社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

時間抽象を行う洗練手法を用いた確率時間システムの到達可能性解析

伊藤　明彦†　　長岡　武志†　　岡野　浩三†　　楠本　真二†

† 大阪大学大学院情報科学研究科
〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{a-ito,t-nagaok,okano,kusumoto}@ist.osaka-u.ac.jp

あらまし　本稿では，確率時間オートマトンの CEGAR を用いた到達可能性解析を行う手法を提案する．確率時間システムでは，反例となるパスの組み合わせが各ロケーションにおける時間経過によって変動するため，単純に複数個のパスの探索では正しい反例を提示できない．本稿で提案する手法では，そのパスの組み合わせを決定するために，元モデルの等価変換を行うことで解決している．
キーワード　確率時間システム，モデル検査，CEGAR，シミュレーション

Reachability Analysis for Probabilistic Timed System based on Timed Abstraction Refinement Technique

Akihiko ITO†, Takeshi NAGAOKA†, Kozo OKANO†, and Shinji KUSUMOTO†

† Graduate School of Information Science and Technology, Osaka University
Yamada-oka 1–5, Suita City, Osaka, 565–0871 Japan
E-mail: †{a-ito,t-nagaok,okano,kusumoto}@ist.osaka-u.ac.jp

Abstract　This paper gives a reachability analysis technique for Probabilistic Timed Automaton (PTA), based on CEGAR loop. The proposed method uses a transformation technique in order to avoid influence caused by dynamic change of behaviour with elapsing time.
Key words　Timed Probabilistic System, Model Checking, CEGAR, Simulation

1. まえがき

本稿では，時間オートマトンに対してクロック変数を除去する時間抽象を行う抽象化洗練手法 [1] を，確率時間オートマトンの抽象化洗練手法として拡張した手法を提案する．確率時間オートマトンの時間抽象化を行うことで，確率モデルに変換し，確率モデルの反例を探索する手法 [2] に基づいて反例を探索することができる．

一般に，確率時間オートマトンでは性質に対する反例が複数個のパスを含む場合が存在する．このため，パスが複数個にわたる場合，反例となるパス群の同時実行可能性について考慮しなければならない．確率時間オートマトンでは非決定的に時間経過が行われるため，それらのパスが同時に実行されることは無いからである．この問題があるため，単純に既存の CEGAR の手法を適用できない．

以下 2. では，まずモデルとして利用される確率時間オートマトンについて述べる．また，本稿で利用する CEGAR ループ，そして時間抽象化を用いた洗練手法について簡潔に述べる．次の 3. では，本研究で提案する確率時間オートマトンの到達可能性解析手法の概要と，アルゴリズムについて説明する．4. では，提案する変換手法の一部について正当性の定理を提示し，5. でまとめる．

2. 準　　備

2.1 確率時間オートマトン

2.1.1 確率分布

有限集合 Q 上の（離散的な）確率分布は関数 $\mu : Q \to [0,1]$ によって割り当てられ，$\sum_{q \in Q} \mu(q) = 1$ である．また，$\mathsf{support}(\mu)$ を $\mu(q) > 0$ である $q \in Q$ の部分集合する．

2.1.2 クロック変数とゾーン

確率時間オートマトンモデルでは，非負数の実数の集合 $\mathbb{R}$ 上の値である，クロック変数を使用する．クロック変数の有限集合 $\mathcal{X}$ を 1 つ定めるする．関数 $\nu : \mathcal{X} \to \mathbb{R}$ は，クロック評価として参照される．すべてのクロック評価の集合は，$\mathbb{R}^{\mathcal{X}}$ を意味する．また，すべてのクロック変数の値が 0 であるようなクロック評価を ν_0 とする．

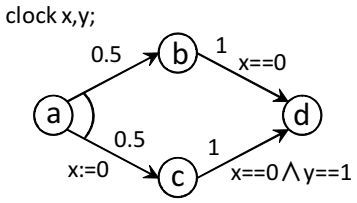

図 1　確率時間オートマトンの例

任意の $\nu \in \mathbb{R}^{\mathcal{X}}$，$t \in \mathbb{R}$ と $X \subseteqq \mathcal{X}$ に対し，t と，0 にリセットする X 中のクロックを評価するための $\nu[X := 0]$ および，すべてのクロック変数をインクリメントする ν においてクロック評価を意味する，$\nu + t$ の 2 つを使用する．

