TM-1241

高レベルペトリネットに基づく並行プログラミング環境 MENDELS ZONE の評価

内平　直志、本位田　真一（東芝）

December, 1992

© 1992, ICOT

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

高レベルペトリネットに基づく並行プログラミング環境 MENDELS ZONE の評価

MENDELS ZONE: Petri-net-based Programming Environment for Cooperating Discrete Event Systems

内平 直志† 本位田 真一†
Naoshi UCHIHIRA Shinichi HONIDEN

†(株) 東芝 研究開発センター システム・ソフトウェア生産技術研究所
Systems & Software Engineering Lab., R & D Center, TOSHIBA Corporation

概 要

高レベルペトリネットに基づく並行プログラミング環境 MENDELS ZONE について述べる．MENDELS ZONE では，プログラマは高レベルペトリネットである MENDEL ネットを用いて並行プログラムを構築する．MENDEL ネットは，従来の DESIGN/CPN などで導入されている「サブネット指向」の階層性とは異なり，「プロセス指向」の階層性を持つ．MENDEL ネットは並列プログラミング言語 KL1 にコンパイルされ，並列マシン Multi-PSI 上で実行される．MENDELS ZONE は，時相論理で記述された仕様を満たすように MENDEL ネットを階層的に調整する，などの様々な開発支援機能を提供する．本稿では，リフト制御システムの記述実験を通じて，リアクティブ・システムの開発環境である STATEMATE と比較することにより，MENDELS ZONE を評価する．

1 はじめに

近年，制御システムや通信システムなどのリアクティブ・システムの大規模化，複雑化に伴い，リアクティブ・システムを対象とした CASE が活発に研究されている．これらの多くは，有限状態機械，CCS，ペトリネットなどの状態遷移ベースのモデルに基づいている．我々は，並行プログラミング環境 MENDELS ZONE を開発してきたが，本稿では，並行プログラムを記述するための高レベルペトリネットおよびその開発支援環境とその評価について述べる．

従来から，多くの高レベルペトリネット（色付きペトリネット [1]，述語／遷移ネット [2]，代数的ペトリネット [3] など）が提案されている．また，高レベルペトリネットに基づくペトリネットツールも報告されている [4]．特に，DESIGN/CPN [1] は完成度の高いツールとして評価が高い．これらのツールでは，大規模なシステムを記述するためにペトリネットに階層構造を導入している [5]．しかし，この階層構造は部分ネットを 1 つのノードに縮約して表示するもの（サブネット指向の階層）であり，CCS や CSP におけるプロセスを階層の単位とするコンポジショナルな階層構造（プロセス指向の階層）とは異なる．複数の並行に動作するプロセスの同期関係を表現するには DESIGN/CPN の階層性は適当でない．すなわち，プロセス間のハンドシェイク型の同期を表現する手段がない．そこで，本稿では，並行プロセスを階層単位とするために，ペトリネットに CCS のコンポジショナルな枠組みと同期機構を導入した MENDEL ネットを提案する．

また，MENDELS ZONE は，MENDEL ネットの段階的設計支援機能，時相論理で記述されたタイミングに関する仕様を満たすように MENDEL ネットを階層的に調整する機能，などの従来のペトリネットツールにない様々な開発支援機能を提供し

SICE 第 10 回離散事象システム研究会．
† 〒 210 川崎市幸区柳町 70, TEL: 044-548-5474, FAX: 044-533-3593, E-MAIL: uchi@ssel.toshiba.co.jp

ている．さらに，作成されたMENDELネットは並列プログラミング言語KL1にコンパイルされ，並列マシンMulti-PSI上で実行される点も大きな特徴である．

2 MENDELネット

MENDELネットは，標準的なペトリネットに以下の特徴：(1)プレースの分類，(2)トランジションの発火条件および動作のＫＬ１による詳細記述，(3)トランジションの発火の優先度，(4)階層（モジュール）性，(5)同期／非同期通信機構，(6)配列表現，を付加したものである．

上記の追加機能により，MENDELネットの記述能力は十分に高く，実用レベルの制御／通信システムの記述が可能である（例えば，発電所制御システムの記述実験を行なった[6]）．同時に，プログラムのスケルトンがペトリネットとして抽出でき，その情報をプログラム検証／調整に活用できる．

2.1 プレース

MENDELネットでは，並行プログラムの観点からプレースを３種類に分ける（図１）：

状態要素：プロセスの状態を表わすプレース．トークン数は１（オン：プロセスがその状態にある）か０（オフ：プロセスがその状態にない）である．図式表現では円で表わされる．

