時間オートマトンによるソフトリアルタイムシステムの性能解析手法

山根 智[†]

近年，ハードリアルタイムシステムの仕様記述言語としては，タイミング制約が記述可能な時間オートマトンが定着しており，最近，時間オートマトン上でのハードリアルタイムオペレーティングシステムのスケジューラビリティ検証手法が開発された．一方，最近では，分散システムやマルチメディアシステムなどのソフトリアルタイムシステムが増加しており，その重要性が認識されている．ソフトリアルタイムシステムでは，その性能解析手法が重要である．本論文では，時間オートマトンのハードリアルタイムシステムのスケジューラビリティ検証手法を価値の概念により拡張して，ソフトリアルタイム性の性能解析手法を開発する．

Performance Analysis Method of Soft Real-time Systems Using Timed Automata

Satoshi Yamane[†]

Generally, hard real-time systems have been specified using timed automata, and moreover recently, the verification method of schedulability of real-time operating systems using timed automata have been developed. On the other hand, as soft real-time systems such as distributed systems and multimedia systems have been increasing, it is important to design soft real-time systems. Especially, performance analysis methods are important for soft real-time systems. In this paper, we develope the performance analysis method of soft real-time systems by extending the verification method of schedulability of hard real-time systems using utility functions.

1. まえがき

リアルタイムシステムはリアルタイムオペレーティングシステムとともに動作して，信頼性保証が難しいシステムであり，仕様記述と検証は重要である[1]．リアルタイムシステムでは，デッドラインを絶対的に維持しなければならないハードリアルタイムシステムとデッドラインを超えるとシステムの価値が減少するソフトリアルタイムシステムに分類される[2]．ハードリアルタイムシステムでは，タスクの最悪応答時間がデッドライン以下であるかどうかを検証する，スケジューラビリティ検証が重要である[3]．一方，ソフトリアルタイムシステムでは，タスクの計算時間に従った価値の総和として求められる，性能によって，システムを評価することが重要である[4]．

最近，時間オートマトン上でのハードリアルタイムシステムのスケジューラビリティ検証手法が開発された[5),6]．2002 年の Yi らの研究[6] の以前には，リアルタイムオペレーティングシステムの仕様記述とスケジューラビリティ検証では，プリエンプティブスケジューリングのために，時間オートマトンによる仕様記述とスケジューラビリティ検証ができないと考えられており，ハイブリッドオートマトンのサブクラスであるストップウォッチオートマトンで実現されていた[7)〜11)]．

一方，ソフトリアルタイムシステムの性能解析に関する研究のほとんどは，計算時間がデッドラインを超す確率や計算時間に対する価値を性能評価の指標としており，以下のような代表的な研究がある．

(1) 1985 年，CMU の Jensen らはリリースからの経過時間をパラメータとする 4 つの価値関数により，プロセススケジューリングの性能評価を行っている[13]．システムの負荷の高低にかかわらず，最も短い計算時間を有するタスクを優先して実行させる方式が最も良い性能評価値を示すことを報告している．

(2) 1993 年，プリンストン大学の Kao らは分散型ソフトリアルタイムシステムのサブタスクに

† 金沢大学大学院自然科学研究科電子情報科学専攻
Division of Electrical and Computer Engineering, Graduate School of Natural Science and Technology, Kanazawa University

デッドラインを割り付ける 4 つの手法を提案して，デッドラインミス率により，その性能評価を行っている[14),15)]．

(3) 1994 年，テキサス大学の Kavi らはリアルタイムなタスクのデッドラインミスの確率により，性能を評価する手法を提案している[16)]．これにより，マルチタスクの性能評価を実現している．

(4) 1999 年，イリノイ大学の Gardner はタスクの計算時間を確率分布でモデル化して，デッドラインミスの確率により，性能を評価する手法を提案しており，デッドラインミスの確率は CPU 利用率と同様な指標になりうることを示している[17)]．

他方では，ソフトリアルタイムシステムを対象とした関連研究として，確率の概念を導入した仕様記述と検証の研究がある．代表的な研究としては，Hansson らの確率プロセス代数とその形式的検証[18)]，Alur らの連続確率分布を持つ確率時間オートマトンとその形式的検証[19)]，Kwiatkowska らの離散確率分布を持つ確率時間オートマトンとその形式的検証[20)]，D'Argenio らの確率時間プロセス代数とその形式的検証[21)] などがあるが，ソフトリアルタイムシステムの性能評価には言及していない．

本論文では，ソフトリアルタイムシステムの性能評価としてよく使われており，その特徴をよく表現している価値関数と，リアルタイムシステムの形式的仕様記述言語として有用な時間オートマトンとを統合することにより，新たなソフトリアルタイムシステムの性能解析手法を提案する．具体的には，時間オートマトン上でのハードリアルタイムシステムのスケジューラビリティ検証手法[5),6)] を価値の概念で拡張して，新たな時間オートマトンを開発することにより，価値関数を使って，ソフトリアルタイムシステムの性能解析を実現する．本論文の手法は以下の点において，Yi らのスケジューラビリティ検証手法[5),6)] と大いに異なる．

(1) Yi らの手法[5),6)] では，タスクは最悪応答時間とデッドラインの 2 つのパラメータを持つが，本論文では，さらに，リリースされてからの経過時間の価値関数が追加される．これにより，時間オートマトンの意味の定義が異なる．

(2) Yi らの手法[5),6)] では，スケジューラビリティ検証問題を到達可能解析に帰着しているが，本論文では，性能解析問題をすべての経路（計算木）のすべての状態ごとの価値の計算に帰着している．

また，従来のソフトリアルタイムシステムの性能解析手法はシミュレーション方式であり，タスクの計算時間，デッドラインや周期などのパラメータを与えて，性能を計算するものである．一方，提案手法は形式的手法を用いた方式であり，時間オートマトンの到達可能解析問題に帰着して，性能を計算するものである．

本論文では，新たな時間オートマトンによるソフトリアルタイムシステムの性能解析手法を開発する．

以降の本論文の構成は以下のとおりである．2 章では，ソフトリアルタイムシステム向きの時間オートマトンを提案する．3 章では，ソフトリアルタイムシステムの性能解析手法を提案する．4 章では，ソフトリアルタイムシステムの性能解析手法の実装を説明する．最後に，5 章では，まとめと今後の課題を述べる．

2. ソフトリアルタイムシステム向き時間オートマトンの提案

本章では，Yi らの時間オートマトン[5),6)] を拡張して，ソフトリアルタイムシステム向き時間オートマトンを提案する．

2.1 ソフトリアルタイムシステムの定義

まず，ソフトリアルタイムシステムを定義する．ハードリアルタイムシステムでは，最悪応答時間がデッドラインを超えると，システムの価値がなくなる．一方，ソフトリアルタイムシステムでは，最悪応答時間がデッドラインを超えると，システムの価値が減少する．以下に，ソフトリアルタイムシステムの価値の変化例を図示する．図のソフトリアルタイムシステムの価値関数では，最悪応答時間 C がデッドライン D 以下であれば，システムの価値はハードリアルタイムシステムと同じであり，最悪応答時間がデッドラインを超えると，価値関数 $v(t)$ に従って，システムの価値は減少する．ここで，t はタスクがリリースされてからの経過時間である．ただし，周期タスクの場合は，最悪応答時間は周期以下でなければならないし，周期タスクでない場合は，最悪応答時間はシステムの性能に関わるある値以下でなければならない（**図 1**）．

2.2 時間オートマトンの構文

まず，タスクを定義する．Yi らのアイディア[5),6)] では，タスクは最悪応答時間とデッドラインの 2 つのパラメータを持つが，本論文では，さらに，リリースされてからの経過時間の価値関数が追加される．これにより，性能解析を実現する．

