ソフトウェア工学 86－T 2
（1992. 7. 21）

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

Application of Petri Nets to the specification phase of communication software development

長谷川晴朗
Haruo Hasegawa
沖電気工業(株) 通信ネットワーク事業本部
Telecommunications Group, Oki Electric Industry Co., Ltd.

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･高速化に伴い,通信ソフトウェア開発の重要性が飛躍的に増大している. そして,「ソフトウェアクライシス」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),2),3)]は,非同期･並行系システムのモデル化技法として様々な分野に利用されているが,性能評価等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特徴について述べた後,仕様の論理的な検証項目について説明する. 5章から7章では,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3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.

(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,単位列ベクトルをe_jとして, $M_0 \geqq A^{-T} \cdot e_j$が成立することである. また, M_0から別のマーキングに至る発火系列で,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, $M=M_0+A^{T} \cdot \gamma$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, $A^{T} \cdot x=0$, $A \cdot y=0$を満足する時,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,他の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.

(4) サービス　交換システムでサービスとは,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特徴

ソフトウェアの開発工程は,図1に示すように一般にウォータフォールモデルで表される. そして,ソフトウェアの中でも通信ソフトウェアは,以下に掲げるような特徴を有する[4)].

(1) ハードウェアの監視と制御
(2) 高信頼性

*)〒261-71 千葉市美浜区中瀬 2-6 WBG マリブイースト
Tel.(043)297-6901　Fax.(043)297-6905
E-mail: hasegawa@wbg.telcom.oki. co.jp

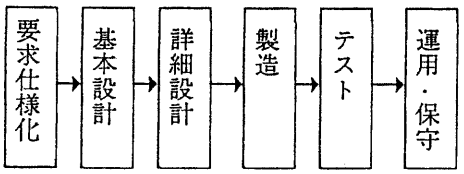


図1 ソフトウェアの開発工程

(3)超多重処理
(4)実時間性
(5)複雑なサービス
(6)膨大かつ変更の多いプログラム
(7)プロセス間の相互作用

これらの理由により,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.

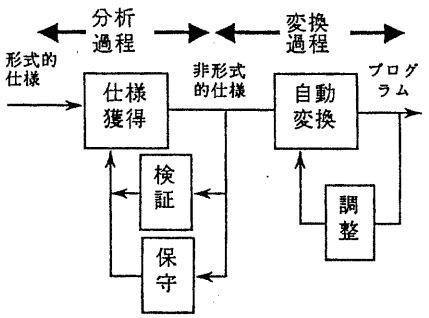


図2 新しいソフトウェア開発のパラダイム

一方,新しいソフトウェア開発のパラダイムによれば,図2に示すように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・保守すれば,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している[5]. 従って,作成した仕様を検証することが極めて重要なものとなっており,そのための記述法として様々なものが提案されている. 例えば,CCITT(国際電信電話諮問委員会)より通信システム用仕様記述言語としてSDL (Functional Specification and Description Language)[6]が勧告されており,これをベースとして様々な開発支援システムが構築されている[7],[8].

4. 仕様の論理検証項目

通信仕様は,以下に列挙する項目に関して検証する必要がある.

(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の時安全性という.

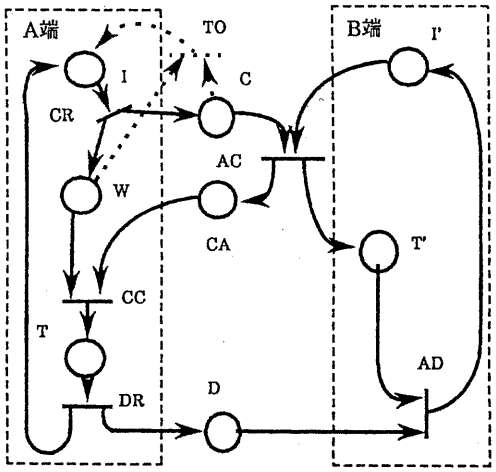
(3)活性 ・・・状態遷移が実行可能であること.

(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.

