無人搬送制御システムへのペトリネットの応用*

長尾 陽一**・太田 英明**・占部 博信**・
熊谷 貞俊***・児玉 慎三****

Application of Petri Net to Automated Guided Vehicle Control System*

Yoichi NAGAO**, Hideaki OHTA**, Hironobu URABE**,
Sadatoshi KUMAGAI*** and Shinzo KODAMA****

This paper describes an application of a programming system, K-NET to the development of control software for an automated guided vehicle system. K-NET is based on a high-level Petri net which makes it simple to express sequence controls. Procedural language can be also used for information processing. K-NET has an editor to input and edit control specifications expressed with Petri net; a simulator to verify edited specifications; a generator to convert the net to C source programs for a controller; a reporter to print control specifications; and a monitor to display controller status in real-time. K-NET can support all development stages including general design, detailed design, programming and testing. K-NET has been used in the development of control software for an automated guided vehicle system, and results show a 2/3rds cost-saving over development with conventional methods in which only procedural language is used.

1. ま え が き

最近, FA (Factory Automation) の進展に伴い, 生産現場において複数台のロボット, NC 機械, 搬送装置, 立体倉庫などを組み合わせた生産システムを構築し, 製品の多様化, 製品サイクルの短期化に対応する傾向が強くなっている。工場内の多数の作業ステーション間で材料や製品を搬送するための AGV (Automated Guided Vehicle) は, そのフレキシビリティ性のゆえに特に注目されている。この AGV を多数導入して, 全体として効率的に運用することにより, 工場内物流を担う無人搬送システムが構築されるようになってきた。

無人搬送システムを制御するための計算機ソフトウェアの開発においては, 従来 Basic や C などの手続き型言語を用いて, 人間の設計した制御仕様に基づいてプログラミング作業を行っている。これらの方法では,

- プログラミング時に誤りが発生し易い,
- 作成したプログラムが設計した仕様を満足するかどうかの検証が困難である,
- 作成した人間でないとプログラムで実現された制御の流れを理解するのが困難であり, 制御ソフトウェアの保守が容易でない,

といった問題があった。

これに対して, 最近, ペトリネットを用いたシーケンス制御システムの記述モデルが注目を集めている。ペトリネットは, 人間が理解し易い図的表現であり, 無人搬送システムにおいて要求される同期, 排他, 並行動作といったシーケンス制御上の機能の表現が容易である。そのため, ペトリネットに基づいたシーケンス制御のためのプログラミングツールがこれまでにいくつか開発されている^{1)~3)}。しかし, ペトリネットの利用だけでは高度な無人搬送システムの制御ソフトウェアを構築するには不十分であった。

そこで, 筆者らは, 上記問題を解決するために, 従来の手続き型言語とペトリネットという図的言語を融合して, 複雑な制御仕様を簡単に記述できるモデルを提案し, 概略設計, 詳細設計, プログラミング, テストの各開発段階を一貫して支援する FA 制御ソフトウェア開発支援

* 原稿受付 1992. 4. 10

** 川崎重工業 株式会社 技術開発本部 Kawasaki Heavy Industries, Ltd.; 1-1, Kawasaki-cho, Akashi 673, JAPAN

*** 大阪大学 大型計算機センター Osaka University; 2-1, Yamada-oka, Suita, Osaka 565, JAPAN

**** 大阪大学 工学部 Osaka University; ditto

Key Words: Petri net, sequence control, graphical programming, simulation, automated guided vehicle.

システム K-NET を開発した。本論文では、上述の無人搬送システムの制御ソフトウェアの開発に対して K-NET を適用し、有効な結果を得たので以下に報告する。

2. K-NET の概要

K-NET は、制御仕様をペトリネットで記述することにより制御ソフトウェアを開発するためのプログラミング環境である。K-NET には、ペトリネットで表現した制御仕様を入力・編集するためのエディタ、その内容を検証するためのシミュレータ、入力されたネットをコントローラで動作できる形式、すなわち C 言語記述のプログラムに変換するためのジェネレータ、およ

びペトリネットによる制御仕様を整理した形式で印刷するためのレポータが設けられている。また、コントローラに接続してその制御状況をリアルタイムに表示するモニタ機能がある。K-NET の機能構成を Fig. 1 に示す。