Definition1（タスク）

$\mathcal{P}$ は P，Q，R などのタスクタイプの集合を表記する．タスクタイプは違ったインスタンスであるタスクを持つ．なぜならば，同じプログラムは違った入力を

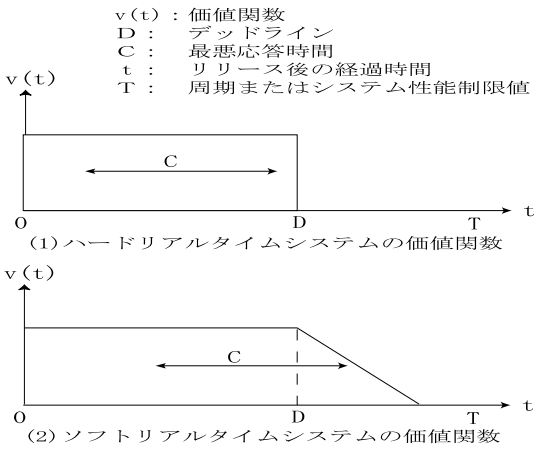

図 1 リアルタイムシステムの価値関数
Fig. 1 Utility functions of real-time systems.

受け取ると異なる動作をするので，違ったタスクを持つと考える．各タスクは $P(D_P, C_P, v_P(t_P))$ として特徴付けられる．ただし，C_P は最悪応答時間，D_P はデッドライン，$v_P(t_P)$ はタスク P がリリースされてからの経過時間 t_P の価値関数 v_P の値である．なお，D_P と C_P は知られているとする．また，$C_P(P)$ と $D_P(P)$ はタスク P の最悪応答時間とデッドラインを表す．

■

次に，時間オートマトンの定義の準備を行う．

Definition2（準備）

以下の定義を行う．

(1) Act は有限のアルファベットであり，外部からリアルタイムシステムにくるイベントの有限集合である．$Act = \{a, b, c, \ldots\}$ とする．

(2) $\mathcal{C}$ は実数値をとるクロック変数の有限集合である．$\mathcal{C} = \{x_1, x_2, x_3, \ldots\}$ とする．

(3) $\mathcal{B}(\mathcal{C})$ はクロック制約 g の集合である．クロック制約 g は $x_i \sim d_i$ または $x_i - x_j \sim d_{ij}$ の論理積である．ここで，$x_i, x_j \in \mathcal{C}$，$\sim \in \{\leq, <, \geq, >\}$，$d_i$ と d_{ij} は自然数の定数である．

■

次に，時間オートマトンの構文を定義する．以下の時間オートマトンでは，Yi らのアイディア[5),6)] に従って，ノードにタスクを割り付ける．これは，リアルタイムシステムにおいて，イベントが発生すると，そのイベントに関連付けられたタスクが到着するという現象をモデル化している．

Definition3（時間オートマトンの構文）

時間オートマトンは $\mathbf{A} = (N, l_0, E, I, M)$ の 5 つ組で定義される．ここで，

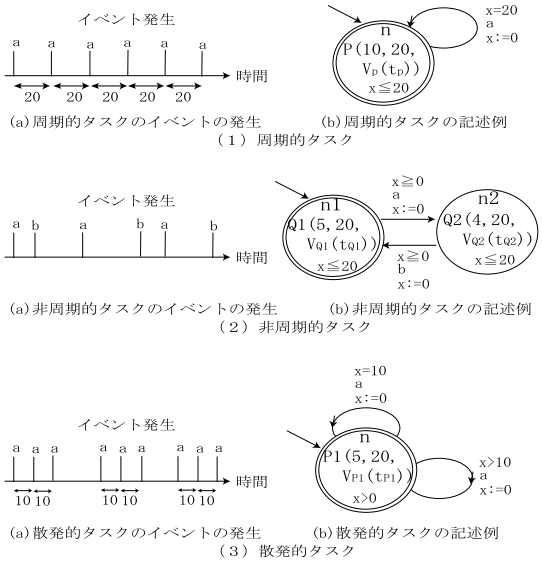

図 2 タスクの時間オートマトンの記述例
Fig. 2 Example of specification of tasks using timed automata.

(1) N はノードの有限集合．

(2) $l_0 \in N$ は初期ノード．

(3) $E \subseteq N \times \mathcal{B}(\mathcal{C}) \times Act \times 2^{\mathcal{C}} \times N$ はエッジの有限集合．

(4) $I : N \hookrightarrow \mathcal{B}(\mathcal{C})$ はノードにクロック制約を割り付ける部分関数である．

(5) $M : N \hookrightarrow \mathcal{P}$ はノードにタスクを割り付ける部分関数である．

■

次に，タスクのモデル化の事例を示す．

Example1（タスクのモデル化の事例）

タスクには，周期的タスク，非周期的タスクと散発的タスクの 3 つに分類できる[22),24)]．外部からのイベントが周期的に発生し，そのイベントに関連付けられたタスクが周期的に起動するシステムを周期的タスクと呼ぶ．それに対して，周期的でないものを非周期的タスクと呼ぶ．また，イベントの発生がある時間は集中的に起こり，その後はしばらく続かないものを散発的なイベントと呼び，そのイベントに関連付けられたタスクが散発的に起動するシステムを散発的タスクと呼ぶ．

図 2 に，周期的タスク，非周期的タスクと散発的タスクのイベントの発生パターンとその時間オートマトンによる仕様記述例を示す．

■

2.3 時間オートマトンの意味

次に，時間オートマトンの意味を定義する．

時間オートマトンは 2 つのタイプの遷移を実行する．時間遷移は高い優先度を持つ実行中タスクの実行に対応して，他のタスクの実行待ちに対応している．離散遷移は新しいタスクの到着に対応している．

まず，時間オートマトンの意味的な状態を定義する．Yi らのアイディア[5),6)] と異なり，本論文では，意味的な状態とその遷移に，リリースされてからの経過時間の価値関数が追加される．

Definition4（時間オートマトンの意味的な状態） C から非負実数 $\mathbf{R}$ への関数として，時間オートマトンのクロックの値を表現する．C のクロック値割当ての集合を $\mathcal{V}$ によって表現する．時間オートマトンの意味的な状態は組 (l, u, q) である．ここで，$l \in N$，$u \in \mathcal{V}$，q は現在のタスクキューである．なお，タスクキュー q は以下の形式で表現する：

$$[P_1(c_{P_1}, d_{P_1}, v_{P_1}(t_{P_1})), \ldots\ldots, P_n(c_{P_n}, d_{P_n}, v_{P_n}(t_{P_n}))]$$

ここで，$P_i(c_{P_i}, d_{P_i}, v_{P_i}(t_{P_i}))$ は残実行時間が c_{P_i}，デッドラインが d_{P_i}，価値が $v_{P_i}(t_{P_i})$ である，タスクタイプ P_i のリリースされたタスクを表す．ただし，t_{P_i} はタスク P_i がリリースされてからの経過時間を表す．

■

ここで，あるスケジューリングポリシに従って，リリースされたタスクを実行させる，固定数のプロセッサがあると仮定する．新しいタスクが到着するときはいつでも，固定優先度スケジューリング FPS（Fixed Priority Scheduling）や動的優先度スケジューリング EDF（Earliest Deadline First）などのスケジューリングポリシはタスクキューをソートする．離散遷移によって新たにリリースされたタスク P により，タスクキューがソートされる．t 時間の時間遷移によって，タスクキューの先頭のタスクが t 時間だけ実行されて，そのタスクの残実行時間とデッドラインを t 時間だけ減少させられ，リリースからの経過時間 t_P が計測されて価値 $v_P(t_P)$ が計算される．なお，価値 $v_P(t_P)$ はスケジューリングにも使用できる．また，タスクの残実行時間がゼロになれば，タスクの P の最終的な価値として $v_P(t_P)$ が計算されて，タスクキューから削除される．つまり，時間オートマトンのクロック変数の引き算を行う．以上をまとめると，以下のようになる：

