“Matrix-Based Discrete-Event System Controller”を拡張するためのETSCから時間ペトリネットへの変換手法とその統一的支援システム

山場　久昭[a)]・北野　翔一郎[b)]・高塚　佳代子[c)]・片山　徹郎[d)]・岡崎　直宣[e)]・冨田　重幸[e)]

A Transformation Method from ETSC to Timed Petri Net to Improve Matrix-Based Discrete-Event System Controller and its Unified Support System

Hisaaki Yamaba, Shoichiro Kitano, Kayoko Takatsuka, Tetsuro Katayama,
Naonobu Okazaki and Shigeyuki Tomita

Abstract

A sophisticated matrix-based framework “Matrix-Based Discrete-Event System Controller (MDEC)” proposed by Jose Mireles et al is one of the promised method to design control systems of discrete manufacturing systems. In the previous work, we improved MDEC by introducing Timed Petri net and “Expanded Timed-State-Chart” that was developed in our laboratory to describe complex behavior of discrete manufacturing systems (MDEC2). In this work, a computer system supporting to design such control systems was implemented based on MDEC2. ETSC models which users draw through GUI are transformed into Timed Petri nets in the form of matrices; such matrices are embedded into controllers of MDEC2. Through a series of experiments, it was confirmed that obtained controllers could work regularly.

Keywords: Discrete-event system, Control system, Matrix-based, Timed Petri net, Computer support system

1　緒言

いくつかのリソースを複数のジョブの処理で共有している離散型生産システムでは、ジョブの投入順序などの制御を適切に行わないとブロッキングやデッドロックなどの重大な問題が生じてしまう。そのため、健全な制御系を設計して与える必要があるが、そのような設計手法の一つとして、ペトリネットを活用したMatrix-Based Discrete-Event System Controller [1]（以下MDEC）がある。さらに筆者等は、制御対象の挙動を表すモデルの設計や修正にとって便利な「拡張時間ステートチャート[2]（以下ETSC）」をMDECの挙動表現モデルとして採用する方法を提案している。さらに、MDECの状態管理モデルを従来のペトリネットから、時間を扱える時間ペトリネットにすることで、時間ペトリネットに基づいて得られた最適スケジュールをMDECの制御方法に反映できるような拡張も行っている[4]（MDEC2）。また、MDEC2ではETSCを数式処理が可能なペトリネットへ変換する必要があり、その方法の開発も同時に行った。ただし、ETSCから時間ペトリネットへの変換は人手で行われていた。

そこで本研究では、対象システムのETSCによるグラフィカルな記述・編集、作成されたETSCから時間ペトリネットへの自動変換、MDEC2に組み込むためのマトリクス形式で表現された時間ペトリネットのファイルへの出力、を統一的に行えるシステムの開発を行った。また、その過程でETSCから時間ペトリネットへの変換方法の整理も行った。さらに、開発した支援システムの自動変換機能が正しいものであることを簡単なETSCを使って確認した上で、ETSC記述から制御設計までが正しく行えていることをFA実験装置を対象に行った。

2　Matrix-Based Discrete-Event System Controller

MDECは、制御対象の挙動表現モデルとしてのペトリネットを状態管理モデルとしてもそのまま採用している。そして、そのペトリネットの中で次に発火すべきトランジションを、状態方程式 (図1 -(1)) を使って算出し、発火に伴うペトリネット上のマーキングの変化をマーキング遷移方程式を使い特定するという形で対象システムの動作を記述する。

図1にMDECの概観を示す。各記号の意味は以下の通り

[a)]情報システム工学科助教
[b)]情報システム工学科学部生
[c)]教育研究支援技術センター技術専門職員
[d)]情報システム工学科准教授
[e)]情報システム工学科教授

