〔公　開〕

ＴＲ－Ｃ－０１３１

通信システム仕様の要求理解における高速計算方式

上田　佳寛
Yoshihiro UEDA

１９９６　２．２９

ＡＴＲ通信システム研究所

通信システム仕様の要求理解における高速計算方式

上田 佳寛

ＡＴＲ通信システム研究所

目次

1 はじめに

要求獲得からプログラム仕様作成, プログラム開発, 試験, 運用に関わるソフトウェアの開発全工程を一元的に支援する環境が必要とされている.

我々も上記問題点の解決に対し, 自然言語による要求から形式的な要求仕様に変換するフェーズ, 要求仕様の誤りや矛盾を除去するフェーズ, さらに設計仕様 (SDL) を生成するフェーズという 3 フェーズに分けて研究を行なっており, さらにユーザが記述する自然言語から設計仕様を自動的に出力するソフトウェアの自動生成の研究を行なっている. この研究により, ソフトウェア開発に関わる工数を飛躍的に短縮できると期待される.

本論文では要求仕様の誤りや矛盾を除去するフェーズの問題点, 解決方法について述べる.

FSM や PN 等ので記述された仕様記述の検証に関しては, 強力な検証手法が提案されているが [3], 現状では個々の単体のツールであり, それを統合的な環境での利用を考えられている文献はない.

統合的な支援環境下の要求仕様検証において PN 等を利用するためには以下の 3 つの技術課題を解決する必要がある.

1. 数学的検証能力のある言語 (数学モデル) への自動的な変換および逆変換
   要求仕様に含まれる矛盾や誤りの有無を検証するためには, 矛盾や誤りを定義できかつその矛盾や誤りを抜けなく検出できる必要がある. そのためには要求仕様を数学モデルへの自動的に変換する手法が必要となる. このような数学的背景を持つ言語にプロセス代数を基本とする LOTOS[1][2] もあるが, ここではペトリネット [3][4] の利用を考慮する.

2. 数学モデルと要求仕様との関係の把握と検証項目
   数学モデルでの検証項目が要求仕様におけるどのような性質に当たるのか, および要求仕様における検証するべき項目すべてが数学モデルでの検証ができるのかについて検討する必要がある.

3. 検証に関する計算量問題の回避手法
   reachability[5] や liveness[5] といったものの検証の計算量は NP- 完全であり, 現状の計算機では, 検証できないという結果になりかねない. また, 一般には, あらゆるシステムにおいて, その動的性質を研究するためには, すべての可達集合 (す

べての起こり得る相互作用の集合) を求めることがもっとも有効な手段ともされている. そのため, 計算量問題を回避する手段が必要になってくる. ここでは, 要求仕様の対象とするシステムの性質を利用して計算量問題を回避する手法を検討する.

2 諸定義

まず, この論文で使用する諸定義について述べる.

2.1 プロダクションシステム [6]

$x_i \in X, X =$ { 変数集合 }
$P_i \in P, P =$ { 述語記号の集合 }
$a_i \in A, A =$ { 個体集合 } および
$\rightarrow$(状態遷移記号) で定義されるルールの集合である. □
e.g.
$P_1(x_1), P_2(x_1, x_2) \rightarrow P_3(x_1, x_2)$
上記のルールは $P_1(x_1), P_2(x_1, x_2)$ に対応する
$P_1(a_i), P_2(a_i, a_j)$ が存在する場合, $P_3(a_i, a_j)$ に遷移可能であることを意味している.

2.2 HLPN

HLPN の定義は文献 [7] に詳しい. プロダクションシステムにおける変数 x はアークの色に対応し述語 P はプレースに個体 a および個体の組 (a_i, a_j) はトークンに対応する. 個体 a をカラー要素呼ぶ時もある. また, 状態遷移記号はトランジションである.

2.3 状態

1. 実状態
   実状態とは, HLPN におけるマーキングに対応するものである. □
   起こり得る実状態数はトークン数の指数オーダになる.

2. 広域状態
   広域状態とは, 2 変数以上の述語により関係づけられた変数の状態のことをいう. □
   この状態の意義は, 並行独立に遷移するトークンの状態は別の状態とみなすことで並列処理の解析に利用可能となる点である. その上, 状態検索空間を狭めるという副作用を満たす.

   例えば, A さんと B さんが接続されている状態と C さんと D さんが接続されているという 2 つの通話が同時に発生しても, その通話間に関係がなければ別の状態とみなすことができるということである.

2.4 連鎖 [8]

ある**広域状態** N, N_1, N_0', N_1' が与えられ,

$$N, N_0' \rightarrow N_1, N_1'$$

かつ

$$N \subset N_1$$

のとき, **連鎖**があるといい, $(N_1 - N)$ を**連鎖状態**という. ただし, N_0', N_1' が $\emptyset$ を含む. □

連鎖があれば, 広域状態 N_0' が存在する間は, 連鎖を起こした同じ発火系列が発火可能となり, N_k (N に k 個の連鎖状態が付け加わった広域状態) が存在する.

2.4.1 強連鎖 [9]

広域状態 N に含まれる個体集合 A_1, 広域状態 N_1 に含まれる個体集合 A_2 かつ N と N_1 が連鎖となる時, $A_1 \subset A_2$ であれば, N と N_1 との関係を強連鎖という. □
強連鎖の例を図 1 に示す.