スロット：メモリ上に格納されたデータを表わすプレース．トークン数は常に１である．例えば，データベースはスロットで表わされる．図式表現では，構造化分析手法のデータフローダイアグラムのデータストアと同様に並行な２本棒で表わされる．

ポート：データフローや非同期通信のための無限長バッファを表わすプレース．トークン数は０以上．図式表現では楕円で表わされる．

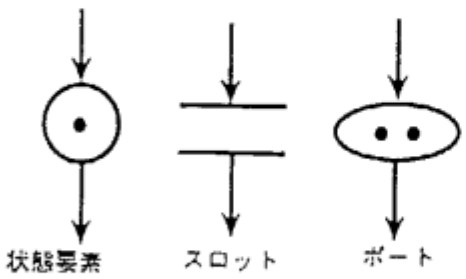

図１　３種類のプレース（状態要素，スロット，ポート）

2.2 トランジション

MENDELネットではトランジションを「メソッド」と呼ぶ．メソッドの詳細な発火条件と動作は，並列論理型言語KL1[8]で記述される．すなわち，トークンはKL1の変数を除く項（ここでは，値と呼ぶ）であり，アトム，数，リストなどのKL1で許される様々なデータを保持できる．メソッドは図式表現では長方形で表わされる．テキストでは以下の形式をとる：

method(<メソッド名>,<交換項>,
<入力状態要素リスト>,<出力状態要素リスト>,
<入力スロットリスト>,<出力スロットリスト>,
<入力ポートリスト>,<出力ポートリスト>,<優先度>) :－
<ＫＬ１ガード部> | <ＫＬ１ボディ部>;

図式表現では各状態要素，スロット，ポートとメソッドはデータフロー（アロー"→"）で接続される．これは，データフロー型の同期を実現している．また，メソッド間には「同期制約」を設定することができる．同期制約は，ハンドシェイク型の同期を実現するものである．図式表現では，同期制約はメソッドを点線"…"で接続することで表現される．

メソッドの発火条件と動作は次のように定義される．発火条件が真のときその動作が実行される：

【発火条件】● 入力状態要素リストの各状態要素がオンである．● 入力スロットリストの各スロットから読み出された値が，メソッドに記述されたスロット項と単一化できる．項が変数の場合は，その変数に値が代入され，その変数はＫＬ１ガード部およびボディ部で参照される．● 入力ポートリストの各ポートにトークンが存在し，トークンの値がポート項と単一化できる．● スロット項およびポート項に単一化された値に対して，ＫＬ１のガード部が成り立つ．● 同期制約で接続されたすべてのメソッドが発火可能であり，接続されたメソッドの交換項が単一化可能である．● 競合するメソッドが複数存在する場合は優先度が高い (express) メソッドを低い (normal) メソッドより優先する．

【動作】● ＫＬ１のボディ部を実行する．● 入力状態要素リストの各状態要素をオフにし，出力状態要素リストの各状態要素をオンにする．● 出力スロットリストの各スロットにボディ部で計算された値を書き込む．● 入力ポートリストの各ポートからトークンを取り出し，出力ポートリストの各ポートにボディ部で計算された値を持つトークンを追加する．

メソッドの動きを例（図２）を用いて説明する．

例：　method(move, _, [ready], [busy], [x(N1),y(M1)], [x(N2),y(M2)], [type(T)], [ack(A)], normal) :-

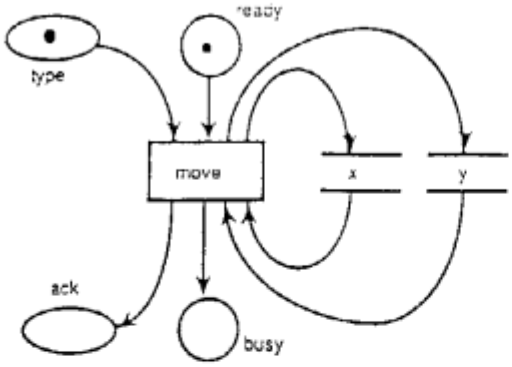

図2　メソッドの例

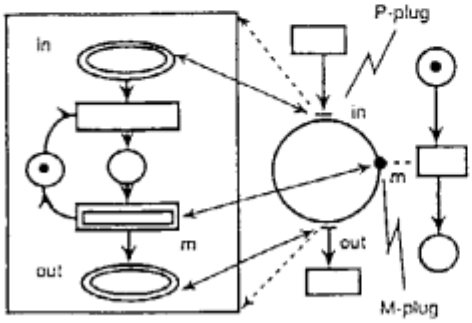

図3　プロセス指向の階層性

$N1 > 0, M1 > 0, T = job(ID) \mid N2 := N1 + 1, M2 := M1 + 1, A = ok(ID)$;

