マルチメディア通信と分散処理　50－10
（１９９１．５．１７）

交換ＳＤＬ仕様の代数的検証方式

長谷川晴朗　　菊地奈穂美

沖電気工業株式会社

本論文では,CCITT 勧告の仕様記述言語である SDL による通信システムの仕様の検証法として,交換システムを SDL で記述した設計仕様を状態の遷移に着目してペトリネットに変換した後,その数学的な解析手法を用いて検証する.ペトリネットへの変換後,それを構成するプレースとトランジションとの接続関係を示す接続行列から求められる２種類のインバリアントによって,ペトリネットの構造的な側面から有界性と活性に着目して仕様の検証を行う.さらに,設計にはインクリメンタルな仕様設計への対処が重要であり,仕様を追加する前に成り立っていた上記の諸性質がその追加によって失われていないことを数学的に検証する.最後に,以上の検証法を PBX の簡単な例について適用した結果について報告する.

A Method of Verifying a Switching SDL Specification by means of Petri Nets

Haruo Hasegawa　　Nahomi Kikuchi

Oki Electric Industry Co., Ltd.

4-11-22, Shibaura, Minato-ku, Tokyo 108, Japan

This paper describes a method of verifying a SDL specification of a swithing system by utilizing the algebraic properties of Petri Nets. In order to verify the SDL specification, it is converted into Petri Nets. Two kinds of invariants can be calculated from the incidence matrix of the created Petri Nets which means connections between transitions and places. Then, the boundness, the liveness and deadlock-freeness of the specification are verified by the invariants. Besides, during desinging incrementally, designers can ensure the correctness of the specification which was designed before they add new services to it, by checking the invariants.

1 はじめに

近年,通信システムが高度化・多様化・高速化してきており,それに従い通信システムを構築するソフトウェアが益々大規模になってきている.R.Balzer の唱える,新しいソフトウェア開発のパラダイムでは,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は自動的に変換できるとしている[1].つまり,仕様化工程における検証が一層その重大性を増加させている.このパラダイムに則って,
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勧告されている SDL[2] をベースとした,通信ソフトウェアの開発支援システムが各機関で開発されている[3]-[5].

SDL 仕様を検証するものとしては,SDL の文法に適うかどうかを調べる SDL チェッカ,SDL のままで実行してシグナルの流れや変数の値が設計者の思う通りになっているかを調べるシミュレータ等がある.さらに,一旦 SDL 以外の言語による記述に変換して,その段階で種々の性質を調べるものが各機関で考えられている.その中では,従来より非同期・並行系システムのモデル化・検証用ツールとしてのペトリネット[6] を利用したものが挙げられる[7]-[9].しかし,それらは,可達木を作成して到達可能性を調べたものであり,数学的な性質を用いて華麗に,あるいはエレガントに検証したものはないと言ってよい.本稿では,ペトリネットの接続行列から求めたSインバリアントとTインバリアントを利用して各種の解析を行うものである.

まず,Sインバリアントからは,全体のペトリネットがSインバリアントによって被覆されていることによって有界性が成立することを示す.次に,Tインバリアントからは,全体のペトリネットがTインバリアントで被覆されていることによって活性が成立することを示す.さらに,仕様はただ一回の設計で完成するものではなく,インクリメンタルに設計していくものであり,その場合仕様を追加する前のものが仕様追加によりその正当性を失っていないかをそれらのインバリアントが新しいペトリネットでも求められることより調べる.

本論文は以下の構成からなる.まず第2章で SDL の概要及び本論文で扱う SDL モデルについて述べる.第3章で SDL 仕様の検証項目について記す.次に第4章でペトリネットの基本用語及びそれを使用した解析手法について述べる.第5章で SDL からペトリネットへの変換方法について記す.第6章では,SDL から変換したペトリネット仕様をインバリアントを利用して解析する方法について述べる.第7章では,簡単な例についてその解析をしてみる.最後の第8章はまとめである.

