制約指向モデルで記述された対称性を持つ並行システムの形式的検証

梅津高朗[†] 山口弘純[†] 安本慶一[††]
中田明夫[†] 東野輝夫[†]

本論文では，ネットワーク会議などネットワークを介した複数ノード間の分散協調システム（並行システム）の仕様記述のための制約指向モデルと効率の良いデッドロックの検出法を提案する．提案する手法では，並行システム全体の動作仕様（要求仕様）を並行システムに含まれる各ノードの動作を他ノードとの関係を含まない形で個別に定義したカラーペトリネット（実行プロセス）群と，複数ノード間の相互関係やシステム全体として満たすべき制約を指定したカラーペトリネット（制約プロセス）群，および，その間の同期指定，で記述する．同じ動作をする実行プロセス群は複数のカラートークンを持つ１つのカラーペトリネットとして与えることにより記述する．このような仕様記述方法を用いた場合，制約条件すべてを満たすような実行系列が存在しない可能性がある．提案手法においては，システムの持つ対称性を利用して高速に可達性解析を行う方法を提案する．

Constraint-oriented Model for Specifying Distributed Cooperative Systems and Efficient Deadlock Detection Using Symmetries

Takaaki Umedu,[†] Hirozumi Yamaguchi,[†] Keiichi Yasumoto,[††]
Akio Nakata[†] and Teruo Higashino[†]

In this paper, we introduce a formal model for designing distributed cooperative systems (concurrent systems) with symmetries and propose an efficient deadlock detection method on this model. In our method, we describe a specification of a system by a set of coloured Petri-nets and synchronization among them. Each coloured Petri-net represents either participant's behavior or the constraint about the temporal ordering of multiple participants' behavior. For this specification, if given constraints are inconsistent with each other, the total system may have a deadlock state. Since such systems often include much symmetries, by merging equivalent states in a given specification, we can reduce the cost necessary for the reachability analysis. Here, we propose an efficient reachability analysis method using symmetries.

1. はじめに

近年のネットワークの高速化にともない，ネットワーク電子会議や遠隔授業，マルチメディアを利用した分散協調作業などネットワークを介した複数の計算機（ノード）からなる並行分散システムに対する需要が高まってきている．

これらの並行分散システムでは，それら複数のノードが互いに通信しながら全体として動作する．したがってこのようなシステムを設計する際には，ノード間の複雑な通信手順を直接記述する必要があり，煩雑で誤りも生じやすい．また，ネットワーク環境の変更に応じた異なる複数の制約の付加を求められるような場合も多いが，その場合には複雑な通信手順を含んだ仕様を直接変更する必要があり，多くの手間が必要となる．

そのような並行システムを簡潔に記述するため，従来より制約指向スタイルと呼ばれる仕様記述法が用いられている[1),2)]．制約指向では，各プロセスの動作の定義を行う部分と動作に対する制約（動作間の関係）を定義する部分を別々に記述することにより，機能ごとに別モジュールとするなど，システム全体を構造的

† 大阪大学大学院基礎工学研究科情報数理系専攻
Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University
†† 滋賀大学経済学部情報管理学科
Department of Information Processing and Management, Shiga University

に記述できる．また，仕様変更は制約の追加あるいは変更という形で容易に対応できる．しかし，システム全体の挙動の記述・変更が容易に行える反面，課された制約によっては，それらすべての制約条件を満たすような実行系列が存在しない（すなわちデッドロックが含まれる）場合がある．

一般にシステムの全状態数が有限であれば，その全状態を列挙した可達グラフを作成し，その上での到達可能性解析（可達性解析）を行うことでデッドロックの有無は判定できる．しかし，単純に可達性解析を行った場合，プロセス数が増えるに従い，状態数爆発の問題が生じ検証のコストが指数的に増大する可能性がある．文献 3) では元の可達グラフにおける対称（等価）な状態を表すノードを 1 つにまとめることでサイズを縮小した可達グラフを作成する方法が提案されている．文献 4)，5) では同様に状態間の双模倣等価性を用いてノード数を削減したシンボリック可達グラフを用いた方法が提案されている．文献 6)，7) では，不変式を利用して検証のコストを抑える手法が用いられている．しかし，これらの手法では検証者が直接等価関係や不変式を指定する必要がある点で共通しており，その発見は一般に容易ではない．

一方，並行システムにおいては，たとえばネットワーク会議における聴衆のように，同じ動作を行うプロセスが複数存在する（プロセスが対称性を持つ）場合が多い．このような場合には，ある動作をどのプロセスが実行したとしても，動作実行後のシステムの状態は互いに等価と見なすことができるため，状態間の等価関係を対称性を用いて機械的に導出でき，可達グラフのサイズを縮小できると考えられる．

