プロトコル記述法の一検討

友永 充宏，阿部 豊彦，宮沢 正幸，河岡 司
（日本電信電話公社 横須賀電気通信研究所）

1．まえがき

データ通信網アーキテクチャにおいてはプロトコルの規定が主要な位置を占めている。したがって、その規定内容をドキュメントとして表現するための方法（ここではプロトコルの記述法と呼ぶ）が重要となる。プロトコルの記述法には一般に次の具備条件が必要である。

- 記述が容易なこと
- 記述されたものが直観的にかつ実システムとの対応においてわかり易いこと
- 規定内容が厳密であること
- プロトコルの検証に有効であること
- 例外処理等の網羅的規定が可能なこと
- 記述されたものがプロトコルのインプリメントに有効なこと

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，全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ては、その記述の目的に応じて特に重要な具備条件と、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。本稿では記述の目的を分類し、それに対する適合性の観点から既存の記述法の評価を行い、更に通信の同期動作の解り易い表現という立場から検討したプロトコルの記述法（PWC法）について述べる。

2．プロトコルの記述法

プロトコルは異る2点間の情報送受信に係わる通信規約であり，これにしたがって通信を実行する抽象的な装置をプロトコルマシン（以下PMと略す）と呼ぶ。図1．に示すようにPMは一種の有限オートマトン（一般には有限状態）とみなすことができる。

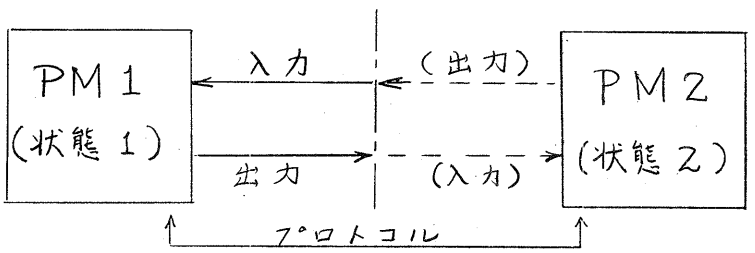


図1．プロトコルマシン（PM）

したがって、プロトコルを規定することは1つの有限オートマトンを定めることと相当し、記述法はその表現方法に対応する。一般に、プロトコルを記述する場合その目的はプロトコルの概要把握のため，インプリメントのためなどが考えられ一様ではない。また、記述の目的により記述内容の詳細度（レベル）及び記述に対する要求条件も異ってくる。本検討では、記述の目的に応じ記述内容の詳細度として表1．の様なレベルを設定した。以下、本稿では特にプロトコル仕様の広報を目的としたレベル1の記述法について述べる。

表1. プロトコル仕様記述のレベル

詳細度	記述の目的，内容	記述法への主な要求条件	記述方法
レベル0	プロトコルの概要を把握するために基本概念，各PMの機能概要を記述	記述の容易さ 解読の容易さ	自然言語 図表
レベル1	レベル0とレベル2の中間にあって，PM対向のプロトコル中核機能を明確に記述（プロトコルの広報が目的）	記述の容易さ 解読の容易さ 記述の厳密さ	自然言語や図表による場合もあるが特定の記述手法が用いられることもある
レベル2	プロトコルをインプリメントする場合の参照マニュアルとするため，全ての例外処理規定を含めて網羅的に記述	規定の網羅性 記述の厳密さ インプリメントへの有用性	特定の記述手法が用いられることが多い

3. 各種記述法とその評価

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，その概要と特徴を述べ，次にそれらをレベル1記述への適用性の観点から評価する。