2 通信システム用仕様記述言語 SDL

2.1 SDL とは

SDL は,CCITT より通信システムの要求動作 (required behaviour) の仕様化 (specification) と,実際の動作 (actual behaviour) の記述 (description) のための言語として勧告されており,各国の各機関で広く使用されている.SDL の記法には,コンピュータ処理が容易なテキスト形式の SDL/PR(textual Phrase Representation) と,設計者にとって理解の容易なグラフィカルな形式の SDL/GR (Graphical Representation) とがあり,基本的に1対1の相互変換が可能である.詳細の説明は参考文献を参照されたい.なお,理解の容易さから以下では基本的に GR の表現形式で SDL 仕様を表す.なお,ここでは紙幅の都合から本論文で述べようとする点について,誤解のない程度に SDL 図をかなり簡略化して記述し,必ずしも SDL の正しい文法に従ってはいないことを断っておく.

2.2 PBX モデル

SDL 仕様を検証するについては,筆者等が対象としている PBX (Private Branch eXchange) がどのような SDL モデルであるかを記述する必要がある.ここで対象とする SDL モデルは,端末からの操作によって交換システムがどのように状態遷移するかを表現するものである.従って,端末や PBX 内部のハードウェア及びソフトウェアといったリソース毎にプロセスを設け,それらのプロセス間のシグナルの送受により各プロセスが状態遷移していく方式としている.

ここでは,図1に示すように PBX をモデル化し,これに従って SDL 仕様が作成されている.このモデルでは,PBX を3ブロックに分ける.一つは,端末やトランクのように,PBX の外部に接続される装置を論理的に表現したものとそれらをサービスとして管理するものからなる call_processing (以下, cp と略す) ブロックであり,もう一つは,呼出信号送出回路や会議用トランクのような PBX 内のハードウェアリソース及びそれらを管理するものからなる resource(以下, rs と略す) ブロックである.最後のものは外部ハードウェア装置を監視・制御する device(以下, dv と略す) ブロックである.cp ブロックは,内線加入者の電話機端末を意味する ext プロセス,局線や専用線との接続インタフェースであるトランクを示す trk プロセス,及びそれらを管理する terminal_management (以下, tm と略す) プロセスからなる.rs ブロックも同様に,呼び出し信号を意味する rgt プロセス,会議トランクを意味する cft プロセス,及びそれらを管理する resource_management(以下, rm と略す) プロセ

スからなる. ext, trk, rgt, cft の各プロセスをリソースプロセスと呼び, tm, rm のプロセスを管理プロセスと呼ぶ. システムの生成時には, 管理プロセスのみが存在し, リソースプロセスのインスタンスはまだ存在していない. 呼が発生しリソースが必要となった時に対応するプロセスインスタンスが生成され, 呼が終了する等不要となった時に消滅する. 管理プロセスは, これらのインスタンスの生成・消滅管理のほかに, ブロック間通信制御を行う. dv ブロックは, ハードウェア・アーキテクチャに大きく依存するものであり, cp ブロックとの間でそれに応じた監視・制御信号がシグナルとして送受される. ただ, 本論文とは関連性が少ないため詳細の説明は割愛する.

図2に, 打ち合わせ通話と3者の会議通話を含む内線相互接続サービスに関する SDL 仕様を示す. 打ち合わせ通話は, 2者間の通話から他の内線加入者を呼び出して打ち合わせをした後, その相手のオンフック操作によりもとの通話に戻るサービスである. 3者会議通話とは, 打ち合わせ通話からフッキング操作により3者間の通話に入るものである. ただし, SDL 図をすべて表すことは紙幅の都合から無理なため, 図2では全体の流れを比較的わかりやすく表現している ext 及び rgt プロセス図のみを示す. なお, 同図で破線で囲んだ部分が3者会議に関する仕様である. 各プロセス図では, 接続されている相手側 (ext, rgt) の PId(Process instance Identifier) を知っているが, そのためのシグナルの送受を記していない. また, dv ブロックへのシグナルについても紙面の都合から割愛している.

3 仕様の検証項目

SDL 仕様の検証項目としては, 以下のものがあげられる.

(a) 無矛盾性　一つ一つの仕様の中に操作が不可能というような矛盾がないこと.