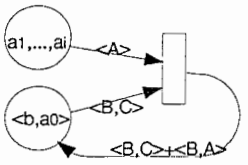


図 1: 強連鎖

2.4.2 弱連鎖 [9]

広域状態 N に含まれる個体集合 A_1, 広域状態 N_1 に含まれる個体集合 A_2 かつ N と N_1 が連鎖となる時, $A_1 = A_2$ であれば, N と N_1 との関係を弱連鎖という. □
弱連鎖の例を図 2 に示す.

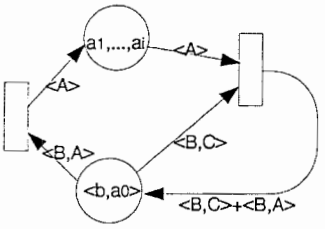


図 2: 弱連鎖

2.4.3 放射型連鎖 [9]

放射型連鎖とは 2 変数の色つきトークンにより実状態が放射上に接続する関係のことをいう.□

例えば,

$$P_1(A_0, A_1), P_1(A_0, A_2), \cdots, P_1(A_0, A_n)$$

のような関係のことをいう.

通信システムサービスでは 3 者間以上の通信の際に放射連鎖が存在する場合がある. これは, 2 者間通話から 3 者間への遷移が存在する場合, 3 者間通話は同時に, 2 者間通話状態を一部に持っていることにより発生する.

例えば, 図 3 のような場合である.

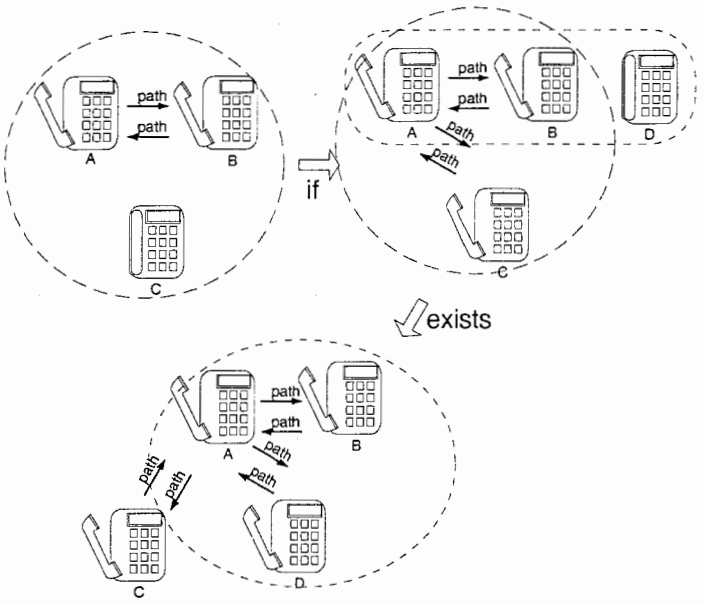


図 3: 放射連鎖

2.4.4 直列連鎖 [9]

直列連鎖とは，2変数の色つきトークンにより実状態が直列に接続する関係のことをいう.□

例えば,

$$P_1(A_0, A_1), P_1(A_1, A_2), \cdots, P_1(A_{n-1}, A_n)$$

のような関係のことをいう．直列連鎖は上記の3者間以上の通信の際にも発生するが，鉄道モデルでも起こり得る．図4

放射連鎖と違いは，連鎖対象の状態が常に変化する点である．例えば，放射連鎖では

$$P_1(A_0, A_1), P_1(A_0, A_2)$$

に接続する状態が増えていくことによる連鎖であるが，直列連鎖では

$$P_1(A_{n-i-1}, A_{n-i}), P_1(A_{n-i}, A_{n-i+1})$$

に状態が増えていく連鎖である．

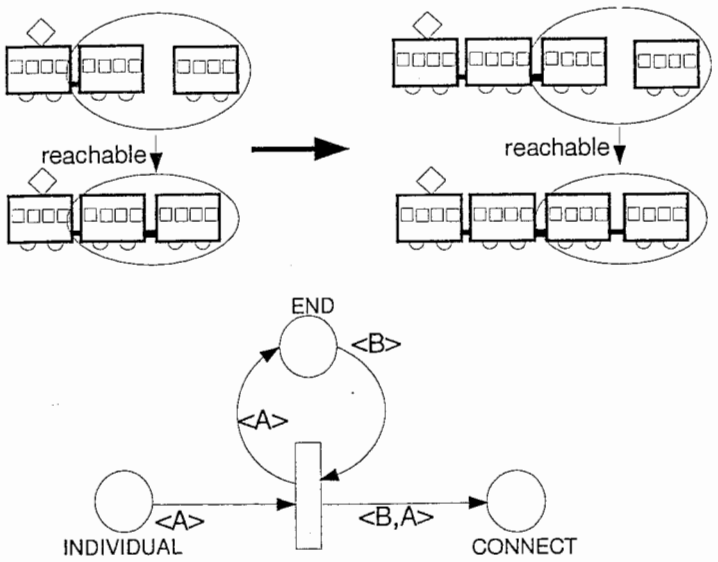


図 4: 直列連鎖をなす状態遷移

3 要求仕様からペトリネットへの変換 [10]

要求仕様はプロダクションシステムで定義している．If-Then 形式で記述するプロダクションシステムは，人間の断片的な知識を表現する上で，馴染みが良く，モジュール性に優れているためサービスの追加が容易に可能となる．ところが，大規模システムにおいてはルール間での予期しない相互作用が起こる可能性があり，誤った動作を起こすことがある．また，起こり得る相互作用を求めることは，一般にはすべてのルールを適応させるという総当たり的な手法しかない.[11][12][13] この手法では組合せ爆発を引き起こすため，大規模システムでは不可能とされている．そのため，数学的背景を持つペトリネット等に変換し，総当たり的な手法ではない相互作用の判別手法が必要となる．

