DEIM Forum 2013 E4-4

到達可能ペトリネットモデルのフィードバック制御

池田　雄太† 　島田　 諭† 　三浦　孝夫†

† 法政大学 工学研究科　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †yuta.ikeda.4s@stu.hosei.ac.jp, ††satoshi.shimada.57@hosei.ac.jp , †††miurat@k.hosei.ac.jp

あらまし　離散事象システムのモデリングツールであるペトリネットでは，作成したモデルで作業の完了を表す終了状態への到達が可能であるかどうかがシステムの検証において問題となる．本研究では到達可能性を損なわないようモデルを拡張していくことで到達可能なモデルのみを作成する．また，シミュレーション結果を得ることで，ユーザーが到達可能性を考慮せずにモデルの改善を行うことができる環境を提案する．
キーワード　ペトリネット, 到達可能性, シミュレーション,

Feedback control of Reachable PetriNet

Yuta IKEDA†, Satoshi SIMADA†, and Takao MIURA†

† Dept.of Elect.& Elect. Engr., HOSEI University　3-7-2, KajinoCho, Koganei, Tokyo, 184–8584 Japan
E-mail: †yuta.ikeda.4s@stu.hosei.ac.jp, ††satoshi.shimada.57@hosei.ac.jp , †††miurat@k.hosei.ac.jp

Abstract
Key words　Petri Net, Reachability, Simulation,

1. 前　書　き

近年, 大規模な物流, 生産施設などが増えてきておりシミュレーションによって, どの程度の効率が見込めるか, それに見合ったコストでの設備投資ができるかなどを事前に検証することは重要になってきている．さらにシミュレーションにより何か問題点が見つかることも考えられる．

そこで離散事象システムを有向グラフでモデル化することができるペトリネットを利用することで直感的なシミュレーションが可能となる．また GUI を用いてペトリネットモデルを作成することで，利用者のイメージと実行イメージを一致させることができる．

ペトリネットの問題として初期状態からある状態への到達が可能かどうかという到達可能性問題がある．シミュレーションを行う際，終了状態へ到達可能であることはモデル作成の最低条件であるが，作成したモデルから到達可能性を求めることは計算量の問題などから非常に困難である．また時間ペトリネットなどの機能拡張を行うと決定不能となる．しかし，到達可能性が保証されることで構築されたペトリネットモデルは健全に動作することを保証できる．

到達可能性を維持したままモデルを作成する Well-Formed PetriNet を利用することで直感的に健全なモデルを作成することができる．しかしペトリネットはモデルの把握などには向いているが, 例えば生産システムなどの場合, 材料が加工されるまでにかかる平均時間や, 加工されるまでの順番待ちの時間といったものの統計を取ることに向いていない．そこでモデルの作成などに専門の知識が必要になるが統計の取得に向いた記述型のブロック型シミュレーションを統計の取得の際に利用することで，ペトリネットの直感性とブロック型シミュレーションの統計情報の両立を行う．

本研究では到達可能であるモデルを拡張していくことで到達可能性を維持したままモデルを作成する Well-Formed PetriNet を利用しペトリネットの到達可能性を保証する．またペトリネットにより作成したモデルの構造を解析し，ブロック型シミュレーションを行い，得られた結果をペトリネットモデルに与えることでペトリネットで作成したモデルでの統計情報の取得，またその結果のフィードバックによるモデルの改善を行える環境を提案する．

2 章ではペトリネットについて述べる．3 章ではペトリネットの到達可能性について述べる．4 章では Well-Formed PetriNet について述べる．5 章でブロック型シミュレーションについて述べる．6 章で GUI によるエディターを示し,7 章で結びとする．

2. ペトリネット

ペトリネットは離散事象システムをモデル化する表現技法である．2 種類のノードがあり, 条件を表すプレースを○, 事象を表すトランジションを｜で表現し, この 2 つをアークという条件と事象の関係を表す矢印で結ぶ．プレースとトランジションにはラベル（名前）をつけることができ，ラベルがある場合はそれぞれの近くに記述する．さらにシステム内を動作する要素

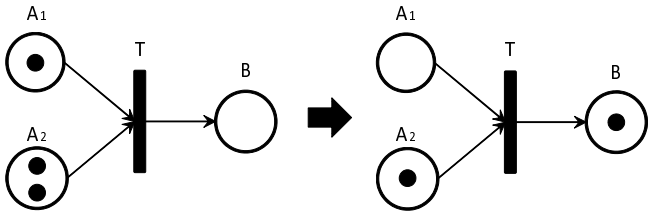

図 1 動 作 例

であるトークンを●で表現し, ペトリネットモデルでは初期状態でプレース内にトークンが存在することがあり，これを初期トークンと呼ぶ 1 つのプレース内に多数トークンがある場合はプレース内の数字によりトークンの個数を表現する．また本稿では，トークンの発生するプレースをソース，トークンが滞留するプレースをシンクと呼ぶ.