(b) 完全性　あらゆる可能な入力が処理され, かつあらゆる出力が受信されること.

(c) 有界性　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.

(d) 活性　状態遷移が実行可能であること.

(e) デッドロックのないこと　どの状態へも遷移できないことがないこと.

(f) 初期状態への復帰　あらゆる状態から初期状態に復帰できること.

(g) 要求充足性　ユーザ要求の仕様への反映を確認すること.

第6章で後述するが, 上記項目のうち (c), (d), (e) 及び (f) に関してインバリアントを用いて検証を行う.

4 ペトリネット

4.1 ピュアなペトリネット

ペトリネットは, 非同期・並行系システムのモデル化及び解析のためのツールであり, 現在各方面で使用されている. 構成は極めて簡単で, 2種類の接合集合 (プレースとトランジション) とそれらの間の有向枝 (アーク) からなるグラフである. 一般に, プレースは事象生起に関するシステムの条件を示し, トランジションは離散事象の生起と完了を表現する. そして, プレースに対応する条件が現時点で成立している場合には, そのプレース中にトークンを置き, トランジションの発火によるトークンの移動によりシステムの状態遷移を表す. 各プレースにトークンが設置された状態をマーキングという. 主要な用語の定義を以下に記す.

1. 構成要素　ペトリネットを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, アークの多重度を示す接続行列 (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 P から T への行列), マーキングである.

2. 発火条件・発火系列　あるマーキング M_0 で j 番目のトランジション (T_j) が発火するための条件は, T_j に入力する各プレースの中のトークンが, T_j とそれらのプレースとを結ぶアークの多重度以上に存在することである. 従って, j 番目のみが1でその他の要素がすべて0である単位列ベクトルを e_j, A^- の転置行列を A^{-T} とおいて, 次式が成立することが必要十分条件となる.

$$M_0 \geq A^{-T} \cdot e_j \qquad (1)$$

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ベクトルを γ とすると, トランジションの発火毎に, 各入力プレースからアークの多重度に等しい数のトークンが抜け, 各出力プレースにアークの多重度に等しい数のトークンが入ることから, (2) 式が成立する.

$$M = M_0 + A^T \cdot \gamma \qquad (2)$$

3. T- 完全性　${}^oT = T^o$ を満足するペトリネットを T- 完全という. ここで ${}^oT(T^o)$ は t の入力 (出力) プレースの集合である.

4. Tインバリアント　ある列ベクトルxが次式を満足するとき，これをTインバリアントと呼ぶ．

$$A^T \cdot x = 0 \qquad (3)$$

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までの発火回数ベクトルを表している．特に，Tインバリアントのうちで和分解不能なものを初等的Tインバリアントという．

本論文でサービスは，各リソースがアイドルであることを示す初期マーキングから再びそのマーキングに戻るまでのトランジションの発火系列を示す．つまり，Tインバリアントは一つのサービスにおける各トランジションの発火回数を意味する．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定義する．すべてのサービスは素サービスの線形結合，特に和による線形結合で表現できるため，素サービスをサービスの基本単位ととらえることができる．なお，Tインバリアントの存在は，発火系列存在のための必要条件であって十分条件ではない．

5. Sインバリアント　ある列ベクトルyが次式を満足するとき，これをSインバリアントと呼ぶ．

$$A \cdot y = 0 \qquad (4)$$

(2)式の両辺にy^Tを左から乗じると，$y^T \cdot A^T = (A \cdot y)^T = 0$であるから，$M_0$から到達可能なすべてのマーキング$M$に対して，$y^T \cdot M = y^T \cdot M_0 =$ 一定となる．つまり，プレース毎にある重み付けをしたトークンの総和が，マーキング状態によらず一定であることを示している．

4.2 色付きペトリネット[10]

色付きペトリネットとは，通常のペトリネットを拡張したものであるハイレベルネットの一つであり，プレース，トランジションやアーク等に識別子(色)を持たせて発火やトークンの出入りを制御するものである．ここでは以下の理由からピュアなペトリネットではなく色付きペトリネットを考える．

