ペトリネットによる並行ソフトウェアシステムの設計時動作解析†

孫 立寧†† 落水浩一郎††

本論文は，複数の動作実体がリンクを介して相互作用するような並行ソフトウェアシステムの動作解析手法について述べたものである．実体（プロセス，モジュール等）を拡張状態遷移図で記述し，実体間のリンクの特性を指定した上で，それぞれをペトリネット表現に変換し，さらにそれらのペトリネット群をまとめて一つのペトリネットに結合，解析し，解析結果を再び拡張状態遷移図，およびリンク表現に逆変換する．VAX 11-780 上の Franz-lisp で解析系のプロトタイプを試作し，いくつかの例に適用することにより，本手法の有効性を確認した結果についても述べる．

1. はじめに

設計段階にあるソフトウェア・システムの動作を解析することは，誤りの早期発見，より良い設計代替案の採用等に有効である．

このためには，システムの動作分析ができるように，特定の観点からシステムの動的振舞いを表現する記述体系が必要である．例えば，対話型情報システムにおけるユーザ・インタフェース（画面遷移）のシミュレーションを実施するための RAPID/USE[1]，システムの動作によって引き起こされる事象を記号表現し，各タスクごとの動作を事象記号列(タスク式)として表した後，シャフル演算子によって，起こりうる動作系列を生成し解析の基礎データとする Constrained Expressions アプローチ[2]，拡張ペトリネットを基礎にした仕様記述法を用いてシステムの動作を表現し，仕様レベルで設計書の Critical な点を洗い出した後，同期に関する誤りをもたない Ada プログラムを自動生成する PROT アプローチ[3]等が最近の研究結果として報告されている．このような研究では，状態遷移図や，ペトリネットなどが記述の基礎として使用されているが，以下の得失がある．すなわち，状態遷移図は，外部信号による状態遷移，および各状態の果たす制御全体における役割が明瞭に記述できるが，複数の状態遷移図がリンクを介して相互作用するような並行システムにおいては，系統的解析が困難である．また並列処理システムのモデル化に用いられたペトリネットは，ネットワーク・プロトコルの検証をはじめ[4]，解析的な面への応用が広がりつつある[5]．ペトリネットはそのモデル化能力が高く，解析モデルとしてもっとも基本的なものであると考えられるが，状態遷移図と比べると，構造が複雑，記述量が多い等の欠点を持つので，大規模ソフトウェアの開発には応用しにくいと考えられる．

これらの点から判断して，本論文では，マン・マシン・インタフェースおよび自動生成技術とのつながりのよさを考慮して，仕様記述の基礎体系として状態遷移図を採用し，また動作解析系としてはその構成が容易であることからペトリネットを採用する．すなわち，複数の動作実体がリンクを介して相互作用するような対象の動作解析において，**図1**に示すように，利用者が実体内部の動作を状態遷移図で表し，実体間の通信方式をリンクの特性として指定した上で，全体を解析系にわたす．解析系では，状態遷移図をペトリネットに変換，リンクを生成し，それぞれのペトリネットをまとめて一つのペトリネットに結合，解析し，解析結果を再び状態遷移図およびリンク表現に逆変換して利用者に返す手法を論じることにより，マン・マシン・インタフェースに状態遷移図を用い，解析の詳細は解析系の内部に隠すという手法に対する基礎を与える．

以下，第2-6章では，資源管理問題を例にとり，拡張状態遷移図，ペトリネットへの変換，リンク，結合，解析，および逆変換等の，解析系の基本技法を説明する．第7章では，本解析手法の適用例について簡単に述べ，第8章で，本解析手法の評価をまとめる．

2. 拡張状態遷移図

2.1 拡張状態遷移図

拡張状態遷移図はソフトウェア・システムを構成す

† Behavior Analysis of Concurrent Software Systems at Design Phase Based on Petri-net by LI-NING SUN and KOICHIRO OCHIMIZU (Department of Computer Science, Faculty of Engineering, Shizuoka University).

