色付きペトリネットに基づくリアルタイム情報処理用ソフトウェア†

村田智洋†† 薦田憲久†† 解良和郎†††

制御用計算機におけるリアルタイム情報処理の多くは，非同期に発生する複数の処理要求に対する並行処理である．このような並行処理を行うソフトウェアを実現する方法の一つに，発生する各処理要求の内容を，それぞれトランザクションと呼ぶデータとして実体化し，それを処理の各ステップを受け持つ処理手続中にパイプライン的に流しながら多重処理を行う方法がある．トランザクション多重処理方式と呼ばれるこの方法は基本的にはデータ駆動型の並行処理とみなすことができ，その処理の流れはペトリネットモデルにより記述できる．本論文では，トランザクション多重処理ソフトウェアのプログラム記述にペトリネットのサブクラスである色付きセーフペトリネットの記述を導入し，色付きトークンによるトランザクションの実体化，トランザクションの流れの並行制御とトランザクションデータに基づく逐次処理の階層記述，およびそれらに基づくリアルタイム情報処理機能の実現等について述べる．

1. まえがき

制御計算機におけるリアルタイム情報処理の多くは，非同期に発生する複数の処理要求に対する並行処理である．このような並行処理を行うソフトウェアを実現する方法の一つに，発生する各処理要求の内容を，それぞれトランザクション* と呼ぶデータとして実体化し，それを処理の各ステップを受け持つ処理手続中にパイプライン的に流しながら多重処理を実現する方法がある．トランザクション多重処理方式と呼ばれるこの方法は，シリアリリユーザブル**な処理手続で並行処理を実現できるため，並行処理の実行管理が方式的に簡単であり，実行オーバヘッドも少ないため並行処理システムの簡易な実現方式としてリアルタイム情報処理のアプリケーションで多く用いられている．しかし，従来の汎用手続型プログラミング言語のみを用いてそのインプリメントを行う場合，処理手続間の同期，排他や情報交換等の記述が手続中にインプリシットに埋め込まれてしまう，トランザクション処理手続の記述と並行処理の実行管理機構の分離が明確でない等の問題があり，ソフトウェア開発容易性や保守性の点で問題があった．このため，アプリケーションレベルでより簡易にトランザクション多重処理ソフトウェアを開発，保守できる方式が求められている．

トランザクション多重処理は，基本的にはデータ駆動型の並行処理とみなすことができる．このような処理はデータの流れの並行制御とデータ処理の各ステップにおける逐次処理の記述を階層化し，階層間のデータの受け渡しに関するインタフェースを用意することにより，見通しよく，かつフレキシブルに記述することができる．データの流れの並行制御を記述するのに適したモデルにペトリネットモデル[1] がある．ペトリネットは並行動作システムの抽象モデルであり，すでに OS，通信ソフトウェア，FA システム，オフィス業務等のモデル化や解析[2)~5)]，FA 制御システムの制御プログラミングへの応用等を中心に研究が行われてきている[6)~8)]．

ペトリネットをトランザクション多重処理記述に適用する場合，ペトリネット上でトランザクションデータをどのように表現するか，トランザクションの流れの並行制御とトランザクション処理における逐次処理の記述をどう階層化するか，任意のトランザクション処理手続をペトリネットに組み込み実行管理するしかけをどうするか等の課題を解決する必要がある．

これらの課題に対し，本論文では，ペトリネットのサブクラスである色付きセーフペトリネットをトランザクション多重処理ソフトウェアのプログラム記述向きに拡張し，色付きトークンによるトランザクションの実体化，トランザクションの流れの並行制御とトランザクションデータに基づく逐次処理の階層記述等について提案し，それらの記述に基づくトランザクショ

† Real Time Information Processing Software Using Colored Petri Net Model by TOMOHIRO MURATA, NORIHISA KOMODA (Systems Development Laboratory, Hitachi, Ltd.) and KAZUO KERA (Omika Works, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所
††† (株)日立製作所大みか工場

* トランザクションという表現は，一般にデータベース更新要求の処理単位を意味するが，ここでは複数の手続間で受け渡される，任意の処理要求を表すデータを意味するものとする．

** シリアリリユーザブル (serially reusable) な手続とは，同時には1個のプロセスでしか使用できないが，使用が終わると再度ローディングせずに使用することができる手続のこと．

ン処理機能の実現方式を示す．最後に，提案方式を実装したリアルタイム情報処理ソフトウェアの実システムへの適用例について述べ，その有効性を示す．

2. 色付きセーフペトリネットモデル

色付きセーフペトリネットは，ペトリネット[1]のプレースに同時に最大1個のトークンしか入らないように制限したセーフペトリネットにおいて，トークンとアークに色を割当てて種別化したモデルであり，対象によってネットの記述を簡潔に行うことができる[9),10)]．

色付きセーフペトリネットの記述を**図1**（a）に示す．図1（a）の丸印がプレースを，縦棒がトランジションを表す．トランジションとプレースを結ぶ矢印はアークと呼ぶ．プレース中の刻印はトークンと呼び，ある状態の成立を示す．また，a1, a2, a3 がアークの色を，u1, u2 がトークンの色を示す．色付きセーフペトリネットの発火規則には様々な定義がありうるが，一般にはアークとトークンの色の組合せにより次のように定義される[9)]．

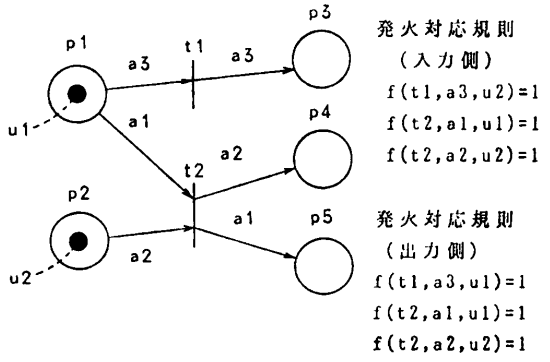
① 色の発火対応規則：トランジション tj の入力側アークの色の集合，および出力側アークの色の集合それぞれについて次の関数 $f(j,k,q)$ により定義される．

$$f(j,k,q)=\begin{cases}1\ ;\ \text{トランジション } tj \text{ に接続するアーク（色 a}k\text{）で，トークン（色 u}q\text{）が発火条件のもとで発火できる．}\\0\ ;\ \text{発火できない．}\end{cases}$$

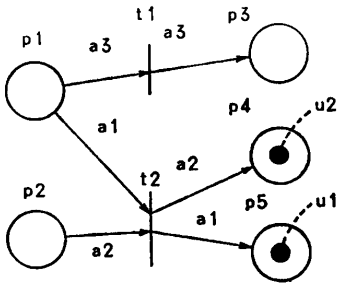
② 発火条件：トランジション j におけるすべての入力アークについて，それに接続するプレース内に①の発火対応規則を満たすトークンが存在し，かつ，トランジションのすべての出力プレースにトークンが存在しない．

③ 発火：トランジション tj において，それぞれの入力アークについて，それに接続するプレースから入力側アークに関する①の発火対応規則で指定された色のトークンを取り出し，それぞれの出力アークについて，それに接続するプレースに出力側アークに関する①の発火対応規則で指定された色のトークンを投入する．

図1（a）の色付きペトリネットは発火可能であり，発火後のマーキングは図1（b）となる．



（a） 色付きセーフペトリネットの記述例
（a） An example of colored safe petri net model.