(a) 正確かつ簡明に仕様を記述できる．

(b) 様々なリソースを明確にモデル化できる．

(c) リソース間の関係を明示的に記述できる．

色付きペトリネットにおいても，4.1節で述べた種々の性質がピュアなペトリネットと同様に成立する．ただ，色付きペトリネットでは，インバリアントを求めるアルゴリズムについて最近ヒントとなるような研究こそなされている[11]ものの必ずしも確立されていない．ただ，後述するようにここで取り扱うものについてはピュアなペトリネットとして求めたインバリアントをほぼそのままの形で利用することが可能である。

本論文では，端末等のPBXの外部に接続する装置の状態遷移に着目して，サービス仕様の観点からペトリネットを考える．

5 SDLからペトリネットへの変換方式

5.1 基本的な変換ルール

SDLからペトリネットへの変換の基本的なルールを以下に示す[9),12)]．なお，ここでは，PBXを外部から見たサービス仕様の観点からとらえるため，プレースへの変換対象となるものは，extやrgtといったリソースプロセス内のステートである．SDLでは，1種類のプロセスは複数のインスタンスを持ち得るが，ペトリネットではインスタンスの概念がなく色付きペトリネットでそれを表現する．

1. ステートはプレースに変換する．ただし，必ずしも一つのステートを一つのプレースに変換するとは限らず，一つの安定した状態に含まれるものはそれらのステートをすべて集めて一つのプレースとする．

2. ステートとステートの間のインプット及びアウトプットはそれらを統合して一つのトランジションとする．ただし，データに関するものは検証の対象としないこと等により，タスクは基本的に何にも変換しない．

3. SDLではプロセス間のシグナルの送受は非同期通信に基づいており，同期通信を可能とするためwaitステートを設けることがあるが，これは図3に示すようにプレースとはしない．

4. プロセスの生成・消滅については，そのリソースを表すトークンのアイドルプレースを設け，そこからのあるいはそれへの状態遷移で表現する．

5.2 並列性の表現

ここで重要なことは，SDL仕様では複数個のプロセスが並列に動作可能であることを利用して，各リソース単位にプロセスが存在し，それらが通信を行いながら並列に動作していることである．それをペトリネットでは自然な形で表現している．

6 ペトリネットでの検証方式

第5章の検証方式を適用して,ハードウェア上のリソースの状態遷移に的を絞った,ペトリネットで記述されたサービス仕様を作成することができる.ここでは,ペトリネットの代数的な性質を利用した検証方式について述べる[13].3章で列挙した検証項目の中で,Sインバリアントを用いて(c)を,Tインバリアントを用いて(d),(e)及び(f)を検証する.

6.1 Sインバリアント

第4章で述べたようにSインバリアントは,トークンの保存性について規定したものである.従って,Sインバリアントの要素が0でないプレースは有界であることが言える.さらに,ペリネット全体がSインバリアントで被覆されていれば,ペリネット自体が有界であると言える.

6.2 Tインバリアント

第4章で述べたように,Tインバリアントは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再びその初期状態に戻るまでのサービスにおいて,トランジションの発火回数を示している.従って,このことからペトリネット全体がTインバリアントで被覆されていれば,すべてのトランジションが活性であると言える.

6.3 インクリメンタルな設計

我々がソフトウェアを設計する時,ゼロの状態から一気に全体を作成することはほとんどないと言ってよく,現実には少しずつ作成してはその仕様を検証し,その後再び設計を繰り返すことが多い.つまり,このようなインクリメンタルな設計を行う場合に注意しなければならないことは,追加設計した部分に誤りが含まれ得ることは充分考えられることとして,本来追加した箇所に関係がない筈の部分にまで誤りを生成してしまうことである.そのような場合,前述のインバリアントに関する諸性質を利用して追加する前のインバリアントを求め得るかどうかにより,追加の前に有していた部分が不変であることを検証することができる.