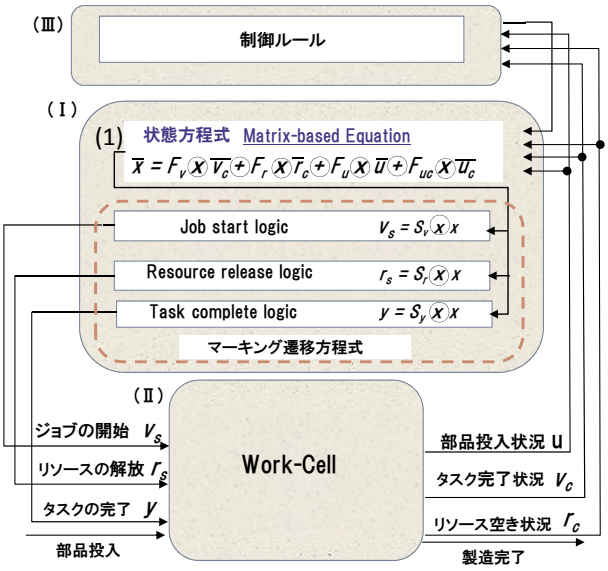

図 1: MDEC の概観

である

$\overline{\boldsymbol{x}}(t)$：時刻 t でのタスクの状態ベクトル

$\boldsymbol{v}$：ジョブ集合のベクトル

$\boldsymbol{r}$：タスクを実行するリソース集合のベクトル

$\boldsymbol{u}$：MDEC への入力集合のベクトル

$\boldsymbol{u_c}$：競合解消のための制御ベクトル

$\boldsymbol{y}$：システムからの出力集合のベクトル

F_v (S_v)：ジョブ先行関係行列

F_r (S_r)：要求リソース行列

F_u (S_u)：外部入力行列

S_y：完了タスク行列

F_{uc}：競合解消行列

状態方程式とマーキング遷移方程式はともに論理代数（⊗:論理積、⊕:論理和、$\overline{\boldsymbol{x}}$:$\boldsymbol{x}$の否定）で記述される。

状態方程式ではジョブ先行関係の充足可能性（$F_v \otimes \overline{\boldsymbol{v}}$）と要求リソースの充足可能性（$F_r \otimes \overline{\boldsymbol{r}}$）、外部入力条件（$F_u \otimes \overline{\boldsymbol{u}}$）、制御入力の値の真偽（$F_{uc} \otimes \overline{\boldsymbol{u_c}}$）の論理和で各トランジションの発火可否を表すベクトルが得られる。$\boldsymbol{u_c}$は競合解消の制御ルールとして外部から与えられる。

マーキング遷移方程式は状態方程式で得られたベクトル$\boldsymbol{x}$と各行列を使ってマーキングの変化を算出する。

3 ETSC と最適スケジュールを組み込むための MDEC の拡張

図 2 は従来の MDEC を拡張した MDEC2 の処理の流れを示している。本節では、まず、本研究で扱う ETSC について説明をし、次に、上記のような MDEC2 で拡張されたいくつかの点について順に説明する。

3.1 拡張時間ステートチャート

離散型生産システムの挙動表現モデルとして本研究室で開発された ETSC（図 3）は、状態間の階層性・並行性と、通信のブロードキャスト性、実時間性、生成イベントの記述能力

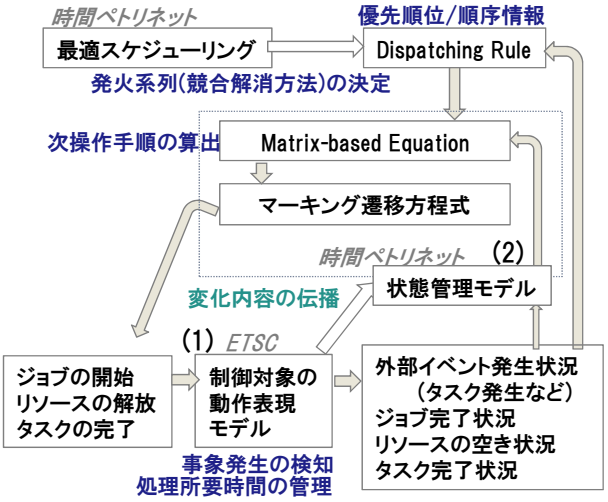

図 2: MDEC2 の枠組み

を持つ。

さらに、人間が理解しやすい、並列挙動の階層的表現能力を保存したまま、事象駆動と時間駆動が混在する複雑なシステムの、挙動を記述できる、表現力を持っている。図 4 の破線で仕切られた四角の枠（ステート）は状態 A,B,C が AND 並列の関係にあることを示しており、仕切られた枠の内部にあるステートは、それぞれが OR 並列であることを示している。ステートの上部にみられる矢印は初期状態を示している。

