数理解析研究所講究録
第1649巻 2009年 39-46

確率時間ゲーム理論による組込みシステムのモデル化，仕様記述及び検証

金沢大学自然科学研究科電子情報工学専攻　林　将志 (Masashi Hayashi)　山根　智 (Satoshi Yamane)
Division of Electrical and Computer Engineering,
Kanazawa University Graduate School of Natural Science & Technology

1　まえがき

近年，マイクロプロセッサの99％以上は組込みシステムに使われており，制御システムから情報家電までの多岐に渡り，safety-criticalな状況で使われることが多く，大規模化しており [1]，組込みシステムの設計を難しくしている要因としては，以下の3つが知られている [2].

1. オープンシステムとして様々な構成要素が並行に動作していること [3]
2. デジタルな離散的状態遷移に加えて，タイミング制約や制御法則などのアナログ動作が混在しているといったリアルタイム性があること [4]
3. 組込みシステムが動作する環境の不確定性があること [5]

これらの要因を解決するために，今までに，以下の手法が開発されている.

1. システムと環境との間のゲーム理論により，オープンシステムの構成要素の並行動作，つまり，リアクティブ性をモデル化している [6].
2. デジタル動作とアナログ動作が混在しているシステムの記述のために，時間オートマトンやハイブリッドオートマトンが開発されている [7, 8].
3. 組込みシステムが動作する環境の不確定性を表現するために，確率オートマトンや確率時間オートマトンが開発されている [9, 10].

注目すべき既存研究として，リアクティブ性とアナログ動作を同時にモデル化するための，A.Pnueliらの時間ゲームの研究がある [11]. 時間ゲームには，以下の重要な特徴がある；
（1）ゲームには，システムと環境が交互に動作を提案するturn-basedゲーム及び同時並行に動作を提案するconcurrentゲームがある．時間ゲームはconcurrentゲームである.
（2）時間ゲームには，遅延動作と離散動作があり，遅延動作よりも離散動作が優先される.

一方，A.Pnueliらの時間ゲームにおいて，到達可能ゲームを効率的に検証するアルゴリズムがK.G.Larsenらによって提案されている [12].

本論文では，我々は，ワイヤレスLAN[13]などの確率リアルタイムシステムのゲーム理論的な仕様記述と検証を提案する．A.Pnueliらの時間ゲーム [11] を確率で拡張して，concurrentな確率時間ゲーム理論を提案して，さらに，K.G.Larsenらの検証アルゴリズム [12] を確率で拡張して確率時間ゲームの到達可能ゲーム検証アルゴリズムを提案する.

以下，2節で意味モデルである確率時間ゲームを，3節で仕様記述言語確率時間ゲームオートマトンをそれぞれ定義し，4節で検証法を説明し，最後に5節で，まとめと今後の課題について述べる.

2　確率時間ゲーム

2.1　確率時間ゲーム

定義 1 離散確率分布
集合 S 上の有限な確率分布の集合を $\mu(S)$ とする．$p \in \mu(S)$ は関数 $p\colon S \to [0,1]$ である．ただし，$\Sigma_{s\in S}p(s) = 1$ である. □

確率時間ゲームの構造を，以下のように定義する.

定義 2 確率時間ゲーム
確率時間ゲームを組 $G = (S, s_0, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta)$ で定義する.

(1) S は状態の集合

(2) $s_0 \in S$ は初期状態

(3) $Acts_1, Acts_2$ はそれぞれプレイヤー *1, 2* のアクションの集合．$Acts_1, Acts_2$ は互いに素な集合である．プレイヤーの行動は時間領域 $\boldsymbol{R}$ とアクションの組となる．$M_i = \mathbf{R} \times Acts_i (i=1, 2)$

(4) $\Gamma_i\colon S \to 2^{M_i}$ は，状態に対してプレイヤー i が選択可能な行動を割り当てる関数である.

