解　説

IPS

ネット指向パラダイムを求めて

8. 通信ソフトウェア要求仕様化設計へのペトリネットの応用†

長谷川　晴朗††

1. まえがき

近年，交換を中心とする通信システムの高度化・多様化・高速化にともない，通信ソフトウェア開発の重要性が飛躍的に増大しており，その需要に対処するため従来から下流工程を中心に種々の開発技法が試みられている．一方，ペトリネット[1)~4)]は，非同期・並行系システムのモデル化技法としてさまざまな分野に利用されており，性能評価など通信システムにも数多く使用されている．ここでは通信ソフトウェア開発の上流工程に利用している例について述べる．

まず，本稿で使用するペトリネットの基本用語の説明を行う．次に，通信ソフトウェアの仕様記述法について述べた後，仕様の論理的な検証項目について説明する．**5.**から**7.**では，ペトリネットを通信仕様の記述と検証への利用のかかわり方に応じて三つ（記述のみ，記述と検証，検証のみ）に分類し，それぞれ例をまじえながら説明する．

2. ペトリネットの基本用語

本稿で使用する，ペトリネット（以下，PN と略す）の用語について簡単に触れる．

（1） PN を4項組，$N=(P, T, A, M)$ で表す．ここで，P, T, A, M はそれぞれプレースの集合，トランジションの集合，接続行列（行：トランジション，列：プレース，$A=A^+-A^-$; A^+ は T から P への行列，A^- はその逆），マーキングである．したがって，P, T, A はペトリネットの構造的な性質を示す．一方，M は各プレースへのトークンの配置状態を示すもので，構造を表すものではない．

（2） 発火条件・発火列…あるマーキング M_0 でトランジション t_j が発火するための条件は，j 番目のみの要素が1でそれ以外の要素がすべて0である単位列ベクトルを e_j として，$M_0 \geqq A^{-T} \cdot e_j$…(1)が成立することである．ここで A^{-T} は A^- の転置行列を意味する．また，M_0 から別のマーキング M に至る発火系列で，各トランジションの発火回数を示す発火回数列ベクトルを γ とすると，$M=M_0+A^T \cdot \gamma$…(2)が成立する．

（3） T(S)インバリアント…ある列ベクトル x 及び y が，$A^T \cdot x=0$, $A \cdot y=0$ を満足するとき，それらをそれぞれTインバリアント，Sインバリアントという．特に，ほかのTインバリアントに和分解不能なTインバリアントを初等的Tインバリアントという．(2)式で $M=M_0$ とすれば $A^T \cdot \gamma=0$ となることから，Tインバリアントは，あるマーキングから再びそのマーキングに戻る各トランジションの発火回数を示すものであることが分かる．また，(2)式の両辺に左から y^T を乗じると，$y^T \cdot M=y^T \cdot M_0+y^T \cdot A^T \cdot \gamma$ が得られる．ここでSインバリアントでは $y^T \cdot A^T=(A \cdot y)^T=0$ となることから $y^T \cdot M=y^T \cdot M_0$ が成立する．これは，あるマーキングから到達可能なすべてのマーキングにおいて，プレースごとにある重み付けしたトークンの総和が一定であることを示している．

（4） 色付きペトリネット…以上ではトークンはいわゆる無色であり，トークンに区別はなく数量だけが意味をもつものである．これに対し，トークンになんらかの識別子をもたせてトランジションで制御を行うなど，記述レベルを高めようとするものが色付きペトリネットである．この採用によりシステム全体の記述量を削減することができる．

（5） サービス…交換システムでサービスと

† Application of Petri Nets to the Specification Phase of Communication Software Development by Haruo HASEGAWA (Network Systems Development Center, Telecommunications Group, Oki Electric Industry Co., Ltd.).

