EXPA: パータベーション解析に基づく通信プロトコルの検証法†

白鳥則郎†† 郷原純一†† 野口正一††

本論文では，状態遷移に注目したプロトコルの検証法について議論し，従来の検証法を包含し，検証能力の高い検証アルゴリズムを与える．現在，プロトコルが複雑化，大型化するにつれ対応する通信ソフトウェアも大規模かつ複雑となり，コストの増大が大きな問題となっており，生産性を向上させる設計法の確立が課題となっている．このような設計法において，仕様やソフトウェア誤りの検出・修正は，開発の初期の段階で行われるほど効果的である．そのためプロトコル仕様の論理的矛盾やあいまいさなどを検出する検証法が重要となる．本論文では，プロトコルは有限状態グラフで記述され，状態遷移を基本とする検証法について議論する．このなかで，とくに，パータベーション解析を包含し，検証能力の向上を達成した検証法 EXPA を与える．EXPA では，逆パータベーションの概念を導入することにより，従来のパータベーション解析と比べて，誤りの原因となった遷移の検出や誤りに至った過程を指摘できるなどの点で検証能力が向上している．

1. まえがき

計算機ネットワークなどの通信網は，高度な OA の実現や分散処理システムを構成するうえで必要不可欠なシステムとなっている．このようなシステムにおいて，通信プロトコルは，円滑な情報の交換を保証するために定められた規則の集合であり，ハードウェアとソフトウェアによって実現される．

プロトコルが複雑化，大型化するにつれ対応する通信ソフトウェアも大規模かつ複雑となりコストの増大が大きな問題となっている．そのため現在，とくに，開発・保存の生産性を向上させるプロトコルと通信ソフトウェアの設計法の確立が課題となっている．

このような設計法において，プロトコル仕様やソフトウェア誤りの検出・修正は，開発の初期の段階で行われるほど効果的である．したがって，プロトコル仕様の論理的矛盾やあいまいさなどを検出する検証法が重要となる．

これまでに提案されているプロトコル検証法は，（a）状態遷移を基本にする方法[1),2)]，（b）論理関係式の証明を与える方法[8)]，（c）前記の二つを併用する方法[9)]，に大別される．

前述のいずれの検証にも長所と短所が混在している．また，一般に，プロトコルの完全な正しさを証明することは困難である．そのため，現在，各検証法において，（1）制限条件の緩和，（2）検証能力の向上，（3）解析効率の向上などの方策について種々の観点から研究が進められている．

本論文では，プロトコルは有限状態グラフで記述され，状態遷移を基本とする前記の分類（a）の検証法について議論する．このなかで，とくに，パータベーション解析による検証法[2)]を包含し，検証能力の向上を達成した検証法 EXPA アルゴリズムを与える．

ここで，パータベーション解析による検証法は，CCITT X. 25 のパケットレベルの仕様など実際のプロトコルに適用され，その有効性が確認されている[2),3)]．

EXPA では，逆パータベーションの概念を導入することにより，従来のパータベーション解析と比べて，誤りの原因となった遷移の検出や誤りに至った過程を指摘できるなどの点で検証能力が向上している．なお，制限条件や生成される状態数については従来のパータベーション解析[2)]と同一である．

2. プロトコルの開発と検証法

プロトコル開発の生産性を向上させる手法の一つとして**図1**に示すように，検証法を積極的に活用した設計法[6)]（DAI-設計法と呼ぶ）が考えられる．ここで，D-表現は，設計者がプロトコル仕様を論理的に記述するための表現であり，理解の容易性と厳密性が具備条件となる．A-表現は，プロトコル検証用の表現であり，検証法に依存する．I-表現は，インプリメント用の表現であり，プログラミング言語などに対応している．

DAI-設計法では，D-表現として仕様記述言語 NESDEL[10)]，I-表現としてプロトコル向きプログラミ

† EXPA: Validation Method of a Communication Protocol Based on the Perturbation Analysis by NORIO SHIRATORI, JUNICHI GOHARA and SHOICHI NOGUCHI (Research Institute of Electrical Communication, Tohoku University).