7 検証例

図2に示したものは,PBXにおいて内線相互接続を中心としたサービスについてのSDL設計仕様である.このうち,破線部内に示す3者会議通話サービスを除いたものを5.1節で記述した変換ルールによって図4に示す色付きペトリネット(点線部を除いたもの)に変換する.ここでは,プレースを楕円で,トランジションを太線で表している.また,アークの横にその上を通過するトークンの種類とその数を示す.トークンには,本ペトリネットでリソースである内線加入者と呼び出し音の2種類のものが存在するが,ここでは簡単のため後者のみに(rg)と付し,前者には特にそれを記さず複数個ある時にのみその数を示す.

このペトリネットについて以下のような検証を行うことができる.

7.1 Sインバリアント

図4における接続行列を図5(a)に示す.ここで,内線加入者の変数をX,Y及びZで表し,その全体集合をUで表す.なお,図5(b)は図4のペトリネットで色を無色にしたものについて求めた接続行列である.図5(a)の接続行列のSインバリアントを求めると,図6に示すように2個のSインバリアント,y1及びy2が得られる.また,これらのSインバリアントから導かれるトークンの保存性について成立する等式を同図に示す.ここで,M(Pi)はプレースPiに存在するトークンを示す.これらの列ベクトルのすべてについてその要素が0であるプレースは存在しない.つまり,ペトリネットはSインバリアントで被覆されることがわかる.従って,このペトリネットは有界である.

y1, y2は,それぞれ内線加入者,呼び出し音のリソースについて有界であることを示している.また,図5(b)について求めたSインバリアントからも全く同様の結果が得られる.

7.2 Tインバリアント

7.1節と同じように図5の接続行列から図7に示すように5個のTインバリアントが得られる.4.1節で記したように,Tインバリアントは基本的にある状態から再びその状態に戻るまでのトランジションの発火回数を示している(発火系列は特定しない).各Tインバリアントが示す発火系列を同図に示す.従って,このTインバリアントのうち,x1, x2, x3は,特に初期状態から再び初期状態に戻るサービスである素サービスであり,x4, x5はそうではないサービスを示している.例えば,x4は2者通話中からフッキングしてダイヤル中に再びフックキングによってもとの2者通話中に戻るものである.この後者のサービスが開始されるマーキングが素サービスの中で存在し得るマーキングであることを前提として以下のことが言える.$^{\circ}T = T^{\circ}$が成立することにより,全トランジションが活性であるためには,ペトリネットがTインバリアントにより被覆されていればよい.図7に示すTインバリアントのすべてについてその要素が0であるトラ

ンジションは存在しない. つまり, トランジションはすべて活性である.

7.3 インクリメンタルな設計

図2に示すように, 打ち合わせ通話を含む内線相互接続サービスに, 破線部に示す3者会議電話を行うサービスを追加することを考える.

上記のサービスを追加する際に考慮しなければならないのは, 追加するサービスは各リソースのアイドル状態から再びアイドル状態に戻ることを考えているのではなく, 既に作成された仕様にないものだけを追加すればよいことである. つまり, 内線相互接続状態から他内線を呼び出しその応答後2者間の通話をしている時に, フッキング操作をした後3者間の会議通話に入る部分だけを追加すればよいのである. SDL レベルで言えば, ext プロセスで内線相互通話中のステートからフッキングした後, 他内線に発信してその応答後, 通話に入って再びフッキング操作により3者の会議電話に入る. この時, シグナルの送受は別としてプロセス図で大きく変更又は追加されるのは ext プロセスと cft プロセスである.

この3者会議を含むSDL図からペトリネットに変換したものが図4 (点線部を含む) である. 例えば, ここで最後に呼び出された内線の解放処理が抜けていた場合 (図2で一点鎖線部で示す), ペトリネットに変換してTインバリアントを求めてもトランジション T14 に相当するものがなく, P10 及び P11 が S インバリアントに被覆されなくなるため誤りであることがわかる. また, 新しいペトリネットで両インバリアントを求め, 従来から存在したインバリアントを包含することを確かめることにより変更前の仕様をそのまま引き継いでいることを確認できる.

8 おわりに

SDL仕様をペトリネットを用いて代数的に検証する方式について述べた. ソフトウェア開発全体の工程に占めるその重要性から, 他言語を使用しての仕様化段階での検証は極めて意義が大きいと考える. 特に, 状態遷移に的を絞って有界性と活性の検証を行えることを明らかにした. 今後, さらにその検証範囲の拡大を目指していく予定である.