本論文では，対称性を持つプロセスを含む並行分散システムの仕様記述法と効率の良いデッドロック検出法を提案する．提案する手法では，並行システムの要求仕様を，(1) プロセスの動作仕様群，および，(2) プロセス間で満たすべき制約群として，カラーペトリネットを用いて記述する．(1) のプロセスの仕様は各プロセスの動作内容とその実行順序を他のプロセスとは独立した形で記述する．互いに対称なプロセスの動作は，複数の色のトークンを持つ単一のカラーペトリネットとして記述することで対称性を明示的に記述できる．(2) のプロセス間の制約は，(1) のプロセスの動作仕様における動作が全体としてどのような順で実行されるべきかを指定する．

ただし，与えられた制約によっては一部のプロセスの動作が制限され，動作仕様では対称として記述されたプロセスが対称性を満足しなくなる可能性もある．そこで提案手法ではプロセスが対称性を保持するために制約が満たすべき妥当な十分条件を与えている．これにより，その対称性から機械的に可達状態間の等価関係を導き，可達グラフの状態数を削減できる．さらに，可達性を保持するカラーペトリネットの縮退規則を定義し，可達状態数そのものを減らす工夫もしている．これらの工夫により，多数のプロセスを含む並行システムの可達性解析が高速に行えると考えられる．

提案手法に基づく検証系を試作し，ネットワーク会議の例題に対し，その可達性解析に要する時間を測定し，提案手法の有効性を評価した．

以下，2 章において提案する仕様記述法について述べ，3 章で提案モデルに対する対称性を利用した可達性解析の手法を述べる．4 章でまとめと今後の課題を述べる．

2. 制約指向に基づくペトリネットモデル

2.1 ペトリネットとカラーペトリネット

ペトリネットはプレースとトランジションの 2 種類のノードからなる重み付き有向 2 部グラフである．各重み付き有向辺はアークとよばれ，それぞれには重み（$W(a)$ と記述する）が付与される．各プレース p には複数のトークンを配置でき，この配置をマーキングと呼ぶ．マーキング m におけるプレース p のトークン数は $m(p)$ で得られる．マーキングはペトリネットの状態を表す．トランジション t はその各入力プレース p に，p から t へのアーク a の重み $W(a)$ 以上の個数のトークンが存在する場合のみ発火可能であり，t が発火すると各プレース p から $W(a)$ 個のトークンが取り除かれ，同時に，t の各出力プレース p' に，t から p' へのアーク a' の重み $W(a')$ の個数のトークンが追加される．

提案方式で用いるカラーペトリネット（Coloured Petri Net，以下 CPN）は，高階ペトリネット（High Level Petri Net）[8] の 1 つであり，各トークンは色とよばれる型（整数型や実数型，文字列型など）を持ち，その色に属する値を保持する．また，アーク a の重み $W(a)$ は，トークンとバインディング変数の多重集合として定義される．バインディング変数とはそれと同じ色のトークンを代入できる変数であり，多重集合は同一要素を複数個含むことのできる集合である．また，マーキング m によるプレース p へのトークンの配置 $m(p)$ はトークンの多重集合である．以下では，要素 e_k $(k = 1 \ldots n)$ が w_k 個含まれる多重集合を $\sum_k w_k e_k$ で表す．さらに各トランジション t には，t の入力アークの重みに利用されるバインディング変数

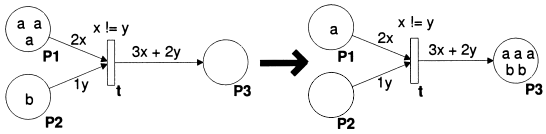


図 1 CPN におけるトランジションの発火
Fig. 1 Firing of transition in CPN.

を用いた論理式（ガード式）を定義する．

トランジション t は (i) t とアーク a で接続されている各入力プレース p に $W(a)$ の各バインディング変数に代入できるトークンが存在し，(ii) その代入が t のガード式を満たす場合にのみ発火できる．その結果，t の入力プレース p から $W(a)$ に代入されたトークンが取り去られ，t とアーク a' で接続されている各出力プレース p' に $W(a')$ の変数に代入されたトークンが追加される．発火の際には各バインディング変数への代入が決定される必要があるため，$W(a')$ で用いられているバインディング変数は t のいずれかの入力アークの重みで用いられている変数群の一部である必要がある．以下では，各 $W(a)$ のバインディング変数へのトークンの代入をバインディングと呼ぶ．

図 1 に発火の例を示す．a，b をある色の定数，x，y をそれぞれ a，b と同色のバインディング変数とする．バインディング $[x = a, y = b]$ は t のガード式 "$x! = y$" ($x \neq y$) を満足するため，t はこのバインディングで発火できる．その結果，$P1$ および $P2$ から，それぞれトークンの多重集合 $2a$ および b が取り除かれ，トークンの多重集合 $3a+2b$ が $P3$ に追加される．

2.2 制約指向に基づく CPN による仕様記述法

提案手法では，制約指向に基づき，システムの要求仕様を，(1) 各プロセスごとの動作内容とその実行順序を記述したカラーペトリネット（以下，動作ネットと呼ぶ）の組，(2) それらのプロセス間での動作の実行順序の制約や実行条件などを指定するカラーペトリネット（以下，制約ネットと呼ぶ）の組および，(3) 動作ネットと制約ネットのトランジション間の同期指定の組，で記述する．要求仕様記述は以下の規則に従うとする．