(1) **Sch** はタスクキューのソート関数である．ソート関数 **Sch** をスケジューリングポリシーとも呼ぶ．たとえば，EDF($[P(3.1, 10, v_p(t_p)), Q(4, 5.3, v_q(t_q))]$) $=$ $[Q(4, 5.3, v_q(t_q)), P(3.1, 10, v_p(t_p))]$ である．
Sch のパラメータは，リリースからの経過時間によって変化する，残実行時間，デッドライン，価値の 3 つなので，固定優先度や動的優先度のスケジューリングに対応している．

(2) **Run** は，時間 t とタスクキュー q を受け取って，t 時間後のタスクキュー q を返す関数である．
たとえば，$q = [Q(4, 5, v_q(t_q)), P(3, 10, v_p(t_p))]$ とすると，$\mathbf{Run}(q, 6) = [P(1, 4, v_p(t_p + 6))]$ であり，Q の価値は $v_q(t_q + 4)$ である．

次に，時間オートマトンの操作的意味を定義するために，新たな記法 $u \models g$，$u+t$，$u[r \mapsto 0]$，$x := x - c$ を定義する．

(1) $u \models g$ はクロック割当て u がクロック制約 g を満たすことを表現する．

(2) $u + t$ は各クロック変数 x に値 $u(x) + t$ を割り付けるクロック割当てを表現する．

(3) $u[r \mapsto 0]$ は $r \subseteq \mathcal{C}$ の中の各クロックをリセットする．

(4) $x := x - c$ はクロック変数 x から c を引き算することを表現する．なお，c は自然数である．

以下に，遷移ルールによって，時間オートマトンの操作的意味を定義する．

Definition5（時間オートマトンの意味的な状態遷移）
スケジューリングポリシ **Sch** が与えられたとき，初期状態 (l_0, u_0, q_0) を持つ時間オートマトン $\mathbf{A} = (N, l_0, E, I, M)$ の意味は，以下のような遷移ルールを持つラベル付き遷移システムである．

(1) 離散遷移：
もし $l \xrightarrow{g,a,r} m$ かつ $u \models g$ ならば，
$(l, u, q) \xrightarrow{g,a,r}_{Sch} (m, u[r \mapsto 0], \mathbf{Sch}(M(m) :: q))$

(2) 時間遷移：
もし $(u + t) \models I(l)$ ならば，
$(l, u, q) \xrightarrow{t}_{Sch} (l, u + t, \mathbf{Run}(q, t))$
ここで，スケジューラはできるだけ長くノードに滞在してタスクを実行させると考えて，性能解析においては，時間遷移は t の最大値をとるとする．

ただし，$M(m) :: q$ はタスクキュー $M(m)$ に q が挿入されていることを表現する．

■

この時間オートマトンのモデルは，*nonZeno*[23)] と仮定する．すなわち，有限な時間間隔では有限の動作しか発生しないと仮定する．*nonZeno* の仮定は，リ

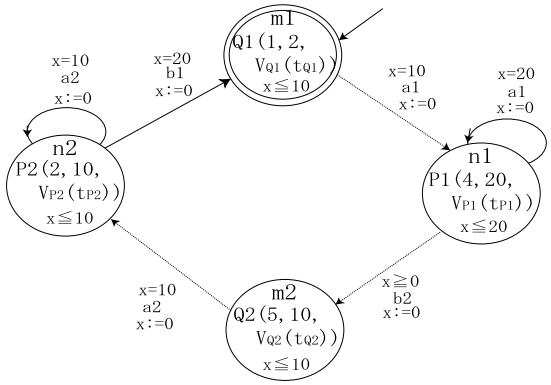

図 3 時間オートマトンの例
Fig. 3 Examples of periodic tasks, aperiodic tasks, sporadic tasks.

アルタイムソフトウェアの物理的動作をモデル化するための重要な仮定である．

以下に，タスクの操作的意味の例を示す．

Example2（タスクの動作例）

図 3 に，タスクのモデル化の例を示す．P_1 と P_2 は周期 20 と周期 10 の周期タスクであり，Q_1 と Q_2 はイベント駆動型のタスクである．まず，状態 m_1 でタスク Q_1 が動作して，10 時刻後にイベント a_1 が到着して，状態 n_1 に遷移する．次に，状態 n_1 でタスク P_1 が動作して，20 時刻で周期起動される．その後，イベント b_2 が到着して，Q_2 を起動する．その後，10 時刻後にイベント a_2 が到着して，状態 n_2 に遷移する．次に，状態 n_2 でタスク P_2 が動作して，10 時刻で周期起動される．

図のタスクの動作例を示す．

(1) まず，**Sch** を EDF とする．

$$
\begin{aligned}
&(m_1, [x=0], [Q_1(1,2,v_{Q_1}(0)]) \\
&\quad \xrightarrow{1}_{Sch} (m_1, [x=1], [Q_1(0,1,v_{Q_1}(1)]) \\
&\qquad = (m_1, [x=1], []) \\
&\quad \xrightarrow{9}_{Sch} (m_1, [x=10], []) \\
&\quad \xrightarrow{a_1}_{Sch} (n_1, [x=0], [P_1(4,20,v_{P_1}(0)]) \\
&\quad \xrightarrow{4}_{Sch} (n_1, [x=4], [P_1(0,16,v_{P_1}(4)]) \\
&\qquad = (n_1, [x=4], []) \\
&\quad \xrightarrow{16}_{Sch} (n_1, [x=20], []) \\
&\quad \xrightarrow{a_1}_{Sch} (n_1, [x=0], [P_1(4,20,v_{P_1}(0)]) \\
&\quad \xrightarrow{2}_{Sch} (n_1, [x=2], [P_1(2,18,v_{P_1}(2)]) \\
&\quad \xrightarrow{b_1}_{Sch} (m_2, [x=0], [Q_2(5,10,v_{Q_2}(0)), \\
&\qquad\qquad P_1(2,18,v_{P_1}(2)]) \\
&\quad \xrightarrow{5}_{Sch} (m_2, [x=5], [Q_2(0,5,v_{Q_2}(5)), \\
&\qquad\qquad P_1(2,18,v_{P_1}(7)]) \\
&\qquad = (m_2, [x=5], [P_1(2,18,v_{P_1}(7)]) \\
&\quad \xrightarrow{2}_{Sch} (m_2, [x=7], [P_1(0,16,v_{P_1}(9)]) \\
&\qquad = (m_2, [x=7], []) \\
&\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots
\end{aligned}
$$

