形式手法を用いたネットワーク分散システムのシミュレーションと検証に関する研究

代表研究者　岡　野　浩　三　大阪大学　大学院　情報科学研究科　准教授
共同研究者　Behzad Bordbar　University of Birmingham
共同研究者　長　岡　武　志　大阪大学　大学院情報科学研究科

1 まえがき

近年，情報システムの高信頼設計にモデル検査を利用することが望まれている．モデル検査によって，設計されたシステムの振る舞いに関する不具合を早期に発見できることが期待されるためである．モデル検査は，システムを有限の状態遷移系として形式的に記述し，また検査を実行する性質を論理式で入力することで，状態遷移系の全状態を網羅的に探索する．それにより，設計されたシステムが仕様（性質）を満たしているかどうかを保証する手法である．しかし，大規模なシステムに対しては状態数爆発を起こすなど,スケーラビリティの低さが課題となっている．モデル検査のスケーラビリティの低さを改善する手法として，検査する性質ごとにモデルの状態数を適切に削減し，さらにはその抽象化の度合いを自動的にコントロールするモデル抽象化手法が注目されている[RP08]．とりわけ，モデル検査時における反例を用いて抽象度を自動コントロールするCEGAR (CounterExample Guided Abstraction Refinement) ループ[CF03]が大きな注目を浴びている．
実際に検査対象となるモデルとしては，ネットワークプロトコルなどのランダム性を持つモデルを検証する場合，確率モデルが用いられる．確率モデルとは，有限の状態遷移系に，状態遷移確率を付加したモデルである．確率モデルの最も代表的なものに，マルコフ決定過程が挙げられる．確率モデルを検査するための代表的なツールとして，確率的モデル検査ツールPRISM[KN04]があり，これを用いて実際のネットワークシステムの検証が広く行われている[F06,DK04,NO09,KN02,NI11]．
本研究では，大規模センサネットワークなどの分散システムのモデルをモジュールごと確率時間オートマトンでモデル化し，全体動作の検証をシミュレーションで，主要ノードごとの検証を確率時間オートマトンのCEGARループで行う手法を提案する．分散システムのモデルを確率時間オートマトンでモジュールごとモデル化することにより，分散システムの個別ノードごとの動作仕様の導出を容易にしている．

2 確率モデル

2.1 確率オートマトン
確率オートマトンは，有限状態数のオートマトンに対し，状態遷移を行う条件に，確率の値を付加しているものである．確率オートマトンは，3つのオートマトンに大別される[P02]．

・離散時間マルコフ連鎖（DTMC:Discrete Timed Markov Chain）
・連続時間マルコフ連鎖（CTMC:Continuous Timed Markov Chain）
・マルコフ決定過程（MDP:Markov Desicion Process）

ここではMDPおよび，確率オートマトンの時間拡張である確率時間オートマトンに注目する．

定義2.1 (マルコフ決定過程の構文)**.** マルコフ決定過程MDPは以下の*4* 項組MDP = (*S, s0, Steps, L*)によって定義される．
・S: 状態の有限集合
・$s_0 \in S$: 初期状態

・$Steps : S \times S \to [0, 1]$：遷移関数
・$L : S \to 2^{AP}$ ：ラベル付け関数
確率遷移行列$\mathbf{P}$ は，各$s \in S$ を$Dist(S)$ の空でない部分集合にマッピングする関数である．すなわち，すべての確率分布の集合である．直観的に，状態$s \in S$ に対し，$Steps(s)$ の要素は，「非決定的な選択」を表している．各々の非決定的な選択は，他の状態への遷移の可能性を与える確率分布である．

2.2 マルコフ決定過程のモデル検査手法

マルコフ決定過程MDPに対するモデル検査手法のうち，代表的な手法としてValueIteration[B95]が挙げられる．ValueIterationでは，パラメータに設定された値に従って，到達可能性，安全性における最大確率値，最小確率値を計算する．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（反例集合）を求めることができる．