(5) $\delta\colon S \times (M_1 \cup M_2) \to \mu(S)$ は遷移関数. □

確率時間ゲームは二人非協力ゲームであり，各プレイヤーは状態sにおいて，行動$m_i \in \Gamma(s)$を同時に，かつ独立的に選択し，その結果，どちらか一方の行動m_iが選択され，$\delta((s,m_i))$により，不確定に次の状態に遷移する．

確率時間ゲームを実際のシステム上で次のように適用する．

1. プレイヤーはコントローラと環境である．（プレイヤー1がコントローラ，プレイヤー2が環境）
2. コントローラと環境は，同時に，かつ独立して動作している．
3. コントローラと環境がシステムに対して行った行動により，システムは確率的に変化する．

システムをコントローラ，または環境により動作するものとし，環境はシステムの正常な動作を妨害するものとして，定義する．

各プレイヤーが行動を起こした時に，どちらか一方のプレイヤーの行動を選択することにより，システムの状態遷移が起きる．

行動の選び方を次のように定義する．

プレイヤー1が行動$\langle \Delta_1, a_1 \rangle \in \Gamma_1(s)$を起こし，同時に，プレイヤー2が行動$\langle \Delta_2, a_2 \rangle \in \Gamma_2(s)$を起こした時，$\Delta_1 < \Delta_2$ならば，$\langle \Delta_1, a_1 \rangle$が，選択される．$\Delta_2 < \Delta_1$ならば，$\langle \Delta_2, a_2 \rangle$が，選択される．$\Delta_1 = \Delta_2$ならば，$\langle \Delta_2, a_2 \rangle$を選択する．今回は，このゲームのプレイヤー1をコントローラ，プレイヤー2を環境と考えて検証を行う．従って，同時にアクションが起きる場合は，環境を優先して選択することで，より安全な検証を行う．

形式的には，選択関数$\tilde{\delta}: S \times M_1 \times M_2 \to \mu(S)$を用いて以下のように定義する．

$$\tilde{\delta}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \delta(s, \langle \Delta_1, a_1 \rangle) & \Delta_1 < \Delta_2 \\ \delta(s, \langle \Delta_2, a_2 \rangle) & \Delta_1 \geq \Delta_2 \end{cases}$$

次に，確率時間ゲームの表記法として，確率時間ゲームオートマトンを定義する．

3 確率時間ゲームオートマトン

確率時間オートマトンを拡張して，確率時間ゲームオートマトンを定義する．確率時間ゲームオートマトンを確率時間ゲームとして，その動作を定義する．

3.1 構文

定義 3 クロック変数
クロック変数は非負の実数値をとる変数であり，同じ速さで増加し，アクションが起こるときに0にリセットすることができる．$\boldsymbol{R}$上のクロック変数の集合をXとする． □

次にクロック制約条件とクロック変数の評価を定義する．クロック制約により，遷移のタイミング制御を行う．

定義 4 クロック制約
X上のクロック制約は$x \prec c$または$x - y \prec c$のブール結合である．ここで，cは整数，$x, y \in X, \prec$は$<$または$\leq$である．X上のすべてのクロック制約の集合を$\Xi[X]$と表記する． □

定義 5 クロック変数の評価
クロック変数の評価は関数$v: X \to \mathbf{R}$である．Xのすべてのクロック変数に0を割り付ける評価を$\boldsymbol{0}$と表記する．Xのすべての評価を$V(X)$と表記する．評価$v \in V(X)$を与えたとき，すべてのクロック変数$x \in X$に対して，評価$(v+\delta)(x) = v(x) + \delta$を$v + \delta$と表記する．クロック変数の集合$r \subseteq X$に対して，$r$の中のクロック変数に0を割り当てることを$v[r := 0]$と表記する．クロック制約$g \in \Xi$と評価$v$に対して，$v$が$g$を満たすならば，$v \models g$と表記する．また，$[\![g]\!] = \{v \in V(X) | v \models g\}$とする． □

確率時間ゲームオートマトンを以下のように定義する．

確率時間ゲームオートマトンは確率時間オートマトン[10]のアクションを制御可能なものと制御不能なものとに分けることにより実現する．制御不能なアクションは環境のアクションである．これにより，確率時間ゲームオートマトン上の遷移は2種類存在する．

定義 6 確率時間ゲームオートマトン
確率時間ゲームオートマトンは組$A = (Q, q_0, X, Acts_c, Acts_u, Inv, \rho)$で定義する．

