招待論文

時間オートマトンのモデル検査

中田　明夫[†]

Model Checking of Timed Automata

Akio NAKATA[†]

あらまし　実時間システムの検証技術の一分野である時間オートマトンのモデル検査技術について，到達可能性解析に焦点を絞り概説する．まず，時間オートマトンの構文と意味の定義を述べ，時間オートマトンの到達可能性解析技術について，特にそのキーアイデアであるクロックリージョンおよびクロックゾーンの概念に重点をおいて解説する．最後に，時間オートマトンのモデル検査技術に関する近年の研究動向について簡単に紹介する．

キーワード　実時間システム，検証技術，時間オートマトン，モデル検査，到達可能性解析

1. まえがき

近年，航空機・自動車やFA機器から携帯電話・情報家電など広範囲な分野でハードウェア/ソフトウェアシステムが使用されてきている．それらのシステムの多くは設計誤りが人命にかかわったり多大な損害につながるなどのため，高い信頼性が要求される．一方で，これらのシステムの多くは定められた時間制約内に動作を行うことを要求される実時間システムとなっている．そのため，高信頼な実時間システムの設計開発技術への要求が高まっている．高信頼なシステムの設計開発技術の一つとしてモデル検査 [1] が注目を集めている．

モデル検査はシステムを状態遷移機械でモデル化し，システムの振る舞いの正しさを自動的に検証する手法である．振る舞いの正しさとは，外界と相互作用しながら動作し続けるシステム（リアクティブシステム）の正しさの基準の一つであり，デッドロックなどの望ましくない状態に決して陥らない（安全性）や，いつかは必ず望ましい動作を行う（活性）などといった性質を満たすことを指す．特に実時間システムにおいては動作のタイミングの正しさも含まれる．モデル検査では，システムのモデル，および，振る舞いの正しさを定義した検証性質から，モデルが検証性質を満たすか否かを判定する．モデル検査はモデルと性質の記述から全自動で判定できるため定理証明ほど専門知識を要さず，手軽に用いることが出来るという利点がある．現在までにさまざまなモデル検査技術が提案されており，実時間システムを対象としたものもいくつか提案されてきている．実時間システムのモデルとしては時間オートマトン [2] が標準的に良く用いられており，他の実時間システムのモデル，例えば時間ペトリネット (timed Petri nets) [3] や，スケジューリング理論で用いられるリアルタイムタスクモデル [4] を時間オートマトンに変換する手法も提案されている [5], [6]．本稿では，時間オートマトンのモデル検査技術の基礎について概説し，最新の研究動向の一部について紹介する．

2. 時間オートマトン

$\mathbb{N}$ を (0 を含む) 自然数全体の集合，$\mathbb{R}$ を実数全体の集合，$\mathbb{R}^+$ を非負実数全体の集合とする．**クロック変数 (clock variable)** とは，非負実数値を領域とする状態変数のことであると定義する．クロック変数の有限集合を C とする．$x \in C$，$n \in \mathbb{N}$ のとき，$x \sim n$ ($\sim \in \{<, \leqq, >, \geqq, =\}$) の形の原子述語の論理積で構成される論理式を C 上の**クロック制約 (clock constraint)** と呼ぶ．C 上のクロック制約の全体集合を $CC(C)$ とする．

時間オートマトン (timed automaton) [2], [7], [8][(注1)]は 6 つ組 $M = (S, Act, C, E, Inv, s_0)$ で

[†] 広島市立大学 大学院情報科学研究科
Graduate School of Information Sciences, Hiroshima City University

(注1)：時間オートマトンの定義にはさまざまなバリエーションが存在するが，ここでは，[2] にて初めて定義された時間オートマトン (Alur-Dill Timed Automaton) に，ロケーション不変条件 [7] を追加した定義

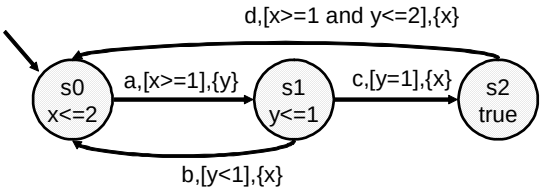

図 1　時間オートマトンの例
Fig. 1　Example of a Timed Automaton

定義される．ここで，S は**ロケーション (location)** の有限集合，Act はアクションの有限集合，C はクロック変数の有限集合，$E \subseteqq S \times Act \times CC(C) \times 2^C \times S$ は遷移関係，$Inv : S \mapsto CC(C)$ は**ロケーション不変条件 (location invariant)**，$s_0 \in S$ は**初期ロケーション (initial location)** である．M において遷移 $(s, a, g, r, s') \in E$ が存在するとき，$s \xrightarrow{a,g,r} s'$ と書くことにする．$g \in CC(C)$ を遷移 $s \xrightarrow{a,g,r} s'$ の**ガード条件 (guard condition)**，$r \in 2^C$ を**リセット集合 (reset set)** と呼ぶ．

時間オートマトンの直観的な動作意味は次の通りである．時間オートマトン M は，まず，ロケーションが初期ロケーション s_0 であり，全てのクロック変数の値は 0 である状態から動作を開始する．いま，時間オートマトン M のロケーションが s であるとする．このとき，M は s のロケーション不変条件 $Inv(s)$ を満たしている限り，同じロケーション s に留まることができる．同じロケーション s に留まっている間，全てのクロック変数 $x \in C$ は時間経過とともに同じ速さで連続的に値が増加するものとする．

現在の各クロック変数の値において遷移 $s \xrightarrow{a,g,r} s'$ のガード条件 g が真となるときにアクション a が実行可能で，実行後は瞬時にロケーションは s' に遷移し，この遷移のリセット集合 r に指定したクロック変数 $y \in r$ が全て 0 にリセットされる．

［例 1］　時間オートマトンの例を図 1 に示す．図中の丸はロケーションを表し，内部の s_0, s_1, s_2 はロケーション名である．初期ロケーションは s_0 で，各ロケーションに付随するクロック制約はロケーション不変条件である．遷移枝に付随するラベルは，それぞれアクション名，ガード条件，リセット集合である．　□

形式的には，時間オートマトンの動作意味は次のように定義される．C から非負実数 $\mathbb{R}^+$ への写像 $\sigma : C \mapsto \mathbb{R}^+$ をクロック変数の集合 C への**付値 (valuation)** と呼ぶ．C の付値の集合を $Val(C)$ と書く．付値 $\sigma \in Val(C)$ がクロック制約 $g \in CC(C)$ を満たすことを $\sigma \models g$ と書く．任意の付値 $\sigma \in Val(C)$ および $d \in \mathbb{R}^+$ に対して，付値 $\sigma + d$ を，$\sigma(x) = v$ ならば $(\sigma + d)(x) = v + d$ を満たすものと定義する．また，任意の $r \subseteqq C$ に対して，$\sigma[r \mapsto 0]$ を $x \in r$ ならば $\sigma[r \mapsto 0](x) = 0$, さもなければ $\sigma[r \mapsto 0](x) = \sigma(x)$ を満たすものと定義する．このとき，時間オートマトン $M = (S, Act, C, E, Inv, s_0)$ の意味モデルは一般に無限状態の状態遷移モデル $T(M) = (S \times Val(C), Act \cup \mathbb{R}^+, E', (s_0, \sigma_0))$ で定義される．ただし，$E' \subseteqq ((S \times Val(C)) \times (Act \cup \mathbb{R}^+) \times (S \times Val(C)))$ である．$T(M)$ を M に対応する**時間遷移システム (timed transition system)** と呼ぶ．$T(M)$ の状態は M のロケーション $s \in S$ と C への付値 $\sigma \in Val(C)$ の組 (s, σ) である．これを**具体的状態 (concrete state)** と呼ぶことにする．具体的状態の集合 $S \times Val(C)$ を CS と略記する．任意のクロック変数 $x \in C$ に対して $\sigma_0(x) = 0$ となる付値 σ_0 を初期クロック付値とする．$T(M)$ の**初期状態 (initial state)** は M の初期ロケーション s_0 と初期クロック付値 σ_0 の組 (s_0, σ_0) である．$T(M)$ の遷移 E' には $\mathbb{R}^+$ または Act いずれかの遷移ラベルが付加されており，それぞれ，**遅延遷移 (delay transition)**，**アクション遷移 (action transition)** と呼ぶ．それぞれの遷移の定義は以下の通りである：

