マルチメディア通信と分散処理　101－15
コンピュータセキュリティ　12－15
（2001．2．21）

タイマを用いる有限状態機械でモデル化されたシステムの検証手続き

徳田 康平 † 森 亮憲 † 樋口 昌宏 ‡ 谷口 健一 †

† 大阪大学
大学院基礎工学研究科 情報数理系専攻

‡ 近畿大学
理工学部 電気工学科

本論文ではタイマを用いる有限状態機械モデルで記述されたシステムに対する検証手続きについて議論する．検証はタイマと有限状態機械からなる系における到達可能な状態の列挙に基づいて行われる．しかしタイマの設定時間が大きい場合，到達可能な状態数が膨大となるため，到達可能な状態を列挙するにはメモリや計算時間の点で問題となる．そこでタイマの取り得る値を連立不等式を用いて表し，複数の状態と状態遷移を一括して扱い到達可能な状態を列挙する手法を提案した．また提案手法による到達可能な状態の列挙に基づく検証手続きを実装し，DHCP に適用した．その結果，DHCP の到達可能な状態を現実的な時間で列挙し，タイマと有限状態機械間のやりとりに不具合がないことを検証することができた．

A Verification Procedure for Systems Modeled as Finite State Machines with Timers

Kouhei Tokuda † Takanori Mori †

Masahiro Higuchi ‡ Kenichi Taniguchi †

†Graduate School of Engineering Science
Osaka University

‡School of Science and Engineering
Kinki University

In this paper, we discuss a verification procedure for systems modeled as finite state machines with timers. We can verify such systems enumerating reachable states of the system from the initial state. If timers can store large integer value, it is troublesome to enumerate reachable states because of the large number of states of the system. We propose a procedure to enumerate reachable states of such system efficiently with simultaneous inequalities. We develop a verification system based on the proposing procedure, and apply to DHCP. As a result, the verification was executed in practical time.

1. まえがき

システムが正しく動作することを保証するために，仕様段階において検証を行うことは重要である．検証はシステムが到達可能な状態を列挙することによって行われることが多い．本論文ではオペレーティングシステム (OS) の提供するタイマ機能を使用し，時間を考慮した振舞いをするシステムの検証について議論する．OS の提供するタイマ機能はシステムコールにより利用することができる．例えば Linux ではシステムコール **set_timer()**, **dell_timer()** により OS の提供するタイマ機能を利用することができる．検証対象となるシステムをタイマ T と有限状態機械 M でモデル化する．M と T からなる系 M_T を検証するために，M_T 上で到達可能な状態の列挙を行う．しかし，M_T 上で到達可能な状態を 1 つずつ列挙する方法では，到達可能な状態数が膨大となり計算時間やメモリの点で問題となる．そこで本論文ではタイマの取り得る値を連立不等式を用いて表し，複数の状態と状態遷移を一括して列挙する手法を提案した．

以降，2. ではタイマを用いる有限状態機械モデルについて，3. では検証法について述べる．4. では検証に求められる到達可能な状態の列挙手続きについて述べ，5. で実験を行い，その実験結果について述べる．

2. タイマを用いる有限状態機械モデル

本論文では図 1のようなシステムモデルについて考える．以下タイマ部とタイマを用いる有限状態機械部について説明する．

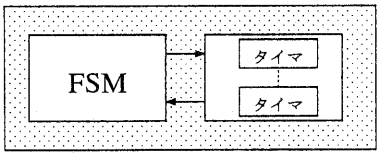

図 1　タイマを用いる有限状態機械モデル

2.1 タイマ部

タイマは既存の機能として提供されており，一般にタイマは複数個存在する．n 個のタイマを使用する場合，それぞれのタイマを区別するために，各タイマに対して 1 から n の番号がついているものとする．タイマに関する操作は設定する (S)，解除する (D)，何も行わない (N) の 3 つの操作のみとする．各タイマには単位時間の整数倍である設定時間が決まっており，タイマ操作 S が行われた後タイマ操作 D が行われることなく設定時間が経過すると，そのタイマがタイムアウトしたことを通知する．