ETSC の状態遷移や通信はイベントで行われており、これはイベント生成則によって表されている。イベント生成則とは、イベントの生成・発火タイミングに関する記述方法を定めた規則である。破線アークでイベントの生成を、アークに付随する自然数の組でその発火タイミング制約を表す（生成イベント）。

なお、1 つの状態遷移の生起に伴って、複数のイベントが生成される場合は、そのイベント間の関係が AND 型、あるいは、XOR 型かに応じて扱いが変化する。AND 型の場合は、その関係にある全てのイベントが生成され、各々の発火タイミング条件に従って発火する。XOR 型の場合は、その関係にあるイベントのいずれか 1 つだけが生成され、当該イベントの発火タイミング条件に従って発火し、それ以外のイベントは生成されない。

生成されたイベントは、対象システム内で生成された全てのイベントを、各々が発火、あるいは、消滅するまで格納しておくイベントプールにより管理される。このイベントの管理を実現するために、イベントプール・マネージャが導入されている。これは、イベントプール・マネージャ上の各イベントに対して、次のような処理を同時並行的に繰り返す。

1. イベント生成則に従って生成された、全てのイベントを、イベントプールへ格納する
2. イベントプールに格納されたイベントのうち、発火可能条件が充足された全てのイベントを、システム全体へブロードキャストする
3. 発火可能な全てのイベントを発火する
4. 発火した全てのイベントをイベントプールから抹消する

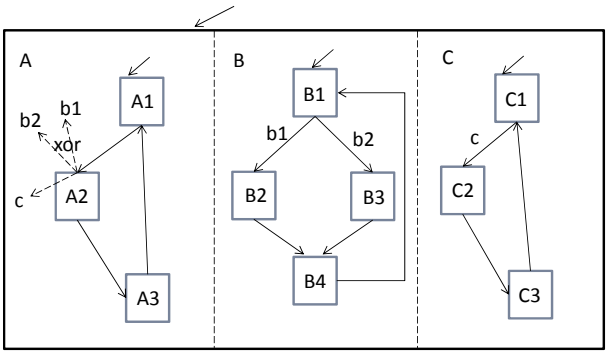

図 3: ETSC の例

5. 発火可能期間が終了した全てのイベントを抹消する

しかし、ETSC はその表現力の柔軟さゆえに、ETSC で記述されたモデルには個人差が生じてしまう。そこで、本研究室では、ETSC の自由度を一部制限し、階層の深さを高々二階層に抑えた記述方法を提案しており [3]、本研究でも、これを採用する。

3.2　制御対象の挙動表現モデルとしての ETSC の導入

MDEC は、制御対象の挙動表現モデルとしては人間にとって理解しやすいという理由から、状態管理モデルとしては、状態操作の計算に都合がよいという理由から、ともにペトリネットを採用している。しかし、本研究では ETSC は先に述べたように階層性を表現できることから、ペトリネットに対して優位性を持つ。ETSC は状態管理モデルとして採用された時にペトリネットとの間の整合性も取り易い。そこで、MDEC2 では、挙動表現モデルとして ETSC を採用した（図 2-(1)）。

3.3　制御対象の状態管理モデルの時間ペトリネットへの拡張

MDEC2 では、従来の状態管理モデルであるペトリネットを時間が扱える時間ペトリネットに拡張する（図 2-(2)）。その理由は、外部から与られる最適スケジュールから、制御ルールを生成する際に、時間ペトリネットが活用できるという利点があるからである。

3.4　ETSC から時間ペトリネットへの変換

既に述べたように、MDEC2 では、制御対象の挙動表現モデルである ETSC を、時間ペトリネットへ変換する必要がある。以前の研究ではこの変換は人手で行われており、本研究では、この変換の自動化を行うが、その過程でいくつかの不備が見つかったので。変換の方法とその手順を整理し直した。