メソッド move は，状態要素 ready がオンであり，スロット x および y の値 N1 および M1 がそれぞれ 0 より大きく，ポート type のトークンの値 T が job(ID) と単一化可能であり，このメソッドより優先度が高くて発火可能なものがないならば，発火可能である．さらに，メソッド move が発火すると，状態要素 ready をオフにするとともに busy をオンにし，スロット x および y に N1+1 および M1+1 を代入し，ポート type からトークンを取り出す (pop) とともに ack に値が ok(ID) であるトークンを追加する (push)．

2.3　階層性

MENDEL ネットの階層（モジュール）単位はプロセスである．プロセスはいくつかのプロセス（サブプロセス）から構成することができる．プロセスとサブプロセスあるいはサブプロセス間の接続は，「外部ポート」と「外部メソッド」によるプロセス間通信としてモデル化される．このプロセス指向の階層性の例を図 3に示す．

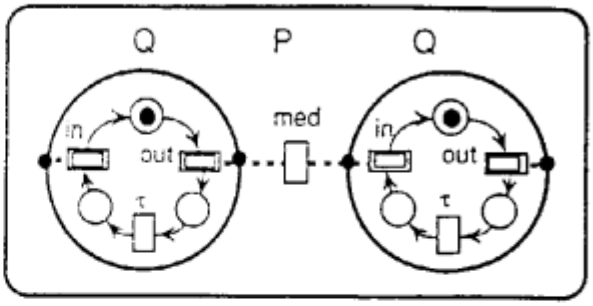

図4　CCS と MENDEL ネットの階層性

ポートはプレースの 1 つであり無限長バッファである．上位階層のプロセスからアクセスできるポートを「外部ポート」と呼び，図式表現では 2 重楕円で表わす．上位階層からはサブプロセスの外部ポートに対するトークンの送受信が可能である．これは，非同期式通信を実現している．

同様に上位階層のプロセスからアクセスできるメソッドを「外部メソッド」と呼び図式表現では 2 重長方形で表わす．上位階層のメソッドとサブプロセスの外部メソッドは同期制約を設定することが可能である．これは，同期式通信を表現している．すなわち，「プロセス指向の階層性」および「プロセス間の同期制約」により，CCS や CSP のプロセス間の同期式通信が表現可能になる．

図式表現では，各階層（プロセス）は実線で囲むことによって表わされる．囲まれた階層は別のページに書くことができる．階層間のインターフェイスは「P-プラグ」と「M-プラグ」で図示される．P-プラグは外部ポートとの接続点を表わし，図式表現では短いバー（ポストボックスのイメージ）で表される．T-プラグは外部メソッドとの同期制約の接続点を表わし，図式表現では黒塗丸（電極のイメージ）で表される．例えば，CCS 記述：$P \equiv Q\,[med/out] \mid Q\,[med/in]$，$Q \equiv in.\,\tau.out.\,Q$ は，MENDEL ネットで表現すると図 4のようになる．

なお，最新のペトリネットの国際会議で発表された Modular CP-net [7] は，我々と同様のプロセス指向の階層性の定式化を行なっている．

3　MENDELS ZONE

3.1　設計手順

MENDELS ZONE における制御システムの設計方法論 [12] は以下の 5 つのフェイズから構成され

る（図5）：

（フェイズ1）要素プロセスの設計，
（フェイズ2）要素プロセスの接続と制御プロセスの生成，
（フェイズ3）制御プロセスとメソッドの設計，
（フェイズ4）検証と調整，
（フェイズ5）コード生成およびその実行と確認．

フェイズ1では，並行プログラムの要素プロセスのMENDELネットをMENDELネットエディタ（図6）を用いて設計する．次のフェイズ2で，要素プロセスをデータフローや同期制約で接続し，フェイズ3でそれらを制御する制御プロセスを設計する．この設計は，最初からMENDELネットを用いて設計するのは困難であるため，因果関係マトリックスを用いた段階的詳細化による設計を行う．ここでは，メソッドの抽出，スロットとポートの抽出，状態要素の抽出の順で設計を進める．またフェイズ3では，MENDELネットの各メソッドの詳細な挙動をKL1を用いて記述する．ここで，Metisを用いて代数的仕様記述から各メソッドのKL1プログラムを自動的に生成することも可能である[10]．このフェイズでMENDELネットの設計が完了する．次にフェイズ4で，時相論理を用いて，設計されたMENDELネットのタイミングに関する検証を行う．もしバグがあるならば調整を行う．最後に，設計されたMENDELネットは，そのテキスト形式である並行プログラミング言語MENDELに変換される．MENDELプログラムは，並列論理型言語ＫＬ１にコード生成され，並列推論マシンMulti-PSI上で実行でき，動きを視覚的に確認できる．ここでは，MENDELの各プロセスが各CPUに割り振られる．また，MENDELプログラムから分散システム上のC言語およびOccamのコードを生成することも可能である．