5. ペトリネットによる仕様の記述

一般に通信システムは有限状態マシン(FSM: Finite State Machine)で表現できるが,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. 例えば,図3は通信プロトコルをモデル化したものである.



プレース
I : アイドル状態
W : 待ち状態
C : 接続要求状態
CA : 接続確認
T : データの送受信
D : 切断要求状態

トランジション
CR : 接続要求
AC : 接続受付
CC : 接続確認
DR : 切断要求
AD : 切断確認
TO : タイムアウト

図3. データ送信のペトリネット

これは,片方(A端)からのみデータ送信要求及び切断要求を可能としたものである. このシステムでは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.この遷移図はFSMであり,これをトークンマシン(TM: Token Machine)[9),10)]という.例えば図3のPNに対するTMを図4に示す.

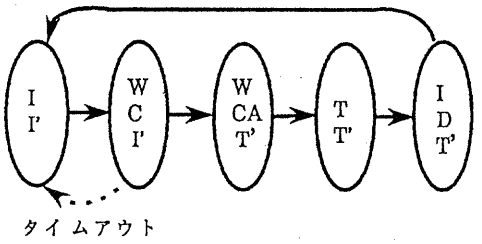


図4 図3のトークンマシン(TM)

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.例えば,図4のTMで異常処理あるいは準正常処理の一つとして待ち状態中にACK信号が到来しなかった場合のタイムアウト処理が抜けていることがわかるため,これを追加する.(図3の中で破線で示すトランジション部)

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.FSMでいう状態はPNでいうマーキングであり,従って一般にはプレースの数はFSMの状態数に比較して少ないため,設計量を減少させることを可能としている.

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他の仕様記述言語によることなく,ペトリネットで記述するが,さらに検証にまでペトリネット,特にその構造的な性質を利用するものを取り上げる.

6.1 仕様の簡略化と検証の例

6.1.1 仕様の簡略化

ここで取り上げるPNは,図3がデータ送信要求も切断要求も一方向のみのデータ通信であるのに対し,双方向のデータ通信を可能としている.そのためPN自体がかなり複雑なものとなり得るが,膨大な計算量をできるだけ少なくするため,完全性・有界性・活性等の諸性質を変化させることなくPNをリダクションにより簡略化しておくことがよく行われる.図5はその例であり,以下に示す簡略化の手順を用いて簡略化したPNである.

(1)プレースWは,プレースC又はプレースCAにトークンがある時にのみトークンが存在するため,冗長である.

(2)トランジョンCR·AC·CCは常にシーケンシャルに実行されるため,これらを一つのトランジションACに置き換える.

(3)データの送受信を両方向とした場合,(2)項で置き換えられたトランジションが2つできるが,これらは入力プレースも出力プレースも等しいため1個のトランジョンで置き換える.

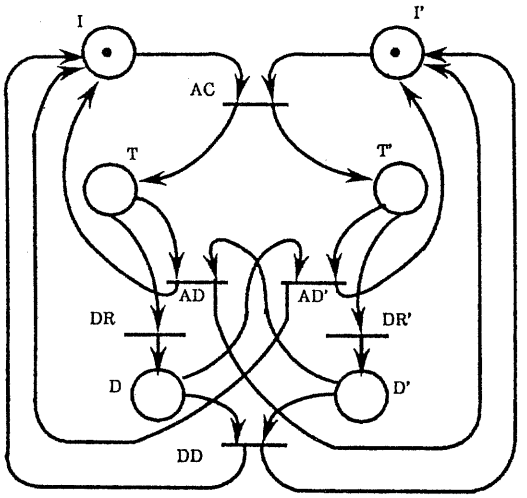


図5 両方向のデータ通信を示すペトリネット

6.1.2 仕様の検証

以下に列挙する項目を検証する[12)].

(1)安全性 図5のPNの接続行列から4個のSインバリアントが求められる.