3.4.1　変換方法

ETSC と時間ペトリネットが共通に持つ要素として、状態とその遷移、並列性が挙げられる。この共通の要素のうち状態の変換は 1 対 1 の関係で行う。具体的には、「状態 A から状態 B への遷移」は「トランジションの発火でトークンを消費するプレース A とその発火によりトークンを得るプレース B」へと変換する。このとき、ETSC のもつ OR 並列の表現も時間ペトリネットに変換されたことになる。

次に、ETSC には階層性の表現が存在するが、時間ペトリネットにはそのような表現が存在しない。しかし、本研究で扱う ETSC の階層は二階層に限定している [3]。そのため、第一階層で表現される AND 並列を、第二階層の初期状態をプレースに変換する際に、トークンを一つ持たせて表現すれば、階層性の変換は完了する。

また、ETSC にはイベントの生成のタイミングとその発火可能期間を示した「生成イベント」の記述が存在する。この生成イベントの変換は、次の 3 ステップで行われる

1. 生成されるイベントに対応したプレースを生成する。
2. 遷移時に生成されうるイベントの組合せの数だけトランジションを生成し、そのトランジションから各プレースに向かうアークを引く。
3. 生成されるイベントを条件に持つ遷移に対応したトランジションへ向かうアークをそのイベントに対応したプレースから引く。

最後に、イベントの生成を行わない遷移に関しては、状態と同様に、1 対 1 の関係で変換するだけでよく、1 つの遷移に対して、1 つのトランジションを生成すればよい。

3.4.2　変換の手順

ここでは変換の手順を示す。以下では、ETSC 内の破線で区切られた AND 並列性を表現するステートを AND ステート、1 つの AND ステート内に存在する OR 並列性を表現するステートを OR ステート、状態の遷移を表す矢印を ETSC トランジション、時間ペトリネット内のトランジションを Petri net トランジションと呼ぶ。

1. 1 つの OR ステートに対して 1 つのプレースを生成する。全ての AND ステートの持つ OR ステートについてこれを行う。また OR ステートが初期状態であるなら、生成されたプレースには 1 つのトークンを持たせる。
2. 1 つの ETSC トランジションが生成イベントを持っていないなら 3 へ。持っているなら 4 へ。
3. ETSC トランジションに対して 1 つの Petri net トランジションを生成し 7 へ。
4. ETSC トランジションのもつ全ての生成イベントの生成するイベントについて、各々に対応したプレースを生成する（イベントプレースとする）。
5. この ETSC トランジションが発火した際に生成されうるすべてのイベントの各組み合わせについて 1 つの Petri net トランジションを生成する。
6. 1 つの組合せのイベントに対応したイベントプレースに向かうアークを、1 つの組み合わせについて生成された Petri net トランジションから引き 7 へ。
7. 元の ETSC トランジションと接続関係にある、OR ステートに対応したプレースと、6 で生成された Petri net トランジションを、元の接続関係を保持する形でアークを引く。全ての ETSC トランジションの変換が完了したなら 8 へ。そうでないなら 2 に戻る。

8. 全てのトランジションの変換が完了したら、生成された各イベントプレースが出力するイベントを発火条件に持った Petri net トランジションに向かうアークをイベントプレースから引く。

3.5 状態方程式の拡張

従来の MDEC の状態方程式には、イベントに対応したプレースに関する項が存在しない。そこで MDEC2 では、状態方程式に、「イベント発火の充足可能性 $F_e \otimes \overline{\boldsymbol{u_e}}$ 」の項が追加されている。F_e はイベントの接続行列であり、$\overline{\boldsymbol{u_e}}$ はイベントの集合のベクトルである。

3.6 MDEC2 の処理の流れ

ここでは、図 2 に示した MDEC2 の処理の流れについて説明する。

まず、挙動モデルである ETSC の持つ情報を、状態管理モデルである時間ペトリネットに反映させ、状態方程式を使って、次に発火させるべきトランジションを算出させる。そして、トランジションの発火によるマーキングの遷移状況を、マーキング遷移方程式で算出し、時間ペトリネット上のマーキングを更新する。最後に、システムの仮想時間を進め、時間の経過による変化、即ち、ジョブの完了やそれに伴うリソースの解放、及び、外部イベントの発生等を、ETSC に反映させる。以下、同様の流れを繰り返す。