3.2 プログラム検証／調整

MENDELS ZONEでは，メソッドの時間的順序関係に関する制約条件を線形時間時相命題論理(LPTL)で記述し，MENDELネットが制約条件を満たしているかを検証する．理論的には一般のペトリネットの検証も可能である[9]が，計算時間が実用的でないため，ここでは有界なペトリネットに限定する．すなわち，MENDELネットの骨格を有限状態遷移グラフとして抽出し，LPTLのモデル検査法で検証する．

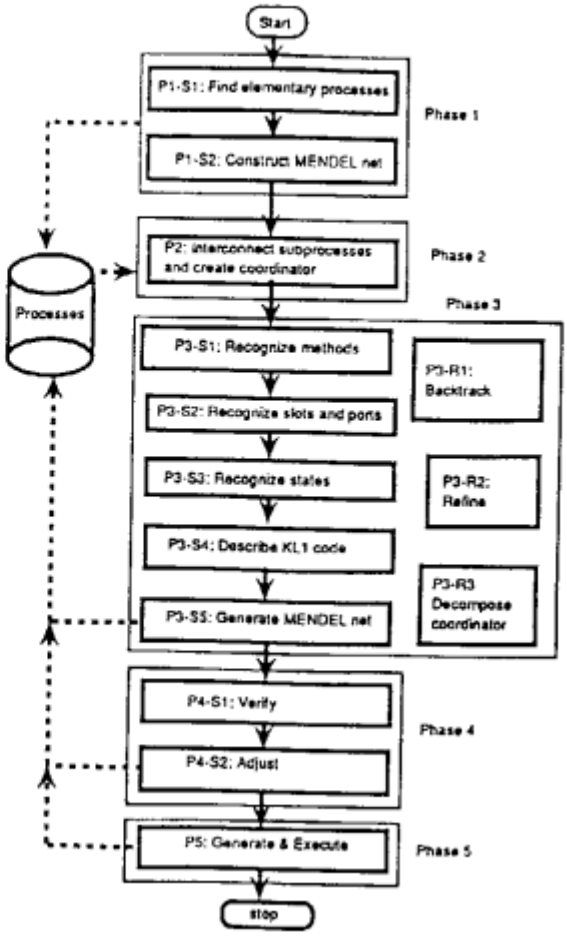

図5　MENDELS ZONEにおける設計方法論

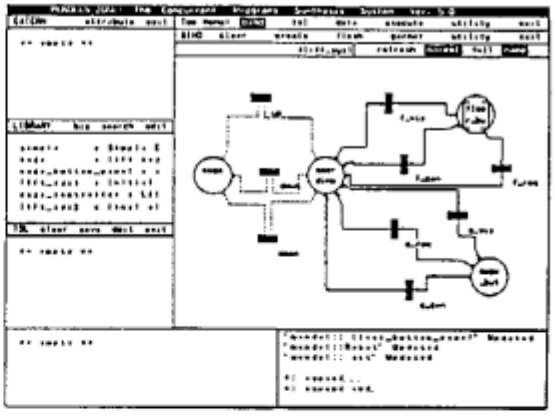

図6　MENDELS ZONE（画面イメージ）

検証の結果，MENDELネットが制約条件を満たさないことが判明した場合は，MENDELネットを調整する．ここでプログラム調整とは，元のMENDELネットにメソッドの発火順序を明示的に規定する状態要素と状態遷移を付加し，MENDELネットが与えられた制約条件を満たすように自動調整することである[11]．プログラム調整は，トーイプログラムしか生成できなかった時相論理を用いた

プログラム合成と比べてより実用的なアプローチである．

4 記述実験

4.1 例題：リフト制御システム

問題 [13]：m 階建てのビルに 1 台のリフト（エレベータのこと）が設置されている．以下の制約条件のもとでリフトの動きを制御するプログラムを設計せよ：(1) カゴの中には各階を示すボタンが付いている．行きたい階に対応するボタンを押すとそのボタンにランプがともり，カゴはその階に向かって移動し，その階に到着するとランプが消える．また，一回押したボタンをキャンセルすることもでき，そのときにもランプは消える．(2) 各階には行きたい方向に対応する 2 つのボタン（上方向と下方向）が付いている．ボタンを押すとランプがともり，カゴが到着するとボタンのランプが消え，カゴはボタンの示す方向に移動する．また，一回押したボタンをキャンセルすることもでき，そのときにもランプは消える．

