ICOT Technical Memorandum: TM-0243

TM-0243

ペトリネットを利用した
通信システム設計仕様の解析

長谷川晴朗，田中　亘
柴田健次，山口政巳
（沖電気工業）

November, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

ペトリネットを利用した
通信システム設計仕様の解析

Analysis of Design Specifications in a Communication System
by utilizing Petri Nets

長谷川晴朗　田中亘　柴田健次　山口政巳
Haruo HASEGAWA　Wataru TANAKA　Kenji SHIBATA　Masami YAMAGUCHI

沖電気工業株式会社
OKI Electric Industry Co.,Ltd.

１．まえがき

筆者らは、交換を中心とする通信システムにおいて、自然言語で記述されたユーザ要求から設計仕様を自動作成するエキスパートシステムＥＸＰＲＥＳＳ(EXPert system for ESS) を開発中である。同システムでは、拡張ペトリネットにより設計仕様を記述し、同時にペトリネットを利用した代数的解析手法を利用して設計仕様の論理検証を行っている。ここでは、システムの概要を述べた後でペトリネットの関数行列を利用した解析について報告する。

２．通信システム用仕様記述

通信システムにおける仕様記述言語については、国際電信電話諮問委員会第Ｘ研究委員会（ＣＣＩＴＴ　ＳＧＸ）において審議されてＳＤＬ(Specification and Description Language)が勧告されており、現在国内においても幅広く利用されている。ただ、これは従来より交換システム等で使用されているものを集約したと考えられ、最近では別の観点から一部見直しがなされようとしている。つまり、通信システム用記述言語の中心としての位置付けは不変であるが、検証可能性や実働性等の理論的な解析についての要求が出始めており、この点からＳＤＬ以外の仕様記述言語に興味が持たれている。

特に、ペトリネットは従来より非同期・並行システムをモデル化し解析する手法として知られており、また、知識表現に使用されたりロジックプログラミングへ適用されたりしている。筆者らが今回通信システムの仕様記述にペトリネットを選んだのもこれらの理由からである。

３．仕様設計エキスパートシステムの概要

図１にEXPRESS のシステム構成概念図を示す。本システムは、基本仕様獲得サブシステム、要求理解仕様作成サブシステム、知識ベース及び交換ハードウェア制御部の４つから構成される。