	I	T	D	I'	T'	D'
y1=[	1	1	0	0	0	0]^T
y2=[	0	0	1	1	1	1]^T
y3=[	1	0	0	0	1	1]^T
y4=[	0	1	1	1	0	0]^T

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5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章の(2)項で述べた発火条件を満足する発火トランジションを認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．

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[13),14)]．

6.2　サービス仕様の統合･検証

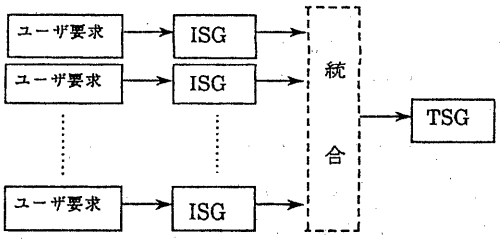
ここでは，設計仕様ではなくユーザの立場から見たサービス仕様の統合及び検証について述べる[15),16)]．

6.2.1　サービス仕様の統合

個々のユーザ要求(個別サービスグラフ:ISG)から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6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7に示す．これは，正常接続の場合の他に，呼び出し音を聴取している時にオンフックした途中放棄のサービス，及びホットラインサービスの正常接続の3サービス(ISG)から合成されている．実際のシステムでは，TSGは数百以上のサービスを合成したものとなる．

ISGの作成手順についてはここでは割愛し，複数のISGからTSGを統合する手順を以下に示す．なお，実



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6　仕様獲得過程

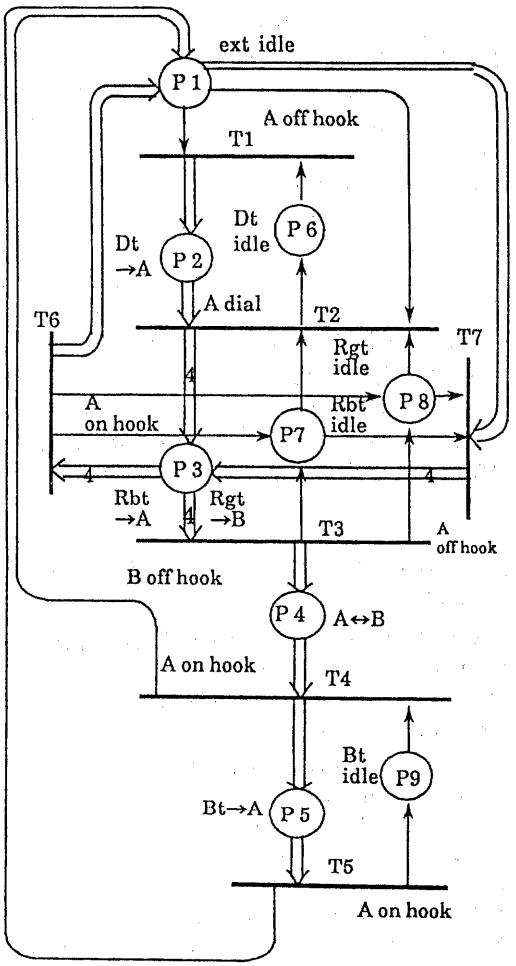


図7　内線相互接続に関する
3サービスを統合したTSG

際には新しいISGを既に作成しているTSGに統合する過程を説明することになるが,これは動作毎に以下の2フェーズに分けて行う.

(1)プレースの探索　ISG中の前状態を示すプレースと一致するプレースをTSGの中から探索する.もしあれば(2)に進み,なければそのプレースを追加する.

(2)トランジションの照合　プレースが一致したとき,そのISGのトランジションを照合し,さらにISGの後状態を表すプレースをTSG中のプレースと照合する.両方が一致すれば統合し,そうでなければそれらのプレースやトランジションをTSGに追加する.

例えば,最初に発呼者がオフフックした場合,通常ダイヤル音聴取中に遷移するが,ホットラインサービスでは呼出中状態に遷移する.このように,「オフフックする」という同じ端末操作であっても次状態が異なれば別のトランジションを作成する.

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項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2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時,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.例えば図7におけるPN接続行列から以下の4つの初等的Tインバリアント及びサービスを求めることができる.ここで,p,q及びrは,上で述べた3つのサービスを表しており,残りのsが新しいサービスを示している.