†† 東北大学電気通信研究所

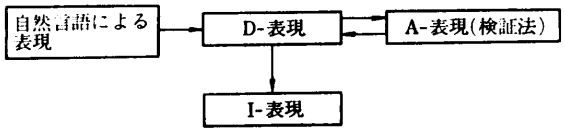

図 1 DAI-設計法によるプロトコルの開発
Fig. 1 Protocol development using DAI-design method.

ング言語 IDL[11]，A-表現に対応する検証法としては本論文で与える EXPA を想定している．検証法 EXPA と対応する A-表現については次章以降で詳述する．

図1において，はじめに設計者は自然言語で書かれたプロトコル仕様の素案をもとに厳密な D-表現を作成する．次に，論理的な誤りを検出するために A-表現に変換し検証する．誤りが検出されれば，対応する D-表現を修正し，再度 A-表現に変換し検証法を動作させる．誤りが検出されなくなるまでこの手順を繰り返す．最後に，誤りが検出されなくなった D-表現を I-表現に変換しソフトウェアを作成する．

以上のように，EXPA は DAI-設計法において，プロトコル開発の生産性を向上させるうえで重要な構成要素として位置づけられている．

なお，現在，DAI-設計法において，D-表現，A-表現，I-表現用の言語の設計が終了し，これらの言語間の変換アルゴリズムを開発中である．

3. プロトコルのモデル化と諸定義

プロトコルは，種々の有用性から階層的に構成されるのが一般的となっている．各階層のプロトコルにしたがった機能を実行する論理的な機能モジュールをプロセスと呼ぶ．たとえば，階層 I のプロトコルに対応するプロセスをプロセス I と呼び $P(I)$ で示す．また，$P(I)$ の上位階層と下位階層のプロセス群を一括して，それぞれ $H(I)$ と $L(I)$ で示す．とくに，**図2** に示すように，$L(I)$ と $L'(I)$ をチャネルとしてモデル化する．ここで，$L'(I)$ は $P(I)$ の通信相手である $P'(I)$ の下位階層に対応したプロセス群を示す．したがって，各プロセスは相手プロセスとチャネルを用いて通信すると考えることができる．

[定義 1] イベントと送受信動作

プロセス間で交換する情報の基本単位をイベントと呼び，正の整数で示す．また，イベントの送信と受信動作をそれぞれ負（−）と正（+）で表示する．ここで，正と負のついたイベントをアクションと呼ぶ．たとえば，−2 と +3 は，それぞれイベント 2 の送信とイベント 3 の受信動作を示している．イベントの具体例としては，パケットや制御用コマンドなどがある． (定義終)

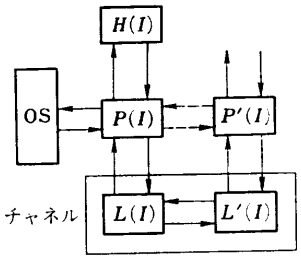

図 2 プロトコルのモデル化
Fig. 2 Model of a protocol.

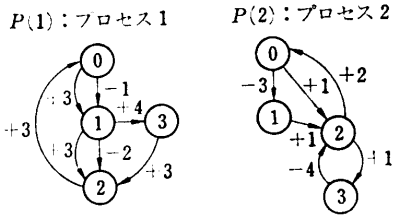

図 3 プロセスの有限状態グラフ表現
Fig. 3 Finite state graph representation of processes.