（b） t2 が発火した後のマーキング
（b） The marking after t2 has fired.

図1 色付きセーフペトリネット
Fig. 1 Colored petri net.

3. トランザクション多重処理向きの記述仕様

トランザクション多重処理では非同期に発生する処理要求の内容をそれぞれトランザクションと呼ぶデータとして実体化し，それを処理の各ステップを受け持つシリアリリユーザブルな処理手続間で受け渡しながらパイプライン的に処理することにより，並行処理を行う．このような機能を実現するためには個々のトランザクションデータの格納とアクセスの制御，トランザクション処理手続の実行順序の制御，およびトランザクションデータの受渡の制御等の機能を実現する必要がある．

これらの機能の実現方法は種々ありうるが，アプリケーションレベルでよく用いられる方法としてはトランザクション受け渡しのためのキューを幾つか用意し，あらかじめインプリシットに決められたトランザクション処理フローに従い，各トランザクション処理手続がそれらのキューに関するトランザクションの取り出し，投入を自主的に行う方法である．しかしその方法ではトランザクション処理間の同期，排他や情報交換等の記述が個々の手続中にインプリシットに埋め込まれてしまい，トランザクション処理手続の記述と並行処理の実行管理機構の分離が明確でない等の問題があり，ソフト開発性や保守性の点で問題があった．

これらの内容を，アプリケーションレベルでわかりやすく記述するには，以下の記述が行えることが必要である．

（1） トランザクション処理の実行順序に関する制御（同期，排他，逐次）を，トランザクション処理手続の内部処理と分離してわかりやすく記述できる．

（2） 処理手続間のトランザクションの流れ，トランザクションのキューイング等の制御をわかりやすく記述できる．

（3） トランザクションの処理中に発生した異常に対し例外処理をわかりやすく記述できる．

2章で述べた色付きセーフペトリネットモデルはシステム内に存在する複数のトランザクションの流れを陽に表現できるため，トランザクション多重処理の見通しの良い記述に向いている．次章では，色付きセーフペトリネットにもとづく記述仕様について述べる．

4. 色付きセーフペトリネットによるトランザクション処理の記述

4.1 処理トランザクションの定義

処理手続間で受け渡されるトランザクションを，トランザクションデータを格納する記憶場所（スロット）を持つトークンとして実体化し，トランザクションの流れをペトリネットのトークンの流れとして記述する．各トークンは，それぞれタイプ，スロット値を持つデータ型として**図2**に示すように定義し，処理要求を表すトランザクションが発生した時点で，発生したトランザクションの情報を持つトークンをペトリネットのプレース内に発生させる．各トークンは個別の情報を反映する色付きトークンとなる．

4.2 プレースによるトランザクション処理手続の定義と実行管理

実際のトランザクション処理を行う任意の手続をプレースに定義し，処理の起動と完了判定，起動した手続とトークンとの情報交換等の制御をペトリネットのモデルに基づいて行う．具体的には**表1**に示す5種類のプレースを設け，それぞれ次に示す機能拡張を行う．

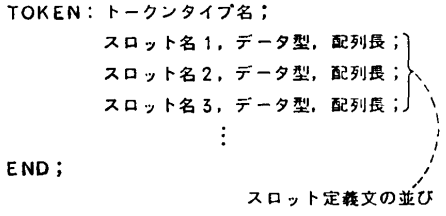


図2 トークン情報の定義
Fig. 2 Definition of token data structure.

（1） 処理プレース

トランザクション処理の各ステップの処理を行う任意の手続をプレース手続として実行する．

（a） プレース手続の指定：プレース手続の定義例を**図3**に示す．プレース手続はその起動方式と手続名（英数文字，カナ文字を用いた手続名称）で指定する．

表1 機能拡張したプレースの種類
Table 1 Kinds of function-extended places.

種別	記法
処理プレース	
起動プレース	
キュープレース	
ダミープレース	
判定プレース	

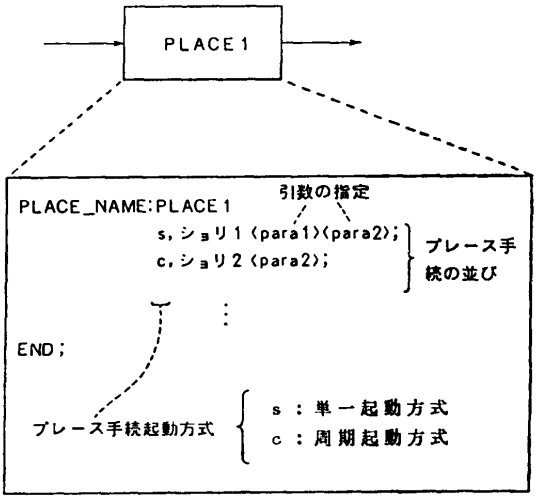


図3 プレース手続の定義
Fig. 3 Assignment of place procedures.