- 遅延遷移はロケーションを変えずに時間のみが経過する状態遷移であり，M の状態 s，付値 $\sigma \in Val(C)$，および，遅延量 $d \in \mathbb{R}^+$ に対して，もし任意の $0 \leqq e \leqq d$ に対して $\sigma + e \models Inv(s)$ ならば，$T(M)$ の状態 (s, σ) から，d をラベルに持つ遷移 $(s, \sigma) \xrightarrow{d} (s, \sigma + d)$ が実行可能，すなわち，$((s, \sigma), d, (s, \sigma + d)) \in E'$ であると定義する．
- アクション遷移は時間経過なしにアクションを瞬時に実行し，ロケーションを変える遷移であり，M の状態 s，付値 $\sigma \in Val(C)$，および遷移 $s \xrightarrow{a,g,r} s'$ に対して，もし $\sigma \models g \wedge Inv(s')$ ならばアクション $a \in Act$ をラベルに持つ遷移 $(s, \sigma) \xrightarrow{a} (s', \sigma[r \mapsto 0])$ が実行可能，すなわち，$((s, \sigma), a, (s', \sigma[r \mapsto 0])) \in E'$ であると定義する．

M の時間遷移システム $T(M)$ の任意の具体的状態 $q \in CS$ および遷移ラベル $\alpha \in Act \cup \mathbb{R}^+$ に対して，もし $q \xrightarrow{\alpha} q_1$ かつ $q \xrightarrow{\alpha} q_2$ ならば常に $q_1 = q_2$ であるとき，M は**決定性 (deterministic)** であると呼び，

(Timed Safety Automaton と呼ばれる) を紹介する．

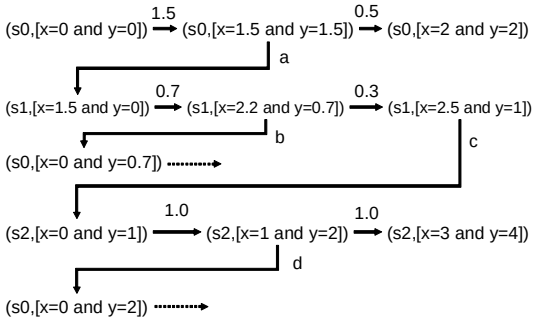

図 2　図 1 に対応する時間遷移システム
Fig. 2　Timed Transition System Corresponding to Fig. 1

さもなければ**非決定性 (nondeterministic)** であると呼ぶ.

［例 2］　例 1 の動作意味は図 2 に示す時間遷移システムで表される．図中の $(s_0, [x = 0 \text{ and } y = 0])$ は，ロケーションが s_0，$C = \{x, y\}$ への付値が $x = 0$ および $y = 0$ であるような具体的状態を表す．図中の矢印は遷移を表し，ラベルとして時間経過を表す実数値またはアクション名が記述されている．図では水平方向の矢印が遅延遷移，垂直方向の遷移がアクション遷移となるように記載している．点線の矢印は遷移先状態を省略したことを表す．遅延遷移に関しては図では特定の時間値による遷移しか示していないが，実際には (ロケーション不変条件を満たす範囲での) 任意の実数値に関する遅延遷移が無数に存在する．　□

3. 時間オートマトンのモデル検査

モデル検査問題とは，システムのモデル M およびシステムの振る舞いに関する性質 ϕ が与えられたときに，M が ϕ を満たすか否かを判定する問題である．性質 ϕ としてはさまざまなものが考えられるが，ここでは，最も単純な性質である安全性 (safety) の検証に焦点を絞って述べる．

安全性とは端的にいえば，モデル M が「悪い状態に到達しない」という性質のことである．悪い状態としては例えば並行システムにおけるデッドロック状態，あるいは，排他的な資源の利用が競合した状態，などが挙げられる．

安全性の検証は到達可能性解析に帰着して行うことができる．すなわち，モデル M が有限状態機械であれば，その状態遷移グラフに対して，初期状態から悪い状態に到達可能か否かをグラフ探索アルゴリズム（例えば，深さ優先探索や幅優先探索など [9]）によって判定すればよい．

時間オートマトンにおいて到達可能性解析を行う際に問題となるのは，時間のとりうる値，すなわち，時間領域 (time domain) が稠密な値（実数値）であるため，状態数が無限となる点である．稠密な時間領域のことを稠密時間 (dense time) と呼ぶ．時間オートマトン $M = (S, s_0, Act, C, E, Inv, s_0)$ の実際の動きは $T(M)$ で表されるため，到達可能性問題は $T(M)$ において考えることになる．このとき，$T(M)$ の状態 (具体的状態)(s, σ) $(s \in S\ \sigma \in Val(C))$ の総数は一般に非可算無限個となる．計算機で自動的に解析を行うためには，何らかの方法で時間オートマトンの無限個の状態を有限的に扱う必要がある．

3.1　クロックリージョン

時間オートマトンのような稠密時間モデルを有限状態に落とし込む方針にはいくつかある．一つは時間領域を離散値 (離散時間 (discrete time)) に制限することである．　もしクロック制約やロケーション不変条件の原子述語が $x \leqq c (x \in C, c \in \mathbb{N})$ という形のみで構成されるなら，この方法で有限状態モデルに変換できる．しかし，クロック変数の値に上限が無い場合はそのままでは有限状態にならない．この問題自体は，クロック変数の値がある値以上になれば，それ以上の変化は無視する，という方針で等価な有限状態モデルに落とし込むことができるので，それほど重大な問題ではない．しかし，もしシステムが単体の計算機のみから構成されるものでなければ，離散時間モデルがシステムの動作を忠実に表現したものでなくなるおそれがある．あるいは，制御対象の物理量を入力とするシステムの場合は，離散時間モデルは一般に不適切となる．前者の典型的な例はネットワーク接続された計算機群で構成された分散システムであり，後者はエンジンや化学プラントなどの制御システム，あるいは非同期ハードウェアなどである．このとき，あるイベントの発生時刻の離散値による近似がたとえ同一であったとしても実際には前後している可能性がある．例えば資源の調停に関して，誰の資源獲得要求メッセージが先に到着したかによって，その後の動作が大きく異なるなど，イベントの発生順序は計算機の動作タイミングのみならず，一般に動作内容自体に影響する．従ってたとえ時間差が計測不能なほど近いものであってもイベントの順序の情報はモデル検査においては正確に扱う必要がある．