プレースの入力に所望のトークンが滞留すると遷移（トランジション）可能となり，実際の遷移が生じることを発火するという．ペトリネットの状態はトランジションの発火によって遷移し，トランジションに入力されるプレース（以下入力プレース）すべてに条件を満たす数のトークンが存在する場合, トランジションは発火可能となる．トランジションが発火すると, 各入力プレースからトランジションに接続しているアークの重み分だけトークンが消費され, トランジションから出力されるプレース（以下出力プレース）に重みの分だけトークンが発生する．トランジションに付けられた添字で発火可能になった時間から発火までの遅延時間を表し，これによって作業時間を表現することができる．ただし添字がない場合は発火可能となった時にただちに発火する．またアークは正の整数の重みを持ち, アークの重みだけトークンは消費, 発生する．重みは矢印の本数, または添字で表現される.

ペトリネットの動作する例を図 1 に示す．図 1 ではトランジション T が発火することで入力プレース A1,A2 からそれぞれ 1 つのトークンが消費され, 出力プレース B に 1 つのトークンが発生している.

3. ペトリネットの到達可能性

ペトリネットの到達可能性とは，初期状態からある状態に遷移が可能であるかどうかを表す状況をいう．ある動作をペトリネットで表現した場合，動作の終了を示す終了状態に到達可能であることがその動作が正常に終了するために必要となり, ちょうどプログラムの停止性判定と同じ意味を持つ.

初期状態から遷移する状態ノードの連結性を全て調べても到達可能性は分からない，というのは発火可能性を検査せずに到達するかどうかが判定できないからである．よく似た性質に連結性がある．ソースノードがシンクノードに連結するかを見ればよく，ノードが多くなると状態数が爆発的に多くなってしまうが，経路の決定は可能である.

到達可能性の検証について，与えられた多項式がある多項式イデアルに含まれるかどうかを決定する問題に還元できること，そして，その計算の複雑さが指数領域完全（EXPSPACE-complete）となるという有名な結果が 1981 年 Mayr によって得られている [7]．効率向上を行うため本稿では到達可能性を 2 つに分けて考える．1 つ目はモデル構造による到達可能性であり，各ノードの繋がり方などモデル構造の点から到達可能な状況のみを繰り返してモデルを構成する．文法やグラフ構造に強い制約を加え到達可能な状況だけを保証する．2 つ目は内部状態における到達可能性である．確率的あるいは非決定的に定まる分岐や分裂・合体を想定しながら終了状態を判定できるか, という問題は，時間やカラー概念を含む場合には決定不能である．実際，終了状態に遷移する前にデッドロックに陥るなど, 実行状況に依存する要素が少なくない．到達可能性も不可能性も判定不能である.

ペトリネットにおいてモデルが到達可能でなくなる原因として (1) 1 つのプレースやトランジションから複数のアークが出ていく多重アーク (2) 同一のイベントを繰り返すループ (3) イベントによるトークンの分岐，合体があり，これらの機能によって到達不可能な状態となる.

本稿では終了状態まで到達可能である基本的なモデルを，到達可能性を維持したまま部分的に変形させることを繰り返すことで終了状態まで到達可能なモデルを作成する Well-Formed PetriNet（WFPN）を利用する.

WFPN の初期状態と終了状態のみで構成された基本モデルを図 2(a) に示す．このモデルから次に示す変形を行うことでモデルを拡張する．ここではそれぞれの図において線で囲まれた部分が変形された部分である.

[1] イベントの追加　[図 2(b)] イベントが増えても途中の状態が増えるのみ
[2] イベントの分岐　[図 2(c)] イベントが複数あっても遷移後は同様の状態
[3] 並行イベント　[図 2(d)] 分裂したトークンは分裂したもの同士で合体してから次の状態へ遷移
[4] ループイベント　[図 3(a)] 本稿では同様の動作を X 回行った後に次の状態へ移る
[5] サブルーチンイベント　[図 3(b)] 終了状態へ到達可能なモデルなら，モデル内部のイベントと置換可能

本研究ではこのような操作だけを用いてモデルの作成を行う.

