社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－SE－154 (4)
2006／11／27

確率時間ゲーム理論に基づく リアルタイム組込みシステム設計検証手法

山根 智
金沢大学大学院自然科学研究科
syamane @ is.t.kanazawa-u.ac.jp

最近の組込みシステムは多数の構成要素からなり，各構成要素はオープンシステムであり，リアルタイム性があり，不確定性を有するものとして扱う必要がある．すなわち，組込みシステムを構成要素単位に仕様記述して，それらの構成要素を合成してシステムを構築して，システムが正しく動作する必要があり，さらに，構成要素の記述に時間制約と確率を含める必要がある．本論文では，オープン性，リアルタイム性，不確定性を扱うために，以下の理論と技術を開発する：まず，構成要素の仕様記述言語として，確率時間ゲームオートマトンを開発して，その意味を確率時間ゲームで与える．次に，構成要素の正しさを保証するために，確率時間ゲーム検証手法を開発して，構成要素からシステムを構築する手法を開発して，最後に，段階的詳細化開発するために，確率時間交互模倣検証を開発する．

Design and Verification method based on Probabilistic Timed Game Theory for Real-Time Embedded Systems (Extended Abstraction)

S. Yamane
Graduate School of Natural Science,Kanazawa University
syamane @ is.t.kanazawa-u.ac.jp

Recent embedded systems are composed of many components, and each component is an open system, and it has both real-time properties and uncertainties. We specify each component using timing constraints and probabilities, and compose systems from all the components. In order to treat them, we develop the followings: First we develop probabilisitc timed game automata based on a probabilistic timed game as a computational model. Next we develop probabilistic timed game verification and probabilistic timed alternating refinement relations as the refinement theory of probabilistic timed game automata.

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]．

とりわけ，最近，L. de Alfaro や T.A. Henzinger などがリアクティブシステム向きのコンポーネントウェアのためのインターフェース理論 [11, 12, 14] を研究していることが注目される．しかし，彼らの理論はゲー

ム理論より弱い条件によって正当性を保証している.

本論文では, ゲーム理論, 時間オートマトン, 確率の概念を統合化することにより, L. de Alfaro や T.A. Henzinger などの理論を拡張して, 以下のような理論や技術を開発して, 組込みシステムの仕様記述と検証を実現する. なお, 組込みシステムの設計は, 構成要素単位に仕様記述して, それらの構成要素を合成してシステムを構築して, システムの正しさが保証されるものと考える.

1. まず, リアルタイム組み込みシステムの構成要素の仕様記述言語として, 確率時間ゲームオートマトンを開発して, その意味を確率時間ゲームで与える.
2. 次に, 構成要素の正しさを保証するために, 確率時間ゲーム検証手法を開発する.
3. 最後に, 段階的詳細化開発するために, 確率時間交互模倣検証を開発する.

以降の本論文の構成は以下のとおりである. 2節では, 確率時間ゲームオートマトンを定義する. 3節では, システムの構成手法を定義する. 4節では, 検証手法を定義する. 最後に, 5節では, まとめを述べる.

2 確率時間ゲームオートマトン

まず, 確率時間ゲームオートマトンを定義して, その意味を確率時間ゲーム理論で定義する. ただし, 本論文における確率とは離散確率である.

2.1 確率時間ゲームオートマトン

まず, 離散確率分布を定義する.

Definition 1 (離散確率分布)
有限集合 Q 上の離散確率分布の集合を $\mu(Q)$ と表す. 各 $p \in \mu(Q)$ は関数 $p : Q \to [0,1]$ である. ただし, $\sum_{q \in Q} p(q) = 1$ である.

■

組込みシステムは多数の相互作用する構成要素から構成される. 我々は, 構成要素を確率時間ゲームオートマトンで仕様記述する. 以下に, 確率時間ゲームオートマトンを定義する.

Definition 2 (クロック制約とその評価)
$\mathbf{R}$ 上のクロック変数の集合を χ とする. χ 上のクロック制約条件は $x \prec c$ または $x - y \prec c$ のブール結合である. ここで, c は整数, $x, y \in \chi$, $\prec$ は $<$ または $\leq$ である. χ 上のクロック制約条件の集合を $\Xi[\chi]$ とする.

クロック変数の集合 χ の評価は関数 $v : \chi \to \mathbf{R}$ である. χ のすべてのクロックに 0 を割り付ける評価を $\mathbf{0}_\chi$ と表記する. χ のすべての評価を $\mathcal{V}(\chi)$ と表記する. 評価 $v \in \mathcal{V}(\chi)$ に対して, すべての $x \in \chi$ に対して $(v + \Delta)(x) = v(x) + \Delta$ を $v + \Delta$ によって定義される評価を $v + \Delta$ と表記する. クロックの集合 $r \subseteq \chi$ に対して, $x \in r$ ならば x に 0 を割り付けてその他ならば $v(x)$ を割り付ける評価を $v[r := 0]$ と表記する. クロック制約条件 $\varphi \in \Xi[\chi]$ に対して, 評価 v の下で φ が真ならば, $v \models \varphi$ と表記する. $r \subseteq \chi$ に対して, すべての $x \in r$ を 0 で置換することによって φ から得られる条件に対し, $\varphi[r := 0]$ と表記する. 明らかに, $v[r := 0]$ *iff* $v \models \varphi[r := 0]$ である.