4.2 MENDELS ZONE による記述実験

3.1の設計手順に従って，リフト制御プログラムを MENDELS ZONE で開発したときの，実際の設計過程を以下に示す．ここで,「Pi-Sj」は MENDEL ネットの設計方法論（図 5）におけるフェイズ i のステップ j を意味している．

(1) P1-S1：リフトシステムは 4 つの要素プロセス (cage, button, floor_button_panel, cage_button_panel) を持っている．

(2) P1-S2：各要素プロセスの MENDEL ネットを MENDEL ネットエディタで作成する．ここで，2 つのプロセス floor_button_panel, cage_button_panel はいくつかの button をサブプロセスとして構成される複合的な要素プロセスである．以下の MENDEL ネットの説明において，c は cage，f は foor，req は request，can は cancel，vis は visit を表している．

(3)P2：トップレベルのリフト制御システムは，3 つのサブプロセス (cage,floor_button_panel, cage_button_panel) から構成される．ここで，制御プロセス (coordinator) を導入し，サブプロセスのプラグ (open, up, down, c_req, c_can, c_vis, f_req, f_can,f_vis) を制御プロセスに接続する．

(4)P3-S1：因果関係マトリックスを用いて制御プロセスの MENDEL ネットを設計する．最初に，接続されてたプラグの情報から初期の因果関係マトリックスが自動生成される．すなわち，初期マトリックスは外部プラグに対応する 6 つの外部ポートと 3 つの外部メソッドだけを持っている．ここで，設計者は制御プロセスの「機能」を抽出しなければならない．抽出された機能はメソッド候補としてマトリックスに書き込まれる．ここでは，以下の機能が抽出された：

● open：カゴが目的階に到着したときドアを開ける．● up：目的階に向かって上昇する．● down：目的階に向かって下降する．● c_req / f_req：cage_button_panel / floor_button_panel からリクエストを受け付け，それを登録する．● c_can / f_can：cage_button_panel / floor_button_panel からキャンセルを受け付け，登録されているリクエストを消去する．● c_vis / f_vis：カゴがリクエストされている階に到着したとき，cage_button_panel / floor_button_panel に到着を通知する．

(5) P3-S2：上記のメソッドとサブプロセスのプラグの抽象レベルの因果関係を設定する．因果関係の設定を行う過程で,「データ」を抽出する．ここでは，以下に示す 3 つのデータ（スロット）とそれに付随する 1 つの追加機能（メソッド）が抽出しマトリックスに追加した：

● req_que（スロット）：リクエストを蓄積しているキュー．● current_f（スロット）：現時点においてカゴがいる階．● target_f（スロット）：カゴの目的階．● start（メソッド）：リクエストキューから目的階を選択する．

(6) P3-S2&S3：因果関係マトリックスの因果関係を段階的に詳細化していく．その過程で，新しい機能やデータが抽出できる．さらに，設計者は，この過程で状態遷移（状態要素）抽出する．

(7) P3-R1：設計者は,「カゴが目的階に向いながら通過する各階においても，もしその階がリクエストキューに登録されているならばドアを開けなければならない（リクエストされている階を無視して通過できない」ことを見過ごしていた．この見過ごしに気づいた時点で，設計の手戻りが発生する．ここでは，メソッド open の見直しが行われた．

(8) P3-R3：この時点でマトリックスはかなり複雑になった．そこで，制御プロセスを 2 つ（cage_controller,request_controller）に分割する．これは，「カゴ側の制御」と「リクエスト処理側の制御」という自然な分割になっている．この分割に伴い，cage_controller と request_controller のとのインターフェイスとして，以下のプラグが導入された：

● start（P-plug）：カゴへの起動指令．● end（M-plug）：カゴが目的階に到着したという通知．● exit(M-plug)：目的階に設定されているリクエストのキャンセル通知．● current(P-plug)：カゴの現在いる階．● command(P-plug)：カゴの停止／通過に対する指令．