K-NET の特長は以下のとおりである。

- (1) ハイレベル・ペトリネットによる複雑な制御仕様の表現

K-NET で用いるペトリネットでは、トークンが属性を持つことができ、トークンの種類に応じてネット内の流れを変えることができる。さらに、事象の発生および処理時間の確率的扱いを可能としている。

Fig. 1 System configuration of K-NET

(2) IF-THEN ルールによる動作フロー条件の表現
従来、状態遷移の条件はネットの有向枝を用いて表現されていたが、K-NET ではこれを IF-THEN ルールにより表現することにより、動作フローのみがネット表現され、制御の流れが理解し易くなっている。

(3) ペトリネットと手続き型言語の融合

K-NETでは、シーケンス制御における同期、排他、並行動作を含む処理フローをベトリネットで、システムの状態変化に伴う煩雑な情報処理を従来の手続き言語Cで記述することにより、両者の長が組み合わせられて、制御の流れを理解し易い形で表現できるとともに、複雑な処理の記述も可能となっている。

3. K-NET における制御仕様の記述モデル

K-NETにおいて制御仕様を表現するために用いるペトリネットは、特定の機能を有する数種類のボックス、処理の流れを制御するゲート、両者間の関係を表わす有向枝、およびその有向枝に沿ってネット内を流れるトークンから構成され、さらに手続き型言語Cを用いて処理内容を記述できるユーザ定義関数、入出力を定義する

論理 I/O が用いられる (Fig. 2). 各要素の詳細について以下に説明する.

3.1 トークン

トークンはシステム内を流れる物や情報に対応する。トークンはたとえばワークのタイプ、形状などの属性情報を持つことができ、“クラス”と呼ばれる属性の構造体によって分類される。

3.2 ボックス

3.2.1 ボックスの状態

ボックスはトークンの存在の状況により、システムおよび各機器の状態を表現する。ボックスに収容可能なトークンの数をそのボックスの“容量”と呼び、通常、ボックスの容量は1である。ボックスの状態としては“受容状態”および“励起状態”が定義される。受容状態とは、トークンを受け入れることができる状態、すなわち容量と同じ数のトークンが存在しない状態である。また、励起状態とは、ゲートが発火してトークンの移動が可能な状態である。通常、ボックスはトークンが存在する場合に励起状態にあるが、後述の時間ボックス、処理ボックス、マクロボックスなどではトークンが存在するからと

いて、必ずしも励起状態にはなく、システムの状態によって変化する。

3.2.2 ボックスの種類

ボックスはその機能によって、以下のような種類を用意している (Table 1).

(1) 状態ボックス

容量1の単純なボックスである。これは、機能的に以下のボックスの完全な部分集合であり、それらを用いて表現することができるが、入力 of 簡便さおよびネットの見易さから導入した。機器間のインターロックを表現する際に、機器の稼働状態あるいは遊休状態に本ボックスを対応させる。

(2) 容量ボックス

1 以上の容量を持つボックスであり、バッファあるいは待行列を表現する。トークンの取り出し順序として、先入先出、先入後出、あるいはトークンの特定の属性値の昇順/降順を指定することができる。

Fig. 2 Net elements in K-NET

Table 1 Kinds of box

name	symbol	capacity	specifying items
state box		1	nothing
buffer box		n	capacity, token exiting order
time box		1	time t
count box		m	count value m
calculation box		1	operator, logical I/Os
signal box		1	logical I/Os, output values
process box		1	process, condition, post-process, time
source box		n	class, capacity, condition, process
macro box		-	nothing
entrance box		1	nothing
exit box		1	nothing

(3) 時間ボックス

時間の概念を導入したボックスである。トークンがこのボックスに入ってから指定時間経過後、ボックスは励起状態に移移する。時間は数値、論理 I/O の値、確率分布のいずれかで指定される。時間ボックスは、シーケンス制御におけるタイマを表現する。また、制御対象のモデルに使用することにより、シミュレーションによるサイクルタイム計算に利用される。

(4) 処理ボックス

通常、動作フローはネットにより表現されるが、定型的な動作フローを一つのボックスで表現することにより、全体ネットの作成が容易になり、かつネット構造の見通しが良くなる。そこで、制御において典型的な動作フロー“実行—監視—チェック”を表現する特別のボックスとして、処理ボックスを導入した。処理ボックスの動作をネットにより模式的に表現すると Fig. 3 のようになる。すなわち、処理ボックスにトークンが入ると、まず、