2.3 確率時間オートマトン

2.3.1 確率時間システム

定義2.2 (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TPSPTA 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$PTA = (S, s_0, TSteps, L)$ によって定義される．
・S : 状態の有限集合
・s_0 : 初期状態
・$TSteps \subseteq S \times \mathbf{R} \times \mathrm{Dist}(S)$: 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．
・$L : S \to 2^{AP}$ ：状態に原子命題式を割り当てるラベル付け関数
タプル(s, t, μ) のt は，*duration* と呼ばれる．確率的なシステムと同様に，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．

2.3.2 確率時間オートマトン[KN07]

定義2.3 (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PTA $= (L, l_0, Act, X, inv, prob, L)$ 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
・X : クロック変数の有限集合
・$inv : L \to Zones(X)$: インバリアントを割り当てる関数
・$prob \subseteq L \times Zones(X) \times \mathrm{Dist}(2^X \times L)$: 確率エッジ関係を割り当てる有限集合
・$L : L \to 2^{AP}$ ：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

定義2.4(確率時間オートマトンの遷移). $(l, p, g) \in prob$ 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* 項目(l, g, p, X, l') である．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，edges = { edges(l, p, g) | $(l, g, p) \in prob$ } とする．

定義2.5（確率時間オートマトンの意味）．確率時間オートマトンPTA = $(L, l_0, Act, X, inv, prob, L)$の意味は，時間確率システムTPSPTA = $(S, s0, TSteps, L')$ によって以下のように定義される．
・$S \subseteq L \times \mathrm{R}^X$ ，$(l, v) \in S$ かつそのときに限り，$v \triangleleft inv(l)$ である．
・$s0 = (l0, v0)$
・$((l, v), t, \mu) \in TSteps$ の場合，かつそのときに限り，以下に従う．
time transitions
$t \geqq 0$ のとき，$\mu = \mu(l, v+t)$ である．
また，すべての$0 \leqq t' < t$ に対し，$v + t' \triangleleft inv(l)$ である．
discrete transitions
$t = 0$ であり，$v \triangleleft g$，すべての$(X, l') \in \mathrm{support}(p)$ に対し，

$v[X := 0] \triangleleft inv(l')$ である$(l, g, p) \in prob$ が存在する場合，すべての$(l', v) \in S$ に対し，

$$\mu = \sum_{X \subseteq \chi \wedge v' = v[X:=0]} p(X, l')$$

・$L((l, v)) = L(l)$ *for all* $(l, v) \in S$

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-formed* であるとする．確率時間オートマトンは，確率的な各遷移$(l, g, p) \in prob$ におけるガード式を置き換えることで，単純に*well-formed* な確率時間オートマトンに変換可能である．
本研究で扱う確率時間オートマトンは，すべて*well-formed* な確率時間オートマトンであるとする．

2.4 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ω_α とω_βが以下の性質を満たすとき，ω_α とω_β は同時実行可能である．

2つのパスにおけるi 番目の状態が同じ状態で，そのロケーションにおける時間経過が同じ，さらに，$i+1$ 番目の状態が異なる．

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 が以下の性質を満たすとき，Ω は同時実行可能である．

集合Ω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．

3 提案手法

提案する手法は次の通りである．モデルのシミュレーションについて文献[NI11]の手法にしたがう．

1．　分散システムを複数の確率時間オートマトンのネットワーク N(PTA)で記述する．この際，ネットワーク部，システムモジュールなどを個別のオートマトンで記述する．
2．　Digital clocks[KN06]の技法を使い，N(PTA)を確率オートマトンの１つであるマルコフ決定過程(MDP)に変換する．
3．　２で得られたシステムは PRISM 上でシミュレーション可能であるため，これを用いて全体レベルのシミュレーションを行う．この結果より，性能指標を得る．
4．　分散システムの定性的性質（デッドロック，安全性の保証）の検証は N(PTA)に対して次章で説明する確率時間オートマトンの CEGAR を用いて行う．