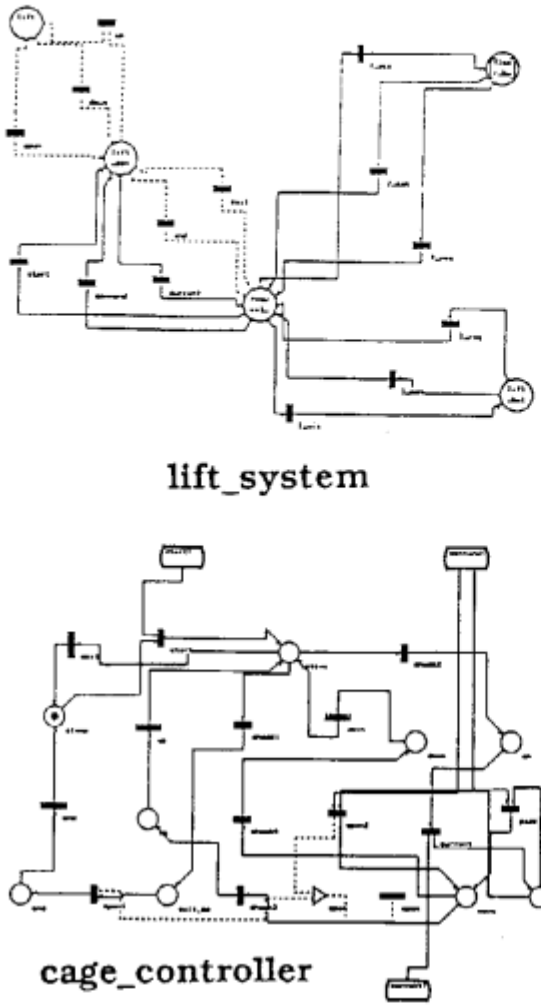

図7　リフト制御システムのMENDELネット

(9) P3-S3（cage_controllerに対して）：メソッド間の競合を抽出する．その結果，メソッド current, command, open, pass, up, down の生起に明示的な順序関係を入れる必要を認識し，9個の状態要素を導入した．

(10) P3-S5：マトリックス詳細化の最終段階として，メソッドの詳細をKL1で記述する．各マトリックスから制御プロセスのMENDELネットは自動的に生成される．図7はトップレベルのプロセス (lift_system) とその1つの制御プロセス (cage_controller) のMENDELネットである．

(11)P4:生成されたリフト制御プログラムのMENDELネットは有界ネットであり，以下の制約条件を満たすか否かの検証が可能である：

●デッドロックを起こさない．

●リクエストされた階には，それがキャンセルされないかぎりいつかは到着し，ドアを開ける．

この場合，MENDELネットはこれらの制約条件を満たすことが検証できた（すなわち，プログラム調整のプロセスは不用であった）．

(12)P5:最後に，MENDELネットからKL1プログラムが自動生成され，マルチPSI上で実行される．実行過程はMENDELネットの画面上に表示され，動きが視覚的に確認できる．

4.3　STATEMATEとの比較

Haralらによって開発されたSTATEMATE† [14]は，リアクティブ・システムの開発環境として定評のあるツールである．ここでは，記述実験を通してMENDELS ZONEとSTATEMATEを比較することにより，MENDELS ZONEの記述法（MENDELネット）およびツールの評価を行なう．‡

4.3.1　STATEMATE

STATEMATEは，対象システムを，3つの視点（構造的視点，機能的視点，動的視点）から，以下の3つのチャートを用いて記述する．

- module-chart: システムの物理的構造を表現する．3種類のモジュール (module)（システム，環境，記憶媒体）とモジュール間のデータと制御のフローを記述する．ポイントは，システムと環境との境界の明確化，および物理的制約からくるシステムの構造の決定である．
- activity-chart: モジュール内の機能 (activity) とデータストアを抽出，階層化し，それらの間のデータと制御のフローを記述する．構造化設計のデータフローダイアグラムに相当する．
- state-chart: 機能の実行順序関係を階層的状態遷移機械として記述する．

設計手順としては，まずmodule-chartを設計し，module-chartからactivity-chartを設計し，activity-chartをもとにstate-chartを設計する．さらに，activity-chartとmodule-chartを一貫性を保ちながら双方を段階的に詳細化していく．機能の詳細は，form languageにより記述される．計算機支援機能としては，3つのチャートの一貫性チェック機能，最も詳細化された段階でのシミュレーション機能，到達可能性解析機能などが用意されている．