3.1 各記述法の概要とその特徴

(i) Petri-Net

Petri-Netは並列処理を表現するグラフモデルであり，それはP節点（"○"で表わす）とT節点（"|"又は"—"で表わす）と呼ばれる2種類の節点を有向枝（アーク）で結合した有向グラフである。ただし，P節点及びT節点相互間は直接結合されない。P節点にはトークン（"●"で表わす）を複数個配置できる。Petri-Netの1つのT節点に関してその各々の入力P節点（そのT節点を終端とするようなアークの始端の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（firable）であるという。発火可能なT節点が発火（fire）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節点を始端とするようなアークの終端のP節点）にトークンを1個づつ付け加える。発火可能でないT節点は発火することはない。一般に，T節点は1つの事象に対応し，その発火は事象の発生に対応する。したがって，1つのT節点の各入力P節点はそのT節点に対応する事象が発生するための条件（そこにトークンが存在する場合，その条件が満たされている）を構成するものと考えられる。

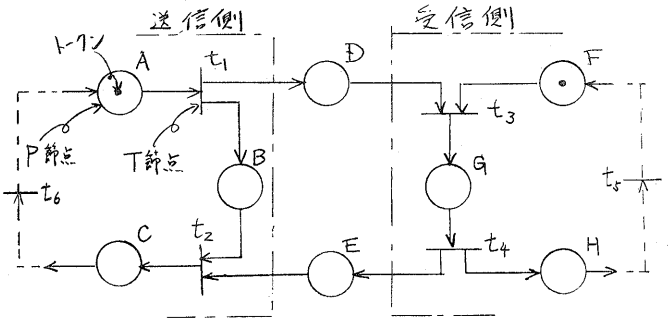


図2. Petri-Net（メッセージ送受信モデル）

図2.はPetri-Netにより簡単なプロトコルを記述した例である。

1つのPetri-Netにおいて、ある時点での全てのP節点に対するトークンの配置をマーキングと呼ぶが、これは一般に 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という。これは一種の状態遷移図であり、それを追跡することにより、そのPetri-Netで表現された並行処理系が異常状態（例えばデッドロックなど）に陥ることがないかなどの検証に用いることができる。図2.のPetri-Netに対するトークンマシンの例を図3.に示す。

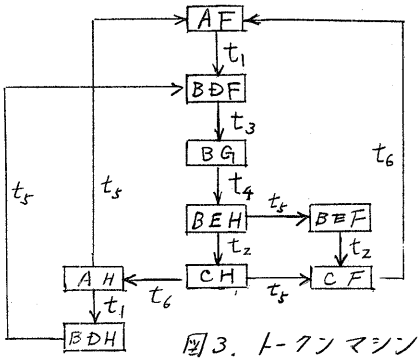


図3. トークンマシン

Petri-Netの特徴は、通信に係わる同期動作を厳密で明解に記述できる点にある。ただし、記述に熟練を要し、例外処理等の網羅的規定が困難であり、また記述対象の範囲に制約が多いという欠点を持つ。トークンマシンによる検証は単純なモデルに対しては有効であるが、実システムでは遷移図が複雑になりあまり実用的な検証法とは言い難い。

(ii) IBM.SNA.FAP（[10]）の方法

IBM.SNA.FAPでは、通信の同期動作をFSM（Finite State Machine）と呼ばれる状態遷移図により記述し、非同期的な情報、制御の流れはBD（Block Diagram）というフローチャートに似た図式を用いて記述している。FSMは一種の有限オートマトンの表現モデルであり、図4.の様式にしたがって記述される。またBDはFSMを表わすブロックと情報及び制御の分岐を表わす判断ブロック、及び処理を表わすブロックを相互にアークで結合したものである。BDでは通信に係わる同期動作はFSM名のみから成るブラックボックスとして記述される。（即ち、FSMとBDとの記述は階層をなしている）

FSMの記法は非常に整然と規則的に描かれる点で解り易く、BDの併用により同期動作と処理動作の明確な分離が可能となり、例外処理の規定や、PM間インタフェースの記述も容易である。しかしながら、同期的に通信するPMを個別に記述するためシステム全体としての同期並行動作の直観的な把握は難しい。