手続名称のうち〈 〉で囲まれた変数は引数名の指定である．

（b） プレース手続の実行：処理プレースには，指定したプレース手続は，プレースにトークンが投入された時点で実行される．プレースに定義した処理手続のシリアリリユーザブル性を保障するために，プレースに同時に入れるトークン数は最大1個（セーフ条件）とする．プレース手続の起動方式は次の2つである．

① 単一起動方式：トークンの投入時点ごとに，プレース手続をサブルーチンとして1回コールする．

② 周期起動方式：トークンの投入時点でプレース手続をコールし，以後そのリタンコードが真となるまで周期的にコールする．

各プレースには，これらの起動方式で実行する手続を複数個，実行順に定義する．単一起動方式と周期起動方式の手続を組み合わせることにより，例えば単一起動方式の手続で制御信号等を出し，その結果を周期起動方式の手続で判定するような非同期処理がプレース単位で行える．

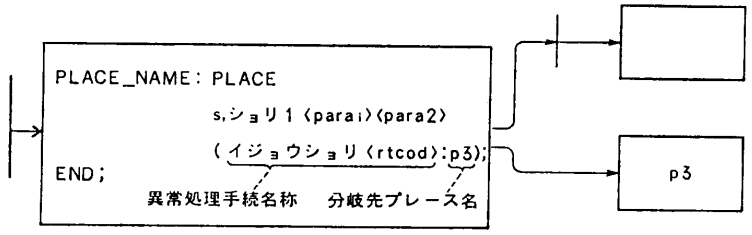
（c） プレース手続とトークンの情報交換：プレース手続では，コールされた時点でそれが定義されたプレース内に存在するトークンのスロット情報をアクセスできる．プレース中のトークンは，一種のオブジェクトデータとして扱い，トークンのスロット値の設定，参照，更新等の操作をマクロ（トークンマクロ）を用いて行う．プレース手続でトークンの内容をアクセスするには引数として，システム変数〈TOKEN〉を指定することにより，プレース内のトークンのアクセスキーが渡され，そのキーをもとにトークンマクロにより手続内でスロット情報を参照，更新する．

（d） 異常処理の記述：処理プレースでのプレース手続の実行結果はリタンコードとして返し，それにより処理の成否を判定する．プレース手続の後にそれが異常終了した場合に行う処理を '(異常処理手続名称)' の形で指定することにより，プレース手続がエラーリタンした場合に，指定した異常処理手続が実行される．引数としてシステム変数〈RTCOD〉を指定することにより，異常処理手続の中でリタンコードの値を異常要因として参照することができる．異常処理が成功した後，特定のプレースにトークンを強制的に分岐させるには '(異常処理手続名：分岐先プレース名)' を指定する．異常処理不成功の場合は，トークンの移動は禁止される．処理プレースの中にプレース手続とそれに対する異常処理を並べて記述するのは記述上の見やすさとモジュラリティの向上のためである．これらは，別々の処理プレースとして等価なセーフペトリネットで記述することができる．異常処理付きの処理プレースの記述例を図4（a）に，それと等価なセーフペトリネットを図4（b）に示す．

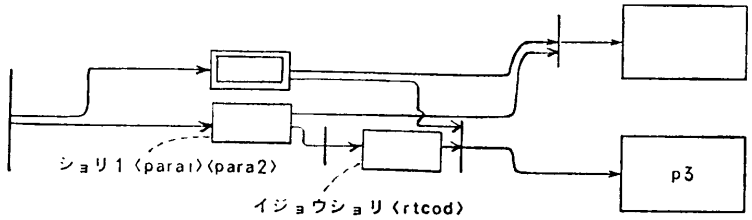
（2） キュープレース

処理プレース間で受け渡すトークンをキューイングしたい場合に用いる．キュープレースからのトークンの取り出しはFIFO順が標準であるが，キュープレースにキュー操作手続を指定することによりFIFO順以外の取り出しを行うことができる．キュー操作手続はキュープレースからトークンを取り出す直前に実行され，その中でシステム変数〈GETP〉に取り出しトークンの位置をセットすることにより指定位置のトークンが取り出される．キュープレースにはキュー操作手続以外のプレース手続は指定できない．

キュープレースは直接にはセーフ条件を満たさないが，容量がnのキュープレースは，容量が1のセーフなキュープレースを用いて等価な



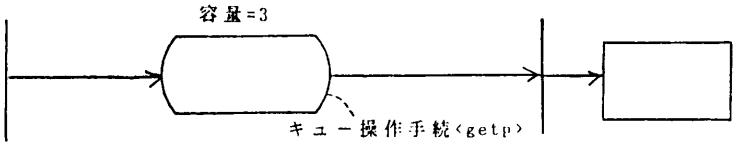
（a） プレースにおける異常処理の記述
（a） Description of exception handling procedure in a place.



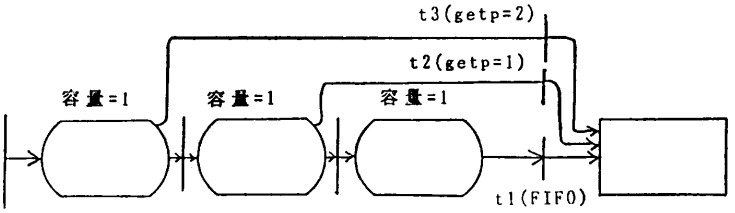
（b） （a）の記述と等価なセーフペトリネット
（b） An equivalent safe petri net of description (a).

図4 異常処理の指定
Fig. 4 Assignment of an exception handling.

セーフペトリネットで記述することができる．プレース容量が3のキュープレースと等価なセーフペトリネットを**図5**に示す．図5において，トランジション t1 を発火させることで FIFO の取り出しを，トランジション t2,t3 の任意の1つを発火させることでキューの任意の位置からの取り出しを表現できる．

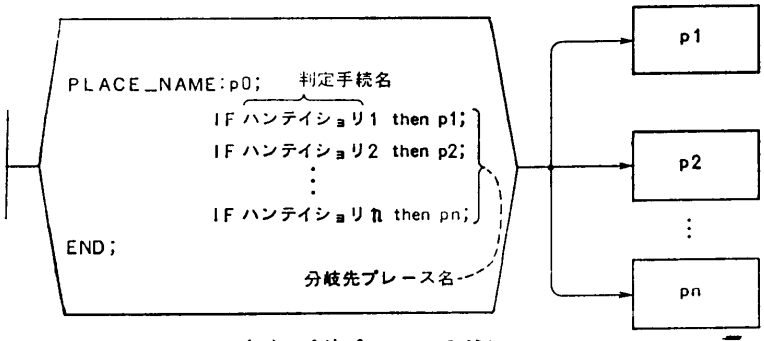


(a) キュープレースの記述例
(a) Description of a queue place.

