論　文

時間付きペトリネットとモジュラステートネットを用いたヒューマノイドロボットの行動計画*

小林　啓吾†・仲谷　篤人‡・高橋　秀行‡・潮　　俊光‡

Motion Planning for Humanoid Robots Using Timed Petri Nets and Modular State Nets*

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

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.

1. はじめに

近年，ヒューマノイドロボットの行動計画に関する研究が盛んに行われている．しかし，歩行などの個別の動作に関する研究が進んでいるのに比べて，それらの動作単位を組合せて目的にかなった動作を実行するための研究は遅れている．ヒューマノイドロボットに対して，多数の行動単位からなるネットワークを用意し，ロボットの行動をネットワーク内の状態遷移として制御する手法として，ステートネットアーキテクチャが Kanehiro らにより提案されている [1]．このアーキテクチャでは，ロボットの行動空間がセンサ空間内に組み込まれた状態遷移グラフとして表されており，それゆえ，行動空間の逐次的拡張および統合の容易性，障害回復の自動化といった利点を持つ．しかし，ヒューマノイドロボットには多くのセンサが取り付けられているため，その状態空間の次元が大きくなってしまう．この結果，ロボットに多彩な行動をとらせるための状態数が多くなり，ネットワーク全体の見通しが悪くなってしまう．

本論文では，ヒューマノイドロボットを制御するための新たな手法として，モジュラステートネットと時間付きペトリネットによる 2 層構造を持ったアーキテクチャを導入する．モジュラステートネットとは，ロボットの腕，脚などといった各要素ごとの動作を表現するステートネットであり，ロボット全身の動作は，その各パーツに対するモジュラステートネットの組合せによって表現される．そしてロボット全身の実行可能な動作を得るため，全モジュラステートを抽象モデル化した時間付きペトリネットを導入し，離散事象システム理論によって最適規範に基づくスーパバイザ制御を行う．また，適用例としてヒューマノイドロボット HOAP-1 に対して，手旗信号をさせた時のシミュレーションおよび実験結果を示す．

2. モジュラステートネット

ヒューマノイドロボットの行動制御の手法としてステートネットアーキテクチャが，Fig. 1 に示すようなセンサ空間内に組み込まれた状態遷移グラフとして提案されている [1]．センサ空間内の点 p は各関節角のセンサ情報 s_i を用いて，座標 $p=[s_1\ s_2\ \cdots\ s_N]$ で表される．ステートネットにおいてノードは，ヒューマノイドロボットが静止している状態を表している．一方，行動している状態はセンサ空間内における時間付き軌道 $p(t)$ として定義され，あるノードから始まり別のノードへ至って終わるものとする．ステートネットアーキテクチャでは，新しいノードまたはアークを付け加えることによりネットワークを更新することができる．また現在のセンサ情報と実行中の行動状態に対する座標を比較することにより，障害の発生を検知することができる．しかし，ヒューマ

* 原稿受付　2002 年 10 月 29 日
† 九州工業大学 情報工学部　Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology; 680-4 Kawatsu, Iizuka city, Fukuoka 820-8502, JAPAN
‡ 大阪大学 大学院 基礎工学研究科　Graduate School of Engineering Science, Osaka University; 1-3 Machikaneyama-cho, Toyonaka city, Osaka 560-8531, JAPAN
Key Words: humanoid robot, modular state net, timed Petri net, supervisory control.

ノイドロボットは多くのセンサが取り付けられており状態の次元が大きくなると，多様な運動を与えるために十分な行動をあらかじめ準備することが難しくなる．そこで，状態ベクトル $p=[s_1\ s_2\ \cdots\ s_N]$ を $p=[p_1\ p_2\ \cdots p_M] = [\overbrace{s_1\cdots s_{i_1}}^{p_1}|\overbrace{s_{i_1+1}\cdots s_{i_2}}^{p_2}|\cdots|\overbrace{s_{i_{M-1}+1}\cdots s_N}^{p_M}]$ というサブベクトルに分割することを考える．ここで各サブベクトル p_j はそれぞれが腕，脚などといったヒューマノイドロボットの体の各部分を表すように設計する．各部分空間に対してステートネットを導入し，それをモジュラステートネットとよぶ．

モジュラステートネットアーキテクチャでは，ヒューマノイドロボットの動作が各パーツごと個別に設計され，ロボット全身の動作はそれらのモジュラステートネットの行動の組合せによって与えられる．組合せの数は大きいので，様々な行動を生み出すことが期待できる．しかし，組合せによっては動作の幾何学的干渉を引き起こすものや，動的な拘束を満たさないものがあり得るため，すべての組合せが必ずしも実行可能とは限らない．そこで実行可能な経路を見つけるため，全モジュラステートネットの集合を抽象化した離散モデルを導入する．