4 開発した支援システム

本研究で開発した支援システムの主な機能について説明する。本支援システムは、ETSC をグラフィカルに記述、編集することができる。また、ファイルへの入出力機能があり、記述した ETSC を保存、再編集することが可能である。そして、出来上がった ETSC を時間ペトリネットへ自動変換することができ、その時間ペトリネットは、本研究室が実装した MDEC2 へ入力するためのマトリクス形式のデータとして出力することができる。しかし、MDEC2 へ入力するための時間ペトリネットには、対象システムとは別に動作する、処理対象を入力する外部入力と、処理が完了した対象を出力する外部出力に対応した要素が必要になる。そこで本支援システムには、ETSC を編集する段階で外部入出力の設定を行うことで、時間ペトリネットへの変換時に、自動で各々に対応した要素を追加する機能も実装した。

図 4 は図 3 の ETSC の例を実際に本支援システムを使って記述したものである。さらに、図 5 は図 4 を本システムを使って時間ペトリネットに変換したものである。この変換は、図 3 の ETSC の本来持っている情報とは別に、状態 A1 から状態 A2 への遷移に対して外部入力を、状態 C2 から状態 C3 への遷移に対して外部出力を図 4 の ETSC に設定してから行われている。

以下では、まず、この外部入出力の時間ペトリネットの記述方法について説明する。次に、本システムを構成する主なクラスについて説明する。

4.1 外部入出力の時間ペトリネットの記述方法

開発したは支援システムでは、ETSC を時間ペトリネットに変換する際に、外部入出力の自動生成を行う機能を持っている。ここでは、外部入出力の時間ペトリネットでの記述方法について説明する。

まず外部入力について説明する。本研究では、外部入力は「あるトランジションを発火するための特殊な条件」と考える。具体的には、外部入力を接続させたいトランジション（T）に、トランジションの発火の条件のイベントを表すプレース（E）と、処理対象の発生を表すプレース（Poccur）を、T1 に向かうアークで接続する。また、外部入力は、処理対象の発生待ち状態と、処理対象の発生状態の 2 つ状態を持つと考えられる。そこで待ち状態を表すプレース（Pwait）と、この Pwait がトークンを持つとき発火可能になるトランジション（T2）を新たに追加し、T2 から e と Poccur に向かうアークで各々を接続する。このとき、T1 から Pwait へ向かうアークで各々を接続する。さらに、T2 を発火させる条件のイベント（生産要求）を表すプレース（Ereq）を追加し、Ereq から T2 に向かうアークで接続する、発火時に Ereq を生成するトランジション（T3）を追加し、アークで接続することで、あるリソースに対して、処理対象を投入する外部入力を表すペトリネットを記述できたことになる。本研究で開発した支援システムでは、T2、E は外部入力を接続させるトランジションごとに生成され、それ以外のプレース、トランジションについては、処理する対象ごとに生成ものとしている。

次に外部出力について説明する。外部出力については、あるトランジションが発火した時にトークンを持つプレース、を記述するだけで表現できたことになる。つまり、あるトランジションから、外部出力のプレースに向かうアークを引くことで、外部出力の記述は完了する。

4.2 主なクラス

ここでは本システムを構成する主要なクラスについて説明する。

4.2.1 インターフェース部分

- ExtendedTimedStateChartPanel
  基本的な拡張時間ステートチャートの記述をグラフィカルに行うためのインターフェース。ETSC の各オブジェクトを生成するために必要な Point はこの Panel の MouseEvent から得る。また ETSC が記述する各情報は右クリック時に表示されるポップアップメニューを利用する。
- ExtendedTimedStateChartTree
  ETSC の階層構造をわかりやすくするための Tree。生成イベントの付加は Panel からだとやりにくいので、XOR 関係の生成イベントの追加と削除はこの Tree のポップアップメニューからのみ行えるものとする。生成イベント以外の新たなオブジェクトの生成は Tree からは行えない。それ以外の各情報は基本的に Panel と同様に行える。
- ExtendedTimedStateChartFrame
  ETSC をグラフィカルに記述するための Panel と Tree を表示するための Frame。ツールバーから Panel の振る舞いを切り替える。またメニューバーからは editor に対して、時間ペトリネットへの変換命令、ファイル処理命令を行える。