3.1 要求仕様記述

まず，プロダクションシステムを基本とする STR(State Transition Rule)[14] について簡単に定義する．

STR 記述

STR 記述は 「現状態」，「イベント」，「次状態」の 3 つの部分から構成される．状態は，端末の状態と，適当な 2 端末間の関係で表現される．状態は，状態要素から構成される．状態要素は，属性の違いによる固有の名称と，端末対応のための変数を持つ．状

態要素が持つ端末対応のための変数には, 1 変数のものと, 2 変数からなるものがある. どちらのものも, 第一変数には動作対象の端末を記述する. 第二変数は, 第一変数の端末に束縛されているという関係を示す.

例えば, 以下のように記述される.

$$ringback(B, A),\ ringing(A, B)\ offhook(A) :$$

$$path(A, B),\ path(B, A).$$

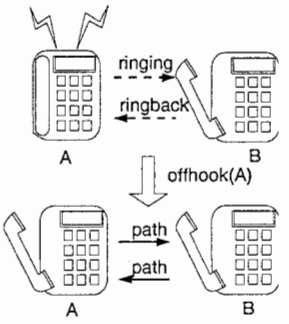


図 5: STR 記述と状態遷移イメージ

図 5の規則は「端末 A が 端末 B から呼び出しを受けている時, 端末 A が offhook すると, 端末 A, B 間で通話状態に遷移する」ことを規定している.

図 5のようにユーザにとって視覚的に判断ができる範囲での記述となるため, 通信サービスについてスキルの乏しいユーザであっても無理なく記述できるものである.

また, このように端末の状態遷移のみの記述であるため, 以下の制約が成り立つことがわかる.

[**第一変数保存**] 現状態に存在するすべての状態要素の第一変数は, 次状態のある状態要素の第一変数に含まれなければならない. また, 次状態のすべての状態要素の第一変数は, 現状態の状態要素の第一変数に含まれなければならない. □

e.g.

$P1(A),\ P2(B)\ e1(A, B) :\ P3(A),\ P4(A),\ P4(B).$

$P5(A, B),\ P6(B, A)\ e2(A) :\ P1(A),\ P2(B).$

上記の記述は, 第一変数保存の法則を満たす. 以下に, この法則を満たさないものを示す.

$P1(A),\ P2(B)\ e1(A, B) :\ P3(A),\ P4(A).$

(次状態に B を第一変数に持つ状態要素がない.)

$P5(A,B)\ e2(A):\ P1(A),\ P2(B).$

(現状態に B を第一変数に持つ状態要素がない.)

- 物理的意味

  第一変数は端末の実態を表している. もし, この条件が満たされないとすると, 端末がなくなるということか, あるいは端末が発生するということになる. これは, 現実には矛盾とみなせるためこの法則は妥当であるといえる.

[**自由変数の排除**] 現状態に出現する変数以外が次状態に現れてはいけない. いい変えれば, 現状態で拘束されていない変数が次状態に現れてはいけない. □

この法則に矛盾する場合, 端末の特定が不可能である. 以下に例を示す. e.g.

$P1(A)e1(A):P3(A,B).$

(現状態にない B が次状態に現れている.)

例えば, NTT に契約している数千万端末のうち, どの端末が B となるのか対応つかない.

これらの制約は, ユーザの記述ミスのチェックに利用することもできる.

3.2 ペトリネットによるサービスモデル

プロダクションシステムの実行および解析にもっとも時間がかかるのが, 条件のパターンマッチである. その解決策として, Forgy はプロダクションシステム言語 OPS5 に対し, RETE[9] と呼ばれる高速アルゴリズムを提案している. これはプロダクションシステムでのモデルを拡張ペトリネットに変換することによる高速なパターンマッチアルゴリズムである. 本稿では, プロダクションシステムをカラードペトリネット (ここでは HLPN[7]) に変換することを試みる.

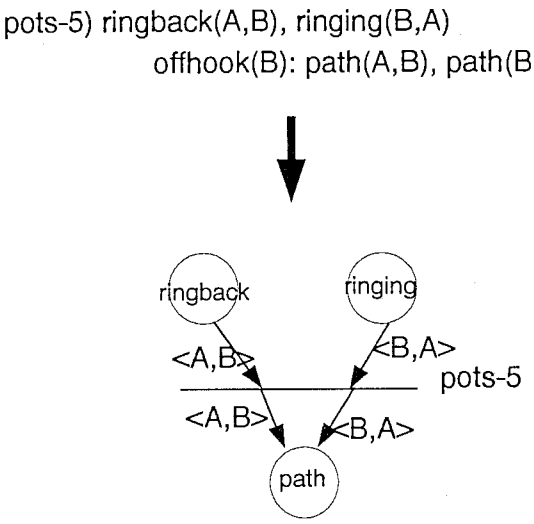


図 6: STR から変換された HPN の例

プロダクションシステムの仕様から HLPN への変換は, 以下の手順を踏む.

- ルールをトランジションに変換する.
  一つのルールを一つのトランジションに変換する. この時, イベントの変数を, トランジションに対するカラーとする. このことより, 仕様上のルールの数と同様のトランジションを有するネットが作成される. このことは, 任意のサービスを構成するルールとそのヒット回数を, 発火したトランジションとその発火回数で保証することを意味する.