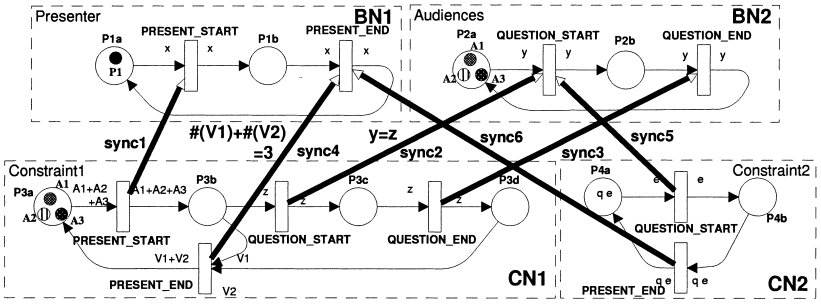
- システムが $1, 2, \ldots n$ の n 個のプロセスからなる場合，$\{1 \ldots n\}$ からなる列挙型の色 "process" を定義し，プロセスの動作を k 個 ($k \leq n$) の動作ネット BN_i ($1 \leq i \leq k$) で記述する．BN_i はそれぞれ色 "process" のトークンのみからなる初期マーキングを持つ．BN_i の初期マーキングに含まれるトークンの集合を o_i ($1 \leq i \leq k$) とすると，$\cup_{1 \leq i \leq k} o_i = \{1 \ldots n\}$ かつ $o_i \cap o_{i'} = \emptyset$ が成り立つとする．
- 任意の h 個の制約ネット CN_j ($1 \leq j \leq h$) を記述する．制約ネットで用いることができるトークンの色は，"process" および値を持たない色である "e" を含め，列挙型のみとする（上下限を定めた整数もこれに含まれる）．バインディング変数として，任意のトークンの多重集合を代入できる特別な変数（ヴァリアント型変数と呼ぶ）を利用できるとする．
- l 個の同期指定 $sync_x$ ($1 \leq x \leq l$) を記述する．各同期指定 $sync_x$ は BN_i のあるトランジション t_u と CN_j のあるトランジション t_v が同期して発火する必要があることを表し，t_u，t_v の各入力アークに用いられているバインディング変数に関するガード式が定義できるとする．また，ヴァリアント型変数 v に対し v に代入されたトークンの数を返す関数 $\#(v)$ が利用できる．

ここで，BN_i および CN_j の各トランジション t の入力アークの重みの総和と出力アークの重みの総和は等しいとする．この条件はトランジションの発火の前後でトークンの総量に変化がないことを保証する．したがって，システムが有界であることが保証され，可達性判定問題を決定可能とする．なお，ヴァリアント変数は，たとえば "3 個以上のプロセスがある状態にあれば動作 a が実行できる" など，ある動作を実行するプロセス数を特定したくない場合なども簡潔に記述できる（記述例は次節参照）．このような変数は一般のカラーペトリネットには含まれないが，提案する可達性解析ではシステムの有界性を利用し，ヴァリアント変数を等価変換により消去する方法も述べている．これらの詳細は 3 章で述べる．

2.3 仕 様 例

図 2 は 4 個のプロセス（1 人の発表者と 3 人の聴衆）からなる簡単なネットワーク会議制御システムの仕様を記述した例を示している．BN_1 と BN_2 はそれぞれ発表者の動作と聴衆の動作を表す動作ネットであり，CN_1 と CN_2 は制約ネットである．図 2 (b) では 6 つの同期指定 $sync_1, \ldots, sync_6$ が指定されており，これらは図 2 (a) で矢印として示されている．

BN_1 には発表者の 2 つの動作 "PRESENT_START"（プレゼンテーションの開始）と "PRESENT_END"（プレゼンテーションの終了）が定義されている．BN_2 においては聴衆の 2 つの動作 "QUESTION_START"（質問開始）と "QUESTION_END"（質問終了）が定義されている．x と y はそれぞれ色 "process" のバインディング変数である．これらの動作ネットに対して



(a) CPN の定義

	Behavior Net		Constraint Net		Guard
	Net	Transition	Net	Transition	
$sync_1$	BN_1	PRESENT_START	CN_1	PRESENT_START	true
$sync_2$	BN_2	QUESTION_START	CN_1	QUESTION_START	$y = z$
$sync_3$	BN_2	QUESTION_END	CN_1	QUESTION_END	true
$sync_4$	BN_1	PRESENT_END	CN_1	PRESENT_END	$\#(V1) + \#(V2) = 3$
$sync_5$	BN_2	QUESTION_START	CN_2	QUESTION_START	true
$sync_6$	BN_1	PRESENT_END	CN_2	PRESENT_END	true

(b) 同期指定

図 2 制約指向に基づくネットワーク会議制御システムの仕様例
Fig. 2 CPN specification of simple network meeting system in constraint oriented style.