$\mathcal{X}$ のゾーンの集合を $Zones(\mathcal{X})$ と記述し，文法的に以下のように定義する．

$$\zeta ::= x \sim c \mid x - y \sim c \mid \zeta \wedge \zeta \mid \texttt{true}$$

ここで，$x, y \in \mathcal{X}$ であり，$\sim \in \{<, >, \leqslant, \geqslant\}$ である．ゾーン ζ は，ζ を満たす（$\nu \triangleright \zeta$ を意味する）クロック評価 ν の集合を示す．すなわち，$\nu(x)$ で true となる各クロック x の代替として，ζ を用いる．

2.1.3　確率時間システム

定義 2.1 (確率時間システムの構文)．確率時間オートマトン PTA の意味論を示す確率時間システム $\mathsf{TPS}_{\mathsf{PTA}}$ は，ラベル付けされたマルコフ決定過程であり，以下の 4 項組 $\mathsf{TPS}_{\mathsf{PTA}} = (S, s_0, TSteps, \mathcal{L})$ によって定義される．

- S：状態の有限集合
- s_0：初期状態
- $TSteps \subseteqq S \times \mathbb{R}_{\geqslant 0} \times \mathsf{Dist}(S)$：確率時間遷移関係 $(s, t, \mu) \in TSteps$ かつ $t > 0$ である場合，μ は点分布である．タプル (s, t, μ) の t は，$duration$ と呼ばれる．確率時間システムに対するアドバーサリとパスを導入する．
- $\mathcal{L} : S \to 2^{AP}$：状態に原子命題式を割り当てるラベル付け関数

2.1.4　確率時間オートマトン

定義 2.2 (確率時間オートマトンの構文)．確率時間オートマトン PTA は，7項目 $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ によって定義される．

- L：ロケーションの有限集合
- l_0：初期ロケーション
- Act：アクションの有限集合
- $\mathcal{X}$：クロック変数の有限集合
- $inv : L \to Zones(\mathcal{X})$：インバリアントを割り当てる関数
- $prob \subseteqq L \times Zones(\mathcal{X}) \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$：確率エッジ関係をを割り当てる有限集合
- $\mathcal{L} : L \to 2^{AP}$：ロケーションへ原子命題式を割り当てるラベル付け関数

図 1 に，確率時間オートマトンの例を示す．

定義 2.3 (確率時間オートマトンの遷移)．確率時間オートマトン $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ に対して，$(l, g, p) \in prob$ によって生成される確率時間オートマトンの遷移は，$p(X, l') > 0$ である 5 項組 (l, g, p, X, l') である．$\mathsf{edges}(l, g, p)$ を (l, g, p) によって生成される遷移の集合とする．また，$\mathsf{edges} = \{\mathsf{edges}(l, p, g) \mid (l, g, p) \in prob\}$ とする．

定義 2.4 (確率時間オートマトンの意味)．確率時間オートマトン $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ の意味は，時間確率システム $\mathsf{TPS}_{\mathsf{PTA}} = (S, s_0, TSteps, \mathcal{L}')$ によって以下のように定義される．

- $S \subseteqq L \times \mathbb{R}^{\mathcal{X}}$，$(l, \nu) \in S$ かつそのときに限り，$\nu \triangleright inv(l)$ である．
- $s_0 = (l_0, \nu_0)$
- $((l, \nu), t, \mu) \in TSteps$ の場合，かつそのときに限り，以下に従う．

a）　$time\ transitions$

$t \leqslant 0$ の時，$\mu = \mu_{(l, \nu+t)}$ である．また，すべての $0 \geqslant t' \geqslant t$ に対し，$\nu + t' \triangleright inv(l)$ である．

b）　$discrete\ transitions$

$t = 0$ であり，$\nu \triangleright g$，すべての $(X, l') \in \mathsf{support}(p)$ に対し，$\nu[X := 0] \triangleright inv(l')$ である $(l, g, p) \in prob$ が存在する場合，すべての $(l', \nu) \in S$ に対し，

$$\mu = \sum_{X \subseteqq \mathcal{X} \,\&\, \nu' = \nu[X := 0]} p(X, l')$$

- $\mathcal{L}'((l, \nu)) = L(l)$ $for\ all$ $(l, \nu) \in S$

2.1.5　$well\text{-}formed$ な確率時間オートマトン

確率時間オートマトンにおける確率的な遷移が常に有効である場合に限り，その確率時間オートマトンを $well\text{-}formed$ であるとする [3]．形式的には，確率時間オートマトン $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ が，常に以下の条件を満たせば well-formed である．

$$\forall (l, g, p) \in prob.\ \ \forall \nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}.\ (\nu \triangleright g)$$
$$\to (\forall (X', l') \in \mathsf{support}(p).\ \nu[X := 0] \triangleright inv(l')).$$