4. ブロック型シミュレーション

本研究では，作成したペトリネットモデルをシミュレートした場合の結果を取得するためにブロック型のシミュレーションを利用する．ブロック型のシミュレーションは，特定の動作を行う数種類のノードを矢印アークでそれぞれ接続することでモデルを作成する．ノードにパラメータを与え，それぞれのノードの繋がりを指定することによってシミュレーションを行い，各ノードにおける統計結果を得ることができる．しかし，ブロック型シミュレーションでのモデルは文法に従い制御文を書くことなどで作成されるため，利用するシステム固有の知識が必要となってしまう．また作成したモデルに文法上のミスはな

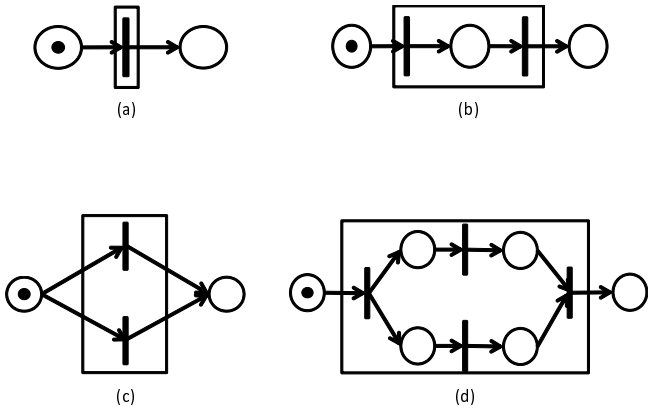

図 2　ペトリネット変形 1

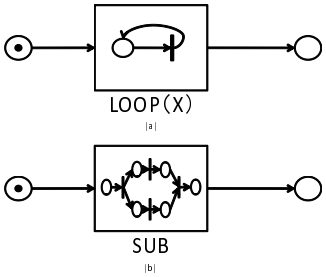

図 3　ペトリネット変形 2

くとも，イメージしているモデルと一致しているかどうかの検証も必要となる．

そのため本研究では，モデルの作成を直感的に行えるようペトリネットで行い，完成したモデルをブロック型のシミュレーションの制御文に変換することで作成したモデルの統計情報を得る．加えて得られた統計情報をペトリネットモデルの各ノードごとに関連付けることによってシミュレーション結果の考察が容易となる．

ブロック型シミュレーションでは，一般的にシミュレーション終了までの時間の他に各種設定した統計値の平均，標準偏差，最大値や最小値などを得ることができる．本研究ではトランジション毎に”最大待ち行列長”,”平均待ち時間”,”利用率,”,”連続稼働時間”,”連続不稼働時間”,”作業回数”を取得する．これらの統計値はペトリネットでは以下のように考えることができる．

[1] 最大待ち行列長
トランジションで作業を行う前にトークンが最大で何個待ち状態になったかを表す．

[2] 平均待ち時間
トランジションに到達してから作業が完了するまでのトークンの滞在時間の平均値を表す．

[3] 利用率
シミュレーションが終了するまでの時間でトランジションがどれだけ動作していたかを表す．

[4] 連続稼働時間
シミュレーション中にトランジションが連続でどれだけ稼働していたかを表す．

[5] 連続不稼働時間
シミュレーション中にトランジションが連続でどれだけ稼働していなかったかを表す．

[6] 作業回数
何回トランジションで作業が行われたかを表す．

これらの情報から改善の必要な作業などを発見することができる．

5. Well-Formed PetriNet エディタ

ここでは，GUI によるペトリネットモデルの作成，統計情報の取得について述べる．Well-Formed PetriNet エディタ（以下 WFPN エディタ）を起動すると，基本モデルとして図 2(a) の基本イベントが用意されている．ユーザはこのモデルを WFPN のルールに従い変形させていくことでモデルの作成を行う．サブルーチンを利用する場合はサブルーチンモデルを別のウインドウで作成する．

ここでは自転車の製造モデルを例としている．図 4 のメインモデルでは初めに本体の作成と塗料の準備が並行して行われる．ここでは上側のサブルーチン 1 が図 5 で表されるような本体の製造を行い，下側では塗料の準備が行われる．塗料は色の選択によって 2 つの作業のうちどちらか一方が行われる．本体の製造と塗料が揃うと自転車は塗装作業を終えて完成する．その後，何台かの自転車はサブルーチン 2 が表す詳細なテストを行い，他の自転車は簡単なテストをして出荷される．

