解　説

プロトコルの記述法と検証法†

河岡　司†† 友永充宏†† 高橋祥兼††

1. まえがき

近年，各種のコンピュータ・ネットワーク・アーキテクチャが相次いで発表されているが，それらはいずれも複雑化したコンピュータ・ネットワークを実現する技術を統一整理された体系としてとらえようとするものである．その中で重要な位置を占めるプロトコル(通信規約)は，コンピュータ・ネットワークを構成する各ノード（ホスト計算機，前置処理装置，端末等）上の各種ハードウェア，ソフトウェアが相互に通信し合うための情報送受信に関するとり決め（例えば，情報転送単位の長さ，そのビット構成，情報送受信順序など）のことである．プロトコルは規約の集合（仕様：specification)であり，各装置上で通信を実行する主体となるハードウェア，ソフトウェアはすべてこの仕様にもとづいて実現される．したがって，もしもプロトコルの仕様記述の中に，何通りにも解釈できる様な曖昧な点があると，それにもとづき実現されたコンピュータ・ネットワークは，通信するノード相互間でプロトコルの相違を生じ，正しく通信することができなくなるであろう．この様な事態を回避するためには，プロトコルの仕様を一意に解釈可能な形式に記述することが必要であり，ここにプロトコルの記述法を検討する意義がある．一方，プロトコルがこの様な記述法によって曖昧さなく記述されていたとしても，もともと規定内容それ自体に何らかの誤り（例えば，情報送受信順序に関する通信主体相互間での不統一など）があったとすれば，やはりそれにもとづくシステムは期待通りには動作しない可能性がある．ゆえに，プロトコルを設計する場合，規定内容にそういった論理ミスを含まないことを確認しておくことが必要であり，したがって，プロトコルの論理的無矛盾性を保証する方法，すなわちプロトコルの検証法が重要となってくる．

本稿では，上記の各技術について，既存技術の概要と特徴および今後の課題を述べる．

2. プロトコルの記述法

2.1 プロトコルの特徴

プロトコルにしたがって通信を実行する抽象的な装置を考え，これをプロトコルマシン（以下 PM と略記する）と呼ぶことにする(図-1)．プロトコルとして規定される内容は，基本的には PM が通信相手 PM から情報を受信した際に，その情報種別と受信のタイミングに対応して，とるべきアクション（一般には通信相手 PM への出力）を定めるものである．この様な PM の動作は一種のオートマトン（一般には有限状態）としてモデル化することができ，したがって，プロトコルを定めることは1つの有限オートマトンを定めることに対応し，プロトコルの記述法はその表現法に対応している．通信はプロトコルにもとづく2つの(一般には複数の) PM の並列的相互動作であり，このことからプロトコルの規定では通信相手 PM との同期を考慮した規定が重要な位置を占めている．この様な規定は情報の送受信シーケンスに係わるものが主体であり必ずしも完結したアルゴリズムの形をとらないことが多い．一方，プロトコルには，フロー制御の一方式であるウィンドウプロトコルの様に，データのシーケンス番号のカウントの様なパラメトリックな処理手順も存在し，この場合，その規定内容は一般に，発生するイベント対応の PM 動作としてプログラムロジック的なアルゴリズムの形式で与えられる．更に，こういった2つの要素が複合された内容のプロトコルも存

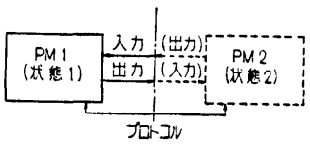


図-1　プロトコルマシン (PM)

† Protocol Description and Verification Techniques by Tsukasa KAWAOKA, Mitsuhiro TOMONAGA and Yoshikane TAKAHASHI (Data Communication Network Section, Yokosuka Electrical Communication Laboratory, N. T. T.).

†† 日本電信電話公社横須賀電気通信研究所

在する．これらはいずれも本質的には有限オートマトンの動作としてモデル化できるものであるが，それぞれに適した表現法は大きく相違すると考えられる．また，プロトコルが単なる並列処理の仕様とは異なる点として，通信相手 PM の動作の誤りや，伝送媒体上のデータの紛失あるいは損傷といった例外的事象が発生するタイミングを網羅的に考慮し，各々に対するシステムの回復動作の規定が必要なことがあげられる．

2.2 プロトコル記述法の具備条件

プロトコルの規定はプログラムの規定とは異なり，一般には，特定のハードウェア／ソフトウェアとは独立な仕様として規定されるものであるが，それをもとに PM を実際の装置上にインプリメントすることが可能なためには，それが何らかの形で一意に解釈できるドキュメントとして表現されていなければならない．更に，このドキュメントの記述法は以下の様な条件を備えていることが必要と考えられる．

- 記述が容易なこと
- 記述されたものが解り易いこと
- 規定内容が厳密であること
- プロトコルの検証に有効なこと
- 例外処理を含めた網羅的規定が可能なこと
- プロトコルのインプリメントに対し有効なこと
- 仕様のメンテナンスが容易であること

しかしながら，これらの条件の中には相反した性質のものもあって，すべてを満足する様な記述法の開発は困難である．そこで通常は，プロトコルの記述に際し，その記述の目的に応じて特に重要な具備条件と記述対象プロトコルの特性とを考慮して最も適した記述法が使用される．