(2) 次に，**Sch** を FCFS とする．

$$
\begin{aligned}
&(m_1, [x=0], [Q_1(1,2,v_{Q_1}(0)]) \\
&\quad \xrightarrow{1}_{Sch} (m_1, [x=1], [Q_1(0,1,v_{Q_1}(1)]) \\
&\qquad = (m_1, [x=1], []) \\
&\quad \xrightarrow{9}_{Sch} (m_1, [x=10], []) \\
&\quad \xrightarrow{a_1}_{Sch} (n_1, [x=0], [P_1(4,20,v_{P_1}(0)]) \\
&\quad \xrightarrow{4}_{Sch} (n_1, [x=4], [P_1(0,16,v_{P_1}(4)]) \\
&\qquad = (n_1, [x=4], []) \\
&\quad \xrightarrow{16}_{Sch} (n_1, [x=20], []) \\
&\quad \xrightarrow{a_1}_{Sch} (n_1, [x=0], [P_1(4,20,v_{P_1}(0)]) \\
&\quad \xrightarrow{2}_{Sch} (n_1, [x=2], [P_1(2,18,v_{P_1}(2)]) \\
&\quad \xrightarrow{b_1}_{Sch} (m_2, [x=0], [P_1(2,18,v_{P_1}(2), \\
&\qquad\qquad Q_2(5,10,v_{Q_2}(0))]) \\
&\quad \xrightarrow{2}_{Sch} (m_2, [x=2], [P_1(0,16,v_{P_1}(4)], \\
&\qquad\qquad Q_2(5,10,v_{Q_2}(2))) \\
&\qquad = (m_2, [x=2], [Q_2(5,10,v_{Q_2}(2))]) \\
&\quad \xrightarrow{5}_{Sch} (m_2, [x=7], [Q_2(0,5,v_{Q_2}(7))]) \\
&\qquad = (m_2, [x=7], []) \\
&\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots
\end{aligned}
$$

■

3. 性能解析手法の提案

本章では，時間オートマトンによるソフトリアルタイムシステムの性能解析手法を提案する．2002 年，Yi らは，到達可能解析が決定可能なクラスであるストップウォッチオートマトンを発見して，到達可能解析により，時間オートマトン上のハードリアルタイムシステムのスケジューラビリティ検証手法を開発した[5),6)]．本論文では，Yi らの手法を拡張して，前章で定義した時間オートマトンの意味を基礎として，ソフトリアルタイムシステムの性能評価手法を開発する．しかし，Yi らの手法では，スケジューラビリティ検証問題を到達可能解析に帰着しているが，本論文では，性能解析問題をすべての経路（計算木）のすべての状態ごとの価値の計算に帰着している．本論文で提案する手法は仕様レベルにおける性能解析手法である．一般的には，リアルタイムシステムの仕様には非決定性が存在しており，提案した時間オートマトンにも非決定性は存在する．非決定性時間オートマトンの到達可能解析は計算木を探索する必要があることが知られている[25)]．

以上を考慮して，本論文では，以下のように性能解析を行う．与えられた時間オートマトンから，意味的

な状態列を構成し，非決定性の動作があれば分岐構造を持つ計算木を構成する．ここで，計算木のルートは意味的な初期状態 (l_0, u_0, q_0) であり，各節点は意味的な状態 (l_k, u_k, q_k) である．なお，計算木を構成しながら，q_k の中の評価値を計算して，(l_k, u_k, q_k) にラベル付けする．つまり，最終的には，ラベル付き計算木が構成される．

まず，時間オートマトンの到達可能性の概念を定義する．

Definition6（時間オートマトンの到達可能性）
時間オートマトンの到達可能性は以下のように定義される：

$$\omega = (l_0, u_0, q_0) \xrightarrow{t_0} \ldots \overset{g_{i-1}, a_{i-1}, r_{i-1}}{\longrightarrow} (l, u, q)$$

または

$$\omega = (l_0, u_0, q_0) \xrightarrow{t_0} \ldots \xrightarrow{t_{i-1}} (l, u, q)$$

のときに限り，時間オートマトンは (l, u, q) に到達可能であると呼ぶ．

■

次に，ソフトリアルタイムシステムの性能解析手法を定義する．

Definition7（性能解析手法）
与えられた時間オートマトンから，到達可能な意味的な状態 (l_k, u_k, q_k) の列を構成する．なお，時間オートマトンに非決定性の動作があれば，状態の列は分岐構造を持つ計算木となる．この計算木のルートは意味的な初期状態 (l_0, u_0, q_0) であり，各節点は意味的な状態 (l_k, u_k, q_k) である．なお，深さ優先的に，決定性の動作ごとに，計算木を構成しながら，q_k の中の価値を以下のように計算して，計算木の節点 (l_k, u_k, q_k) にラベル付けして，ラベル付き計算木を構成する．

ここで，$(l_k, u_k, q_k)(k = 0, 1, 2, \ldots, k_s)$ について，$q_k = [{P_1}^k({c_{P_1}}^k, {d_{P_1}}^k, v_{P_1}({t_{P_1}}^k)), \ldots, {P_n}^k({c_{P_n}}^k, {d_{P_n}}^k, v_{P_n}({t_{P_n}}^k))]$ を調べて，以下のようにシステムの性能を計算する．

ただし，k_s は到達可能な有限な状態数であり，n は状態 (l_k, u_k, q_k) において存在する，有限なタスクの数である．これらの有限性は，時間オートマトンの *nonZeno* 性により保証される．

(1) 決定性の動作ごとに，初期状態より，到達可能な意味的な状態 (l_k, u_k, q_k) の列を構成して，価値を以下のように処理する：

(a) ${t_{P_i}}^k > T_{P_i}$ のとき：

システムはエラーであり，状態 (l_k, u_k, q_k) にエラーをラベル付けして，解析を打ち切る．

ただし，${t_{P_i}}^k$ はタスク P_i のリリースからの経過時間であり，T_{P_i} は P_i の周期またはシステムの性能を維持する制限値である．

(b) ${t_{P_i}}^k \leq T_{P_i}$ のとき：

システムの性能は以下のとおりである：

$\forall i$ に対して，${c_i}^k = 0$ のときの価値 $v_{P_i}({t_{P_i}}^k)$ を状態 (l_k, u_k, q_k) にラベル付けして，以下を計算する：

$$\sum_{i=1}^{n} v_{P_i}({t_{P_i}}^k)$$

(2) 非決定性の動作が存在すれば，非決定性の数だけ $\sum_{i=1}^{n} v_{P_i}({t_{P_i}}^k)$ が計算できる．これは決定性の動作ごとに，システムの性能が得られると考える．

■

また，上で述べたように，時間オートマトンの性能解析では，時間オートマトンのクロック変数の引き算を扱う必要がある．一般的には，クロック変数の引き算などの更新を有するような時間オートマトンの到達可能解析問題は決定不能であることが知られている[11]．しかし，有界な引き算を持つ時間オートマトンでは，その到達可能解析問題は決定可能である[5),6)]．本論文で提案した時間オートマトンも Yi らの時間オートマトンと同様に，有界な引き算を持つ時間オートマトンのクラスであり，以下の定理が導ける．

Theorem1（決定可能性）
有界な引き算を持つ時間オートマトンの到達可能解析問題は決定可能である．

Proof1 時間オートマトンの任意のクロックは最大定数によって制限されるので，任意のノード l に対して，状態の同値クラスの有限数が存在する[5),6)]．時間オートマトンのノードの数は有限なので，時間オートマトンのすべての状態空間は同値クラスの有限数に分割される[5),6)]．ゆえに，有界な引き算を持つ時間オートマトンの到達可能解析問題は決定可能である．

■

4. 性能解析手法の実装

本章では，提案した性能解析手法を実装する．

4.1 性能解析手法の実装のアルゴリズム

性能解析手法を実装するためには，**Definition5** の時間オートマトンの意味的な状態遷移に従って，**Definition7** の性能解析を実現することである．この場合のポイントとしては，ある状態で経過する時間と

タスクの残実行時間を比較して，価値を計算するところである．時間オートマトンの時間経過などを計算する手法としては，DBMs（Difference Bounds Matrices）[26)~31)] が知られている．DBMs は状態がとりうる時間ゾーンを記号的に表現する方法であり，ある時間ゾーンから経過しうる時間ゾーンなどを計算できる．