謝辞

日頃, 貴重な討論をして戴く当社電子通信事業本部ソフトウェア技術開発部の諸氏に感謝する.

文献

(1)R. Balzer: Software Technology in the 1990's: Using a New Paradigm, IEEE Computer, Vol.16, No.11, pp.39-45 (1983)

(2)CCITT Recommendation: Z.100: Functional Specification and Description Language (1988)

(3)H. Ichikawa, M. Itoh and M. Shibasaki: Protocol-Oriented Service Specifications and Their Transformation into CCITT Specification and Description Language, IEICE Trans. E-69, No.4, pp.524-535 (1986)

(4)K. Miyake, Y. Shigeta, W. Tanaka and H. Hasegawa: Automatic Code Generation from SDL to C++ for an Integrated Software Development Support System, Proc. of 3rd FORTE, pp.677-680 (1990)

(5)Y. Wakahara, Y. Kakuda, A. Ito and E. Utsunomiya: ESCORT: An Environment for Specifying Communication Requirements, IEEE Trans. Software Eng., Vol.6, No.2, pp.38-43 (1989)

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

(7)G. Comparin, G. A. Lanzarone, W. Panzeri and A. Torgano: Analysis of SDL Specifications using Petri Nets Techniques, Proc. of 2nd SDL Forum, pp.1-12 (1986)

(8)E. Kettunen and M. Lindqvist: Towards Practicality of Predicate/ Transition Petri Net Reachability Analyusis of SDL, Proc. of 3rd SDL Forum, pp.285-294 (1987)

(9) 宗森純, 武田捷一: ペトリネットによるSDLの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)

(10)K. Jensen: Coloured Petri Nets, Petri Nets: Central Models and Their Properties. Advances in Petri Nets 1986-Part I, Lecture Notes in Computer Science, Vol.254, Springer-Verlag, pp.248-299 (1987)

(11) 上田佳寛, 本坊正浩: CP-nets における P-invariant の解法, 電子情報通信学会, 回路とシステム研究会, CAS90-98 (1990)

(12) 田口毅, 安藤津芳, 長谷川晴朗: 中間言語にペトリネットを用いたSDL仕様設計の一方式, 第39回情処全大, 6S-3 (1989)

(13)G. Berthelot and R. Terrat: Petri Nets Theory for the Correctness of Protocols, IEEE Trans. Commun., Vol.30, No.12, pp.2497-2505 (1982)