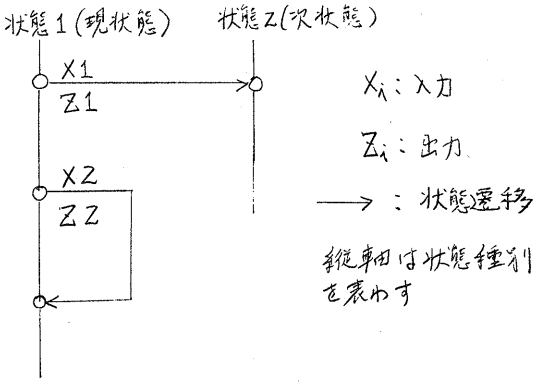


図4. FSMの記述方法

(iii) 状態遷移表

状態遷移表は、現状態と状態遷移要因の全ゆる組合せに対して、とるべき制御動作と遷移先状態とを一覧表の形に記述するもので、図5.の様なマトリクス形式のものや、図6.の様にシーケンシャルな形式のものがある。状態遷移表には全ての状態と全ての状態遷移要因との組合せが現われるため規定の網羅性にすぐれており、規定漏れの検証に対して有効である。ただし、その反面、正常系の

動作がとらえにくくなるという欠点をもっている。

遷移要因＼状態	S_1	S_2	----	S_m	----
t_1					
⋮					
t_n				NS / A	
⋮					

NS：次状態
A：とるべき動作

図5. 状態遷移表（マトリクス形式）

状態	遷移要因	次状態	とるべき動作
S_1	t_1	NS_{11}	A_{11}
	t_2	NS_{12}	A_{12}
	⋮	⋮	⋮
S_2	t_1	NS_{21}	A_{21}
	⋮	⋮	⋮

図6. 状態遷移表（シーケンシャル形式）

(iv) 状態遷移図

状態遷移図は基本的には状態遷移表をより直観的にとらえることができるように図式化したものである。しかしながら、状態遷移図はスペースファクタや図の見易さを図るという立場から、状態遷移表の様な網羅的規定は難しい。図7.に状態遷移図の一例を示す。状態遷移図は一般に通信するPMの動作を仕引に記述するものであるが、Rusbridge[7]あるいはSunshine[9]は通信中のシステムの状態を互いに通信する2つのPMの状態と、その間を転送中のデータとを合成したものとして定義し、その遷移の図式化した表現（合成状態図）を用いてプロトコルの検証を行う方法を提案している。図7.のプロトコルに対する合成状態図を図8.に示す。しかしながら、この方法はPetri-Netにおけるトークンマシンと同様、単純なプロトコルの検証には有効であるがプロトコルの複雑化とともに状態図が煩雑化するという問題点がある。