- 状態要素をプレースに変換する.
  状態要素名をプレース名としてプレースを作成する. ここでは同じ状態要素名を

持つものは, 一つしか作成しない. なぜなら, 状態要素は個々の端末の状態を表し, 同じ状態要素名で規定される端末の状態は同一視できるためである.

- アークに色をつける.
  状態要素が持つ端末変数をトランジションとプレース間のアークとして表現する.

このフェーズまでで, 一つのルールにおけるネットが作成されたことになる. カラーの導入により, トランジションの発火条件がルールのヒット条件と等価になり, トランジションの遷移後のマーキングがルールのヒット後の状態と等価になっていることが理解できる.

- すべてのルールから変換した PN を結合する.
  これは, 同じ名称を持つプレースを機会的に統合することにより実現している.

ここで, 作成された, ネットが与えられたルールを完全にシミュレートできることは容易に理解できる.

図 6に STR から HLPN に対応させた例を示す.

4　ペトリネットでの検証

ここでは, 検証項目およびその計算量問題の回避手法について述べる. まず, 要求仕様に求められる性質について明らかにする.

4.1　通信システムサービスの性質

通信サービス特有の性質として, 以下のものがあげられる.

1. サービスは何度も (無限に) 利用可能である.[15]
   通信システムにおけるサービスは, 使い捨てられることがなく, 何度も (無限に) 使用されるという性質を持つ. そのため, サービスを状態変化の集合としてとらえると, あるサービスに存在するすべての状態は, 無限に出現可能でなければならない. また, サービスを連続した状態とすれば, 任意の状態からそのサービスの初期状態に可達でなければ, そのサービスを享受できない. このことから, 以下のことがいえる.
   サービスを端末の状態変化としてとらえる場合
   ある端末状態が出現したら, その端末状態から可達なすべての状態から元の状態に可達でなければならない.(可逆性)
   ある状態を与えた場合
   サービスにおけるすべての遷移が潜在的に無限回起こり得なければならない. (活性)

2. トランジションの発火により端末が増えることはない.
   遷移の途中で端末が増えるということは, 端末が発生するということである. これは, 現実には矛盾とみなせる. また, カラートークンは端末または端末間の関係をとして対応づけているため, ネット全体でカラー数は無限に増えることはない. (有界性)

これらの性質は, 通信システムだけでなく, FA や鉄道モデル等の離散事象システムにも適用可能である.

4.2　ペトリネットによる解析手法

ペトリネットを導入することにより, 対象システムの性質を定式化可能となる. それらの性質を利用することによる総当たりによらないサービスの検証方式を提案する.

まず, HLPN からカラーに関する情報を除いたネットにおける検証を行う.

このペトリネットは, 可達性, 可逆性, 活性の必要条件を与え, 有界性の十分条件を与える. 基本概念としては, 以下のことを利用している.

- HLPN の構造のみを取り出した PN での検証の有効性

  一般に, HLPN の検証は困難かつ PN の得意とする接続行列での解析を行なえないので, HLPN を PN に変換し, 変換した PN 上で解析する手法がとられている. しかし, その場合のボトルネックとして以下の 2 点があげられる.

  1. HLPN と等価な PN への変換の困難さ
  2. 変換前の HLPN よりはるかに大きい PN が生成されることによる検証コストの問題

  そのため, 縮約により解析効率を向上する手法を提案 (図 7 C1) している論文もある. [5][16] しかし, 依然として前述のボトルネックは解消されていない. そこで, 本論文では, HLPN からカラーを抜いた PN により解析する手法 (図 7 C2) を採用する. この手法の利点は, PN への変換が容易であることであり, 変換前の HLPN とネットの規模が等しい PN が作成されることであり, さらにはその PN での解析結果が HLPN の必要条件を満足している点である. つまり, この PN は効率のよい変換と縮約を行なった PN であるともいえる.

- トークン数に依存しないノード数の状態木で, 起こり得るすべての状態を検出可能

  例えば, 図 8 のペトリネットでは, すべての可達集合を求めるためには n ノードの状態木を生成する必要がある. しかし, マーキング $(n-i, i)$ の存在が状態生成

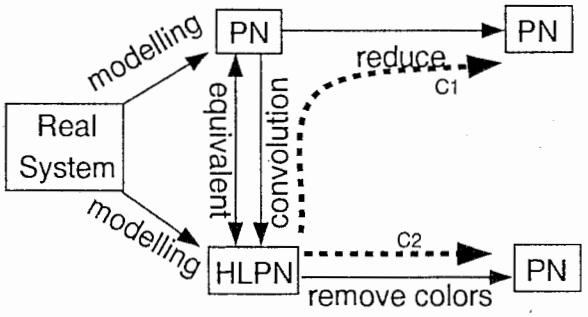
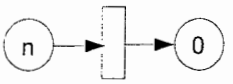


図 7: HLPN と PN の関係



(n, 0) -> (n-1, 1) -> ... -> (0, n)

図 8: PN and State tree

なしに理解可能. (ただし, $0 < i < n$)
パーシステントネットあれば, 可達問題が多項式時間で可解である必要十分条件であることが知られている. 上記のネットもパーシステントであることは容易に理解できる. しかし, 通信システムサービス仕様のようなカラーを持つ HLPN のパーシステント性を調べることは困難であるが, 通信システムのようにあらかじめネットが有界かつ可逆的であることを利用すると, ある種の繰り返しパターンが存在することが理解できる. この繰り返しパターンを検出することにより PN ネットと同様に状態生成なしにすべての状態が理解可能となる.