モデルの作成が終わったら，各ノードに作業時間などのパラメータを入力する．その後シミュレーション実行ボタンを押すことで，作成したペトリネットモデルがブロック型シミュレーションによって実行され結果が取得される．取得した結果の中から利用率が大きくなっている工程があった場合は，ボトルネックになっている可能性が高いとして該当するトランジションが図 5 のように赤く表示されるというように，モデルにフィードバックが行われる．また各トランジションを選択することでそれぞれの統計情報が図 4 のように表示される．サブルーチンモデルでも図 5,6 のように同様に表示される．また図 7 のように全てのトランジションをまとめて表示することも可能で，複数のトランジションの比較などに利用できる．

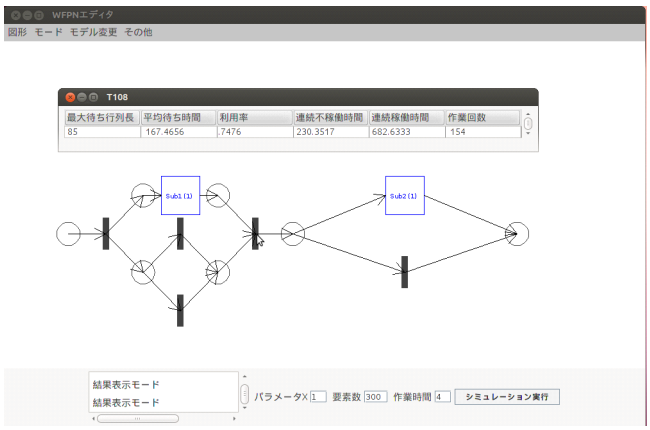

図 4　メインモデル

赤く検出されたトランジションの情報を見ると T205 という

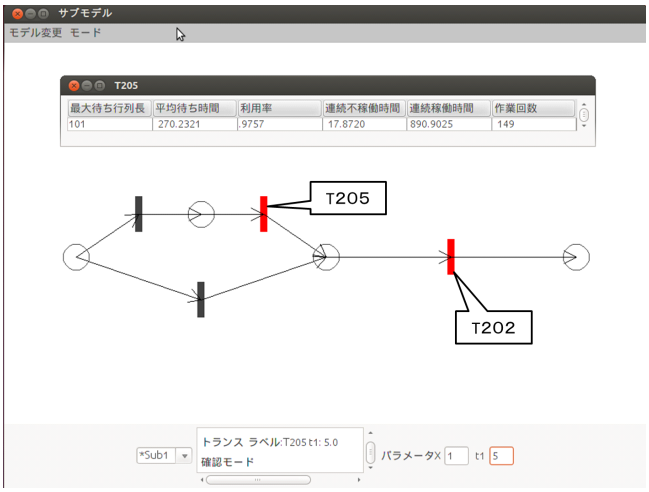

図 5　サブモデル 1

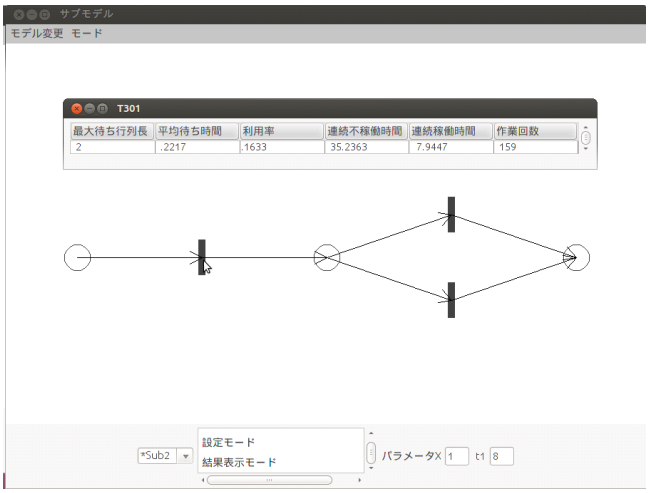

図 6　サブモデル 2

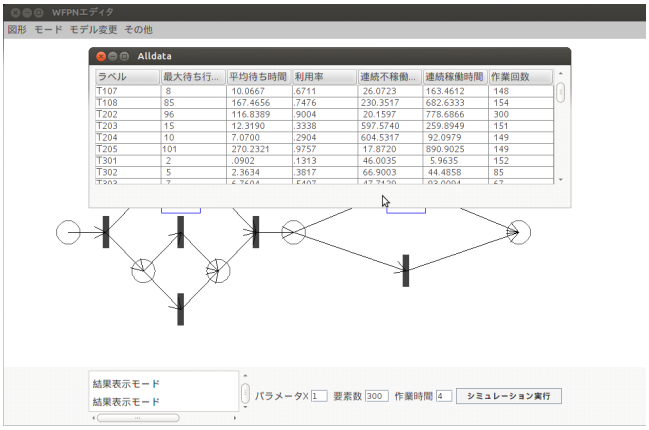

図 7　全　結　果

トランジションの最大待ち行列長が 101 と大きくなっている．そこで T205 での作業時間を平均 5 分のところを平均 4 分に変更して再びシミュレーションを実行することで，他の部分は変えずに T205 だけ新しいパラメータに更新されシミュレーションが行われる．このように各パラメータを調節してシミュレーションを実行することでモデルの変更結果を容易に得ることができる．

図 8 の上部にて T205 の修正前，下部にて修正後の結果を示す．修正後には T205 の最大待ち行列長が 79 に減少していることが確認できる．しかし，T205 の後に繋がる T202 の最大待ち行列長や平均待ち時間が増加し，T205 の改善は全体として効果が小さい．全体の作業時間を見ると 300 台の自転車が完成するまでに修正前が”913 分”，修正後が”897 分”となっている．ここで T205 ではなく T202 の平均作業時間を 1 分減らすと作業完了までに”817 分”と T205 を変更するより大きな効果が出ている．このことからより少ない変更でより良い結果を得るためにも，統計情報の大小だけでなく，いろいろなパラメータで試行錯誤することが有効であることがわかる．

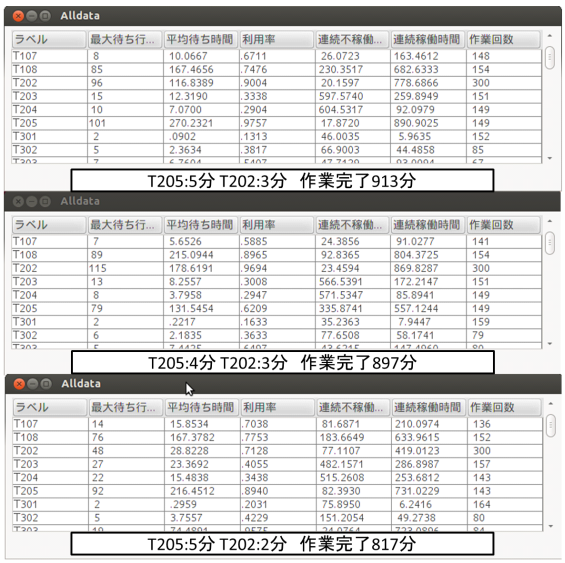