（１）基本仕様獲得サブシステム

交換知識を持った設計者（専門家）が自然言語により基本的な交換動作を入力す

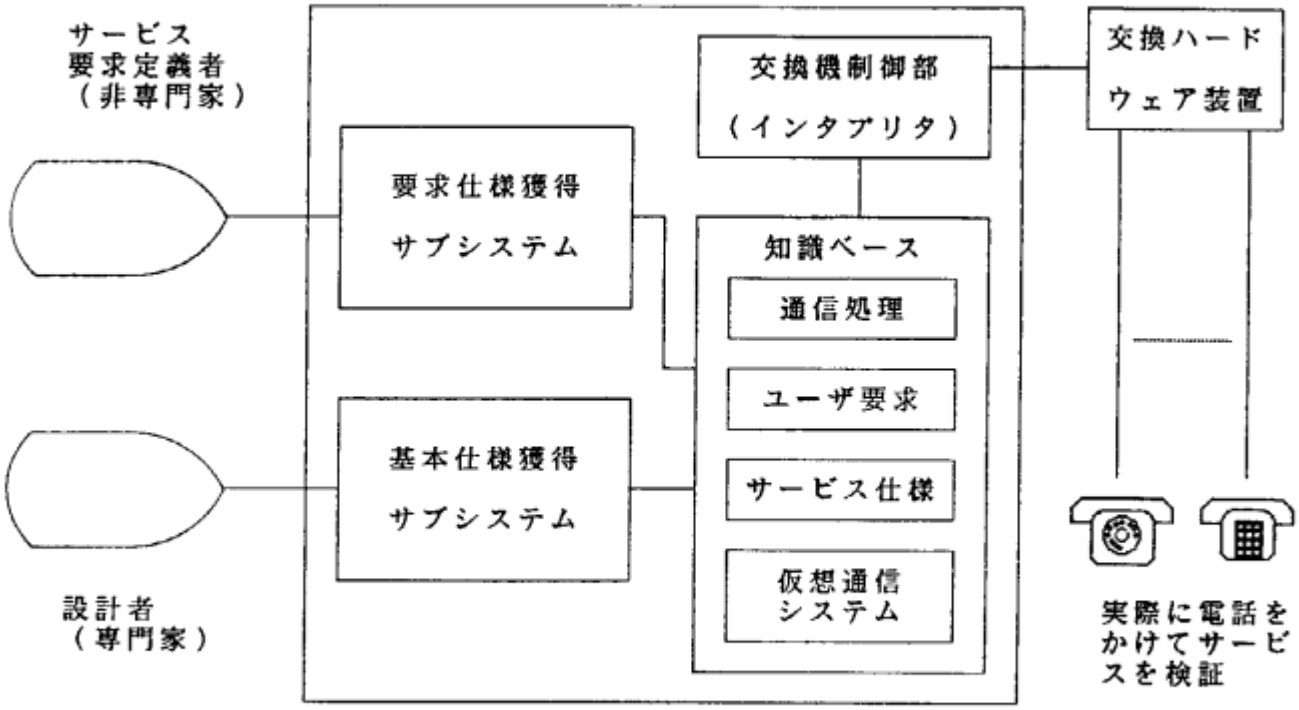


図１　ＥＸＰＲＥＳＳシステム構成概念図

ると、それより知識を獲得して知識ベースを構築または更新する。

（２）要求理解仕様作成サブシステム

サービス要求定義者（非専門家＝ユーザ）が自然言語により交換サービスに関する曖昧な要求仕様を入力すると、知識ベースに基づいて要求理解を行う。入力される要求仕様は一般に個々のサービス単位に断片的であって、従って相互に矛盾する可能性を有しているが、本サブシステムはこれから誤りのない設計仕様を作成する。また、知識ベースと矛盾するものが入力された時は、ユーザとインタラクションをとることにより矛盾が解消される。

（３）知識ベース

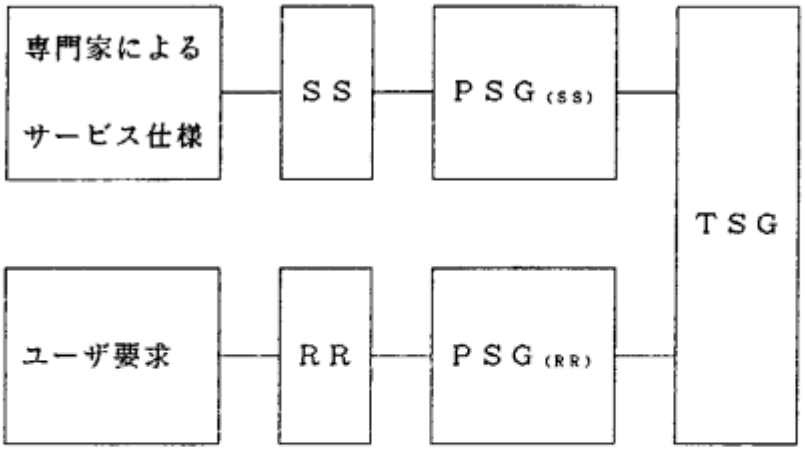
上記（１）、（２）の各サブシステムを実行するために必要な知識及びその実行結果が蓄積されている。ここには交換処理、交換ハードウェア、ユーザ要求、交換サービス仕様等に関する情報が知識ベース化されている。

（４）交換ハードウェア制御部

知識ベースに蓄積された交換サービス仕様に従ってインタプリタを起動し、外部交換ハードウェア装置を制御し、ユーザ要求が正確に理解されているか否かの意味検証を行う。

４．サービスグラフの統合

EXPRESS において、ユーザ要求或いは専門家によるサービス仕様から相互に矛盾のない仕様を作成する過程を図２に示す。



ＲＲ　：要求仕様　　Requirement Representation
ＳＳ　：サービス仕様　　Service Specification
ＰＳＧ：部分サービスグラフ　Partial Service Graph
ＴＳＧ：統合サービスグラフ　Total Service Graph