†† 静岡大学工学部情報工学科

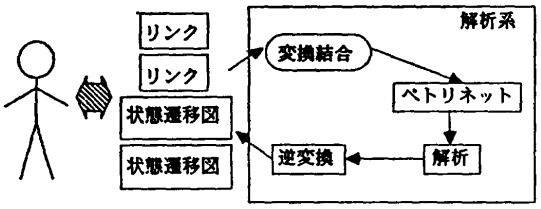


図 1 解析系の概要
Fig. 1 The outline of analysis system.

る実体集合（プロセス，モジュール等）の動作を記述するために，状態遷移図に動作遅延時間の概念を導入したものである．普通の状態遷移図では，ソフトウェア・システムの実現予定コード部分を表現目的な分析目的により有限個の状態で抽象する．拡張状態遷移図では，さらに各状態における実際の処理時間をその状態における遅延時間として付加する，例えば，図2において，状態 $S1$, $S2$ における動作遅延時間が 2, 3 である．すなわち，事象 a の発生後，$S2$ が現在の状態になってから，事象 b を受け付けるまでには，3単位時間必要とするという意味である．このとき，遅延時間中に発生した入力は発生順に記憶することとし，また入出力としては基本事象の論理積表現も許すものとする．

[定義1]

拡張状態遷移図は5項組 $(Q\times\tau,\ \Sigma,\ \Delta,\ \delta,\ \Gamma,\ \tau)$ である．

$Q\times\tau$：状態の有限集合，τ が遅延時間，自然数．$\tau=0$ のとき，単に Q と略記することもある．

Σ：入力の有限集合

Δ：出力の有限集合

$\delta: Q\times(\tau>0)\rightarrow Q\times(\tau-1)$

遅延時間の経過，または

$Q\times(\tau=0)\times(\cap\Sigma)\rightarrow Q\times\tau$

現在の状態と現在の入力を次の状態へ写像する状態推移関数の有限集合

$\Gamma: Q\times(t=0)\times(\cap\Sigma)\rightarrow\cap\Delta$

現在の状態と現在の入力を出力へ写像する出力関数の有限集合 □

[例1]

図3に複数のプロセスが共通の資源をアクセスする問題の拡張状態遷移図による記述を示す．実体プロセスにおいて，

$Q\times\tau=$(Inital, Wait, Access×2, Process×1, End)

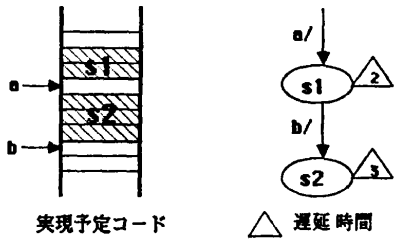


図 2 拡張状態遷移図によるソフトウェアの抽象
Fig. 2 Abstract software representation in ETDs.

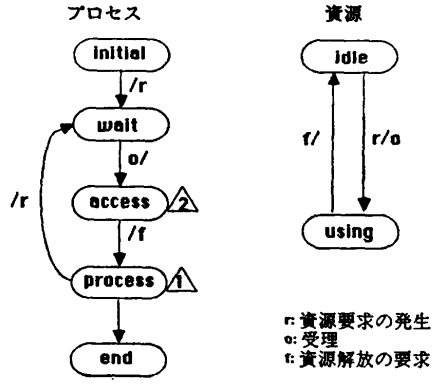


図 3 拡張状態遷移図表現による資源管理問題
Fig. 3 Resource management problem in ETDs representation.

$\Sigma=(o)$

$\Delta=(r,\ f)$

遅延時間の経過関数は

δ: Access×2 → Access×1

Access×1 → Access

Process×1 → Process

である． □

2.2 ペトリネットへの変換

拡張状態遷移図をペトリネットで表現するためには，遅延時間の表現できるペトリネットが必要となる．文献 6) では遅延時間をトランジションの発火に要する時間であるとして時間をトランジションに割り付けるにの対して，文献 7) では遅延時間をプレースの経過に要する時間であるとして時間をプレースに割り付ける．これらの提案はペトリネットの記述要素を増加させ，また発火規則を複雑にし，その結果，解析力を低下させる．本論文では拡張状態遷移図の特性から，通常のペトリネットで，遅延時間を図4のようにトークン数により表現する方式を採用する．この方式はタイムド・ペトリネットに比して，ペトリネットの解析力を低下させない特徴をもつ．すなわち，$S1$ において，トランジション $t1$ を二回発火してから，入

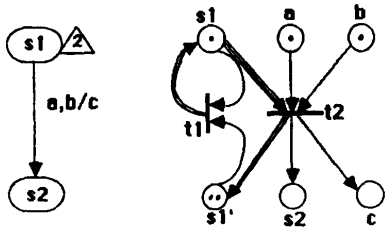


図 4 拡張状態遷移図からペトリネットへの変換
Fig. 4 Conversion from ETDs to Petri-net representation.