タイマ i の設定時間を T_i と表す．タイマ i のある時刻でのタイムアウト通知を出すまでの残り時間をタイマ値と呼び τ_i で表す (τ_i は整数値)．動作していない (解除された) タイマのタイマ値は $\perp$ で表す．タイマの初期値は $\perp$ である．τ_i は，$0 \leq \tau_i \leq T_i$ の整数値および $\perp$ の値をとる．$\mathcal{I}_i = \{0, \cdots, T_i, \perp\}$ とする．τ_i は，タイマ操作 (S) により T_i になり，タイマ操作 (D) により $\perp$ になる．任意の整数 x に対して，$\perp - x = \perp$ とする．全てのタイマのタイマ値は，単位時間ごとに一斉に 1 ずつ減少し，値が 0 の時にタイムアウト通知を行う．タイムアウト通知を行った後タイマ値は $\perp$ となる．タイマ値が 0 のままで単位時間が経過することはないものとする．複数のタイマが同時にタイムアウトした時には，番号の若いタイマから順にタイムアウト通知が行われる．タイマ値が 0 でまだタイムアウト通知を行っていないタイマに対してタイマ操作 (S) またはタイマ操作 (D) が行われると，タイマ値はそれぞれ $T_i, \perp$ になる．各タイマのタイマ値をまとめて n 次元のタイマ値ベクトル $\bar{\tau} = (\tau_1, \ldots, \tau_n)$，$\tau_i \in \mathcal{I}_i$ で表し，特に n 個の要素全てが $\perp$ である場合 $\perp^n$ と表す．

2.2 タイマを用いる有限状態機械部

タイマを用いる有限状態機械は Mealy 型の DFSM として，次の 6 字組 (S, X, n, Y, H, s_0) で定義されるものとする．

S ： 状態の有限集合．

X ： 外部入力記号の有限集合．

n ： 使用するタイマの数．

Y ： 外部出力記号の有限集合．

H ： 状態遷移 $(u, v, x, y, \bar{p})$ の有限集合．

$u, v \in S$：始状態，終状態

$x \in (X \cup \{\mathrm{Timeout}_1, \ldots, \mathrm{Timeout}_n\})$：入力

$y \in Y$：外部出力

$\bar{p} \in \langle \mathrm{S}, \mathrm{D}, \mathrm{N} \rangle^n$：タイマ出力

s_0 ： 初期状態．

状態遷移は，外部から入力が与えられた時，またはタイムアウト通知を受け取った時に発生する．外部入力による状態遷移を外部入力遷移，タイマからのタイムアウト通知による状態遷移をタイムアウト遷移と呼ぶ．状態遷移は，遷移前の状態と入力（外部入力またはタイムアウトしたタイマ番号）から遷移後の状態と出力（外部出力とタイマ出力）を一意に決定する．タイマ出力は，各要素が S, D, N のいずれかであるような n 次元ベクトルである．ベクトル $\bar{p}$ の第 i 要素 p_i はタイマ i へのタイマ操作を表す．

3. 検証法

3.1 検証項目

タイマ (T とする) とタイマを用いる有限状態機械 (M とする) からなる系 (M_T とする) を検証対象とする．今回検証する項目として (a)T-M 間でのタイムアウト通知，タイマ操作のやりとりが不具合なく行われるか (b) 系 M_T がある特定の性質に反する動作を行わないかの 2 つの場合について考える．

(a) T-M 間のやりとりの不具合として以下の 2 種類が考えられる．

- M 側がタイムアウト通知を待っているが T 側がタイムアウト通知を出さない場合．
  この時 M_T がデッドロックに陥る可能性がある．
- M 側がタイムアウト通知を待っていないが T 側がタイムアウト通知を出す場合．
  この時 M 側で定義されていないタイムアウト入力が発生する．