図２．　設計仕様の作成過程

（１）ユーザ要求

自然言語で記述するが、ユーザは通信システムに関する知識を持たないため，曖昧な要求表現を許容する。

（２）専門家によるサービス仕様

自然言語による表現も可能であるが、記述量を少なくし作業効率を高めることができる形式的な記述を中心とする。

（３）ＲＲ（要求仕様）とＳＳ（サービス仕様）

ＲＲとＳＳは、表現の厳密さにおいてのみ異なるものである。ＳＳは、専門家によるサービスの開始から終了までの、端末の動作とその動作によって変化する端末の状態を正確に記述したものであり、ＲＲと異なり曖昧な点を含まない。

（４）ＰＳＧ（部分サービスグラフ）

ＰＳＧは、ＳＳによって記述されたサービス仕様を拡張ペトリネットによって変換したものである。ＳＳの動作をトランジションとし、対象毎に変化前の状態と変化後の状態をそれぞれ入力プレースの集合、出力プレースの集合として表す。ここで、ペトリネットを使用したことにより、単に状態を表現するだけでなく、発火させてトークンを移動させて状態の推移を明確にできる利点がある。

（５）ＴＳＧ（統合サービスグラフ）

ＴＳＧは、複数個のそれぞれには断片的なＰＳＧを統合したものであって、求める最終の設計仕様である。表現形式は、ペトリネットとの対応を明確にしたものとしており、単一の状態に一つのプレースを対応させ、かつ、一つのプレースに接続し得るトラン

ジションを明確にしている。

このようにして次々と作成される複数個の断片的なＰＳＧから１つの統合サービスグラフ（ＴＳＧ）に統合する。このＴＳＧが求める最終の設計仕様であり、やはりペトリネットにより表現される。図３に内線相互サービスの発呼者先掛時、相手話中時及び呼出中途中放棄に対するＰＳＧと、それらを統合したＴＳＧを示す。また、図４、図５にそれぞれＰＳＧ、ＴＳＧの表現形式の例を示す。

このような統合は、１つのＰＳＧをＴＳＧとパターンマッチングさせ、ＴＳＧの中に存在しないものがあれば、これをＴＳＧに付加することにより行われる。

```
knt(kn(内線相互接続，発呼者先掛),1).
place([rel(token(発呼者,ext,nil),idle,nil)]).
place([rel(token(ダイヤルトーン,dt,nil),idle,nil)]).
tr(token(発呼者,ext,nil),offhook,nil).
place([rel(token(発呼者,ext,nil),receive,token（ダイヤルトーン,dt,nil))]).
```

図４　ＰＳＧの表現形式例（ＰＳＧ１のＰ１→Ｔ１→Ｐ２）

・呼出中のＰＪＮＣ（Ｐ４）：

```
pjnc([T3],[T4,T8],[(TK7(ext),receive,TK8(rbt)),(TK9(ext),receive,TK10(rgt))],[rel(token(発呼者,ext,nil),receive,token(呼出音,rbt,nil)),rel(token（被呼者,ext,nil),receive,token(電話,rgt,nil))])
```

・ダイヤル中から通話中へのＴＳＧ＿ＴＲ（Ｔ３）：

```
tsg_tr([P1,P3],[P4],[gtr(TK3,dial)],[ttoken(TK3,TK7),ttoken(TK4,TK2),ttoken(TK5,TK8),ttoken(TK6,TK10),ttoken(TK1,TK9)])
```

図５　ＴＳＧの表現形式例（Ｐ４及びＴ３）

５．ペトリネットの関数行列

ここで簡単にペトリネットの行列について述べる。トランジションの入力関数及び出力関数を表す行列D^-，D^+を定義する。各行列はトランジションに対応するｍ個の行とプレースに対応するｎ個の列から成る。D^-のｊ行ｉ列の要素はＰｉからＴｊに入るアークの多重度を、またD^+のｊ行ｉ列の要素はＴｊからＰｉに出ていくアークの多重度を示す。各プレースに存在するトークンの数をｎ次元ベクトルμ（マーキングと呼ぶ）で表すと、(1)式が成立する時、Ｔｊが発火する。