(1) Qはロケーションの有限集合．

(2) $q_0 \in Q$は初期ロケーション．

(3) Xはクロック変数の有限集合．

(4) $Acts_c$は制御可能*(controllable)*なアクション．

(5) $Acts_u$は制御不能*(uncontrollable)*なアクション．

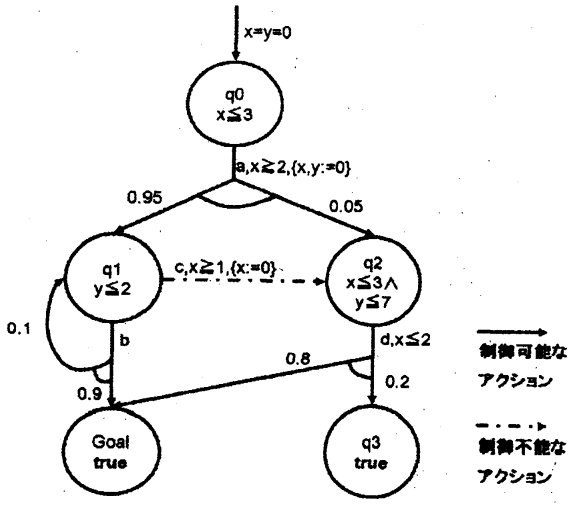

図 1: 確率時間ゲームオートマトンの一例

(6) $Inv\colon Q \to \Xi[X]$ は各ロケーションに不変式を割り当てる関数.

(7) $\rho \subseteq Q \times \Xi[X] \times \{Acts_c \cup Acts_u\} \times 2^X \times \mu(Q)$ は遷移関係.$(q,g,a,r,p) \in \rho$ に対して，$q \in Q$ は遷移元のロケーション，$g \in \Xi[X]$ は遷移が起きるときのクロック制約条件，$a \in Acts_c \cup Acts_u$, は遷移にラベル付けられたアクション，$r \in 2^X$ は遷移によってリセットされるクロック集合，$p \in \mu(Q)$ はロケーション上の確率分布である. □

3.2 意味

確率時間オートマトンの状態はロケーションとクロックの評価値の組 $(q,v) \in Q \times V(X)$ である. $S = Q \times V(X)$. プレイヤー i は，Δ 時間経過する間，$Inv(q)$ が満たされているならば，すなわち，すべての $\Delta' \leq \Delta$ に対して $v+\Delta' \models Inv(q)$ となっているならば，行動 $\langle \Delta, \perp \rangle$ が (q,v) で可能で，行動 $\langle \Delta, \perp \rangle$ により，状態 $(q, v+\Delta)$ へ遷移する．この遷移を $(q,v) \to (q, v+\Delta)$ と表記する.
状態 (q,v) で，アクション $a \in Acts_i$ に対して，以下の条件を満たすとき，行動 $\langle \Delta, a \rangle$ が可能である.

1. Δ 時間経過するとき，$Inv(q)$ が連続的に満たされる.
2. ρ 内に $(q, v+\Delta)$ で可能な遷移 (q,g,a,r,p) が存在する.
3. $Inv(q')$ が満たされている．ただし，$p(q') \neq 0$.

行動 $\langle \Delta, a \rangle$ は (q', v') へ遷移させる．ここで，v' は $v+\Delta$ において r 内の全てのクロック変数をリセットしたものである．またこのとき $(q, v+\Delta) \xrightarrow{a} (q', v')$ と表記する.
形式的に，確率時間ゲームオートマトン $A = (Q, q_0, X, Acts_c, Acts_u, Inv, \rho)$ は確率時間ゲーム $G = (S, s_0, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta)$ として意味づけられる．$S = Q \times V(X)$, $Acts_1 = Acts_c \cup \{\perp\}, Acts_2 = Acts_u \cup \{\perp\}$, 各状態 $\langle q, v \rangle \in S$ に対して，$\Gamma_i(\langle q, v \rangle)$ は

$$\Gamma_i(\langle q,v\rangle) = \{\langle \Delta, a\rangle \in M_i \mid \forall \Delta' \in [0,\Delta].$$
$$v+\Delta' \models Inv(q) \wedge (a \neq \perp \Rightarrow \exists q' \in Q.$$
$$((q,g,a,r,p) \in \rho \wedge (v+\Delta) \models g \wedge p(s') \neq 0 \wedge$$
$$(v+\Delta)[r:=0] \models Inv(q'))\} \cup \{\langle 0, \perp\rangle\}.$$

遷移関数 δ は $\delta(\langle q,v\rangle, \langle \Delta, a\rangle) = \tilde{p}$
全ての $\langle \Delta, a\rangle \in \Gamma_i(\langle q,v\rangle), \langle q', v'\rangle \in S$ に対して
$a = \perp$ のとき

$$\tilde{p}(\langle q', v'\rangle) = \begin{cases} 1 & \langle q', v'\rangle = \langle q, v+\Delta\rangle \text{ のとき} \\ 0 & \text{上記以外} \end{cases}$$