† STATEMATEはi-Logix社のトレードマークです．

‡ ここでは，CASEツールとしてのSTATEMATEを実際に使用したわけではなく，論文レベルの設計方法論を用いて，手作業で記述実験を行なった．

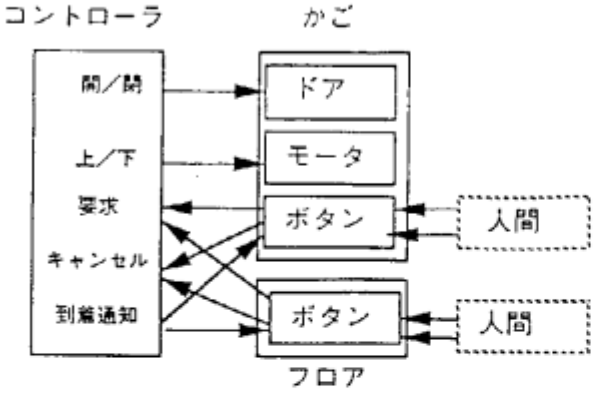

図 8　リフト制御システムの module-chart

4.3.2　STATEMATE による記述実験

STATEMATE でリフト制御システムを記述した．まず，module-chart（図 8）で，コントローラ，かご，ボタン，人間といった物理的要素の関係を明確にする．MENDELS ZONE では，システムの環境（ここでは，人間）を明示的に記述できなかった．次に，activity-chart（図 9）を記述する．activity としては，リクエスト受付，キャンセル受付，目的フロア選択，リフト制御，目的フロア到着処理，などがある．また，データストアとしてはリクエストキュー，目的フロア，カレントフロアなどがある．これらをデータ／制御フローで連結する．最後に，state-chart（図 10：遷移ラベルは省略）を記述する．ここでは，リフト制御システムの制御のモード（待機，アクティブ，目的設定，処理，目的達成など）をトップダウン的に，階層的状態遷移機械の特徴を生かしながら構築する．activity-chart と state-chart は基本的に独立した視点でお互いを意識せずに書かれる．そのため，2 つのチャートの詳細化の過程で，お互いを関連づける作業が非常にたいへんになる．実際，多くの手戻りが発生した．

4.3.3　評価

STATEMATE と MENDELS ZONE は，両方ともシーケンス制御システム，通信システム，組み込みシステムなどの離散事象システムを対象とした状態遷移ベースの CASE ツールである．記述方法に関しては，STATEMATE の上の 3 つのチャートは MENDEL ネットにほぼ対応させることができる（下記参照）．また，支援機能に関しても，同様にほぼ対応がとれる．§

§ STATEMATE は実際のツールを使っていないので，開発工数等の比較はできなかった．

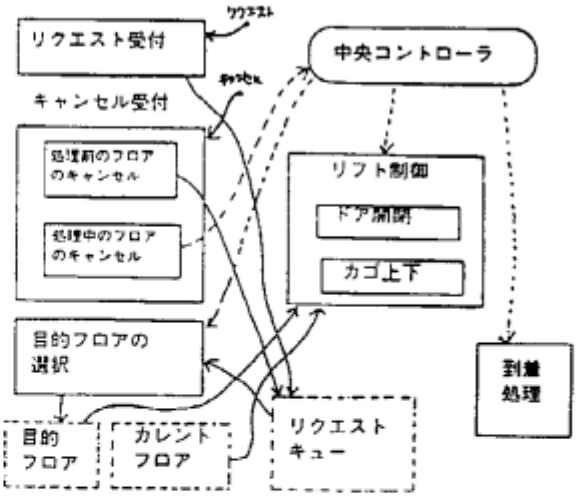

図 9　リフト制御システムの activity-chart

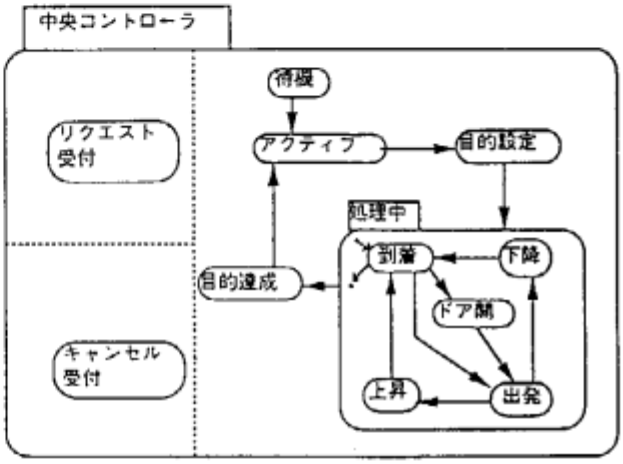

図 10　リフト制御システムの state-chart

STATEMATE	MENDELS ZONE
module (module-chart)	process
control activity (activity-chart)	coordinator
activity (activity-chart)	method
data store (activity-chart)	slot, port
state (state-chart)	state
data flow	arrow
control flow	arrow
form language	KL1

以下に，STATEMATE と比較した場合の MENDEL ネットの評価を述べる．

- アプローチ：機能に関する記述をデータフロー図で，制御に関する記述を状態遷移機械で図式的に記述するアプローチは，離散事象システムのための CASE ツールの本命である．MENDELS ZONE も STATEMATE も本質的には同じアプローチである．
- 視点に関して：STATEMATE は 3 つのチャー