この状態生成の縮小法によりトークン数の指数オーダの計算量の状態検索がプレース数の指数オーダの計算量で状態検索が行なえる. (一般にトークン数は不定: 日本における通信網の端末数は数千万個にもなる. しかし, プレース数は高々数千個である.)

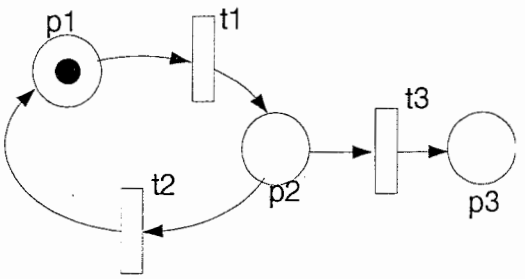
4.3 PN におけるデッドロックの判別

ここでのデッドロックとは, 非活性になるトランジションの存在のことをいう. 通信システムサービスは活性, 有界でかつ可逆的でなければならないので, デッドロックフリーを保証することが必要である.

そのため, PN の検証として 3 種類の検証項目を調べている.

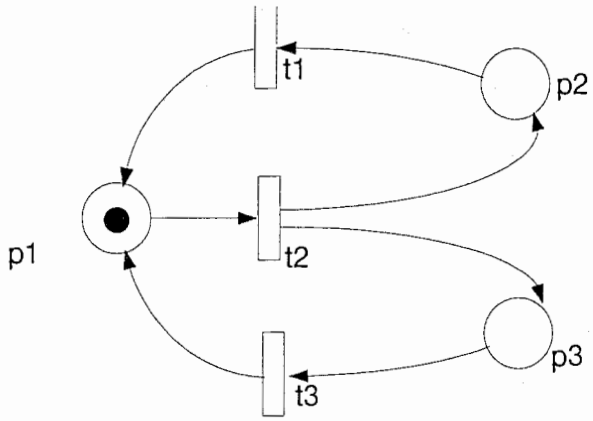
- すべてのトランジションは T- インバリアントの台集合に含まれていること
- すべてのプレースは P- インバリアントの台集合に含まれていること
- トラップおよびサイフォン

トークンの色を考慮しないネットの場合, 上記の条件を調べることで, 活性, 有界および可逆性を保証することができる. (ただし, 必要な初期マーキングが存在する場合.) 逆に, 上記の検証項目から以下のことがわかる.



T-インバリアント=(1 1 0)^T
台集合={t1, t2}

図 9: T- インバリアントの台集合に含まれないトランジションを持つネットの例



S-インバリアントなし

図 10: S- インバリアントを持たないネットの例 (非有界ネットの例)

1. T- インバリアントの台集合に含まれないトランジションは存在しない.

   (T- インバリアントの台集合に含まれないトランジションが存在するならば, そのトランジションの発火後に初期マーキングに戻る遷移はない.
   図 9に T- インバリアントの台集合に含まれないトランジションを持つネットの例を示す.)

   図 9において $t3$ の発火により初期マーキングに遷移不可能になる.

2. P- インバリアントの台集合に含まれないプレースが存在するならば, そのプレースは有界でないか保存的ではない. (図 10に有界でないネットの例を示す.)

3. ネットがサイフォンを持てば, そのサブネットは必ずトラップを持つこと.
   (これは, デッドロックを持たないことの必要条件になっている.)

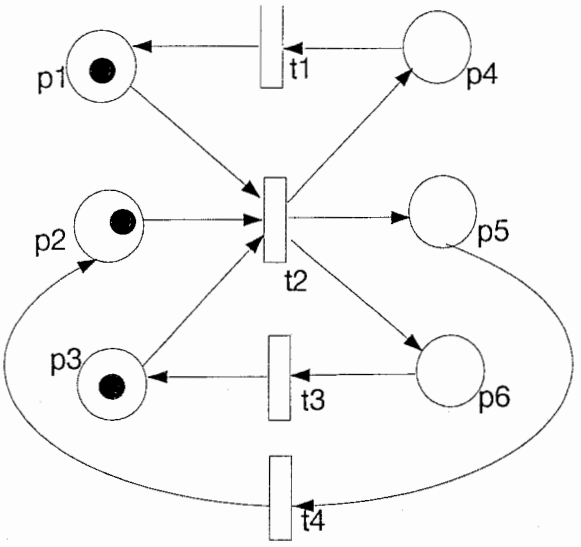
上記の 1,2,3 成立するなら, ある十分な初期状態を与えることによりデッドロックフリーであることが証明できる. また, インバリアント計算にかかる計算量 (計算時間) は, 多項式オーダ (多項式時間) とはなり得ないが, Farkas のアルゴリズム等の高速アルゴリズムが存在するし, サイフォン・トラップの計算は多項式オーダである.