■

Definition 3 (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
$\mathbf{A} = (Q_A, {q_A}^{init}, \chi_A, {Acts_A}^I, {Acts_A}^O, Inv_A, \rho_A)$
によって定義される. ただし,

1. Q_A はロケーションの有限集合である.
2. ${q_A}^{init}$ は初期ロケーションである.
3. χ_A はクロック変数の有限集合である.
4. ${Acts_A}^I$ と ${Acts_A}^O$ はそれぞれ入力アクションの集合と出力アクションの集合であり, $Acts_A = {Acts_A}^I \cup {Acts_A}^O \cup \mathbf{R}$ とする. 入力アクションは環境のアクションであり, 確率時間ゲームオートマトンが制御できない. 一方, 出力アクションは確率時間ゲームオートマトンのアクションであり, 確率時間ゲームオートマトンが制御できる.
5. $Inv_A : Q_A \to \Xi[\chi_A]$ は各ロケーションに不変式を割り当てる関数である.
6. $\rho_A \subseteq Q_A \times \Xi[\chi_A] \times \{{Acts_A}^I \cup {Acts_A}^O\} \times 2^{\chi_A} \times \mu(Q_A)$ は遷移関係である. $(q_A, g_A, a_A, r_A, p_A) \in \rho_A$ に対して, $q_A \in Q_A$ は遷移元のロケーション, $g_A \in \Xi[\chi]$ は遷移が起きるときのクロック制約条件, $a_A \in {Acts_A}^I \cup {Acts_A}^O$ は遷移にラベル付けられたアクション, $r_A \subseteq 2^{\chi_A}$ 遷移によってリセットされるクロック集合, $p_A \in \mu(Q_A)$ はロケーション上の確率分布である.

■

Example 1 (確率時間ゲームオートマトンの例)
以下に, 確率時間ゲームオートマトンの例を図 1 に示す. 実践の矢印は確率時間ゲームオートマトンが制御できる動作を表して, 点線の矢印は確率時間ゲームオートマトンが制御できない動作を表す. ここでは, 到達ゲームを考えて, ゴールを確率 *0.8* 以上で順序対 $< {q_A}^{init}, x \geq 2 >$ に到達するかどうかとする. 到

達ゲームの問題とは，確率時間ゲームオートマトンが制御できない動作に関わらず，確率時間ゲームオートマトンが制御できる動作により，いつかはある確率以上でゴールの状態に到達する戦略を発見する問題である．図1では，初期ロケーションとその評価の順序対 $< q_A{}^{init}, x = 0 >$ から $x > 1$ のときは，ゴールに到達しない．$< q_A{}^{init}, x = 0 >$ から $x \leq 1$ のときは，ゴールに到達する．つまり，$< q_A{}^{init}, x \leq 1 >$ に対しては，勝つ戦略が存在する．

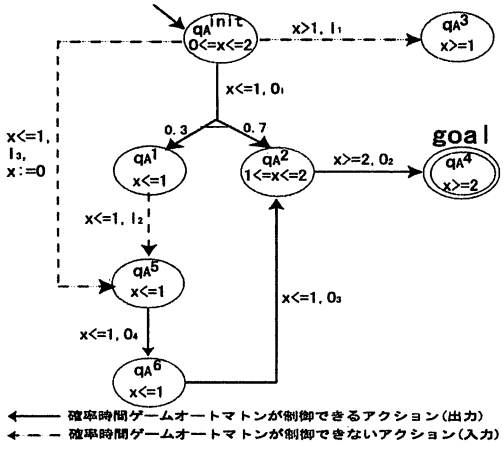

図 1: 確率時間ゲームオートマトンの例（その1）

■

2.2 確率時間ゲーム理論

確率時間ゲームは，**到達ゲーム**，**安全性ゲーム**，**無限ゲーム**に分類される．本論文では，到達ゲームについて考える．到達ゲームとは，確率時間ゲームオートマトン $\mathbf{A}$ と順序対 $(G \subseteq Q_A \times \mathbf{R}, \geq p)$ からなる到達条件 K に対して，戦略 f を発見する問題である．

確率時間ゲームオートマトン $\mathbf{A}$ の状態は，$< q_A, v_A > \in Q_A \times \mathbf{R}$ である．ここで，$v_A \in \mathcal{V}(\chi_A)$ とする．

確率時間ゲームオートマトン $\mathbf{A}$ の状態 $< q_A{}^{init}, \mathbf{0}_{\chi_A} >$ からの実行列の集合 $\mathbf{Runs}(< q_A{}^{init}, \mathbf{0}_{\chi_A} >, \mathbf{A})$ の要素 ρ としては以下のようなもの [15] がある：

$\rho =$

$< q_A{}^{init}, \mathbf{0}_{\chi_A} > \xrightarrow{\Delta_0}$

$< q_A{}^{init}, \mathbf{0}_{\chi_A} + \Delta_0 > \xrightarrow{g_0, i_0, r_0, p_0}$

$< q_{A1}, \mathbf{0}_{\chi_A} + \Delta_0[r_0 := 0] > \xrightarrow{\Delta_1}$

$< q_{A1}, \mathbf{0}_{\chi_A} + \Delta_0[r_0 := 0] + \Delta_1 > \xrightarrow{g_1, o_0, r_1, p_1}$

$< q_{A2}, \mathbf{0}_{\chi_A} + (\Delta_0[r_0 := 0] + \Delta_1)[r_1 := 0] > \xrightarrow{\Delta_2}$

$< q_{A2}, \mathbf{0}_{\chi_A} + (\Delta_0[r_0 := 0] + \Delta_1)[r_1 := 0] + \Delta_2 > \xrightarrow{g_2, i_1, r_2, p_2}$

.....

