding modular state net by **command server**. If a starting event is sent to a **modular state net**, the corresponding action starts. Once an action starts, the modular state net refuses to recieve the corresponding ending event until the action is over. The result of executing the commands are detected by **event translator**.

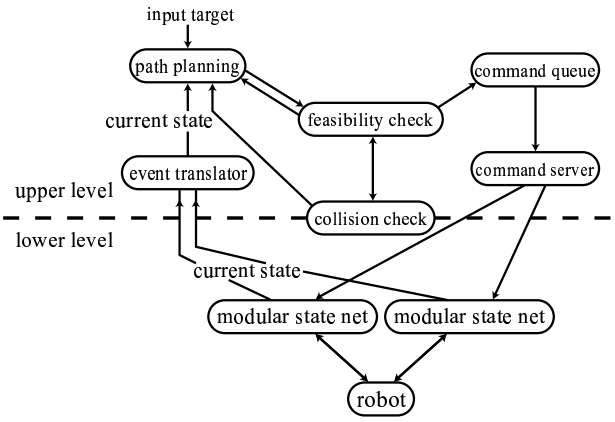

Fig. 7. Control system

VI. Simulation Result

To show the effectiveness of our method, we show an simulation result. A model of a humanoid robot is designed for a humanoid robot HOAP-1 (Fujitsu Automation Company) whose height is 0.438[m], weight is 5.67[kg], number of joints is 20 (each arm has 4 joints). The task for the robot is a simple example of signaling with flags, that is to find an optimal path from any given initial state to any given goal state and to control the robot along the path.

First, we make modular state nets for both arms, that are shown in Figs. 8,9. The Nets for right and left arms have states s_0^r to s_7^r and s_0^l to s_8^l, respectively. The arc from s_i^r to s_j^r is denoted by a_{ij}^r, and so on. The executing time of each arc is shown in Tables. I, II. These state nets are abstracted by a

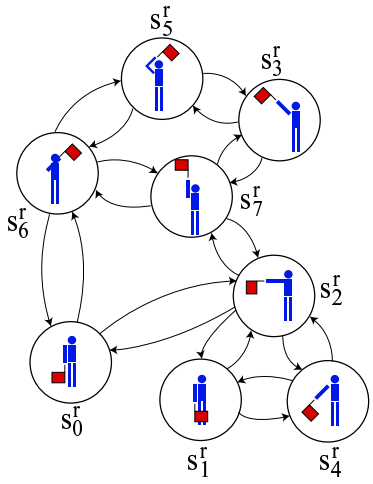

Fig. 8. Module of right arm

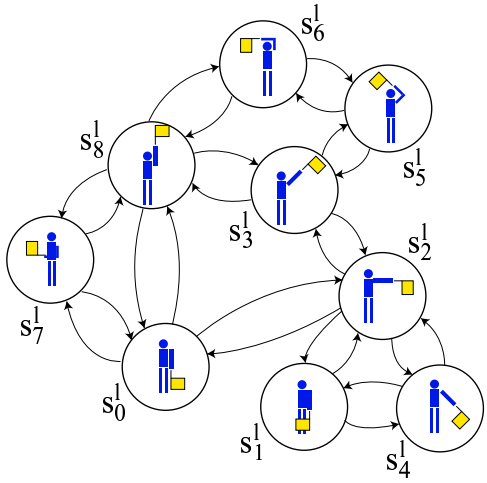

Fig. 9. Module of left arm

timed Petri net. The set of static place is given by $P_s = \{p_{si}^r, p_{sj}^l | i = 0, 1, \cdots, 7, j = 0, 1, \cdots, 8\}$. The places p_{si}^r and p_{sj}^l means the nodes s_i^r and s_j^l, respectively. The set of dynamic place is given by $P_d = \{p_{dij}^r, p_{dkl}^l | (i, j) = 0, 1, \cdots, 7, (k, l) = 0, 1, \cdots, 8\}$. The places p_{dij}^r and p_{dij}^l means the arcs a_{ij}^r and a_{ij}^l, respectively. The sets of transitions are denoted by $T_s =$