稠密時間モデルとしての時間オートマトンを有限状態に落とし込むもう一つの方針は，時間オートマトン $M = (S, s_0, Act, C, E, Inv, s_0)$ の具体的状態の集合 $S \times Val(C)$ 上に，検証性質を保存するような何らかの同値関係 (等価性) を導入し，有限個の同値類に分割することである．

この方針による手法として最初に考えられたものは**リージョンオートマトン (region automaton)** [2] による方法である．リージョンオートマトンのアイデアは，今後の動作の遷移条件の成否に影響しないようなクロック変数の付値の違いを無視するという等価性によって，付値の集合を**リージョン (region)** と呼ばれる部分集合に同値類分割する点にある．時間オートマトンにおいては，任意の遷移条件においてクロック変数と比較される定数は整数のみとしている．もし有理数定数との比較の必要がある場合も，時間オートマトン全体においてガード条件に現れる定数の総数は高々有限個であるので，各定数値を適切にスケーリング（例えば全ての定数の分母の最小公倍数で乗じるなど）すれば全て整数にすることができる．ガード条件中の定数として無理数を指定する必要性はほとんど無いと考えられるので，時間オートマトンの定数を整数のみに制限しても十分な一般性を持っているといえる．このとき，各クロック変数への付値の整数部分が一致し，小数部分の大小関係が等しいような 2 つの付値 σ,σ' に対して，時間オートマトンの具体的状態 (s, σ)，(s, σ') は，時間の経過量を無視して実行可能なアクションの系列のみを考慮すると，共に同じ動作を行う．また，各クロック変数 $x \in C$ に対して，ガード条件またはロケーション不変条件において x と比較される最大の整数を $k(x)$ とすると，x の値が $k(x)$ を超えた時点で，x の具体的な値は今後の時間オートマトンの動作に影響しないので，$x > k(x)$ であるという情報以外は無視できる．

上記の方針に基づいて，まず，クロック変数への付値の集合 $Val(C)$ に対して次のような等価性，**リージョン等価性 (region equivalence)** [2] を導入する．クロック変数 $x \in C$ に対して，x と比較される最大の整数値 $k(x)$ を対応させる写像を k とする．k を**クロック上限 (clock ceiling)** と呼ぶ．任意の実数 $d \in \mathbb{R}$ に対して，d の小数部分を $fract(d)$ と書く．すなわち，$fract(d) \stackrel{\text{def}}{=} d - \lfloor d \rfloor$ とする．C への付値 $\sigma, \sigma' \in Val(C)$ が k に関して**リージョン等価 (region equivalent)** であるとは，以下の条件を全て満たすこ

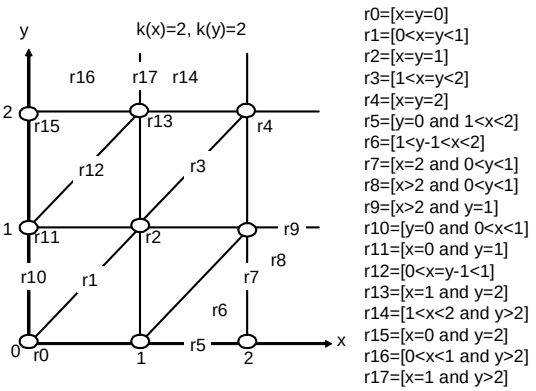

図 3　$C = \{x, y\}$, $k(x) = k(y) = 2$ の場合のクロックリージョン

Fig. 3　Clock Regions for $C = \{x, y\}$ and $k(x) = k(y) = 2$

とであると定義し，$\sigma \sim_k \sigma'$ と記述する：

（1） 任意の $x \in C$ に対して，$\lfloor \sigma(x) \rfloor = \lfloor \sigma'(x) \rfloor$ であるか，$\sigma(x) > k(x)$ かつ $\sigma'(x) > k(x)$ であるかのいずれかが成り立つ．

（2） 任意の $x \in C$ に対して，$fract(\sigma(x)) = 0$ であることと $fract(\sigma'(x)) = 0$ であることが同値である．

（3） 任意の $x, y \in C$ に対して，もし $\sigma(x) \leqq k(x)$ かつ $\sigma(y) \leqq k(y)$ ならば，$fract(\sigma(x)) \leqq fract(\sigma(y))$ であることと $fract(\sigma'(x)) \leqq fract(\sigma'(y))$ であることが同値である．

リージョン等価性が有限位数となることは容易に示される．

リージョン等価性 $\sim_k$ によるクロック変数への付値 $\sigma \in Val(C)$ の同値類を**クロックリージョン (clock region)** あるいは単に**リージョン (region)** と呼び，付値 σ を含むリージョンを $[\sigma]_k$ と書く．k が明らかな場合は，これを省略して単に $[\sigma]$ と書くことにする．リージョン等価性が有限位数であることより，リージョンの総数は有限となる．

[例 3]　$C = \{x, y\}$,$k(x) = k(y) = 2$ の場合のクロックリージョンは図 3 のようになる．クロック変数 2 つの場合，各リージョンは 2 次元平面の格子点，格子点によって区切られた線分または半直線の（格子点を含まない）内部，および，線分または半直線で分割された領域（境界を含まず）で表される．図中に $r_1, \ldots, r_{17}$ で示した一部のリージョンについては，右側にそれらが表す領域をクロック変数に関する制約式の形で示し

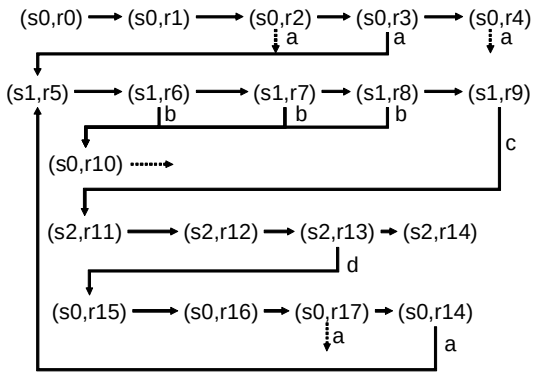

図 4　図 1 に対するリージョンオートマトン
Fig. 4　Region Automaton of the Timed Automaton in Fig. 1

ている．□

時間オートマトンの具体的状態 (s,σ) に対して，$(s,[\sigma])$ を (リージョン等価性に関する) 記号的状態 (symbolic state) と呼ぶ．**リージョンオートマトン (region automaton)** [2] とはリージョン等価性に関する記号的状態間の状態遷移モデルであり，その遷移関係 $\overset{\alpha}{\leadsto}_r(\alpha \in \{\epsilon\} \cup Act)$ が以下のように定義されるものであるとする：

- ある $d \in \mathbb{R}^+$ に対して，もし $(s,\sigma) \xrightarrow{d} (s,\sigma')$ ならば，$(s,[\sigma]) \overset{\epsilon}{\leadsto}_r (s,[\sigma'])$.
- ある $a \in Act$ に対して，もし $(s,\sigma) \xrightarrow{a} (s',\sigma')$ ならば，$(s,[\sigma]) \overset{a}{\leadsto}_r (s',[\sigma'])$