4.4 HLPN におけるデッドロックの判別

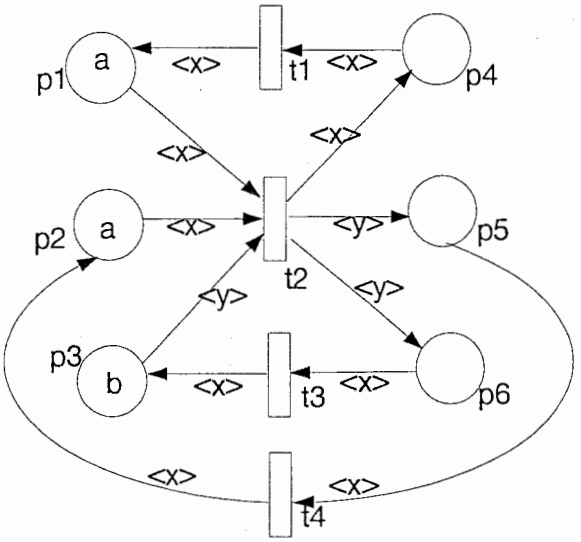
HLPN においてのインバリアント計算は, かなり複雑であり, 単純な行列計算では求めることはできない. また, そのデッドロック検出は色なしのネットに比べはるかに複雑な処理を必要とする. ここでは, 通信システムサービスのような可逆性を有する HLPN のデッドロックの検証が比較的簡単に行えることを示す.

[**HLPN の可逆性の必要条件**] HLPN からカラーに関する情報を取り除いた PN において T- インバリアントを持たなければ, HLPN においても T- インバリアントを持たない. □

[**証明**] これは自明である. □



可逆性を満たすデッドロックフリーなＰＮ



デッドロックとなるＨＰＮ

図 11: PN が可逆的であるが HPN が可逆的でないもの

[系 1: **可逆的な構造を持つ HLPN のデッドロック**] PN において可逆性を満たす初期マーキングに対応する HLPN のマーキングにおいて, トークンにどのような色を与えてもデッドロックを起こすものが存在する. □

例を以下に示す. 色なしネットにおいてデッドロックを起こさない初期マークングにおいて, 図のようにトークンの色の組合せが変化するものは, その初期マー

キングにおいてデッドロックを起こす. 図の場合

$$(a,\ a,\ b,\ 0,\ 0,\ 0) \rightarrow (a,\ b,\ b,\ 0,\ 0,\ 0)$$

となっている. もしこれが,

$$(a,\ a,\ b,\ 0,\ 0,\ 0) \rightarrow (b,\ b,\ a,\ 0,\ 0,\ 0)$$

となる場合 (色が変化しただけで色の組合せは変化しない場合: すべての a が b に b が a になる場合) はデッドロックを起こさない.

ただし, 初期マーキングが色, 数ともに十分あればデッドロックを起こさなくすることはできる. 例えば, 図11において,

$$(a+b,\ a+b,\ a+b,\ 0,\ 0,\ 0)$$

であればデッドロックは回避できる.

しかし, このようなトークンを追加するという解消は実際のモデルの意味から, しばしば無意味である場合がある. 例えば, 輸送システムのモデルであれば, 「異なる駅 (異なるプレース) に同時に同じ列車 (同じ色を持つトークン) が存在し得ない」とか, ここで対象とする通信のモデルであれば, 「話し中 (という状態に対応するプレース) の端末 (に対応する色つきトークン) は同時にベルがなるという状態にはならない. 」という意味を考慮すれば, 図11の例では

$$(a,\ a,\ a,\ 0,\ 0,\ 0)$$

となるマーキングが無意味である等があらかじめ把握できる.

このように, 通信システムのサービスモデルでは意味を考えることにより, 初期マーキングをあらかじめ与えることが可能である.

[系2: **HLPN のサイフォンによるデッドロック**] HLPN がサイフォンとなるため可逆的であってもデッドロックを起こす場合がある. □

通信システムにおける簡単な例で説明する. 図12は相手呼びだし中切断の例である.

ルール1) *idle(A) offhook(A): dt(A).*

ルール2) *dt(A), idle(B) dial(A,B): rbt(A,B), rgt(B,A).*

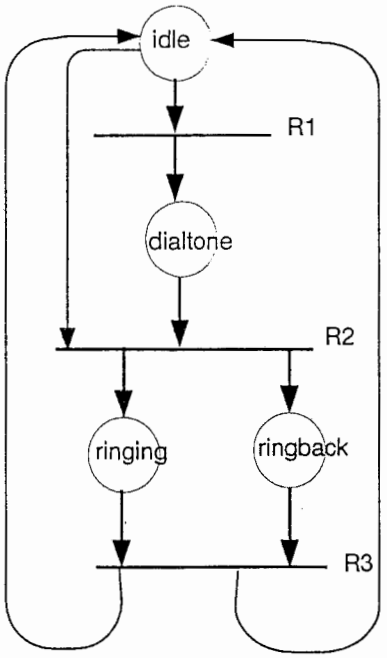


図 12: サイフォンを持ちサイフォンがトラップを持たない例

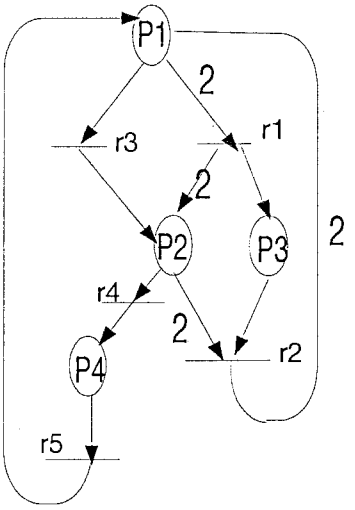
ルール3 *) rbt(A,B), rgt(B,A) onhook(A): idle(A), idle(B).*
というルールが与えられ, 状態要素の集合

$$S = \{idle, rgt, rbt\}$$

を考えた時, その状態要素を前状態に含むルールはルール 1, ルール 2, ルール 3 であり次状態に含むルールはルール 2, ルール 3 となりサイフォン構造を持つ. この時, 実状態として S に含まれる状態要素を持たないならば, その実状態からどのような遷移を行っても S の状態要素を持つ実状態は存在 (出現) しない. さらには S の状態要素を発火条件とするルール 1 , ルール 2 , ルール 3 はいずれも発火しない. つまりすべてのトークンが $\{dt\}$ のみに存在すれば, デッドロックとなる.