[定義 2] プロセスの表現（**図3** 参照）

プロセス $P(I)$ を次式で定義し，有限状態グラフで表現する．ここで，I はプロセス番号を示す．

$$P(I)=\langle K^I, A^I, \delta^{+I}, k_0^I, Q_f^I\rangle$$

ただし，

① K^I：プロセス I の状態集合．

② $A^I=\{\pm e_i^{IJ}\}$：プロセス I のアクション集合，アクションは状態間の遷移に伴うプロセスの送受信動作を示す．たとえば，$-e_i^{IJ}$ はプロセス I におけるプロセス J へのイベント e_i の送信動作を示す．以後，とくにまぎらわしくない場合は，$\pm e_i^{IJ}$ を $\pm e_i$ と略記する．

③ δ^{+I}：$K^I\times A^I\to K^I$ プロセス I の状態遷移関数．

④ k_0^I：プロセス I の初期状態．

⑤ Q_f^I：プロセス I の最終状態の集合．(定義終)

[定義 3] チャネルの表現

プロセス I からプロセス J への単方向伝送のチャネル $C(I,J)$ を次式で定義する．

$$C(I,J)=\langle K^{IJ}, A^{IJ}, \delta^{+IJ}, k_0^{IJ}\rangle$$

ただし，

① K^{IJ}：チャネル $C(I,J)$ の状態集合．

② A^{IJ}：チャネル $C(I,J)$ のアクション集合．

③ δ^{+IJ}：$K^{IJ}\times A^{IJ}\to K^{IJ}$ チャネルの状態遷移関数．

④ k_0^{IJ}：初期状態および最終状態.

チャネルの動作は，プロセスの動作に付随して生起するものとする．また，チャネルの初期状態と最終状態をともにイベントがない空の状態とする．さらに，チャネルの状態を蓄積されているイベントの到着順のシーケンスで表示する．たとえば，イベント1に続いてイベント2が入力された場合，チャネル状態を1，2で示す．（定義終）

[定義 4] システム状態

プロセス群とチャネル群の全体をシステムと呼ぶ．システムの状態 S_i をプロセスとチャネルの合成状態として次式で定義する．

$$S_i=\begin{bmatrix} Ps(1) & Cs(1,2)\cdots\cdots & Cs(1,N) \\ Cs(2,1) & Ps(2) & \\ Cs(3,1) & Cs(3,2) & \ddots \\ \vdots & \vdots & \\ Cs(N,1) & \cdots\cdots & Ps(N) \end{bmatrix}$$

ここで，$Ps(I)$ と $Cs(I,J)$ は，それぞれプロセス I とチャネル (I,J) の状態を示す．

システムの初期状態 S_0 と最終状態 S_f をすべてのプロセスとチャネルが，それぞれ初期状態と最終状態にある場合と定義する．システムの最終状態の集合を S_F とおく．（定義終）

[定義 5] システム状態の分類（図4参照）

システム状態 S_i から S_j へ，プロセスの有限回のアクションで推移できるとき，S_i は S_j へ到達可能という．この到達可能性の概念を用いて，システム状態を次の2種類に分割する．

① 正常状態：初期状態から到達可能で，かつ最終状態の集合に到達可能なシステム状態.

② 不適状態：初期状態から到達可能で，かつ最終状態の集合に到達不可能なシステム状態.

正常状態であってかつ不適状態への推移アークをもつシステム状態をとくに臨界状態と呼ぶ．また，臨界状態から，最終状態を除くターミナル状態（定義8）へ至る状態の系列を誤りシーケンスと呼ぶ.

正常状態の集合と不適状態の集合をそれぞれ NS と BS で示す．（定義終）

[定義 6] パータベーション

状態 X_i へのパータベーションを，X_i からの単一の推移によって到達できる状態と定義する．定義より，システム状態に対するパータベーションは一つのプロセスの状態を変化させ，その結果，対応するチャネル状態も変わる．二つ以上のプロセスによる同時推移はパータベーションのシーケンスとして表すことができる．（定義終）

[定義 7] 逆パータベーション

状態 X_i への逆パータベーションを単一の推移によって X_i へ到達できる推移元の状態の集合と定義する．（定義終）

[定義 8] ターミナル状態

システム状態のなかで，パータベーションをほどこしても推移先のない状態をターミナル状態と定義する．ターミナル状態のなかで，最終状態と受信不能状態（定義9）以外をデッドロックと呼ぶ．（定義終）