$$\mu \geqq e[j] \cdot D^- \qquad \cdots (1)$$

但し、ｅ［ｊ］は第ｊ成分が１でそれ以外の成分が０のｍ次元ベクトルである。

また、マーキングμ_0において、Tj が発火すると、新しいマーキングは次のようにな

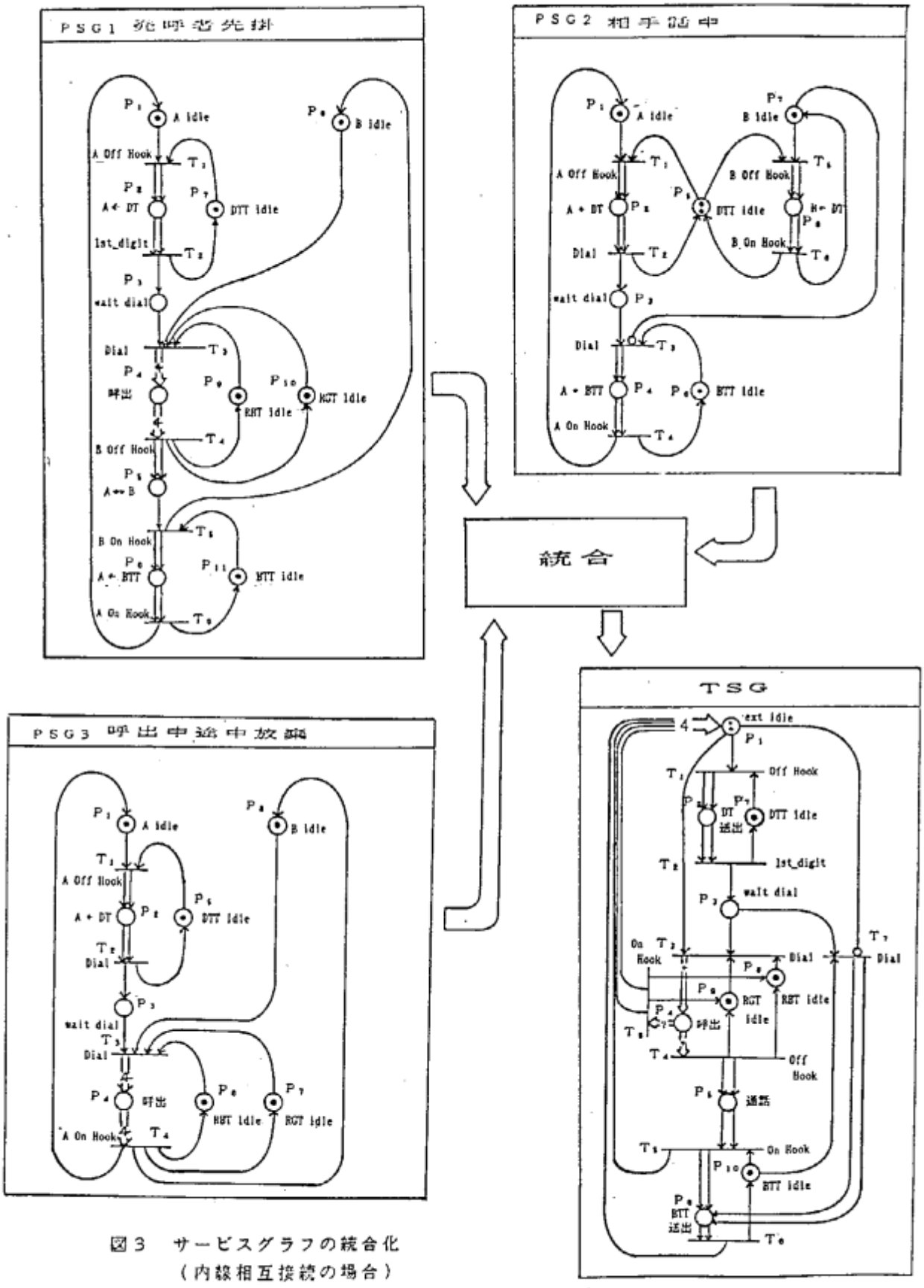


図３　サービスグラフの統合化
（内線相互接続の場合）

ることが知られている。

$$\mu = \mu_0 + e[j] \cdot D \quad \cdots (2)$$

但し、$D = D^+ - D^-$

従って、マーキングμ_0において、発火系列$\sigma = Tj_1\ Tj_2 \cdots Tj_k$が発火すると、新しいマーキングは次のようになる。

$$\mu = \mu_0 + (e[j_1] + e[j_2] + \cdots + e[j_k]) \cdot D \quad \cdots (3)$$
$$= \mu_0 + f(\sigma) \cdot D$$