†† 沖電気工業(株)通信ネットワーク事業本部ネットワークシステム開発センター

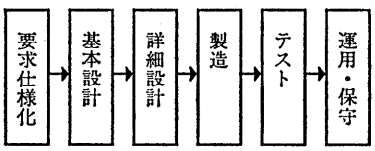
は，各リソースがアイドル状態であることを示す初期状態から最終的にその初期状態に戻る発火列である．つまり，Tインバリアントで示される発火可能ベクトルから得られるトランジションの発火系列を意味する．特に，初等的Tインバリアントに相当するサービスを素サービスという．

3. 通信ソフトウェアの仕様記述法

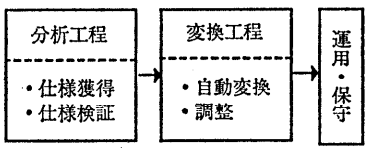
3.1 ソフトウェア開発における仕様の位置付け

通信ソフトウェアは，①ハードウェアの監視と制御，②高信頼性，③超多重処理，④実時間性，⑤複雑なサービス，⑥膨大かつ変更が多い，⑦プロセス間の相互作用といった特徴を有することから，定型処理の多い事務処理ソフトウェアと比較して著しく生産性が悪いものとなっている．したがって生産性向上には開発の下流工程だけではなく，上流工程から十分な対策をとる必要がある．

図-1 は新旧ソフトウェア開発工程を示したものである．**図-1(a)**はウォータフォールモデルと呼ばれる従来の開発工程を表している．ここでは，工程が進むにつれてユーザ要求を徐々に詳細化していく．一方，新しいソフトウェア開発のパラダイムでは，**図-1(b)**に示すように仕様を重要視し，全体を仕様作成までの分析工程と仕様作成後の変換工程とに分ける．つまり，要求仕様という非形式的仕様をもとに形式的仕様さえ設計・検証すれば，その後のソースプログラムへの変換は自動的に行えるというものである[5]．したがって，作成した仕様を検証することがきわめて重要なものとなっており，そのための記述法としてさまざまなものが提案されている．たとえば，国際的な標準機関から勧告されているものだけでも，CCITT (International Telegraph and Telephone Consultative Committee) 勧告の SDL (Functional Specification and Description Language)[6], ISO (International Organization for Standardization) 勧告の LOTOS (Language of Temporal Ordering Specification)[7] 及び Estelle[8] があり，これらをベースとしてさまざまな開発支援システムが構築されている[9),10)].



(a) 従来の開発工程



(b) 新しい開発工程パラダイム

図-1 ソフトウェアの開発工程

3.2 仕様記述法

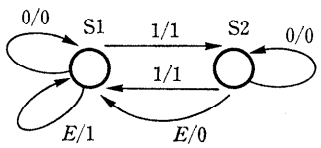
通信システムを記述するにあたっては，大きく分けて二つの方法が考えられる．一つは，端末の操作に対するシステムの状態遷移を示す有限状態マシン (FSM : Finite State Machine) に基づくものである．もう一つは，端末の操作とそれに対するシステムからの応答を時系列に表すプロセス代数に基づくものである．**3.1** で記した仕様記述言語の中，SDL と Estelle は前者に属し，LOTOS は後者の例である．ただ，現実には理解の容易性などの点から FSM に基づいて設計することが多い．

なお，FSM は完全に PN への変換が可能である．逆は成立しない．FSM を5項組，$(Q, \Sigma, \Delta, \delta, \Gamma)$ で表す．ここで，$Q, \Sigma, \Delta, \delta, \Gamma$ は，それぞれ状態の集合，入力の集合，出力の集合，現在の状態と入力から次状態を決定する関数，現在の状態と入力から出力を決定する関数である．このとき，次式のように変換することにより，PN となる[1)]．

$$P=Q\cup\Sigma\cup\Delta,\ T=\{t_{q,\sigma}\mid q\in Q,\ \sigma\in\Sigma\},$$

$$I(t_{q,\sigma})=\{q,\sigma\},\ O(t_{q,\sigma})=\{\delta(q,\sigma),\Gamma(q,\sigma)\}.$$

簡単な例を**図-2** に示す．図-2 はシリアルに入力する2進数（最後に E が入力）に奇パリティを付加する FSM を示している．初期状態は S1 であり，0 が入力する場合状態が変わらず，1 が入力するとそのたびに状態が S1 と S2 の間で変わる．最後に E が入力した時点で状態が S1 であれ