[定義 9] 受信不能状態とオーバフロー状態

プロセスあるいはチャネルの送信動作に対して，通信相手のチャネルあるいはプロセスに，対応する受信動作がないとき，その送信動作に対応するシステム状態を受信不能状態と呼ぶ．また，プロセスの送信動作によってチャネルのバッファ容量を越すイベントが送出されるとき，その送信動作の結果のシステム状態をオーバフロー状態と呼ぶ．（定義終）

[定義 10] 到達可能性グラフ

システム状態の到達可能性グラフ RG を次式で定義する．

$\mathrm{RG}=\langle S, A, \delta^+, S_0, S_F\rangle$

S：システム状態の集合.

$A=\bigcup_I A^I$：アクションの集合，A^I はプロセス I のアクション集合（定義2）.

δ^+：$S\times A\to S$ システム状態遷移関数.

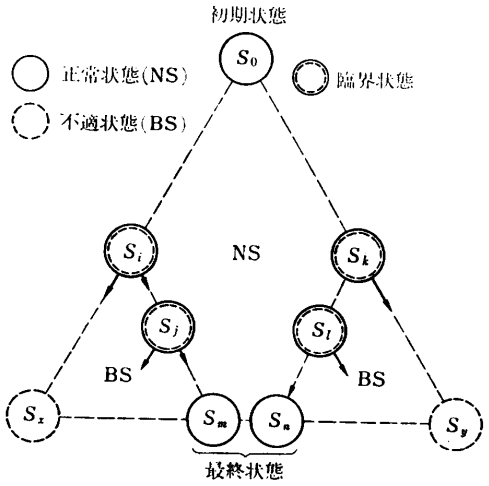

図 4 システム状態の分類
Fig. 4 Classification of system states.

S_0：システムの初期状態.

S_F：システムの最終状態の集合.

ここで，δ^+ はシステム状態に対してパータベーションを与える関数であり，後述する命題2で与えられる. (定義終)

[定義 11] 状態遷移逆関数

プロセス I の各状態に対して逆パータベーションを与える関数をプロセス状態遷移逆関数と呼び δ^{-I} で示す．同様にして，チャネル状態遷移逆関数とシステム状態遷移逆関数を，それぞれ δ^{-IJ} と δ^- で示す. (定義終)

本論文では，次の制限条件のもとでプロトコルの検証法について議論する.

[制限条件]

(1) プロセスとチャネル間でイベントの伝送に要する時間をゼロとする.

(2) チャネルに誤りがなく，そのバッファ容量は有限とし，オーバフローが生起しない範囲内でプロセス間で送受するイベントの受信順序は送信順序と同一とする.

4. 諸 命 題

本章では，3章で導入した諸概念を用いて，5章で与える検証アルゴリズムの基礎となるいくつかの命題を導く.

4.1 状態遷移関数と遷移逆関数

本論文では，被検証プロトコルは対応するプロセスとチャネルが有限状態グラフの形式（定義2と定義3）で表現されて与えられる.

δ^{+I} は定義よりプロセス I において，ある状態からアクションに応じて推移する推移先の状態を規定する．したがって，δ^{+I} は与えられたプロトコルに応じて決まる.

また，δ^{-I} は逆パータベーションを与える関数であり，δ^+ を用いて容易に構成することができる．たとえば，**図5**において，

$$\delta^{+I}(X_1, a_1)=X_3,\ \delta^{+I}(X_2, a_2)=X_3,$$

$$\delta^{+I}(X_3, a_4)=X_4,\ \cdots.$$

これらの関係式から δ^{-I} を構成することができる．状態 X_3 については，上記の関係式において最初の二つの式から，$\delta^{-I}(X_3)=\{X_1, X_2\}$ となる．他の状態についても同様にして得られる.

以上の関係を要約すると次の命題1が成立する.