この ρ で到達する状態の確率は以下のように計算できる：

$$p_0(q_{A1}) \times p_1(q_{A2}) \times .....$$

到達条件を $(G \subseteq Q_A \times \mathbf{R}, \geq p)$ とする．実行列 ρ_i に対して，$< q_{Ak}, v_{Ak} > \in G$ を満たすすべての k について $\sum_k \{p_0(q_{A1})^k \times p_1(q_{A2})^k \times \ldots\} \geq p$ ならば，$\mathbf{A}$ は**勝つ**と呼ぶ．$< q_A{}^{init}, \mathbf{0}_{\chi_A} >$ からの $\mathbf{A}$ の勝つ実行列の集合を $\mathbf{WinRuns}(< q_A{}^{init}, \mathbf{0}_{\chi_A} >, \mathbf{A})$ とする．

以下では，確率時間ゲームの戦略と結果を定義する．

まず，O. Maler, A.Pnueli, J.Sifakis が開発した時間ゲームの戦略 [18] を拡張して，確率時間ゲームの戦略を定義する．

Definition 4 (確率時間ゲームの戦略)
$\mathbf{A} = (Q_A, q_A{}^{init}, \chi_A, {Acts_A}^O, {Acts_A}^I, Inv_A, \rho_A)$ を確率時間ゲームオートマトンとする．$\mathbf{A}$ 上の戦略 f は以下のような $\mathbf{Runs}((q_A{}^{init}, \mathbf{0}_{\chi_A}), \mathbf{A})$ から ${Acts_A}^O \cup \mathbf{R}$ への部分関数であり，以下のゲームの規則に従って決められる．

1. まず，ロケーション q_A において，以下の規則に従って，動作 α は決定される．ただし，動作 $\alpha \in \mathbf{R}$ は確率時間ゲームオートマトンのロケーションに留まった時間経過を意味して，動作 $\alpha \in Acts_A$ は確率時間ゲームオートマトンの状態遷移を意味する．
   (a) もし $\alpha_I, \alpha_O \in \mathbf{R}$ ならば，$\alpha = min\{\alpha_I, \alpha_O\}$ がプレイヤーとして選択される．つまり，$\alpha_I < \alpha_O$ ならば α_I であり，そうでなければ α_O である．ただし，I は入力プレイヤーであり，O は出力プレイヤーである．
   (b) もし $\alpha_I \in {Acts_A}^I$ かつ $\alpha_O \in \mathbf{R}$ ならば，$\alpha = \alpha_I$ がプレイヤーとして選択される．
   (c) もし $\alpha_O \in {Acts_A}^O$ かつ $\alpha_I \in \mathbf{R}$ ならば，$\alpha = \alpha_O$ がプレイヤーとして選択される．
   (d) もし $\alpha_I \in {Acts_A}^I$, $\alpha_O \in {Acts_A}^O$ ならば，環境の動作が優先されるので $\alpha = \alpha_I$ である．
   (e) もし $\alpha_I, \alpha_{I'} \in {Acts_A}^I$ ならば，非決定的に $\alpha = \alpha_I$ または $\alpha = \alpha_{I'}$ である．
   (f) もし $\alpha_O, \alpha_{O'} \in {Acts_A}^O$ ならば，非決定的に $\alpha = \alpha_O$ または $\alpha = \alpha_{O'}$ である．

2. 次に, α より, 一意に確率分布 p が決定されて, その中の一つのロケーション $q_{A'}$ が選択される.

■

また, Luca de Alfaro, Thomas A. Henzinger, Rupak Majumdar [19] は, 戦略 f の実行列をゲームの結果として定義するアイディアを開発した. ここで, 戦略 f により生成される確率時間ゲームオートマトン $\mathbf{A}$ の実行列は確率時間ゲームオートマトン $\mathbf{A}$ のすべての実行列の集合の部分集合である. Luca de Alfaro らのアイディア [19] を用いて, 我々は, 戦略 f の実行列を確率時間ゲームの結果として定義する.