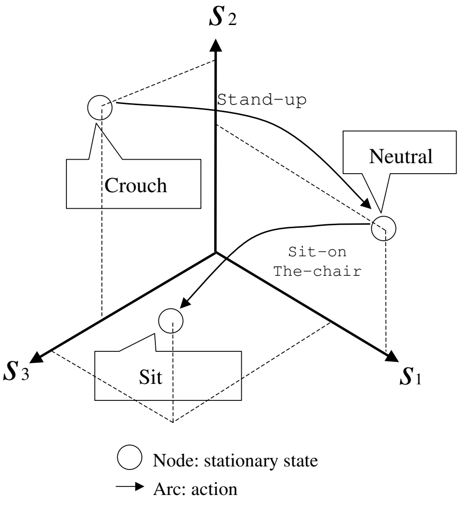

Fig. 1　StateNet

3.　時間付きペトリネット

モジュラステートネットの離散モデルとして時間付きペトリネット [3] を導入する．時間付きペトリネットは 6 項組み $G=(P,T,A,F,\theta,m_0)$ で与えられる．P はプレースの集合であり，二つの部分集合 P_s と P_d に分解される．すなわち $P=P_s\cup P_d$，$P_s\cap P_d=\emptyset$ とする．集合 P_s に属するプレースはモジュラステートネットのノード，すなわち静止状態を表す．一方，集合 P_d に属するプレースはモジュラステートネットのアーク，すなわち行動の実行を表す．T はトランジションの集合であり，2 つの部分集合 T_s, T_e に分解される．すなわち $T=T_s\cup T_e$, $T_s\cap T_e=\emptyset$ とする．集合 T_s 内のトランジションの発火は行動の開始を表し，集合 T_e 内のトランジションの発火は行動の完了を表す．T_s 内のトランジションはすべて制御可能，すなわち発火を禁止することができる．一方 T_e 内のトランジションはすべて不可制御である．なぜなら行動は時間付きの軌道で与えられており，決められた時刻に行動終了状態に達するので，その発火を禁止できないからである．$A:(P\times T)\cup(T\times P)\rightarrow N$ はプレースとトランジションの接続を表す写像である．$F:P\times T\rightarrow N$ は禁止アークの集合である．行動の実行に要する時間は $\theta:P_d\rightarrow\Re$ で表される．$m_0:P\rightarrow N$ は初期マーキングである．

トランジション t の前置プレース集合を ${}^{\bullet}t=\{p\in P|A(p,t)>0\}$ とする．トランジション $t\in T$ は

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

かつ

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

を満たすとき発火可能であるといい，これを $m[t>$ で表す．トレースをトランジション列で定義する．トレースの長さをそのトレースの中に含まれるトランジションの個数とする．トレース $\sigma=t_1t_2\ldots t_n$ が $m_0[t_1>m_1[t_2>\cdots m_{n-1}[t_n>$ を満たすとき σ は発火可能であり，これを $m_0[\sigma>$ と表す．

発火可能なトランジション t が発火したとき，マーキングは

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

で与えられる m' に変化する．これを $m[t>m'$ で表す．時刻 τ でプレース $p\in P_d$ にトークンが入ると，そのトークンは時刻 $\tau+\theta(p)$ まで，発火のために使用可能にならないとする．そして通常は時刻 $\tau+\theta(p)$ になると p からの入力アークをもつ不可制御トランジション $t\in T_e$ が発火する．なお，トランジションの発火では時間は経過しないものとし，同時刻に二つのトランジションが発火することは，二つのトランジションが時間間隔 0 で続けて発火するものとして解釈する．

ペトリネットは図で表現することが可能である．ペトリネットの各要素の図的表現を Table 1 に示す．アークの重み $A(p,t)$, $A(t,p)$ が 1 より大きいときには対応するアーク (p,t), (t,p) に重みを記述する．

Table 1　Graphical expression

P_s 内のプレース	○
P_d 内のプレース	□
トランジション	|
接続 (アーク)	→
禁止アーク	—○
トークン	●

禁止アークは，異なるモジュラステートネット間の行動の，順序的または並行的な組合せを避けるために使われる．禁止アークの結合には 2 種類ある．ひとつは P_s の要素である静的プレースから T_s 内のトランジションへつながるもので，これは順序的な動作の実行の禁止を表している．Fig. 2 の例では, 右腕の動作がスタートするまで左腕の動作を開始することが禁止されている．もうひとつは P_d の要素である動的プレースから T_s 内のトランジションへつながるもので，これは並列動作の実行の禁止を表している．Fig. 3 の例では, 右腕の動作が終了するまで左腕の動作を開始することが禁止されている．