図 8　T205 修正前後

6. 結　　論

本研究では到達可能なペトリネットモデルを構築する手法を利用した GUI ベースのペトリネットエディタを実装し，作成したモデルの統計情報をブロック型のシミュレーションを利用して取得することで，ユーザーが到達可能性を考慮せずにモデルの改善を行う環境を提案した．多くの作業工程の中から自動的にボトルネックになっている可能性が高いものをフィードバックし，パラメータを変更してのトライアンドエラーを容易にすることで，単純な数値の大小に寄らず効果的な改善案を得ることができた．

今後の課題としては，パラメータの変更前後の統計情報の比較の補助や，構造などを加味したより効果的なフィードバックの実装，ペトリネットモデルによるシミュレーションの途中状態の表示などが考えられる．

文　　献

[1] 松本 篤, 三浦 孝夫 ：GPSS による時間ペトリネットの性能解析, 2001 年電子情報通信学会総合大会
[2] 椎塚 久雄 ：実例ペトリネット, コロナ社,1992.
[3] J.L. ピータースン 市川 惇信 小林 重信 ：ペトリネット入門, 共立出版株式会社,1984.
[4] 池田 雄太, 三浦 孝夫 ：Reachable PetriNet Editor, DEIM2012
[5] 池田 雄太, 三浦 孝夫 ：Prototyping Color Timed Petri Nets,

PACRIM2011
[6] 池田 雄太, 島田 諭, 三浦 孝夫 ：Well Formed PetriNet for Reachablity, SITIS2012
[7] Mayr, E.W.: An Algorithm for the General PetriNet Reachability Problem, ACM STOC, 1981, pp.238-246
[8] Gomaa,A., Adam, N. and Atluri, V.: Color Time PetriNet for Interactive Adaptive Multimedia Objects, 11th International Multimedia Modeling Conference (MMM'05), 2005 pp.147-157
[9] Jensen, K.: Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Springer, March 1995
[10] Reisig, W.: Petri Net, *Springer-Verlag*, 1985