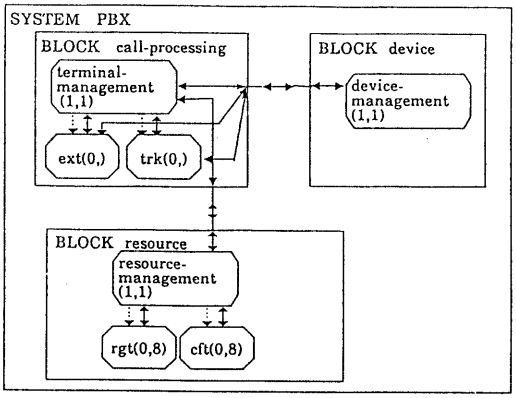

図1. PBXのSDLモデル

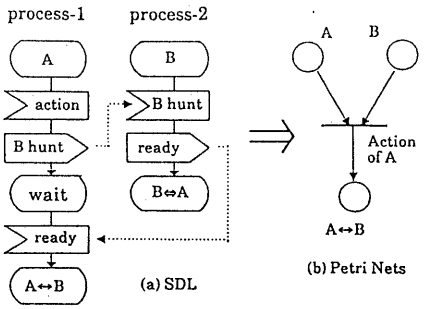

図3. 並列性を持つSDLの変換例

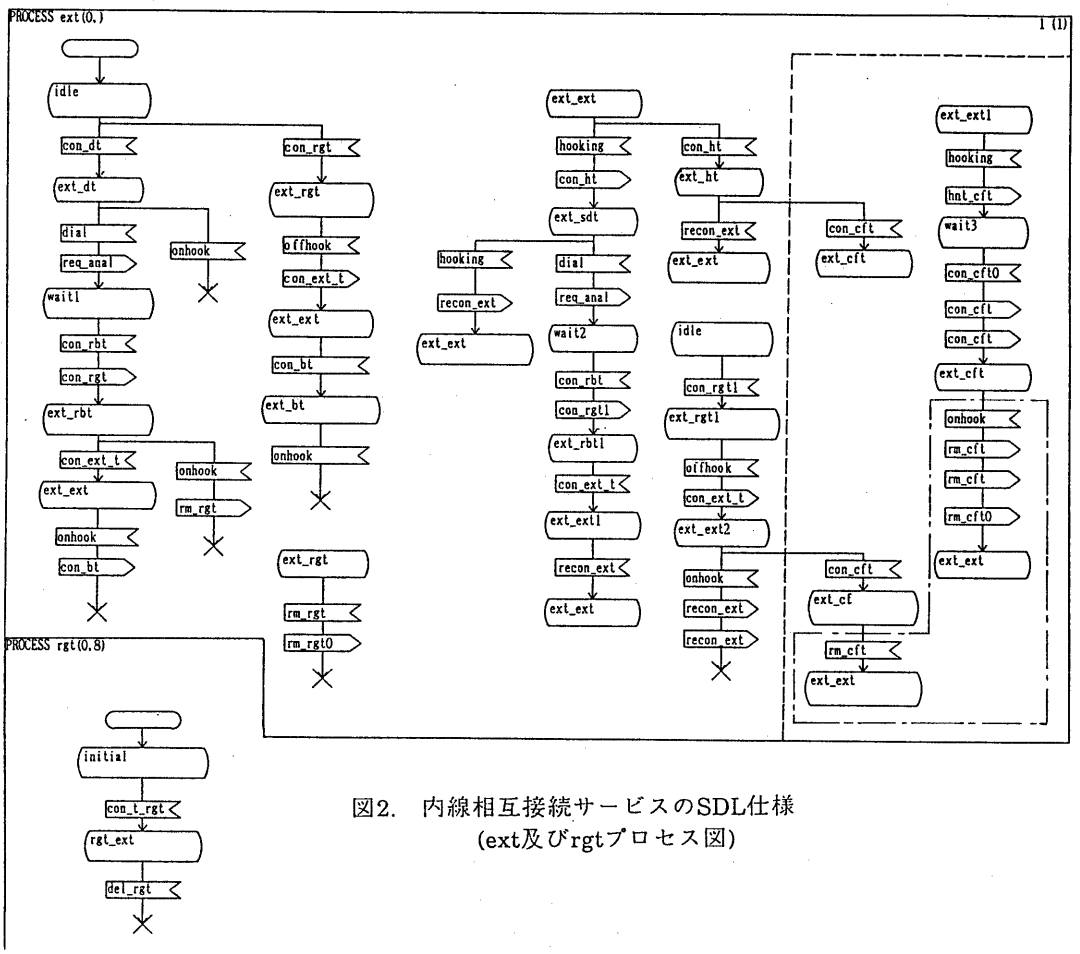

図2. 内線相互接続サービスのSDL仕様
(ext及びrgtプロセス図)