本研究で扱う確率時間オートマトンは，すべて $well\text{-}formed$ な確率時間オートマトンであるとする．

2.1.6　確率時間オートマトンの状態空間の表現

確率時間オートマトンの無限の状態空間の表現方法として，ゾーンを用いている．確率時間オートマトンが持つ状態空間は有限個のゾーンから構成されるゾーングラフで表現することができる．また，ゾーンが持つ制約の集合は，DBM(Difference Bound Matrix) とよばれる行列として表現することができる [4]．

2.1.7　確率時間オートマトンの同時実行可能性

確率時間オートマトンでは，ある性質を満たす確率で構成されるパスが複数個ある場合が存在する．この時，同じロケーションを出発するパスは，同じゾーンによって遷移している必要がある．確率時間オートマトンにおける時間

遷移は，非決定的な遷移であり，異なるゾーンで各パスが出発している場合，それらのパスは異なるアドバーサリで動作しているためである．

ここで，同時実行可能性に関する次の補題を与える．

補題 2.1 (2 つのパスの同時実行可能性)．確率時間オートマトン PTA 上の 2 つの任意のパス
$\omega^\alpha = \langle l_0^\alpha, \nu_0^\beta\rangle \xrightarrow{p_0^\alpha, t_0^\alpha} \langle l_1^\alpha, \nu_1^\alpha\rangle \xrightarrow{p_1^\alpha, t_1^\alpha} \ldots \xrightarrow{p_{n-1}^\alpha, t_{n-1}^\alpha} \langle l_n^\alpha, \nu_n^\alpha\rangle$ と $\omega^\beta = \langle l_0^\beta, \nu_0^\beta\rangle \xrightarrow{p_0^\beta, t_0^\beta} \langle l_1^\beta, \nu_1^\beta\rangle \xrightarrow{p_1^\beta, t_1^\beta} \ldots \xrightarrow{p_{m-1}^\beta, t_{m-1}^\beta} \langle l_m^\beta, \nu_m^\beta\rangle$ が以下の述語を満たすとき，ω^α と ω^β は同時実行可能である．

$$\mathsf{isCompatible}(\omega^\alpha, \omega^\beta) = \begin{cases} true & \text{if}(\ \exists i \in \mathbb{N} \text{ such that } \omega^\alpha(i+1) \neq \omega^\beta(i+1) \\ & \wedge \forall j \leqslant i \in \mathbb{N}.\ \omega^\alpha(j) = \omega^\beta(j)).\ t_i^\alpha = t_i^\beta \\ false & otherwise \end{cases}$$

true になる条件としては，パスにおける i 番目の状態で同じ状態のものの中で，そのロケーションにおける時間経過が同じものである．さらに，$i+1$ 番目の状態が異なるようなパスである．単純には，i 番目まで全く同一の遷移が行われ，$i+1$ 番目の遷移によって異なる遷移が行われるものである．さらに，この補題を元に，3 つ以上のパス集合に関する同時実行可能性についての補題を与える．

補題 2.2 (2 つ以上のパスの同時実行可能性)．確率時間オートマトン PTA 上のパス集合 Ω が以下の述語を満たすとき，Ω は同時実行可能である．

$$\mathsf{isCompatible}(\Omega) = \begin{cases} true & \text{if}(\ \exists i \in \mathbb{N} \text{ such that } \bigwedge_{\substack{\omega^\alpha, \omega^\beta \in \Omega \\ \wedge \omega^\alpha \neq \omega^\beta}} \omega^\alpha(i+1) \neq \omega^\beta(i+1) \\ & \wedge (\forall j \leqslant i \in \mathbb{N}.\ \bigwedge_{\substack{\omega^\alpha, \omega^\beta \in \Omega \\ \wedge \omega^\alpha \neq \omega^\beta}} \omega^\alpha(j) = \omega^\beta(j)). \\ & \bigwedge_{\substack{\omega^\alpha, \omega^\beta \in \Omega \\ \wedge \omega^\alpha \neq \omega^\beta}} t_i^\alpha = t_i^\beta \\ & \wedge (\forall \Omega' \in 2^\Omega \text{ such that } \Omega' \neq \Omega \wedge |\Omega'| \geqq 2). \\ & \mathsf{isCompatible}(\Omega') \\ false & otherwise \end{cases}$$