2.3 各種記述法

ここでは，既に適用され，あるいは提案されている各種のプロトコル記述法について，その概要を紹介しそれらの特徴について考察する．

（1） 状態遷移図

最も広く用いられている方法であり，記法には種々の相違があるが，いずれも本質的には，状態を表わす節点（ノード）と，状態間の遷移を表わす有向枝（アーク）から成る有向グラフとして構成される．アークには一般に，遷移要因とそれに伴うアクション（通常は出力）とが対応している．PM の動作が直観的に解り易いことがこの方法の特徴である．**図-2** は状態遷移図の一例である．

（2） 状態遷移表

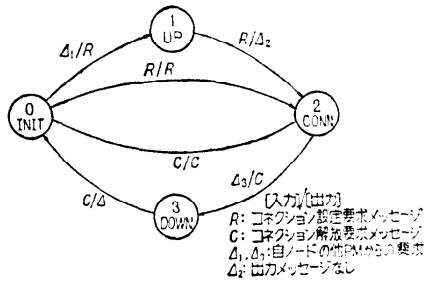


図-2 状態遷移図の例（コネクション設定／解放プロトコル）

これは基本的には状態遷移図を表形式に描いたものであり，状態と状態遷移要因のあらゆる組合せに対して，遷移先状態とアクションとを一覧表の形に記述する方法である．この方法は状態遷移図に較べてスペースファクタが良く，また規定の網羅性の点でもすぐれているが，記述内容の解読の容易性という点では劣っている．**図-3** は状態遷移表の記法の一例である．

（3） 状態遷移表と処理状態遷移表の組合せ[28),29)]

一般に，状態遷移表のみでは状態遷移に伴う処理手順（例えば，カウンタの更新の様なパラメトリックな処理）の詳細な規定が困難である．この方法では，こ

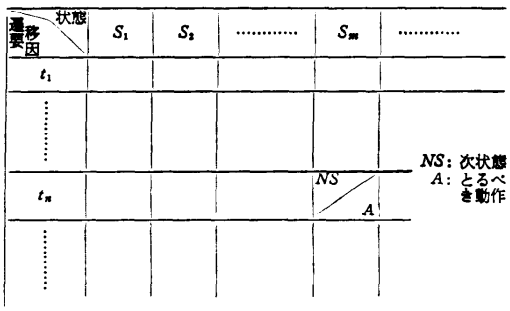


図-3 状態遷移表

入力とその解析	処　　理	出　　力	番　　号
（処理を実行する条件）	（カウンタ等の更新）	（出力データ種別）	状態遷移表で指定される番号
⋮	⋮	⋮	⋮
$L \leqq N(R) \leqq S^*$	$L \leftarrow N(R)^{**}$	——	A
⋮	⋮	⋮	⋮

* 到着したメッセージの特定のフィールドの値 $N(R)$ が妥当な範囲内にあること．

** L を $N(R)$ で更新

図-4 処理状態遷移表

れらの処理手順を別表（処理状態遷移表）として記述しておき，状態遷移表のアクション記述（状態と遷移要因との交点）において対応する処理のエントリ番号を指定してマッピングをとることにより，処理手順の明確な表現をはかっている．図-4 に処理状態遷移表の一例を示す．

（4） Petri Net[23),24)]

Petri Net は，P節点（"○" で表わす）とT節点（"｜" 又は "－" で表わす）の2種類の節点とそれらを相互に結合するアークとで構成される並列処理表現のグラフモデルである．P節点にはトークン（"・" で表わす）を複数個配置できる．1つのT節点に関してその各々の入力P節点（そのT節点を終端とする様なアークの始端のP節点）にトークンが1個以上配置されているとき，そのT節点は発火可能であるという．発火可能なT節点が発火すると，その各々の入力P節点からトークンを1個づつ取り除き，各出力P節点（そのT節点を始端とする様なアークの終端のP節点）にトークンを1個ずつ付け加える．P節点にトークンが存在することはそのP節点に対応する条件が保持されていることに対応し，T節点の発火は事象の発生又はある動作の完了に対応する．図-5 に Petri Net による簡単なプロトコルの記述例を示す．この方法の特徴は，通信する複数の PM およびその間の伝送媒体を一体の系として表現するため同期動作が明確にとらえ易い反面，非同期的に発生する例外処理（例えば，伝送回線上でのデータの紛失など）に対する動作規定が困難な点にある．

（5） UCLA グラフ[30),31)]

UCLA グラフは頂点（vertex）とアークとから成る Petri Net に類似した並列処理表現モデルであり，その頂点およびアークはそれぞれ Petri Net のT節点およびP節点にほぼ対応した意味をもっている．また，Petri Net と同様，節点に対応付けられたアクションの実行に伴いアーク上をトークンが移動するが，UCLA グラフでは，1つの節点の実行可能条件とトークンの除去／付加の条件に各々 EOR(何れか1つのアークを対象とする)と AND(すべてのアークを対象とする）の2つを設けて多様な条件付動作の記述を図っている．図-6 にメッセージ送受信プロトコルの UCLA グラフによる記述例を示す．