Definition 5 (確率時間ゲームの結果)
$\mathbf{A}=(Q_A, {q_A}^{init}, \chi_A, {Acts_A}^I, {Acts_A}^O, Inv_A, \rho_A)$ を確率時間ゲームオートマトンとして, $\mathbf{A}$ 上の戦略 f とする. 状態 (q_A, v_A) から戦略 f の結果 $\mathbf{Outcome}(< q_A, v_A >, f)$ は $\mathbf{Runs}(< q_A, v_A >, \mathbf{A})$ の部分集合であり, 以下のように帰納的に定義される:

1. $< q_A, v_A > \in \mathbf{Outcome}(< q_A, v_A >, f)$,
2. もし $\rho \in \mathbf{Outcome}(< q_A, v_A >, f)$ ならば, $\rho' = \rho \xrightarrow{Acts_A} < q_{A'}, v_{A'} >$ である. このとき, $\rho' \in \mathbf{Runs}(< q_A, v_A >, \mathbf{A})$ であり, 以下のいずれかが成り立つ:
   (a) $Acts_A = (g, i, r, p)$,
   (b) $Acts_A = (g, o, r, p)$ かつ $o = f(\rho)$,
   (c) $Acts_A = \Delta \in \mathbf{R}$ かつ $\forall \leq \Delta' \leq \Delta$, $\exists < q_{A''}, v_{A''} > s.t.\ last(\rho) \xrightarrow{\Delta'} < q_{A''}, v_{A''} > \wedge f(\rho \xrightarrow{\Delta'} < q_{A''}, v_{A''} >) = \Delta'$.
   ただし, $last(\rho)$ は実行列 ρ の最後の状態である.
3. 無限な実行列 ρ に対して, ρ のすべての有限なプレフィックスが $\mathbf{Outcome}(< q_A, v_A >, f)$ の要素ならば, $\rho \in \mathbf{Outcome}(< q_A, v_A >, f)$ である.

■

入力アクションは制御できないアクションであり, ゲームを負けるように行動する. 場合によっては, 入力アクションによってゲームが終わるかもしれない. 出力アクションは制御できるアクションであり, ゲームを勝たせるように行動する [18]. ここで, 最大の実行列 ρ を定義する. 最大の実行列 ρ は以下のいずれかである:

1. NonZeno な無限な実行列 [20]
2. 以下のいずれかの有限な実行列である:
   (a) 実行列の最後の状態が G の要素である $(last(\rho) \in G)$
   (b) 実行列の最後の状態から可能な唯一の遷移は制御できない入力アクションである (もし $\rho \xrightarrow{(g,a,r,p)}$ ならば $a \in {Acts_A}^I$)

$\mathbf{Outcome}(< q_A, v_A >, f)$ の中のすべての最大の実行列が $\mathbf{WinRuns}(< q_A, v_A >, \mathbf{A})$ の要素ならば, 戦略 f は $< q_A, v_A >$ から勝つことができる. もし $\mathbf{A}$ の $< q_A, v_A >$ から勝つ戦略 f が存在するならば, $< q_A, v_A >$ は勝つことができる. $\mathbf{A}$ の勝つ状態の集合を W とする. $< q_A, v_A >$ から勝つ戦略の集合を $\mathbf{WinStrat}(< q_A, v_A >, \mathbf{A})$ とする.

3 リアルタイム組込みシステム設計

リアルタイム組込みシステムは多数の確率時間ゲームオートマトンで仕様記述された構成要素から、合成演算によって設計される. 2つの確率時間ゲームオートマトンからリアルタイム組込みシステムを合成演算によって設計する. この合成演算を繰り返し適用すると, 多数の確率時間ゲームオートマトンからリアルタイム組込みシステムを構成できる.

もし ${Acts_A}^O \cap {Acts_B}^O = \emptyset$ かつ $\chi_A \cap \chi_B = \emptyset$ ならば, 確率時間ゲームオートマトン $\mathbf{A}$ と $\mathbf{B}$ は構成可能である. それらの共通なアクションは $shared(\mathbf{A}, \mathbf{B}) = Acts_A \cap Acts_B$ である.

Definition 6 (合成演算)
2つの構成可能な確率時間ゲームオートマトン $\mathbf{A_1}$ と $\mathbf{A_2}$ に対して, 合成演算 $\mathbf{A_1} \otimes \mathbf{A_2}$ は以下の要素から構成される:

1. $Q_{\mathbf{A_1}\otimes\mathbf{A_2}} = Q_{\mathbf{A_1}} \times Q_{\mathbf{A_2}}$, かつ ${q_{\mathbf{A_1}\otimes\mathbf{A_2}}}^{init} = ({q_{A_1}}^{init}, {q_{A_2}}^{init})$.
2. $\chi_{\mathbf{A_1}\otimes\mathbf{A_2}} = \chi_{\mathbf{A_1}} \cup \chi_{\mathbf{A_2}}$.
3. ${Acts_{\mathbf{A_1}\otimes\mathbf{A_2}}}^I = {Acts_{\mathbf{A_1}}}^I \cup {Acts_{\mathbf{A_2}}}^I \setminus shared(\mathbf{A_1}, \mathbf{A_2})$, かつ ${Acts_{\mathbf{A_1}\otimes\mathbf{A_2}}}^O = {Acts_{\mathbf{A_1}}}^O \cup {Acts_{\mathbf{A_2}}}^O$.
4. $Inv_{\mathbf{A_1}\otimes\mathbf{A_2}}(q_{\mathbf{A_1}}, q_{\mathbf{A_2}}) = Inv_{\mathbf{A_1}}(q_{\mathbf{A_1}}) \wedge Inv_{\mathbf{A_2}}(q_{\mathbf{A_2}})$
5. $\rho_{\mathbf{A_1}\otimes\mathbf{A_2}}$ は遷移 $((q_{\mathbf{A_1}}, q_{\mathbf{A_2}}), g_{\mathbf{A_1}} \wedge g_{\mathbf{A_2}}, a, r_{\mathbf{A_1}} \wedge r_{\mathbf{A_2}}, p)$ の集合である. ただし, $p(q_{\mathbf{A_1}}', q_{\mathbf{A_2}}') = p_{\mathbf{A_1}}(q_{\mathbf{A_1}}') \times p_{\mathbf{A_2}}(q_{\mathbf{A_2}}')$, である.