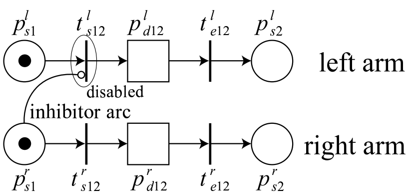

Fig. 2　Inhibitor arc from static place

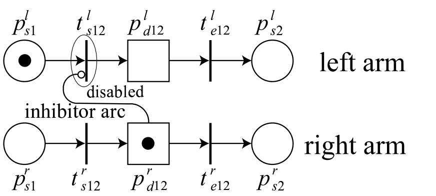

Fig. 3　Inhibitor arc from dynamic place

この時間付きペトリネットにおいては，与えられたトレースが実行可能かどうかは必ずしも自明ではない．例えば，$m_0[\sigma>$ を満たすトレース $\sigma = t^l_{s12}t^r_{s12}t^r_{e12}t^l_{e12}$ が存在するとする．しかし，σ は $\theta(p^l_{d12}) < \theta(p^r_{d12})$ のときには実行可能でない．なぜなら，もし時間付きトレースを $(t^l_{s12},\tau_1)(t^r_{s12},\tau_2)(t^r_{e12},\tau_3)(t^l_{e12},\tau_4)$ とすると，$\tau_4 = \tau_1 + \theta(p^l_{d12})$ と $\tau_3 = \tau_2 + \theta(p^r_{d12})$ は $\tau_1 \leq \tau_2 \leq \tau_3 \leq \tau_4$ に矛盾するからである．あるトランジションがこの意味において実行可能ならば，そのトランジションに対して実行に要する最小時間が定義できる．時間付きペトリネット $G = (P,T,A,F,\theta,m_0)$ がセーフであるなら，すなわち $m(p) \leq 1 (\forall p \in P, m_0[\forall\sigma > m)$ なら，以下の方法により実行可能性と最小時刻を計算できる．まず時間付きマーキングを $m: P \to \mathcal{B}$, $\psi: P \to \Re^+$ の対で定義する．ここで $\mathcal{B} = \{0,1\}$ であり，$\Re^+$ はすべての非負の実数の集合である．$m(p) = 1$ のときには，$\psi(p)$ はトークンが p に入った時刻を表す．トランジション t が時刻 τ で発火したとき，ψ は次のように更新される．

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

(3) 式と (4) 式をまとめて $(m,\psi)[(t,\tau) > (m',\psi')$ と書く．トランジション $t \in T_e$ は，$m[t>$ かつ，$A(p_a,t) > 0$ である $p_a \in P_d$ に対して

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

であるとき発火可能である．また発火が可能となる時刻は

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

で与えられる．$t \in T_s$ に対しては $\tau_{min}(m,\psi,t) = \max_{p \in P} \psi(p)$ とする．(4)〜(6) 式の繰り返しによりトレース $\sigma = t_1 t_2 \cdots t_n$ に対する実行可能性のチェックと実行時間の計算ができる．$\tau_i = \tau_{min}(m_{i-1},\psi_{i-1},t_i)$ かつ $(m_{i-1},\psi_{i-1})[(t_i,\tau_i) > (m_i,\psi_i)$ $(i = 1,2,\cdots,n)$ が成り立つとき，σ は (m_0,ψ_0) で実行可能であるという．σ が実行可能であるとき，$\tau_{min}(m_0,\psi_0,\sigma) = \tau_n$ が定義される．$\psi_0(p) = 0$ $(\forall p \in P)$ を仮定すると，$\tau_{min}(m_0,\sigma) = \tau_n$ も定義できる．すべての実行可能なトレースの集合を $L_F(m_0) \subset T^*$ と表記する．ここで T^* は T のクリーネ閉包である．

時間付きマーキング (m,ψ) が，$m(p_a) = 1$ かつ

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

であるようなプレース $p_a \in P_d$ をもつとき (m,ψ) はエラー状態なので，スーパバイザ [2] を設計する際にはこのような状態を避けるようにしなければならない．

4.　最適経路プランニング

現在の状態から任意に与えられた状態への最適経路を見つけるために，一種の limited lookahead policy スーパバイザ制御器 [4] を用いる．木の長さを一段一段伸ばしていき，すべての可能な有限の長さの経路からなる木を計算し，現在の状態から最短時間で目標の状態へ至る最適経路を見つける．