力 a, b を受け付けるトランジション $t2$ を発火できる状態となる.

[定義 2]

拡張状態遷移図 $(Q\times\tau,\ \Sigma,\ \Delta,\ \delta,\ \Gamma)$ に対して, ペトリネット $(P,\ T,\ I,\ O)$ を次のように対応させる.

P はプレースの有限集合であり

$P=S\cup R\cup D$.

ここで, S は状態プレースの有限集合であり

$Q\times\tau$ に対応させ,

R は時間プレースの有限集合であり

$Q\times\tau$ $(\tau\neq 0)$ に対応させ,

D はデータ・プレースの有限集合であり

$\Sigma\cup\Delta$ に対応させて定義する.

このとき, トランジションの有限集合 T は

$$T=\{ts,\ r\ |s\in S,\ r\in R\}\cup\{ts,\ d\ |s\in S,\ d\in D\}$$

のように定義する. ここで, ts, r は遅延の経過に伴う遷移を表し, $ts,\ d$ は入力事象の発生による状態遷移を表す. トランジション t を入力/出力プレースの多重集合へ写像する入力/出力関数をそれぞれ

$$I(t)=\{((\tau+1)*s,\ d)|ts,\ d\in T\}\cup\{(s,\ r)|ts,\ r\in T\}$$

$$O(t)=\{(\delta(s,\ d),\ \Gamma(s,\ d),\ \tau*r)|ts,\ d\in T\}\cup\{(2*s)|ts,\ r\in T\}$$

のように定義する. □

プレース中のトークン数は以下のように解釈する.

●時間プレース中のトークン数は残遅延時間を表し, 初期マーキングでは, 遅延時間に設定される.

●状態プレース中のトークンは拡張状態遷移図の現状態に対応し, 状態プレースに n $(n>1)$ 個のトークンがあることは, その状態での計算が既に $n-1$ 単位時間を経過したことを表す.

●データ・プレース中のトークンは入出力事象の発生を表す.

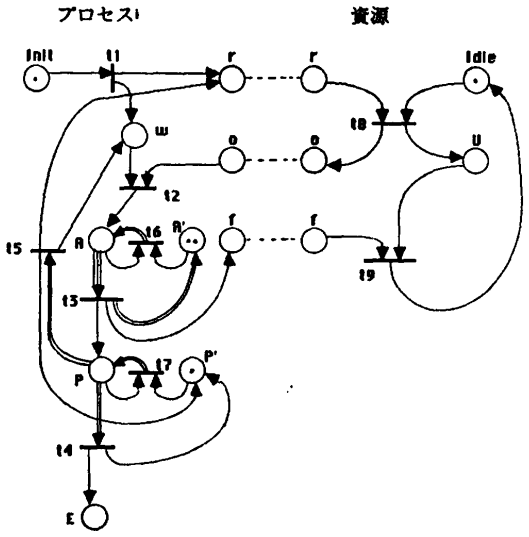


図 5 ペトリネット表現による資源管理問題
Fig. 5 Resource management problem in Petri-net representation.

[例 2]

図 5 は図 3 に示す資源管理問題のペトリネット表現である. 実体プロセスにおいて,

$S=(Init,\ W,\ A,\ P,\ E)$

$R=(A',\ P')$

$D=(r,\ o,\ f)$

$ts,\ r=(t6,\ t7)$

$ts,\ d=(t1,\ t2,\ t3,\ t4,\ t5)$

である. □

図 5 において, ある実体で閉じない入出力事象 $(r,\ o,\ f)$ が存在する. これは他の実体に伝える/から伝えられる事象を表し, 外部事象と呼ぶことにする. 利用者は外部事象を転送するリンクを定めることにより, 実体間の通信が可能になる.