$a \neq \perp$ のとき，$(q,g,a,r,p) \in \rho$ に対して

$$\tilde{p}(\langle q', v'\rangle) = \sum p(q')$$

定義 7 パス
確率時間ゲームオートマトン A の状態 (q,v) から始まる状態列の集合を $Paths((q,v), A)$ と表記し，$Paths((q_0, \mathbf{0}), A)$ を $Paths(A)$ と表記する．また，有限状態列 ω の最後の状態を $last(\omega)$ とし，ω の n 番目の状態を $\omega(n)$ と記す. □

確率時間ゲームオートマトンの実行例は以下のようになる.
$(q_0, \mathbf{0}) \xrightarrow{\Delta_0} (q_0, \mathbf{0}+\Delta_0) \xrightarrow{g_0,r_0,a_0,p_0} (q_1, \mathbf{0}+\Delta_0[r_0 := 0]) \xrightarrow{\Delta_1} (q_1, \mathbf{0}+\Delta_0[r_0:=0]+\Delta_1) \xrightarrow{g_1,a_1,r_1,p_1} (q_2, (\mathbf{0}+\Delta_0[r_0:=0]+\Delta_1)[r_1:=0]) \to \cdots$ 各状態においてシステム動作を決定させる戦略を次のように定義する.

定義 8 戦略
プレイヤー i の戦略は状態列から行動への関数で $f_i\colon S^+ \to M_i$ を純戦略といい，$f_i\colon S^+ \to \mu(M_i)$ を混合戦略と呼ぶ．ここで，S^+ は状態の 1 回以上の繰り返しである．ゼロ和 2 人ゲームでは，混合戦略

の範囲で均衡点が存在することが知られているため，*[19]*各プレイヤーの戦略を混合戦略とする．また，混合戦略は純戦略を包含しているため，混合戦略の場合のみを考えれば十分である．$\forall\omega,\omega' \in Paths(A)$ に対して，$last(\omega) = last(\omega')$ ならば $f(\omega) = f(\omega')$ とする． □

各プレイヤーの戦略により，確率時間ゲームのパスが決定される．

定義 9 ゲームの経歴
プレイヤー i の戦略 f_i により，ゲームの経歴が決定される．ゲームの経歴 $Hist((q,v), f_1, f_2)$ はパスの部分集合であり，帰納的に以下のように定義される．

- $(q,v) \in Hist((q,v), f_1, f_2)$
- $\omega \in Hist((q,v), f_1, f_2)$ ならば $\omega' = \omega \xrightarrow{e} (q',v')$ ただし以下の条件を満たすものとする．
  - $\omega' \in Paths((q,v), A)$
  - $e \in M_1 \cup M_2$
  - $e = \tilde{\delta}(last(\omega), f_1(\omega), f_2(\omega))$

□

4 到達ゲームの検証

4.1 利得行列不変条件

確率時間ゲームの状態は，ロケーションとクロック値の組で表わされるため，無限の状態数を持つこととなる．そこで，連続した無限の状態空間を有限の集合に置き換えた後に均衡点を計算し，その確率値を求める．確率時間ゲームオートマトンの各状態において，利得行列の行はコントローラのアクション，列は環境のアクションとして利得行列が形成される．また，各プレイヤーは $\perp$ がどの状態でも可能であるため利得行列は必ず存在する．ただし，ロケーション内に利得行列が複数あるため同一ロケーション内で均衡点が一意には決まらない．そこで，新たに利得行列不変条件を導入し各利得行列に対して均衡点を計算する．利得行列不変条件をその条件内では利得行列が変化しないクロック条件として定義する．