ユーザ定義関数 A が呼び出される。処理関数 A では、通常、外部プロセスに対する信号出力、タスク起動、その他の情報処理が実行される。その後、処理完了条件 B の成立を待ち、同時にタイムアウト監視のためのタイマ C を起動する。論理 I/O に関する論理式の積和形式で記述された完了条件 B が成立するか、あるいはタイムアウト発生時には、後処理関数 D が呼び出されるが、引き数によってその起動要因を知ることができる。後処理では、完了条件を調べることにより、タイマ C をキャンセルしてボックスを励起状態に移させるか、あるいは再び完了待ちの状態に戻るかを制御することができる。

(5) 発生ボックス

論理 I/O に関する論理式の積和形式で記述された条件が成立した時点で、指定したクラスのトークンを発生させる。トークンは 1 度だけ、あるいは条件の

Fig. 3 Function of process box

成立している間、決定的/確率的な時間間隔で発生させることができる。発生したトークンの属性値は、クラスで記述された初期値を設定されるが、引き続きユーザ定義関数が呼び出されてこれを変更することもできる。

(6) マクロボックス

より詳細なネットで置き換えられるボックスである。Fig. 4 に示すように、マクロボックスにトークンが入ると、対応する詳細ネットの入口ボックスにトークンが入り、以降、詳細ネット内をそのネット構造にしたがって移動する。そして、トークンが出口ボックスに入った瞬間、上位のマクロボックスは励起状態となる。マクロボックスは、システム全体の概略設計から始まって詳細化へ至るトップダウン設計を支援し、かつシステム全体の構造の見通しを良くする。

Fig. 4 Macro box and detailed net

(a) Net representation

Firing condition $(f1 \cdot f2) \vee f3$

Token flow condition

box A $\rightarrow$ box X, Y executing g

box B $\rightarrow$ terminate

generate $\rightarrow$ box Z

(b) Description format

Fig. 5 Gate specification

(7) その他のボックス

上述のボックスのほかに、シーケンス制御におけるカウンタを表現する計数ボックス、論理 I/O の値に関する算術演算を実行する演算ボックス、論理 I/O に信号を出力する信号ボックスがある。

3.3 ゲート

ゲートはトークンの流れを制御するものである。ゲートは、基本的には、その入力ボックスのすべてが励起状態にあり、出力ボックスのすべてが受容状態にある場合に発火し、すべての入力ボックスからトークンを一つ取り出し、すべての出力ボックスにトークンを一つ置く。複数のゲートの発火が競合する場合、固定的な優先順位によって発火するゲートが決定される。以上の基本機能に加え、Fig. 5 に示すような IF-THEN ルールの記述

により、以下に示す発火制御が可能である。

(1) 発火条件

ボックス内のトークンの有無、その属性値、論理 I/O の値等の論理式の積和形式による発火条件を IF 部に記述する。発火条件は複数個並記することができ、最初に成立した条件に対応する制御が実行される。

(2) トークンのフロー制御

上記発火条件を満足する IF 部に対応した THEN 部の内容に従って、トークンの移動および属性継承を含むフロー制御が実行される。この時、入力ボックスごとにユーザ定義関数の呼び出しを指定することができ、トークンの属性値変更のほか、トークン移動に伴う任意の処理を行うことができる。

3.4 ユーザ定義関数

手続き型言語 C で記述されたユーザ定義関数は、前述のように処理ボックス、発生ボックスあるいはゲートから呼び出される。ユーザ定義関数内では、論理 I/O やトークン属性値にアクセスするためのシステム関数を用いることができる。

3.5 論理 I/O

I/O 設計は、論理設計と物理設計の 2 段階で行われる。論理 I/O は、前述のように時間ボックスの時間、発生ボックス

の発生条件, 処理ボックスの完了条件, ゲートの発火条件などに使用される。外部プロセスへの信号入出力以外に, ネット要素間での信号の授受にも使用可能である。論理 I/O を物理 I/O に割り当てることにより, 実際の信号が外部プロセスに出力, あるいは外部プロセスから入力される。

4. K-NET の機能

4.1 エディタ

K-NET には, ペトリネットで制御仕様を記述するためのグラフィックエディタを設けている。編集機能として, ネットの挿入, 修正, 削除, 移動, 複写機能がある。さらに, 論理 I/O の編集機能, トークンの属性構造体を規定するクラスの編集機能が設けられている。