X/Y : X は入力，Y は出力

図-2 奇パリティを計算する FSM

ば1を，S2であれば0をパリティとして付加する．このFSMを上式の適用によりPNに変換したものが**図-3**である．図-3から分かるように，PNでは状態はもちろん入力や出力までプレースを使用して表現しており，逆に言うと各プレースのもつ意味がそれぞれ異なっている．

状態遷移モデルに分類されるペトリネットはFSMと比較して次のような長所を有する．

（1） 単に安定した状態を表すのではなく，トークンの移動により動作をダイナミックに表現できる．

（2） 複数のプロセスが協調しながら処理を進めていく場合の同期の問題を自然に表現できる．

（3） 数学的な検証を行うことができる．

一方，短所としては以下にあげるものがある．

（1） 現実に使用することの多いデータの処理を扱うのが不得意である．

（2） きわめてシンプルで記述能力が十分ではなく，具体的な記法などは利用者に任せられている．

4. 仕様の論理検証項目

通信仕様では，以下に列挙する項目に関して検証する必要がある[11]．

（1） 完全性……すべての可能な入力が処理され，すべての出力が受信されること．

（2） 有界性または安全性……メッセージの数及びバッファが有限であること．特にその数が1のとき安全という．

（3） 活性……状態遷移が実行可能であること．

（4） デッドロックフリー……どの状態へも遷移できないことがないこと．

（5） 初期状態への復帰……交換システム特有の性質として，あらゆる状態から初期状態（すべてのリソースがアイドルである状態）に戻れること．

たとえば，1対1の通信プロトコルをペトリネットで記述した場合，発火し得るすべてのトランジションを発火させ，生成されるマーキングを作り出して上記の諸性質を設計者自身が確認することはよく行われている．以下ではそれ以外の検証法について述べる．

5. ペトリネットによる仕様の記述

一般に通信システムをFSMで表すことが多いが，状態の数が多くなり過ぎるという点からペトリネットを利用することが考えられる．たとえば，**図-4**は両端のいずれからも切断要求が可能なデータ通信のプロトコルをモデル化している．現在トークンはアイドル状態を示すプレースIおよびI′のみにそれぞれ1個存在しており，トランジションACの発火によりトークンはプレースTおよびT′に遷移し通信状態となる．T側から切断要求があるときはまずトランジションDRの発火によりプレースTのトークンがプレースDに遷移し，次にT′側からの応答信号によりトランジションAD′が発火して最初の状態に戻る．反対にT′側からの切断要求があったときも同様である．

仕様の検証は以下の手順で行われる．まず，システムをPNで記述した後，そのPNで取り得る

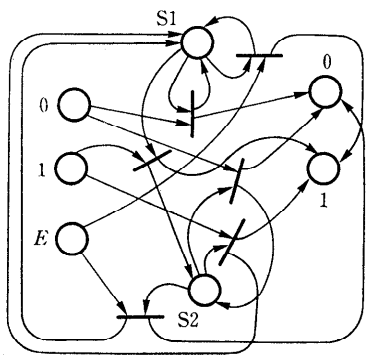
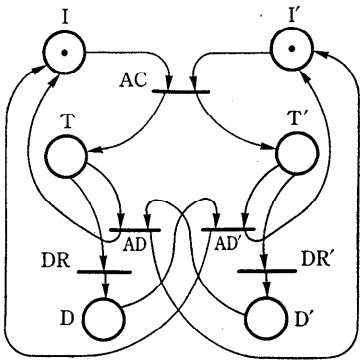


図-3 図-2のFSMから変換したPN



I：アイドル状態　AC：接続
T：データの送受信　DR：切断要求
D：切断要求状態　AD：切断確認

図-4 データ通信を示すペトリネット

すべての状態，つまりすべてのマーキングの遷移を作成する．ここで作成される遷移図は FSM であり，この到達可能グラフ (Reachability Graph) をトークンマシン (TM : Token Machine)[11),12)] という．たとえば図-4 の PN に対する TM を**図-5** に示す．

ここでは，各時点においてトークンが存在するプレースの集合が一つの状態を形成している．そして，TM の段階で各種項目を検証し，それらを満足しなければ適宜 PN を修正・変更していこうとするものである．たとえば，図-5 の TM で異常処理あるいは準正常処理の一つとして，同時に両端から切断要求があった場合の処理が抜けていることが分かるため，これを追加する．図-5 の中で点線で示したものがこれである．また，図-4 にプレースD及び D′ のトークンがプレースI及びI′に遷移するためのトランジション DD を追加したものを**図-6** に示す．