6．論理矛盾の検出

全体として整合のとれた設計仕様（ＴＳＧ）を作成するには、それ自身に矛盾を含まないことが必要である。ところが、入力のサービス要求に矛盾が含まれていたり、或いは、設計仕様を作成する過程で矛盾が作りこまれる可能性がある。最終的にはインタプリタ起動により交換ハードウェア装置を起動させて要求仕様の確認を行うのであるが、この段階で矛盾を検出してもその修正にかなりの工数を要する。従って、設計仕様の作成段階で各種の矛盾を検出し除去しておくのが望ましい。

ところで、矛盾の検出には論理検証と意味検証とがある。後者は交換に関する知識が要求されるものであり、入力のサービス要求が正しく実現されているかを検証することが主であって、一部交換の知識を利用して検証することも可能であるが、基本的には上記のプロトタイピング技法によらざるを得ない。一方、前者は必ずしも交換ハードウェア装置の動作を必要とせず解析的手法により可能である。

6．1　設計仕様に含まれ得る論理矛盾

ＴＳＧに矛盾があるということは、２つの場合が考えられる。ＰＳＧ自体が矛盾を有する場合と、ＰＳＧ間に既に矛盾がありＰＳＧ－ＴＳＧ変換の際に矛盾が作りこまれる場合である。

ここで、ＰＳＧ自体に矛盾があるとは以下の場合である。

①すべての或いは幾つかのトランジションを経由した後、初期状態（トークンの個数も含めて）に戻らない。（ループの関係）

なお、これは統合した結果のＴＳＧに引き継がれ得るものであり、また新たに作成されることによって存在し得る矛盾である。

ＰＳＧ間に矛盾があるとは以下の場合である。

②表現の詳しさがＰＳＧ間により異なる。（一方が詳細で、他方が省略等）

③入力プレースの集合と動作が一致し、出力プレースの集合が異なる。

上記の場合、それぞれＰＳＧを作成する時、及び、ＴＳＧに統合する時に矛盾を除去することが必要である。

6．2　ペトリネットを利用した矛盾検出

上記①及び③についてペトリネットの行列を用いて解析を行うことにより、設計仕様の

検証・確認を行いつつＰＳＧ或いはＴＳＧを作成してゆく。

（３）式を利用してまず①の検証を行うことができる。例えば図３の場合、行列Ｄは以下のように表される。

$$D = D^{+} - D^{-}$$

$$= \begin{pmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} -1 & 2 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & -4 & 2 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 2 & 0 & 0 & 0 & -1 \\ 2 & 0 & 0 & -4 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

この時、初期状態からT_1，T_2，T_3，T_8が発火すると元の状態に戻り、マーキングが変化しないこと（保存性）は下式により示される。

$$\mu = \mu_0 + (1\ 1\ 1\ 0\ 0\ 0\ 0\ 1) \cdot D$$
$$= \mu_0 + (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$
$$= \mu_0$$
$$= (2\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1)$$

７．行列を利用した統合

ＴＳＧを作成する場合、ＰＳＧの一節からＴＳＧに変換・作成し統合していく。ここではそれを関数行列の各行各列を増加させていくことにより達成される。入力関数行列及び出力関数行列についてもプレースやトランジションが一致しないものを追加してゆくことにより、最終的にはＴＳＧの入力関数行列及び出力関数行列ができあがる。なお同種のリソースについては統合がなされる。