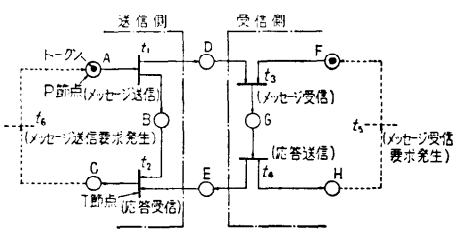


図-5 Petri Net（メッセージ送受信モデル）

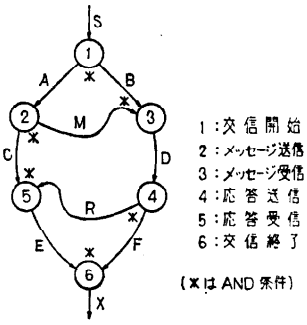


図-6 UCLA グラフの記述例

（6） PDC 法[25),40)]

これは Petri Net をベースとして，それに通信の同期動作のより解り易い表現と，実システムの制御動作との対応付けを容易にするための改良を加えた記述法である．PDC (Protocol Description Chart) は表-1 に掲げる様な各節点をアークによって結合した一種の有向グラフである．PDC では Petri Net と同様に，E節点の発火によるトークン移動として通信系の動作を表現するが，更に実システムとの対応を容易にするためにE節点における処理手順の記述や，論理判断によるトークンの振り分けなどを可能としている．図-7 にウィンドウ方式によるフロー制御プロトコルの PDC 法による記述例を示す．

（7） プロトコルグラフ[11)]

これは PM の動作を図-8 に掲げる4種の基本動作によって流れ図的に記述する方法であり，図-9 の記述例でもわかる様に，メッセージの送受信シーケンスの表現に重点をおいた記述法といえる．また，送信／

表-1 PDC を構成する節点

節点名称	記　号	実システムへの対応
E 節点	AND / OR	制御動作の実行完了事象の発生
P		動作の実行中または実行待ち
Q		待ち行列（キュー）
S		制御またはデータの複数方向への同時移行
L		論理判断による制御またはデータの分岐

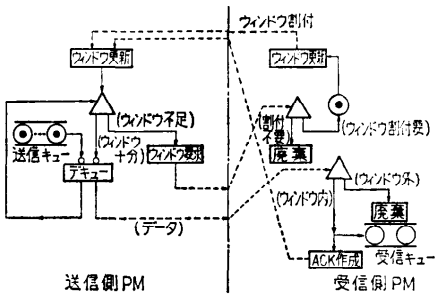


図-7 PDC による記述例

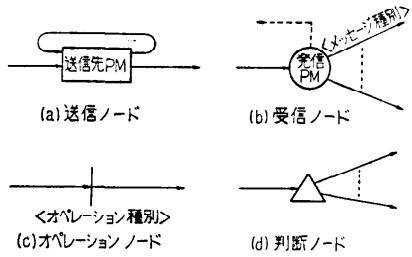


図-8 プロトコルグラフの構成要素

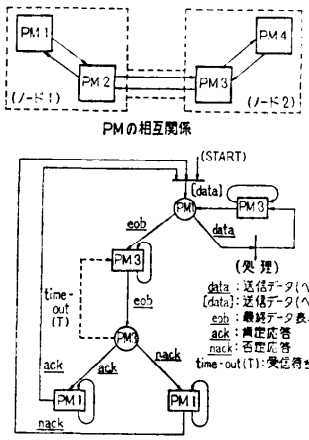


図-9 プロトコルグラフの記述例 (PM2 の動作)

受信ノードには，それぞれ送信／受信メッセージの宛先／発信元 PM 名を記述できるため，複数個(3以上)の PM が通信する場合の記述にも適用性があるが，常に唯一つの PM からの受信待ち状態しか考慮できないため，各 PM からのメッセージ受信タイミングを網羅的に記述することが困難となる．

(8) 高級言語による方法[2),7),15),34)]

PL/I や PASCAL などの汎用プログラム言語に近い言語をベースとして，それにプロトコルの表現に適した固有の命令セット（例えば，SEND, WAIT, ENQUEUE などのステートメント）を追加した言語仕様によって記述する方法である．この方法の特徴は実システムの動作に即した処理手順を詳細に記述することが可能であり，プロトコルのインプリメントへの適応性が高いこと，および図式的な記法に較べてプロトコル仕様書のマシン管理が容易であることなどがあげられる．一方，プロトコルの視覚的把握が困難なこと，および処理順序の一意的記述により過剰規定に陥り易いことなどの欠点がある．

(9) その他の方法

以上述べた方法のほかに次の様なものがある．

12)では，プロトコルにもとづく通信の開始から終了までにおける PM の情報送受信シーケンスが1つの正則文法から生成されるセンテンスに対応することに着目して，その正則文法の生成規則を記述することによりプロトコルを規定する方法を提案している．また，20)では，PM の動作メカニズムを，通信相手 PM とのメッセージ送受信動作と自 PM のユーザとの間のコマンドおよびテキストの授受動作とに分解してとらえる colloquy の概念を導入し，送信するメッセージ，コマンドおよびテキストと入力および状態変数との相互関係を論理マトリクスとして表現することによりプロトコルの形式定義を行っている．また，8)は，同様の考え方によるプロトコルの定義方法を述べているが，そこでは論理マトリクスの代りに ALGOL 的な言語が用いられている．また，14)では階層構成をなすプロトコルの各階層を3入力2出力オートマトン（各階層の上位／下位の階層との入出力とその階層に固有のステータス信号入力とを有する有限状態機械）としてモデル化し，状態と入力の対ごとに遷移先状態と出力とを網羅的に記述する（状態遷移関数を定義する）ことによる規定方法が述べられている．