２．で得られたモデルは一般に状態爆発を起こし，モデル検査は容易でない．そのため，上記の 4 のように確率時間オートマトンの CEGAR を用いて状態爆発の回避をはかる．

図１に，N(PTA)で表記した分散システムの表記例を与える．

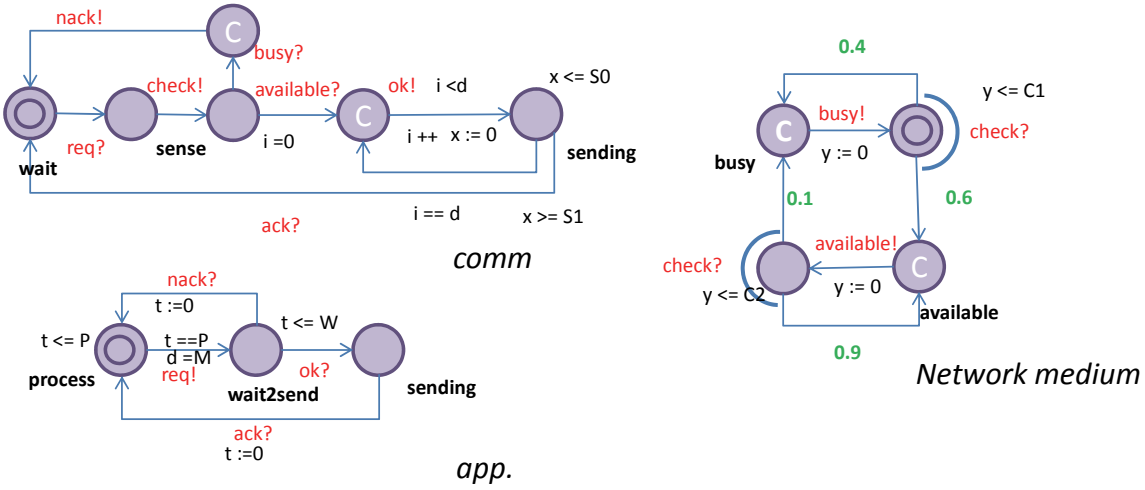

図 1:　N(PTA)で表記した分散システム

4 確率時間オートマトンCEGAR

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，著者らが提案した時間オートマトンの抽象化洗練手法[NO10]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．
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．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．

4.1 初期抽象化

初期抽象化では，クロック変数に関する制約をすべて除去することで，過大近似 を満たすように抽象化を行う[NO10]．
確率時間オートマトンのクロック変数に関する制約をすべて除去するため，この初期抽象化では，確率時間オートマトンをマルコフ決定過程に変換することになる．よって，後述のモデル検査では，マルコフ決定過程の既存の検査手法を採用し，マルコフ決定過程における到達可能性解析を実行する．

4.2 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を適用する．

ValueIterationでは，到達可能性の最大確率を計算することが可能であるが，入力された性質のp を上回る確率を持つ場合の具体的なパス（群）を示さない．よって，抽象モデル上で反例となるパス（群）を探索するためにPCTL 式に対する反例を探索する手法を用いる．

4.3 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，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[KNSS02].

4.4 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，抽象状態の複製による手法を用いる．

4.5 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（補題2.2）を調べる必要がある．確率時間オートマトンのある性質に対する反例中に同時実行可能性が存在する場合，その反例は正しくない.
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.

4.5.1 後方シミュレーション[KN07]

後方シミュレーションでは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.
つづいて，初期状態まで後方シミュレーションを行う．このとき，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.
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得る.

4.5.2 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，各ロケーションの出発条件が求められたが，同時実行可能性の補題を満たしているかを調べる必要がある．このアルゴリズムでは，後方シミュレーションによって求められた出発条件その条件を持つパスの集合を元に，同時実行性を検査する．最終的に，同時実行できないロケーションの集合が求められる.
具体的には，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，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.