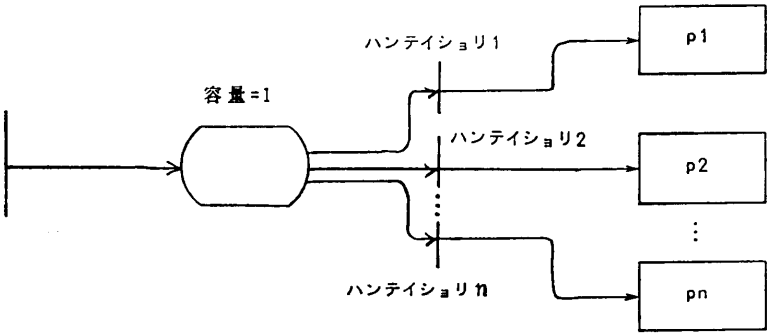


(b) (a) の記述と等価なセーフペトリネット
(b) An equivalent safe petri net of description (a).

図5 キュープレース
Fig. 5 A queue place.



(a) 分岐プレースの記述例
(a) Description of conditional token branch place.



(b) (a) の記述と等価なセーフペトリネット
(b) An equivalent safe petri net of description (a).

図6 分岐プレース
Fig. 6 A conditional token branch place.

(3) 起動プレース

特定のイベント発生に対するイベント処理の記述に用いる．起動プレースは入力アークを持たない．起動プレースに検出すべきイベント番号とイベント検出時に実行すべきプレース手続を指定することにより，指定したイベントが発生した時点で起動プレースの手続が実行される．起動プレースのプレース手続ではトランザクション処理要求の発生に対応して，トークンの発生処理等を行う．起動プレースはまた，キュープレースとしての機能を持っており，トークンをキューイングすることにより複数のイベントやトランザクション処理要求の発生を記憶することができる．

起動プレースも直接にはセーフ条件を満たさないが，プレース容量が n の起動プレースはプレース容量が1の起動プレースとプレース容量が1のキュープレースを用いて図5と同様のセーフペトリネットで記述することができる．

(4) 分岐プレース

トークンの内容やその他の条件によりトークンの流れの分岐を動的に制御する必要がある場合に用いる．分岐プレースの記述は，**図6**(a)に示すように分岐先のプレースを分岐プレースの出力側にアークで直接結び，プレース内に (IF 判定手続名 THEN 分岐先プレース名) の形式のルールを複数個並べて指定する．分岐プレースにトークンが投入されると，ルールの並びの IF 部に指定された判定手続が実行され，そのリタンコードが真であれば THEN 部に指定されたプレースにトークンを投入する．すべてのルールの IF 条件が成立しない場合は，いずれかの分岐条件が成立するまで一定の周期ごとに判定を行う．なお，分岐プレースには最大1個の

トークンしか入れない．分岐プレースは，容量1のキュープレースとその複数の出力側トランジションに付加した分岐ルールを，見やすさの観点から分岐プレース内にまとめて記述したものである．図6(a)の分岐プレースと等価なセーフペトリネットを図6(b)に示す．

(5) ダミープレース

複数の処理プレースの実行を排他制御したり，ある状態の成立を記憶するのに用いる．ダミープレース中には最大1個のトークンしか入れない．ダミープレース中のトークンによりダミープレースの入力側トランジションの発火を抑止することにより，ダミープレースの入出力トランジションで囲まれた区間での処理プレースの実行を排他制御する．ダミープレース中のトークンは情報を持たず，ダミープレースの入力側トランジションの発火時にダミープレースに投入され，ダミープレースの出力側トランジションの発火時に自動的に消滅する．図7において，ダミープレース p4 により，プレース p1,p2,p3 の少なくとも1つにしかトークンが投入されないように制御している．

4.3 トークンによるトランザクションの流れの制御

(1) 色付きペトリネットの発火規則の簡略化

トランザクションを実体化したトークンは個別の情報を反映しており，トークンの流れの制御をトークンの色対応に記述する必要がある．これに対しては，2章で述べた色付きペトリネットの発火規則を用いることができるが，トランザクション処理型の多くのアプリケーションでは，処理手続間で受け渡すトランザクションのデータ型（トークンの色）が処理フローごとに静的に決まっている場合が多い．このため，2章で述べた発火対応規則 $f(j, k, q)$ によって発火条件を記述すると，不要に記述が繁雑になる恐れがある．

これを避けるため，本論文では，2章で述べた色付きペトリネットの発火規則に次の制約条件を付加し，

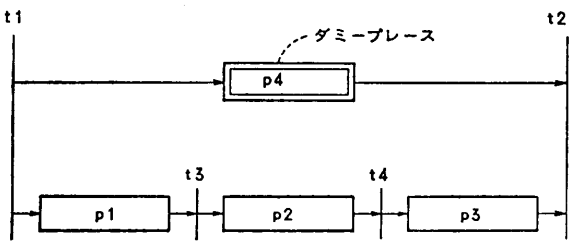


図7 ダミープレースによる排他実行区間の記述
Fig. 7 Description of exclusive execution section using dummy place.

もとの色付きペトリネットの発火規則のサブセットとして，トランザクション処理型のアプリケーション記述に適した形で色付きペトリネットの発火規則を簡略化することを考える．

[制約条件1] 各トランジションにおいて，同色の入力側アークと出力側アークに関する色の発火対応規則 $f(j, k, q)$ が同一である．

[制約条件2] 出力側アークの色の集合が入力側アークの色の集合の部分集合である．

これらの制約条件は，処理手プレース間で受け渡すトークンの色が保存されるという条件を反映している．これらの条件のもとで，もとの色付きペトリネットの発火規則と等価な発火規則として次の[発火規則]，[発火条件]，[発火処理]が得られる．

[発火規則1] トランジション tj が発火可能となるそれぞれの入力側プレース内のトークン色の組合せを規定する．

[発火規則2] トランジション tj の発火時，どの入力側プレースのトークンがどの出力側プレースにトークンの色を保存しつつ移動するかの組合せを規定する．

[発火条件]

① [発火規則1]が成立している．

② すべての出力プレースにトークンが無い．

[発火処理]

[発火規則2]の指定に従って入力側プレースのトークンを出力側プレースにトークンの色を保存しつつ移動させる．

上記の[発火規則]，[発火条件]および[発火処理]の内容が，[制約条件1，2]のもとで，もとの色付きペトリネットモデルの発火規則と等価なことを以下に示す．

(証明)

トランジション tk に関し，入力側アークと入力側プレースの接続はそれぞれ一意に決まるので，トランジション tk に関し，入力側プレースはそれが接続している入力側アークの色 ak と同じ色を持つと考えてよい．したがって，入力側の色 ak についての発火対応規則 $f(j, k, q)$ は，ak と同色とみなせる入力側プレースのそれぞれに色 aq のトークンが存在するという条件（上記[発火規則1]）におきかえることができる．すなわち，上記[発火規則1]と発火対応規則 $f(j, k, q)$ は等価である．