ゴールマーキングを m_G とし，$m_0[\sigma_0 > m_G$ かつ

$$\tau_{min}(m_0,\sigma_0) \leq \tau_{min}(m_0,\sigma) \quad (m_0[\forall\sigma > m_G) \tag{8}$$

となる最適トレース $\sigma_0 \in L_F(m_0)$ を探すことを考える．

長さ N の実行可能なトレースの集合を S_N とし，$R_N \subset S_N$ を長さ N でゴールに至るトレースの集合とする．すなわち，$\sigma \in R_N$ ならば $m_0[\sigma > m_G$ である．集合 $\cup_{i \leq N} S_i$ は長さ N の木を与えるが，このうちでゴールに至る最適トレースの集合を G_N とし，ゴールに至る最小時間は τ_N とする．このとき次のアルゴリズムによりすべての最適経路を求めることができる．

(1) $S_0 = \{\epsilon\}, R_0 = \emptyset, G_0 = \emptyset, \tau_0 = \infty, N = 1$ とする．

(2) $S_N = \{\sigma' | \sigma' = \sigma t \in L_F(m_0), t \in T, \sigma \in S_{N-1} \setminus R_{N-1}, \tau_{min}(m_0,\sigma') < \tau_{N-1}\}$, $R_N = \{\sigma | \sigma \in S_N, m_0[\sigma > m_G\}$ とする．

(3) $\tau_N = \min\{\tau_{N-1}, \min_{\sigma \in R_N} \tau_{min}(m_0,\sigma)\}$ とする．

(4) $\tau_N < \tau_{N-1}$ のとき $G_N = \{\sigma | \sigma \in R_N, \tau_{min}(m_0,\sigma) = \tau_N\}$ とし，それ以外は $G_N = G_{N-1} \cup \{\sigma | \sigma \in R_N,$

$\tau_{min}(m_0,\sigma)=\tau_N\}$ とする．

(5) $S_N \neq \emptyset$ なら $N=N+1$ とし，(2) へ戻る．そうでなければ G_N が最適トレースの集合であり，τ_N が最小時間である．

(2) では，不可制御事象によって禁止状態に陥ることのない事象列の集合である最大可制御部分言語 [7] を得るために，ゴールノード，実行不可能なノード，実行時間がそれまでの最適時間を超えているノードから続く枝を刈っている．(3) では，今までに見つかっている最適時間よりも短い時間をもった新たなゴールが存在するとき，最適時間を更新している．(4) では，(3) で最適時間が更新されたとき G_N を新たな最適時間をもった新たなゴールノードで置き換え，そうでなければ新たに見つかったゴールノードを G_N に追加している．

すべてのトランジション $t \in T_e$ は発火までに正の時間を要するので，このアルゴリズムは有限ステップで終了する．このアルゴリズムの例を Fig. 4 に示す．ここで m_i はノードのマーキングで，τ はノードに到達する最小時間である．

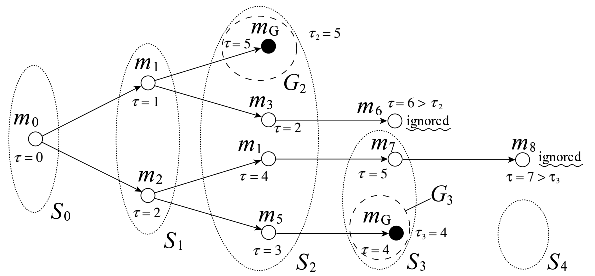

Fig. 4　Tree for finding the optimal traces

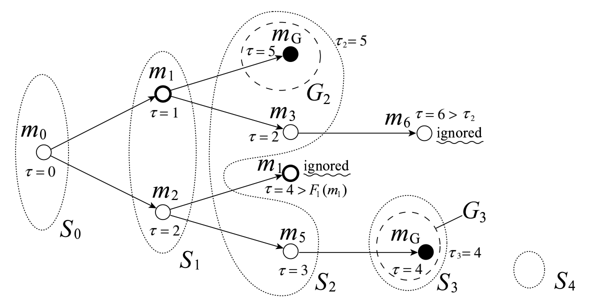

Fig. 5　Tree by improved algorithm

上記のアルゴリズムでは同じ軌道を行って戻るような明らかにむだな探索が含まれている．そこで計算コストを減らすため，もうひとつアルゴリズムを示す．いま $m_0[\sigma_1>m_a$ かつ $m_0[\sigma_2>m_a$ かつ $\tau_{min}(m_0,\sigma_1)<\tau_{min}(m_0,\sigma_2)$ を満たす 2 つのトレース σ_1 と σ_2 があったとする．σ_2 から伸びる枝は σ_1 から伸びる枝よりも最適経路を持っていることを期待できない．そこでノード σ_2 を無視することにより大幅に探索時間を削減することができる．