4.5 HLPN の有界性

色を除いた PN において有界でない構造であれば, 状態要素が解放されず残ってしまうことが起こり得る. 図 13では, P1 から P1 への遷移は r1,r2 および r3,r4,r5 の 2 系列である. しかし, r1,r4,r4,r5,r5 を繰り返すことにより, P3 が無限に発生する.



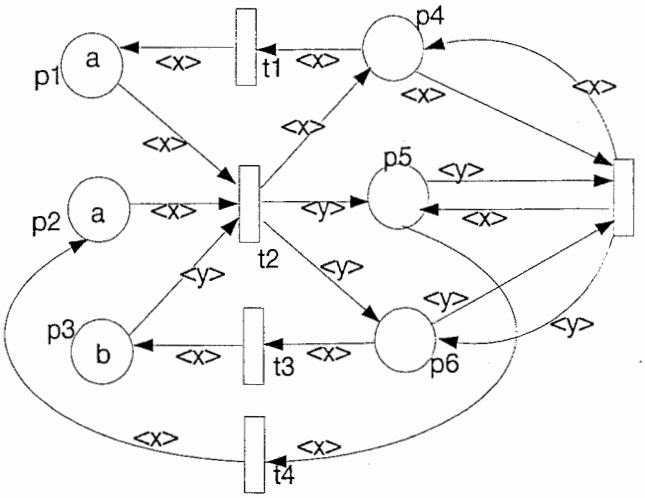
r1) P1(A), P1(B)　　e1(A,B): P2(A), P2(B),P3(A,B).
r2) P2(A),P2(B),P3(A,B)　e2(A): P1(A),P1(B).
r3) P1(A)　　e3(A): P2(A).
r4) P2(A)　　e4(A): P4(A).
r5) P4(A)　　e5(A): P1(A).

図 13: T- インバリアント且つ非有界

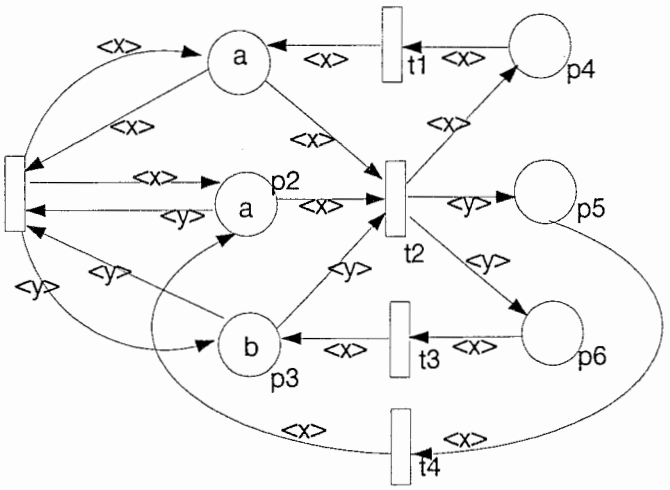
- 異常な状態遷移の構造的解釈
  個々の個別サービスが活性且つ有界であるのに, 結合したサービスが異常な状態になる場合がある. 異常な状態とは, サービスを統合することにより個々のサービスには現れない状態が出現することをいう. このことが検出される構造的な条件としては, 有界性が崩れる時, および部分ネットがサイフォンとなる場合が考えられる. この例として, 図 13 には r1, r2 と r3,r4,r5 の 2 つのサービスが存在すると仮定すると, それぞれのサービスは活性かつ有界である. しかし, ネットは非有界になる.

4.6 HLPN のデッドロックの解消

通信サービス仕様にデッドロックが存在することは, そのサービスが使えないということだけでなく, ユーザの端末が利用できなくなることであり, 最悪の場合は交換機のシステムダウンを引き起こす可能性がある. ここでは, 検出したデッドロックを解消する手法について述べる.



デッドロックの結果部にトランジションを追加



デッドロックの条件部にトランジションを追加

図 14: デッドロックの解消例

- 4.2 章系 1 のデッドロックするネットに T- インバリアントを追加することによる解消

  系 1 より T- インバリアントを遷移させた場合, 色の組合せを変化させる遷移であれば, デッドロックであった. この時, 他の T- インバリアントにより色の組合せを元に戻すことができればデッドロックを解消することができる. 図 14では二つの解消例を示している. どちらのものも, トークンの色を元に戻す系列を持つ. しかし, デッドロックを起こすトランジション $t2$ の結果部を変えるトランジションの追加では, デッドロックが解消されていないことが理解できる. このこ

とから, 以下のことがわかる.

T- インバリアントが複数ある場合, 個々の T- インバリアントがデッドロックを起こす場合であっても, 他の T- インバリアントでデッドロックとなるトランジションの条件部の色を元に戻す場合は, デッドロックを解消できる.

- 4.2 章系 2 のデッドロックするネットにトランジションを追加することによる解消

  ネットがサイフォンである場合, サイフォン構造を崩すようにトランジションを追加することで, 解消可能である. 図 12では,

$$dt(A) \rightarrow idle(A)$$

  となるトランジションを追加すれば良いことがわかる.