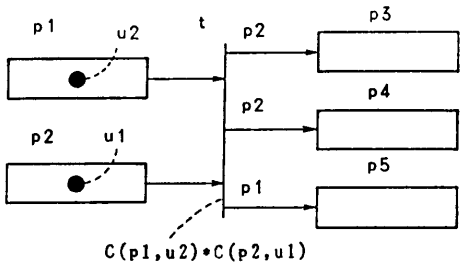
同様に，トランジション tk に関し，出力側アークで tk に接続された出力側プレースは，それが接続し

ている出力側アークの色と同じ色を持つと考えてよい．また，[制約条件2]により出力側プレースと同色の入力側アークが必ず存在する．トランジション t_k の発火時，色 a_k とみなせる出力側プレースには，[制約条件1]により，色 a_k の入力側アークと同色とみなせる入力側プレースのトークンと同じ色のトークンが発生する．よって，トランジションの発火による出力側プレース内のトークンの色の組合せは上記[発火規則2]により一意に決まる．したがって，出力側アークの色 a_k についての発火対応規則 $f(j, k, q)$ の指定と上記[発火規則2]とは上記[制約条件1，2]のもとで等価である．

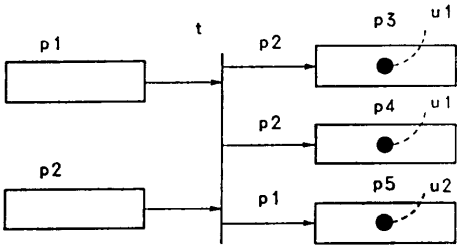
以上により，上記2つの制約条件のもとで，もとのモデルの発火規則と本論文中で簡約化した発火規則は，等価である．　(証明終わり)

(2) 発火規則の具体的記述法

上記[発火規則1]，[発火規則2]の具体的記述形式を図8(a)に示す．[発火規則1]はプレース内のトークンの色を調べるマクロ 'C (プレース名，比較するトークンの色)' を用いて論理式で記述する．また，[発火規則2]はトランジションの出力アークにどの入力プレースからのトークンを流すかを入力プレース名で指定することにより記述する．ただし，入力プレースが1つの場合は出力アークへのプレース名の指定は不要である．また，[発火規則1]の指定は省略可能であり，これを省略した場合，入力プレース内のトークンの色の組合せに関する条件は無条件で成立しているものとみなされる．



(a) トークンの流れの制御の記述例
(a) An example of token flow control description.



(b) tが発火した後のマーキング
(b) The marking after fire of t.

図8 入力プレース指定によるトークンの流れの制御の記述
Fig. 8 Description of token flow control using input place assignment.

(3) トランザクションの制御に必要なその他の発火条件の追加

複数の入力プレースにおけるトランザクション処理のためのプレース手続実行終了の同期を取るための判定を発火条件として追加する．また，トランジションの発火時に成立すべき任意の条件をトランジション条件として追加する．ここで，トランジション条件とは，各トランジションに必要に応じて付加できる手続であり，トランザクションの制御に必要な任意の条件の成否をリタンコードとして返す．上記[発火規則2]の指定も以後，トランジション条件に含める．

この結果，[発火条件]は次のようになる．

① すべての入力プレース状態が正常完了である．(入力プレースにおけるプレース手続実行終了の同期)

② すべての出力プレースにトークンが無い．(プレース手続のシリアリリユーザブル条件)

③ トランジション条件が真．(①，②以外の任意の条件の成立)

(4) 発火処理の実現

上記発火条件成立時の発火処理は以下のようになる．

① トランジションの入力プレースからトークンを取り出し，そのプレース名が指定された出力アークに接続するプレースに投入する．

② 複数の出力アークに同一の入力プレース名が指定されている場合は，指定された入力プレースから取り出したトークンをそれらのアークに接続しているプレースの数だけ複製し，それぞれのプレースに投入する．

③ 出力カークに指定されていない入力プレースのトークンは消滅させる．

以上の処理により，色付きトークンの移動の制御を行う．これによれば，図8(a)のt発火後のマーキングは図8(b)となる．

これらの発火条件の簡略化により，すべてのトランジションについて $f(j, k, q)$ を指定することにともなう記述の繁雑さを軽減しつつ，トランザクション処理型のアプリケーションに必要な色付きトークンの流れの制御を簡潔に記述することができる．

5. インプリメンテーション

4章で述べた仕様に従って記述した色付きセーフペトリネットの作成，実行を行う処理系の概要について述べる．

5.1 ソフトウェア構成

ネット処理系のソフトウェア構成を図9に示す．インタプリタ，エディタ，モニタはそれぞれ汎用リアルタイム OS の下で動作するタスクとしてインプリメントしている．エディタはグラフィカルなペトリネット入力をサポートする．ペトリネットのプレースにリンクするプレース手続はアプリケーション対応に作成し，サブタスク，またはサブルーチンとしてインタプリタに登録する．インタプリタは複数のネットについて，以下のテーブル，ファイルの内容にもとづき，プレース間のトークンの移動，プレース手続の実行管理，トークン情報のアクセス管理を行う．

① ネット定義情報テーブル：3章で述べた仕様にもとづき記述したネットの定義情報（複数のネットの定義が可能）を実行形式で記憶する．格納情報には，定義したプレースとトランジションの接続関係を記憶するネット構造定義情報，起動すべきプレース手続，および検出すべきイベント等を記憶するプレース定義情報，どのアークにどの入力プレースのトークンを流すかを記憶するアーク定義情報，各プレース手続のエントリアドレス，引数アドレスを記憶するプレース手続リンケージ情報等がある．

② ネット状態テーブル：プレース内のトークンの配置（マーキング情報），および各プレースにおけるプレース手続の実行状態を記憶する．

③ ネット起動管理テーブル：定義した各ネットの処理優先度，起動の有無等を記憶する．

④ トークン情報ファイル：各ネット中のトークンのスロット情報を記憶する．トークンが発生するとトークン情報ファイル内にスロット情報格納用のエリアを確保し，そのエントリアドレスがアクセスキーとして記憶される．ファイルはトークンのタイプ別になっており，1つのトークンのスロット情報は連続した領域に割り付けられる．また，トークンのスロット情報のアクセスのため，トークンのデータ構造をタイプ別にテンプレートとして記憶している．