補題 2.1 では 2 つの場合を与えたが，補題 2.2 ではパスの集合 Ω に関するものである．$\mathsf{isCompatible}(\Omega)$ が true になる条件は，集合に含まれる任意の 2 つのパスが同時実行可能であり，かつその部分集合も同様である，というものである．

続いて，同時実行可能性を満たさないようなパスに関する例を与える．図 1 におけるロケーション a から d へ遷移するパスを求めた場合，ロケーション b を経由して d に遷移するパスは，$\omega^\alpha = \langle a, x = 0 \wedge y = 0\rangle \xrightarrow{0.5,0} \langle b, x = 0 \wedge y = 0\rangle \xrightarrow{1.0,0} \langle d, x = 0 \wedge y = 0\rangle$，となり，ロケーション c を経由して d に遷移するパスは，$\omega^\beta = \langle a, x = 0 \wedge y = 0\rangle \xrightarrow{0.5,1} \langle c, x = 0 \wedge y = 1\rangle \xrightarrow{1.0,0} \langle d, x = 0 \wedge y = 1\rangle$，となる．$\omega^\alpha$ で d へ遷移するためには，$b \to d$ の遷移に対するガード式 $x == 0$ が存在するため，a において時間遷移無しで動作する必要がある．一方，ω^β で d へ遷移するためには，$b \to d$ の遷移に対するガード式 $x == 0 \wedge y == 1$ が存在するため，a において 1 単位時間経過しておかなければ，d へ到達できない．

図 1 の確率時間オートマトンのようにロケーションにおける離散遷移の分岐において出発条件が異なる場合，各パスの時間的な振る舞いを考慮して反例となるパス群を考慮する必要がある．

2.2 時間オートマトンの抽象化洗練手法

著者らは文献 [1] で，Clarke らの Counter-example Guided Abstraction Refinement [5] の枠組みを利用した，時間オートマトンに関する抽象化洗練手法を提案している．この手法では，時間に関する制約をすべて除去する抽象化行い，抽象モデル上で偽反例が出力された場合，抽象状態の複製によって抽象モデルの洗練を行う．また，初期抽象化によって除去したクロック制約の復元を行わないため，生成する抽象モデルは常にクロック制約を持たない有限オートマトンである．

2.3 マルコフ決定過程

確率時間オートマトンに対し時間抽象化を行ったモデルは，マルコフ決定過程である．ここでは，マルコフ決定過程に関する概念を簡単に示す．

2.3.1 アドバーサリ

マルコフ決定過程では，アクションと呼ばれる非決定的な遷移が存在し，特定のアドバーサリによって，与えられた性質を満たす各パスがどの遷移を選択するかを決定することで，性質を満たす確率が与えられる．

2.3.2 モデル検査手法

マルコフ決定過程のモデル検査手法のうち，代表的な手法として ValueIteration [6] が挙げられる．ValueIteration では，入力されたパラメータに従って，到達可能性，安全性における最大確率値，最小確率値の 4 つの性質を求めることができる．さらに，各状態において，パラメータに適合するアクションが選択されるため，ValueIteration によって求められた確率を示すアドバーサリを求めることができる．

3. 提 案 手 法

本章では，本稿で提案する確率時間オートマトンの抽象化洗練手法を示す．提案する抽象化洗練手法では，著者らが文献 [1] で提案した時間オートマトンの抽象化洗練手法を利用している．さらに，前節で述べた，同時実行可能性を考慮した反例を生成するために，後方シミュレーションに加えて，時間遷移を明確化したロケーションを生成することで，同時実行可能性を明確化した確率時間オートマトンを生成する．

図 2 は本稿で提案する抽象化洗練の枠組みである．図で示すように，同時実行可能性を明確化するため，一般的な CEGAR の枠組みにモデル変換のフローを追加している．