2 つの制約ネットと同期指定が加えられている．その制約は以下のとおりである．

(1) 各聴衆はプレゼンテーションが開始されるまで質問をできず，たかだか 1 回しか質問できない．

(2) 最低 q 回の質問が行われなければプレゼンテーションを終了できない．

制約 (1) は CN_1 と同期指定 $sync_1, \ldots, sync_4$ によって実現される．$sync_1$ により BN_1 と CN_1 の 2 つのトランジション “PRESENT_START” が同期し，$sync_2$ により BN_2 と CN_1 の 2 つのトランジション “QUESTION_START” が同期する．CN_1 の構造から “QUESTION_START” はプレース P_{3b} にトークンがなければ実行できない．よって，“QUESTION_START” は “PRESENT_START” が発火した後でなければ発火できない．また，“PRESENT_END” を実行するためには $sync_4$ のガード式が真になる必要がある．このガード式は任意の個数の任意のトークンを割り当てることができるヴァリアント型の変数 $V1$ と $V2$ を含んでいる．ここで，プレース P_{3b} のトークンはまだ質問を行っていない聴衆を，プレース P_{3d} のトークンは “QUESTION_START” と “QUESTION_END” の発火によってすでに質問を済ませた聴衆をそれぞれ表している．P_{3b} と P_{3d} のすべてのトークンは “PRESENT_END” が発火する際にガード式 “$\#(V1) + \#(V2) = 3$” の条件により（$V1$ と $V2$ に割り当てられるため）取り去られる．すなわち，質問を済ませた聴衆は次に “PRESENT_END” が実行されるまでは質問することはできない．

制約 (2) は CN_2 と同期指定 $sync_5$，$sync_6$ によって定義されている．CN_2 の “QUESTION_START” が発火するごとにトークン “e” がプレース P_{4b} に生成される．P_{4b} に置かれたトークンはプレゼンテーション開始後に何回の質問がなされたかを示している．“PRESENT_END” を発火させ，プレゼンテーションを終了させるためには少なくとも q 個のトークン “e” が P_{4b} に存在する必要があるため q 回の質問がなされるまでプレゼンテーションは終了できない．

3. 可達性解析によるデッドロック判定

制約の矛盾によって生じうるデッドロック状態の有無を検証するため，可達グラフの一種である OS グラフ[9]を用いて可達性解析を行う．

3.1 制約指向に基づく複数 CPN による仕様から単一 CPN への変換

まず，複数の CPN とその間の同期指定からなる仕様を単一の CPN で記述されたシステム全体の仕様に等価変換する．その変換は次のようにして行う．

(1) 同期指定された制約ネットのトランジションと

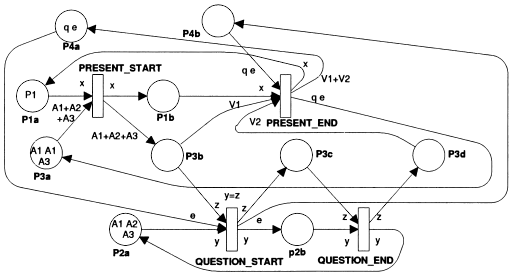


図 3 図 2 より得られるシステム全体の仕様
Fig. 3 Specification of total system derived from Fig. 2.

動作ネットのトランジションを 1 つに合成し，同期条件が指定されていればそれをガード式に追加する（**図 3**）．

(2) ヴァリアント型変数を等価変換により消去する．具体的には各ヴァリアント変数 v に対し以下を適用する．

(a) v が付加されたアークの入力プレースを p，出力トランジションを t とする．p が保持する可能性のあるトークンの色（列挙型）の要素の集合 $\{e_1, e_2, \ldots, e_q\}$ を決定する．これは p への入力アークの重みに用いられる変数の色から決定できる．

(b) トークン e_s をバインディングできるバインディング変数 x_s を定義し，バインディング変数の多重集合 $\sum_s w_s * x_s$ $(w_s \geq 0)$ を列挙する．ここで，w_s の上限は初期マーキングに含まれるトークン e_s の総数として決定できる．また，ガード式を満足しない多重集合は含めない．

(c) 列挙した各多重集合ごとに t を複製する（そのそれぞれを t' とする）．p から t' のアークの重みとしてその多重集合を与える．

このようにして図 3 のペトリネット群から変換されたものが**図 4** である．(2) の処理の例として，この図のトランジション “PRESENT_END11”, . . . , “PRESENT_END41” があげられる．これらのトランジションは，図 3 の BN_1 と CN_1 の “PRESENT_END” のヴァリアント型変数 $V1$ および $V2$ の消去により生じたものである．まず，$V1$ に対して展開を行う．この入力プレース P_{3b} が保持しうるトークンは $\{A1, A2, A3\}$ であり，したがって w_s は 0 または 1 である．さらに，“$\#(V1) + \#(V2) = 3$” なる条件を満たすバインディング変数 x_s の多重集合として $\{\emptyset, x_1, x_1 + x_2, x_1 + x_2 + x_3\}$ が列挙される．そして，それぞれに対してトランジション “PRESENT_END1”, . . . , “PRESENT_END4” が作