これらのテーブルの概略構成を図10に示す．モニタはこれらのテーブルの内容をペトリネット図とともにモニタ端末に表示する．

5.2 インタプリタによるネット実行管理

（1） プレースの状態管理：プレースは，待機中，処理待ち，処理中，正常完了，異常完了の5つの状態をとる．トークンが存在しないプレースの状態は待機中である．プレースにトークンが投入されると処理待ちに，続いてプレース手続がコールされると処理中の状態となる．プレースに定義したすべての手続の実行が正常に終了した場合，プレース状態は正常完了となり，トークンがプレースから出て行くと待機中に戻る．プレース手続の実行がエラー終了した場合のプレースの状態管理は次のように行う．

① プレース状態を異常完了とし，トークンの移動を禁止する．

② プレースに異常処理手続が定義されており，その実行が正常完了した場合，そのプレースに他に実行すべきプレース手続が無ければプレース状態を正常

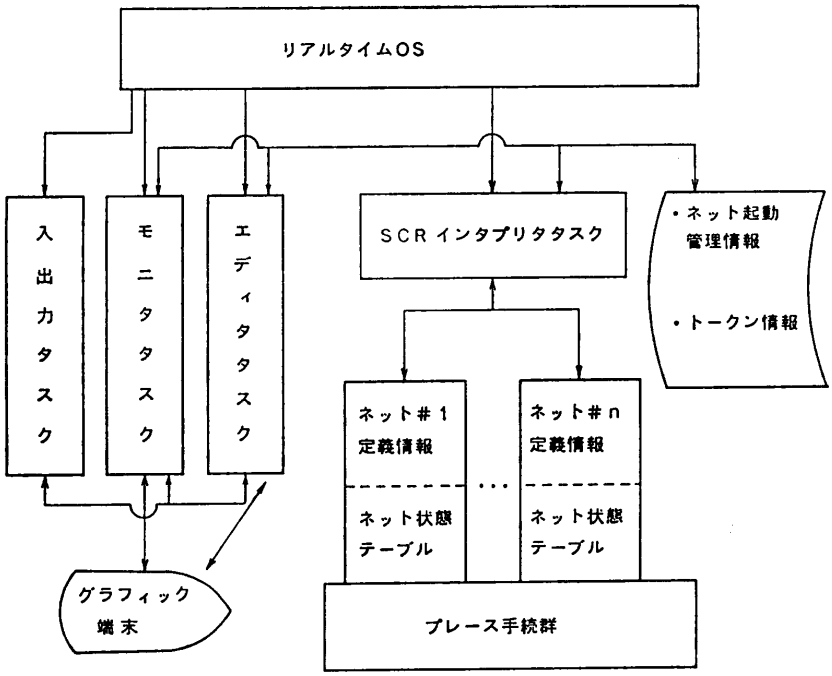


図9 ネット処理系のソフトウェアアーキテクチャ
Fig. 9 Configuration of net processing software.

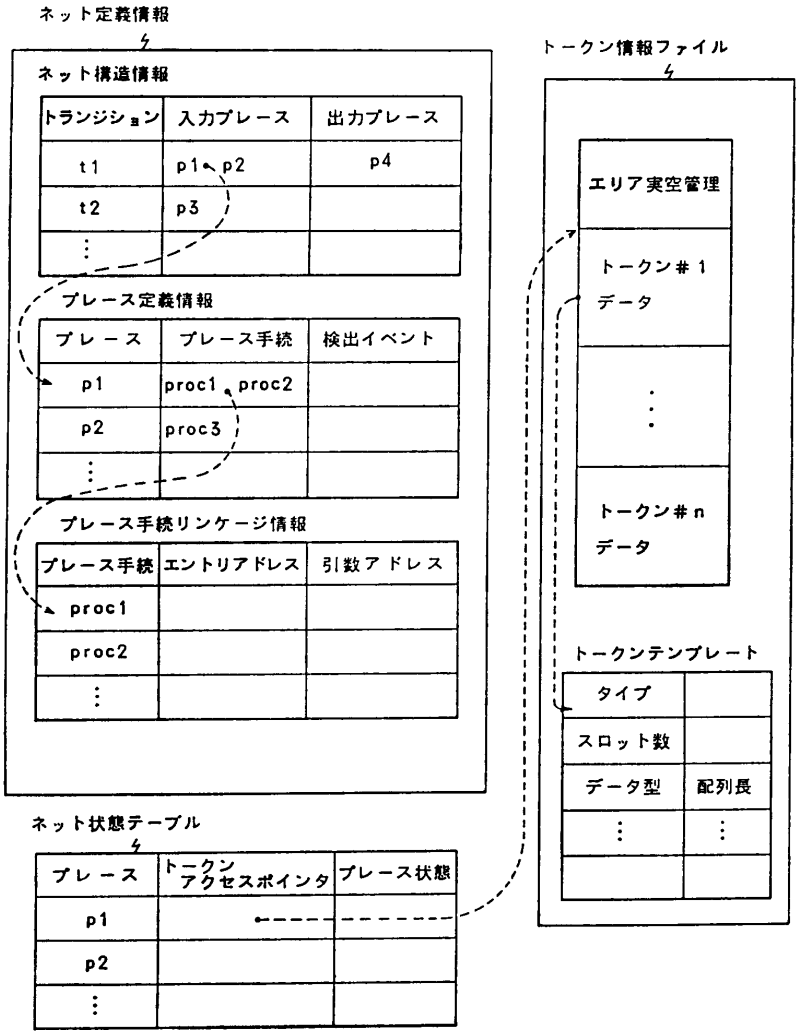


図 10 ネット処理系のテーブル構成
Fig. 10 Configurations of net execution management tables.