このとき，リージョンオートマトンは時間経過量を無視すれば元の時間オートマトンとアクションの実行系列の集合が等価(注2)な有限オートマトンとなる．

［例 4］　図 1 の例に対するリージョンオートマトンは図 4 のようになる．図中の $r_1,\ldots,r_{17}$ はそれぞれ図 3 に記載されたリージョンに対応する．図 2 と同様に，水平方向の矢印は遅延による遷移，垂直方向の矢印はアクションによる遷移，点線矢印は遷移先状態を省略したことを表す．□

3.2　クロックゾーン

リージョンオートマトンの状態数，すなわちリージョン等価性による同値類分割の総数は，定数の整数部分の値の大きさに依存したサイズになるので，効率が悪いことがある．例えば 10000 という定数が記述された時間オートマトンに対しては 10000 個の状態が導入される．すべての定数の最大公約数で定数倍 (スケーリング) すれば，ある程度この問題は回避可能であるが，例えば 10000 と 9999 という 2 つの定数が現れる場合など，ある 2 つの定数が互いに素である場合は状態数を削減できない．そこで，検証の効率化のためにリージョンの概念を発展させて，**クロックゾーン (clock zone)**(あるいは単に**ゾーン (zone)**) と呼ばれる概念が考案された [8], [12]．ゾーンとはクロック変数と整数の比較，および，2 つのクロック変数の差と整数の比較，およびこれらの論理積で表現される制約式，すなわち，$x \sim n$ または $x - y \sim n$ $(x,y \in C,\ n \in \mathbb{N},\ \sim \in \{<,\leqq,>,\geqq,=\})$ の形の原子述語の論理積で表現されるクロック制約式であると定義する．クロック変数 C に対する全てのゾーンの集合を $DCC(C)$ と記述する．

ゾーン $D \in DCC(C)$ に対して，$[D] \in 2^{Val(C)}$ を D を満たす付値の部分集合，すなわち，$[D] \overset{\text{def}}{=} \{\sigma | \sigma \models D\}$ と定義する．以下では混乱が生じない限り，ゾーン D とそれが表現する付値の集合 $[D]$ を同一視し，$[D]$ を単に D と書くことにする．

D をゾーンとするとき，$D\uparrow \overset{\text{def}}{=} \{\sigma + d | \sigma \in D \wedge d \in \mathbb{R}^+\}$，$D[r \to 0] \overset{\text{def}}{=} \{\sigma[r \to 0] | \sigma \in D\}$ とする．直観的には $D\uparrow$ は D から任意の時間経過で到達可能な付値の集合，$D[r \to 0]$ は D から r に属するクロック変数のリセットで到達可能な集合を表す．任意のゾーン D に対して，D における全てのクロック変数 x の自由な出現を $x + d$ で置き換えて得られる制約式 (ゾーン) を $D + d$ とする．このとき，$D\uparrow$ と $D[r \to 0]$ は一階述語論理式を用いてそれぞれ $D\uparrow = \exists d[d \geqq 0 \wedge D + d]$, $D[\{y_1,\ldots,y_k\} \to 0] = [(\exists y_1,\ldots,y_k[D]) \wedge y_1 = 0 \wedge \cdots \wedge y_k = 0]$ と表すことができる．これらの右辺の存在限定子は除去可能であり，除去して得られる論理式はゾーンとなる．従って，D がゾーンならば $D\uparrow$ および $D[r \to 0]$ もゾーンとなる．一般に，ゾーンの集合は演算 $D\uparrow$(時間進行演算),$D[r \to 0]$(クロックリセット演算) および $D_1 \wedge D_2$(共通集合演算) について閉じている [8], [13].

ゾーンの概念を用いて，多くの場合リージョンオートマトンより状態数が小さい**ゾーンオートマトン (zone automaton)** [8] を次のように得ることができる(注3)．ロケーション $s \in S$ とゾーン D の

(注2)：正確には，元の時間オートマトンと非時間双模倣等価 (untimed bisimulation equivalent [10], time-abstracted bisimulation equivalent [11]) となる．

(注3)：実用的にはゾーンオートマトンの状態数はリージョンオートマトンより小さくなるが，一般にゾーンはリージョンの部分集合なので，可能なゾーンの全体はリージョンの総数の指数関数となる．すなわち，

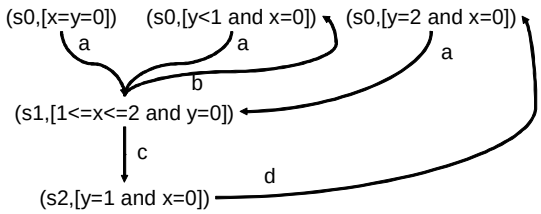

図 5　図 1 に対するゾーンオートマトン
Fig. 5　Zone Automaton of the Timed Automaton in Fig. 1

組 (s, D) を (ゾーンに関する) 記号的状態と呼ぶ．任意の $a \in Act$ に対して記号的状態上の遷移関係 $\overset{a}{\leadsto}_D \subseteq (S \times 2^{Val(C)}) \times Act \times (S \times 2^{Val(C)})$ を以下のように定義する：

もし $s \xrightarrow{a,g,r} s'$ ならば，$(s, D) \overset{a}{\leadsto}_D (s', (D \uparrow \wedge Inv(s) \wedge g)[r \to 0] \wedge Inv(s'))$

このとき，ゾーンオートマトンは時間経過量を無視すれば元の時間オートマトンとアクションの実行系列の集合が等価な有限オートマトンとなる．直観的にはゾーンオートマトンにおける 1 遷移で，以下の一連の動作が表現される．

（1） ロケーション不変式の範囲内での時間経過によるクロック変数の進行 $(D \uparrow \wedge Inv(s))$

（2） ガード条件成立によるアクションの実行 (g)

（3） リセット集合に属するクロック変数のリセット $([r \to 0])$

（4） 次ロケーションへの遷移，ただしロケーション不変条件を満たすこと $(Inv(s'))$

[例 5]　図 1 の例に対するゾーンオートマトンは図 5 のようになる．　□

ゾーンの定義より明らかなように，リージョン等価な任意の 2 つの付値は必ず同じゾーンに属する．従って任意のゾーンはある有限個のリージョンの直和と等しい．リージョンの総数が有限個であることより，原理的には付値の集合としてのゾーン $[D]$ の総数は有限個である(注4)．しかし，制約式によるゾーンの表現 D は一意ではなく，一般に無限通りの可能性がある．そこで，一意な表現に簡約化可能なゾーンの表現として **DBM(difference bound matrix)** [14] が考案された．DBM は次のようなデータ構造である．まず各クロック変数 $x \in C$ に対して添え字 $i = 1, \ldots, |C|$ を付加する．すなわち $C = \{x_1, \ldots, x_{|C|}\}$ とする．次に x_0 をその値が常に 0 であるような特別なクロック変数とする．このとき任意のゾーン $D \in DCC(C)$ は $x_i - x_j \prec_{ij} c_{ij}$ $((i = 0, \ldots, |C|) \prec_{ij} \in \{<, \leqq\})$ という形の原子述語の論理積で表される $(x_i \sim c_i$ $(\sim \in \{<, \leqq, >, \geqq, =\})$ の形の制約式は x_0 を用いて $x_i - x_0 \prec_{i0} c_i$ または $x_0 - x_i \prec_{0i} c_i$ の論理積で表される事に注意せよ)．このとき，i 行 j 列の要素が $(\prec_{ij}, c_{ij})$ または ∞ である $|C| + 1$ 次元の正方行列 $M(D)$ を D の **DBM 表現 (DBM representation)** と呼ぶ．$M(D)$ の i 行 j 列の要素を $M_{ij}(D)$ と書く．直観的には $M_{ij}(D)$ が $(\prec_{ij}, c_{ij})$ であることは，x_i と x_j の値の差が c_{ij} 以下 ($\prec_{ij}$=$\leqq$ の場合) または未満 ($\prec_{ij}$=$<$ の場合) であるという制約を意味している．また，$M_{ij}(D) = \infty$ であることは，x_i と x_j の値に対する (直接の) 制約がないことを意味する．DBM 表現はクロック制約による表現と同様に，同じゾーンに対して複数の表現がある．