example 1 図 *2* のような確率時間ゲームオートマトンのロケーションにおいて，クロック値の値により，利得行列が変化していく．

そこで，新たに利得行列不変条件を定義し，この条件に従いノードを生成し，このグラフに対して均衡点を計算する．

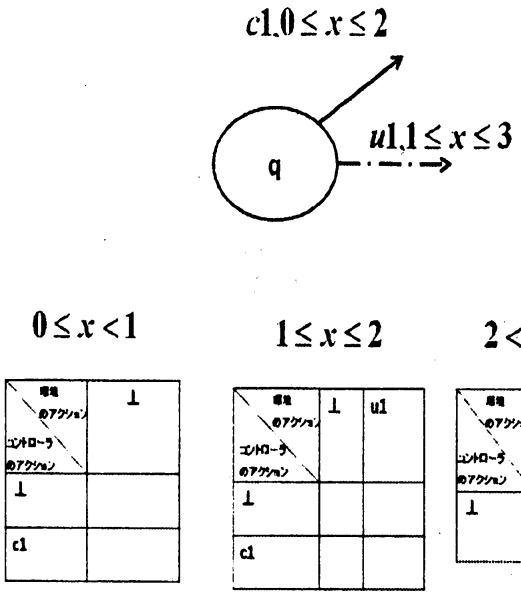

図 2: ロケーション内での利得行列の変化

以降では，このグラフを確率時間ゲームグラフと呼ぶ．
まず，確率時間ゲームグラフを定義するために，準備を行う．

定義 10 利得行列不変条件
確率時間ゲームオートマトンの各ロケーション $q \in Q$ に対して，利得行列不変条件の集合を割り当てる関数 $mInv : Q \to 2^{\Xi(X)}$ を以下のように定義する．
$\Gamma(q,v)$, $(q,g,a,r,p) \in \rho$ に対して
$mInv(q) = \{(q,h) \mid h = [\![h_0 \wedge h_1 \wedge \cdots \wedge h_n]\!], h_i = g$ または $\bar{g}\}$
また，$mInv(Q) = \{mInv(q) \mid q \in Q\}$ と表記する．
□

定義 11 *successor*
*(1)**discrete-successor***
$X \subseteq S$ と $a \in \{Acts_c \cup Acts_u\}$ に対して，X の *a-successor* を以下のように定義する．

$$\mathbf{Post}_a(X) = \{(q',v') \mid \exists(q,v) \in X, (q,v) \xrightarrow{a} (q',v')\}$$

*(2)**timed-successor***
timed-successor を以下のように定義する．

$$X \nearrow = \{(q,v+d) \mid (q,v) \in X, v, v+d \models Inv(q), d \geq 0\}$$

□

次に，確率時間ゲームグラフを定義する．

定義 12 確率時間ゲームグラフ
確率時間ゲームグラフを遷移システム $(mInv(Q), S_0, \rightarrow)$ と定義する．ここで，

1. $mInv(Q)$ は Q 上の利得行列不変条件の集合
2. $S_0 = (q_0, h_0)$
   h_0 はクロック評価 $\boldsymbol{0}$ を含むようなロケーション q_0 の利得行列不変条件．
   $(q_0, v_0) \in mInv(Q), \mathbf{0} \in h_0$
3. $\rightarrow$ は時間遷移 $\xrightarrow{1}$ と離散遷移 $\xrightarrow{a}$ の和集合
   - 時間遷移
     $mInv' \in mInv(q)$ かつ $mInv + \epsilon \models mInv'(0 < \epsilon < 1)$ を満たす $mInv'$ が存在するとき，
     $(q, mInv) \xrightarrow{1} (q, mInv')$.
   - 離散遷移
     $(q, g, a, r, p) \in \rho$ かつ $mInv' = ((mInv \cap [\![g]\!])[r := 0]) \nearrow$ が存在するとき，
     $(q, mInv) \xrightarrow{a} (q', mInv')$. ただし，
     $p(q') \neq 0$ □

確率時間ゲームの到達ゲームの検証を行う．まず到達ゲームに関する定義をし，検証を行う．検証は次のような手順で行う．

1. **Goal** に到達可能な状態を全て求める．
2. 各状態の均衡点を求め，ゲームの結果により，**Goal** に到達する確率を求める．

ここで，**Goal** は到達すべきロケーションの集合である．つまり，与えられた確率時間ゲームオートマトンの初期状態 $(q_0, \mathbf{0})$ から **Goal** $\subseteq Q$ への到達可能性を検証する．
制御不能なアクションはシステムが **Goal** に到達するのを阻止するようなものと考える．つまり，制御不能なアクションでは **Goal** に到達できないとして検証を行う．そこで，K.G. Larsen らの考え方 [12] を採用して，次のように最大パスを定義する．

定義 13 最大ゲーム
下記の条件を満たすゲームの経歴を最大ゲームとする．

- 長さが無限
- 長さが有限で次の条件のどちらかを満たすもの
  1. $last(\omega) \in \mathbf{Goal} \times V(X)$
  2. $\omega \xrightarrow{a}$ ならば，$a \in Acts_u$