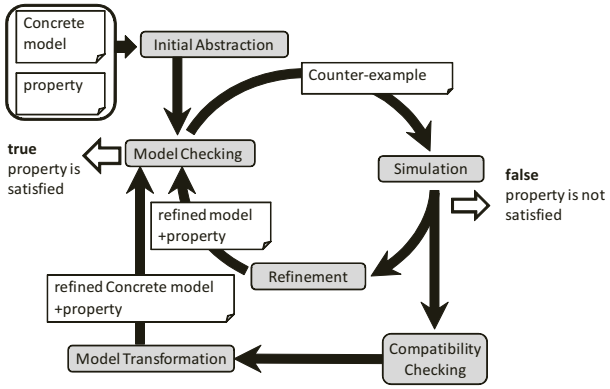

図 2　提 案 手 法

入力として，確率時間オートマトン PTA，性質を与える．検査対象とする性質は，$P_{<p}$ [$true\ U$ err] という PCTL 式に限定する．これは，err（エラー状態を示す）に到達する確率は p 以下である，という到達可能性解析に関するモデル検査を実行する式である．err には，PCTL の状態式に関する任意の記述を与える．

3.1　初期抽象化

初期抽象化では，文献 [1] と同様に，クロック変数に関する制約をすべて除去することで，over approximation を満たすように抽象化を行う．

3.2　モデル検査

モデル検査では，抽象モデルとして生成されたマルコフ決定過程に対して ValueIteration を適用し，到達可能性の最大確率を計算する．また，各状態において選択されたアクションを決定する．

そして，計算された結果が入力された性質の p を満たしている場合，確率時間オートマトンのモデル検査結果を $true$ として計算を終える．

3.3　シミュレーション

シミュレーションでは，k-最短路探索によって得られたパス (群) に対して，元の確率時間オートマトン上で実行可能かどうかを調べる．本手法では，文献 [1] で提案されているシュミレーションアルゴリズムに従って，DBM の演算によって到達可能か判定する．

3.4　抽象モデルの洗練

抽象モデル上で発生した偽反例を，元の確率時間オートマトン上で発生しないように，抽象モデルの洗練を行う．本手法では，文献 [1] で提案されている抽象状態の複製による手法を用いる．

3.5　同時実行可能性の検査

k-最短路探索によって得られたパス (群) が全て元の確率時間オートマトン上で実行可能であり，かつ確率の和が与えられた確率 p を超える場合，さらにそれらのパスの同時実行可能性を調べる必要がある．

そのために，反例として出力されたパス群に対し，同時実行可能性が偽となる分岐が存在しているかどうかを判定する．まず後方シミュレーションを行って，各パスを通過する各ロケーションにおいて目的ロケーションへ到達するために必要となる出発条件を求める．その後，補題 2.2 に従って，各分岐に対して同時実行可能かどうかを検査する．

Algorithm 1 BackwardSimulation(PTA, ω)

1: $\zeta := true$
2: **for** $i := n$ down to 2 **do**
3: 　{n はパス ω の長さ }
4: 　$l_{curr} := \omega[i], l_{prev} := \omega[i-1]$
5: 　$\zeta := \zeta \wedge inv(l_{curr})$
6: 　$e := (l_{prev}, g, p, X, l_{curr})$ {$\in$ edges}
7: 　$\zeta := \mathsf{free}(\zeta, e) \wedge g_e \wedge inv(l_{prev})$
8: 　$\mathsf{Z}_\omega[i] := \zeta$ {$\mathsf{Z}_\omega[i]$ に出発条件を追加 }
9: 　ER := ER $\cup (l, l_{curr}, e, \zeta)$ { 出発条件と遷移の関係を保持 }
10: 　down(ζ)
11: **end for**
12: **return** (Z, ER)

3.5.1　後方シミュレーション

後方シミュレーション（Algorithm1）では 1 つのパスに対し，そのパスが通過するすべてのロケーションに対する出発条件を求める．

離散遷移 $e = (l_{prev}, g_e, p_e, X_e, l)$ に対してロケーション l から l_{prev} への後方シミュレーションを考える．

まず，後方シミュレーションを行った際に，リセット式によって除去される制約式を求める関数 free を定義する．定義の詳細は，文献 [7] で示す．

free を用いてロケーション l_{prev} から l を示す遷移 e 上で後方シミュレーションを実行し，l_{prev} から l に出発条件となる遷移可能なゾーンを求めると，以下のようになる．

$$\zeta^{l_{prev}} = \mathsf{free}(inv(l), e) \wedge g_e \wedge inv(l_{prev})$$

ここで，$\zeta^{l_{prev}}$ を l_{prev} における出発条件として決定する．つづいて，$l_{prev} = l_0$ でなければ，続けて後方シミュレーションを行う．この時，$\zeta^{l_{prev}}$ に対し，時間経過の逆算である down 演算を行い，先ほどの式を用いてシミュレーションを継続していく．down 演算についての定義は，文献 [7] で示す．