これらの不具合は，M の各状態でタイムアウトする可能性のあるタイマを特定し，M の各状態で必要かつ十分なタイムアウト遷移が記述されているかを調べることで検出することができる．

(b) システムの動作は入力記号列 $x^n(=x_1 \ldots x_n, n$ は正整数, $x_i \in (X \cup \{\mathrm{Timeout}_1, \ldots, \mathrm{Timeout}_n\}))$ で表される．$F(\subseteq S)$ を M の受理状態とし，入力記号列 x^n により，M 上で初期状態 s_0 から $s_F \in F$ に遷移する時，M は x^n を受理するという．F が指定されていない場合 $F=S$ である．M で受理される入力記号列 x^n を常に M' が受理する時，$M \subseteq M'$ という．

系の持つ性質は入力記号列の集合 α で表すことができる．$x^n \in \alpha$ のみを受理するタイマを用いる有限状態機械を $M'=(S', X, n', Y, H', s'_0, F'(\subseteq S'))$ とする．この時 $M \subseteq M'$ なら，M は系の持つ性質 α に反する動作を行わないといえる．$M \subseteq M'$ かどうかは，$M \wedge M'$ が初期状態 $\langle s_0, s'_0\rangle$ より $\langle s_f, s'_f\rangle (s_f \in F, s'_f \in \{S'\}-\{F'\})$ に到達可能であるかを調べることで判定できる．$M \wedge M'$ 上では，$(u, v, x, y, \overline{p}) \in H$ かつ $(u', v', x, y, \overline{p'}) \in H'$ の時，$\langle u, u'\rangle$ から $\langle v, v'\rangle$ へ到達可能である．しかし $(u, v, \mathrm{Timeout}_k, y, \overline{p}) \in H$ かつ $(u', v', \mathrm{Timeout}_k, y, \overline{p'}) \in H'$ の時，$\langle u, u'\rangle$ から $\langle v, v'\rangle$ へは $\langle u, u'\rangle$ でタイマ k がタイムアウトする可能性がある時のみ到達可能である．つまり，系 M_T がある特定の性質 α に反する動作をしないかは，$M \wedge M'$ の各状態でタイムアウトする可能性のあるタイマを特定しタイムアウト遷移の実行可能性を調べながら到達可能な状態を列挙することで調べることができる．

3.2 到達可能な列挙に基づく検証

前節の検証項目は (a)(b) 共にある状態においてタイムアウトする可能性のあるタイマを特定することで検証できる．つまり M_T 上で M_T の初期状態から到達可能な状態を列挙し，各状態でとり得るタイマ値を調べることが今回の検証で必要となる．

M_T の状態は M の状態 s とタイマ値ベクトル τ の組 $\langle s, \tau\rangle$ $(s \in S, \tau=(\tau_1, \ldots, \tau_n), \tau_i \in \mathcal{I}_i)$ で表される．M_T の初期状態は $\langle s_0, \perp^n\rangle$ である．$|\mathcal{I}_i|$ は有限であるのでタイマ値の組である τ の数も有限である．また $|S|$ も有限であるので M_T の状態 $\langle s, \tau\rangle$ の数は有限といえる．よって M_T において，初期状態より到達可能な状態の集合は有限時間で求めることができる．

4. 到達可能な状態の列挙手続き

4.1 単純な合成による手法

M_T 上で初期状態 $\langle s_0, \perp^n\rangle$ より到達可能な状態を列挙する．到達可能な状態の列挙が終了すれば，列挙した状態 $\langle s, \tau\rangle$ の τ を調べることでタイムアウトする可能性のあるタイマを特定することができる．

ここで M_T 上の遷移について説明する．状態遷移は (1)M に外部入力が与えられた時，(2) 時間が経過した時に発生する．また，(2) は (2-a) タイマ値が減少するだけの場合と (2-b) タイムアウトが発生する場合がある．