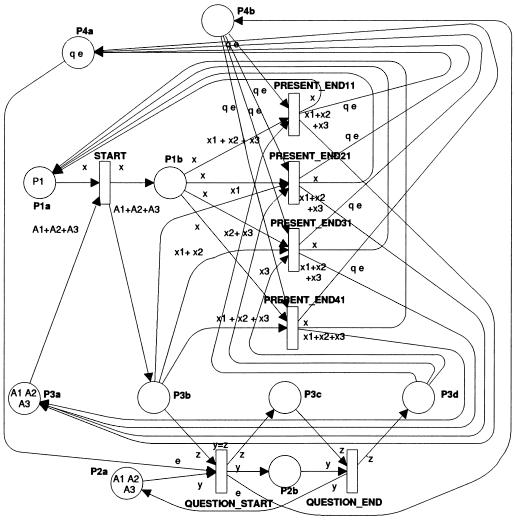


図 4 CPN に変換して得られた全体の仕様
Fig. 4 Transformed specification of total system.

成される．次に $V2$ に対しても同様の処理が行われる．たとえば，“PRESENT_END2” では，$V1$ は 1 つの変数 x_1 に置き換えられており，ガード式は $1+\#(V2) = 3$ となるため，ここでは $V2$ に対して列挙されるバインディング変数の多重集合は $\{x_2 + x_3\}$ のみとなり，“PRESENT_END21” が作成される．他のトランジションも同様に展開される．このようにしてヴァリアント型を含む並列有限カラーペトリネットは単一の有限カラーペトリネットに等価変換できる．

3.2 OS グラフを用いた検証

n 個のプロセスからなる並行システムを考える．このシステムは n 個のカラートークンを持ったカラーペトリネットとしてモデル化され，その仕様から合成したカラーペトリネットの検証を行う場合に，そのまま n 個のトークンとして扱うと可達グラフ（以下，Occurrence グラフ）の状態数は聴衆の人数が増えるに従い非常に大きくなる．

しかし，聴衆のうちのいずれかが質問を行う，といった仕様である場合には，質問をした人間が誰であっても，システム全体の動作は同じであると見なすことができる．この場合は，Occurrence グラフを作成する際に聴衆を区別する必要はなく，代わりに対称性を用いてサイズを小さくした OS グラフ（Occurrence Graph With Symmetries）[9] とよばれるグラフを作成することで全状態数を削減し可達判定を高速に行うことができる．OS グラフは，可達グラフのノードを与えられた対称性から導かれる等価関係を用いて同値類にしたものであり，互いに等価関係にある複数のノード（状態）を 1 ノードで表現できるため，多くの等価関係が

存在するようなペトリネットの検証に用いれば，グラフのサイズを大幅に縮小できる．

Occurrenceグラフは (M, A) の2字組で表される．$M = \{m_1, \ldots, m_n\}$ は初期マーキングから可達なすべてのマーキングの集合であり，$A \subseteq M \times M$ は遷移関係を表す．あるマーキング m_i からあるマーキング m_j へあるトランジションの1回の発火で到達可能なとき，またその場合に限り，$(m_i, m_j) \in A$ とする．Occurrenceグラフから，与えられた等価関係 E に基づいて以下のように作成したグラフが一定の条件を満たすならば，そのグラフはOSグラフと呼ばれる．

OSグラフ： 等価関係 E に基づきOccurrenceグラフから新たなグラフ $(\mathbf{M}, \mathbf{A})$ を作成する．ここで，$\mathbf{M} = \{M_1, \ldots, M_m\}$ は E によって等価とされるマーキングの集合族で，$\mathbf{A}$ は $\mathbf{M}$ 上での状態遷移を表す．以下では $[m]$ で m と等価なマーキングの集合を表すとする．各 M_i はそれぞれ M_i に含まれるマーキング $m'_i \in \mathbf{M}_i$ を用いて $[m'_i]$ と表せる．$\mathbf{A}$ は

$$\forall (m_i, m_j) \in A : ([m_i], [m_j]) \in \mathbf{A} \quad (1)$$

を満足するように定義する．ここで，関係

$$([m_i], [m_j]) \in \mathbf{A} \rightarrow \forall m'_i \in [m_i], \exists m'_j \in [m_j] : (m'_i, m'_j) \in A \quad (2)$$

が成立する場合，$(\mathbf{M}, \mathbf{A})$ がデッドロック状態を含むことと元のOccurrenceグラフがデッドロック状態を含むことが同値となり，$(\mathbf{M}, \mathbf{A})$ はOSグラフとなる．