□

つまり，長さが有限ならば最終的に **Goal** に到達するようなものを考える．しかし，**Goal** に到達する前にシステムの行える制御可能なアクションが無い為にシステムが停止する可能性のあるパスも考慮に入れて (2). の条件を付ける．

zeno となるときには正しく検証を行うことができないため，記述されたオートマトンに対し，zeno の判定を行うことが必要である．確率時間ゲームオートマトンの zeno の判定には，既存の確率時間オートマトンに関する判定法 [18] を利用する．確率時間ゲームオートマトンに対して，システムが到達ゲームの勝利を以下のように定義する．

定義 14 勝利ゲーム
各プレイヤーの戦略 f_i により，決定される最大ゲーム $\omega = (q_0, v_0) \xrightarrow{e_0} (q_1, v_1) \xrightarrow{e_1} \cdots \xrightarrow{e_n} (q_{n+1}, v_{n+1}) \cdots$ に対して，$k \geq 0$ で $(q_k, v_k) \in \mathbf{Goal} \times V(X)$ が存在するとき，そのゲームの経歴を勝利ゲームとする．
□

さらに，状態に関して勝利可能状態を以下のように定義する．

定義 15 勝利可能状態
状態 (q, v) に対して，その状態から始まる勝利ゲームが存在するとき，その状態を勝利可能状態とする．
□

検証アルゴリズムは次のようなステップから成る．

ステップ 1 到達可能性判定を行う．

ステップ 2 各ノードに対して，均衡点を計算して確率を求める．

4.2 到達可能性判定

ステップ 1 では，時間ゲームに対するアルゴリズム [12] を確率遷移を含むようなオートマトンに拡張したアルゴリズムを用いる．与えられたオートマトンが **Goal** に到達するかを解析する．このアルゴリズムを図 3 に示す．このアルゴリズムはゲームグラフの作成を行いつつ，**Goal** への到達可能性を検証する．まず，初期状態から 0 以上の確率で到達可能な状態を求め，$Waitng$ に入れながら確率時間ゲームを作成する．到達可能な状態を求める際には $Waiting$ から一つ取り出し $e = (S, \alpha, S')$ として，S' に対して **Goal**

Input
$A = (Q, q_0, X, Acts_c, Acts_u, Inv, \rho)$
Initialization:
$S_0 = \{q_0, \mathbf{0}\}\nearrow$;
$Passed \leftarrow \{S_0\}$;
$Waiting \leftarrow \{(S_0, a, S') \mid S' = \mathbf{Post}_a(S_0)\nearrow\}$;
Main:
while $(Waiting \neq \emptyset)$ **do**
  $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
  if $Inc = \{S'' \mid S'' \in Passed, S' \subseteq S''\} \neq \emptyset$ **then**
    $Passed \leftarrow Passed \cup S'$;
    if $(\bigcup_{c \in Acts_c} \mathbf{Post}_c(S') \neq \emptyset)$ **then**
      $Waiting \leftarrow Waiting \cup \{(S', a, S'') \mid S'' = \mathbf{Post}_a(S')\nearrow\}$;
      while $(minv = \{m \mid m \in mInv(q), (q, v) \in S'\} \neq \emptyset)$ **do**
        $Waiting \leftarrow Waiting \cup \{(S', \perp, S'') \mid S'' \in minv, \{(S' + \epsilon) \cap minv\} \neq \emptyset\}$;
        $S' \leftarrow S''$
      endwhile
    endif
  endif
endwhile

図 3: 到達可能性検証アルゴリズム

か否かを判定する．ここで，$\alpha \in \{Acts_c \cup Acts_u \cup \perp\}$ である．また，その際に S' を $Passed$ に入れ，ノード作成済みであることを記憶する．この一連の操作を $Waiting$ の中身がなくなるまで繰り返す．ただし，この繰り返しは S' がすでに，$Passed$ に含まれている記号的状態 S'' に含まれる場合は新たに追加する操作をせずに，次の $Waiting$ の中身に対して，操作を行う．最終的に作成された確率時間ゲームグラフのノードに **Goal** が含まれていれば，オートマトンは **Goal** に到達可能であると言える．