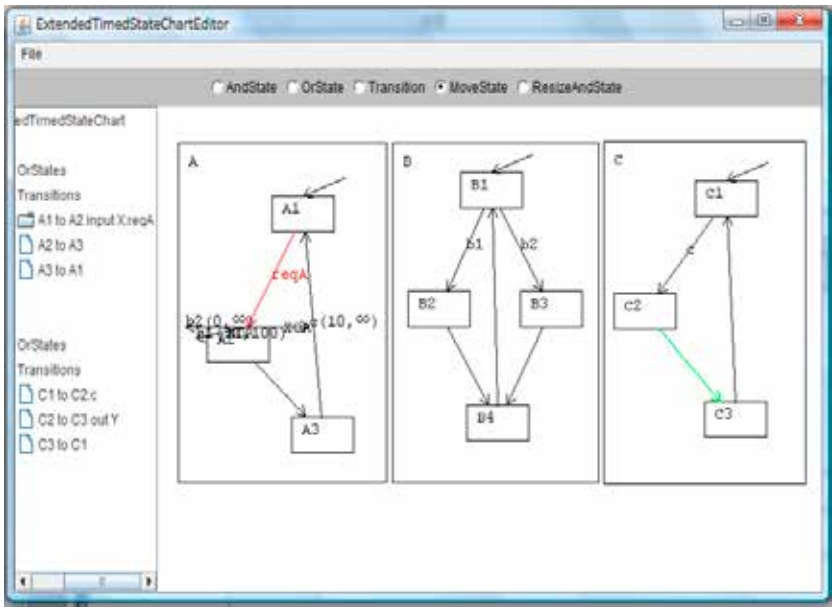

図 4: 本システムを使った ETSC の記述例

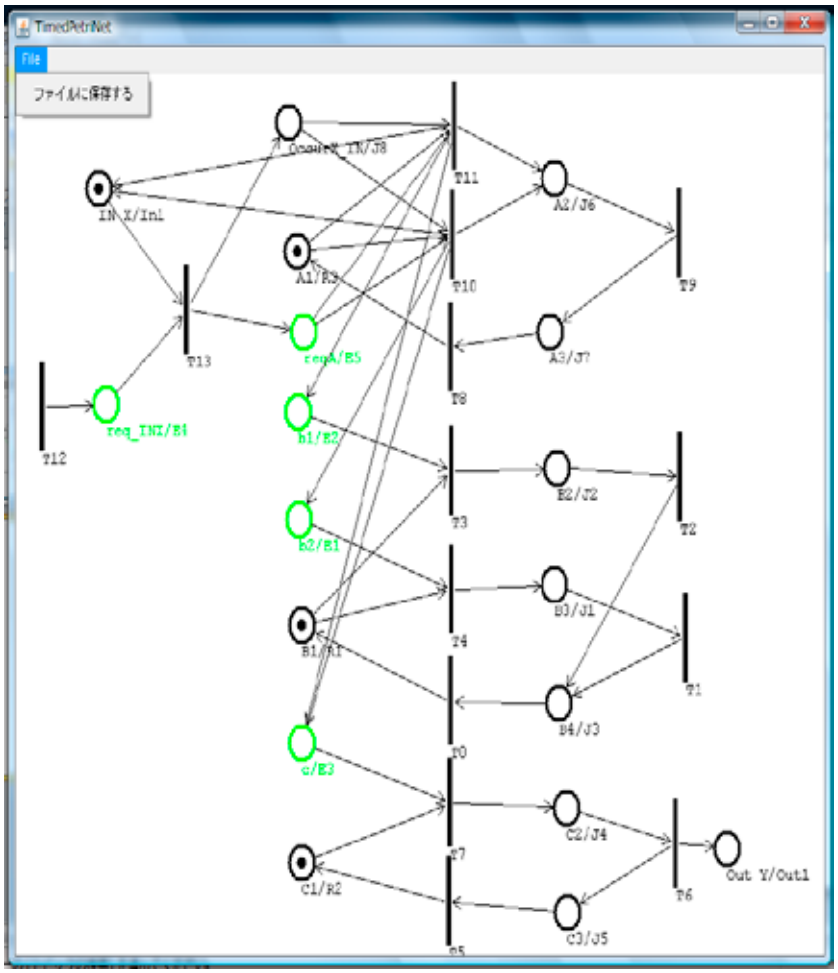

図 5: ETSC を時間ペトリネットへ変換した例

- TimedPetriNetPanel
  時間ペトリネットの各オブジェクトの位置の操作と、時間ペトリネットの描画を行うユーザインターフェース。
- ExtendedTimedStateChartFrame
  メニューバーからのファイルの書き込みの命令とパネルの表示を行う。

4.2.2 モデル

- ExtendedTimedStateChart
  拡張時間ステートチャートそのもの。このクラスは AndState のみを保持しておく。AndState に関する操作と自分が時間ペトリネットに変換できるかの判断を行う。基本的な Point による各オブジェクトの操作等は全て ETSC エディタが管理している。
- TimedPetriNet
  ETSC の各オブジェクトを時間ペトリネットのオブジェクトに変換したオブジェクトをもたせるクラス。基本的にはオブジェクトの保持と描画のみを行う。

4.2.3 モデルの要素

- AndState
  ETSC の階層性を表現するためのクラス。AndState 同士は AND 並列の関係になる。また OrState は全てこのクラスの領域内にしか生成することはできない。今回対象とするシステムの階層の大きさは高々 2 であるため、この AndState のなかにさらに AND 並列を持つことはない。
- OrState
  AND 並列のステート内の OR 並列の関係を表現するクラス。同じ AndState に参照されている OrState 同士は OR 並列の関係となっている。このクラスのオブジェクトは必ず AndState 内に存在している必要がある。
- ArcTransition
  トランジション（遷移）を表現するためのクラス。イベント、生成イベント、始点となる OrState、終点となる OrState を保持している。