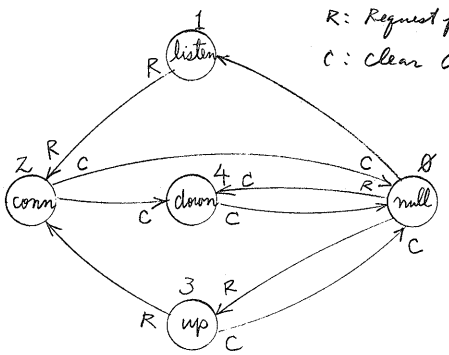


図7. 状態遷移図の例
（コネクション設定，解放プロトコル）

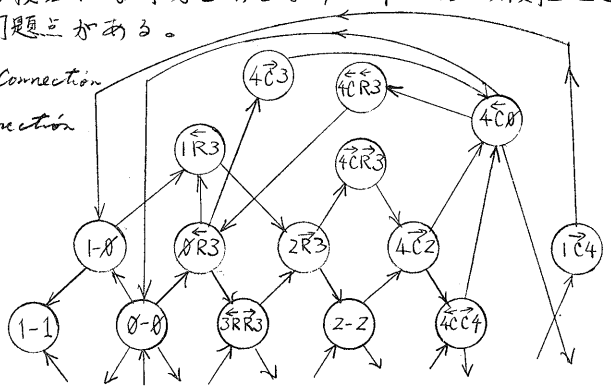


図8. 図7.に対する合成状態図
（対称となる部分は省いてある）

(v) プロトコルグラフ

Manning[3]はPMの通信動作を、4種の基本的動作によりフローチャート的に記述する方法（プロトコルグラフ）を提案している。プロトコルグラフの構成要素を図9.に示す。この方法の特徴はPM動作の直観的把握が容易な点と、オペレーションノードにより実システム動作との対応がとり易い点にある。しかしながら、単独PM

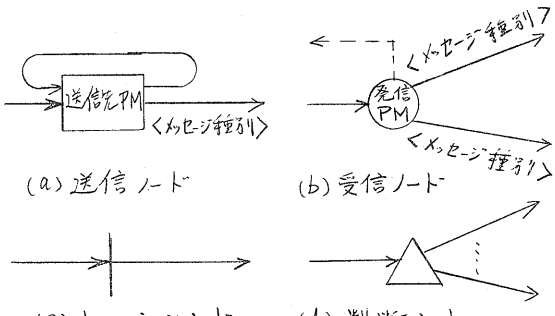


図9. プロトコルグラフの構成要素

の仕別規定であるため、通信の同期動作はPetri-Netの様に通信する2つのPMを一体の系として記述する方法に比べるとわかりにくい。

(V) コロキー

Le Moli [5]はコロキー(Colloquy)という概念でプロトコルの形式的表現を行っている。コロキーとは2つのインタロキュタ(Interlocutor:PMに相当)が一定の規約のもとに情報の送受信を行い、また外部(インタロキュタのユーザ)との間でコマンドやテキストの授受を行うことである。コロキーにおけるインタロキュタの状態は相手インタロキュタからの入力情報と外部ユーザのコマンドによって変化する1つの論理マトリクスによって表現される。1つの状態における情報受信時のインタロキュタの動作はこのマトリクスと受信情報種別とをもとにマトリクス演算によって機械的に決定される。その点でこの方式はインプリメントに有効な記述であるが、状態変数の増大やその値域の拡大とともに表現が複雑となりプロトコルの直観的理解は難しい。

(VI) 高級言語による方法

汎用プログラミング言語に近い言語を用いてプロトコルの記述を試みる例として、例えばDanthine [2]は通信をコロキーの概念により形式化してとらえ、それを構成するインタロキュタの内部メカニズムをALGOL的なif-then-else節を用いて記述する方法を提案し、それを実際にCYCLADESのトランスポートプロトコルの記述に適用している。また、Stenning [8]はプログラミング言語PASCALを用いてホスト/ホストレベルプロトコルを記述し、その形式化の下でプロトコル属性の検証を行っている。これらの方法の特徴は、プロトコル仕様の記述からインプリメントのための記述への移行がスムーズに行えることと、記述する過程で規定漏れの発見が容易なことである。一方、記述されたものを見てもプロトコルの理解が難しいこと、処理シーケンスの一意な記述などのため過剰規定に陥り易いといった問題点がある。

以上、既存の主な記述法の概要とその特徴を述べたが、これらは大別して通信に関与する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けられる。ここに掲げた記述法ではPetri-Net及び合成状態図は前者に属し、それ以外の方法は後者に属する。一般に前者の方法では通信に係わる同期動作の明確な記述はできるが、基本的には通信相手と同期した動作しか記述できないため非同期的な例外処理規定が困難である。これに対し後者の方法は例外処理規定の網羅性の点ですぐれているが、通信システム全体としての処理の流れが把握しにくいという欠点をもつ。