2.4 記述法のまとめ

前節で述べた各種のプロトコル記述法は，大別して通信する2つの PM を一体の系として記述する方法と，個々の PM を個別に記述する方法とに分類でき

表-2 プロトコル記述法の分類 (1)

記述法	特徴評価	適用領域	例
PM 対の一体記述	・同期動作の把握が容易(○) ・非同期的例外処理の規定が困難(×)	プロトコルの中核機能のみを正確に記述する必要がある場合など．	・Petri Net ・UCLA グラフ ・PDC 法
各PMの個別記述	・例外処理の規定が容易(○) ・同期動作がわかりにくい(×)	例外処理を含めて網羅的な規定を必要とする場合．	本稿で述べた上記以外の各記述法

○：長所，×：欠点

表-3 プロトコル記述法の分類（2）

記述法	特徴評価	適用領域	例
状態遷移の記述	・同期動作の把握が容易(○) ・実システムの処理手順との対応が困難(×)	同期動作を主体とするプロトコル（コネクション設定／解放プロトコルなど）	・状態遷移図（表）
処理手順の記述	・実システム動作との対応が容易(○) ・インプリメントに有効(○) ・同期動作がわかりにくい(×)	処理動作を主体とするプロトコル（データ転送プロトコルなど）	・高級言語 ・プロトコルグラフ
状態遷移と処理手順を併せた記述	・上記2者の中間的特徴をもつ	同期動作と処理手順とが混在するプロトコル	・状態遷移表と処理状態遷移表の組合せ ・Petri Net ・UCLA グラフ ・PDC 法

○：長所，×：欠点

る．各々の一般的特徴とその適用領域を表-2 に示す．一方，別の観点からの分類として，通信の進行のタイミングを状態としてとらえ，その遷移に着目した方法と，実際のシステムにおけるプログラムの処理手順に即した方法およびその中間的な方法の3つに分類される．これらの特徴とその適用領域を表-3 に示す．表-2, 表-3 は記述法の原理的特徴からみた分類，評価であり，各カテゴリに属する具体的記述法が共通に持つ本質的な利害得失といえる．これらの各カテゴリはいずれかが他よりすぐれたものであるというわけではなく，一般に，記述対象プロトコルの特性と記述の目的とによってその適用性を異にするものである．今後も各種のプロトコルの設計／インプリメントとともに，種々の記述法が考案されるものと考えられるが，望ましい方向の1つは，表-2, 表-3 に掲げた各カテゴリ対応に標準的な記述法を制定することであろう．1978年に設けられた開放形システム間相互接続のための国際標準アーキテクチャを検討する ISO/TC 97/SC 16 分科委員会にもアーキテクチャの形式的記述法を制定するためのプロジェクトが設けられており，そこでの結論あるいは勧告案はこの様な標準記述法に関する一つの方向を示すことになると思われる[13]．なお，ここに述べた各記述法はいずれもプロトコルの仕様記述を目的としたものであり，プロトコルのインプリメントを目的とした記述法（プロトコル処理プログラムの記述法）ではない．後者の立場での記述法は未だ例を見ない様であるが，プロトコル実現上の技術として今後の課題の一つであろう．

3. プロトコルの検証法

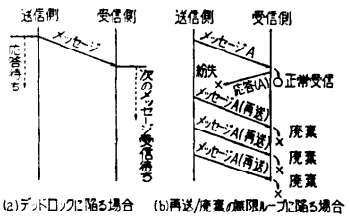
3.1 検証の必要性

プロトコルとしての規定内容が複雑になると，それがたとえ前述の様な形式的記述法によって厳密に記述されていたとしても，それにもとづく通信において所期の目的を達成できるかどうかは必ずしも明らかではなくなる．すなわち，例えば次の様なプロトコルエラーが存在する可能性がある（図-10 参照）．

a）送信側はメッセージの送信後，受信側からの応答を待つのに対し，受信側はそのメッセージに対し応答を返送することなく次のメッセージの受信待ち状態になる様に規定されていたために，システムがデッドロック状態（互いに相手のアクションを待っており通信が進行しない状態）に陥る（図-10(a)）．

b）送信側はメッセージの送信後，タイマをセットして受信側からの応答を監視し，一定時間経過するごとに同一メッセージの再送を行い，受信側はメッセージ受信後直ちに応答を返送するが，再度同一メッセージが到着した場合はこれを単純に廃棄するのみと規定されていたとする．この時もしも図-10(b)に示す様に受信側からの応答が転送途中で紛失すると，システムはメッセージの再送／廃棄という明らかに不当な無限ループに陥る．

これらはいずれも比較的単純な規定ミスの例であるが，プロトコルが複雑になるといくつかの例について図-10 の様なシーケンスチャートを描いて調べるだけでは，その完全性の保証には十分とは言えなくなる．したがって，プロトコルの設計段階において，この様な論理的無矛盾性の検証を行うためのシステマティックな手法を考えることが必要となる．また，この様な検証が厳密に行われるためには，検証に先立ってプロトコル自体が正確に記述されていることが必要であり検証はそれをベースとして行われる．このことからプロトコルの検証法は記述法と密接な対応関係をもって



(a)デッドロックに陥る場合　(b)再送/廃棄の無限ループに陥る場合

図-10 プロトコルエラーの例

いる.