(1) M で $t=(u, v, x, y, \overline{p})$ が実行された場合，$\langle u, \tau\rangle$ から $\langle v, \overline{\tau'}\rangle$ へ遷移する．ただし，

$$\tau'_i = \begin{cases} T_i & (p_i = \mathrm{S}) \\ \tau_i & (p_i = \mathrm{N}) \\ \perp & (p_i = \mathrm{D}) \end{cases}$$

(2-a) 状態 $\langle u, \tau\rangle$ で $\tau \neq \perp^n, \forall i\ (1 \leq i \leq n)\ \tau_i \neq 1$ の時，時間経過により $\langle u, \tau - 1^n\rangle$ へ遷移する．ただし，1^n は n 個の要素がすべて 1 のベクトルを表す．

(2-b) 状態 $\langle u, \tau\rangle$ で $\exists i\ (1 \leq i \leq n)\ \tau_i = 1$ の時，時間経過により $\langle u, \tau - 1^n\rangle$ へ遷移する．その後，$\tau_i - 1 = 0$ であるタイマのうち最も若いタイマ (k とする) に対応するタイムアウト遷移 $(u, v, \mathrm{Timeout}_k, y, \overline{p}) \in H$ を実行し，$\langle u, \tau - 1^n\rangle$ から $\langle v, \overline{\tau'}\rangle$ へ遷移する．ただし，

$$\tau'_i = \begin{cases} T_i & (p_i = \mathrm{S}) \\ \tau_i - 1 & (i \neq k \wedge p_i = \mathrm{N}) \\ \perp & (p_i = \mathrm{D} \vee i = k \wedge p_i = \mathrm{N}) \end{cases}$$

遷移を実行した後もタイマ値が 0 のタイマがある場合は，その中で最も番号の若いタイマのタイムアウト遷移を実行する．タイマ値が 0 のタイマがなくなれば，タイムアウト遷移は終了する．

M_T 上の到達可能な状態は，初期状態 $\langle s_0, \perp^n\rangle$ から新たに到達可能な状態 $\langle s, \tau\rangle$ が見つからなくなるまで，上記の遷移を調べることで列挙することができる．状態 $\langle s, \tau\rangle$ を用いた M_T 上で到達可能な状態を列挙するアルゴリズムを図 2に示す．

PASSED は M_T 上で到達可能な状態かつその状態から到達可能な状態が全て列挙されているような状態の集合である．WAIT は M_T 上で到達可能な状態であるがその状態から到達可能な状態を調べていない状態の集合である．アルゴリズム開始時は PASSED は空集合で，WAIT は M_T の初期状態 $\langle s_0, \perp^n\rangle$ のみを持つ．アルゴリズムは WAIT が空集合になるまで続く．WAIT が空集合でなければ，5 行目で WAIT から状態を 1 つ取り出し，その状態

```
1  PASSED := φ
2  WAIT := {⟨s0, ⊥^n⟩}
3  repeat
4    begin
5      get ⟨s, τ̄⟩ from WAIT
6      if τ̄ ≠ τ̄'' for all ⟨s, τ̄''⟩ ∈ PASSED then
7        begin
8          PASSED := PASSED ∪{⟨s, τ̄⟩}
9          WAIT := WAIT ∪{⟨s', τ̄'⟩|⟨s, τ̄⟩ → ⟨s', τ̄'⟩}
10       end
11   end
12 until WAIT=φ
```

図 2　状態 $\langle s, \bar{\tau}\rangle$ を用いた列挙アルゴリズム

が PASSED に存在しない状態であれば 8 行目で PASSED に加え，9 行目でその状態から到達可能な状態を WAIT に加える．この動作は M_T 上で到達可能な状態の数だけ行われる．到達可能な状態は有限なので，5 行目で WAIT から状態を取り出す動作により，有限時間内で WAIT が空集合になりアルゴリズムは終了する．アルゴリズム終了時には，PASSED の中に M_T 上で初期状態 $\langle s_0, \perp^n\rangle$ から到達可能な状態の集合が入ることになる．