完了とし，そのプレースの処理を終了する．他に実行すべきプレース手続があればプレース状態を処理中とし，その実行を行う．

③ 異常処理手続の実行が異常終了した場合は，プレース状態を異常完了としたままそのプレースの処理を終了する．

（2） ネットの実行：作成したネットは，モニタ端末からネット名を順次入力することにより起動する．ネットインタプリタは，次の処理によりネットを実行する．

（a） イベント処理；起動プレースに指定されたイベントを割込みにより検出し，実行中のネットの該当する起動プレースの処理，およびプレース状態の更新を行う．

（b） トークン移動処理，およびプレース実行処理；ネット起動管理テーブルに指定された優先度に従い，起動されたネットの内から実行するネットを決定し，限定スキャン方式[7]とよぶ方式により発火候補トランジションを選択する．そしてそのトランジション条件として指定された手続の実行結果が真となったトランジションを発火させ，トークンを移動させる．続いて，実行中のネットに含まれるすべての処理待ち状態のプレースを選びだしそのプレース手続を順次実行し，プレース状態を更新する．実行対象のネット内に処理待ち状態のプレースが無くなるまでこの処理を繰り返し，それが無くなった時点で次の優先度のネットを実行の対象とすることにより，複数ネットの多重処理を行う．

（3） トークン情報のアクセス管理：トークンの情報はトークンファイルと呼ぶオンラインファイルで一元管理し，標準のアクセスマクロを用意することによりプレース手続からのトークン情報のアクセスを実行する．トークンのスロット情報のアクセスでは，まずプレース内のトークンのアクセスキーとタイプを調べ，それをもとに該当するタイプのテンプレートを用いてスロット情報格納アドレスを求めた後，スロット情報の読みだし，更新を行う．また，プレース手続中でさらにタスクが起動され，複数のタスクが同一のトークンに対してアクセスする場合の競合を想定し，トークンファイルに対するアクセスの排他を行うロックマクロを提供している．（ただしタスクそのものの実行管理は本処理系の対象外である．）

5.3 マンマシン機能

（1） プレース手続の単体テストモード：作成したペトリネットのテストは，グラフィック画面を用いて，対話的に行うことができる[11]．単体テストモードでは，画面上のネットを見ながらテストしたいプレースをライトペンで指定することにより，そのプレース

に定義したプレース手続を単体で実行し，リタンコード，および引数の値を画面上に表示する．

（2） 実行中ネット状態の表示および異常回復のためのオフライン操作：実行中のネットのトークンのマーキング，プレースの状態等は必要に応じてカラーモニタ端末に表示させることができる．特に，プレース手続が異常終了して異常完了状態となったプレースは赤色で表示されるので，エラー発生の状況把握が容異である．異常完了状態のプレースに対しては，そのプレースを含むネットの実行をいったん停止し，プレース手続の再実行や，プレース内のトークンのデータ修正等の回復操作を手動で行った後ネットを再起動する．

（3） 履歴収集機能：異常の回復操作等でネット状態を手動で修正する場合に操作ミスが生じると，回復に必要な情報が失われる場合がある．このようなことを防ぐために，トークンの投入，抜取り，トークンデータの変更等のネットの状態変更を行った履歴を取る機能をモニタに持たせている．これにより，操作ミスの回復を可能としている．

6. アプリケーション

5章で述べた処理系は SCR (Station Controller) と名付けたソフトウェアとして 16 ビットマイクロコンピュータシステムに実装され，すでにいくつかのシステムに適用されている[4),7),8),11)]．ここでは，列車発着案内自動放送システムへの SCR の適用例について述べる．

列車発着案内自動放送システミは，駅に近づいて来た列車を経電装置（列車の接近を電気的に検出する装置），および列車追跡装置により検出し，あらかじめ作成された列車発着スケジュールを参照して列車を同定し，発着案内表示装置，および音声案内装置により，乗客への列車発着案内を自動的に行うシステムである．システム構成を図 11 に示す．

SCR では列車のトラッキング処理（駅構内の複数の番線への列車の進入，出発の監視処理），発着案内表示，放送用のテキストデータの組み立て，送信処理，異常監視等を行っている．これらの処理は駅構内の各番線ごとに次の 3 種類のネットにより記述し，それを複数番線に関して並行処理している．

① 起動要因判定用ネット：経電装置から入力される列車検出割込みを起動プレースで検出しトークンを発生させ，割込み要因の解析，合理性チェックを行った後，起動要因をトークンにセットし列車認識ネットに渡す．

② 列車認識用ネット：スケジュールファイルより列車 ID を取り出し，列車追跡装置から受け取ったデータとの一致判定を行った後，列車の入線をトラッキングし列車 ID をトークンにセットし放送制御ネットに渡す．

③ 放送制御用ネット：受け取ったトークンの列車 ID をもとに，発着案内表示，放送用のテキストデータを組み立て，発着案内表示装置，および音声案内装置へのテキストデータ送信処理を行う．

全体のネット数は 27，プレース総数は約 800 であり，作成したプレース手続は約 50 個である．ネット記述の一部を図 12 に示す．また，プレース手続の起動オーバヘッドはプレース当り約 15ms であった．これらのネットの実行中に不合理なトークン情報等を検出

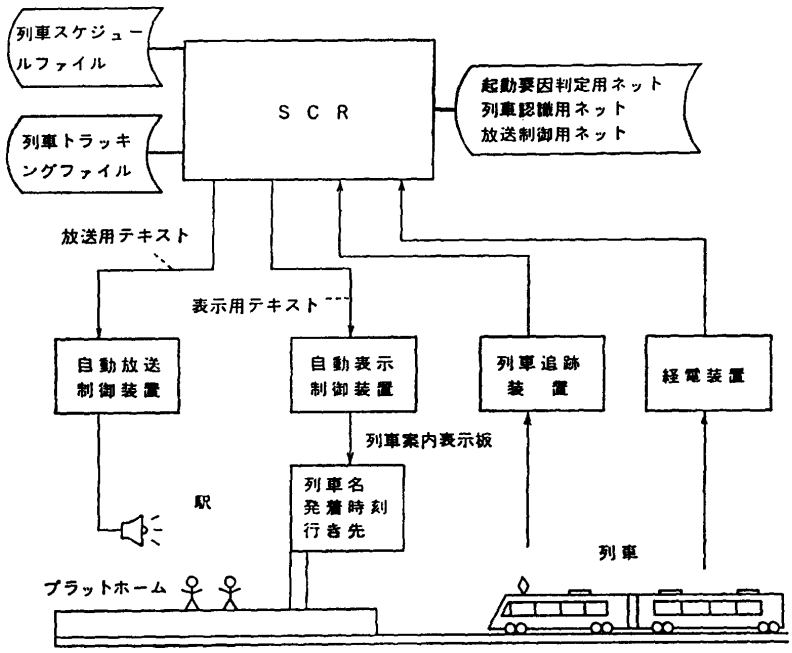
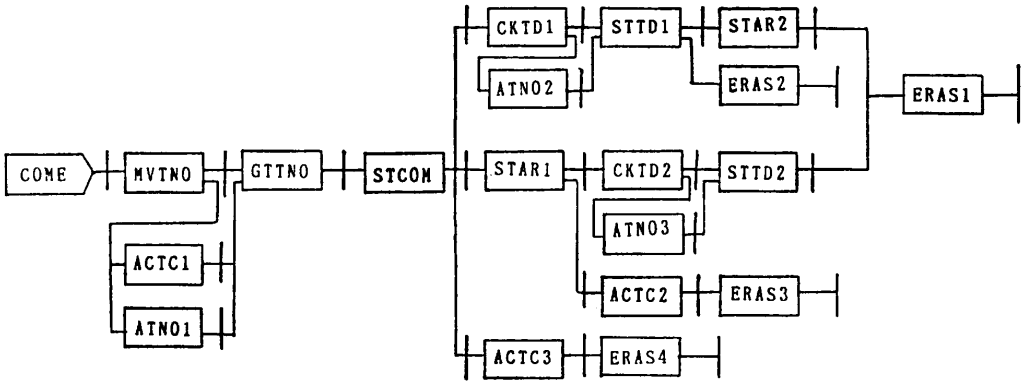