3.2 各種検証法

ここでは，既に公表されている主なプロトコル検証法についてその概要を紹介する.

(1) 状態遷移図を主体とした方法

Zafiropulo[49),50)]は，状態遷移図で規定されたプロトコルについて，その初期状態から出発して再び初期状態に戻るまでのイベント系列 (unilogue) を，通信する2つの PM に対して組合せたもの (duologue) が，(a)正しく実行可能であるか，(b)起り得ない系列であるか，あるいは(c)プロトコルエラーの可能性がある ((a), (b)以外) かに分類して検証する方法を提案している．この検証法のベースを与える(a)の条件は次の3つからなる.

① 送信されたデータは必ず相手に受信されること (post-transmission condition)

② 受信されるデータは必ず相手によって送信されたものであること (pre-reception condition)

③ データの到着があり得るタイミングでは必ずそれを受信可能なこと (completeness condition)

例えば，**図-11** のプロトコルをこの方法によって検証すると以下の様になる.

先ず，PM_A および PM_B の unilogues はそれぞれ，$A_1=\{-2,+3\}$，$A_2=\{+2,+3\}$ および $B_1=\{+2,-3\}$，$B_2=\{-2,-3\}$ となり，これから duologues は $[A_1, B_1]$，$[A_1, B_2]$，$[A_2, B_1]$，および $[A_2, B_2]$ の4つとなる．これらの duologues に前記のアルゴリズムを適用すると，$[A_1, B_1]$ および $[A_2, B_2]$ は正しく実行可能，$[A_2, B_1]$ は起り得ない系列，そして $[A_1, B_2]$ はプロトコルエラー(すなわち，図-11 のプロトコルでは要求の衝突に対する考慮がなされていない) となることがわかり，図-11 のプロトコルの論理ミスが検出される.

この検証手順は自動化されているが[44),45)]，適用に際しては状態遷移図に初期状態を含まないループが存在してはならないことなど，かなり制約条件が多い．また，Danthine[9)]は，先ず通信の実行において考えられるあらゆるデッドロック状態を見い出し，実際に通信系がその様な状態に到達するか否かを調べるために，通信する2つの PM の状態対をデッドロック状態から初期状態へ向けて逆向きにたどることを提案している．この手順は自動化されているというが，9)にはその具体的アルゴリズムは述べられていない．一方，6)は2つのPM間の伝送媒体を転送中のメッセージが存在しない状態のみに着目し (empty medium abstraction)，その間の遷移を描いた状態遷移図を解析することによる検証を述べている．この方法は，伝送媒体の状態変化を両 PM 状態対間の遷移要因の中で考慮するため場合によっては解析をかえって煩雑なものにするが，半二重通信など，一般に多数のデータが同時に伝送媒体中に存在することが少ない場合には有効と考えられる．また，17)では，伝送媒体中のメッセージが実質的に高々1個とみなされている場合についてデッドロックやループの検出等のアルゴリズムが述べられている.

(2) 合成状態図を用いた方法

Sunshine[35)] および Rusbridge[32)] は，通信する2つの PM の状態対と PM 間を転送中のメッセージの状態 (種別，方向，順序など) を同時に考慮した合成状態 (composite state) を定義し，その遷移を描いた図(合成状態図)を用いて通信系が到達可能な状態を確認する方法を提案している．**図-12** は2.3節の図-2で規定されたプロトコルの合成状態図による検証例である．この方法は基本的には考えられる合成状態間の遷移を網羅的に描くことを必要とするが，これはスペースファクタや記述工数などの点でも困難であり，35)では検証上本質的な状態のみを描くという方法をとっている．ただし，その場合には真に網羅的な検証が行われたということを別の方法で確認する必要がある．West[46)]は同様の方法を3つ以上の PM が相互に通信を行う場合にも適用し，更に伝送回線上のメッセージ・オーバフローに関するエラー検出問題も取扱って

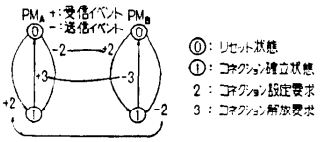


図-11 検証対象プロトコル

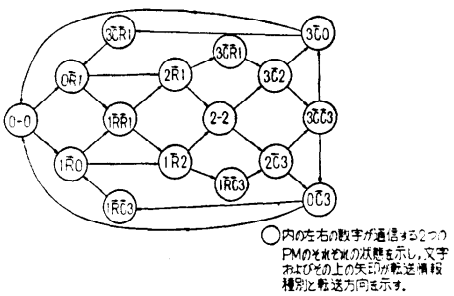


図-12 図-2 に対する合成状態図

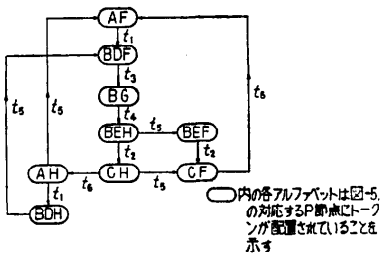


図-13 図-5 の Petri Net に対するトークンマシン

おり，そのアルゴリズムは完全に自動化されている．

（3） Petri Net のトークンマシンによる方法[21)~24)]