表 1　DBM 表現 $M(D)$ および $M'(D)$
Table 1　DBM representations $M(D)$ and $M'(D)$

	DBM 表現 $M(D)$			DBM 表現 $M'(D)$		
	0	1	2	0	1	2
0	∞	$(\leqq, 0)$	$(<, 0)$	$(\leqq, 0)$	$(\leqq, 0)$	$(<, 0)$
1	$(<, 2)$	∞	∞	$(<, 2)$	$(\leqq, 0)$	$(<, 2)$
2	$(<, 1)$	$(\leqq, 1)$	∞	$(<, 1)$	$(\leqq, 0)$	$(\leqq, 1)$

[例 6]　制約式 $(0 \leqq x_1 < 2) \wedge (0 < x_2 < 1) \wedge (x_1 - x_2 \geqq 0)$ が表すゾーン D に対する DBM 表現は表 1 に示す $M(D)$ または $M'(D)$ のようになる [13]．　□

DBM 表現に対して，先に述べた時間進行，共通集合，クロックリセットの各演算を効率よく行うことができることが示されている [8]．例えば時間進行演算 $D \uparrow$ に関しては，$M(D)$ における各クロック変数 x_i の上限のみを無限大に変更，すなわち，$M_{i0}(D \uparrow) = \infty$，かつ任意の $j > 0$ に対して $M_{ij}(D \uparrow) = M_{ij}(D)$ とすることで求められる．これらの演算により，初期ロケーションと初期クロック付値の DBM 表現から始めて，到達可能な状態を順次構成していくことにより，ゾーンオートマトンを効率的に構成することができる．また，DBM 表現に対しては，それが表現するゾーンに対して一意な正規形 (canonical form) を次のように構成できる．各クロック変数 x_i をノードとみなし，各制約 $x_i - x_j \prec_{ij} c_{ij}$ に対して，x_i と x_j の間の距離

最悪の場合を考慮すると，ゾーンオートマトンの方が状態数が多くなる．従って，決定問題の計算量を評価する目的には一般にリージョンオートマトンの方が適する．
(注4)：正確にはリージョンの構成法と同様にクロック上限 k を考慮したある正規化処理を行って有限個のゾーンに落とし込む [8]．

c_{ij} の辺を対応させる．得られた距離付きグラフに対して，負の距離も考慮した最短距離導出アルゴリズムである Floyd-Warshall アルゴリズム [9] を適用することにより，同じゾーンを表現する最簡な DBM 表現を求めることができる．DBM 表現の正規形への変換により，既に訪れた状態と等価な状態を重複して生成することを回避して，常に有限状態のゾーンオートマトンを得ることができる．

本節の内容の詳細については，文献 [1, 17 章] や文献 [8] などを参照されたい．

3.3 時間制約付き到達可能性解析

実時間システムにおいては，単純な到達可能性ではなく，例えば「2 秒以内に状態 s_2 に到達する」などの時間制約の付いた到達可能性を検証したいことがしばしばある．しかし，前述のリージョンオートマトンやゾーンオートマトンは，具体的に進んだ時間量は全て捨象されたモデルになっており，そのままでは時間制約つきの性質を検証できない．

この問題は，検証対象の時間オートマトン M と，その時間制約付き性質を観測する振る舞いを行う時間オートマトン M_p の積オートマトン $M||M_p$ を構成し，$M||M_p$ に対する単純な到達可能性に帰着することにより解決できる．一般に時間オートマトンに対する時間制約つきの性質の検証問題は，クロック変数を用いて観測動作を行う時間オートマトンとの積オートマトンに対するモデル検査問題に帰着可能である [2], [15]．検証できる性質のクラスは観測用に用いるクロック変数 (仕様クロック変数 (specification clock)) の個数によって一般に変化する [10]．詳細については [2], [10], [15] を参照されたい．

4. 最新の研究動向

本節では，本稿で詳細を紹介できなかった時間オートマトンのモデル検査技術の最新の研究動向について，その概要を簡単に紹介する．

4.1 実時間システムの振る舞い性質の記述および検証

時間オートマトンのモデル検査においては本稿で紹介した以外の性質，例えば，「リクエストがあれば 5 秒以内に必ず応答する」（時間制約付き活性）などの性質の検証も可能である．このような性質を含めて，リアクティブな実時間システムにおいて興味のある多くの (一般に時間制約を含む) 性質を，大きく分けて 2 種類の手段で記述可能である．１つは実時間時相論理 (real-time temporal logic) [16]，もう１つは時間 ω オートマトン [2] である．

時相論理 (temporal logic) によるシステムの振る舞いに関する性質記述法は，古くから多くの研究がなされてきた [17]〜[19]．古典的な時相論理においては，1 ステップで 1 単位時間経過するというモデルを前提としており，システムの実時間的な振る舞いを記述できなかった．近年，時相論理の実時間拡張がいくつか提案されてきた [7], [15], [16], [20], [21]．時相論理には大きく分けて，線形時相論理 (linear-time temporal logic) および分岐時相論理 (branching-time temporal logic) の 2 種類がある．線形時相論理はシステムの動作系列に関する性質を記述する時相論理であり，分岐時相論理はシステムの状態およびそれ以降の分岐構造（計算木 (computation tree)）に関する性質を記述する時相論理である．それぞれの実時間拡張を **線形実時間時相論理 (linear-real-time temporal logic)**，**分岐実時間時相論理 (branching-real-time temporal logic)** と呼ぶ．線形実時間時相論理に関しては，線形時相論理 LTL [17] の自然な実時間拡張として TPTL [20] が提案されたが，稠密時間モデルに関してはモデル検査問題が決定不能であることも同時に示された．その後，TPTL の決定可能なサブクラスとして，MITL [22],TLTL$_{ec}$ [23] などが提案されている．分岐実時間時相論理に関しては，分岐時相論理 CTL [19] の自然な実時間拡張として TCTL [7], [15] が提案された．TCTL のモデル検査に関しては決定可能であり，具体的なモデル検査アルゴリズムとして，CTL モデル検査手法 [24] をクロックリージョンの概念を用いて実時間拡張した手法 [15]，および，CTL の記号モデル検査 [25] を実時間拡張した手法 [7] などが提案されている．