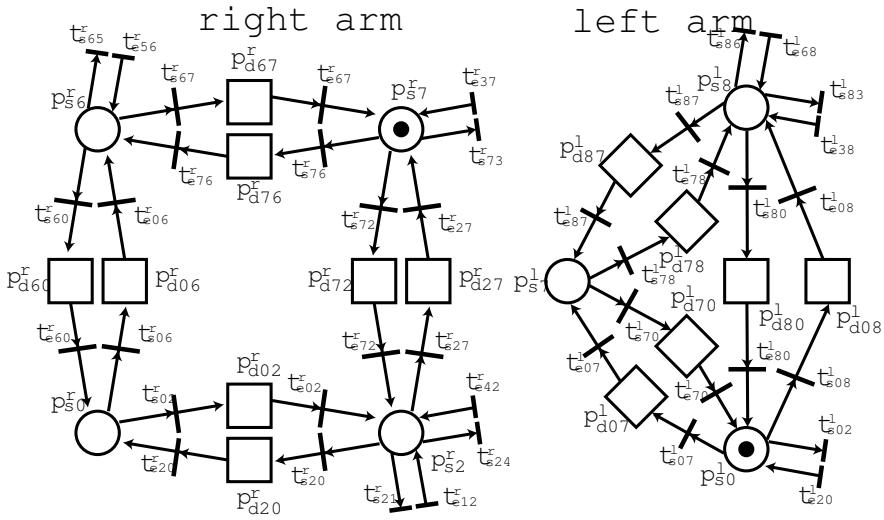

Fig. 10. A part of Petri net model

TABLE I
EXECUTION TIME OF RIGHT ARC

arc	a^r_{02}	a^r_{06}	a^r_{12}	a^r_{14}	a^r_{24}
time[s]	4.06	4.09	3.88	1.65	3.60
arc	a^r_{27}	a^r_{35}	a^r_{37}	a^r_{56}	a^r_{67}
time[s]	4.06	2.60	1.60	2.03	2.64

TABLE II
EXECUTION TIME OF LEFT ARC

alc	a^l_{02}	a^l_{07}	a^l_{08}	a^l_{12}	a^l_{14}	a^l_{23}
time[s]	4.06	2.84	4.60	3.88	1.65	3.60
alc	a^l_{24}	a^l_{35}	a^l_{38}	a^l_{56}	a^l_{68}	a^l_{78}
time[s]	3.60	2.60	1.60	1.60	2.60	4.21

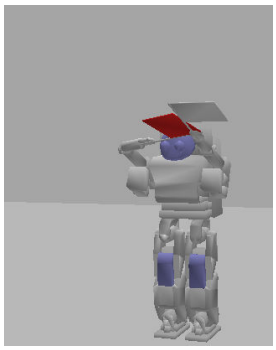

Fig. 11. Conflict

$\{t^r_{sij}, t^l_{sij}\}$ and $T_e = \{t^r_{eij}, t^l_{eij}\}$. These transitions mean start and completion of the actions a^r_{ij}/a^l_{ij}. A part of the Petri net is shown in Fig. 10.

Let an initial state (s^r_7, s^l_0) and a goal state (s^r_0, s^l_8). Controlling loop starts at **path planning** to give an optimal trace t^r_{s76} t^l_{s08} t^r_{e76} t^r_{s60} t^l_{e08} t^r_{e60}. This path is checked by **feasibility check** and **collision check** finds a collision between p^r_{d60} and p^l_{d08} (Fig. 11), and the inhibitor arcs $p^r_{d60} \mapsto t^l_{s08}$ and $p^l_{d08} \mapsto t^r_{s60}$ are added to the Petri net. Then an optimal path is searched again and we get t^r_{s72} t^l_{s08} t^r_{e72} t^r_{s20} t^l_{e08} t^r_{e20}, that is feasible. The optimal path is illustrated in Fig. 12. Then the transitions of the optimal trace is sent to **modular state nets** one by one, and the simulation is completed successfully. Captured screens in the simulation are shown in Fig. 13. The graphical output is done by OpenHRP[5].

VII. EXPERIMENT

We also carried out an experiment for a humanoid robot HOAP-1. The CPU of the host computer is Pentium III 1GHz, the OS is RT-Linux. The control program is written in C++ with STL. The host computer controls the HOAP-1 directly through an USB interface. The initial and goal state is given as the simulation, and the same optimal path is derived. The robot was controlled along the path. The results are shown in Fig. 14.

VIII. CONCLUSION

In this paper, we have proposed a supervisory control system for motion planning of humanoid robots. The proposed system is hierarchically structured into two levels. The lower level controls and monitors the robots using modular state nets. The upper level generates an optimal sequence of actions for user's requirements using timed Petri nets. We have shown some numerical simulations and some experimentations for a humanoid robot HOAP-1.