Petri Net におけるすべてのP節点に対するトークンの配置(マーキング)はT節点の発火とともに変化する．この変化の状況をグラフ化したものをトークンマシン (TM) と呼ぶが，Petri Net で記述されたプロトコルの場合 TM は一種の合成状態図と考えることができ，それを解析することにより(2)と同様の検証が可能である．例えば，図-5 の Petri Net に対する TM は図-13 の様になり，これから，このプロトコルの下ではデッドロックが存在しないことがわかる．ただし，Petri Net ではトークン相互の識別はできないため，例えば転送データの種別が問題となる様な検証は，その記述と同様，非常に困難となる．一方，Merlin は Petri Net のT節点に，それが発火可能となってから実際に発火するまでの最小時間と最大時間を指定するという制限を付加した Time Petri Net の概念を導入し，それを用いてトークンの紛失（転送中メッセージの紛失に相当）に対するプロトコルの誤り回復可能性の問題を取扱っている．

（4） UCLA グラフを用いた方法[30),31)]

UCLA グラフのアーク上のトークン配置は節点の実行に伴って変化し，Petri Net の TM と類似した性質をもつから，それを用いて(3)と同様の検証が可能である．一方，その1つの節点の実行によるトークン配置のローカルな変化は変換表現 (transformation expression : TE) と呼ばれ，その集合に機械的代入操作 (reduction procedure) を行うことによって状態の到達可能性（特に通信の終了性）に関する検証が可能である．例えば，図-6 の UCLA グラフに対する TE 全体の集合は次の様になる．

S⟶A, B;
A⟶C, M;
B, M⟶D;
D⟶F, R;
C, R⟶E;
E, F⟶X;

これから次の様な reduction step が生成され，通信の終了性が証明される．

S⇒A, B⇒C, M, B⇒C, D⇒C, F, R⇒F, E⇒X

（5） 処理ロジックを解析する方法

Bochmann[2)] および Stenning[34)] は，PASCAL-like のプログラム言語により記述されたデータ転送プロトコルについて，通信実行時における状態変数（送受信シーケンス番号など）間に成立つ論理関係式(不変式)の検証を行っている．各関係式の検証には並列処理プログラムの正当性の証明[1),18)]と同様，以下の様なプロトコルの実行に伴うイベント系列に関する帰納法が適用される．

〔帰納原理〕

x, x'：状態ベクトル

x_0：状態ベクトルの初期値

P：x に関する検証すべき不変式

→：イベントの発生に伴う状態変数の変化（状態遷移）

としたとき，

$$P(x_0) \wedge ((\forall x, x')[P(x) \wedge x \rightarrow x'] \Rightarrow P(x'))$$

このときPはプロトコル実行中のすべての時点で成立つ．

ただし，この方法で検証される内容はプロトコルにもとづく通信が“実行されること”を前提とした上での正当性（すなわち，部分正当性）の証明であり，実行されるかどうかの検証は別途，他の方法で行う必要がある．

（6） 状態遷移と処理ロジックの解析を併せた方法

Bochmann[4)] は，1つのプロトコルを状態遷移図による記述部分と手続き的言語による処理手順記述部分とに分けて記述し，(1)と(5)を併用した様な方法によって，プロトコルが実行され，かつ実行結果が正しいこと（全正当性）を証明している．また，41)では PDC 法により記述されたプロトコルについて同様な検証を行っている．

（7） その他の方法

Brand[7)] は，ALGOL 型の文法と，それに同期処理を含むいくつかの命令セットを追加した言語により記述されたプロトコルについて，仕様の各ステートメントのシミュレーション的な記号実行 (symbolic execution) による検証を提案している．それは，系の初期状態から出発して処理シーケンスを網羅的に生成 (proof

tree の生成）し，特定の制御点（例えばあるステートメントの実行直前）に付与された関係式 (assertions) を検証するものであり機械的に実行することが可能である．

Gouda[10]は，プロトコルの中で，特にメッセージの送受信シーケンスのメカニズムのみに着目した SR マシンと呼ばれる概念を用いて，階層化された複数個の PM が相互動作する場合の情報送受信シーケンスの論理チェックを行う方法を提案している．Kimura[19]は，プロトコルの概念を純粋に抽象代数系として定義し，その上で生じる数学的諸問題の可解性を取扱っている．すなわち，プロトコルを1つの状態集合（セマフォー値の集合）上の1対1部分変換半群（セマフォー操作の集合）の各要素に具体的イベントを対応付けたものとして定義し，デッドロックの決定可能性問題等をフォーマルに論じている．

3.3 検証法のまとめ

前節で述べた各種の検証法とその特徴は，手法の基本的な原理から，そのベースとなった記述法に対応して表-4 の様に3つに分類することができる．しかしながら，これまで知られている手法はいずれも未だ実用的なプロトコルに対し十分に有効とは言い難い様である．すなわち，合成状態図を用いる方法では，プロトコルの複雑化によって合成状態数がたとえ計算機によっても処理不可能なほどに増大してしまい，また，処理ロジックを解析する方法では，プログラムの正当性証明と同様，フォーマルな証明法としてあまり有効なものはない現状である．今後プロトコルの検証法が有用なソフトウェア工学として発展するためには次の様な課題に対する研究が必要であろう．