4.2 シミュレータ

前記編集機能で入力されたネットが設計どおりになっているか否かを検証するためにシミュレーション機能を設けている。シミュレーションを実行する前には, ネットのボックス上にトークンの初期配置を行う。実行には, 現時点で移動可能なトークンを 1 回だけ移動させる方法と, 連続的にトークンを移動させる方法が用意されてい

る。デバッグのために, シミュレーション途中の任意の時点で, 各ボックスにおけるトークンの配置/除去, 各論理 I/O の値の変更を行うことができる。また, ネット上のトークンの各属性の現在値, およびトークンのボックス滞在, ゲート通過, さらに論理 I/O のオン/オフに関するタイムチャートを得ることができる。シミュレーション時のネット画面を Fig.6 に示す。

以上のほかに, 確率モデルを利用したシミュレーションにより, FA システムの性能評価を行うこともできる。その結果として, トークンのボックス滞在時間, ゲート通過間隔に関する統計情報が得られる。

4.3 レポート

本機能は, ネット情報およびシミュレーション結果を整理した形で印刷・出力する。本機能により, 出力される項目は次のとおりである。

- ① ネット階層図および構成図
- ② ボックス, ゲート, 論理 I/O, トークンクラスの仕様および名前一覧表
- ③ 各論理 I/O を参照するボックス, ゲートの名前一覧表
- ④ タイムチャート図
- ⑤ ボックス, ゲート, 論理 I/O に関する統計情報

Fig. 6 Window for simulation

4.4 ジェネレータ

本機能は、入力・検証されたネットを現場で使用するコントローラに搭載するために、ネット情報をC言語記述のプログラムに変換する。

4.5 モニタ

本機能は、コントローラをK-NETシステムに接続した状態で動作し、実機試験を支援する。コントローラから送られて来る制御状況に関する情報を基にして、シミュレーション時と同じネット図におけるトークン移動として、コントローラが制御を実施する状況をリアルタイムに表示する。

5. K-NET による制御ソフトウェアの開発手順

K-NETを用いて制御ソフトウェアを開発する手順は一般に以下ようになる (Fig. 7)。

- ① エディタを用いて、制御ネットおよび外部プロセスの動作を表現する制御対象ネットを作成する。ここで、それぞれのネット間の情報の授受は、論理I/Oによって表現する。
- ② シミュレータを用いて、制御ネットの妥当性を検証する。

- ③ 制御ネットおよび制御対象ネットを分離した後、ジェネレータを用いてC言語プログラムに変換し、それぞれを制御用計算機および制御対象のシミュレーション用計算機上に実現する。この時、ネット間の情報授受のために記述した論理I/Oは、各計算機間を接続する物理I/Oに割り当てる。ここで、シミュレーション用計算機は制御対象の動作を模擬するリアルタイムシミュレータとして機能する。
- ④ リアルタイムシミュレータを用いて、コントローラのデバッグを行う。この時、コントローラの制御動作はモニタを用いて確認できる。
- ⑤ リアルタイムシミュレータを実際の制御対象に置き換え、実機を用いたコントローラのデバッグを行う。この時にも、コントローラの動作はモニタを用いて確認できる。
- ⑥ 以上のテストを終了した後に、レポートを用いて制御ネットの仕様書を作成する。

6. K-NET の無人搬送制御システムへの適用

6.1 無人搬送システム制御装置の開発⁴⁾

6.1.1 無人搬送制御システムの概要

対象とする無人搬送システムは、Fig. 8に示す走行経路と、自動倉庫ならびに48か所のステーションを有し、最大5台のAGVを投入できるように設計されている。各AGVは、ステーション停止中に、光通信装置を経由して無人搬送システム制御装置（以下、AGVS制御装置と呼ぶ）と情報の授受を行う。AGVS制御装置からは、走行、荷積み、荷卸しの指令ならびに状態の問い合わせが行われ、各AGVからは指令データ受取応答、動作完了情報、および状態情報が送信される。AGVは走行指令として、“直進”、“左折”、“右折”、“ステーション通過”あるいは“ステーション停止”等の動作指示の系列を通知され、走行経路の分岐点ならびにステーションにおいて順次この指示を実行しながら自律的に移動する。なお、各AGVの状態情報は、ステーション間を走行中にも誘導通信装置を経由して一定時間間隔でAGVS制御装置へ通知される。