まず，DBMs を定義する．

Definition8（DBMs の定義）

$\mathbf{R}^n$ の DBMs はクロック変数 $x_1, \ldots, x_n$ を持ち，$(n+1) \times (n+1)$ の行列 D である．任意の i に対して，要素 D_{i0} はクロック変数 x_i の上限を表して，要素 D_{0i} はクロック変数 x_i の下限を表す．つまり，x_0 は 0 を意味する仮想クロックである．任意の i，j に対して，要素 D_{ij} はクロック変数 x_i と x_j の差の上限を表す．厳密な境界と厳密でない境界を区別（たとえば，$x < 2$ と $x \leq 2$ の区別）して，境界が存在しないことを表すために，上限と下限の値の領域は $\mathbf{Z} \times \{<, \leq\} \cup \{\infty\}$ とする．具体的には以下のとおりである：∞ は境界が存在しないことを表して，境界 $(c, \leq)$ は $\leq c$ を表して，境界 $(c, <)$ は $< c$ を表す．たとえば，$(0 \leq x_1 < 2) \wedge (0 < x_2 < 1) \wedge (x_1 - x_2 \geq 0)$ は，以下のように，DBMs で表される：

$$\begin{pmatrix} \infty & (0, \leq) & (0, <) \\ (2, <) & \infty & \infty \\ (1, <) & (0, \leq) & \infty \end{pmatrix}$$

■

また，DBMs は最短パスを求める Floyd-Warshall のアルゴリズムにより，標準形が計算できることが知られている．

次に，DBMs の演算を定義する．

Definition9（DBMs の演算）

本論文では，以下の演算を用いる：

(1) 標準形 $\mathbf{CF}(D)$：
DBMs D の標準形を求める演算子である．

(2) 積演算 $\mathbf{Conjunction}(D, D\prime)$：
DBMs D と $D\prime$ に対して，DBMs の積 $\mathbf{Conjunction}(D, D\prime)$ は，以下のように計算できる DBMs $D\prime\prime$ である：
$\forall i, j$ に対して，$D_{ij}\prime\prime = min\{D_{ij}, D_{ij}\prime\}$
ただし，$min\{D_{ij}, D_{ij}\prime\}$ は，D の ij-要素と $D\prime$ の ij-要素の最小値から構成する DBMs である．

(3) 時間前進演算 $\mathbf{Time\text{-}Sccessor}(D)$：
標準形の DBMs D の時間前進演算子では，時間の前進を制限する要素を削除するために，任意の i に対して，$x_i - x_0 \leq d_{i0}$ または $x_i - x_0 < d_{i0}$ を取り除く．

(4) リセット演算 $\mathbf{Reset}(D[r := 0])$：
DBMs D のリセット演算 $D[r := 0]$ では，リセットされるクロック集合 r の任意の要素 x_i に対して，$x_i - x_0 \leq 0$ かつ $x_0 - x_i \leq 0$ とする．

■

次に，DBMs によるタスクキューの計算方法を定義する．

Definition10（DBMs によるタスクキューの計算方法）

状態 $(l_k, u_k, q_k)(k = 0, 1, 2, \ldots, k_s)$ について，

$q_k = [{P_1}^k({c_{P_1}}^k, {d_{P_1}}^k, {v_{P_1}}({t_{P_1}}^k)), \ldots,$
${P_n}^k({c_{P_n}}^k, {d_{P_n}}^k, {v_{P_n}}({t_{P_n}}^k))]$

を解析する手法を定義する．

以下では，時間遷移と離散遷移に場合分けして計算する．

1．時間遷移

最初に，時間遷移を計算する．

まず，状態 (l_k, u_k, q_k) において経過する時間を計算する．

状態の入り口の DBMs を D とする．この D は前の状態からの遷移直後の状態である．ただし，初期状態では，すべてのクロックは 0 とする．

ここで，状態 (l_k, u_k, q_k) における経過時間を計測するクロック変数 t_{l_k} を導入して，D に追加して，標準形を計算した結果を $D\prime$=$\mathbf{CF}(D)$ とする．なお，$t_{l_k} = 0$ であり，最小値をとるクロック変数とする．つまり，以下を D に追加する．

$t_{l_k} - x_0 \leq 0$, $t_{l_k} - x_1 \leq 0$, …,
$t_{l_k} - x_n \leq 0$, $x_0 - t_{l_k} \leq 0$

これを元に，以下のように，状態 (l_k, u_k, q_k) における経過時間を計算する．

(1) まず，ノード l_k でとりうる DBMs を作る：
$\mathbf{Conjunction}(D\prime, D_{I(l_k)})$

(2) 次に，ノード l_k でとりうる DBMs が時間前進した DBMs を作る：
$\mathbf{Time\text{-}Sccessor}(\mathbf{Conjunction}(D\prime, D_{I(l_k)}))$
ただし，$D_{I(l_k)})$ はノード l_k に割り付いているタイミング制約である．

(3) 次に，ノード l_k でとりうる DBMs の時間前進した DBMs が実際にノード l_k でとりうるように，$D_{I(l_k)}$ と積演算する：
$\mathbf{Conjunction}(\mathbf{Time\text{-}Sccessor}$
$(\mathbf{Conjunction}(D\prime, D_{I(l_k)})), D_{I(l_k)})$

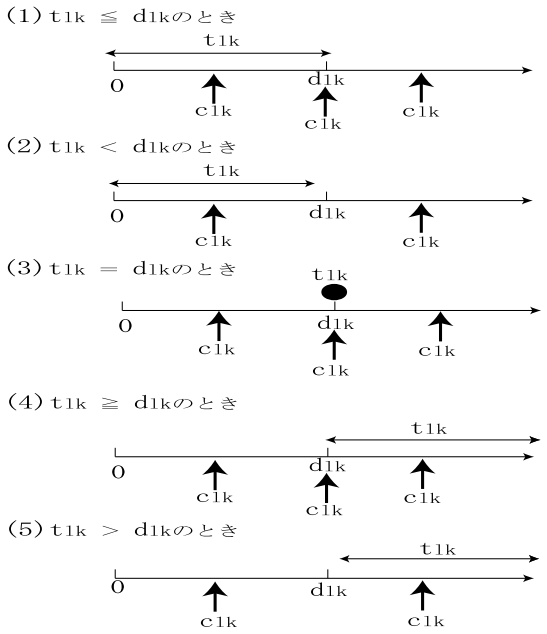

図 4 場合分けのパターン
Fig. 4 Cases of t_{l_k}.

(4) 次に，ノード l_k から遷移する状態への枝に付いてるクロック制約 g により，上記の DBMs を制限する必要があるので，積演算をする：
Conjunction(**Conjunction**
(**Time-Sccessor**(**Conjunction**
$(D\prime, D_{I(l_k)})), D_{I(l_k)}), D_g)$
ただし，D_g はクロック制約 g の DBMs とする．

(5) 最後に，上記の標準形を求める．

ここで得られた t_{l_k} を用いて，$(l_k, u_k + t_{l_k}, \mathbf{Run}(q_k, t_{l_k}))$ を計算する．

以下では，t_{l_k} を場合別けして，$\mathbf{Run}(q_k, t_{l_k})$ を計算する．

ここで得られた t_{l_k} は，$t_{l_k} \leq d_{l_k}$，$t_{l_k} = d_{l_k}$，$t_{l_k} < d_{l_k}$，$t_{l_k} > d_{l_k}$，$t_{l_k} \geq d_{l_k}$ のいずれかである．さらに，d_{l_k} に関して，${c_{P_1}}^k$ との大小関係において，${c_{P_1}}^k \leq d_{l_k}$，${c_{P_1}}^k = d_{l_k}$，${c_{P_1}}^k < d_{l_k}$，${c_{P_1}}^k > d_{l_k}$，${c_{P_1}}^k \geq d_{l_k}$ の場合分けが必要である．