木 $\cup_{i\leq N}S_N$ 内のすべてのマーキングからなるマーキングの集合 M_n および，マーキングに到達する最小時間を表す写像 $F_n: M_N \rightarrow \Re^+\cup\{0\}$ を導入する．改良したアルゴリズムは次のようになる．

(1) $S_0=\{\epsilon\},R_0=\emptyset,G_0=\emptyset,\tau_0=\infty,N=1$ とする．

(2) $S_N=\{\sigma'|\sigma'=\sigma t\in L_F(m_0),t\in T,\sigma\in S_{N-1}\setminus R_{N-1},\tau_{min}(m_0,\sigma')<\tau_{N-1},(m_0[\sigma'>m'\in M_{N-1}\Rightarrow\tau_{min}(m_0,\sigma')\leq F_{n-1}(m'))\}$, $R_N=\{\sigma|\sigma\in S_N, m_0[\sigma>m_G\}$ とする．

(3) $\tau_N=\min\{\tau_{N-1},\min_{\sigma\in R_N}\tau_{min}(m_0,\sigma)\}$ とする．

(4) $\tau_N<\tau_{N-1}$ のとき $G_N=\{\sigma|\sigma\in R_N,\tau_{min}(m_0,\sigma)=\tau_N\}$ とし，それ以外は $G_N=G_{N-1}\cup\{\sigma|\sigma\in R_N,\tau_{min}(m_0,\sigma)=\tau_N\}$ とする．

(5) $M_N=M_{N-1}\cup\{m|\sigma\in S_N,m_0[\sigma>m\}$ とする．$\sigma\in S_N$かつ $m_0[\sigma>m$ であるすべての m に対して，$F_N(m)=\tau_{min}(m_0,\sigma)$ とする．またすべての $m\in M_N\setminus\{m|\sigma\in S_N,m_0[\sigma>m\}$ に対し，$F_N(m)=F_{N-1}(m)$ とする．

(6) $S_N\neq\emptyset$ なら $N=N+1$ とし，(2) へ戻る．そうでなければ G_N が最適トレースの集合であり，τ_N が最小時間である．

(2) では，マーキング m' が既に木の中に現れている (すなわち $m'\in M_{N-1}$) ノードで，なおかつ実行時間 $\tau_{min}(m_0,\sigma')$ がそのマーキングの最小到達時間より長いノードを刈っている．M_N は木の中に現れる全ノードの集合であり，(5) ではすべての新しいノードのマーキングを付け加えている．新しいノードは M_{N-1} に含まれていないか，あるいは最小実行時間が最小であるマーキングをもっているので，F_N はすべての新しいマーキングに対して更新される．アルゴリズムの例を Fig. 5 に示す．この例では，マーキング m_1 に対するノードが計算コスト削減のためカットされている．

なお，厳密には $(m_0,\psi_0)[\sigma_1>(m_a,\psi_a)$，$(m_0,\psi_0)[\sigma_2>(m_b,\psi_b)$ における ψ_a,ψ_b の関係によっては，$\tau_{\min}(m_a,\sigma_1)<\tau_{\min}(m_b,\sigma_2)$ であっても，あるトレース s に対して $\tau_{\min}(m_a,\sigma_1 s)<\tau_{\min}(m_b,\sigma_2 s)$ となる場合がある．しかしそのような場合はまれであり，上記のアルゴリズムで十分良い解が見つかる．

なお，時間付きペトリネットに対する最適パスの計算手法として A* アルゴリズムとその改良版を用いることが提案されている [8,9]．本論文では不可制御なトランジションがあるためにこれらの手法を直接適用することはできないが，A* アルゴリズムの拡張等によって，より効率的な行動計画を行うことは今後の課題である．

5.　システム全体構成

本章ではユーザからの要求に応じたヒューマノイドロボットの最適な行動計画を作成するシステムを提案する．Fig. 6 のように，このシステムは階層的な構造となっている．下位レベルはモジュラステートネットにより構成されており，それぞれのモジュラステートネットはヒュー