略証： 条件(1)より，明らかに，グラフ (M, A) において m_j が m_i から可達ならばグラフ $(\mathbf{M}, \mathbf{A})$ において $[m_j]$ は $[m_i]$ から可達となる．次に条件(2)よりグラフ $(\mathbf{M}, \mathbf{A})$ において $[m_j]$ が $[m_i]$ から可達であるならば，すべてのマーキング $m'_i \in [m_i]$ に対して $m'_j \in [m_j]$ であり m'_i から可達であるような m'_j が存在する． □

3.3 対称性の十分条件

前述の(1)と(2)を満たすような等価関係を自動的に導出する一般的な方法は見つかっていない．しかし，提案手法では等価関係を機械的に導くために仕様が満足すべき妥当な十分条件を対称性を元に与えることができる．以下ではそれについて述べる．

今，仕様の初期マーキング，各重みおよびガード式に含まれるすべてのトークンの値の多重集合を $S_1, \ldots, S_q$ で表す．各 S_i に対し以下を満足するトークンの集合 S に対し，S は対称性を持つという．

- S_i または S_i から可能な限り集合 S を取り除いた多重集合 S' は S の要素をまったく含まない．

たとえば，図2の仕様において，トークンの集合 $S = \{A1, A2, A3\}$ を考えると，その初期マーキング，重み，ガード式は S の要素をすべて含まないか，もしくは，S のすべての要素をちょうど1個ずつ含んでいる．またすべての重みとガード式も同様にこの条件を満たしているため，$S = \{A1, A2, A3\}$ は対称性を持つトークンの集合であるといえる．

与えられた制約指向に基づくCPNの仕様に対してすべての初期マーキング，重み，ガード式を順に調べ，それらに含まれるトークンの集合を抽出する．そして，それらすべてに対して前述の条件を調べることで，機械的に対称性を持つトークンの集合を抽出できる．以後では，仕様から自動的に抽出された対称性を持つトークンの集合から有効な等価関係 E を生成する方法について述べる．

ここで，トークンの集合 $S = \{A1, A2, A3\}$ に対して，たとえば，$\Pi_1(S) = [A1, A2, A3]$ や $\Pi_2(S) = [A2, A1, A3]$ を用いて S のすべての要素の順列を表すものとする．ここでは，こういった Π_1 や Π_2 を置換と呼ぶ．また，以下ではペトリネットのマーキングを，各プレースに存在するトークンのベクトルの形で記述する（たとえば，図1のトランジションの発火の前後のマーキング m_1，m_2 はそれぞれ $(m_1(P_1), m_1(P_2), m_1(P_3)) = (3a, 1b, \emptyset)$，$(m_2(P_1), m_2(P_2), m_2(P_3)) = (a, \emptyset, 3a + 2b)$ で表す）．

対称性を持つトークンの集合 S が与えられたとき2つの可達なマーキング m_i と m'_i に関する等価関係 E を次のように定義する．

等価関係 E： S のある置換 Π が存在し，m_i に含まれる S に属するトークンを Π によって置き換えることで m'_i が得られる場合に2つのマーキング m_i，m'_i は等価である．

たとえば，図2において対称性を持つトークンの集合 $S = \{A1, A2, A3\}$ を用いて $m_i = (\emptyset, P1, A2 + A3, A1, \emptyset, A2 + A3, A1, \emptyset, 2e, e)$ と $m'_i = (\emptyset, P1, A1 + A3, A2, \emptyset, A1 + A3, A2, \emptyset, 2e, e)$ は，互いに，それぞれが含む $[A1, A2, A3]$ をその置換の1つ $[A2, A1, A3]$ で置き換えることで得られるため，この2つのマーキングは等価関係 E を満たす．よって，m_i と m'_i はOSグラフにおいては1つのノードにまとめられる．

この等価関係 E を用いて作成されるグラフはOSグラフである．直感的には，すべての対称性を持つトークンは同じように動き，等価な状態 m_i と m'_i を考えた場合に，あるトークンの m_i での移動とそのトークンと対称なトークンの m'_i での移動は同じ動作であると見なせることから明らかである．以下に証明の概略

を述べる．

略証： 初期マーキングから可達な2つのマーキング m_i と m'_i が対称性を持つトークンの集合 S に基づき m'_i は m_i が含む S の要素をその置換の1つ Π で置き換えることで得られるとする．また，m_i からバインディング B でトランジション T の発火によって可達なマーキングを m_j とする．ここで，S に含まれるトークンは対称なため，バインディング B が含むこれらのトークンも置換 Π によって新たなバインディング B' に置き換えることができる．m'_i は置換 Π によって得られるマーキングであるため，m'_i からのバインディング B' による T の発火による状態遷移が可能である．遷移先のマーキングを m'_j とすると，置換 p は m_j における T の入力（出力）プレースに存在する S に含まれるトークンを m'_j の同じ入力（出力）プレースに存在するトークンに置き換えるため，m'_j は m_j から置換 Π によって得られるものに等しい．結果として，すべてのマーキング $m'_i \in [m_i]$ に対して，マーキング m'_i から $m'_j \in [m_j]$ となる m'_j への状態遷移が存在する（$(m'_i, m'_j) \in A$）．よって，3.2節の関係(2)は満たされる． □