以上の流れを用いて，パス ω に対する後方シミュレーションのアルゴリズムを Algorithm1 に示す．入力には，元の確率時間オートマトン PTA と，反例候補 $\omega = l_0 \xrightarrow{p_1} l_1 \xrightarrow{p_2} \cdots \xrightarrow{p_n} l_n$ を用いる．

このアルゴリズムでは，得られたパスを後方にシミュレーションを行い，$\zeta^{l_{prev}}$ に対しゾーンを求めていく．

パス ω 中の i 番目の各ロケーションにおいて求められた出発条件 ζ を，$\mathsf{Z}_\omega[i]$ に追加しておく．

上記の条件を満たし，$l_{prev} = l_0$ であれば，シミュレーションを終える．そうでなければ $\zeta^{l_{prev}}$ に対し，時間経過の逆算を行い，ゾーンを拡大する．その後，残りの経路に基づいて上述のゾーン演算を実行していく．

3.5.2　同時実行可能性の判定

後方シミュレーションによって各パス，各ロケーション

Algorithm 2 IsCompatible($\Omega, \mathsf{Z}, index$)

1: $\zeta := true, \mathsf{NSIM} := \emptyset, NLS := \emptyset$
2: **for all** $\omega' \in \Omega$ such that $|\omega'| > index$ **do**
3: $l_{next} := \omega'[index + 1]$
4: **if** $l_{next} \notin NLS$ **then**
5: $NLS := NLS \cup \{l_{next}\}$
6: $\Omega_{l_{next}} := \emptyset$
7: **end if**
8: $\Omega_{l_{next}} := \Omega_{l_{next}} \cup \{\omega'\}$
9: **end for**
10: **if** $|NLS| \geqslant 2$ **then**
11: **for all** $\omega' \in \Omega$ **do**
12: $\zeta := \zeta \wedge \mathsf{Z}_{\omega'}[index]$
13: **if** $\zeta = false$ **then**
14: $\mathsf{NSIM} := \mathsf{NSIM} \cup \{\omega'[index]\}$ { 同時実行できないロケーション集合 NSIM を作成 }
15: break
16: **end if**
17: **end for**
18: **for all** $l_{next} \in NLS$ **do**
19: $\mathsf{NSIM} := \mathsf{NSIM} \cup$ IsCompatible$(index + 1, \Omega_{l_{next}})$
20: **end for**
21: **end if**
22: **return** NSIM

の出発条件が求められたが，同時実行可能性の補題における isCompatible を満たしているかを調べる必要がある．そのために，IsCompatible（Algorithm2）を実行する．

出発条件の集合 Z と抽象モデル上のパスであるロケーションの系列 ω の集合 Ω が与えられ，Ω に含まれるパスの分岐が存在するかどうかを調べる．その分岐上で出発条件を比較し，同時に実行できないと判断された場合，同時実行できないロケーションの集合 NSIM に追加される．分岐において同じ遷移を示すパスは，その集合を用いて再帰的に IsCompatible を実行し，パス中のすべてのロケーションに対する同時実行可能性を検査する．

3.6 同時実行可能性を考慮したモデル変換

同時実行可能性の検査によって，同時実行可能性が偽であると判断された場合，偽であると判定された分岐において，時間遷移による振る舞いの違いを明確にするために，分岐先ロケーションを複製し，遷移関係を操作する．この複製されるロケーションを，本稿では複製ゾーンロケーションと呼ぶ．

具体的には次のように処理が実行される．Algorithm2 によって得られた集合 NSIM の要素であるロケーション l に対し，l において出発条件が異なるパスが存在する，つまり同時に実行不可能な分岐であると判断した場合，TransformPTA(Algorithm3) によって複製ゾーンロケーションの生成を実行する．ただし，TransformPTA に入力される確率時間オートマトン PTA は，時間に関する制約のラベルを除去した抽象モデル MDP において，各状態において取りうるアクションが（ValueIteration によって）決定されている PTA とする．

Algorithm 3 TransformPTA(PTA, Z, NSIM)