4.6 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，新たなロケーションを生成し，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.
直観的には，あるロケーション上の時間遷移が取り得るすべてのパターンを，離散遷移上で明示的に行うようにする. この時間遷移が取れるパターンに応じたロケーションの複製を行って, 元モデルPTA に対し変換を施す．同時実行可能性となる反例パスが出力されてしまうのは，PTA 上の時間制約に関するラベルをすべて除去してしまうため, 非決定的な時間遷移について同じパス群に含まれてしまうためである．このようなケースに対し，時間遷移を離散遷移のように扱うことで抽象モデル上における異なる非決定的な遷移として扱うようになり，抽象モデル上でも同時に実行可能でない，と判断できる.

5 実験

5.1 実験の目的

提案する手法がどの程度スケーラビリティの低さを改善できているかを評価することが，この実験の目的となる．メモリ消費量の節約ができていれば，提案手法が有用であるということができる．以下では，実験環境と実験方法について述べた後，実験の対象となるモデル2つを紹介し，最後に実験結果として得られたデータを掲載する．

ただし，本研究における「複製ゾーンロケーションの生成」部に関しては未実装であるため，同時実行可能性が偽であるようなモデルには適用できない．以降に示す実験結果は，同時実行可能性はすべて真であったものである．

5.2 実験対象となる確率時間オートマトン

本研究で実験の対象としたモデルは，FireWire Root Contention Protocol[KN03]である（図2）．

「deadline までにリーダーが選出されない確率がp 未満である」とい内容で検証を行う．

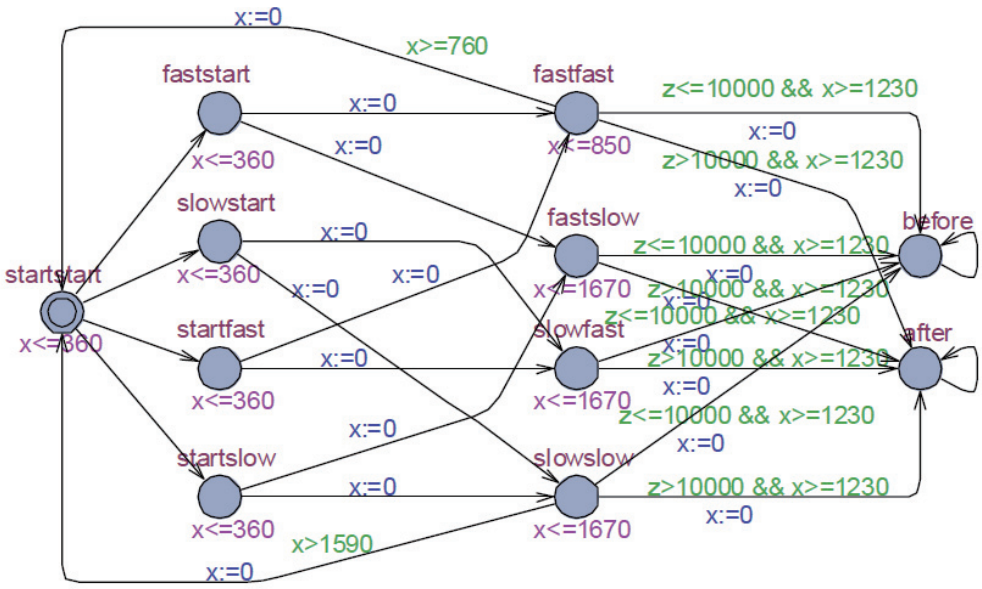

図2：FireWire Root Contention Protocol

5.3 実験方法

各モデルに対し，性質中のDeadline によってモデルサイズに大きく影響するため，Deadlineを増やしていった場合に，モデル検査の状態数，モデルをビルドする時間，検査を実行する時間の観点でDigital clocks[KN06]との比較を行う．入力となる性質に含まれる閾値p は，Digital clocks によってあらかじめ求められた最大確率の半分に設定し，入力とする．