一方、別の観点からの分類として、状態遷移図(表)系統の記述法と、実システムの制御動作に即した方法とに分けることができる。前者の方法では、情報の送受信によってPMの状態がどの様に変化するかをとらえ易いが、状態遷移要因発生時にPMがとるべき制御動作が不明確になり易い。一方、後者は処理機能を明確に記述できるが、処理シーケンスの記述などのためプロトコルの過剰規定に陥りがちである。前掲の記述法の中でプロトコルグラフ、高級言語による方法は後者の例であり、Petri-Netも後者に近い方法である。

3.2 レベル1記述への適用性

前節に述べた様に既存のプロトコル記述法にはそれぞれ利害得失があり、全ゆる要求条件に適合するものは見当らない。プロトコルのレベル1記述の目的からはプロトコル制御動作の直観的理解の容易性が重要であり、特に次の2点が記述法の具備すべき必要条件と考えられる。

① 通信の同期動作を直観的にわかり易く表現できること。

② 記述されたものを実システム動作に対応させることが容易であること。

前節で比較した各種記述法の中で、先ず条件①の点では個々のPMを個別に規定する方法よりも互いに通信する2つのPMを一体の系として記述する方法がより有効であると考えられる。また、条件②の点では状態遷移図系統の方法よりも実システムの制御動作に重点をおいた記述法が適当である。前掲の記述法の中で通信する2つのPMを一体の系として表現し、しかも実システムの制御動作に重点をおいた記述法はPetri-Netによる方法のみであるがこれにも次の様な問題点が存在する。

(1) なじみの薄い特殊な記法を用いるため読みにくく、また記述に際しかなり熟練を要すること。

(2) 非同期的例外処理規定が難しい。

(3) 数値処理や論理判断動作の記述が難しく図の煩雑化をまぬき易い。

(4) (1)～(3)の理由で実システム動作との対応は必ずしも容易でない。

これらの問題の中で(2)は本来レベル1記述での要求度は比較的小さいと考えられる。しかしながら、(1),(3)はレベル1記述の要求条件であるプロトコルの直観的理解の容易性を著しくそこなうため、レベル1記述にPetri-Netをそのまま適用することは好ましくない。

4. PDC法

PDC (Protocal Description Chart) 法は、プロトコルのレベル1記述を目的とした記述法であり、通信の同期動作を直観的にわかり易く表現でき、また、実システム動作との対応が容易な方法である。

4.1 PDCの定義

PDC法は通信する2つのPMを一体の系として記述する方法であり、Petri-Netの考えをもとにし、その長所を生かすとともに欠点をカバーするための改良を行った方法である。

(1) PDCの構成

PDCとは表2.(次頁)に掲げる各節点間を、表3.(次頁)の結合法則にしたがってアークにより結合し、いくつかのP節点又はQ節点にトークンを配置した有向グラフである。

(2) PDC動作上の規約

Petri-Netではシステムの動作はT節点の発火によるP節点間のトークン移動による状態遷移として表現される。PDCでもPetri-Netと同じく動作の完了、又は事象の発生をT節点の発火に対応させ、それによるトークン移動でシステムの動作を表現する。ただし、Petri-Netの場合とはトークンの配置、除去の規則を若干異にする。即ち、PDCではトークン移動に関し以下の規則を設ける。

表2. PDCの構成要素

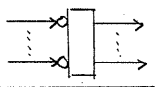
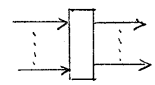
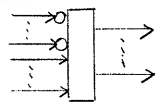
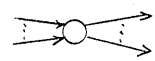
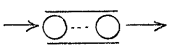
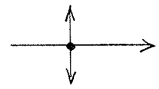
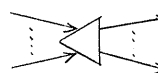
名称		記号	実システムとの対応	対応するPetri-Netの要素
E節点	ANDモード		制御動作の実行又は事象の発生	T節点
	ORモード		同上	なし
	AND/OR混在モード		同上	なし
P節点			制御動作の実行中又は実行待ち	P節点
Q節点			情報の待ち行列(キュー)	なし
S節点			制御又はデータの複数方向への同時移行	なし
L節点			論理判断による制御又はデータの分岐	なし