図 1 のオートマトンに対してこのアルゴリズムを行った結果を図 4 に示す．確率時間ゲームグラフが作成され，このグラフに対して確率計算を行う．

4.3 確率計算法

図 3 のアルゴリズムにより作成されるグラフに対して，確率ゲームでの手法を用いて，確率値（均衡点）を求める．Alfaro らの手法 [16, 17] を用いて **Goal** に到達する確率を計算する．

4.3.1 均衡点

生成されたグラフの各ノードからの **Goal** への到達確率に基づき，各プレイヤーの利得を定義する．プレイヤー 1 の利得を到達確率 p とし，プレイヤー 2 の利得を $1-p$ と想定することにより，確率時間ゲームは 2 人ゼロ和非協力ゲームとなる．2 人非協力ゲームは混合戦略の範囲で均衡点が求まることが知られており，あるノードにおいて，ゲームの均衡点が，そのノードからの到達確率となる．

4.3.2 確率計算

確率時間ゲームの均衡点として，マックスミニ値を求める．ゲームグラフのノードに対して，図 5 のような利得行列となるとき，ノードの均衡点（マックスミニ値）を求めることは，以下の問題を解くこ

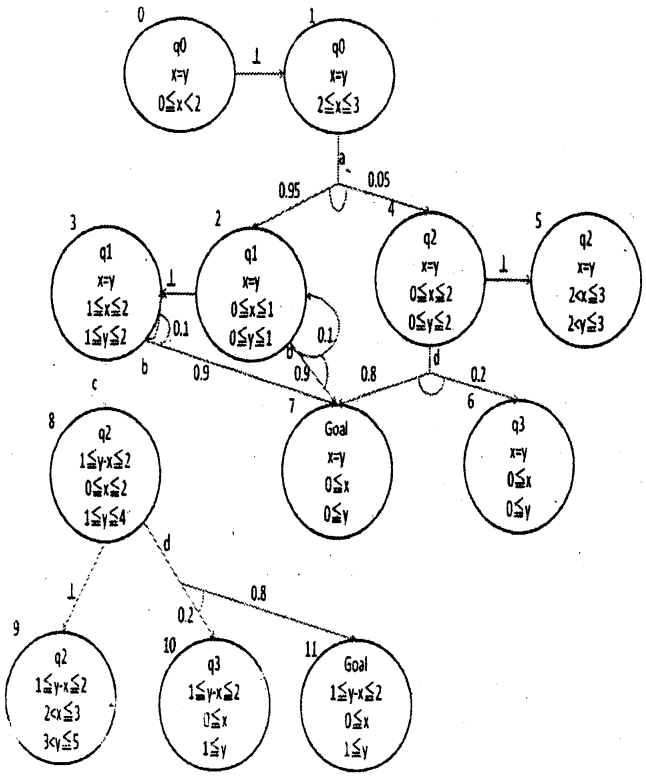

図 4: 確率時間ゲームグラフ

と に等しい.

$$
\begin{aligned}
&\text{最大化}: x_i \\
&\text{制約条件}: x_i \leq \sum_{0\leq k\leq m} r_k b_{k0} \\
&\qquad x_i \leq \sum_{0\leq k\leq m} r_k b_{k1} \\
&\qquad \vdots \\
&\qquad x_i \leq \sum_{0\leq k\leq m} r_k b_{kn} \\
&\qquad \sum_{0\leq k\leq m} r_k = 1
\end{aligned}
$$

ここで, $b_{ij} = \sum p(j)x_j, p = \delta((q,v),\langle\Delta,a\rangle), (q,v) \in i, \langle\Delta,a\rangle \in \Gamma((q,v))$

制約条件で, プレイヤー 2 の確率を最小化するような動作を表し, そのなかでプレイヤー 1 が到達確率を最大化することにより, マックスミニ値を求める.

example 2 ノード $\mathcal{J}(q1, 1 \leq x = y \leq 2)$ の利得行

環境 のアクション / コントローラ のアクション	u0	u1	···	un
c0	b00	b01	···	b0n
c1	b10	b11	···	b1n
⋮	⋮	⋮	⋮	⋮
cm	bm0	bm1	···	bmn

図 5: 利得行列

環境 のアクション / コントローラ のアクション	⊥	c
⊥	0	x_8
b	$0.1x_3 + 0.9x_7$	x_8

図 6: ノード 3 の利得行列

列は図 6 のようになる. したがって, ノード 3 においてマックスミニ値を求める問題は以下のようになる.