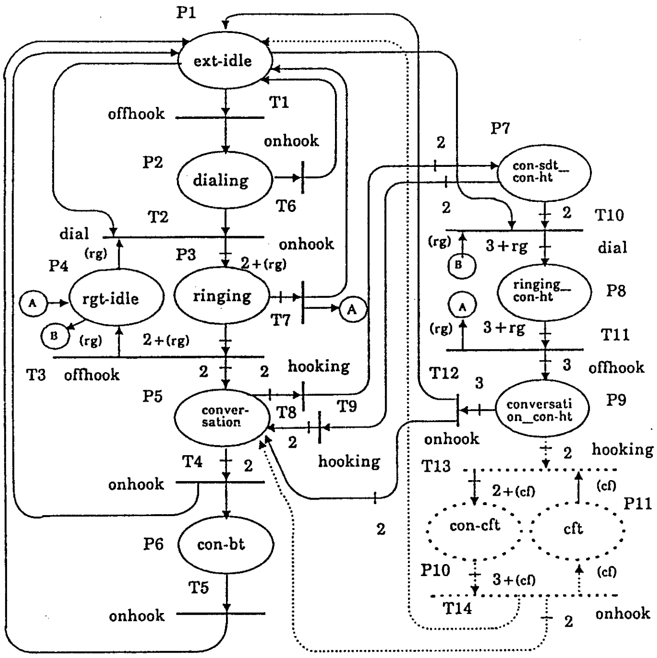

図4. 図2のSDL図から変換したペトリネット
(ただし，点線部は3者会議に関する部分)

$A =$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
T_1	-X	X	0	0	0	0	0	0	0
T_2	-Y	-X	Q	-rg	0	0	0	0	0
T_3	0	0	-Q	rg	P	0	0	0	0
T_4	Y	0	0	0	-P	X	0	0	0
T_5	X	0	0	0	0	-X	0	0	0
T_6	X	-X	0	0	0	0	0	0	0
T_7	P	0	-Q	rg	0	0	0	0	0
T_8	0	0	0	0	-P	0	P	0	0
T_9	0	0	0	0	P	0	-P	0	0
T_{10}	-Z	0	0	-rg	0	0	-P	S	0
T_{11}	0	0	0	rg	0	0	0	-S	R
T_{12}	Z	0	0	0	P	0	0	0	-R

$P = X + Y, \quad Q = X + Y + rg$
$R = X + Y + Z, \quad S = X + Y + Z + rg$
$X, Y, Z \in U$(端末の集合)

(a)色付き接続行列

$A =$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
T_1	-1	1	0	0	0	0	0	0	0
T_2	-1	-1	3	-1	0	0	0	0	0
T_3	0	0	-3	1	2	0	0	0	0
T_4	1	0	0	0	-2	1	0	0	0
T_5	1	0	0	0	0	-1	0	0	0
T_6	1	-1	0	0	0	0	0	0	0
T_7	2	0	-3	1	0	0	0	0	0
T_8	0	0	0	0	-2	0	2	0	0
T_9	0	0	0	0	2	0	-2	0	0
T_{10}	-1	0	0	-1	0	0	-2	4	0
T_{11}	0	0	0	1	0	0	0	-4	3
T_{12}	1	0	0	0	2	0	0	0	-3

(b)無色の接続行列

図5. 図4に示すペトリネットの接続行列

$$\begin{array}{lccccccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 \\ y_1 = [& 1 & 1 & f_{13} & 0 & 1 & 1 & 1 & f_{18} & 1]^T \\ y_2 = [& 0 & 0 & f_{23} & 1 & 0 & 0 & 0 & f_{28} & 0]^T \end{array}$$

y_1より、$M(P1) + M(P2) + f_{13} \cdot M(P4) + M(P5) + M(P6) + M(P7) + f_{18} \cdot M(P8) + M(P9) = \sum U$

但し、$f_{13} \cdot (X + Y + rg) = X + Y$, $f_{18} \cdot (X + Y + Z + rg) = X + Y + Z$

y_2より、$f_{23} \cdot M(P3) + M(P4) + f_{28} \cdot M(P8) = \sum rg$

但し、$f_{23} \cdot (X + Y + rg) = rg$, $f_{28} \cdot (X + Y + Z + rg) = rg$

図6. 図4に示すペトリネットのSインバリアント
及びトークンの保存性

$$\begin{array}{lccccccccccccl} & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} & T_{11} & T_{12} & \\ x_1 = [& 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^T & \cdots T1 \to T2 \to T3 \to T4 \to T5 \\ x_2 = [& 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0]^T & \cdots T1 \to T6 \\ x_3 = [& 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0]^T & \cdots T1 \to T2 \to T7 \\ x_4 = [& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0]^T & \cdots T8 \to T9 \\ x_5 = [& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1]^T & \cdots T8 \to T10 \to T11 \to T12 \end{array}$$

図7. 図4に示すペトリネットの初等的な
Tインバリアント及びその発火系列