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

7. ペトリネットによる仕様の検証

本章では,他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.

通信システム用仕様記述言語として,CCITT勧告のSDL,ISO勧告のLOTOS[17]及びEstelle[18]が有名である.LOTOSについてもいったんPNに変換した後FSMに変換して検証することが行われている[19]が,ここではテキスト表現PR (text Phrase Representation) の他にグラフィカル表現GR (Graphical Representation) も有する等,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のSGXで審議されて4年毎に新しい勧告が出されている.この仕様記述言語は有限状態マシン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等かなり実際のシステムに適用し得る機能が拡張されている.また,構造的な概念

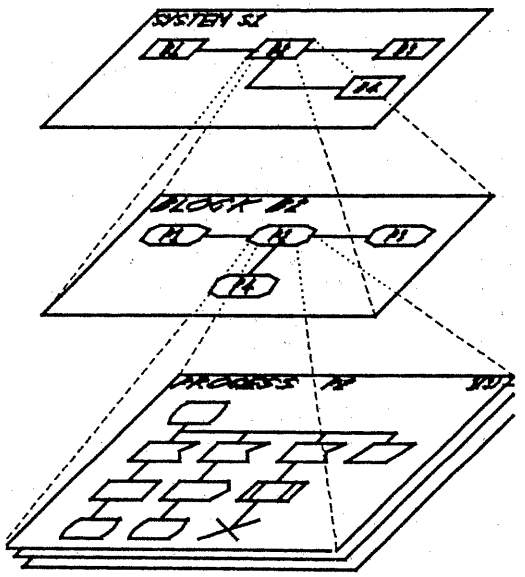


図8 SDLの構造化概念

を有しており,図8に示すように全体のシステムを複数のブロック,またブロックを複数のプロセスに静的に分割し,プロセスの中を有限状態マシンとして表現する.SDLの主要な構成要素を図9に示す.また,簡単なプロセスダイアグラムの例を図10に示す.これは,2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則ったものではないことを断っておく.

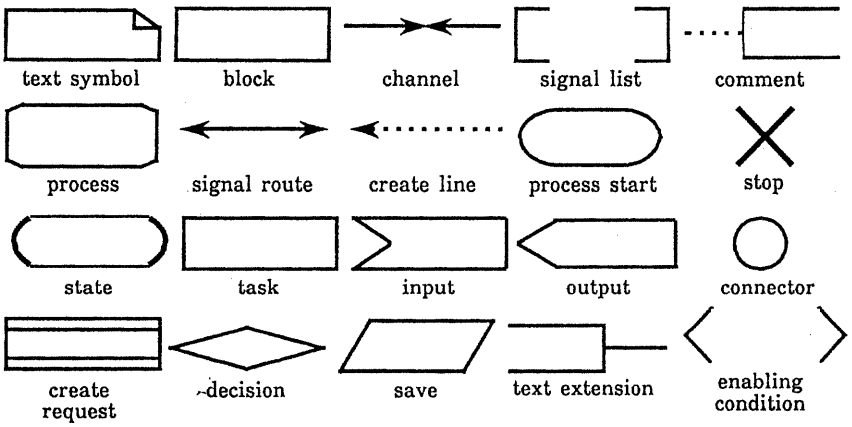


図9 SDLの主要な構成要素(GR)

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等を検出することが考えられる[20),21)].ここで,その最も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時にトークンがあるプレースとしてモデル化される.

(2)個々のプロセスの持つFIFOキューは,n項のトークンを持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.

(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等に応じて,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)項によりこのPNはハイレベル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2個,ステート数は5個であるが,これから変換したPNでは,プレース数が8個(各ステートに対して1個,プロセス毎のFIFOキューとして各1個,環境-ENV-として1個),トランジション数が10個(ステート毎の暗黙の消費を明示するためにはさらにプレース数に等しい5個を加えて15個)となる.