[命題 1] プロセスとチャネルの状態遷移関数 δ^{+I} と δ^{+IJ} から，それぞれ対応する状態遷移逆関数 δ^{-I} と δ^{-IJ} を構成することができる. (命題終)

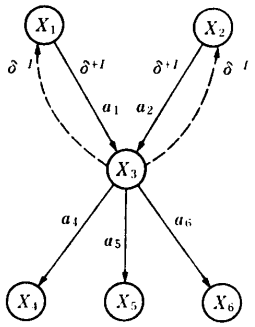

図 5 状態遷移逆関数
Fig. 5 State transition inverse function.

定義4と定義6から，次の命題2が成立する.

[命題 2] システム状態遷移関数 δ^+ は，すべてのプロセスとチャネルの状態遷移関数 δ^{+I} と δ^{+IJ} を用いて構成することができる. (命題終)

命題1を得た手順を用いると，同様にして，システム状態遷移関数について次の命題が成立する.

[命題 3] システム状態遷移逆関数 δ^- は，システム状態遷移関数 δ^+（命題2）を用いて構成することができる. (命題終)

4.2 RG の構成とシステム状態の分類

有限状態グラフの形式で与えられたプロトコルに対して，定義4のシステム状態に関する到達可能性グラフを以下のようにして構成する.

まずシステム初期状態を生成する．次に命題2を用いて前記の初期状態にパータベーションをほどこし，子ノードに対応するシステム状態を求める．以降，生成された各状態に命題2をくり返し適用し，ターミナル状態に至るまで続けると定義 10 のシステム状態に関する到達可能性グラフ RG が得られる.

また，状態 S_i にパータベーションをほどこし，得られた状態 S_j が初期状態から S_i に至るパス上の状態に一致するときは，ループが生起したとし，S_j を再度パータベーションの対象とはしないものとする．これは，パータベーションによって得られた状態に，付加情報として，初期状態からその状態に至るパス上にある状態のシーケンスを表示することにより可能となる.

以上のことから，次の命題4が成立する.

[命題 4] 与えられたプロトコルに対して，命題2を用いると，システム状態に関する到達可能性グラフ RG を構成することができる．また，RG 中のループ

も検出できる. (命題終)

RG において，システムの最終状態から出発し，命題3の δ^- を用いて，各状態に順次，逆パータベーションをほどこすと，最終状態に到達可能なシステム状態を類別できる．また，RG はシステム初期状態を起点として構成されるので，RG 中の状態はすべて初期状態から到達できる．したがって，システム状態を定義5で導入した正常状態と不適状態に分類できる．

さらに，命題2の δ^+ を用いて正常状態からの推移アークを調べることにより，正常状態でありかつ不適状態への推移アークをもつ臨界状態を類別し，誤りシーケンスを検出できる．

以上を要約すると次の命題5が成立する．

[命題 5] 到達可能性グラフ RG において，最終状態を出発点とし，逆パータベーションをほどこすことにより，RG 中のシステム状態を正常状態と不適状態に分類できる．また臨界状態も類別し誤りシーケンスも検出できる. (命題終)

5. 検証アルゴリズム

本章では，まず文献 2) のパータベーション解析を示し，次に本論文の検証法 EXPA を与える．

5.1 パータベーション解析[2)]

パータベーション解析によるプロトコル検証法の手順を以下に要約する．

Step-1：システム状態の集合を SS とおく．アルゴリズムが開始する時点では，$SS=\{S_0\}$．ここで，S_0 はシステムの初期状態である．

Step-2：SS のなかからパータベーションがほどこされていない状態 S_k を選ぶ．もし，そのような状態がなければ，アルゴリズムは停止する．

Step-3：Step-2 で選択された S_k へパータベーションをほどこすことによって得られるシステム状態の集合 SS_p を求める．

Step-4：SS_p が空ならば S_k をターミナル状態として出力する．

Step-5：SS_p のなかに受信不能エラーやオーバフローエラーのシステム状態が存在すれば，これらの状態を出力し SS_p から除去する．

Step-6：Step-5 で SS_p から除去した残りのシステム状態で SS 中にないものを SS に加える．

Step-7：Step-2 へもどる．

(アルゴリズム終了)