AGVS制御装置は、生産システム全体を管理するFAコンピュータから通知

Fig. 7 Procedure for development of control software

Fig. 8 AGV control system

される搬送要求（荷積みステーションと荷卸しステーションの対）をもとに、光通信装置および誘導通信装置を経由して、AGVの運行制御と運行監視を行う。そのため、AGVS制御装置には、

- ① 各搬送要求に対し、最適なAGVを割り当てる台車割当機能
- ② 割り当てられた搬送要求に従って、AGVに対して荷積みステーションへの走行、荷積み、荷卸しステーションへの走行、荷卸しを順次実行させる荷物搬送機能
- ③ 搬送の完了した時点で、実行すべき搬送要求が存在しない場合、AGVを待機ステーションへ移動させる待機走行機能
- ④ バッテリ低電圧となったAGVを搬送終了後に充電ステーションへ移動させる充電走行機能

が設けられている。このほか、異常時の対応、マンマシン、および周辺機器との通信機能がある。

6.1.2 K-NETによる開発

上記AGVS制御装置のソフトウェア開発にK-NETを適用した。本装置の機能の中で、荷物搬送、待機走行、充電走行など、AGVのシーケンス的な動作フローをペトリネットで、台車割当などのAGV動作に伴う状態変化時の複雑な情報処理をC言語で記述した。一方、高速な処理が必要な周辺機器との通信機能、および画面表示を伴うマンマシン機能は、外部プロセスとして従来手法によるC言語タスクで実現した。なお、開発用計算機としては、事務所ではワークステーションSUN4を、現場では東芝のラップトップパソコンJ3100を用い、また制御用計算機としてはNECのFAパソコンFC9801を用いた。K-NETの利用によって、制御プログラムの設計、作成から現地試験まで含めた本AGVシステムの制御ソフトウェア開発は、従来の開発方式で要する費用の約1/3で達成できた。

6.1.3 ペトリネットモデル

AGV群の運行制御に対するネットの概念図をFig.9に示す。AGVの荷物搬送および待機の基本シーケンスは、各AGVごとにボックス1～5で表現される。FAコンピュータから搬送要求が通知されると、それに対応するトークンが

発生ボックス10に発生し、搬送要求バッファを表現する容量ボックス11に移る。AGV待機中に相当する状態ボックス5にAGVに対応するトークンが存在して、処理ボックス13あるいは14の台車割当処理により、ボックス11内の搬送要求トークンにAGV1が割り当てられると、ゲート12が発火し、ボックス1にトークンが移動して荷物搬送が始まる。荷積み完了後に、荷卸しステーションへの走行に対応するボックス3にトークンが入ると、トークンはボックス6に移動し、光通信装置を経由してAGVに動作指令を発信した後、走行の完了をボックス7で待つ。AGVから走行完了が通知されると、発生ボックス9にトークンが送生し、ゲート8が発火して、次の荷卸し処理を行うためにトークンはボックス4に移動する。

本AGVS制御装置を実現するために作成したネットの規模は、ボックス284、ゲート281、論理I/O63、トークンクラス5、ユーザ定義関数は82であった。

Fig. 9 Net model for AGV system control

Fig. 10 Net model of AGV and route

6.2 無人搬送システムシミュレータの開発⁵⁾

6.2.1 シミュレータの構成

前述の無人搬送システムのリアルタイムシミュレータを K-NET を用いて開発した。シミュレータは実機系のモデルとマンマシン部により構成される。マンマシン部は路線図や AGV の位置・状態の表示機能と、AGV の投入・削除、ならびにバッテリー低電圧や異常状態を強制的に発生させるためのキー入力を受け付ける機能を有している。AGV の動作を模擬する実機系のモデルは、

AGVS 制御装置もしくはマンマシン部からの指令に基づいて動作し、その結果を AGVS 制御装置とマンマシン部に通知している。

6.2.2 ペトリネットによるモデル

実機系のモデルとして、走行経路の構成やステーションの配置などの情報は、実経路に近い 2 次元的な図で表現できるペトリネットで、複雑な情報処理を伴うマンマシン部および AGVS 制御装置とのインターフェイスは、C 言語によるプログラムで実現した。

ステーション停止中の AGV は AGVS 制御装置と情報の授受を行い、動作開始処理を必要とするので、ステーションは C 言語によるユーザ定義関数を記述可能な処理ボックスで、ステーションや経路の分岐点および合流点の間の部分的な走