なお，時間オートマトンの意味のところで述べたように，t_{l_k} の最大値を考える．

これらの場合分けの様子を**図 4** に示す．

以下では，この場合分けに従って，$\mathbf{Run}(q_k, t_{l_k})$ を計算する．

ただし，リリースからの経過時間が T_{P_i} より大きくなると，システムエラーとなるために，つねに，経過時間をチェックする必要がある．

(1) $t_{l_k} \leq d_{l_k}$ のとき

このときは，${c_{P_i}}^k < d_{l_k}$，${c_{P_i}}^k = d_{l_k}$，${c_{P_i}}^k > d_{l_k}$ の 3 つの場合を考える．

(a) ${c_{P_1}}^k < d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
${P_1}^k(0, {d_{P_1}}^k - {c_{P_1}}^k, v_{P_1}({t_{P_1}}^k + {c_{P_1}}^k))$,
$\cdots\cdots\cdots$,
${P_i}^k(0, {d_{P_i}}^k - ({c_{P_1}}^k + \ldots + {c_{P_{i-1}}}^k)$,
$v_{P_i}({t_{P_i}}^k + ({c_{P_1}}^k + \ldots + {c_{P_{i-1}}}^k)))$,
${P_{i+1}}^k({c_{P_{i+1}}}^k - (d_{l_k} - ({c_{P_1}}^k + \ldots + {c_{P_i}}^k)), {d_{P_i}}^k - d_{l_k}, v_{P_i}({t_{P_{i+1}}}^k + d_{l_k}))$,
$\cdots\cdots\cdots$,
${P_n}^k({c_{P_n}}^k, {d_{P_n}}^k - d_{l_k}, v_{P_n}({t_{P_n}}^k + d_{l_k}))]$
ここでは，$t_{l_k} = d_{l_k}$ である．
ゆえに，価値は以下のようになる．
$v_{P_1}({t_{P_1}}^k + {c_{P_1}}^k)) + \ldots + v_{P_2}$
$({t_{P_2}}^k + {c_{P_1}}^k + \ldots + {c_{P_a}}^k))$
ただし，$a \leq n$ である．

(b) ${c_{P_1}}^k = d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
${P_1}^k(0, {d_{P_1}}^k - {c_{P_1}}^k, v_{P_1}({t_{P_1}}^k + {c_{P_1}}^k))$,
${P_2}^k({c_{P_2}}^k, {d_{P_2}}^k - {c_{P_1}}^k, v_{P_2}({t_{P_2}}^k + d_{l_k}))$,
$\cdots\cdots\cdots$,
${P_n}^k({c_{P_n}}^k, {d_{P_n}}^k, v_{P_n}({t_{P_n}}^k + d_{l_k}))]$
ここでは，$t_{l_k} = d_{l_k}$ である．
ゆえに，価値は以下のようになる．
$v_{P_1}({t_{P_1}}^k + {c_{P_1}}^k))$

(c) ${c_{P_1}}^k > d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
${P_1}^k({c_{P_1}}^k - d_{l_k}, {d_{P_1}}^k - d_{l_k}, v_{P_1}({t_{P_1}}^k + d_{l_k}))$,
${P_2}^k({c_{P_2}}^k, {d_{P_2}}^k, v_{P_2}({t_{P_2}}^k + d_{l_k}^k))$,
$\cdots\cdots\cdots$,
${P_n}^k({c_{P_n}}^k, {d_{P_n}}^k, v_{P_n}({t_{P_n}}^k + d_{l_k}))]$
このとき，どのタスクの残実行時間も 0 にならないので，価値の計測はない．
ここでは，$t_{l_k} = d_{l_k}$ である．

(2) $t_{l_k} < d_{l_k}$ のとき：
このときは，${c_{P_i}}^k < d_{l_k}$，${c_{P_i}}^k \geq d_{l_k}$ の 2 つの場合を考える．

(a) ${c_{P_1}}^k < d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
${P_1}^k(0, {d_{P_1}}^k - {c_{P_1}}^k, v_{P_1}({t_{P_1}}^k + {c_{P_1}}^k))$,
$\cdots\cdots\cdots$,

$P_i{}^k(0, d_{P_i}{}^k - (c_{P_1}{}^k + \ldots + c_{P_{i-1}}{}^k),$
$v_{P_i}(t_{P_i}{}^k + (c_{P_1}{}^k + \ldots + c_{P_{i-1}}{}^k))),$
$P_{i+1}{}^k(c_{P_{i+1}}{}^k - (d_{l_k}{}^- - (c_{P_1}{}^k + \ldots +$
$c_{P_i}{}^k)), d_{P_i}{}^k - d_{l_k}{}^-, v_{P_i}(t_{P_{i+1}}{}^k + d_{l_k}{}^-)),$
$\ldots\ldots\ldots,$
$P_n{}^k(c_{P_n}{}^k, d_{P_n}{}^k - d_{l_k}{}^-,$
$v_{P_n}(t_{P_n}{}^k + d_{l_k}{}^-))]$
ただし，$d_{l_k}{}^-$ は d_{l_k} に限りなく近く d_{l_k} よりも小さな数字である．
ここでは，$t_{l_k} = d_{l_k}{}^-$ である．
ゆえに，価値は以下のようになる．

$$v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)) + v_{P_2}(t_{P_2}{}^k + c_{P_1}{}^k + \ldots + c_{P_b}{}^k))$$

ただし，$b \leq n$ である．

(b) $c_{P_1}{}^k \geq d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
$P_1{}^k(c_{P_1}{}^k - d_{l_k}{}^-, d_{P_1}{}^k - d_{l_k}{}^-, v_{P_1}(t_{P_1}{}^k + d_{l_k}{}^-),$
$P_2{}^k(c_{P_2}{}^k, d_{P_2}{}^k - d_{l_k}{}^-,$
$v_{P_2}(t_{P_2}{}^k + d_{l_k}{}^-)),$
$\ldots\ldots\ldots,$
$P_n{}^k(c_{P_n}{}^k, d_{P_n}{}^k - d_{l_k}{}^-,$
$v_{P_n}(t_{P_n}{}^k + d_{l_k}{}^-))]$
このとき，どのタスクの残実行時間も 0 にならないので，価値の計測はない．
ここでは，$t_{l_k} = d_{l_k}{}^-$ である．

(3) $t_{l_k} = d_{l_k}$ のとき
このときは，$c_{P_i}{}^k < d_{l_k}$，$c_{P_i}{}^k = d_{l_k}$，$c_{P_i}{}^k > d_{l_k}$ の 3 つの場合を考える．

(a) $c_{P_1}{}^k < d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
$P_1{}^k(0, d_{P_1}{}^k - c_{P_1}{}^k, v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)),$
$\ldots\ldots\ldots,$
$P_i{}^k(0, d_{P_i}{}^k - (c_{P_1}{}^k + \ldots + c_{P_{i-1}}{}^k),$
$v_{P_i}(t_{P_i}{}^k + (c_{P_1}{}^k + \ldots + c_{P_{i-1}}{}^k))),$
$P_{i+1}{}^k(c_{P_{i+1}}{}^k - (d_{l_k} - (c_{P_1}{}^k + \ldots +$
$c_{P_i}{}^k)), d_{P_i}{}^k - d_{l_k}, v_{P_i}(t_{P_{i+1}}{}^k + d_{l_k})),$
$\ldots\ldots\ldots,$
$P_n{}^k(c_{P_n}{}^k, d_{P_n}{}^k - d_{l_k}, v_{P_n}(t_{P_n}{}^k + d_{l_k}))]$
ここでは，$t_{l_k} = d_{l_k}$ である．
ゆえに，価値は以下のようになる．

$$v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)) + v_{P_2}(t_{P_2}{}^k + c_{P_1}{}^k + \ldots + c_{P_c}{}^k))$$