$$
\begin{aligned}
&\text{最大化}: x_3 \\
&\text{制約条件}: x_3 \leq r_0 0 + r_1(0.1x_3 + 0.9x_7) \\
&\qquad x_3 \leq r_0 x_8 + r_1 x_8 \\
&\qquad r_0 + r_1 = 1
\end{aligned}
$$

さらに, 各ノードのマックスミニ値は以下の問題を解くことで求められる.

$$
\begin{aligned}
&\text{最大化}: \sum x_i \\
&\text{制約条件}: x_i \leq \sum_{0\leq k\leq m} r_k b_{kl} (0 \leq l \leq n) \\
&\qquad \sum_{0\leq k\leq m} r_k = 1
\end{aligned}
$$

図4のグラフについて上記の問題を解くと，ノード0の均衡点（到達確率）は0.99となる．

5 まとめと今後の課題

今回の結果で到達ゲームに関しては不確定性を持つシステムと環境との相互作用を記述できた．またその記述を用いて，確率時間ゲームオートマトンの到達ゲームに関する検証を行い，その検証アルゴリズムの動作を実際に計算機上で確認をした．

今回は到達ゲームに関する検証を行ったが．今後は到達ゲーム以外のゲームに関しての検証を行う．

参考文献

[1] T.A. Henzinger, C.M. Kirsch. Embedded Software: Proceedings of the First International Workshop, EMSOFT '01. *LNCS 2211*, P.504, Springer-Verlag, 2001.

[2] T.A. Henzinger. Games, time, and probability: Graph models for system design and analysis.Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), *LNCS 4362*, pp. 103-110 , Springer, 2007,

[3] D. Harel, A. Pnueli. On the Development of Reactive Systems. *NATO ASI Series F, Vol. 13*, pp.477-498, SpringerVerlag, 1985.

[4] M.K. Inan, R.P. Kurshan. Verification of Digital and Hybrid Systems. *NATO ASI Series F: Computer and Systems Sciences*, Vol. 170, Springer-Verlag, 2000

[5] H.A. Hansson. Time and Probability in Formal Design of Distributed Systems. *PhD thesis, Upsala University*, 1991.

[6] A. Pnueli, R. Rosner A Framework for the Synthesis of Reactive Modules. *LNCS 335*, pp.4-17, Springer 1988.

[7] R. Alur, D.L. Dill. A theory of timed automata. *TCS*,Vol.126, pp.183-235, 1994.

[8] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, Pei-Hsin Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The Algorithmic Analysis of Hybrid Systems. *TCS*, Vol.138, No.1, pp. 3-34, 1995.

[9] R. Segala, N.A. Lynch. Probabilistic Simulations for Probabilistic Processes. *Nordic Journal of Computing*, Vol.2, No.2, pp.250-273, 1995.

[10] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston. Automatic verication of real-time systems with discrete probability distributions. *TCS 282*, pp 101-150, 2002

[11] O. Maler, A.Pnueli, J.Sifakis. On the Synthesis of Discrete Controllers for Timed Systems. *LNCS 900*, pp.229-242, 1995.

[12] F. Cassez, A. David, E. Fleury, K.G. Larsen, D. Lime: Efficient On-the-Fly Algorithms for the Analysis of Timed Games. *LNCS 3653*, pp.66-80, 2005.

[13] Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) Specifications . ANSI/IEEE Std 802.11, 1999 Edition.

[14] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. IEEE Trans. on Automatic Control, 43:1399-1418, 1998.

[15] A. Bianco and L. de Alfaro. Model Checking of Probabilistic and Nondeterministic Systems. In FST TCS 95: Foundations of Software Technology and Theoretical Computer Science, LNCS 1026, pp. 499-513. 1995.

[16] L. de Alfaro. Computing Minimum and Maximum Reachability Times in Probabilistic Systems. In CONCUR 99: 10th International Conference on Concurrency Theory, *LNCS 1664*, pp.66-81, 1999.

[17] L. de Alfaro, R. Majumdar: Quantitative Solution of Omega-Regular Games. In *Journal of Computer and System Sciences 68*, pp 374-397, 2004.

[18] M. Kwiatkowska, G. Norman, J. Sproston and F. Wang. Symbolic Model Checking for Probabilistic Timed Automata. Information and Computation, 205(7), pp. 1027-1077. July 2007.

[19] 鈴木光男，新ゲーム理論，(社) 勁草書房，東京，1994