3. リンクと結合

実体間の通信はリンクを介して行う. 本論文では通信方式として, 送り主は一切待つことのない無限容量のリンクを介する非同期方式と, 有限容量のリンクを介するバッファ方式を考えることにする.

3.1 リンク

リンクはバッファ, キューを抽象したものであり, 容量, 通信手段等の特性を指定できる.

3.1.1 バッファ

バッファは同種類のトークンを転送するリンクであり, 有限容量, または, 無限容量のものを指定できる.

[定義 3.1]

有限容量のバッファは**図6**(a)に示すペトリネットにより定義する．プレースdはトークンを貯める役割を果たし，リンク・プレースと呼び，プレース1中のトークン数はバッファの残容量を表し，容量プレースと呼ぶことにする． □

[定義 3.2]

有限容量バッファの容量プレース中のトークン数が無限のとき，無限容量バッファという．このとき，容量プレースは記述しない（図6(b)）． □

3.1.2 キ ュ ー

キューは異種類のトークンを First-in-First-out で記憶・転送するリンクである．容量と転送するトークン型を指定できる．

[定義 4.1]

容量が1，トークン型の種類が1，のキューは容量1のバッファとし，単位キューと呼び，Kで表す（**図7**(a)）． □

[定義 4.2]

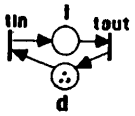
容量がn，トークン型の種類が1，のキューはn個の単位キュー($K1, K2, \ldots, Kn$)を直列につなぎあわせる（$1 \leqq i \leqq n-1$に対して，Kiのトランジション$t$outと，$Ki+1$のトランジション$t$inとを重ね合わせる）ことにより構成されたものとし，nキューと呼ぶ（図7(b)）． □

[例3]

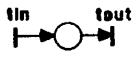
図7(c)に3キューのトークン転送例を示す． □

[定義 4.3]

容量がn，トークン型の種類がm，のキューはm個のnキュー $\{(K11, \ldots, K1n), \ldots, (Km1, \ldots, Kmn)\}$ を並列につなぎあわせる（$1 \leqq i \leqq n$, $1 \leqq j \leqq m-1$ に対して，Kijの容量プレースと$Kij+1$の容量プレースを重ね合わせる）ことにより構成されたものとし，$m \times n$

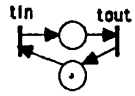


(a) 容量3のバッファ
(a) The buffer of capacity 3.

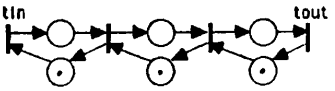


(b) 無限容量のバッファ
(b) The buffer of infinite capacity.

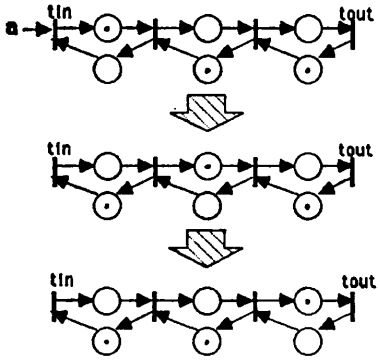
図6 バッファ
Fig. 6 Buffer.



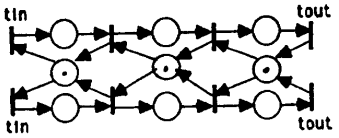
(a) 単位キュー
(a) Unit queue.



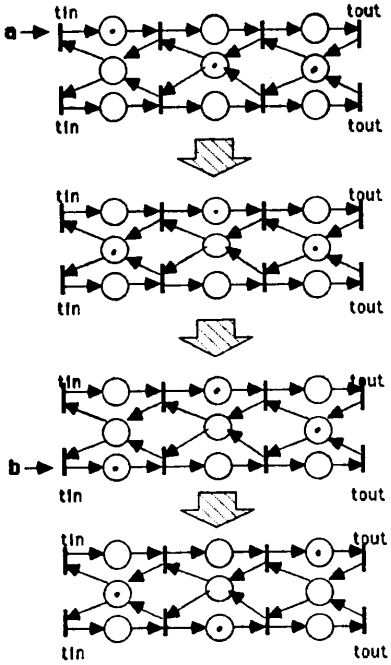
(b) 3キュー
(b) 3 queue.