例えば入力関数行列D^-についてその作成過程を図６に示す。ＰＳＧ１の中のプレースP_1とP_{11}とは同一の端末ということで統合される。

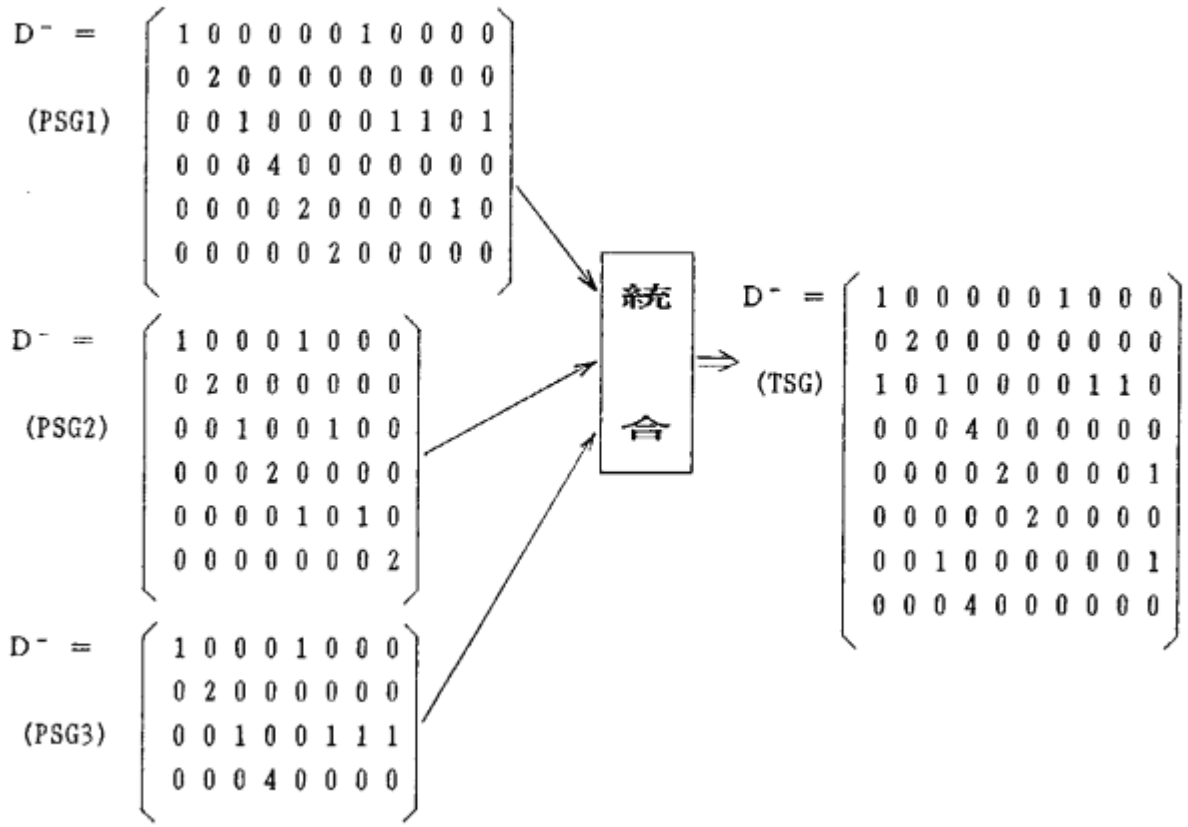


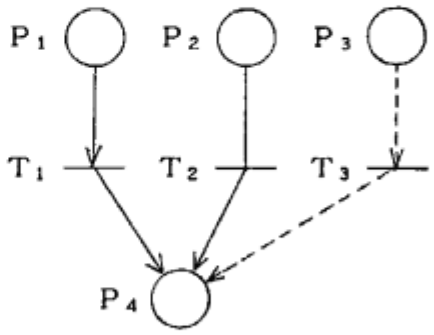
図６　入力関数行列の統合例

８．新サービスの提示

作成されたペトリネットグラフより、別々の要求仕様から作成されたものを元に新たなルートを作り上げることができる。つまり、初期状態は全リソースがアイドル状態であるとし、そこから発火可能なトランジションを（１）式により探索し、それらを次々と発火させてゆくことにより、入力時には考えていなかったルートを発見することができる。図７にその例を示す。$P_1 \rightarrow T_1 \rightarrow P_2 \rightarrow T_2 \rightarrow P_3$ 及び $P_4 \rightarrow T_4 \rightarrow P_5 \rightarrow T_5 \rightarrow P_6$ は入力であり、これを統合した結果、$P_2 = P_5$ であるとすると、図７(c)のようになる。この時、$P_1 \rightarrow T_1 \rightarrow P_2 \rightarrow T_5 \rightarrow P_6$ 及び $P_4 \rightarrow T_4 \rightarrow P_2 \rightarrow T_2 \rightarrow P_3$ という新たなサービスが作り出される（次頁図７参照）。

さらに、これは関数行列を利用したものではないが、知識工学の点からプレースを増加させることなく既にあるプレース間に新たなトランジションを設けることで新サービスを作ることができる（右図参照）。例えば次の場合である。