マノイドロボットの各部の監視と制御を行っている．また，上位レベルは全身の最適な行動計画を作成している．**path planning** では各部のモジュラステートネットを時間付きペトリネットで表している．**collision check** ではロボットの腕部を覆う球体の列と胴体を覆う直方体を考え，全身の時間軌道にそって幾何学的な干渉が存在するかどうかをチェックしている．判定の精度（時間の刻み幅）を適切に定めることによりオンラインでのチェックが可能となっている．また **feasibility check** は **collision check** で行われた判定結果を管理しており，すでに衝突の有無が分かっている軌道の組合せのチェックを再び行うむだを省いている。作成された行動計画に衝突が起こりうることが判明した場合，時間付きペトリネットは禁止アークを追加して経路の再検索を行う．衝突がなければ計画された経路は **command queue** に送られ，**commandserver** からそれぞれのモジュラステートネットに一つずつ送られる．モジュラステートネットは開始トランジションを受け取るとそれに応じた行動を行い，行動が終了するまでは次の終了トランジションを受け取らない．実行結果は **event translator** によって検出される．

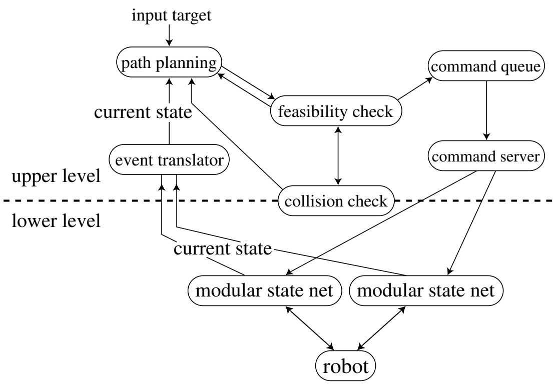

Fig. 6　Control system

6.　シミュレーション・実験

6.1　シミュレーション結果

提案する手法の有用性をシミュレーションによって示す．ここでは，ヒューマノイドロボット HOAP-1(富士通オートメーション (株)) を用いた．このロボットの全長は 0.438[m]，重量は 5.67[kg]．20 の関節を持ち，各腕は四つの自由度を持つ．今回は手旗信号を例とし，任意の初期状態から任意の目標状態に対し最適な経路を検索し，得られた軌道に沿って行動制御を行った．

まず初めに，Figs. 7, 8 に示すように両腕のモジュラステートネットを作成した．右腕はステート s^r_0 から s^r_7，左腕は s^l_0 から s^l_8 をそれぞれ持ち，ステート s^r_i から状態 s^r_j へのアークを a^r_{ij} のように定めた．Tables 2, 3 に各アークの実行時間を示す．

ここで，提案システムによる干渉回避の例を示す．ここでは初期状態を (s^r_7, s^l_0) とし，目標状態を (s^r_0, s^l_8) とした．まず，**path planning** で $t^r_{s76}, t^l_{s08}, t^r_{e76}, t^r_{s60}, t^l_{e08}, t^r_{e60}$ が最適な軌道として得られた．次に Fig. 9 のように，**feasibility check** および **collision check** によって p^r_{d60} と p^l_{d08} の間で衝突が検出され，禁止アーク $p^r_{d60} \mapsto t^l_{s08}$ と $p^l_{d08} \mapsto t^r_{s60}$ がペトリネットに追加される．このように実行可能でない軌道は禁止アークによってネットワークに記録される．

これらのステートネットは時間付きペトリネットにより抽象化されており，静的プレースは $P_s = \{p^r_{si}, p^l_{sj} | i = 0,1,\ldots,7, j = 0,1,\ldots,8\}$ と表現され，プレース p^r_{si}, p^l_{sj} はそれぞれノード s^r_i, s^l_j を表す．また，動的プレースは $P_d = \{p^r_{dij}, p^l_{dkl} | (i,j) = 0,1,\ldots,7, (k,l) = 0,1,\ldots,8\}$ と表現され，プレース p^r_{dij}, p^l_{dkl} はそれぞれアーク a^r_{ij}, a^l_{kl} を表す．なお，トランジションは $T_s = \{t^r_{sij}, t^l_{sij}\}$ および $T_e = \{t^r_{eij}, t^l_{eij}\}$ と表される．これらのトランジションは a^r_{ij}, a^l_{ij} による行動の開始と終了を表す．ペトリネットの一部を Fig. 10 に示す．さらに，最適な経路を再検索すると，Fig. 11 のように実行可能な経路 $t^r_{s72}, t^l_{s08}, t^r_{e72}, t^r_{s20}, t^l_{e08}, t^r_{e20}$ が得られた．この最適経路のトランジション列は各部の **modular state net** に一つずつ送られ，シミュレーションにより動作が確認された．OpenHRP によるシミュレーション [5,6] の実行画面を Fig. 12 に示す．