(c) 3キューの転送例
(c) An example of transfer in 3 queue.



(d) 2×3 キュー
(d) 2×3 queue.



(e) 2×3 キューの転送例
(e) An example of transfer in 2×3 queue.

図7 キュー
Fig. 7 Queue.

キューと呼ぶ（図7（d））. □

[例4]

図7（e）に 2×3 キューのトークン転送例を示す. □

$m \times n$ キューは，容量が n であり，各キューに種類ごとにトークンを管理するとともに，全体の入力順序を保持するキューである.

作成した解析系では，バッファ，キューに対する容量等の指定を行うと，対応するペトリネットを単位キュー等をもとに生成する機能を与えている.

3.2 結　　合

[定義5]

結合とはペトリネット表現の動作実体と解析系のツールにより生成されたリンク等の複数のペトリネットを一つのペトリネットにまとめあげることである. まとめあげられた一つのペトリネットを複合ペトリネットと呼ぶ. □

図8に示す二つの動作実体の外部事象 a が，**図9**に示すように結合される.

図5に示す資源管理問題においては，実体プロセスと資源とは，3個の無限容量のバッファ r, o, f を介して，結合され，非同期方式で通信を行うことになる.

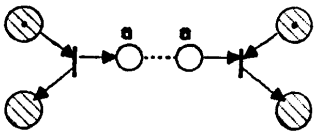
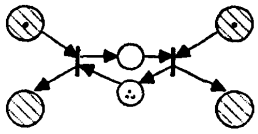
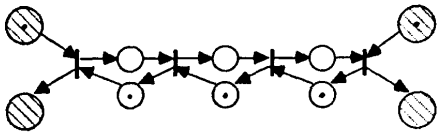


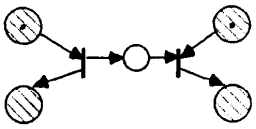
図8 結合される前の二つの動作実体
Fig. 8 Two objects before combining.



（a） 容量3のバッファによるバッファ方式
（a） Buffering discipline in the buffer of capacity 3.



（b） 3キューによるバッファ方式
（b） Buffering discipline in the 3 queue.



（c） 非同期方式
（c） Asynchronous discipline.

図9 結合（◎は動作実体のプレースを表す）
Fig. 9 Combine (◎ representation the place of object).

4. ペトリネットの解析

4.1 解 析 手 法

複合ペトリネットそのものは単に一つのペトリネットなので，今まで研究されたいくつかの解析方法はそのまま使える. 本解析系では，文献 8) による可達木を以下のように修正して，解析する.

本来の可達木は初期マーキングから，ペトリネットの発火ルールに従い，可達可能なマーキング集合を木の形式で実現するものであり，もし，ある時点で，一つ以上のトランジションが発火条件を満たすとき，おのおののトランジションを発火させ新しいマーキングを得る. ところが，それはペトリネットの解析に対して指数オーダのメモリ領域と時間を要求するので，実際的な応用には使いにくい. 本解析系では，実体は他の実体との同期をとる部分以外は，並行に独立動作するという性質から，同時に発火可能な（競合関係にない）トランジションは同時に並行発火させるような変形の可達木（以後は簡単に可達木と呼ぶ）を求め，解析の効率を上昇させる.

解析の効率は，問題に依存するが，最良の場合線形のメモリ量と時間により解析可能となる.

[定義6]

発火可能なトランジションの集合 $T=(t1, t2, ..., tk)$ において，同時に発火可能なトランジションの集合 $Tx=(T1, T2, ..., Th)$ を次のように定義する.

a. $Tl \subset T$ $(l=1, 2, ..., h)$.

b. $^{\forall}ti, tj \in Tl$, ti と tj は競合関係ではない.

c. $^{\forall}tf \in T-Tl$, $^{\exists}tg \in Tl$, tf と tg は競合関係である.

d. そのほかの $^{\forall}Tz \subset T$, $Tz \notin Tx$, Tz は a, b, c を満足しない. □

図10に資源管理問題の可達木を示す. マーキング $(P, 2A', P', f, U)$ では，$t7$ と $t9$ を同時に発火させ，マーキング $(2P, 2A', Idle)$ では，$t4$ と $t5$ に分岐する.