4.7 連鎖判定手法

通信システムの性質を持つネットでは, 端末数に依存しない数だけの状態数と連鎖状態ですべての状態が生成可能である. このことは, 以下の定理から証明可能である.

[定理] 以下の条件を満たす HLPN が存在する.

- 可逆性を満たす
- 有界性を満たす
- 活性である
- カラー要素が増えることがない
- マーキングに含まれるトークン数を必要なだけ取り得ると仮定する.
- 連鎖状態とマーキングに非依存な数の状態ですべての状態を生成可能である

上記定理の証明のため以下の補題を与える.

[補題] 以下の条件を満たす PN が存在する.

- 可逆性を満たす
- 有界性を満たす

- 活性である
- マーキングに含まれるトークン数を必要なだけ取り得ると仮定する.
- 連鎖状態とマーキングに非依存な数の状態ですべての状態を生成可能である

[補題の証明] 可逆性を満たすため, ネットはT- インバリアントを持つ. ここで, 極小なT- インバリアントの集合を $T = \{t_i\}$ とする. ある t_i について, 活性であり, 可逆的であるマーキング M_i が存在し, t_i の遷移により生成されるマーキングを $\{m_{ij}\}$ (j はトランジションの発火回数以下の任意数) とする. また, 2倍の t_i について活性かつ可逆的であるマーキングは $2M_i$ であり, $2t_i$ の遷移により生成されるマーキングは $\{m_{ij_1} + m_{ij_2}\}$ (j_1, j_2 はトランジションの発火回数以下の任意数) で表される.
この時, t_i の台集合で生成されるネットは有界である. よって, ある初期マーキング M が与えられた場合, すべての状態は有限回の発火で生成できる. その時生成される状態は, 以下のようになる.

$$M - k * M_i + (m_{ij_1} + m_{ij_2} + \cdots m_{ij_k})$$

ただし, $j_1, j_2, \cdots j_k$ はトランジションの発火回数以下の任意数である. これは, すべてのT- インバリアントの組合せに対しても成立し, すべての状態は,

$$M - \sum_i (a_i * M_i) + \sum_i \sum_j (m_{ij})$$

ただし, a_i は適当な定数, m_{ij} の i は
$0 \leq i \leq$ 極小T- インバリアントの数,
j は $0 \leq j \leq a_i$ である. よって, マーキングに非依存な数の m_{ij} を生成することにより, すべての状態を生成可能となる. □
HLPN においてもカラー要素が増えないことを利用して同様に証明することができる.

5 おわりに

本論文では, 以下のことについて議論した.

- HLPN から色を抜いた PN による検証の有効性
- 通信システム等の離散事象システムにおいて有効となる性質の洗いだし
- トークン数に非依存となる計算量で状態生成を行なえること.

[謝辞] 本研究を進めるにあたり, 御指導と励ましをいただきました ATR 通信システム研究所の研究員の皆様に深く感謝致します.

参考文献

[1] ISO8807: Information processing systems - open systems interconnection - LOTOS - a formal description technique based on the temporal ordering of observational behavior, ISO publication(1988).

[2] 高橋 他「LOTOS 言語の特質と処理系の現状と動向」, 情報処理, Vol.31, No.1, pp.35-46(1990)

[3] 村田「ペトリネットの解析と応用」, アルゴリズムシリーズ 5, 近代科学社,1992

[4] J.L. Peterson 著, 市川惇信, 小林重信 訳「ペトリネット入門」, 共立出版, 1984.

[5] T. Murata, et al. "A Petri Net Model for Reasoning in the Presence of Inconsistency," IEEE Trans. on Knowledge and Date Eng., VOL.3, No.3, 1991, pp.281-292.

[6] 小林「知識工学」, 人工知能シリーズ 10, 昭晃堂

[7] H. J. Genrich "Predicate/Transition Nets," Advances in Petri Nets 1986: Part I,LNCS, Vol.254, Springer-Verlag,pp. 207-247. 1987

[8] 上田他「通信システムサービス仕様におけるデッドロック検出方式」, 信学技法, CAS94-65, pp.71-76, 1994

[9] 上田他「離散事象システムにおける可達集合の高速検証方式」, 信学技法, CST95-10, pp.29-34, 1995

[10] 上田他「通信サービス仕様の静的解析手法」, SICE 第 13 回離散事象システム研究会, pp.29-34, 1994

[11] Y. Harada, et al. "A Conflict Detection Support Method for Telecommunication Service Descriptions", IEICE Trans. Commun., VOL. E75-B, No.10 Oct. 1992.

[12] Y. Harada, et al. "A Conflict Detection Support Method for Telecommunication Service Descriptions", IEICE Trans. Commun., VOL. E75-B, No.10 Oct. 1992.

[13] 柴田他「通信サービスにおける状態の到達可能性解析」, 信学会交換システム研究会 SSE92-30,1992.

[14] Y. Hirakawa, et al. "Telecommunication service description using state transition rules," Proc. Sixth Int. Work. Software Specification and Design, Como, Italy, Oct. pp. 140 - 147, 1991.

[15] 上田他「通信システムサービス仕様における正当性検出と解消方式」, 信学技法, KBSE94-44, pp.25-32, 1994

[16] 村田 他「ペトリネットによる並列処理プログラムの解析手法」情報処理, Vol. 34, No.6, pp. 701-709, 1993