- CreatedEvent
  生成イベントを表現するためのクラスであり、発火可能と不能になる時間と生成するイベントを使って生成される。
- Place
  プレースを表現するためのクラスであり、今回プレースの役割ごとにファイルを作る必要があるので、役割を変えるためのフラグ変数を持っている。
- TimedPlace
  イベントを表すプレースであり、通常のプレースを拡張して時間を扱えるようにしている。
- NodeTransition
  時間ペトリネットのトランジションを表現するためのクラス。発火の条件となるイベントを保持している。
- Arc
  時間ペトリネットの各ノードの接続関係を表現するためのクラス。

4.2.4 モデルを操作するエディタ

- ExtendedTimedStateChartEditor
  本来 ETSC が行うべきでないようなオブジェクトの操作を行うエディタ。このシステムの一番重要なクラス。ETSCPanel からの操作、ETSCPanel への再描画と警告の要求、ETSCTree への更新の命令と ETSCTree からの更新の通知の受け取り、ETSC への命令、ファイル処理などを主に行う。
- TimedPetriNetEditor
  時間ペトリネットのオブジェクトの操作を行う。オブジェクトの位置の変更とマトリクス形式のファイルへの変換を行う。

4.2.5 変換機能

- Transformer
  ETSC の各オブジェクトを時間ペトリネットのオブジェクトに変換するクラス。

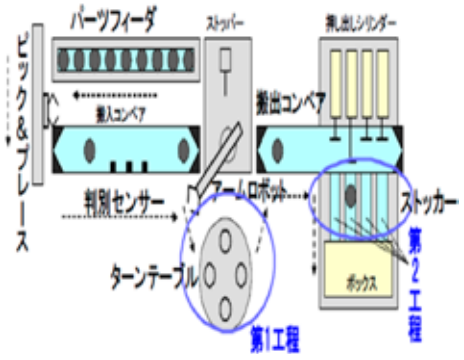

図 6: FA 実験装置

5 実験

本研究で開発した支援システムの各機能の動作実験を行った。

5.1 自動変換機能の動作実験

簡単な ETSC を例に、ETSC から時間ペトリネットへの変換を、人手で行った場合と本支援システムを使った場合とで比較し、本支援システムの変換機能が正しく動作しているかを‘確認した。なお、実験に用いた ETSC は人手で変換を行ってもミスが起こらない規模の物を、2 通り用意して行った。