しかしこの方法ではタイマの設定時間が大きい場合，到達可能な状態数が膨大となり計算時間やメモリの点で問題となる．

4.2　提案手法

提案手法では複数の状態と状態遷移を扱うために，各タイマのタイマ値を整数解として持つ連立不等式 R を用いることにする．R の中の各不等式は以下の形とする．

$$t_i - t_j \leq C_{ij} \ (0 \leq i, j \leq n, i \neq j)$$

t_0 は定数 0 を表す変数，$t_i (i \neq 0)$ は $\perp$ でないタイマ値を持つタイマ i を表す変数である．C_{ij} は R におけるタイマ i とタイマ j の差の上限を表す定数である．特に C_{i0} は R におけるタイマ i の上限，$-C_{0j}$ は R におけるタイマ j の下限を表す．また $\perp^n$ は不等式を 1 つもを持たない連立不等式 $R_\perp$ で表す．

M の状態 s と連立不等式 R の組 $\langle s, R\rangle$ で M_T 上の状態集合 $\{\langle s, \bar{\tau}\rangle | \bar{\tau} = (t_1, \ldots, t_n), t_i$は R の整数解 $\}$ を表すことができる．状態集合 $\langle s, R\rangle$ において，時間経過によりタイマ k がタイムアウトする可能性がある時，状態集合 $\langle s, R\rangle$ でタイマ k がタイムアウト可能という．タイマ i がタイムアウトする条件は変数 t を用いて以下のように記述できる．

$$\begin{cases} t_k = 0 & \\ t_k \leq t_j + 1 & (1 \leq j \leq k-1) \\ t_k \leq t_j & (k+1 \leq j \leq n) \end{cases}$$

これより状態集合 $\langle s, R\rangle$ において状態 s でタイマ k がタイムアウト可能とは，状態集合 $\langle s, R\rangle$ において R' が解を持つ時である．ただし，

$$C'_{ij} = \begin{cases} 0 & (i = 0 \wedge j = k \vee i = k \wedge j = 0) \\ 1 & (i = k \wedge 1 \leq j \leq (k-1)) \\ 0 & (i = k \wedge (k+1) \leq j \leq n) \\ C_{ij} & \text{その他} \end{cases}$$

R' は差分不等式の集合で構成されているため，この解の存在判定は $O(lm)$ でできる [8]．この状態集合 $\langle s, R\rangle$ を用いて M_T 上で到達可能な状態を列挙する．到達可能な状態の列挙が終了すれば，列挙した状態集合 $\langle s, R\rangle$ で各タイマがタイムアウト可能かを調べることで状態 s でタイムアウトする可能性のあるタイマを特定することができる．

ここで状態集合 $\langle s, R\rangle$ を用いた時の M_T 上の遷移について説明する．前節と同様に (1)(2-a)(2-b) について考える．

(1) M で $t = (u, v, x, y, \bar{p})$ が実行された場合，$\langle u, R\rangle$ から $\langle v, R[\bar{p}]\rangle$ へ遷移する．ただし，

$$C[\bar{p}]_{i0} = \begin{cases} T_i & (p_i = \mathrm{S}) \\ C_{i0} & (p_i = \mathrm{N}) \\ - & (p_i = \mathrm{D}) \end{cases}$$

$$C[\bar{p}]_{0j} = \begin{cases} -T_j & (p_j = \mathrm{S}) \\ C_{0j} & (p_j = \mathrm{N}) \\ - & (p_j = \mathrm{D}) \end{cases}$$

$$C[\bar{p}]_{ij} = \begin{cases} T_i - T_j & (p_i = \mathrm{S} \wedge p_j = \mathrm{S}) \\ C_{ij} & (p_i = \mathrm{N} \wedge p_j = \mathrm{N}) \\ T_i + C_{0j} & (p_i = \mathrm{S} \wedge p_j = \mathrm{N}) \\ C_{i0} + T_j & (p_i = \mathrm{N} \wedge p_j = \mathrm{S}) \\ - & (p_i = \mathrm{D} \vee p_j = \mathrm{D}) \end{cases}$$