この方式は少なくとも関心のある部分について取り得るすべての状態を明確にすることによってシステムの動作を解析するものであり，理論的な解析はないものの理解が容易であるという利点を有する．上記の例では非常に簡単なモデルであるため，PN のプレースの数が6個であって FSM の状態数である5個よりも多くなっている．しかし，FSM でいう状態は PN でいうマーキングであり，したがって一般にはプレースの数は FSM の状態数よりも少なくなるため，この手法の採用により設計量を減少させることができる．

たとえば，安全なペトリネットであるとすると，各プレースでトークンがあるかどうかで二つの状態を表現することができる．したがって n 個のプレースがあるとき，全部で 2^n 個の状態を表せる．ところが，FSM では 2^n 個の状態を表すには基本的に 2^n 個だけの状態が必要となる．

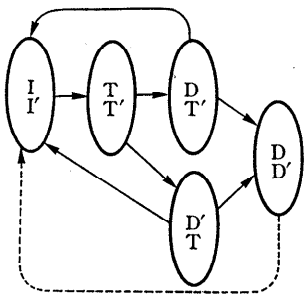
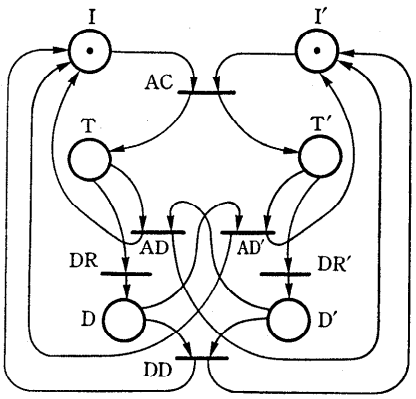


図-5 図-4 のトークンマシン (TM)



I：アイドル状態　　AC：接続
T：データの送受信　　DR：切断要求
D：切断要求状態　　AD：切断確認
　　DD：切断完了

図-6 図-4 に同時切断要求処理を付加したペトリネット

6. ペトリネットによる仕様の記述及び検証

前章と同様に仕様自体をほかの仕様記述言語によることなくペトリネットで記述し，かつ検証を行うものを考える．中でも，トークンを移動させて検証する挙動的解析ではなく，ペトリネットの構造的な性質を利用して検証を行う手法について述べる．

6.1 構造面からの検証の例

図-6 に示す PN について以下に列挙する項目を検証する[14)]．なお，このペトリネットでは1対1のプロトコルをモデル化しており，入力も出力も存在しないため完全性の検証は不要である．また，**図-7** は図-6 の PN の接続行列である．

（1）安全性　図-7 に示す接続行列から以下の4個のSインバリアントが求められる．

$$
\begin{aligned}
&\quad I\ \ T\ \ D\ \ I'\ \ T'\ \ D'\\
y1&=[1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0]^T\\
y2&=[0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1]^T\\
y3&=[1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1]^T\\
y4&=[0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0]^T
\end{aligned}
$$

$$
A=\begin{array}{c} \\ AC \\ DD \\ DR \\ AD \\ DR' \\ AD' \end{array}
\begin{array}{c}
\begin{array}{cccccc} I & T & D & I' & T' & D' \end{array} \\
\begin{bmatrix} -1 & 1 & 0 & -1 & 1 & 0 \\ 1 & 0 & -1 & 1 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & -1 & 1 & -1 & 0 \end{bmatrix}
\end{array}
$$

図-7 図-6 のペトリネットの接続行列

上記のベクトル中の0でない要素は1であり，かつすべてのベクトルについて0となるプレースは存在しないため，最大のトークン数は1であり安全であることが導かれる．

(2) 初期状態への復帰　初期状態を示すマーキング (M_0) から到達し得る任意のマーキングを M として，M から再び初期状態に戻れるためには，初等的Tインバリアントが存在しかつその発火系列が作成するマーキングの中に M が含まれていることが必要十分条件である．図-6では以下の3個のTインバリアントが求められる．