このように求めた可達木において，以下のような用語を定義する.

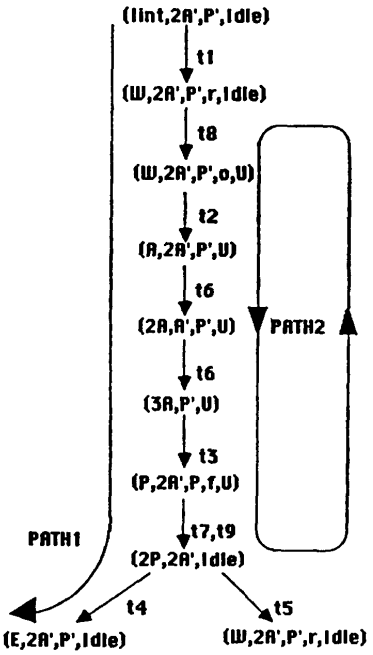


図 10 資源管理問題の可達木
Fig. 10 Reachable tree of resource management problem.

●初期マーキングから可達可能なマーキングの集合をCで表し，そのうち，どのトランジションも発火できない（例 $(E, 2A', P', Idle)$）マーキングの集合を終端マーキングと呼び，Ct で表す．

●マーキング M から，マーキング M' に可達可能なら，M と M' 間に可達パスがあるという．もし，M が初期マーキング，$M' \in Ct$ なら，そのパスを切断パス（例 Path 1）といい，もし，$M'=M$ なら，そのパスを連結パス（例 Path 2）という．

4.2 ペトリネットの性質

可達木を利用してペトリネットにおける数多くの性質が解析できる．状態遷移図の動作解析においては，次に述べるいくつかのペトリネットの性質が有効と考えられる．

[定義 7.1]

Boundedness：もし，$^{\forall}M \in C$ においては，すべてのプレースに含まれるトークン数が有限なら，ペトリネットは Boundedness という． □

[定義 7.2]

Liveness：もし，$^{\forall}M \in C$, $^{\forall}t \in T$ において，一定のトランジション・シーケンスの発火より，t が発火可能なら，ペトリネットは Liveness という． □

[定義 7.3]

Home-state：もし，$^{\exists}M \in C$ がすべての連結パスに含まれるなら，M が Home-state といい，ペトリネットは Home-state をもつという． □

[定義 7.4]

Proper-termination：もし，任意の切断パスの最終マーキング $M \in Ct$ において，すべてのデータ・プレース，およびリンク・プレースにトークン数が0なら，ペトリネットは Proper-termination という． □

[定義 7.5]

Reachability：$^{\forall}M$, $M' \in C$ が与えられたとき，M から M' に至る可達木上のパスが存在する． □

これらの性質の拡張状態遷移図，およびリンクの動作上における解釈は次のようになる．

●Boundedness が成り立つと，ペトリネットでは特にデータ・プレース，およびリンク・プレース上のトークン数が有限であることが保証される．すなわち，リンクに貯まりうる事象の数は有限である．

●Liveness が成り立つと，ペトリネットでは発火しないトランジションがないので，拡張状態遷移図で定義したすべての遷移は論理的に起こりうる．

●Home-state を持つと，システムにおいて，ある特定の状態が再現されることを意味する．そのようないくつかのシステムの動作間に共通の状態があることも意味する．

●Proper-termination が成り立つと，拡張状態遷移図で定義した終了状態において，リンク中に受け付けを待つ事象はない．

●Reachability が成り立つと，ソフトウェアのある特定の動作状態からある特定の動作状態に推移可能であることを意味する．

5. 解析手順

本解析系を利用した解析手順は以下のようになる．

（1）ソフトウェア・システムの動作における解析したい点（解析の仕様）を定める．

（2）解析仕様の実現法を検討する．Boundedness はバッファ容量の設計，Liveness はソフトウェア中の非活性部分の検出，Home-state はソフトウェアの特定の動作状態（例えば，エラー処理状態）の確認，Proper-termination は終了状態の確認とデッドロックの検出，Reachability はソフトウェアの特定の動作状態への可達性の確認に使用できる．