ただし，$c \leq n$ である．

(b) $c_{P_1}{}^k = d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
$P_1{}^k(0, d_{P_1}{}^k - c_{P_1}{}^k, v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)),$
$P_2{}^k(c_{P_2}{}^k, d_{P_2}{}^k - c_{P_1}{}^k, v_{P_2}(t_{P_2}{}^k + d_{l_k})),$
$\ldots\ldots\ldots,$
$P_n{}^k(c_{P_n}{}^k, d_{P_n}{}^k, v_{P_n}(t_{P_n}{}^k + d_{l_k}))]$
ここでは，$t_{l_k} = d_{l_k}$ である．
ゆえに，価値は以下のようになる．

$$v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k))$$

(c) $c_{P_1}{}^k > d_{l_k}$ のとき：
この場合は，以下のようなキューになる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
$P_1{}^k(c_{P_1}{}^k - d_{l_k}, d_{P_1}{}^k - d_{l_k}, v_{P_1}(t_{P_1}{}^k + d_{l_k}),$
$P_2{}^k(c_{P_2}{}^k, d_{P_2}{}^k - d_{l_k}, v_{P_2}(t_{P_2}{}^k + d_{l_k})),$
$\ldots\ldots\ldots,$
$P_n{}^k(c_{P_n}{}^k, d_{P_n}{}^k - d_{l_k}, v_{P_n}(t_{P_n}{}^k + d_{l_k}))]$
このとき，どのタスクの残実行時間も 0 にならないので，価値の計測はない．
ここでは，$t_{l_k} = d_{l_k}$ である．

(4) $t_{l_k} \geq d_{l_k}$ のとき
$c_{P_i}{}^k$ と d_{l_k} との大小関係にかかわらず，以下のようなキューとなる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
$P_1{}^k(0, d_{P_1}{}^k - c_{P_1}{}^k, v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)),$
$P_2{}^k(0, d_{P_2}{}^k - (c_{P_1}{}^k + c_{P_2}{}^k), v_{P_2}(t_{P_2}{}^k + c_{P_1}{}^k + c_{P_2}{}^k)),$
$\ldots\ldots\ldots,$
$P_n{}^k(0, d_{P_n}{}^k, -(c_{P_1}{}^k + \ldots + c_{P_n}{}^k), v_{P_n}(t_{P_n}{}^k + c_{P_1}{}^k + \ldots + c_{P_n}{}^k))]$
ここでは，$t_{l_k} = c_{P_1}{}^k + \ldots + c_{P_n}{}^k$ である．
ゆえに，価値は以下のようになる．

$$v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)) + \ldots + v_{P_n}(t_{P_n}{}^k + c_{P_1}{}^k + \ldots + c_{P_n}{}^k))$$

(5) $t_{l_k} > d_{l_k}$ のとき
$c_{P_i}{}^k$ と d_{l_k} との大小関係にかかわらず，以下のようなキューとなる．
$\mathbf{Run}(q_k, t_{l_k}) = [$
$P_1{}^k(0, d_{P_1}{}^k - c_{P_1}{}^k, v_{P_1}(t_{P_1}{}^k + c_{P_1}{}^k)),$
$P_2{}^k(0, d_{P_2}{}^k - (c_{P_1}{}^k + c_{P_2}{}^k), v_{P_2}(t_{P_2}{}^k + c_{P_1}{}^k + c_{P_2}{}^k)),$
$\ldots\ldots\ldots,$
$P_n{}^k(0, d_{P_n}{}^k, -(c_{P_1}{}^k + \ldots + c_{P_n}{}^k), v_{P_n}(t_{P_n}{}^k + c_{P_1}{}^k + \ldots + c_{P_n}{}^k))]$
ここでは，$t_{l_k} = c_{P_1}{}^k + \ldots + c_{P_n}{}^k$ である．

ゆえに，価値は以下のようになる．

$$v_{P_1}({t_{P_1}}^k + {c_{P_1}}^k)) + \ldots + v_{P_2}({t_{P_2}}^k + {c_{P_1}}^k + \ldots + {c_{P_n}}^k))$$

2．離散遷移

次に，離散遷移を計算する．

先ほどの時間遷移後には，$l \xrightarrow{g,a,r} m$ かつ $u \models g$ は成り立つので，以下の離散遷移が起きる．

$(l, u, q) \xrightarrow{g,a,r}_{Sch} (m, u[r \mapsto 0], \mathbf{Sch}(M(m) :: q))$

ここで，$u[r \mapsto 0]$ はリセット演算 $\mathbf{Reset}(D[r := 0])$ で容易に計算できて，$\mathbf{Sch}(M(m) :: q)$ もノード m に割り当てられているタスク q をキューに追加すればよい．

初期ノードよりノードごとに段階的に，以上の**時間遷移**と**離散遷移**に関する計算を繰り返すことにより，計算木の決定性の動作ごとに，$\sum_{i=1}^{n} v_{P_i}({t_{P_i}}^k)$ が計算できる．

■

4.2 性能解析の実例

ここでは，**Example2** を用いて，上記アルゴリズムの実例の一部を示す．以下では，図 3 の時間オートマトンを対象とする．

(1) m_1 における初期の DBMs について：

(a) 最初は，$x = 0$，$t_{m_1} = 0$ であり，DBMs D は以下のとおりである：

$$\begin{pmatrix} \infty & (0,\leq) & (0,\leq) \\ (0,\leq) & \infty & \infty \\ (0,\leq) & (0,\leq) & \infty \end{pmatrix}$$

(b) この DBMs の標準形 $\mathbf{CF}(D)$ は以下のとおりである：

以降では，これを $D\prime$ とする．

$$\begin{pmatrix} (0,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

(2) ノード m_1 における時間遷移：

(a) まず，ノード m_1 でとりうる DBMs **Conjunction**$(D\prime, D_{I(m_1)})$ を作る：

$$\begin{pmatrix} (0,\leq) & (0,\leq) & (0,\leq) \\ (10,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

ただし，$D_{I(m_1)}$ はノード m_1 に割り付いているタイミング制約である．

(b) 次に，ノード m_1 でとりうる DBMs が時間前進した DBMs を作る：

Time-Sccessor$(\mathbf{Conjunction}(D\prime, D_{I(m_1)}))$

$$\begin{pmatrix} (0,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

(c) 次に，ノード m_1 でとりうる DBMs の時間前進した DBMs が実際にノード m_1 でとりうるように，$D_{I(m_1)}$ と積演算する：

Conjunction(**Time-Sccessor**$(\mathbf{Conjunction}(D\prime, D_{I(m_1)}))$, $D_{I(m_1)})$

$$\begin{pmatrix} (0,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

(d) 次に，ノード m_1 から遷移する状態への枝に付いているクロック制約 g により，上記の DBMs を制限する必要があるので，積演算をする：

Conjunction(**Conjunction**(**Time-Sccessor**(**Conjunction**$(D\prime, D_{I(m_1)}))), D_{I(m_1)}), D_g)$

ただし，D_g はクロック制約 g の DBMs とする．

$$\begin{pmatrix} (0,\leq) & (0,\leq) & (0,\leq) \\ (-10,\leq) & (0,\leq) & (0,\leq) \\ (0,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

(e) 最後に，上記の標準形を求める：

$$\begin{pmatrix} (0,\leq) & (10,\leq) & (10,\leq) \\ (-10,\leq) & (0,\leq) & (0,\leq) \\ (-10,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