AC DD DR AD DR′ AD′

$x1$=[1　0　1　0　0　1]^T

$x2$=[1　0　0　1　1　0]^T

$x3$=[1　1　1　0　1　0]^T

上記の発火可能ベクトルの中で初期マーキングから**2.**で述べた発火条件を満足する発火トランジションを求めていくことにより，以下に示すように初期マーキングから再び初期マーキングに至る系列が4個 ($x1$ より AC→DR→AD′, $x2$ より AC→DR′→AD, $x3$ より AD→DR→DR′→DD 及び AD→DR′→DR→DD) 得られる．つまり，これらの発火系列により生成し得るマーキングからはすべて初期状態に復帰できるといえる．

(3) 活　性　$x1$, $x2$, $x3$ のどのTインバリアントにおいても発火しないトランジションは存在しない．つまり，トランジションはいずれも活性である．

また，PN に関するツールを用いて通信プロトコルを設計し種々の検証を行うものが報告されている[15]．

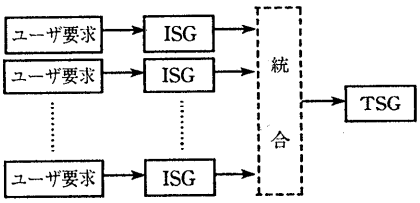
6.2 サービス仕様の統合・検証

図-1(b)の分析工程では高級技術者がユーザ要求を正しく理解して設計仕様を作成する．しかし，場合によってはある程度の技術力をもったユーザがユーザの立場からみたサービス仕様を作成することが考えられる．これにより普通の技術者がこのサービス仕様をもとに設計仕様を作成することを容易にする．ここでは，そのサービス仕様の統合及び検証について述べる[16]．

6.2.1 サービス仕様の統合

ユーザ要求をもとに，それから直接に作成した個々のサービス仕様 (個別サービスグラフ：ISG) を経て最終的に相互に矛盾のないサービス仕様 (統合サービスグラフ：TSG) を作成する過程を**図-8** に示す．ユーザ要求については，ユーザの端末操作とそれに対するシステム側の応答という形式で，基本的に初期状態から再び初期状態に戻るまでを記述する．

PBX (Private Branch eXchange：構内交換機) の最も基本的なサービスである内線相互接続サービス (TSG) を**図-9** に示す．これは，正常接続の場合のほかに，呼出し音を聴取しているときにオンフックした途中放棄のサービス，及び受話器をあげた時点で特定の相手を呼び出すホットラインサービスの正常接続の三サービス (ISG) から合成されている．実際のシステムでは，TSG は数百以



ISG (Individual Service Graph)：個別サービスグラフ
TSG (Total Service Graph)：統合サービスグラフ

図-8 仕様獲得過程

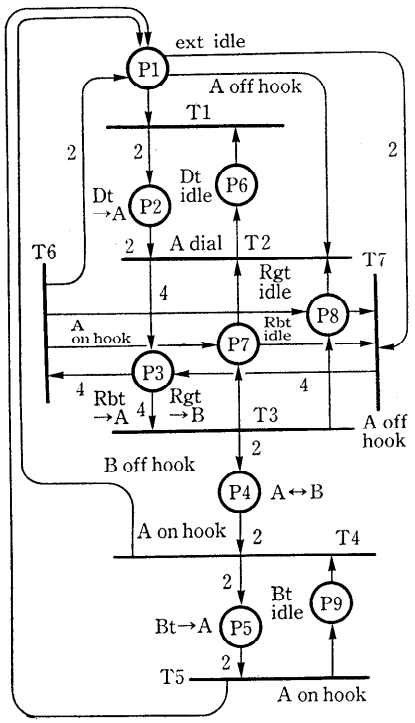


図-9 内線相互接続に関する三サービスを統合した TSG

上のサービスを合成したものとなる．なお，図-9でアークの横の数は多重度を示している．

6.2.2 サービス仕様の検証

ISG の検証については，基本的に **6.1** で述べたものと同様であるためここでは省略し，TSG の検証について説明する．TSG は複数個の ISG から論理的に正しく統合されていることを検証するには以下の二項目を調べる必要がある．