[パータベーション解析の特徴]

この検証アルゴリズムの本質は，パータベーションにより到達可能性グラフを生成する点にあり，① デッドロック状態，② 受信不能状態，③ チャネルのオーバフロー状態を検出することである．誤りの原因や誤りに至る過程については，設計者が到達可能性グラフを詳細に追跡し調べなければならない．

一方，5.2 節で与える EXPA では，アルゴリズム自身で誤りの原因や過程を検出することができる．

5.2 EXPA アルゴリズム

EXPA (EXtended Perturbation Analysis) は 4.1 節で述べたパータベーション解析に，逆パータベーションを導入することにより，検証能力の向上を達成したプロトコル検証法である．

以下の Step-1 から Step-6 までは 4.1 節のパータベーション解析と基本的に同一であり，EXPA の本質は Step-7 以降である．

[EXPA アルゴリズム]

Step-1：① システム状態の集合を SS とおく．アルゴリズムが開始する時点では，$SS=\{S_0\}$．ここで，S_0 はシステムの初期状態であり，到達可能性グラフの頂点とする．

② Step-2 へ進む．

Step-2：① SS のなかから最終状態を除きパータベーションがほどこされていない状態 S_k を選ぶ．もし，そのような状態がなければ，Step-7 へ進む．

② Step-3 へ進む．

Step-3：① Step-2 で選択された S_k へパータベーションをほどこすことによって得られるシステム状態の集合 SS_p を求める．

② $SS_p \neq$ 空

a) S_k の子ノードとして，SS_p のすべての状態を付加し段階的に到達可能性グラフを構成する（命題4）．

b) 到達可能性グラフ中にループが検出されれば，ループを構成するシステム状態のシーケンスを作成する（命題4）．

c) Step-5 へ進む．

③ SS_p＝空：Step-4 へ進む．

Step-4：① SS_p が空ならば，S_k をターミナル状態として類別する．さらに，これを受信不能状態とデッドロック状態に分類する．

② Step-2 へ進む．

Step-5：① SS_p のなかに，オーバフローのシステ

ム状態が存在すれば，これらの状態を類別し SS_r から除去する．

② Step-6 へ進む．

Step-6：① Step-5 で SS_r から除去した残りのシステム状態で SS 中にないものを SS 中に加える．

② Step-2 へ進む．

……以下は逆パータベーションの操作……

Step-7：① 与えられた最終状態の集合からまだ逆パータベーションがほどこされていない状態 S_i を選び $SS=\{S_i\}$ とおく．もし，そのような状態がなければ Step-10 へ進む．

② Step-8 へ進む．

Step-8：① SS の中からシステム初期状態を除き，まだ逆パータベーションがほどこされていない状態 S_j を選ぶ．そのような状態がなければ，Step-7 へ進む．

② Step-9 へ進む．

Step-9：① Step-8 で選択された S_j へ逆パータベーションをほどこすことによって得られるシステム状態の集合 SS_r を求める（命題3）．

② SS_r のすべての状態を正常状態として類別する（命題5）．

③ SS_r のすべての状態を SS に加える．

④ Step-8 へ進む．

Step-10：① Step-3 で構成された到達可能性グラフにおいて，システム初期状態，最終状態と正常状態を除く他のシステム状態を不適状態として類別する．

② 不適状態への推移アークをもつ正常状態を臨界状態として類別する．また，臨界状態からの誤りシーケンスを構成する．

③ Step-11 へ進む．

Step-11：検証結果の出力

① 到達可能性グラフ関係

1) 初期状態，2) 最終状態，3) 正常状態，4) 不適状態，5) 各状態間の推移アーク

② 誤り関係

1) 臨界状態，2) 臨界状態からの誤りシーケンス，3) デッドロック状態，4) 受信不能状態，5) オーバフロー状態，6) ループを構成するシステム状態のシーケンス．

（アルゴリズム終了）

[EXPA アルゴリズムの特徴]