表-4　プロトコル検証法の分類

検証手法	対応する記述法	検証内容	特徴評価
状態遷移をもとにした方法	系の状態遷移に着目した記述法（状態遷移図，表など）	・デッドロックの可能性 ・所期の（又は不当な）ループの存在性 ・特定の状態への到達可能性	・通信系全体としての動作可能性のチェックが容易(○) ・検証の自動化が容易(○) ・処理ロジックの正しさの検証には不向き(×)
論理関係式の証明を行う方法	処理ロジックの記述（高級言語など）	・プロトコル実行中に成立つ状態変数間の関係式の検証	・パラメトリックな処理手順の検証に適す(○) ・検証自動化は困難(×) ・系の動作可能性の検証には不適(×)
上記2者を併用した方法	状態遷移と処理手順の両方を記述する方法（PDC 法など）	・上記2者を併せた内容	・上記2者を併せた様な利害得失を持つが，プロトコルの規定内容にも依存する

○：長所，×：欠点

① 検証手順の自動化

一般に，系の状態遷移に着目した方法では比較的検証手順の自動化は容易なのに対し，処理ロジックの解析をもとに論理式の証明を行う方法では，当面は自動化の望みは薄いと考えられる．それはまた，今後におけるプログラムの自動証明技術の発展にも依存するものであろう．

② 検証の網羅性

プロトコルでは，2.1 にも述べた様に，非同期的に種々のタイミングで発生する例外事象に対する動作規定がなされているため，その検証を完全なものとするためには可能性のあるあらゆる例外事象の発生を考慮することが必要となる．これを網羅的に行う様な統一的手法の開発が望まれる．

③ プロトコルが階層構成をなす場合の検証法

データ通信網アーキテクチャにもとづくプロトコルは階層構成をなすことが多く，この場合 PM は他ノードの同一階層エンティティにある PM との間のプロトコルの実行だけでなく，同一ノードにおける他階層の PM との相互動作（インタフェース）も行うこととなる．既存の検証法の多くはこの様な考慮がなされておらず今後の検討が必要である．

④ 実用的検証法

あらゆる種類のプロトコルに対して 100% の検証能力を有する様な手法の開発は将来とも非常に難しいと考えられる．そこで，ある意味で②とは逆の立場となるが，各種のプロトコルについて，実用上有意とみなされ，しかも検証可能な内容は何かを明らかにするとともにその検証手順を検討することも重要であろう．

⑤ 検証が容易なプロトコル記述法の検討

プロトコルの検証法は，いずれもそのベースとなる記述法に対応している．したがって，記述法の制定に当り，記述や解読の容易性を犠牲にしても，例えば，機械処理による論理チェックや記述する過程で規定の網羅性チェックができるといった検証の容易性を重点的に考慮することも必要である．

⑥ 検証に係わる種々の決定問題の解法

19) や 27) に述べられている様なデッドロックやプロトコルの実行可能性等に関する決定可能性問題も，検証アルゴリズム探索に当っての前提条件を与えるものとして重要である．

4. あ と が き

本稿では主要なプロトコル記述法と検証法について既存技術の概要を紹介し，それらの適用に当って考慮すべき点を明らかにするとともに，今後の動向について若干の考察を行った．プロトコル記述法の中には，既に実用的プロトコルにも適用されているものが多いのに対し，検証法は未だごく限られた範囲のプロトコルにしか適用されておらず十分な応用をみていない様である．一方，本稿では触れなかったが，最近，プログラムの仕様記述法として，抽象データ型の仕様記述法をはじめとした形式的仕様記述法，検証法に関する多くの研究がなされている[42]．プロトコルも最終的にはプログラムとして実現されるべきものであり，ある意味で一種のプログラム仕様ともみなすことができ，したがって，これらの手法の適用も可能と考えられる．いずれにせよ，この分野の研究は未だ日が浅く今後の発展が望まれるところである．