（1） 統合すべき ISG をすべて包含している．

（2） 基本的に ISG 以外のサービスを含まない．

そのため，まず統合した PN の接続行列から素サービスを求め，これが ISG に示す T インバリアントをすべて含むことにより（1）を検証できる．また，ISG の示す T インバリアント以外のものを含むとき，それは新たなサービスということができる．その場合，それをユーザに提示して許容するかどうかを問い合わせる．たとえば図-9 に示す PN の接続行列から，以下の 4 つの初等的 T インバリアントを求めることができる．

$$\begin{array}{cccccccc} & T1 & T2 & T3 & T4 & T5 & T6 & T7 \\ p=[& 1 & 1 & 1 & 1 & 1 & 0 & 0]^T \\ q=[& 1 & 1 & 0 & 0 & 0 & 1 & 0]^T \\ r=[& 0 & 0 & 1 & 1 & 1 & 0 & 1]^T \\ s=[& 0 & 0 & 0 & 0 & 0 & 1 & 1]^T \end{array}$$

ここで，p, q 及び r は，上で述べた三つのサービス（それぞれ T1→T2→T3→T4→T5, T1→T2→T6, T7→T3→T4→T5 の発火系列）を表しており，残りの s が新しいサービス（T7→T6 の発火系列）を示している．

7. ペトリネットによる仕様の検証

本章では，ほかの仕様記述言語で作成された通信仕様をいったんペトリネットに変換して，その状態で検証しようとするものを取り上げる．

3.1 で述べたとおり，通信システム用仕様記述言語として，SDL, LOTOS 及び Estelle が有名である．LOTOS についてもいったん PN に変換した後検証することが行われている[17]が，ここではテキスト表現のほかにグラフィカル表現も有するなど，記述能力に優れ最も普及している SDL を取り上げる．SDL は，1976 年に最初に勧告されてから，CCITT で審議されて 4 年ごとに新しい勧告が出されている．この仕様記述言語は FSM に基づくものであるが，単なる状態の遷移だけを表すのではなく，データの操作やタスクの処理などかなり実際のシステムに適用し得る機能が追加されている．また，構造的な概念を有しており，全体のシステムを複数のブロックに，またブロックを複数のプロセスに静的に分割し，プロセスの中を FSM として表現する．簡単なプロセスダイアグラムの例を**図-10** に示す．これは，二個のプロセスと 5 個のステート及び 6 個のシグナルからなる SDL である．なお，この図は理解性をよくすることを目的として，必ずしも正確なシンタックスに従っていないことを断わっておく．

SDL は安定した状態間の遷移を表現するが，そのダイナミックな動作を表現するものではない．一方，PN はマーキングの移動により状態の遷移をダイナミックに表現することができる．そのため，SDL 仕様を PN に変換して，その可到達性を調べてデッドロックなどを検出することが考えられる[18),19)]．ここで，その基本的な変換ルールを以下に示す．

（1） ステートは，プロセスがその状態にあるときに，トークンがあるプレースとしてモデル化される．

（2） 個々のプロセスがもつ FIFO キューは，n 個のトークンをもち得るプレースとして表現される．SDL では，キューの長さは無限を許容するが，これに制限値を設けても実際上は問題ない．