行経路は走行時間を表現するために時間ボックスで表現し、ステーションと部分的走行経路間の接続関係は、ユーザ定義関数が記述可能なゲートおよび有向枝を用いて表現している。たとえば、Fig. 10 (a) に示す走行経路のペトリネットは同図 (b) のようになる。また、AGV は動作指示データを属性として持つトークンによって表現される。

AGVS 制御装置から走行指令が通知されると、動作指示系列が既定のデータ領域に設定され、AGV トーク

ンの存在する処理ボックスにおいて、動作指示系列の先頭データがトークンの属性に設定される。以後、トークンの移動は、ボックスの出力側ゲートに記述されているトークンフロー条件によって制御される。そして、ゲートが発火するごとに、そこに記述されたユーザ定義関数により、トークンの位置が変更されたことをマンマシン部に通知すると同時に、トークンは次の動作指示データを設定され、ペトリネット上で移動を続ける。トークンがステーションに対応する処理ボックスに到達すると、AGVS 制御装置に動作完了を通知して次の指令を待つ。たとえば、Fig. 10 (a) におけるステーション S1 から S2 への走行指令では、動作指示系列（前進、直進、ステーション停止）が S1 停止中の AGV に与えられる。最初の動作指示データ“前進”を実行した後、その属性値が“直進”に変更された AGV トークンがボックス T1 に入る。つぎに、ボックス T1 の出力側ゲート G2 のトークンフロー条件は Fig. 10 (b) のように記述されており、最初の条件が選択されて、その THEN 部に記述されているユーザ定義関数 MOVE1 によりトークンの属性値が“ステーション停止”に変更され、ボックス T2 にトークンが移動する。

7. あとがき

本論文では、ペトリネットで制御仕様を記述する方式による FA 制御ソフトウェア開発支援システム K-NET を用いた、無人搬送システムの制御ソフトウェアの開発について述べた。ペトリネットにおいては、システムの挙動を事象の発生に伴う各要素の状況変化により分散的に表わし、したがって並列同時進行性、順序性、競合とその解消がシステム構成要素に併せて表現できる。また、システム仕様の各表現レベルに対応した階層的記述が可能である。さらに、ペトリネットは分かりやすい図的プログラミング言語であるため、シミュレータ、モニタなどのデバッグ環境の構築が容易である。ペトリネットをベースにした K-NET を用いることにより、上述のペトリネット表現の利点に加え、動作フローをペトリネット、情報処理を従来の手続き型言語 C で表現することができ、処理の流れとその内容記述の分離が可能となった。また、ゲートの発火条件を IF-THEN ルールで記述することにより、トークンの流れとその制御が分離され、ネットを明瞭に表現できるようになった。さらに、制御仕様の記述から、プログラム作成、デバッグまで一貫して支援できるため、信頼性の高い制御ソフトウェア開発が可能となった。無人搬送システムの制御ソフトウェアの開発に K-NET を適用した結果、ソフトウェアの設計、作成から現地試験まで含めた開発費用は、従来

の手続き型言語 C のみを用いて開発する場合に比較して約 1/3 に低減することができた。

参 考 文 献

- 1) K. Jensen : High Level Petri Nets : Application and Theory of Petri Nets ; Informatik Fachberichte 66, Springer Publishing (1983)
- 2) R. Valette et al. : Putting Petri Nets to Work for Controlling Flexible Manufacturing Systems ; Proc. IEEE Int. Symp. on Circuits and Systems, pp. 929~932 (1985)
- 3) T. Murata et al. : A Petri Net-Based Controller for Flexible and Maintainable Sequence Control and Its Application in Factory Automation ; IEEE Trans. Industrial Electronics, Vol. 33, No. 1, pp. 1~8 (1986)
- 4) 長尾, 山内, 占部, 海老根, 熊谷, 児玉 : 無人搬送車制御システムへのペトリネットの応用 ; 第 34 回システム制御情報学会研究発表講演会講演論文集, pp. 183~184 (1990)
- 5) 長尾, 山内, 太田, 占部, 熊谷, 児玉 : 無人搬送システムシミュレータへのペトリネットの適用 ; 第 29 回計測自動制御学会学術講演会予稿集, pp. 101~102 (1990)
- 6) Y. Nagao et al. : Petri-Net Based Programming System for FMS ; Proc. 16th Annual Conf. of IEEE Industrial Electronics Society, pp. 456~461 (1990)