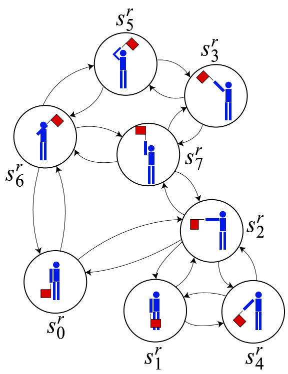

Fig. 7　Module of right arm

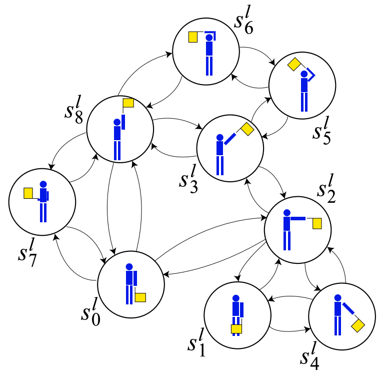

Fig. 8　Module of left arm

Table 2 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 3 Execution time of left arc

arc	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
arc	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	5.30

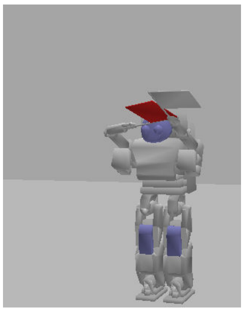

Fig. 9 Conflict

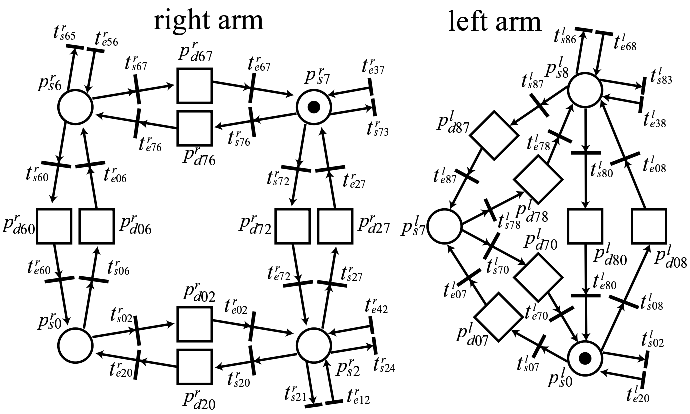

Fig. 10 A part of Petri net model

6.2 実験

6.1 で得られた行動計画が実現できることを，実機を用いて実験をおこなった．制御用のホストコンピュータの CPU は Pentium 1GHz で，OS は RT-Linux，使用言語は C++を用いた．HOAP-1 の制御は USB ケーブルを介してホストコンピュータから直接行った．シミュレーションと同様に最適な制御命令列が計算され，実機を制御することができた．動作の様子を Fig. 13 に示す．