ダイヤル音を聞いている状態でオンフックできる。
呼出音を聞いている状態でオンフックできる。
↓
話中音を聞いている状態でオンフックできる。



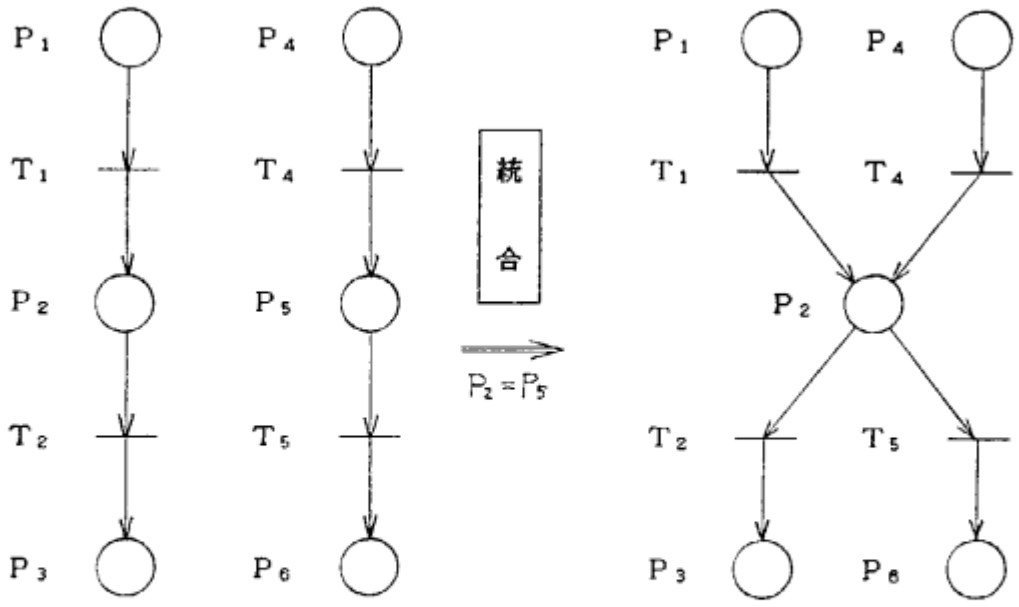


図7　新たなルートの発見

いずれにしても、これらのようにして作成されたサービスをＣＲＴ等に出力することにより、ユーザの気付かなかったサービスを提示することができる。そしてそのサービスを許容するかどうかはユーザの判断に委ねる。

9．おわりに

ペトリネットを用いて表現した設計仕様に関数行列を使用した解析を行って矛盾の検出、統合への利用及び新サービスの発見に適用した。特に総合的な設計仕様を作成する上で各種の矛盾を除去することは極めて重要であり、ここでとりあげた解析法は１つの手法である。行列を用いたペトリネットの解析には、行列は発火順序を規定しないこと、及び、そのすべての情報を表現していないこと等の問題点が残っており、検討の余地も大きい。さらに、本システムにおいても色つきトークンの導入については今後の課題である。

ペトリネットを知識表現に使用することは従来より行われており、また、仕様記述にも有効であると考えられる。今後はその接点について検討を進めていきたい。

なお、本研究は第５世代コンピュータ・プロジェクトの一環として行っているものである。

［参考文献］

[1] 青柳、長谷川、田中、柴田：”通信システムにおける仕様設計エキスパートシステムの一検討”、電子通信学会交換研究会資料、SE86-10、（1986）

[2] J.L.Peterson、市川・小林訳：”ペトリネット入門”、共立出版(1984)

[3] 田中、鹿野、山口：”通信システムに対するユーザ要求に含まれる矛盾の解消手法の一検討”、第３３回情報処理学会全国大会、5T-8、（1986）

[4] 長谷川、柴田、湯山：”通信システムにおける設計仕様の統合方式の一考察”、第３

３回情報処理学会全国大会、5T-9、（1986）
[5] ＣＣＩＴＴ勧告：" Functional Specification and Language(ＳＤＬ)"，ＣＣＩＴＴ ＲＥＤ　ＢＯＯＫ（1985）
[6] Ｍｕｒａｔａ、Ｚｈａｎｇ：" A High-Level Petri Net Model for Parallel Interpretation of Logic Programs"、ＩＣＣＬ'８６
[7] 熊谷、児玉：" 複合ネットによる分散システムのモデル化と動作検証"、電子通信学会第一回ネット理論時限研究会資料（1986）