In this research, only kinematic collision is detected so far, so it is future research to consider dynamic constraint to avoid collapsing. One approach is to check ZMP(zero moment point) of a composite trajectory.

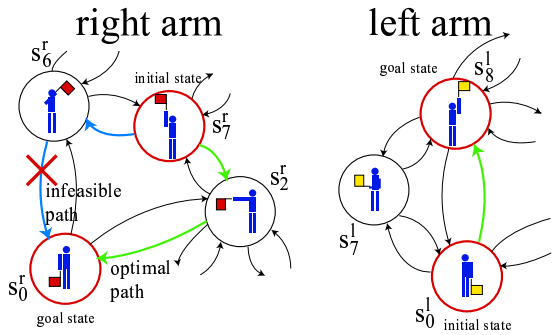

Fig. 12. Optimal path

Acknowledgments

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

References

[1] F. Kanehiro, M. Inaba, H. Inoue, and S. Hirai: Developmental Realization of Whole-Body Humanoid Behaviors Based on StateNet Architecture Containing Error Recovery Functions, Proc. of IEEE RAS International Conference on Humanoid Robots (2000)

[2] 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).

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

[4] 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).

[5] Y. Nakamura and K. Yamane, "Dynamics Computation of Structure-Varying Kinematic Chains and Its Application to Human Figures," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 2, pp. 124–134 (2000)

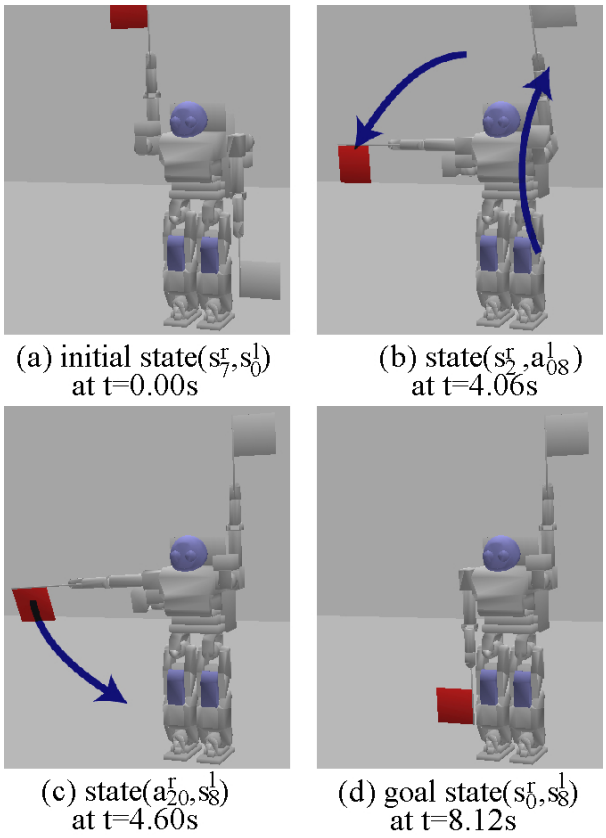

Fig. 13. Results of a simulation

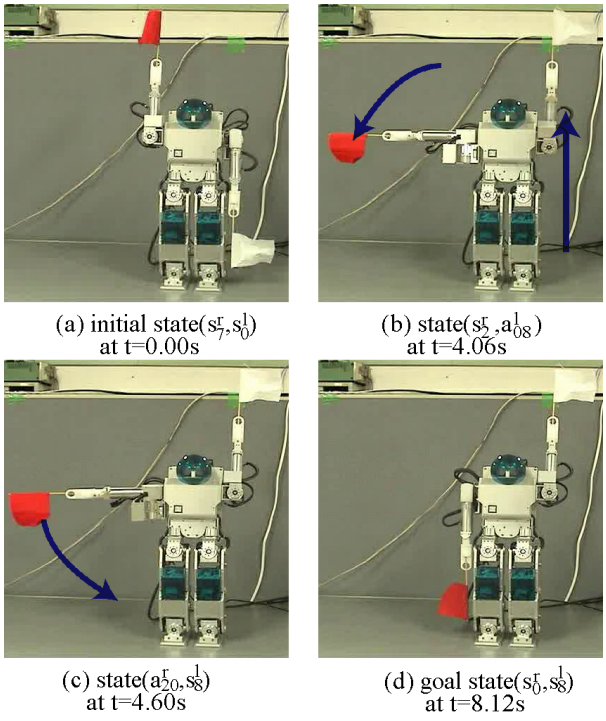

Fig. 14. Results of an experiment