図 11 列車発着案内自動放送システムの構成
Fig. 11 Configuration of automatic passenger guidance announcement system.



プレースの意味

COME	接近信号でトークン発生	CKTD1, 2	列車 ID 入力参照
MVTNO	列車 NO. 移動処理	ERAS1, 2, 3, 4	列車 NO. 消去
GTTNO	列車 NO. 取り込み	ACTC1, 2, 3	CTC 異常報告
STAR1, 2	到着信号入力セット	ATNO1, 2, 3	列車 NO. 異常報告
STCOM	接近信号入力済セット	STTD1, 2	列車 ID 入力セット

図 12 列車認識用ネットの記述（一部）
Fig. 12 Description of a net for train recognition (partial).

した場合には，アラームの出力，トラッキング情報の修正等の必要な異常処理を行った後，手動で不要なトークンを消去し，ネットの状態をイニシャライズしている．

SCR の適用によりソフトウェア開発，および現地調整，保守で従来の手続型言語のみを用いた場合に比べ約 30% から 50% の工数低減が得られた．その内訳は，プログラム記述がネット記述とプレース手続に階層化され，プログラミングが容易化されたことによるプログラム設計段階の工数低減が約 10〜20%，SCR のマンマシン機能によるテスト・デバッグ作業の操作性向上によるテストや現地調整期間の短縮が 20〜30% である．また，テスト・デバッグ以降の工程で，ネット記述の内容を見通し良く追加・修正できる点で効果があった．

7. むすび

トランザクション多重処理によるリアルタイム情報処理ソフトウェアの簡易な実現方式として，色付きセーフペトリネットによる処理記述方式を提案し，その処理系と適用例について述べた．提案方式をツール化した SCR は，アプリケーションソフトウェアの開発に有効に利用されている．

謝辞 本研究に関して御討論いただいた東京工業大学市川惇信教授に感謝いたします．また，本研究の機会を与えていただいた（株）日立製作所システム開発研究所所長川崎淳博士，本研究の御指導をいただいた同システム開発研究所春名公一博士，松本邦顕博士に感謝いたします．また，本システムの開発，適用に関し御協力いただいた同大みか工場の各位に感謝いたします．

参考文献

1) Peterson, J. L.: *Petri Net Theory and the Modeling of Systems*, Prentice Hall Inc., Englewood Cliffs, N. J. (1981).
2) 山崎ほか：ペトリネットの理論と応用，情報処理，Vol. 25, No. 3, pp. 188-197 (1984).
3) King, R. et al.: A Methodology and Tool for Designing Office Information Systems, *ACM TOOIS*, Vol. 3, No. 1, pp. 2-21 (1985).
4) 薦田ほか：ペトリネットの FA 制御への応用，システムと制御，Vol. 30, No. 1, pp. 42-55 (1986).
5) 市川ほか：事象駆動型システムの制御，計測自動制御学会論文集，Vol. 21, No. 4, pp. 8-14 (1985).
6) 長谷川ほか：非連続システムのためのマーク流れ線図の提案，計測自動制御学会論文集，Vol. 20, No. 2, pp. 122-129 (1984).
7) 村田ほか：ペトリネットにもとづく高フレキシブル FA 制御システム，計測自動制御学会論文集，Vol. 20, No. 9, pp. 844-851 (1984).
8) Murata, T. et al.: A Petri Net Based Controller for Flexible and Maintainable Sequence Control and Its Applications in Factory Automation, *IEEE Trans. on Industrial Electronics*, Vol. IE-33, No. 1, pp. 1-8 (Fed. 1986).
9) 林ほか：色付きペトリネットによる論理回路シーケンス制御系の記述，電子通信学会第 2 回ネ

ット理論研究会資料, pp. 85-92 (1986).
10) Viswanadham, N. et al.: Coloured Petri Net Models for Automated Manufacturing Systems, *Proc. of 1987 IEEE Int. Conf. on Robotics and Automation*, pp. 1985-1990 (Mar. 1987).
11) 村田ほか: ペトリネットのシステム制御への応用—その実現方式と適用例—, 電子通信学会第1回ネット理論研究会資料 (1986).

(昭和62年 8月 28日受付)
(昭和63年10月 7日採録)

村田 智洋 (正会員)

昭和28年生. 昭和54年九州大学工学部情報工学科卒業. 同年(株)日立製作所に入社. システム開発研究所にて事象駆動型システムのモデリングおよびシミュレーション, FAシステムの制御プログラミング技術の研究を経て, 現在計算機システム周辺装置の制御ソフトウェアの研究に従事. IEEE, 電子情報通信学会, 計測自動制御学会などの会員. 61年度計測自動制御学会論文賞受賞.

薦田 憲久 (正会員)

昭和25年生. 昭和47年大阪大学工学部電気工学科卒業. 49年同大学院修士課程修了. 同年(株)日立製作所に入社. システム開発研究所にてシステム計画技法, システム構造化技法, ペトリネットなどの事象駆動型システム, 生産・流通業向情報処理システムなどに関する研究に従事. 現在同研究所第1部主任研究員. 56～57年UCLAに留学. 工学博士. IEEE, 電気学会, 電子情報通信学会, 計測自動制御学会などの会員. 61年度計測自動制御学会論文賞, 62年度計測自動制御学会技術賞受賞.

解良 和郎 (正会員)

1948年生. 1970年山形大学工学部電気工学科卒業. 同年(株)日立製作所入社. 現在大みか工場計算制御ソフト第二設計部主任技師. この間主として, 産業FAシステムおよび鉄鋼, 鉄道, 交通分野の計算制御システムの開発に従事. 電気学会会員.