7. おわりに

本論文では，スーパバイザ制御によるヒューマノイドロボットの行動制御システムを提案した．このシステムは 2 つの階層で構成されており，下位レベルはモジュラ

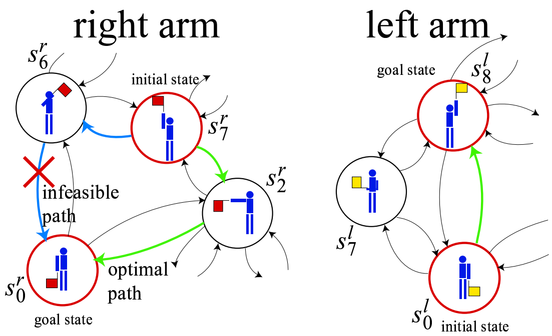

Fig. 11 Optimal path

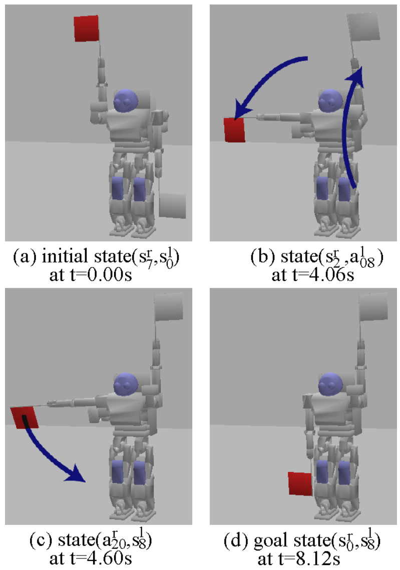

Fig. 12 Results of a simulation

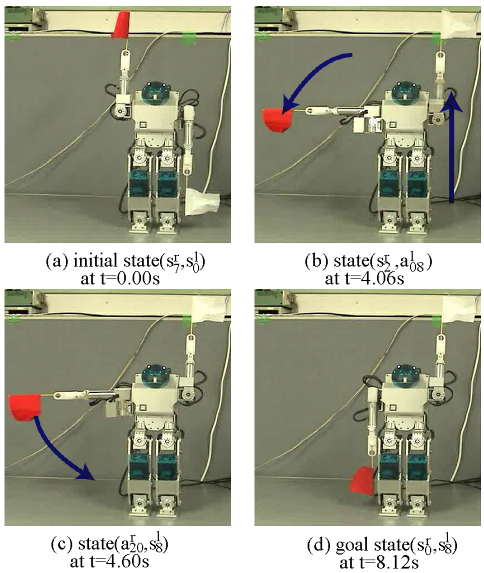

Fig. 13 Results of an experiment

ステートネットを用いてヒューマノイドロボットの制御と監視を行い，上位レベルは時間付きペトリネットを用いて，ユーザの要求に応じた最適な行動計画を作成して

いる．また，ヒューマノイドロボットの HOAP-1 によるシミュレーションと実験の結果を示した．

本論文では，**collision check** による判定は時刻にそっての干渉をチェックしているだけであり，各部の描く軌跡どうしの干渉はチェックしていないので，衝突を回避できる速度を求めるのは容易ではない．提案手法を利用するのであれば，速度の異なった軌道をあらかじめ用意しておく，衝突の前後の地点にノードを追加しやり過ごしを行う，といった方法で対処することは可能だと思われる．また、今後の課題としては動力学による制限と転倒回避が挙げられ，組合せられた軌道の ZMP(zero moment point) を検証する手法などを現在開発中である．

謝　　辞

本研究は科学事業振興事業団戦略的基礎研究推進事業 (CREST) 領域「脳を創る」の補助を受けた．また多くの有益な助言をいただいた中村仁彦先生（東京大学）に謝意を表す．

参 考 文 献

[1] F. Kanehiro, M. Inaba, H. Inoue and S. Hirai: Developmental realization of whole-body humanoid behaivors based on state net architecture containing error recovery functions; *Proc. of the First 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 Publishers (1998)

[4] S.-L. Chung, S. Lafortune and F. Lin: Limited lookahead policies in supervisory control of discrete event systems; *IEEE Trans. Automat. Contr.*, 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 Trans. Robotics and Automat.*, Vol. 16, No. 2, pp. 124–134 (2000)

[6] 比留川博久，小原一太朗，河村進: ＨＲＰ仮想プラットフォームにおける幾何計算サーバ; 日本機械学会ロボティクス・メカトロニクス講演会’99 講演論文集, 2P2-78-094

[7] W. M. Wonham and P. J. Ramadge: On the supremal controllable sublanguage of a given language; *SIAM Journal on Control and Optimization*, Vol. 25, No. 3, pp. 637–659 (1987)

[8] D. Y. Lee and F. DiCesare: Scheduling flexible manufacturing systems using petri nets and heuristic search; *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 2, pp. 123–132 (1994)

[9] A. Inaba, F. Fujiwara, T. Suzuki and S. Okamura: Timed petri net based scheduling for mechanical assembly — integration of planning and scheduling — ; *IEICE Trans. on FUNDAMENTALS*, Vol. E81-A, No. 4, pp. 615–625 (1998)

著 者 略 歴

小林　啓吾 (正会員)

1971 年 5 月 10 日生．1999 年 3 月京都大学大学院工学研究科機械工学専攻博士後期課程修了．同年 4 月大阪大学大学院基礎工学研究科システム科学分野助手，2003 年 4 月九州工業大学情報工学部制御システム工学科助教授，現在に至る．非ホロノミック系・ロボットシステムの研究に従事．博士（工学）．日本ロボット学会，計測自動制御学会，IEEE などの会員．

仲谷　篤人

1980 年 2 月 15 日生．2002 年 3 月大阪大学基礎工学部システム科学科卒業．同年同大学大学院基礎工学研究科システム人間系専攻博士前期課程に進学し現在に至る．

高橋　秀行

1979 年 2 月 8 日生．2002 年 3 月大阪大学基礎工学部システム科学科卒業．同年同大学大学院基礎工学研究科システム人間系専攻博士前期課程に進学し現在に至る．

潮　俊光 (正会員)

論文誌 Vol. 16, No. 3, p. 124 参照