3.4 デッドロックフリーの性質を保存するCPNの縮退則

本論文においては，さらなる検証の高速化のためにデッドロックフリーの性質を保存する以下の縮退則を適用する．ただし，以下では $\bullet t$，$t\bullet$ でそれぞれトランジション t の入力プレースの集合，出力プレースの集合を表す．また，同様に $\bullet p$，$p\bullet$ を用いてプレース p の入力トランジションの集合，出力トランジションの集合を表す．

(1) $|\bullet t| = |t \bullet| = 1$ かつその入出力アークの重みが等しいトランジション t に対してその入出力プレースを結合させて t を削除できる．

(2) $|\bullet p| = |p \bullet| = 1$ かつその入出力アークの重みが等しいプレース p に対してその入出力トランジションを結合させて p を削除できる．

(3) ある2つのプレース p，p' からあるトランジション t へのアークが存在し，その重みがそれぞれ x，y（x，y はバインディング変数）であり，ガード式が $x = y$ と定義されている場合を考える．あるマーキング m において，

$$m(p) \subset m(p') \vee m(p) \supset m(p') \qquad (3)$$

が成立するならば，そのマーキングにおける t が発火可能な任意のバインディングでこのガード式 $x = y$ は必ず真になる．よって，初期マーキング m で条件(3)が満足され，また，あるマーキング m_i で条件(3)が成立すると仮定したときに m_i から $\bullet p \cup p \bullet \cup \bullet p' \cup p' \bullet$ に含まれるどのトランジション t' が発火したとしてもその遷移先 m_j においても同様に(3)が満足されることがいえるならばこのガード式は削除することができる．

(4) ある対称性を持つトークンの集合に対して，それに含まれるすべてのトークンの値に加え，それが代入される可能性のあるバインディング変数もガード式に用いられていない場合にはトークン群の区別はなくすことができる．すなわち，その集合に属するトークンすべてを代表するトークンの値を新たに定義し，元のトークンすべてをそれに置き換える．

縮退則(4)を満たすようなトークン集合 S を考えた場合，この S に属するトークンに対して仕様上にはトークンの値の条件が存在しないため，これらの値を無視しても可達性に影響を及ぼさない．よって，S に属さないトークンに関しては完全に等しく，S に属するトークンに関してはその個数が等しいような2つのマーキング m_i と m'_i は等価であるとする等価関係を用いてOSグラフを作成できる．しかし，等価関係を計算しながらOSグラフを作成するのではなく，縮退則(4)を用いて縮退したペトリネットに対してOccurrenceグラフを作成することでもまったく同型のグラフが得られ，その方が計算コストが少なくなる．

たとえば，図4において，2つのトランジション“QUESTION_START”と“QUESTION_END”は，縮退規則(2)を適用することで1つに結合できる．結合したものをトランジション t とすると，t の入力プレース P_{2a} は初期マーキングでトークンの集合{A1,A2,A3}を保持し，その唯一の入出力トランジション t の発火によってこのトークン集合は変化しない．また，t のもう1つの入力プレース P_{3b} には，A1，A2，A3以外のトークンが置かれることがない．よって，縮退規則(3)により，トランジション t のガード式 $y = z$ は削除できる．削除した結果，仕様中にトークン $A1$，$A2$，$A3$ が代入されうるバインディング変数を用いたガード式がいっさい存在しないため，これらの区別をする必要はなく，それぞれの値をある単一色のトークン A に置き換えて可達性判定を行うことができる．

3.5 検証結果

提案手法に基づく検証系を試作し，図2であげた例題に対して提案手法の実験をフリーウェアのカラーペトリネットのデザイン・シミュレーションツールであるDesign/CPN[10]を用いて行った．実験はPC-UNIX

表 1 デッドロック判定結果
Table 1 Experimental result.

	(i) Occurrence グラフ			(ii) OS グラフ			(iii) OS グラフ (CPN の縮退)		
Audience	#Nodes	#Arcs	Time	#Nodes	#Arcs	Time	#Nodes	#Arcs	Time
3	28	61	1 Sec.	11	14	1 Sec.	11	14	1 Sec.
4	66	177	1 Sec.	11	14	1 Sec.	11	14	1 Sec.
5	132	451	1 Sec.	11	14	1 Sec.	11	14	1 Sec.
6	234	1333	7 Sec.	11	14	3 Sec.	11	14	1 Sec.
7	380	6063	212 Sec.	11	14	586 Sec.	11	14	1 Sec.
15							11	14	3 Sec.
16							11	14	11 Sec.
17							11	14	17 Sec.