ト，MENDELS ZONE は 1 つの MENDEL ネットで記述する点が，最大の相違点である．3 つの視点からの記述は，仕様があやふやな段階では MENDEL ネットより書きやすい（MENDELS ZONE の因果関係マトリックスがこれを一部補っている）．反面，3 つのチャートの関連づけが MENDEL ネットのように明示的ではない．これは，シミュレーションやデバッグ段階において，MENDEL ネットのほうが追跡性が良いということに反映される．

- 階層性に関して：MENDEL ネットの階層性がプロセス指向であるのに対して，STATEMATE の階層性は概念の階層性である．この階層性により段階的詳細化が自然に可能になる．特に，state-chart の状態遷移機械の階層性はリフト制御システムにおいても大いに役に立った．これは，例外処理の記述に非常に有効である．現状の MENDEL ネットでは例外処理がきれいに記述できず，大きな問題である．反面，STATEMATE では，生成されるソフトウェアのプロセス（タスク）構造が明示的に見えないという問題点がある（module-chart ではプロセスの構造まで書かない）．
- 支援機能に関して：時相論理によるプログラム検証／調整に関して，MENDELS ZONE の解析機能には理論的に進んでいる点がある．しかし，現状の計算機能力および検証アルゴリズムでは検証能力に限界があり¶，実際の大規模システム開発ではそれほど効果を発揮できないと思われる．しかし，時相論理による検証を含むシステムのグローバル状態遷移解析というアプローチは将来的には有望であり，今後の実用化研究が非常に重要である．
- その他：MENDEL ネットには，STATEMATE のようなシステムモジュールと環境モジュールの区別がない．

以上の比較結果を大胆にまとめると，STATEMATE は要求定義段階（3 つの視点による要求の立体的具体化，概念の階層性による段階的詳細化），MENDELS ZONE は設計および解析段階（設計されたシステムの追跡性，プロセス構造の明示化，論理的検証の可能性）に重点があるといえるかもしれない．

5 結論

本稿では，MENDELS ZONE における高レベルペトリネットに基づく並行プログラミング環境を提案し，記述実験を通して評価した．今後は，実際の開発への適用に不可欠な設計方法論の確立／洗練化，および検証／調整能力の改善に取り組む．なお，本研究は第 5 世代コンピュータプロジェクトの一環として行なわれたものである．

¶ 現状ではリフト制御システム規模の検証が限界である．

参考文献

[1] K.Jensen: Coloured Petri Nets: A High Level Language for System Design and Analysis, Advances in Petri Nets 1990, LNCS 483, Springer-Verlag (1990).

[2] H.J.Genrich, K.Lautenbach: System Modelling with High-Lebel Petri Nets, Theoret. Comput. Sci. Vol.13 (1981).

[3] W.Reisig: Petri Nets and Algebraic Specification, Theoret. Comput. Sci., Vol.80(1991).

[4] 寺内睦博:ネット指向ソフトウェア構築ツールについて，情報処理学会ソフトウェア工学研究会，86-T5 (1992)．

[5] P.Huber, et al.: Hierarchies in Coloured Petri Nets, Advances in Petri Nets 1990, LNCS483, Springer-Verlag (1990).

[6] FGCS92 デモンストレーション説明資料 (1992).

[7] S.Christensen, L.Petrucci : Towards a Modular Analysis of Coloured Petri Nets, Application and Theory of Petri Nets 1992, LNCS616, Springer-Verlag (1992).

[8] T. Chikayama: Overview of the Parallel Inference Machine Operating System, Proc. FGCS'88 (1988).

[9] N.Uchihira, et al.: Verification and Synthesis of Concurrent Programs Using Petri Nets and Temporal Logic, Trans. IEICE, Vol.E73, No.12 (1990).

[10] S.Honiden, et al.: An Integration Environment to Put Formal Specifications into Practical Use in Real-Time Systems, Proc. 6th IWSSD (1991)

[11] 内平 他：MENDELS ZONE における並行プログラムの階層的調整，情報処理学会ソフトウェア工学研究会 83-20(1992).

[12] N.Uchihira, et al.: Petri-net-based Programming Environment and its Design Methodology for Cooperating Discrete Event Systems, Trans. IEICE, Vol.E75-A, No.10 (1992).

[13] Problem Set, Proc. 4th IWSSD (11987).

[14] D.Harel et al.: STATEMATE: A Working Environment for the Development of Complex Reactive Systems, In Proc. of 10th ICSE (1988).