表3. 節点間の結合条件(アークの経路)

(アークの始端) \ 節点	E	P	Q	S	L
E	○	○	○	○	○
P	○	×	×	×	×
Q	○	×	×	×	×
S	○	○	○	○	○
L	○	○	○	○	○

○：アークによる直接結合可能
×：〃　〃　不可

E節点＝Executive node
P　〃　＝Place　〃
Q　〃　＝Queue　〃
S　〃　＝Split　〃
L　〃　＝Logical　〃

・E節点の発火条件

表現の簡単化のため次の入力条件なる命題を定義しておく。

入力条件：入力アークの始端がP又はQ節点の場合にはその節点にトークンが1個以上配置されており、L，S又はE節点の場合にはそのアーク上にトークンが1個以上配置されている。

この条件を用いてE節点の発火条件は次の様に定義される。

(i) ANDモードの場合

全ての入力アークについて入力条件が成立っていること。

(ii) ORモードの場合

少くとも1つの入力アークについて入力条件が成立っていること。

(iii) AND/OR混在モードの場合

ANDモードの全ての入力アークについて入力条件が成立ち、ORモードの入力アークの中、少くとも1つについて入力条件が成立っていること。

・E節点の発火によるトークンの除去

PDCでのE節点の発火によるトークン除去は各モードにより次の通りである。即ち、ANDモードで入力しているP，Q節点及びトークンを保有している入力アークについては、それらの各々からトークンを1個づつ除去し、ORモードで入力しているものからは何れか1つからトークンを1個除去する。

・E節点の発火によるトークンの配置

PDCでのE節点の発火によるトークンの配置は以下の通りである。即ち、E節点が発火すると、その各出力アーク上にトークンを1個づつ付け加える。

なお、アーク上に配置されたトークンはその終端の節点がE節点であれば、その節点が発火するまで留まるが、それ以外の節点の場合には、直ちにその終端の節点に移動する。更に、各節点に配置されたトークンは次の規則で移動する。

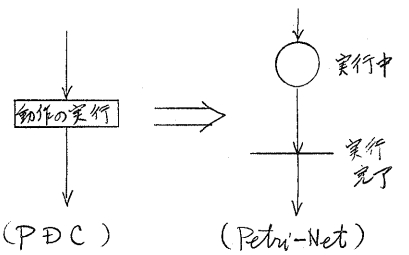
(i) P節点：その出力E節点の発火まで留まる。
(ii) Q節点：出力E節点の発火まで留まり，E節点の発火後、定められたキューイングアルゴリズムにより除去される。
(iii) L節点：直ちに、その論理判断結果にしたがって、唯一つの出力アーク上に移動する。
(iv) S節点：直ちに、その全ての出力アーク上に分離して移動する。

4.2 Petri-Netによる記述との特徴比較

PDCはPetri-Netをベースとした記述法であるが、プロトコルのレベル1記述に適応させるため，いくつかの改良を加えており、それは本来のPetri-Netとは異るものである。以下にPetri-Netの改良という立場からPDCの特徴を述べる。

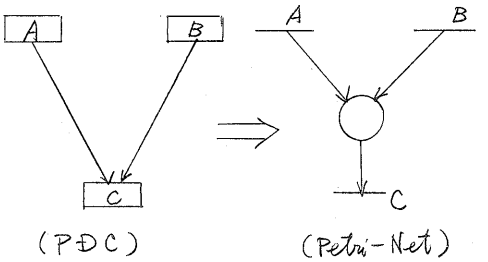
①T節点とP節点の機能の一部を包含するE節点の導入

Petri-NetのT節点は事象の発生又はある制御動作の完了に対応し、その入力P節点はそれぞれ事象の発生待ち又は制御動作の実行中に対応する。しかし、1つの制御動作をその実行中と実行完了とに分離すると実システム動作との対応において不自然な場合がある。そこで、その様な場合には動作の実行中と実行完了とを併せて1つのE節点で表現することにする。(右図)