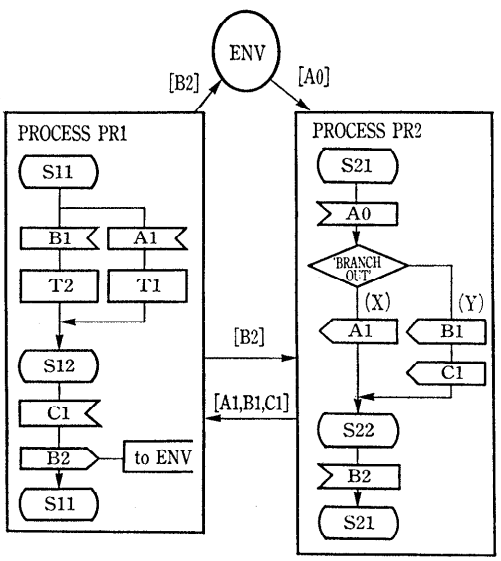


図-10 二個のプロセスからなる SDL 図例

（3） デシジョンは，そのブランチによって異なるアウトプットシグナルなどに応じて，同じインプットシグナルに対して別々のトランジションを作成する．

以上の変換ルールを適用して図-10 を PN に変換したものを**図-11** に示す．上記(2)によりこの PN は色付きペトリネットになっており，アークを通過するシグナル名を図の中に記している．SDL のプロセス数は二個，ステート数は 4 個であるが，これから変換した PN では，プレース数が 7 個（各ステートに対して一個，プロセスごとの FIFO キューとして各一個，環境—ENV—として一個），トランジション数が 7 個（ステートごとの暗黙の消費を明示するためにはさらにプロセス内のステート数に等しい 4 個を加えて 11 個）となる．図-11 の PN から，デッドロックは二カ所で発生することが分かる．一つは X 分岐のときであり，プロセス PR 1 はステート S 12 で絶対に到達することのないシグナル C 1 を待ち続けることになる．もう一つは Y 分岐のときであり，プロセス PR 1 がシグナル B 2 を環境に送信するため，プロセス PR 2 はそのシグナルを受け取ることができない状態になる．

さらに，SDL で記述された通信プロトコル仕様をペトリネットに変換し，既存のペトリネットツールを使用して各種のプロトコルエラーを検出することも試みられている[20]．ここでは，SDL でいうタイムアウト処理を PN ではタイマのプレースを設けることで実現しており，上記のデッドロックのほかにタイマのリセット忘れなども検出できる．

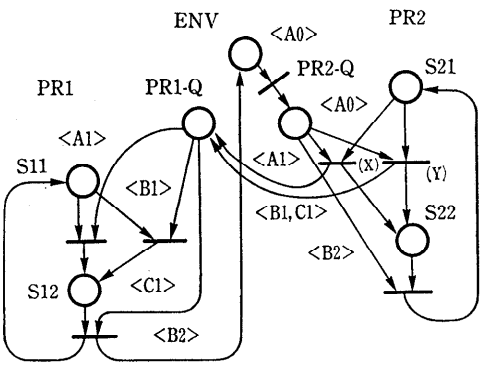


図-11 図-10 の SDL 図から変換したペトリネット

8. むすび

通信ソフトウェアの要求仕様化段階にペトリネットを応用しているいくつかの例をあげた．通信システムはリアルタイム性が要求され，かつきわめて膨大である．したがって，それを基に仕様をできるだけ早い段階で誤りや矛盾を取り除いておくことが必要である．有限状態マシンとしてモデル化されることの多い通信システムに，解析能力の優れるペトリネットを利用することは非常に有効であると考える．一方で PN の利用に問題もある．

PN によるモデル化は，解析力を有するがその記述レベルがプリミティブなだけに計算量の爆発をきたすという欠点である．しかし，処理の高速化が今後ますます予想されることから徐々にではあるが解決されつつある．また，この記述能力の限界を克服するために色付きペトリネットを含むハイレベルネットを利用することが近年行われている．ただ，記述能力と検証能力にはトレードオフの関係があり，たとえばそれらのインバリアントの計算手法を確立して構造的な側面からの検証にどのように活用していくかは今後の課題である．

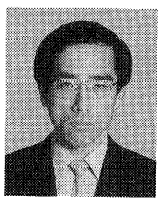
さらに，実用上の問題も存在する．たとえば，SDL では階層化構造，データの入出力などのさまざまな概念が FSM モデルに追加されており，ユーザはそれに従ってシステムを記述するだけでよい．ところが，PN では図-4 に示したようにデータの送受を陽に表現することを苦手とし，入力や出力までプレースとトランジションを駆使して表現するしか方法がない．つまり，高級技術者がフレキシブルに定義して自由に機能を拡張できるという利点を有するとは言えるが，逆にシステムの記述になんらの制約もないことは多人数で行うソフトウェア開発プロジェクトを考えた場合欠点でもある．通信システムを記述する上でなんらかの規範を確立することが必要となろう．

結論として，計算量の爆発に対しては PN の階層化や簡単化により計算可能なレベルにまでコンパクトなものにすることで対処し，特に状態遷移に着目してその正当性を検証するなど，限られた対象のもとできわめて有効なものとなろう．