1: $\mathsf{DLS} := \emptyset, \mathsf{POWZ} := \emptyset$
2: **for all** $l \in$ NSIM **do**
3: $\mathsf{POWZ} := \mathsf{PowZ}(l, \mathsf{Z})$
4: $E :=$getTransitionSet(l) {E は l における遷移の集合 }
5: $\mathsf{DLS} :=$ MakeDuplicateZoneLocation(l, DLS)
6: MakeTransition$(l, a, \mathsf{POWZ}, \mathsf{DLS})$
7: RemoveTransition(E) { 元の遷移 E の除去 }
8: **end for**
9: **return** PTA

Algorithm 4 MakeDuplicateZoneLocation(l, DLS)

1: **for all** $l_{tmp} \in NEXT(l)$ **do**
2: $l_{dup} := newLocation()$ { ロケーションを生成 }
3: **for all** $(l_{prev}, l', e, \zeta_{tmp}) \in$ ER such that $l_{prev} = l$ and $l_{tmp} = l'$ **do**
4: $inv(l_{dup}) :=$ dupinv$(l, l_{tmp}, e, \zeta_{tmp})$
5: **end for**
6: $L := L \cup \{l_{dup}\}$
7: $\mathsf{DLS} := \mathsf{DLS} \cup \{l_{dup}\}$ { 複製ゾーンロケーションの集合を生成 }
8: **for all** $(l_{prev}, g, p, X, l') \in$ edges such that $l_{prev} = l_{tmp}$ and $g \in inv(l_{dup})$ **do**
9: edges := edges $\cup \{(l_{dup}, g, p, X, l')\}$
10: **end for**
11: **end for**
12: **return** DLS

まず，複製されるロケーションのインバリアントを示す．元モデル $\mathcal{M}$ のロケーション l に対応する，複製ゾーンロケーション $\tilde{l}$ における $inv(\tilde{l})$ を示す．

以下のように複製ゾーンロケーションのインバリアント $inv(\tilde{l})$ を求める関数 dupinv を定義する．

$$\mathsf{dupinv}(l, l_{prev}, e, \zeta) =$$
$$\nearrow ((\neg\zeta \wedge inv(l_{prev}) \wedge g_e)[X := 0]) \wedge inv(l)$$

入力となる l は複製元のロケーション，l_{prev} は複製元のロケーションへ 1 ステップで遷移できるロケーションである．また，e はその遷移を示すエッジである．ζ は，l を経由して目的の状態へ到達するために必要となる出発条件である．この処理を行うために，後方シミュレーションで得られた出発条件と，その出発条件がどのロケーションに対する出発条件なのかについての関係の情報を保持しておく必要がある．そのために，後方シミュレーションアルゴリズムの 19 行目において，出発条件と遷移の関係を示す集合 ER を定義しておき，dupinv に対する入力を与える（Algorithm4, 4 行目）．

続いて，複製ゾーンロケーション $\tilde{l}$ の遷移関係について述べる．まず，$\tilde{l}$ からの遷移は，複製元ロケーションからの遷移をそのまま保持する．ただし，ガードが $inv(\tilde{l})$ と矛盾しないような遷移のみとする．この処理は Algorithm4 の

Algorithm 5 PowZ(l, ER)

1: $\zeta_{pow} := true, Z_{oc} := \emptyset, POWZ[l]$
2: **for all** $(l_{curr}, l', e, \zeta) \in \mathsf{ER}$ such that $l_{curr} = l$ **do**
3: $Z_{oc} := Z_{oc} \cup \{\zeta\}$ {l における出発条件の集合を取得 }
4: **end for**
5: **for all** $Z \in \mathsf{POWERSET}(Z_{oc})$ **do**
6: {POWERSET によってべき集合を作成し，要素 Z を取りだす }
7: **for all** $\zeta^l \in Z_{oc}$ **do**
8: **if** $\zeta^l \in Z$ **then**
9: $\zeta_{pow} := \zeta_{pow} \wedge \zeta^l$
10: **else**
11: $\zeta_{pow} := \zeta_{pow} \wedge \neg\zeta^l$
12: **end if**
13: **end for**
14: **if** $\zeta_{pow} \neq \emptyset$ **then**
15: $\mathsf{POWZ}[l] := \mathsf{POWZ}[l] \cup \{\zeta_{pow}\}$
16: **end if**
17: $\zeta_{pow} := true$
18: **end for**
19: **return** POWZ[l]