■

本論文の確率時間ゲームオートマトンは, nonZeno は保証されているが, nonZeno は並列合成演算で閉じていない [16] ので, 各確率時間ゲームオートマトンが環境に勝っても並列合成が環境に勝つとは限らない. こ

こでは, 確率時間ゲームオートマトンが環境に勝つならば, そのオートマトンは望ましい性質（オープン性）を有すると考える. 確率時間ゲームオートマトンの並列演算閉包性 [17] を保証するためには, receptiveness を保証する必要がある. 確率時間ゲームオートマトンの receptiveness を保証するためには, 確率時間無限ゲームにおいて環境に勝つ必要がある. また, 確率時間無限ゲームにおいて環境に勝つ, 2つの確率時間ゲームオートマトンならば, 並列合成された確率時間ゲームオートマトンも環境に勝つ. すなわち, 確率時間ゲームオートマトンは並列合成によっても望ましい性質を保持する.

4 リアルタイム組込みシステム検証

本節では, 確率時間到達ゲーム検証と詳細化検証について説明する.

4.1 確率時間到達ゲーム検証

4.1.1 準備

$\mathbf{A}= (Q_A, {q_A}^{init}, \chi_A, {Acts_A}^I, {Acts_A}^O, Inv_A, \rho_A)$ を確率時間ゲームオートマトンとする. 著名な既存の時間ゲームの理論 [18, 19] においては, 到達ゲームに対して, 勝つ状態の計算は controllable predecessor 演算子を基礎としている. 本論文では, 既存の研究を拡張して, 確率時間ゲームの controllable predecessor 演算子を定義する.

確率時間オートマトンなどでは, ロケーションとクロック変数の評価値との順序対からなる状態ではなく, ロケーションとゾーンとの順序対からなる記号的状態によって検証を実現するのが効率的である [10, 21] ので, 本論文においても記号的状態によって検証を実現する.

Definition 7 (記号的状態)
記号的状態は順序対 $< q_A, \zeta_A >$ であり, $q_A \in Q_A$ と $\zeta_A \in Z_A$ である.

なお, Z_A は以下を満たす評価の集合である：

$$\bigwedge_{0\le i\ne j\le n} x_i - x_j \prec c_{ij}$$

ただし, $\chi = \{x_0, x_1, \ldots, x_n\}$ かつ $x_0 = 0$ であり, c_{ij} は整数, $\prec$ は $<$ または $\le$ である.

■

以下に, 記号的状態による検証のための演算を段階的に定義する.

まず, controllable discrete predecessor を定義する：

Definition 8 (controllable discrete predecessor)
遷移 $e_A = (q_A, g_A, c_A, r_A, p_A) \in \rho_A$ に関する $< q_A\prime, \zeta_A\prime >$ の *controllable discrete predecessor* $\mathbf{Pred}_c(e_A, < q_A\prime, \zeta_A\prime >)$ は以下のように定義される：

$\mathbf{Pred}_c(e_A, < q_A\prime, \zeta_A\prime >) =< q_A, g_A \wedge ([r_A := 0]{\zeta_U}^{q_A\prime}) >$

ただし,

$p_A(q_A\prime) > 0$,

$[r_A := 0]{\zeta_U}^{q_A\prime} = \{v | v[r_A := 0] \models {\zeta_U}^{q_A\prime}\}$.

ここで, ${\zeta_U}^{q_A\prime} = \bigvee\{\zeta | < q_A\prime, \zeta > \in U \subseteq Q_A \times Z_A\}$.

■

次に, controllable timed predecessors を定義する：

Definition 9 (controllable timed predecessors)
記号的状態 $< q_A\prime, \zeta_A\prime >$ の *controllable timed predecessors* $\mathbf{Pred_t}(< q_A\prime, \zeta_A\prime >)$ を定義する：

$\mathbf{Pred_t}(< q_A\prime, \zeta_A\prime >) =< q_A\prime, \swarrow \zeta_A\prime \cap inv_A(q_A\prime) >$

ただし,

$\swarrow \zeta_A\prime = \{v | \exists \Delta > 0 \ . \ (v + \Delta \models \zeta_A\prime \wedge \forall \Delta\prime \le \Delta . (v + \Delta\prime \models \zeta_A\prime))\}$

■

最後に, controllable predecessor 演算子を定義する：

Definition 10 (controllable predecessor)
controllable predecessor 演算子 π は以下のように定義できる：

$$\pi(< q_A\prime, \zeta_A\prime >) = \mathbf{Pred_t}(\mathbf{cPred}(< q_A\prime, \zeta_A\prime >))$$

■

4.1.2 アルゴリズム

確率時間ゲーム検証アルゴリズムは, ゴールの記号的状態から後向きの幅優先探索で初期状態へ到達する確率の合計が到達条件の確率を満たすかどうかをチェックするものである. 基本的には, ゴールの記号的状態から初期状態への後向きの幅優先探索であるが, ゴールの記号的状態と初期状態との間にループがありえる場合では, ループを回る毎に確率が小さくなるので, 計算は停止しない.