5.4 実験結果

実験結果を表1,2 に示す．表1 では，状態数，実行時間，メモリ消費におけるDigital clocksとの比較を示している．なお，表1 の提案手法の結果はパスの最大探索数k を1000 としている．D=50，100 のときは，状態数，実行時間，メモリ消費において提案手法が優れているという結果になった．一方で，D=200 のときは，パス数が1000 の場合でも指定した確率値に達しなかったため，正しい結果が得ら

れずに終了した．表2 では，k の値を変えたときの提案手法の結果を示している．D=50，100 の場合は1000 程度で十分であると考えられるが，D=200 の場合は最大値の半分も達していないため，反例を構成するパス数が非常に多くなると予想できる．

表1: FireWire Abst モデルの実験結果

D	Digital Clocks				提案手法					Max reachability property
	States	Time(s)		Mem (MB)	Loops	States	Time(s)		Mem (MB)	
		Build	MC				Build	MC		
50	298165	7.5	8.43	9.1	27	44	0.86	0.79	4.93	0.21875
100	686165	26.5	56.3	24.4	142	165	0.91	12.74	7.01	0.02526855
200	1462165	97.4	376.3	54.9	563	602	1.86	2233	10.03	0.00037044

D：DeadLine，States：モデルの状態数，Build：モデルのエンコード時間，
MC：モデル検査にかかった時間，Loop：CEGAR ループ回数，
Max reachability probability：最大到達確率をそれぞれ表す．

表2: FireWire Abst モデルの実験結果（k 個の探索まで実行した場合）

D	k	提案手法						Max reachability property
		Loop	States	Time(s)		Mem(MB)	probability	
				Build	MC			
50	10	36	53	0.95	1.14	4.94	0.1445312	0.2187
	100	62	95	0.85	3.87	4.94	0.2187318	
	1000	62	95	1.00	39.30	6.40	0.2187499	
100	10	123	144	0.87	8.12	6.29	0.0046386	0.02526855
	100	157	185	1.13	18.57	7.00	0.0170898	
	1000	216	283	1.20	340.23	7.79	0.0252670	
200	10	473	504	1.84	500.70	9.43	0.0000045	0.00037044
	100	519	552	1.33	661.60	9.67	0.0000259	
	1000	563	602	1.87	2233.11	10.04	0.0000917	

k：k-最短路アルゴリズムによって探索するパスの数，

probability：k 個のパスによって計算された確率を表す

5.5 考察

5.5.1 メモリ消費量・状態数

表1 を見ると，Digital clocks[KN06] に比べて，最大で5 分の1 程度と大きくメモリ消費量が削減されていることがわかる．提案手法では，確率時間オートマトン中のクロック変数に関する制約をすべて除去したモデルであり，クロック変数制約をそのまま自然数として表したモデルよりも状態数が非常小さくなっているためであると考えられる．しかし，メモリ消費量は状態数程の差は見られない．これは，k-最短路によって複数個のパスを探すためのメモリがある程度必要であることが根拠として挙げられる．

5.5.2 実行時間

モデル検査時間に関しても，Digital clocks に比べ早くなっている．これは，状態数の大きな差によるものであると考えられる．一方で，D=200 でパスの数が10 の場合でも計算時間が多く掛っているが，ループ回数や状態数も初期抽象化の段階より10 倍以上に増加しており，これは最短パス近辺で洗練が実行され続けている，ということが言える．本実験において入力した性質は，リーダーを選出するまでにdeadline を超えてしまう，という内容であるため，反例として探索するパスは，deadline を超えるパスである．D=200 では，deadline を超えるパスは，目的ロケーションに到達するまでにモデルのある地点を何度かループする必要があり，最短路はループを行うような長いパスではないため，その点で洗練が多く行われていると考えられる．