参考文献

1) Peterson, J. L.: Petri Net Theory and the Modeling of Systems, Prentice Hall (1981).
2) Reisig, W.: Petri Nets, Springer-Verlag (1982).
3) Murata, T.: Petri Nets: Properties, Analysis and Applications, IEEE Proc., Vol. 77, No. 4, pp. 541-580 (1989).
4) Jensen, K.: Introduction to Coloured Petri Nets, Proc. of the 3rd Internatinal Workshop on Petri Nets and Performance Models, Pre-Workshop Tutorials, pp. 77-116 (1989).
5) Balzer, R.: A 15 Year Perspective on Automatic Programming, IEEE Trans. on Software Eng., Vol. SE-11, No. 11, pp. 1257-1268 (1985).
6) CCITT: Recommendation Z. 100 Functional Specification and Description Language (SDL) (1988).
7) ISO: ISO 8807, Information Processing System —Open Systems Interconnection—LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour (1989).
8) ISO: ISO 9074, Information Processing System —Open Systems Interconnection—Estelle—A Formal Description Technique Based on an Extended State Transition Model (1989).
9) Wakahara, Y., Kakuda, K., Ito, A. and Utsunomiya, E.: ESCORT: An Environment for Specifying Communication Requirements, IEEE Trans. on Software Eng., Vol. SE-6, No. 2, pp. 38-43 (1989).
10) Ichikawa, H., Itoh, M. and Shibasaki, M.: Protocol Oriented Service Specifications and Their Transformation into CCITT Specification and Description Language, Trans. IEICE. Vol. E 69, No. 4, pp. 524-535 (1986).
11) Merlin, P. M.: A Methodology for the Design and Implementation of Communication Protocol, IEEE Trans. on Commun., Vol. Com-24, No. 6, pp. 614-621 (1976).
12) Danthine, A. S.: Protocol Representation with Finite State Models, IEEE Trans. on Commun., Vol. Com-28, No. 4, pp. 632-643 (1980).
13) Merlin, P. M. and Farber, D. J.: Recoverability of Communication Protocols: Implications of a Theoretical Study, IEEE Trans. on Commun., Vol. Com-24, No. 9, pp. 1036-1043 (1976).
14) Berthelot, G. and Terrat, R.: Petri Nets Theory for the Correctness of Protocols, IEEE Trans, on Commun., Vol. Com-30, No. 12, pp. 2497-2505 (1982).
15) Billington, J. and Wilber-Ham, M. C.: Automated Protocol Verification, Proc. of Protocol Specification, Testing and Verification, No. V, pp. 59-70 (1985).
16) Shibata, K., Ueda, Y., Yuyama, S. and Hasegawa, H.: A Study on Verification of Service Specification in Communication Software Development, IEICE Trans., Vol. E 71, No. 12, pp. 1203-1211 (1988).
17) Garavel, H. and Sifakis, J.: Compilation and Verification of LOTOS Specifications, Proc. of Protocol Specification, Testing and Verification No. X, pp. 379-394 (1990).
18) Kettunen, E. and Lindqvist, M.: Towards Practicality of Predicate/Transition Petri Net Reachability Analysis of SDL, Proc. of 3rd SDL Forum, pp. 285-294 (1987).
19) 宗森 純, 武田捷一: ペトリネットによるシミュレーション機能の検討, 電子情報通信学会, 第3回ネット理論研究会, pp. 34-41 (1988).
20) 後藤雅徳, 村田忠夫: ISDN 局間信号方式の色付きペトリネットモデル, 計測自動制御学会, 第8回離散事象システム研究会, pp. 71-78 (1991).

(平成4年12月2日受付)



長谷川晴朗 (正会員)

昭和25年生. 昭和47年東京大学工学部電子工学科卒業. 昭和49年同大学大学院修士課程修了. 同年沖電気工業(株)入社. デジタル交換機の加入者回路, メッセージ交換機の通信制御装置, 及びPBX ソフトウェアの開発に従事. 昭和60年より通信システム仕様記述の研究開発を行う. 現在同社ネットワークシステム開発センターソフトウェア開発第三部長, 通信ソフトウェアの開発環境に興味をもつ. IEEE, 電子情報通信学会, 計測自動制御学会各会員.