R において C_{ij} が存在せず，かつ R' において C'_{ij} が定義されている場合は $R[\bar{p}]$ に不等式 $t_i - t_j \leq C'_{ij}$ を追加する．また"-" は対応する不等式を削除することを意味する．

(2-a) タイムアウト遷移が起こらない場合，時間経過によって状態 $\langle u, R\rangle$ から状態 $\langle u, R\uparrow\rangle$ へ遷移する．ただし，

$$C\uparrow_{ij} = \begin{cases} -1 & (i = 0) \\ C_{ij} & (i \neq 0) \end{cases}$$

(2-b) 状態集合 $\langle u, R\rangle$ でタイマ k がタイムアウト可能ならば，まず $\langle u, R\rangle$ から $\langle u, R(k)\rangle$ へ遷移する．ただし，

$$C(k)_{ij} = \begin{cases} 0 & (i = 0 \wedge j = k \vee i = k \wedge j = 0) \\ C_{kj} - C_{0k} & (i = 0 \wedge j \neq k) \\ C_{ik} - C_{k0} & (i \neq k \wedge j = 0) \\ C_{ij} & \text{その他} \end{cases}$$

その後タイムアウト遷移 $(u, v, \text{Timeout}_k, y, \overline{p}) \in H$ を実行し，$\langle u, R(k)\rangle$ から $\langle v, R(k)[\overline{p}]\rangle$ へ遷移する．遷移を実行した後も $R(k)[\overline{p}]$ で $C_{i0} = C_{0i} = 0$ であるタイマ i があれば，その中で最も番号の若いタイマのタイムアウト遷移を実行する．$C_{i0} = C_{0i} = 0$ であるタイマ i がなくなれば，タイムアウト遷移は終了する．

(2-a) の遷移 $\langle u, R\rangle \rightarrow \langle u, R\uparrow\rangle$ は $\langle u, R\rangle \subseteq \langle u, R\uparrow\rangle$ なので，常に $\langle u, R\rangle$ を $\langle u, R\uparrow\rangle$ に置き換えることで省略できる．よって状態集合 $\langle u, R\rangle$ の遷移は (1) $\langle u, R\rangle \rightarrow \langle v, R[\overline{p}]\uparrow\rangle$ (2-b) $\langle u, R\rangle \rightarrow \langle v, R(k)[\overline{p}]\uparrow\rangle$ (ただし $R\uparrow = R$) の 2 種類となる．M_T 上の到達可能な状態は初期状態 $\langle s_0, R_\perp\rangle$ から新たに到達可能な状態が見つからなくなるまで上記の遷移を調べることで列挙することができる．状態集合 $\langle s, R\rangle$ を用いた M_T 上で到達可能な状態を列挙するアルゴリズムを図 3に示す．

```
1  PASSED := φ
2  WAIT := {⟨s0, R⊥⟩}
3  repeat
4    begin
5      get ⟨s, R⟩ from WAIT
6      if R ⊈ R'' for all ⟨s, R''⟩ ∈ PASSED then
7        begin
8          PASSED := PASSED ∪{⟨s, R⟩}
9          WAIT := WAIT ∪{⟨s', R',⟩|⟨s, R⟩ → ⟨s', R'⟩}
10       end
11   end
12 until WAIT=φ
```