パータベーション解析[2]と EXPA の主たる相違点は，逆パータベーションにある．逆パータベーションの導入により，検証能力が向上したおもな項目は次の2点である．

（1） システム状態を正常状態と不適状態に分類（命題5）．

（2） 臨界状態を検出し，誤りシーケンスによりプロトコル誤りの原因となる動作規定の指摘（命題5）．

その他ループの検出などがある．なお，制限条件と生成されるシステム状態の総数については，文献 2) と同一である．

6. 例 題

いくつかの誤りを含む簡単な二つのプロセス間通信をとりあげ EXPA の適用例を示す．図3は文献 3) で用いられた例題で被検証プロトコルに対応するプロセス $P(1)$ と $P(2)$ を示す．

前記の例に EXPA アルゴリズムを適用すると種々の誤りが検出される．たとえば，生成されるシステム状態に関する到達可能性グラフは**図6**のようになる．同図において，チャネル状態のEはチャネルが空の状態を表す．

初期状態と最終状態をともに S_0，また各チャネルのバッファ容量を5とすると以下の結果が得られる．

（1） 正常状態

S_0, S_1, S_2, S_4, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16}, S_{17}, S_{18}, S_{19}.

（2） 不適状態

S_3, S_5, S_6, S_7, S_{20}, S_{21}, S_{21} に続く状態．

（3） 臨界状態

S_1, S_2, S_4, S_8.

（4） 誤りシーケンス

① $S_1 \rightarrow S_{20} \rightarrow \cdots$, ② $S_2 \rightarrow S_3 \rightarrow S_5$, ③ $S_4 \rightarrow S_6 \rightarrow S_7$, ④ $S_8 \rightarrow S_7$.

（5） 受信不能状態

S_5.

（6） デッドロック

S_7.

以上のように，受信不能状態やデッドロックなどの誤りを検出するだけでなく，臨界状態や誤りシーケンスなど誤りに関係する情報を検出し提供している．これらのことから，EXPA は従来のパータベーション解析より能力が高い検証システムとなっていることがわかる．

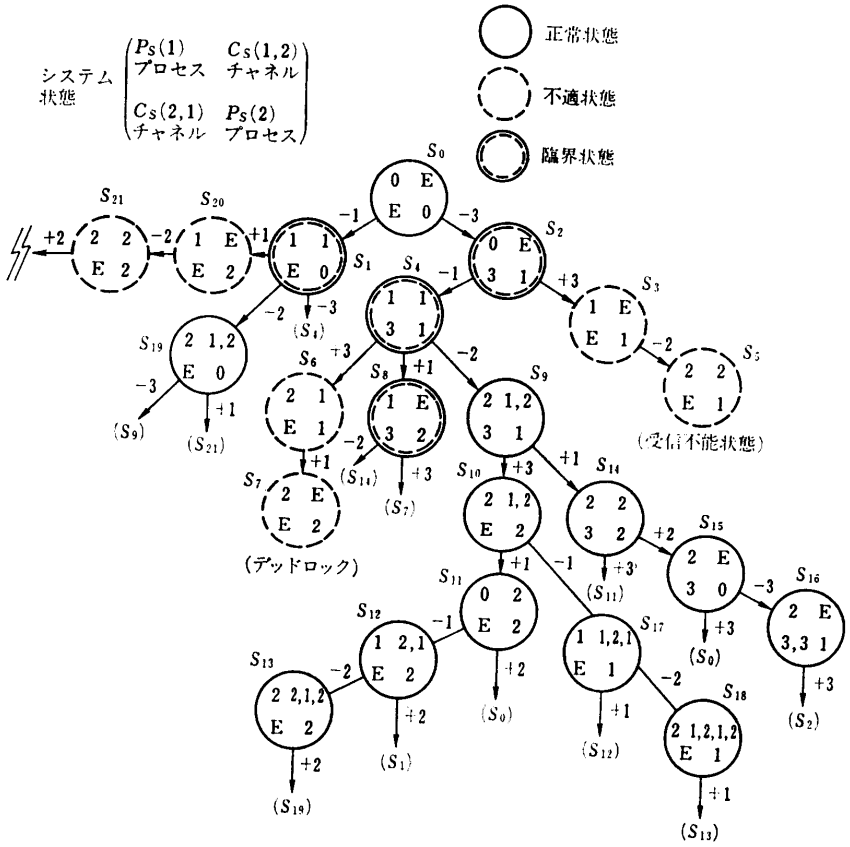

図 6 図2に対応する到達可能性グラフ
Fig. 6 A reachability graph corresponding to Fig. 2.