時間制約つきの活性などの性質は，実時間時相論理以外に，**時間 ω オートマトン (timed ω-automaton)** [2] によっても記述することができる(注5)．ω オートマトン [26] とは古典的な有限オートマトンの受理の概念を，無限長の語に拡張したオートマトンである．無限語の受理の定義の流儀にはいくつかあるが，もっとも単純なものが「指定した受理状態のいずれかを無限回訪れるような入力語を受理する」という条件であり，

(注5)：時間 ω オートマトンは性質を記述する目的のみならず，検証対象のモデルを記述する目的にも利用できる．その場合は，実際にありえない動作系列，例えば複数スレッドの並行実行のモデル化において，公平でない動作系列などを排除する目的で受理の概念を利用する．

Büchi 受理条件 [26] と呼ばれる．Büchi 受理条件を持つ ω オートマトンを Büchi オートマトン，Büchi 受理条件を持つ時間 ω オートマトンを時間 Büchi オートマトンと呼ぶ．例えば活性などの性質を満たす（すなわち，望ましい）動作系列の全体を受理する時間 Büchi オートマトン M_p を構成できれば，時間オートマトン M が性質 M_p を満たすか否かのモデル検査問題は，M の受理言語 $L(M)$ が M_p の受理言語 $L(M_p)$ に含まれるか否か ($L(M) \subseteq L(M_p)$) の判定問題（言語包含関係判定問題）に帰着される．このようなモデル検査のアプローチは**オートマトン理論アプローチ (automata-theoretic approach)** [27] と呼ばれる．言語包含関係の検証は，性質 M_p の受理言語の補集合 $\overline{L(M_p)}$ を受理する ω オートマトンを構成 (補集合演算) できれば，$L(M) \cap \overline{L(M_p)}$ が空か否かの判定問題 (空問題) に帰着する．Büchi オートマトンの空問題判定は，状態遷移グラフにおいて受理状態を含む閉路を検出することで行えるため，効率の良いアルゴリズムが提案されている [28]．時間オートマトンに対してもリージョンオートマトンから等価な Büchi オートマトンに変換可能であることが知られており [2]，同様のアルゴリズムが適用可能であるため，時間 Büchi オートマトンの空問題は決定可能である．また，時間 Büchi オートマトンは共通部分演算に関して閉じていることが知られているが，補集合演算に関しては閉じておらず，言語包含関係判定問題は一般に決定不能である [2]．しかし，時間 Büchi オートマトンが決定性であれば補集合演算に関して閉じているので決定可能となる．また，決定性に変換可能な時間 Büchi オートマトンのサブクラスとして event-clock オートマトン [29] が提案されており，このクラスで性質を記述すれば言語包含関係判定問題は決定可能となる．別のアプローチとして，望ましい動作系列ではなく望ましくない動作系列を受理するオートマトン (never claim [30]) を記述することで，性質を指定する方法もある．この方法では補集合演算の必要がなく，空問題の判定によってモデル検査を行うことができる．

オートマトン理論アプローチは実時間線形時相論理のモデル検査アルゴリズムとしても用いることができる．すなわち，与えられた実時間線形時相論理式に対して，その式を満たさない動作系列の全体を受理する時間 Büchi オートマトンを構成できれば，モデル検査問題は空問題に帰着して解くことができる．MITL は時間 Büchi オートマトンに，TLTL$_{ec}$ は event-clock オートマトンにそれぞれ変換可能であることが示されている [22], [23]．

4.2 時間オートマトンのさまざまな拡張

時間オートマトンモデルに対してはさまざまな拡張が試みられている．その一部を簡単に紹介する．

- **updatable 時間オートマトン** [31] は，クロック変数の 0 へのリセットのみならず，定数や他のクロック変数値の代入などを許す拡張である．代入の形式に応じて空問題の決定可能性が変化することが示されている [31], [32]．
- **ハイブリッドオートマトン (hybrid automaton)** [33] は，時間オートマトンのクロック変数の概念を，連続的に変化する連続量の状態変数 (例えば，速度や温度など) に拡張したオートマトンである．連続量状態変数の遷移関係は一般に微分方程式で記述されるが，よりモデル検査に適したハイブリッドオートマトンのサブクラスとして **rectangular オートマトン** [33] が提案されている．rectangular オートマトンは，連続量状態変数の変化は離散状態の遷移と独立であり，微分係数（変化のレート）の一定の範囲内での変化や離散状態遷移においての微分係数の更新を許すモデルである．rectangular オートマトンは，時間オートマトンやクロック変数の値の変化を途中で止めたり再開したりすることができる**ストップウォッチオートマトン**のクラスを真に含んでいる [33]．
- 時間オートマトンのクロック制約に定数のみならずパラメータ変数を記述可能にした拡張を**パラメトリック時間オートマトン (parametric timed automaton)** [34] と呼ぶ．パラメトリック時間オートマトンのモデル検査問題は，通常のモデル検査の拡張となり，モデルが性質を満たすようなパラメータ変数に関する条件式の導出問題となる．これを**パラメトリックモデル検査 (parametric model checking)** と呼ぶ．パラメトリックモデル検査は一般にモデルと性質のいずれか (あるいは両方) にパラメータ変数を含む場合のモデル検査を意味する．パラメトリックモデル検査によって，性質を満たす範囲でのモデルのパラメータ最適化が可能となる．パラメトリックモデル検査の決定可能性は [34] で初めて議論され，性質にパラメータを含む場合については [35]〜[38] など，モデルにパラメータを含む場合については [39]〜[45]，などの研究が成されてきた．時間オートマトンのパラメトリックモデル検査は，モデル検査技術を援用することによりシステムの時間に関するパラメータの最適化

を支援する技術であり，近年の組込みシステムのコスト・消費電力最適化 [46] やハードウェア変更時のソフトウェア再利用などへの応用が期待される．しかし，パラメトリックモデル検査においては，クロック変数と比較されるのが定数とは限らないため，本稿で紹介したリージョンの概念を用いることができない．リージョンの概念は，到達可能性や他の性質のモデル検査アルゴリズムの決定可能性に重要な意味をもっており，時間オートマトンで決定可能であった多くの問題がパラメトリック時間オートマトンでは決定不能となっている．パラメトリックモデル検査が決定可能となるようなパラメトリック時間オートマトンのサブクラスの考案は [34], [42], [43], [45] などで成されているが，さらなる研究が必要とされている．

- その他の拡張としては，**時間木オートマトン (timed tree automaton)** [47] や **priced 時間オートマトン (priced timed automaton)** [48] などが提案されている．時間木オートマトンは，木構造を受理するオートマトンである木オートマトン (tree automaton) の時間拡張であり，これを用いて分岐実時間時相論理のオートマトン理論アプローチによるモデル検査手法の研究が成されている [47], [49]．priced 時間オートマトンは，時間オートマトンの各遷移にコストを付加した拡張であり，実時間システムのコスト最適化問題への応用が試みられている [50], [51].