図 3　状態集合 $\langle s, R\rangle$ を用いた列挙アルゴリズム

PASSED は M_T 上で到達可能な状態集合かつその状態集合から到達可能な状態集合が全て列挙されているような状態集合の集合である．WAIT は M_T 上で到達可能な状態集合であるがその状態集合から到達可能な状態集合を調べていない状態集合の集合である．アルゴリズム開始時は PASSED は空集合，WAIT は M_T の初期状態 $\langle s_0, R_\perp\rangle$ のみを持つ．R はタイマ値ベクトルの集合なので既に列挙されている状態集合かどうか (6 行目) は，R 同士の包含関係を調べることで判定する．$R \subseteq R'$ かどうかは C_{ij} と C'_{ij} の大小比較によって判定できる．他は前節で述べたアルゴリズムと同様の動作を行う．アルゴリズム終了時には，PASSED の中に M_T 上で初期状態 $\langle s_0, R_\perp\rangle$ から到達可能な状態の集合が入ることになる．

提案手法により複数の状態と状態遷移を一括して扱い，M_T 上の到達可能な状態を効率良く列挙できると考えられる．

5. 実験による評価

前章で述べた到達可能な状態の列挙手続きを作成し，例題に適用し，列挙する状態数と計算時間について調べた．

5.1 例題

小規模な例題として競り上げ式オークションを監視するシステム (図 4) を用いた．これはタイマ 1 により参加者の入札間隔を監視し，一定時間入札が行われない場合，その入札者が品物を落札する．またタイマ 2 によりオークションの制限時間を監視する．制限時間になった場合は，最終入札者が品物を落札する．

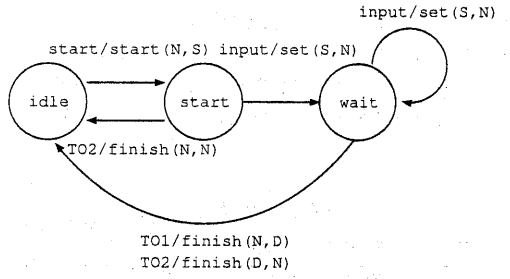

図 4　競り上げ式オークションを監視するシステム

また実用規模の例題として TCP/IP ネットワークに接続されているホストにコンフィグレーション情報を伝達するための仕組みである DHCP(Dynamic Host Configuration Protocol)[9] のクライアント側のプロトコルを用いた．DHCP はクライアント・サーバ方式で，指定された DHCP サーバが IP アドレスを動的に割り当て，コンフィグレーションパラメータとともにクライアントに送信する．DHCP では，IP アドレスの割り当て方法として，「自動割り当て」，「動的割り当て」，「手動割り当て」の 3 種類の仕組みがある．「動的割り当て」では，DHCP サーバは，クライアントに対して IP アドレスを一定期間（もしくは，クライアントがアドレスを返納するまで）貸し出す．クライアントは，サーバに対してアドレス割り当てを要求する時に，IP アドレスの貸し出し期間を指定することができる．貸し出し期間を管理するために，DHCP サーバ，クライアントがともにタイマを利用する．DHCP においては時間は秒単位で扱われる．最小の IP アドレス貸し出し期間は 1 時間である．また，クライアントはサーバへメッセージを送る際にタイムアウトに基づいてメッセージ再送を行う．最初の再送間隔は 4 秒で，再送するごとに間隔が 2 倍になる．クライアントが IP アドレスの貸し出し期間を指定できるようにすると，本研究で対象としているモデルでは記述できないので，IP アドレスの貸し出し期間を 1 時間に限定して，例題として使用することにした．

それぞれの例題をタイマを用いる有限状態機械モデルで記述すると，オークションを監視するシステムは状態数3，状態遷移数5，使用するタイマ数2となり，DHCPのクライアント側のプロトコルは状態数11，状態遷移数74，使用するタイマ数9となった．