Algorithm 6 MakeTransition$(l, \mathsf{POWZ}[l], \mathsf{DLS})$

1: **for all** $\zeta \in \mathsf{POWZ}[l]$ **do**
2: **for all** $l_{next} \in NEXT(l)$ **do**
3: {$NEXT$ は l の次状態の集合を表す }
4: **if** $l_{next} \in \mathsf{DLS}$ **then**
5: $l_{dup} := DupZoneMap(l_{next})$
6: **for all** $(l_{prev}, g, p, X, l') \in \mathsf{edges}$ such that $l_{prev} = l$ **do**
7: $\mathsf{edges} := \mathsf{edges} \cup \{(l_{prev}, g, p, X, l_{dup})\}$
8: **end for**
9: **else**
10: **for all** $(l_{prev}, g, p, X, l') \in \mathsf{edges}$ such that $l_{prev} = l$ and $l' = l_{next}$ **do**
11: $\mathsf{edges} := \mathsf{edges} \cup \{(l, g, p, X, l')\}$
12: **end for**
13: **end if**
14: **end for**
15: **end for**

6 行目に相当する．

複製ゾーンロケーションを追加した場合，そのロケーションへの遷移だけでなく，出発条件の組み合わせに応じた遷移を追加する．例えば，出発条件が ζ_1, ζ_2 の二つのパターンが発見された場合，$\zeta_1 \wedge \zeta_2$，$\neg\zeta_1 \wedge \zeta_2$，$\zeta_1 \wedge \neg\zeta_2$，$\neg\zeta_1 \wedge \neg\zeta_2$ の 4 つのパターンが存在すると考えられる．今回の提案手法では，そのパターンを明確に区別し，そのパターンに応じた遷移を追加することによって，出発条件の異なる，つまり同時実行不可能な遷移を厳密に分解する．ただし，空となるパターンは区別せず，遷移の追加を行わない．以上のパターン分類は，PowZ（Algorithm5）によって予め求めておく．

提案する遷移関係を再構成する Algorithm6 では，出発条件のパターンを用いて，複製ゾーンロケーションへの遷移，元の PTA からそのまま保持するロケーションへの遷移，の 2 つの遷移生成アルゴリズムを示している．それぞれ，6，13 行目によって遷移の追加処理が行われる．

4. モデルの等価性に関して

ここでは，前章において述べてきた変換後の確率時間オートマトンに対する定義を用いて，モデルの変換前後の等価性について次の定理を提示する．

定理 4.1. 入力の確率時間システム $\mathcal{M}$ と，*Algorithm3* による変換後の確率時間システム $\tilde{\mathcal{M}}$ は等価である．

定理 4.1 に関する詳細な証明は，文献 [7] で示す．

証明では，変換前後の各状態からの振る舞い（時間遷移，離散遷移）に関する等価性について議論している．

5. おわりに

本稿では，時間オートマトンに対する時間抽象化を用いた洗練手法を拡張し，一般的な確率時間オートマトンに対する抽象化洗練手法を提案した．

今後の課題としては，提案手法の完全な実装が挙げられる．一般的な DBM では `not` 演算に対応していないため，現段階では完全な実装を行うことができない．`not` 演算に関する既存研究を用いるなどして，解決していきたい．

謝辞 本研究の一部は科学研究費補助金基盤 C (21500036) と文部科学省「次世代 IT 基盤構築のための研究開発」(研究開発領域名：ソフトウェア構築状況の可視化技術の開発普及) の助成による．

文　献

[1] T. Nagaoka, K. Okano, and S. Kusumoto. An abstraction refinement technique for timed automata based on counterexample-guided abstraction refinement loop. *IEICE Transactions on Information and Systems*, Vol. E93-D, No. 5, p. to appear, May 2010.

[2] T. Han and J. Katoen. Counterexamples in probabilistic model checking. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 72–86, 2007.

[3] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Inf. and Comp.*, Vol. 205(7), pp. 1027–1077, 2007.

[4] A. David, J. Hakansson, K. G. Larsen, and P. Petterson. Model checking timed automata with priorities using dbm subtraction. *Lecture Notes in Computer Science*, Vol. 2402, pp. 128–142, 2006.

[5] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. *International Conference on Computer Aided Verification (CAV'00)*, pp. 154–169, 2000.

[6] D.P. Bertsekas. Dynamic programming and optimal control. *Athena Scientific*, 1995.

[7] `http://www-sdl.ist.osaka-u.ac.jp/~a-ito/proof/pta_cegar_full.pdf`.