実験の結果、人手で変換を行ったものと本支援システムで変換を行ったもの、ともにプレース数、トランジション数、イベント数が一致しており、その接続関係も同一のものに変換された。以上のことから、本支援システムの変換機能が正しく動作していることを確認した。

5.2 FA 実験装置を例にとした本システム全体の動作実験

開発した支援システムでの ETSC の記述から制御設計までの動作が正しいものであるかを検証するため、図 6 に示すような FA 実験装置を対象とした検証実験を行った。なお、本装置では、「搬入コンベアからターンテーブルへ」と、「ターンテーブルから搬出コンベアへ」の双方のワーク移送で、同一のアームロボットを使用している。そのため、適切な制御を組み込まなければ、「アームロボットが搬入コンベアからの運ばれてきた対象をつかんだが、要求を受けたターンテーブルが使用中である」といったようなデッドロックが発生する。

また、製造する製品は 2 種類とし、それぞれの製品を加工できる装置は決まっている。

実験の方法は、FA 実験装置の挙動を、本支援システムで ETSC として記述し、時間ペトリネットへ変換、それをマトリクス形式で出力し、以前の研究で本研究室が実装した MDEC2 へ入力しシミュレーションを行う。今回は制御ルールとして、「スケジューリング結果から得られる装置割り当てと順序情報に従うケース」と、「予め定めた優先順位に従うケース」の 2 通りのケースについて検討した。

実験の結果、制御ルールにスケジュールを用いた場合、問題が起きることなくプロセスを終了した。制御ルールに優先度を用いた場合、適切な変換ルールで変換された時間ペトリネットは、ターンテーブルの使用状況に関係なく、製品の加工要求を行う。そのため、同一の処理対象を、1 つのターンテーブルで連続で処理させようとするため、デッドロックが引き起こされる。以上より、制御ルールが適切でなければ、デッドロックのような重大な問題が引き起こされる、という現実の問題がシミュレートできていることがわかる。よって本支援システムによる ETSC 記述から制御設計までの動作は正確なものであることを確認した。

6　結言

本研究室で開発した、設計と実行にとって便利な制御動作実現方法「MDEC2」において、以前までは人手で行われていた ETSC から時間ペトリネットへの変換を自動で行い、時間ペトリネットを MDEC2 に組み込むためのマトリクス形式のデータとして出力する支援システムを開発した。また、簡単な ETSC を例に自動変換機能の動作実験を行いその正しさを確認した。更に、ETSC 記述から制御設計までの FA 実験装置を対象とした動作実験を行い、その動作は正確であることを確認した。

参考文献

1) J. Mireles, Jr., F. L. Lewis: Intelligent Material Handling: Development and Implementation of a Matrix-Based Discrete-Event Controller, IEEE Transaction on Industrial Electronics, Vol.48, No.6, pp.1087-197, 2001.

2) K.Takatsuka, S.Tomita: Modelling of Discrete Manufacturing Systems having multiple jobs for Verification by Model-Checking, Proceedings of 2010 IEEE International Conference on Industrial Informatics (INDIN’10), pp.1136-1141, 2010.

3) K.Takatsuka, S.Tomita: On Model-based Approach for Developing Control Systems of Event-Driven Manufacturing Systems, Proceedings of 2006 IEEE International Conference on Industrial Informatics (INDIN’06), pp.699-706, 2006.

4) K.Takastuka, Y.Sakai, H.Yamaba, S.Tomita: “Matrix-Based Discrete-Event System Controller” の ETSC 挙動モデルを持つ対象への適用のための拡張, 計測自動制御学会第 51 回離散事象システム研究会, pp.77-82, 2012.