参 考 文 献

1) Ashcroft, E. A.: Proving Assertions about Parallel Programs, J. Comp. Sys. Sci., 10, 1 (1975).
2) Bochmann, G. V.: Logical Verification and Implementation of Protocols, Proc. Fourth Data Communication Symposium (1975).
3) Bochmann, G. V.: Communication Protocols and Error Recovery Procedures, Proc. ACM SIGCOMM/SIGOPS Interprocess Communications Workshop (1975).
4) Bochmann, G. V.: A Unified Method for the Specification and Verification of Protocols, Proc. IFIP Congress (1977).
5) Bochmann, G. V.: A Formalized Specification of HDLC Class of Procedures, National Telecommunications Conference in Los Angels (1977).
6) Bochmann, G. V.: Finite State Description of Communication Protocols, Symposium on Computer Network Protocols (IFIP/ACM) Liege (1978).
7) Brand, D.: Verification of Protocols Using Symbolic Execution, Symposium on Computer Network Protocols (IFIP/ACM) Liege (1978).
8) Danthine, A. S.: An Axiomatic Description of the Transport Protocol of CYCLADES, Professional Conference on Computer Networks and Teleprocessing (1976).
9) Danthine, A. S. et al.: Specification and Verification of End-to-End Protocols, Proc. Fourth International Conference on Computer Communication (1978).
10) Gouda, M. G. et al.: On the Modelling, Analysis and Design of Protocols—a Special Class of Software Structures, Proc. Second International Conference on Software Engineering (1976).
11) Gouda, M. G. et al.: Protocol Machines——A Concise Formal Model and its Automatic Implementation, Proc. Third International Conference on Computer Communication (1976).
12) Harangozo, J.: An Approach to Describing a Data Link Level Protocol with a Formal Language, Proc. Fifth Data Communication Symposium (1977).
13) ISO/TC 97/SC 16 資料 N 117: Reference Model of Open Systems Architecture (Version 3), (1978).
14) 井手口, 水野, 松永：通信プロトコルの抽象機械化とその応用, 信学技報 EC 77-30 (1977).
15) IBM: Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic, 1 st edition (1976), 2 nd edition (1978).
16) 河岡, 友永, 高橋：プロトコルの記述方法と検証方法について, ソフトウェア工学研資 ST 8-5 (1978).
17) 川井, 周東, 浅野, 野村：通信制御手順の検証手法に関する一検討, コンピュータネットワーク研資 CN 19-5 (1979).
18) Keller, R. M.: Formal Verification of Parallel Programs, CACM Vol. 19, No. 7 (1976).
19) Kimura, T.: An Algebraic System for Process Structuring and Interprocess Communication, Proc. Eigth ACM Symposium on Theory of Computing (1976).
20) Le Moli, G.: Theory of Colloquies, First European Workshop on Computer Networks ARLES (1973).
21) Merlin, P. M.: A Study of the recoverability of Computing Systems, Ph. D. thesis Univ. of California (1974).
22) Merlin, P. M. et al.: Recoverability of Modular Systems, Proc. ACM SIGCOMM/SIGOPS Interprocess Communications Workshop (1975).
23) Merlin, P. M.: A Methodology for the Design and Implementation of Communication Protocols, IEEE Tran. Comm. (Jun. 1976).
24) Merlin, P. M. et al.: Recoverability of Communication Protocols——Implications of a Theoretical Study, IEEE Tran. Comm. (Sept. 1976).
25) 宮沢, 阿部, 友永：コンピュータ・ネットワークにおけるプロトコルの一記述法, 信学・情報・全大 (1977).
26) 森, 荒木, 谷口, 都倉, 嵩：タイムペトリネットに関する判定問題——通信制御手順の検証への一考察——, 信学論 (D), Vol. J 60-D, No. 10

(1977).
27) 森，谷口，藤井，嵩：配列を用いた平行処理プログラムの検証——抽象プロトコルの不変式——LA シンポジウム (Jul. 1978).
28) 森野，高橋，田島，苗村：ハイレベル・データリンク制御手順の規定方法について，第17情学全大 (1976).
29) 高橋，森野，田島，苗村：ハイレベル・データリンク制御手順の規定とその評価，コンピュータ・ネットワーク研資，CN 8-1 (1976).
30) Postel, J.B.: A Graph Model Analysis of Computer Communication Protocols, Ph. D, thesis, Univ. of California (1974).
31) Postel, J.B. et al.: Graph Modelling of Computer Communications Protocols, Proc. Fifth TEXAS Conference on Computing Systems (1976).
32) Rusbridge, R.E. et al.: Formal Representation of Protocols for Computer Networks, AERE Harwell, Oxfordshire (1974).
33) Schreiber, F.A.: Some Linguistical Problems about Colloquies, ACM Computer Communications Review, Vol. 5, No. 4 (1975).
34) Stenning, N.V.: A Data Transfer Protocol, Computer Networks 1 (1976).
35) Sunshine, C.A.: Interprocess Communication Protocols for Computer Networks, Ph. D. thesis, Stanford Univ. (1975).
36) Sunshine, C.A.: Survey of Communication Protocol Verification Techniques, RAND Corp. Rept. P-5725 (Sept. 1975).
37) Sunshine, C.A.: Survey of Protocol Definition and Verification Techniques, Symposium on Computer Network Protocols, Liege, (1978).
38) 田中：プロトコル(通信規約)，信学誌，61，10，p. 1105 (1978).
39) 戸田，中田：データ通信網アーキテクチャ (DCNA) の基本概念，情報処理，Vol. 20, No. 2 (1979).
40) 友永，阿部，宮沢，河岡：プロトコル記述法の一検討，コンピュータ・ネットワーク研資 CN 12-2 (1977).
41) 友永：プロトコルの状態遷移と処理手順の検証法，第19回情学全大 (1978).
42) 鳥居，二木，真野：プログラミング方法論の展望，情報処理，Vol. 21, No. 1 (1979).
43) 浦野，小野，鈴木：プロトコルマシンにおけるプロトコルとその記述について，第18回情学全大 (1977).
44) West, C.H.: An Automated Techniques of Communications Protocol Validation, International Conference on Communications (1977).
45) West, C.H.: et al.: Automated Validation of a Communication Protocol: the CCITTX. 21 Recommendation, IBM J. Res. Develop., Vol. 22, No. 1 (1978).
46) West, C.H.: General Technique for Communications Protocol Validation, IBM J. Res. Develop., Vol. 22, No. 4 (1978).
47) 八重樫，高橋，野口，川合，海老原：通信制御プロトコルの記述と実現化に関する一考察，第19回情学全大 (1978).
48) 八重樫，野口，高橋，海老原，川合：通信制御におけるプロトコルの記述とプログラム化に関する検討，コンピュータ・ネットワーク研資 CN 17-2 (1978).
49) Zafiropulo, P.: A New Approach to Protocol Validation, International Conference on Communications (1977).
50) Zafiropulo, P.: Protocol Validation by Duologue-Matrix Analysis, IEEE Tran. Comm. (Aug. 1978).

(昭和54年5月8日受付)