以上より，$t_{m_1} = 10$ である．

(3) ノード m_1 からの離散遷移：

次に，x をリセットして n_1 に離散遷移して，タスク P_1 をキューに追加する．

同様に，ノード n_1 の時間経過と離散遷移を計算する．

以上より，以下が得られる：

$(m_1, [x = 0], [Q_1(1, 2, v_{Q_1}(0))])$
$\xrightarrow{1}_{Sch} (m_1, [x = 1], [Q_1(0, 1, v_{Q_1}(1))])$

$$
\begin{aligned}
&= (m_1, [x = 1], [])\\
&\xrightarrow{9}_{Sch} (m_1, [x = 10], [])\\
&\xrightarrow{a_1}_{Sch} (n_1, [x = 0], [P_1(4, 20, v_{P_1}(0)]])\\
&\xrightarrow{4}_{Sch} (n_1, [x = 4], [P_1(0, 16, v_{P_1}(4)]])\\
&= (n_1, [x = 4], [])\\
&\cdots\cdots\cdots\cdots\cdots\\
&\cdots\cdots\cdots\cdots\cdots
\end{aligned}
$$

ここで，価値 $v_{Q_1}(1) + v_{P_1}(4)$ が得られた．

ここでは，ノード m_1 とノード n_1 の価値を計算した．

同様な手順により，m_2 と n_2 の価値も計算して，初期ノード m_1 に戻るまでの価値を合計することにより，最終的な価値を求める．

5. ま と め

本論文では，時間オートマトン上でのハードリアルタイムシステムのスケジューラビリティ検証手法[5),6)]を拡張して，時間オートマトン上のソフトリアルタイムシステムの性能解析手法を開発する．

今後の課題として，現在，以下の研究を進めている：

(1) 性能解析システムを実装する．

(2) 本手法を実問題へ適用する．

謝辞 本研究は財団法人国際コミュニケーション基金の援助の下で実施されました．

参 考 文 献

1) Tilborg, A.M. and Koob, G.M.: *Foundations of Real-time Computing: Formal Specifications and Methods*, p.316, Kluwer Academic Pub. (1991).
2) Liu, J.W.S.: *Real-Time Systems*, Prentics-Hall (2000).
3) Joseph, M. (Ed.): *Real-time Systems: Specification, Verification and Analysis*, Prentics-Hall (1996).
4) Buttazzo, G.C.: *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, The Kluwer International Series in Engineering and Computer sci (1997).
5) Ericsson, C., Wall, A. and Yi, W.: Timed Automata as Task Models for Event driven Systems, *Proc. RTSCA 99*, pp.182–189, IEEE Computer Society Press (1999).
6) Fersman, E., Pettersson, P. and Yi, W.: Timed Automata with Asynchrounous Processes: Schedulability and Decidability, *LNCS 2280*, pp.67–82 (2002).
7) McManis, J. and Varaiya, P.: Suspension automata: A decidable class of hybrid automata, *LNCS 818*, pp.105–117 (1994).
8) Corbett, J.C.: Modeling and analysis of real-time Ada tasking programs, *Proc. Real-Time Systems Symposium*, pp.132–141, IEEE Computer Society (1994).
9) Cassez, F. and Laroussinie, F.: Model-checking for hybrid systems by quotienting and constraints solving, *LNCS 1855*, pp.373–388 (2000).
10) Alur, R., Coucoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicolin, X., Olivero, A., Sifakis, J. and Yovine, S.: The algorithmic analysis of hybrid systems, *Theoretical Computer Science*, Vol.138, pp.3–34 (1995).
11) Bouyer, P., Dufourd, C., Fleury, E. and Petit, A.: Expressiveness of updatable timed automata, *LNCS 1893*, pp.232–242 (2000).
12) Fersman, E., Mokrushin, L., Pettersson, P. and Yi, W.: Schedulability Analysis Using Two Clocks, *LNCS 2619*, pp.224–239 (2003).
13) Jensen, E.D., Locke, C.D. and Tokuda, H.: A time-driven scheduling model for realtime operating systems, *Real-Time Systems Symposium*, pp.112–122 (1985).
14) Kao, B. and Garcia-Molina, H.: Deadline Assignment in a Distributed Soft Real-Time Systems, *Proc. 13th International Conference on Distributed Computing Systems*, pp.428–437 (1993).
15) Kao, B., Garcia-Molina, H. and Adelberg, B.: On building distributed soft real-time systems, *3th Workshop on Parallel and Distributed Real-Time Systems*, pp.13–19 (1995).
16) Kavi, K.M., Youn, H.Y., Shirazi, B. and Hurson, A.R.: A Performability Model for Soft Real-Time Systems, *Proc. 27th Hawaii International Conference on System Sciences*, pp.571–579 (1994).
17) Gardner, M.K.: Probabilstic Analysis and Scheduling of Critical Soft Real-time Systems, Ph.D. thesis, University of Illinois, Computer Science, Urbana, Illinois (1999).
18) Hansson, H. and Jonsson, B.: A calculus for communicating systems with time and probabilities, *Proc. 11 IEEE Symposium on Real-Time Systems*, pp.278–287 (1990).
19) Alur, R., Courcoubetis, C. and Dill, D.L.: Model-checking for probabilistic real-time systems, *LNCS 510*, pp.115–136 (1991).
20) Kwiatkowska, M., Norman, G., Segala, R. and Sproston, J.: Automatic Verification of Real-Time Systems With Discrete Probability Distributions, *LNCS 1601*, pp.75–95 (1999).
21) D'Argenio, P.R., Katoen, J.-P. and Brinksma,

E.: Specification and Analysis of Soft RealTime Systems: Quantity and Quality, *Proc. 20th IEEE Real-Time Systems Symposium*, pp.104–114 (1999).
22) Tindell, K.: Fixed-Priority Scheduling of Hard Real-Time Systems, Ph. D. thesis, University of York, UK (1994).
23) Henzinger, T.A., Nicollin, X., Sifakis, J. and Yovine, S.: Symbolic model checking for real-time systems, *Information and Computation*, Vol.111, pp.193–244 (1994).
24) 白川洋充，竹垣盛一：リアルタイムシステムとその応用，システム制御情報ライブラリー，朝倉書店 (2001).
25) Alur, R., Courcoubetis, C. and Dill, D.L.: Model-Checking in Dense Real-Time, *Information and Computation*, Vol.104, pp.2–34 (1993).
26) Dill, D.L.: Timing assumptions and veri cation of finite-state concurrent systems, *LNCS 407*, pp.197–212 (1989).
27) Alur, R., Courcoubetis, C., Dill, D.L., Halbwachs, N. and Wong-Toi, H.: An implementation of three algorithms for timing verification based on automata emptiness, *Proc. 13th IEEE Real-Time Systems Symposium*, pp.157–166 (1992).
28) Alur, R.: Timed Automata, *LNCS 1633*, pp.8–22 (1999).
29) Yovine, S.: Model Checking Timed Automata, *LNCS 1494*, pp.114–152 (1996).
30) Bengtsson, J. and Yi, W.: Timed Automata: Semantics, Algorithms and Tools, *LNCS 3098*, pp.87–124 (2004).
31) 山根 智，中村一博：実時間システムのための近似手法に基づいた記号モデル検査器の開発と評価. 電子情報通信学会論文誌，Vol.J86-D1，pp.232–247 (2003).

(平成 17 年 3 月 3 日受付)
(平成 17 年 9 月 2 日採録)

山根　智（正会員）

1984 年，京都大学大学院修了．現在，金沢大学大学院自然科学研究科電子情報科学専攻教授．リアルタイム・ハイブリッドシステムの仕様記述と形式的検証の研究に従事．EATCS，ACM，IEEE 等各会員．