5.5.3 パス数

deadline が50，100 の場合は1000 程度で十分であると考えられるが，200 の場合は最大値の半分も達していない．ただし，deadline が200 の場合，探索パス数の上限値まで正しい反例を求めても，入力されたp 値に届かず, 検査をきちんと実行できなかった. これは, deadlineの値を増加させることで，実行時間において述べたように，各パスはdeadline を超えてしまうような経路を選択する必要がある．よって，それぞれのパスの重みは小さくなり，有効な反例とするに必要なパス数が増えているためと考えられる．

6 あとがき

本研究では，確率時間オートマトンで記述された分散システムの全体仕様に対して解析を行うために，従来のDigital Clocksを用いた近似を行い，シミュレーションを行う手法と，時間オートマトンの時間抽象化とその洗練手法を拡張し, 確率時間オートマトンのCEGAR ループに基づくモデル検査手法を提案した．後者においては，反例となるパス群の同時実行可能性について言及し，同時に起こり得ないパスが抽象モデル上で発生しないようにするために，具体モデルの等価変換を行うことで解決した．加えて，完全な実装ではないが，同時実行可能性以外の面を実装したツールを用いて，確率時間オートマトンのCEGAR ループに対する評価実験を行った．
評価実験では，実用的な例題FireWire Root Contention Protocol を用いて，Digital clocks との比較を行った．パスの数が多くない場合だと，実行時間・メモリ消費量の点で優れていることがわかった．不具合の箇所を特定するのに有用な反例を導出することができる．

参考文献

[B95] D.P. Bertsekas. Dynamic programming and optimal control. *Athena Scientific*, 1995.

[CF03] E M. Clarke, A. Fehnker, Z. Han, J Ouaknine, O. Stursberg, and M. Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Int. Journal of Foundations of Computer Science*, 14(4):583–604, 2003.

[DK04] C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the ieee 1394 root contention protocol with kronos and prism. *International Journal on Software Tools for Technology Transfer (STTT)*, Vol. 5(2), pp. 221–236, 2004.

[F06] M. Fruth. Probabilistic model checking of contention resolution in the ieee 802.15.4 low-rate wireless personal area network protocol. *In Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'06)*, November 2006.

[KN04] M. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. *In Proc. 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pp. 322–323, 2004.

[KN03] M. Kwiatkowska, G. Norman, and J.Sproston. Probabilistic model checking of deadline properties in the ieee1394 firewire root contention protocol. *Formal Aspects of Computing*, Vol. 14(3), pp. 295–318, 2003.

[KN06] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, Vol. 29, pp. 33–78, 2006.

[KNSS02] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of realtime systems with discrete probability distributions. *Theoretial Computer Science*, Vol. 282, pp. 101–150, 2002.

[KN02] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the ieee 802.11 wireless local area network protocol. *In H. Hermanns and R. Segala (editors) Proc. PAPM/PROBMIV'02, volume 2399 of Lecture Notes in Computer Science*, Vol. 2399, pp.169–187, 2002.
[KN07] 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.
[NO09] T. Nagaoka, A. Ito, K. Okano, and S. Kusumoto. Qos evaluation for real-time distributed systems using the probabilistic model checker prism. *In Proceedings of International Workshop on Informatics*, pp. 60–66, 2009.
[NO10] 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, pp. 994-1005, 2010.
[NI11] Takeshi Nagaoka, Akihiko Ito, Kozo Okano, and Shinji Kusumoto, QoS Analysis of Real-time Distributed System Based on Hybrid Analysis of Probabilistic Model Checking, IEICE Transactions on Information and Systems, Vol.E94-D, No.5, pp.958-966, (2011)
[P02] D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
[RP08] P. Roy, D. Parker, G. Norman, and L. de Alfaro. Symbolic magnifying lens abstraction in markov decision processes. *In Proc. 5th International Conference on Quantitative Evaluation of Systems (QEST'08), pages 103-112, IEEE CS Press*, 2008.

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
Reachability Analysis of Probabilistic Real-Time Systems Based on CEGAR for Timed Automata	Proceedings of International Workshop on INformatics, IWIN 2010	2010 年 9 月