（CPU：PentiumIII 750 MHz，メモリ：1 GB，OS：Linux）上で行った．例題に対して聴衆の人数と質問の回数を変化させ，デッドロックの有無を判定した．その結果を**表 1** に示す．この表はそれぞれ，(i) 対称性を利用せず Occurrence グラフを作成した場合，(ii) 対称性を利用して OS グラフを作成した場合，および，(iii) 対称性を利用し，さらに縮退則によって値を無視できるトークン群を単一の値のトークン群に置き換えた場合についての得られた可達グラフのサイズと計算時間を示している．(ii) の場合，(i) に比べ可達グラフのサイズは非常に小さくなるが，計算時間の低減にはつながらなかった．これは対称性の判定の計算量がこの処理系においては非常に大きかったためと考えられる．(iii) の場合は，対称性の判定を行いノード数を抑えながらグラフを作成する (ii) と比較して特に問題規模が大きくなった場合に計算時間を大幅に短縮できた．

4. まとめと今後の課題

本論文では対称性を持つ並行システムの制約指向に基づく仕様記述法とデッドロックの検証法を提案した．提案手法では各ノードの動作をそれぞれ独立したプログラムとしてあるクラスのカラーペトリネットで表現し，それらのノード間の関係を制約という形でトランジションの同期を用いて表現する．また，その性質から自動的に対称性を検出し，OS グラフを作成することで高速に可達性判定を行える．ネットワーク会議制御システムの例題に対してこの方法を適用した結果，可達グラフのノード数を大幅に減らすことができ，検証に要する計算時間を短縮できた．

今後の課題は，本研究で考案したモデルを時間制約を記述できるように拡張し，そのモデル上での可達性判定を行う方法を考案することなどが考えられる．

参 考 文 献

1) Bolognesi, T.: Toward Constraint-Object Oriented Development, *IEEE Trans. Softw. Eng.*, Vol.26, No.7, pp.594–616 (2000).
2) Vissers, C.A., Scollo, G. and Sinderen, M.V.: Architecture and Specification Style in Formal Descriptions of Distributed Systems, *Proc. 8th Int. Symp. on Protocol Specification, Testing, and Verification* (*PSTV-VIII*), pp.189–204 (1988).
3) Jorgensen, J.B. and Kristensen, L.M.: Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Colored Petri Nets and Occurrence Graphs with Symmetries, *IEEE Trans. Parallel and Distributed Systems*, Vol.10, No.7, pp.714–722 (1999).
4) Hameurlain, N. and Sibertin-Blanc, C.: Finite Symbolic Reachability Graphs for High-Level Petri Nets, *Proc. 4th Asia-Pacific Software Engineering and Int. Computer Science Conference* (*APSEC '97/ICSC '97*), pp.150–159 (1997).
5) Cortadella, J.: Combining Structural and Symbolic Methods for the Verification of Concurrent Systems, *Proc. Int. Conf. on Application of Concurrency to System Design* (*CSD '98*), pp.152–157 (1998).
6) Miyamoto, T. and Kumagai, S.: Calculating Place Capacity for Petri Nets Using Unfoldings, *Proc. Int. Conf. on Application of Concurrency to System Design* (*CSD '98*), pp.143–151 (1998).
7) Nakamura, M., Kakuda, Y. and Kikuno, T.: Analyzing Non-determinism in Telecommunication Services Using P-invariant of Petri-Net Model, *Proc. INFOCOM '97*, pp.1253–1259 (1997).
8) Jensen, K. and Rozenberg, G. (Eds.): *High-level Petri Nets. Theory and Application*, Springer-Verlag (1991).
9) Jensen, K.: *Coloured Petri Nets*, EATCS Monographs in Theoretical Computer Science, Vol.2, Springer-Verlag (1997).

10) CPN group at the University of Aarhus, Denmark, Design/CPN, Ver. 4.0.4.
http://www.daimi.aau.dk/designCPN/

(平成 13 年 6 月 11 日受付)
(平成 13 年 9 月 12 日採録)

梅津　高朗（学生会員）

平成 13 年大阪大学大学院基礎工学研究科情報数理系専攻博士前期課程修了．同年同大学院博士後期課程進学．プロトコル合成法の応用や実時間システムの検証法等の研究に従事．

山口　弘純（正会員）

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 10 年同大学大学院博士後期課程修了．博士（工学）．同年オタワ大学客員研究員．平成 11 年大阪大学大学院基礎工学研究科助手．現在に至る．分散システムの設計法，マルチキャスト通信の研究に従事．

安本　慶一（正会員）

平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．現在同大学助教授．博士（工学）．平成 9 年モントリオール大学客員研究員．通信プロトコルや分散システムの形式仕様記述・実装法に関する研究に従事．

中田　明夫（正会員）

平成 4 年大阪大学基礎工学部情報工学科卒業．平成 9 年同大学大学院基礎工学研究科物理系専攻博士後期課程修了．博士（工学）．同年広島市立大学情報科学部助手．現在，大阪大学大学院基礎工学研究科助手．実時間システムや分散システムの仕様記述と検証法，プロセス代数，時相論理等の研究に従事．

東野　輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．同年同大学助手．平成 2，6 年モントリオール大学客員研究員．現在，大阪大学大学院基礎工学研究科教授，工学博士．分散システム，通信プロトコル等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．