（3）解析結果を分析する．

解析結果は定義 7.1-7.5 で述べた性質が成り立つ

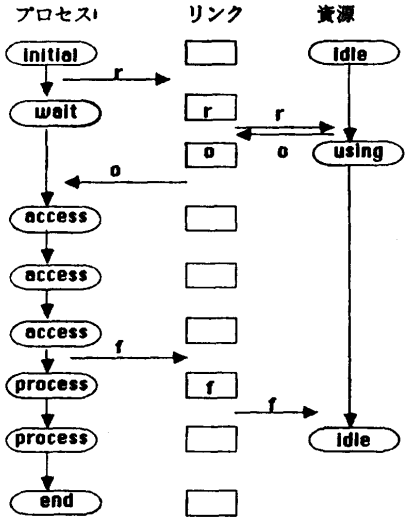


図 11 資源管理問題の切断パス
Fig. 11 Disconnecting path of resource management problem.

か，成り立たないかという形で提示されるが，成り立たない場合において，重要なのはその原因に関する情報を簡潔明瞭に逆変換機能を通して利用者に提供することである．我々は，逆変換機能にわたす解析結果を次のように定めている．

- もし，Boundedness が成り立たない場合には，そのマーキングに到達するパスを表示する．
- もし，Liveness が成り立たない場合には，そのトランジションを表示する．
- もし，Home-state が存在するなら，それを表示する．または，利用者の指定の Home-state を入力させ，これを含まない連結パスを表示する．
- もし，Proper-termination が成り立たない場合には，その切断パスを表示する．

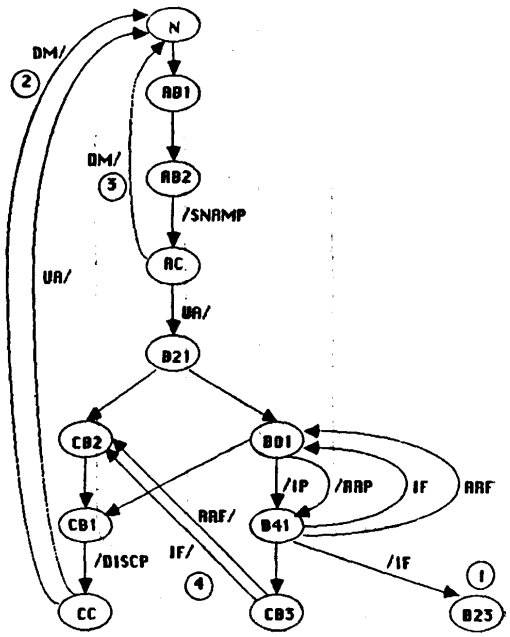
6. 解析逆変換

解析の逆変換では上記したペトリネットの性質の解析結果（マーキング，パス，トランジション）を変換時に作成した拡張状態遷移図，リンクとペトリネットの間の記号的対応関係を利用して，もとの拡張状態遷移図表現に変換する．図 11 に資源管理問題の切断パスの拡張状態遷移図表現の例を示す．

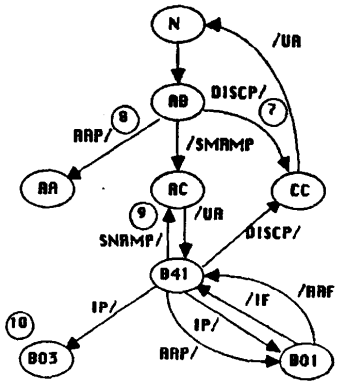
7. 適 用 例

我々の提案した手法を HDLC の解析に適用した例を示す[9]．

クラス UN に関する一次局と二次局の動作を拡張状態遷移図で最初に表現したものを図 12 に示す．



(a) 一次局
(a) Primary side.



(b) 二次局
(b) Secondary side.

図 12 クラス UN の動作の一部
Fig. 12 A part of internal behaviour of class UN.