4.3 時間オートマトンのモデル検査ツール

現在までに，時間オートマトンのモデル検査手法を実装したさまざまなツールが公開されている．最も著名なのが UPPAAL [52] である．UPPAAL はゾーンオートマトンおよび DBM 表現に基づく到達可能性解析を実装したツールであり，同期通信を行う複数の時間オートマトンの組に対して，TCTL の部分クラスで記述された振る舞い性質を検証可能である．UPPAAL はモデルの記述から検証までの作業を GUI で一貫して行うことができるため，初学者が時間オートマトンのモデル検査を学ぶのに適している．UPPAAL は非商用利用に関しては無償で提供されており，Web サイト `http://www.uppaal.com/` よりダウンロード可能である．詳細については当該サイトを参照されたい．他に，時間オートマトン (あるいはその拡張) のモデル検査ツールとして無償でダウンロード可能なものとしては，Kronos [53](注6), HYTECH [40](注7), TReX [41](注8), Times [54](注9) などがある．これらはそれぞれ基づいている要素技術，扱うモデルや性質のクラスなどが異なり，各々特徴を持っている．詳細については各文献や Web サイトを参照されたい．

5. あとがき

本稿では紙数の制約により，時間オートマトンのモデル検査技術の基礎について，安全性/到達可能性検証に焦点を絞って概説した．また，本稿で紹介しきれなかった最新の研究動向に関してはごく簡単に概要を紹介した．本稿が実時間システムの検証技術を学ぶ際の一助となれば幸いである．

謝辞 本稿の執筆に際し，貴重な助言を頂きました電子情報通信学会論文誌フォーマルアプローチ特集号編集委員会の皆様に深謝致します．

文　献

[1] E. M. Clarke, O. Grumberg and D. A. Peled: “Model Checking”, MIT Press (1999).

[2] R. Alur and D. Dill: “A theory of timed automata”, Theoretical Comput. Sci., **126**, pp. 183–235 (1994).

[3] B. Berthomieu and M. Diaz: “Modeling and verification of time dependent systems using time Petri nets”, IEEE Trans. Softw. Eng., **17**, 3, pp. 259–273 (1991).

[4] C. L. Liu and J. W. Layland: “Scheduling algorithms for multiprogramming in a hard-real-time environment”, J. ACM, **20**, 1, pp. 46–61 (1973).

[5] F. Cassez and O. H. Roux: “Structural translation from time petri nets to timed automata - model-checking time petri nets via timed automata”, The Journal of Systems and Software, **79**, 10, pp. 1456–1468 (2006).

[6] E. Fersman, P. Pettersson and W. Yi: “Timed automata with asynchronous processes: schedulability and decidability”, Proc. of the 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002), Springer-Verlag, pp. 67–82 (2002).

[7] T. A. Henzinger, X. Nicollin, J. Sifakis and S. Yovine: “Symbolic model checking for real-time systems”, Information and Computation, **111**, pp. 193–244 (1994).

[8] J. Bengtsson and W. Yi: “Timed automata: Semantics, algorithms and tools”, Lectures on Concurrency and Petri Nets, Vol. 3098 of Lecture Notes in Computer Science, Springer-Verlag, pp. 87–124 (2004).

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein: “Introduction to Algorithms”, MIT Press,

(注6)：`http://www-verimag.imag.fr/TEMPORISE/kronos/`
(注7)：`http://embedded.eecs.berkeley.edu/research/hytech/`
(注8)：`http://www.liafa.jussieu.fr/~sighirea/trex/`
(注9)：`http://www.timestool.com/`

2nd edition (2001).

[10] R. Alur, C. Courcoubetis and T. A. Henzinger: "The observational power of clocks", Proc. of CONCUR'94, Vol. 836 of Lecture Notes in Computer Science, Springer-Verlag, pp. 162–177 (1994).

[11] K. G. Larsen and Y. Wang: "Time abstracted bisimulation: Implicit specifications and decidability", Proc. of 9th Int. Conf. on Mathematical Foundations of Programming Semantics (MFPS'93) (Eds. by S. Brookes, M. Main, A. Melton, M. Mislove and D. Schmidt), Vol. 802 of Lecture Notes in Computer Science, Springer-Verlag, pp. 160–175 (1993).

[12] W. Yi, P. Pettersson and M. Daniels: "Automatic verification of real-time communicating systems by constraint-solving", Proc. of the 7th IFIP TC6/WG6.1 Int. Conf. on Formal Description Techniques (FORTE'94), pp. 223–238 (1994).

[13] R. Alur: "Timed automata", Proc. of the 11th Int. Conf. on Computer Aided Verification (CAV'99), Vol. 1633 of Lecture Notes in Computer Science, Springer-Verlag, pp. 8–22 (1999).

[14] D. L. Dill: "Timing assumptions and verification of finite-state concurrent systems", Proc. of Int. Conf. on Automatic Verification Methods for Finite State Systems, Vol. 407 of Lecture Notes in Computer Sciences, Springer-Verlag, pp. 197–212 (1989).

[15] R. Alur, C. Courcoubetis and D. Dill: "Model-checking in dense real-time", Information and Computation, **104**, pp. 2–34 (1993).

[16] R. Alur and T. Henzinger: "Logics and models of real time: A survey.", Real Time: Theory in Practice (Eds. by J. W. de Bakker, C. Huizing, W. P. de Roever and G. Rozenberg), Vol. 600 of Lecture Notes in Computer Science, Springer-Verlag, pp. 74–106 (1992).

[17] A. Pnueli: "The temporal logic of programs", Proc. of the 18th IEEE Symp. on Foundation of Computer Science (FOCS '77), IEEE Computer Press, pp. 46–57 (1977).

[18] D. Kozen: "Results on the propositional μ-calculus", Theoretical Comput. Sci., **27**, pp. 333–354 (1983).

[19] E. Emerson and J. Halpern: "'sometimes' and 'not never' revisited: on branching time versus linear time temporal logic", J. ACM, **33**, 1, pp. 151–178 (1986).

[20] R. Alur and T. A. Henzinger: "A really temporal logic", Proc. of the 30th IEEE Int. Symp. on Foundations of Computer Science (FOCS 1989), IEEE Computer Society Press, pp. 164–169 (1989).

[21] R. Alur and T. A. Henzinger: "Real-time logics: complexity and expressiveness", Information and Computation, **104**, pp. 390–401 (1993).

[22] R. Alur, T. Feder and T. A. Henzinger: "The benefits of relaxing punctuality", J. ACM, **43**, 1, pp. 116–146 (1996).

[23] D. D'Souza: "A logical characterization of event clock automata", Int. Journal of Foundations of Computer Science, **14**, 4, pp. 625–639 (2003).

[24] E. M. Clarke, E. A. Emerson and A. P. Sistla: "Automatic verification of finite state concurrent systems using temporal logic specifications", ACM Trans. on Program Languages and Semantics, **8**, 2, pp. 244–263 (1986).

[25] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill and L. J. Hwang: "Symbolic model-checking: 10^{20} states and beyond", Information and Computation, **98**, 2, pp. 142–170 (1992).

[26] W. Thomas: "Automata on infinite objects", Handbook of Theoretical Computer Science (Ed. by J. van Leeuwen), Vol. B, Elsevier Science Publishers, pp. 133–191 (1990).

[27] M. Y. Vardi and P. Wolper: "An automata-theoretic approach to automatic program verification", Proc. of the 1st Annual IEEE Symp. on Logic in Computer Science (LICS'86), IEEE Computer Society Press, pp. 332–345 (1986).