図11のPNから,デッドロックは2ケ所で発生する.一つはX分岐またはZ分岐の時であり,プロセスPR10はステートS103で絶対に到達することのないシグナルC10を待ち続けることになる.もう一つはY分岐の時であり,プロセスPR10がシグナルB20を環境に送信するため,プロセスPR20はそのシグナルを受け取ることができない状態になる.

8. まとめ

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挙げた.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.通信システムは極めて膨大であり,ユーザ要求も極めて曖昧である.従って,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.PNによるモデル化は,解析力を有するが計算量の爆発を来たすという欠点がある.しかし,処理の高速が今後益々予想されることから限られた対象のもとでは有効なものと言える.また,ペトリネットではデータの送受をそのままでは陽に表現することが苦手であること等から,記述能力に限界があり,これを克服するために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,例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.

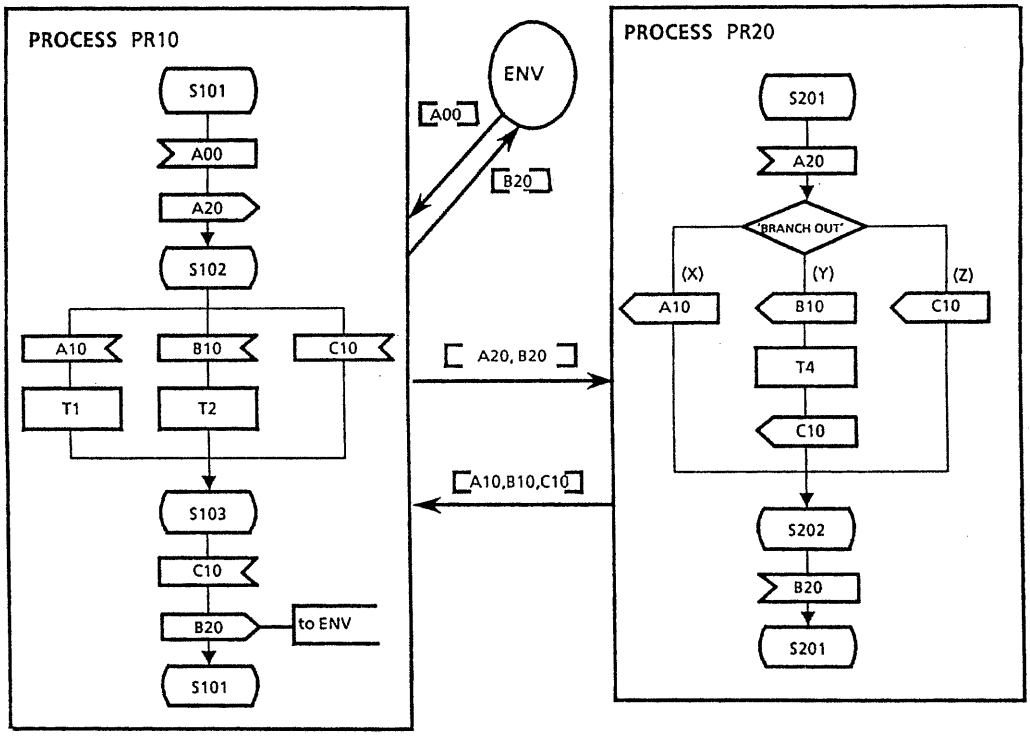


図10 2個のプロセスからなるSDL図例

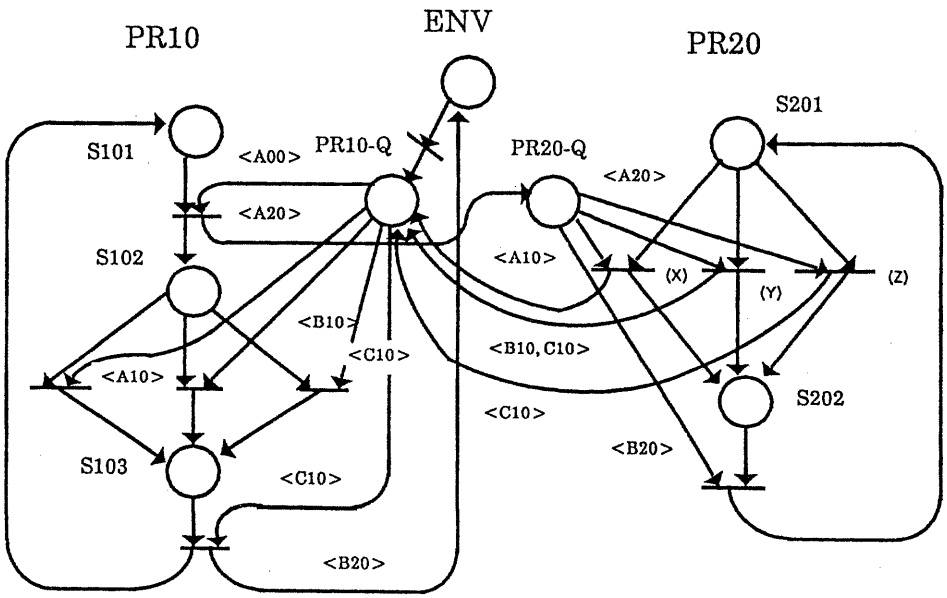


図11 図10のSDL図から変換したペトリネット

参考文献

1)J. L. Peterson: Petri Net Theory and the Modeling of Systems, Prentice Hall (1981)

2)W. Reisig: Petri Nets, Springer-Verlag (1982)

3)T. Murata : Petri Nets: Properties, Analysis and Applications, IEEE Proc., 77-4, 541/580 (1989)

4)電気通信審議会編:電気通信と人工知能, オーム社 (1988)

5)R. Balzer et al. : Software Technology in the 1990's: Using a New Paradigm, IEEE Computer, 16-11 (1983)

6)CCITT: Recommendation Z. 100 Specification and Description Language (SDL) (1988)

7)若原恭他: 通信ソフトウェア要求仕様検証支援システム, 電子通信学会, 交換研究会, SE84-50 (1984)

8)H. Ichikawa, M. Itoh and M. Shibasaki: Protocol Oriented Service Specifications and Their Transformation into CCITT Specification and Description Language, Trans. IEICE, E69, 4 524/535 (1986)

9)P. M. Merlin: A Methodology for the Design and Implementation of Communication Protocol, IEEE Trans. on Commun., Com-24-6, 614/621 (1976)