図2の例のゴールの記号的状態から初期状態への後向きの幅優先探索では, 以下の逆向きの無限な実行列が考えられる：

$$1. < {q_A}^5, x \ge 2 > \overset{x\ge 2, O_4, \{x\}, p_4}{\longleftarrow} < {q_A}^3, 1 \le x \le 2 > \overset{x\ge 1, O_3, \{\}, p_3}{\longleftarrow} < {q_A}^2, x \ge 1 > \overset{x\ge 1, O_2, \{x\}, p_2}{\longleftarrow}$$

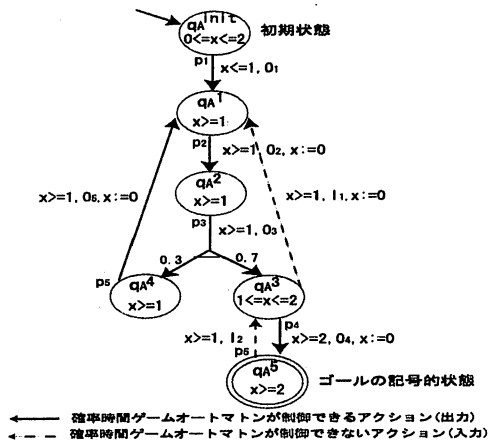

図 2: 確率時間ゲームオートマトンの例（その２）

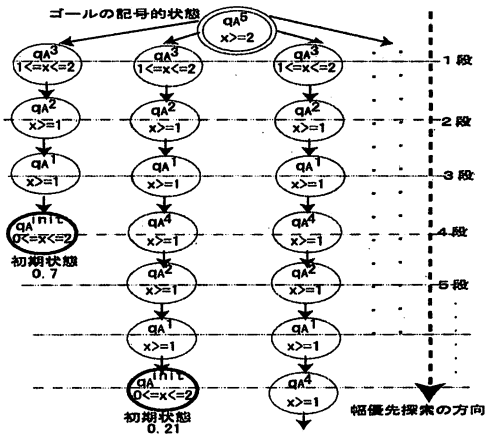

図 3: 確率時間ゲームオートマトンの逆向き幅優先探索

$< q_A{}^1, x \geq 1 > \overset{x\leq 1, O_1, \{\}, p_1}{\longleftarrow} < q_A{}^{init}, 0 \leq x \leq 2 >$

このとき, 確率は, $p_4(q_A{}^5) \times p_3(q_A{}^3) \times p_2(q_A{}^2) \times p_1(q_A{}^{init}) = 0.7$ である.

2. $< q_A{}^5, x \geq 2 > \overset{x\geq 2, O_4, \{x\}, p_4}{\longleftarrow} < q_A{}^3, 1 \leq x \leq 2 >$

$\overset{x\geq 1, O_3, \{\}, p_3}{\longleftarrow} < q_A{}^2, x \geq 1 > \overset{x\geq 1, O_2, \{x\}, p_2}{\longleftarrow}$

$< q_A{}^1, x \geq 1 > \overset{x\leq 1, O_1, \{\}, p_5}{\longleftarrow} < q_A{}^4, x \geq 1 >$

$\overset{x\leq 1, O_5, \{x\}, p_3}{\longleftarrow} < q_A{}^2, x \geq 1 > \overset{x\geq 1, O_2, \{x\}, p_2}{\longleftarrow}$

$< q_A{}^1, x \geq 1 > \overset{x\leq 1, O_1, \{\}, p_1}{\longleftarrow} < q_A{}^{init}, 0 \leq x \leq 2 >$

このとき, 確率は, $p_4(q_A{}^5) \times p_3(q_A{}^3) \times p_2(q_A{}^2) \times p_5(q_A{}^1) \times p_3(q_A{}^4) \times p_2(q_A{}^2) \times p_1(q_A{}^{init}) = 0.7 \times 0.3 = 0.21$ である.

.....

.....

この図２の例では, $q_A{}^1$, $q_A{}^2$, $q_A{}^4$ からなるループにより, 無限な実行列が存在して, それに対応する確率は, 0.7, 0.21, 0.063,... となる. 一般的に, 確率時間ゲームオートマトンの到達可能解析では, 実行列が長くなれば確率が小さくなるので, 計算は停止しない. このために, 逆向き幅優先探索により, 図３のように, 初期状態に到達する確率を逐次的に合計する近似アルゴリズムを考える.