一次局と二次局は，非同期方式による通信を行うとする．

(1) 解析の仕様

a. データ転送用のバッファがオーバ・フローにならないか．
b. 終了モードによる以外は，行きづまることはないか．
c. 切断パスが正しいか．
d. 一次局と二次局間での情報フレーム通信における

順序が正しいか.

e. 状態遷移図の中のすべての状態と遷移が論理的に必要であるか.

(2) 解析の実現法

解析の仕様をもとに, a は Boundedness, b と c は Proper-termination, d は Home-state, e は Liveness を使って, 解析の実現法を設計する.

(3) 解析結果の分析

解析系を利用して解析し, 解析結果を分析することにより, 図12に存在した設計誤りを検出訂正した.

- Proper-termination により, 状態 1, 10 を削除, トランジション4を追加.
- Liveness により, トランジション 2, 3, 7, 8, 9 を削除.

8. おわりに

本報告では並行ソフトウェア・システムに対して, 拡張状態遷移図とリンクによる記述体系, およびペトリネットによるフォーマルな解析手法を提案した. 本手法では, 閉じたペトリネットの世界で解析を行うので, ペトリネット理論をそのまま, きれいに展開でき, しかも, 今後ペトリネット理論の発展によって, より高いレベルの解析が期待できるものと思われる.

現在, VAX 11-780 上の Franz-lisp で実現したプロトタイプを完成し, いくつかの実例を解析してみたが, 解析時間は7章で例で 10 秒ぐらいであり, 変形可達木の効果が大きい. 強力なグラフィックスのマン・マシン・インタフェースを増強し, 本論文で述べた原理的手法をプロトタイピング・ツールにまで発展させることが今後の課題である.

参考文献

1) Wasserman, A. I. et al.: Developing Interactive Information Systems with the User Software Engineering Methodology, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 2, pp. 326-345 (1986).
2) Avrunin, G. S. et al.: Constrained Expressions: Adding Analysis Capabilities to Design Methods for Concurrent Software Systems, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 2, pp. 278-292 (1986).
3) Bruno, G. et al.: Process-Translatable Petri Nets for the Rapid Prototyping of Process Control Systems, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 2, pp. 346-357 (1986).
4) Azema, P. et al.: Design and Verification of Communication Procedures: a Bottom-up Approach, *Proc. 3rd Int. Conf., Software Eng.*, pp. 168-174 (1978).
5) Agerwala, T.: Some Applications of Petri Nets, *Proc. Nat. Electron, Conf.*, Vol. 23, Nat. Eng. Consortium, pp. 149-154 (1978).
6) Ramamoorthy, C. V. et al.: Performance Evaluation of Asynchronous Concurrent System Using Petri-Net, *IEEE Trans. Softw. Eng.*, Vol. SE-6, No. 5, pp. 440-449 (1980).
7) Coolahan, J. F. et al.: Timing Requirements for Time-Driven Systems Using Augmented Petri Nets, *IEEE Trans. Softw. Eng.*, Vol. SE-9, No. 5, pp. 603-616 (1983).
8) J. L. ピーターソン：ペトリネット入門, 共立出版, 東京 (1984).
9) 日本電電公社: DCNA データリンクレベルプロトコル, 財団法人日本データ通信会.

(昭和62年4月1日受付)
(昭和63年6月24日採録)

孫 立寧 (正会員)

1962年生. 昭和60年静岡大学工学部情報工学科卒業. 昭和62年同大学院修士課程修了. 現在東京大学大学院理学研究科情報科学専攻博士課程在学中. コンピュータ・グラフィックスに関する研究に従事. IEEE 会員.

落水浩一郎 (正会員)

昭和21年生. 昭和44年大阪大学基礎工学部電気工学科卒業. 昭和49年同大学院博士課程 (物理系専攻) 修了. 工学博士. 同年静岡大学工学部情報工学科講師. 昭和50年同助教授. 昭和63年情報知識工学科教授. ソフトウェア工学の研究に従事. ソフトウェア開発・保守支援環境の構築に興味をもつ. 電子情報通信学会, 日本ソフトウェア科学会各会員.