7. むすび

本論文では，プロトコルの検証法について検討し，とくに，状態遷移に注目したパータベーション解析を用いる検証法 EXPA を与えた．EXPA では，従来のパータベーション解析に，逆パータベーションを導入することにより，プロトコル誤りの原因となる遷移の検出など検証能力の向上を達成している．

また，EXPA アルゴリズムの実験システムを計算機システム (FACOM-U 100) 上にインプリメントし，実証実験を行い動作を確認した．アルゴリズムは PASCAL で記述されているが，使用する計算機の制約から，ソフトウェアは FORTRAN を用いて作成した．プログラムの大きさは，検証アルゴリズムと被検証プロトコルの入力やエラーの出力関係も含めて，FORTRAN で約 1,000 ステップとなっている．実験システムでは，メモリ容量が十分でなく，また入出力にプリンタ装置を用いているため，マンマシンインタフェースが不十分である．そのため，実用システムの構築にあたっては，ディスプレイ装置を用いて，図6を画面に直接表示するなど，マンマシンインタフェースを向上させることが肝要である．

今後の課題としては，① 制約条件の緩和，② 解析効率の向上などの問題が残されている．① については，たとえば，テンポラルオペレータを用いた時間関係や誤りの導入などを考慮する必要がある．② は，主として生成される状態数の削減法についての課題である．

謝辞 日頃から熱心に討論していただく東北大学野口研究室の諸氏に深謝する．

参考文献

1) Zafiropulo, P.: Protocol Validation by Duologue—Matrix Analysis, *IEEE Trans. Commun.*, Vol. COM-26, No. 8, pp. 1187-1194 (1978).
2) West, C. H.: General Technique for Communications Protocol Validation, *IBM J. Res. Develop.*, Vol. 22, No. 4, pp. 393-404 (1978).

3) Zafiropulo, P. et al.: Towards Analyzing and Synthesizing Protocols, *IEEE Trans. Commun.*, Vol. COM-28, No. 4, pp. 651-661 (1980).
4) 吉田，野村，田中：データの欠落・重複を検査する通信プロトコル検証法，信学論(D)，Vol. J 63-B, No. 11, pp. 1094-1101 (1980).
5) 郷原，白鳥，野口：通信プロトコルの効果的設計法，信学会，計算機研究会技報，EC 81-70, pp. 1-10 (1982).
6) Shiratori, N., Gohara, J. and Noguchi, S.: A New Design Language for Communication Protocols and a Systematic Design Method of Communication Systems, Proc. 6-th ICSE Congress, pp. 403-412 (1982).
7) Itoh, M. and Ichikawa, H.: Protocol Verification Algorithm Using Reduced Reachability Analysis, *Trans. of IECE*, Vol. E 66, No. 2, pp. 88-93 (1983).
8) Stenning, N. V.: A Data Transfer Protocol, *Computer Networks*, Vol. 1, pp. 99-110 (1976).
9) Bockmann, G. V. and Gecsei, J.: A Unified Method for the Specification and Verification of Protocols, Proc. IFIP Congress, pp. 229-234 (1977).
10) 高橋，白鳥，野口：ネットワークソフトウェアの設計法，情報処理学会，ソフトウェア工学研究会資料，28-3 (1982).
11) 高橋，白鳥，野口：IDL：ネットワークソフトウェアの設計記述言語，情報処理学会，分散処理システム研究会資料，22-3 (1984).

(昭和59年6月4日受付)
(昭和59年10月18日採録)