5.2 実験結果

作成した列挙手続きを各例題に対して適用した結果，提案手法を用いた場合は，どちらの例題に対しても到達可能な状態の列挙を行うことができた．単純な合成による手法を用いた場合は，DHCP(最も設定時間の大きなタイマの設定時間=3600)に対して列挙を行うことができなかった．また各タイマの設定時間を小さく抑えた例題に対して同様に列挙手続きを適用しタイマの設定時間の大きさが各手法に与える影響を調べた．列挙した状態数，計算時間を表1に示す．計算時間の単位は秒である．

表1 実験結果

例題	状態数	計算時間
▷状態 $\langle s, \overline{r} \rangle$ を用いた列挙		
AUCTION($\max(T_i) = 100$)	916	0.12
AUCTION($\max(T_i) = 10$)	40	0.0003
DHCP($\max(T_i) = 3600$)	-	-
DHCP($\max(T_i) = 200$)	7096	16
▷状態集合 $\langle s, R \rangle$ を用いた列挙		
AUCTION($\max(T_i) = 100$)	3	0.0003
AUCTION($\max(T_i) = 10$)	3	0.0003
DHCP($\max(T_i) = 3600$)	278	20
DHCP($\max(T_i) = 200$)	278	20

提案手法では状態集合を列挙することで到達可能な状態を効率良く列挙しているといえる．提案手法では，列挙にかかる手間は有限状態機械の状態数や使用するタイマの数が増えると大きくなる．しかしタイマの設定時間を小さく抑えた例題に対しても列挙する状態数，計算時間共に変化はみられなかったので，提案手法の列挙にかかる手間はタイマの設定時間に依存しないことがわかった．

6. まとめ

本論文では，タイマを用いる有限状態機械モデルで記述されたシステムに対する検証手続きについて議論した．

システムの到達可能な状態を効率良く列挙するために，タイマの取り得る値を連立不等式を用いて表し，複数の状態と状態遷移を一括して列挙する手法を提案した．

また例題を用いて実験を行ったところ，実用規模の例題としてDHCPのクライアント側のプロトコルを用いた場合，単純な合成による手法では列挙を行うことができなかったが，提案手法では現実的な時間内で列挙することができた．

また，使用するタイマの設定時間を小さく抑えた例題を用いて同様に実験を行ったところ，提案手法の列挙に要する手間は，検証の対象となる有限状態機械の状態数や使用するタイマ数に依存するが，タイマの設定時間には影響を受けないことがわかった．

References

[1] G. Behrmann, K. G. Larsen, J. Pearson, C. Weise, and W. Yi: Efficient Timed Reachability Analysis using Clock Difference Diagrams, 11th International Conference on Computer-Aided Verification, LNCS. 1633 July 7-10, (1999)

[2] K. Strehl, L. Thiele: Symbolic Model Checking of Process Networks Using Interval Diagram Techniques, Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD-98), San Jose, California, pp.686-692, (1998)

[3] M. Bozga, O. Maler, A. Pnueli, and S. Yovine: Some progress in the symbolic verification of timed automata, *Lecture Notes in Computer Science*, 1254:179-190, (1997)

[4] R.Alur and David L. Dill: A theory of timed automata, Theoretical Computer Science, Vol.126, pp.185-235(1994)

[5] R.Alur: Timed Automata, 11th International Conference on Computer-Aided Verification, LNCS 1633, pp.8-22(1999)

[6] 徳田康平，森亮憲，樋口昌宏: タイマを使用した通信プロトコルの仕様記述の検証法, 第60回情報処理学会 全国大会講演論文集, 2P-03, (2000)

[7] 森亮憲，樋口昌宏: タイマシステムコールを用いるFSMプロトコルの適合性試験について, 情報処理学会研究報告, DPS 99-56, pp.121-126(1999)

[8] T.H.Cormen, R.L.Rivest: Introduction to Algorithms, The MIT Press, pp.539-543(1990)

[9] R.Droms : Dynamic Host Configuration Protocol, RFC 2131, Bucknell University (1997-03)