10)A. S. Danthine: Protocol Representation with Finite State Models, IEEE Trans. on Commun., Com-28-4, 632/643(1980)

11)P. M. Merlin and D. J. Farber: Recoverability of Communication Protocols: Implications of a Theoretical Study, IEEE Trans. on Commun., Com-24-9, 1036/1043 (1976)

12)G. Berthelot and R. Terrat: Petri Nets Theory for the Correctness of Protocols, IEEE Trans. on Commun., Com-30-12, 2497/2505 (1982)

13)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, 71/78

14)J. Billington and M. C. Wilber-Ham: Automated Protocol Verification, Proc. of Protocol Specification, Testing and Verification, V, 59/70 (1985)

15)長谷川晴朗: ペトリネットを利用した交換サービス仕様設計支援, 電子情報通信学会, 交換・通信ソフトウェア仕様記述言語の標準化と技術展望, 専門講習会, 87/96 (1988)

16)K. Shibata, Y. Ueda, S. Yuyama and H. Hasegawa : A Study on Verification of Service Specification in Communication Software Development, IEICE Trans., E71-12, 1203/1211 (1988)

17)ISO: Information processing system - Open systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour, ISO 8807 (1989)

18)ISO: Information processing system - Open systems Interconnection - Estelle - A formal description technique based on an extended state transition model, ISO 9074 (1989)

19)H.Garavel anf J. Sifakis: Compilation and Verification of LOTOS Specifications, Proc. of Protocol Specification, Testing and Verification, X, 379/394 (1990)

20)E. Kettunen and M. Lindqvist: Towards Practicality of Predicate/Tansition Petri Net Reachability Analysis of SDL, Proc. of 3rd SDL Forum, 285/294 (1987)

21)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, 34/41 (1988)