[28] C. Courcoubetis, M. Vardi, P. Wolper and M. Yannakakis: "Memory-efficient algorithms for the verification of temporal properties", Formal Methods in System Design, **1**, pp. 275–288 (1992).

[29] R. Alur, L. Fix and T. A. Henzinger: "Event-clock automata: A determinizable class of timed automata", Proc. of the 6th Int. Conf. on Computer Aided Verification (CAV'94), Vol. 818 of Lecture Notes in Computer Science, Springer-Verlag, pp. 1–13 (1994).

[30] G. J. Holzmann: "The model checker SPIN", IEEE Trans. Softw. Eng., **23**, 5, pp. 279–295 (1997).

[31] P. Bouyer, C. Dufourd, E. Fleury and A. Petit: "Are timed automata updatable?", Proc. of the 12th Int. Conf. of Computer Aided Verification (CAV 2000), Vol. 1855 of Lecture Notes in Computer Science, Springer, pp. 464–479 (2000).

[32] R. Alur and P. Madhusudan: "Decision problem for timed automata: A survey", 4th Intl. School on Formal Mthods for Computer, Communication, and Software Systems: Real Time (2004).

[33] T. A. Henzinger: "The theory of hybrid automata", Proc. of the 11th Annual IEEE Symp. on Logic in Computer Science (LICS'96), pp. 278–292 (1996).

[34] R. Alur, T. A. Henzinger and M. Y. Vardi: "Parametric real-time reasoning", Proc. 25th ACM Annual Symp. on the Theory of Computing (STOC'93), pp. 592–601 (1993).

[35] F. Wang: "Parametric timing analysis for real-time systems", Information and Computation, **130**, 2, pp. 131–150 (1996).

[36] R. Alur, K. Etessami, S. L. Torre and D. Peled: "Parametric temporal logic for model measuring", Proc. of the 26th Int. Colloq. on Automata, Languages, and Programming (ICALP'99), Vol. 1644 of

Lecture Notes in Computer Science, Springer-Verlag, pp. 159–168 (1999).

[37] E. A. Emerson and R. J. Trefler: “Parametric quantitative temporal reasoning”, Proc. of the 14th Annual IEEE Symp. on Logic in Computer Science (LICS’99), pp. 336–343 (1999).

[38] F. Wang: “Parametric analysis of computer systems”, Formal Methods in System Design, **17**, 1, pp. 39–60 (2000).

[39] R. Alur, T. A. Henzinger and P. Ho: “Automatic symbolic verification of embedded systems”, IEEE Trans. on Software Engineering, **22**, 3, pp. 181–201 (1996).

[40] T. A. Henzinger, P.-H. Ho and H. Wong-Toi: “HYTECH: A model checker for hybrid systems”, International Journal on Software Tools for Technology Transfer, **1**, 1-1, pp. 110–122 (1997).

[41] A. Annichini, A. Bouajjani and M. Sighireanu: “TReX: A tool for reachability analysis of complex systems”, Proc. of the 13th Int. Conf. Computer Aided Verification(CAV 2001), Vol. 2102 of Lecture Notes in Computer Science, Springer, pp. 368–372 (2001).

[42] A. Nakata and T. Higashino: “Deriving parameter conditions for periodic timed automata satisfying real-time temporal logic formulas”, Proc. of 21st IFIP TC6/WG6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE2001), Kluwer Academic Publishers, pp. 151–166 (2001).

[43] T. Hune, J. Romijn, M. Stoelinga and F. W. Vaandrager: “Linear parametric model checking of timed automata”, Proc. of the 7th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2001), pp. 189–203 (2001).

[44] V. Bruyere and J.-F. Raskin: “Real-time model-checking: Parameters everywhere”, Proc. of 23rd Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FST & TCS 2003), Vol. 2914 of Lecture Notes in Computer Science, Springer, pp. 100–111 (2003).

[45] T. Tanimoto, A. Nakata, H. Hashimoto and T. Higashino: “Double depth first search based parametric analysis for parametric time-interval automata”, IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, **E88-A**, 11, pp. 3007–3021 (2005).

[46] T. Kitani, Y. Takamoto, K. Yasumoto, A. Nakata and T. Higashino: “A flexible and high-reliable hw/sw co-design method for real-time embedded systems”, Proc. of the 25th IEEE Real-Time Systems Symposium (RTSS 2004), IEEE Computer Society Press, pp. 437–446 (2004).

[47] S. L. Torre and M. Napoli: “Timed tree automata with an application to temporal logic”, Acta Informatica, **38**, 2, pp. 89–116 (2001).

[48] K. G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson and J. Romijn: “As cheap as possible: Efficient cost-optimal reachability for priced timed automata”, Proc. of the 13th Int. Conf. on Computer Aided Verification (CAV 2001), Vol. 2102 of Lecture Notes in Computer Science, Springer-Verlag, pp. 493–505 (2001).

[49] M. Dickhöfer and T. Wilke: “Timed alternating tree automata: The automata-theoretic solution to the TCTL model checking problem”, Proc. of the 26th Int. Colloq. on Automata, Languages and Programming (ICALP’99), Vol. 1644 of Lecture Notes in Computer Science, Springer-Verlag, pp. 281–290 (1999).

[50] R. Alur, M. Bernadsky and P. Madhusudan: “Optimal reachability for weighted timed games”, Proc. of the 31st Int. Colloq. on Automata, Languages, and Programming (ICALP 2004), Vol. 3142 of Lecture Notes in Computer Science, Springer-Verlag, pp. 122–133 (2004).

[51] J. I. Rasmussen, K. G. Larsen and K. Subramani: “On using priced timed automata to achieve optimal scheduling”, Formal Methods in System Design, **29**, pp. 97–114 (2006).

[52] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson and W. Yi: “UPPAAL: a tool suite for automatic verification of real-time systems”, Proc. of Hybrid Systems III, Vol. 1066 of Lecture Notes in Computer Science, Springer-Verlag, pp. 232–243 (1996).

[53] S. Yovine: “Kronos: A verification tool for real-time systems”, Int. Journal of Software Tools for Technology Transfer, **1**, 1/2, pp. 123–133 (1997).

[54] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson and W. Yi: “TIMES: a tool for schedulability analysis and code generation of real-time systems”, Proc. of the 1st Int. Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003), Vol. 2791 of Lecture Notes in Computer Science, Springer-Verlag, pp. 60–72 (2004).

（平成 xx 年 xx 月 xx 日受付）

中田　明夫

平成 4 年大阪大学基礎工学部情報工学科卒業．平成 9 年同大大学院博士課程修了．博士 (工学)．同年広島市立大学情報科学部助手．平成 12 年大阪大学大学院基礎工学研究科助手．平成 14 年同大大学院情報科学研究科助教授．現在，広島市立大学大学院情報科学研究科教授．実時間システム，並行分散システムの設計手法および形式的検証法に関する研究に従事．情報処理学会会員．

Abstract In this paper, we briefly describe model checking of timed automata, one of the verification techniques of real-time systems, where we especially focus on reachability analysis. Firstly, we describe the definition of syntax and semantics of timed automata. Secondly, we describe the reachablity analysis technique of timed automata, where we especially emphasize the key idea of the technique, clock regions and clock zones. Finally, we briefly survey a part of recent advances in model checking of timed automata.

Key words real-time systems, verification techniques, timed automata, model checking, reachability analysis