Definition 11 (確率時間ゲーム検証アルゴリズム)
確率時間ゲームオートマトンを $\mathbf{A}=(Q_A, q_A{}^{init}, \chi_A, Acts_A{}^I, Acts_A{}^O, Inv_A, \rho_A)$ として, 到達条件を $(G \subseteq Q_A \times \mathbf{R}, \geq p)$ とする. 初期状態 $< q_A, \mathbf{0}_\chi >$ からゴールの記号的状態 $G \subseteq Q_A \times \mathbf{R}$ へ確率 $\geq p$ で到達するならば, 戦略 f が存在して, $\mathbf{A}$ は確率時間ゲームに勝つと呼ぶ. 検証アルゴリズムとしては, ゴールの記号的状態の集合 $G \subseteq Q_A \times \mathbf{R}$ の要素から後向きの幅優先探索で初期状態 $< q_A, \mathbf{0}_\chi >$ へ確率の合計が $\geq p$ で到達するかどうかをチェックする.

図４にアルゴリズムを示す.

■

本検証アルゴリズムは, R. Alur, T. A. Henzinger, Pei-Hsin Ho のハイブリッドオートマトンの記号的モデル検査 [22] や我々の確率ハイブリッドオートマトンのモデル検査 [23, 24] と同様に, 不動点の存在が保証されないシステムの検証において, 逐次的に不動点の近似点を計算するものである. 本検証アルゴリズムは確率時間システムの無限な動作を対象としているが, 類似する確率時間オートマトンの検証は有限の動作を対象 [10, 21] としており, この点において本研究とは異なる. なお, E.A.Emerson と Lei がこのような記号的検証を最初に提案したことは特筆すべきである [25, 26].

4.2 詳細化検証

詳細化関係は抽象的な仕様と具体的な実装との間を形式化するものであり, 言語包含関係や模倣関係が知られている [27]. この関係では, 実装の出力動作が仕様によって許容される動作であることを保証する. しかし, 本論文の確率時間ゲームオートマトンの詳細化関係では, 実装は仕様よりも多くの入力を受け付けて,

確率時間ゲーム検証アルゴリズム

/* 初期化する */
$P_A := 1$;
$SumP_A := 0$;
$Reachstate := \emptyset$;
$S_A = \{< q_A, \zeta_A > | < q_A, \zeta_A > \in G \subseteq Q_A \times \mathbf{R}\}$;
/* 逆向き幅優先探索を行う（逆向き記号的探索）*/
while (true) **do**
  $S_A\prime = \{< q_A\prime, \zeta_A\prime > | \forall e_A, \forall < q_A, \zeta_A > に対して,$
    $< q_A\prime, \zeta_A\prime > := \mathbf{Pred}_c(e_A, < q_A, \zeta_A >).$
    ただし, $e_A = (q_A\prime, g_A, c_A, r_A, p_A) \in \rho_A\}$;
  $PROB_A = \{P_A | \forall P_A, \forall p_A, \forall q_A\prime$ に対して,
    $P_A := P_A \times p_A(q_A\prime).\}$;
  $S_A\prime\prime = \{< q_A\prime, \zeta_A\prime\prime > | \forall < q_A\prime, \zeta_A\prime >$ に対して,
    $< q_A\prime, \zeta_A\prime\prime > := \mathbf{Pred}_t(< q_A\prime, \zeta_A\prime >)\}$;
  $REACH = \{Reachstate | \forall < q_A\prime, \zeta_A\prime\prime >$ に対して,
    $Reachstate := Reachstate \cup \{< q_A\prime, \zeta_A\prime\prime >\}\}$;
  $(< {q_A}^{init}, \mathbf{0}_\chi > \subseteq Reachstate)$ を満たす
    すべての P_A を $SumP_A$ に加える ;
  If $SumP_A \geq p$ **then output win ; goto finish** ;
  /* 次の繰り返しのために記号的状態集合を更新する */
  $< q_A\prime, \zeta_A\prime\prime >$ を $< q_A, \zeta_A >$ として,
    新たに, S_A を定義する ;
end while
finish ;

図 4: 確率時間ゲーム検証アルゴリズム.

仕様よりも小さな出力である必要がある. なぜならば, 仕様が動作する環境ならば実装も動作する必要があるためである.

Definition 12 (確率時間交互模倣関係)

2つの構成可能な確率時間ゲームオートマトン

$\mathbf{A} = (Q_A, {q_A}^{init}, \chi_A, {Acts_A}^I, {Acts_A}^O, Inv_A, \rho_A)$ と

$\mathbf{B} = (Q_B, {q_B}^{init}, \chi_B, {Acts_B}^I, {Acts_B}^O, Inv_B, \rho_B)$

が与えられたとする. なお, $\mathbf{A}$ は実装を表して, $\mathbf{B}$ は仕様を表すとする. ここで, $S_A = \{< q_A, v_A > | q_A \in Q_A, v_A \in \mathcal{V}(\chi_A)\}$, $S_B = \{< q_B, v_B > | q_B \in Q_B, v_B \in \mathcal{V}(\chi_B)\}$ とする.

以下の条件を満たす二項関係 $R \subseteq S_A \times S_B$ を確率時間交互模倣関係と呼ぶ.