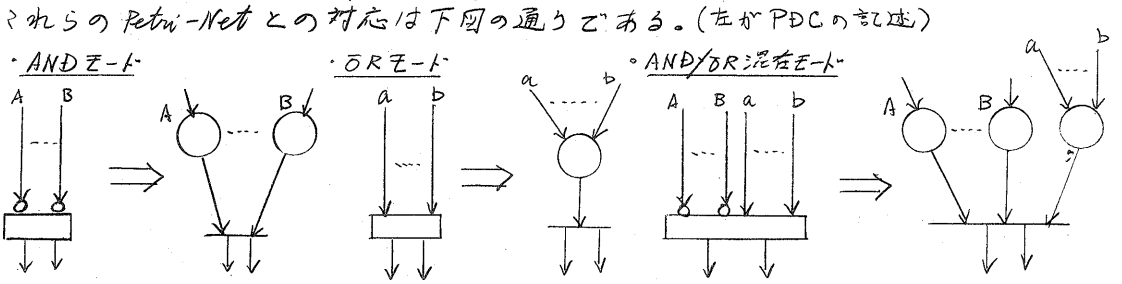
②アークのトークン保持機能

PDCではシステムのプロトコル制御動作の把握に際し、特に同期動作を強調する必要のない場合には図の簡明化を図るため，P節点を除去することを許している。そのため，同期動作表現の必要上アークもトークンを保持し得るものとしている。(右図)



③AND/ORモードE節点の導入

PDCではE節点にAND，OR及びAND/OR混在の3種のモードを設けることにより多様な同期の条件を直観的にわかり易く表現できるようにしている。それらのPetri-Netとの対応は下図の通りである。(左がPDCの記述)



④ L節点の導入

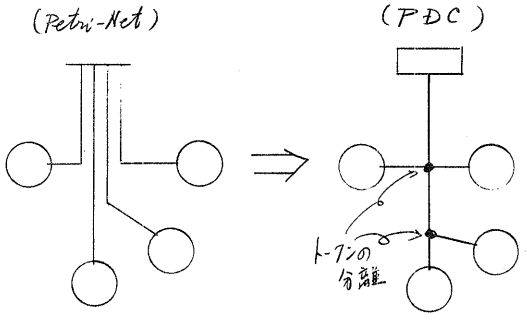
Petri-Netでは論理判断による条件分岐を記述しようとすればP節点を使用することになるが、その場合、本来のP節点の意味（即ち、同期をとるためのトークン供給）との区別が不明確となる。そこで、PDCではL節点を導入し、それに論理判断（例えば状態変数値の判断など）のみを行いトークンを振り分ける機能を持たせている。

⑤ Q節点の導入

Petri-Netでは同一P節点に配置されるトークン相互間の識別はできない。しかしながら、実システムとの対応上トークンをデータとみなしそのキューイング及びデキューイングの動作を記述したい場合がある。そこで、PDCでは、Q節点により与えられたキューイングアルゴリズムをもつトークンのキューを表現できるようにしている。

⑥ S節点の導入

Petri-NetではT節点の出力アーク数と出力P節点の数とは等しく、したがって、アークの数が多くその長さが長いと図を煩雑にする。PDCでは、1つのトークンが複数個に分離するようなS節点を導入して記述の簡略化を図っている。（右図）