1. ${Acts_B}^I \subseteq {Acts_A}^I$ かつ ${Acts_A}^O \subseteq {Acts_B}^O$.
2. 確率時間模倣: *[15]*
   すべての $(< q_A, v_A >, < q_B, v_B >) \in R$ に対して
   (a) 時間模倣条件:
   すべての $\Delta \in \mathbf{R}$ に対して,
   $< q_A, v_A > \xrightarrow{\Delta} < q_A, v_A + \Delta >$ ならば,
   $< q_B, v_B > \xrightarrow{\Delta} < q_B, v_B + \Delta >$ であり,
   $(< q_A, v_A + \Delta >, < q_B, v_B + \Delta >) \in R$ である.
   (b) 確率模倣条件:
   すべての $p_A \in \mu(Q_A)$ に対して,
   $< q_A, v_A > \longrightarrow_\alpha^{p_A} < q_A\prime, v_A\prime >$ ならば,
   $< q_B, v_B > \longrightarrow_\alpha^{p_B} < q_B\prime, v_B\prime >$ であり,
   $p_A(< q_A, v_A >) \sqsubseteq_R p_B(< q_B, v_B >)$ である.
   ここで, $p_A(< q_A, v_A >) \sqsubseteq_R p_B(< q_B, v_B >)$ とは, 以下の条件を満たすような重み関数 $w : S_A \times S_B \to [0,1]$ が存在することである:
   i. $\forall s_A \in S_A$ に対して, $\Sigma_{s_B \in S_B} w(s_A, s_B) = p_A(s_A)$ である.
   ii. $\forall s_B \in S_B$ に対して, $\Sigma_{s_A \in S_A} w(s_A, s_B) = p_B(s_B)$ である.
   iii. $\forall (s_A, s_B) \in S_A \times S_B$ に対して, $w(s_A, s_B) > 0$ ならば $(s_A, s_B) \in R$ である.

   ただし, $(q_A, g_A, \alpha, r_A, p_A) \in \rho_A$, $v_A \models Inv_A(q_A) \wedge g_A$, $v_A\prime = v_A[r_A := 0]$, $v_A\prime \models Inv_A(q_A\prime)$ である.
   また, $p_B \in \mu(Q_B)$, $(q_B, g_B, \alpha, r_B, p_B) \in \rho_B$, $v_B \models Inv_B(q_B) \wedge g_B$, $v_B\prime = v_B[r_B := 0]$, $v_B\prime \models Inv_B(q_B\prime)$ である.

■

確率時間交互模倣関係のアルゴリズムは, 文献 [28] にあるので, 本論文では割愛する.

5 まとめ

本論文では, 以下の3つの成果を述べた:

1. まず, 仕様記述言語として, 確率時間ゲームオートマトンを開発した.
2. 次にに, 確率時間ゲームオートマトンの確率ゲーム検証理論と詳細化検証理論を開発した.

今後は, 本理論を実現するシステムを実装して現実的な問題を対象として実証する予定である.

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*, 2007 (印刷中).

[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] L. de Alfaro, T.A. Henzinger. Interface Theories for Component-Based Design. *LNCS 2211*, pp.148-165. Springer-Verlag, 2001.

[12] L. de Alfaro, T.A. Henzinger. Interface Automata. *FSE* ,pp. 109-120, ACM Press, 2001.

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

[14] L. de Alfaro, T.A. Henzinger, M. Stoelinga. Timed interfaces. *LNCS 2491*, pp. 108-122, 2002.

[15] S. Yamane Probabilistic Timed Simulation Verification and Its Application to Stepwise Refinement of Real-Time Systems. *LNCS 2896*, pp.276-290, 2003.

[16] M. Abadi, L. Lamport, An Old-Fashioned Recipe for Real Time. *ACM TOPLAS*, No. 16, Vol.5, pp.1543-1571, 1994.

[17] R. Alur, T. A. Henzinger. Modularity for timed and hybrid systems. *LNCS 1243*, Springer, 1997, pp. 74-88.

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

[19] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. *LNCS 2154*, pp. 536-550, 2001.

[20] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *LICS*, pp. 394-406, IEEE Computer Society Press, 1992.

[21] M. Kwiatkowska, G. Norman, J. Sproston and F. Wang. Symbolic Model Checking for Probabilistic Timed Automata. *LNCS 3253*,pp. 293-308, 2004.

[22] R. Alur, T. A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, No.22, pp.181-201, 1996.

[23] Y. Mutsuda, T. Kato, S. Yamane. Specification and Verification Techniques of Embedded Systems using Probabilistic Linear Hybrid Automata. *LNCS 3820*, pp.346-360, 2005.

[24] 加藤位明, 陸田陽介, 山根 智. 述語抽象化とその精錬による確率線形ハイブリッドオートマトンの到達可能解析手法. 電子情報通信学会研究報告 *CST2006*, pp.19-24, 2006.(論文誌投稿中)

[25] E.A. Emerson, Chin-Laung Lei. Efficient Model Checking in Fragments of the Propositional Mu-Calculus. *LICS*, pp.267-278, 1986.

[26] M. Vardi. *Personal communication*, 2006.

[27] Robin Milner. An Algebraic Definition of Simulation Between Programs. *IJCAI*, pp.481-489, 1971.

[28] 山根 智, 小寺 広志, 荒井 恒夫. 確率時間オートマトンの確率時間強模倣検証器の開発 (Extended Abstraction 電子情報通信学会研究報告 *CST2006*, 2006.(論文誌投稿中)