4.3 PDCの記述例と評価

図10.にPDCの適用例としてウィンドウ方式（例えば[1]）によるフロー制御プロトコルの記述例を示す。

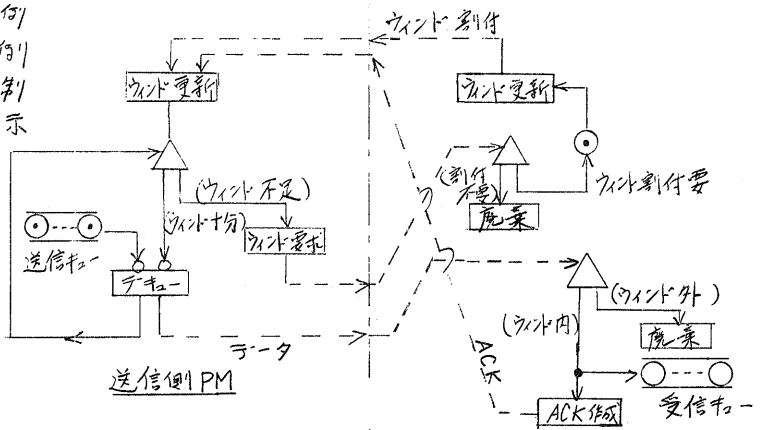


図10. PDCの記述例

本方式は、通信する2つのPMを一体の系として記述する方法であり、Petri-Netの考えをとり入れており、特に通信に係わる同期動作の把握が容易である。さらに、Petri-Netと比較して次の様な利点を持っている。

① E, Q及びL節点の導入により実システム動作との対応が容易である。

② E節点の多種モードの導入，一部のP節点の除去及びS節点の導入により記述が容易になり、記述されたものがわかり易くなっている。

③ L節点の導入により多様な処理規定に対処可能である。

以上述べた様な改良が得られたことから、本方式は、Petri-Netの長所を生かし、その欠点を十分にカバーできると考えられ、プロトコルのレベル1記述に対して適した記述法といえる。

5. むすび

本稿では、既存のプロトコル記述法についてその概要を比較し、プロトコルのレベル1記述に適した一方法を述べた。本方式の特徴は通信システムのプロトコル制御動作（特に通信の同期動作）を直観的にわかり易く記述できる点にある。更に、E節点の論理操作又はL節点の論理判断の名内容が複雑で図の煩雑化が予想される場合には、この部分を別途詳細なPDCとして記述するか、或いは別表としてマッピングをとるなどの階層的記述が考えられる。

今後、コンピュータ・ネットワーク・アーキテクチャの標準化に際してプロトコルの記述方法が重要な課題となるであろう。このためにはアーキテクチャ全体の記述、及びプロトコルのインプリメントをねらいとするレベル2記述等が必要であり、これらに対するPDC法の適用性について検討を進めている。

謝辞

本検討を進めるに当り、御指導いただいた当所制御方式研究室の大島室長はじめ関係各位に厚く感謝いたします。

[参考文献]

(1) Cerf, V., "Internetwork End-to-End Protocol," INWG GN. #96 July 1975.
(2) Danthine, A.S., "An Axiomatic Description of the Transport Protocol of CYCLADES", Professional Conference on Computer Networks and Teleprocessing Mar. 1976.
(3) Gouda, M.G. & Manning, E.G., "Protocol Machines: A Concise Formal Model and its Automatic Implementation," Proc. ICCC 1976.
(4) Merlin, P.M., "A Study of the Recoverability of Computing Systems," Ph.D. diss. Univ of California 1974.
(5) Le Moli, G., "A Theory of Colloquies", First European Workshop on Computer Networks ARLES, April-May 1973.
(6) 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", 昭51 情学全大 No. 325, Nov. 1976.
(7) Rusbridge, R.E. & Langsford, A., "Formal representation of protocals for computer networks", AERE Harwell, Oxfordshire Dec. 1974
(8) Sterning, N.V., "A Data Transfer Protocol", Computer Networks 1, 1976.
(9) Sunshine, C.A., "Interprocess Communication Protocols for Computer Networks," Stanford Univ. Technical Rep. #105, Dec. 1975.
(10) IBM, "Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic", IBM 1976.