ビジネスプロセスとビジネスルールのための モデル検査手法による検証

Verification by Model Checking Method for Business Process and Business Rule

秦 良平 †, 小形 達也 ‡, 飯島 正 ‡
Ryohei HATA†, Tatsuya OGATA‡, and Tadashi IIJIMA‡
† 慶應義塾大学大学院　理工学研究科
‡ 慶應義塾大学　理工学部
†Graduate School of Science and Technology, Keio Univ.
‡Faculty of Science and Technology, Keio Univ.

要旨

近年，業務プロセス管理 (BPM; Business Process Management) と並んで，業務ルール管理 (BRM; Business Rule Management) が重視されている．本研究は，業務プロセスにおいて「時間制約に関する業務ルール」をモデル検査手法を用いて検証することを目指すものである．モデル検査には，時間に関する検証を行うツール UPPAAL を採用する．著者らの提案する業務プロセス表現に時間概念を導入し，それを時間オートマトンに変換することによって，時間制約に関する業務ルールの検証を可能にする．

1. はじめに

近年，業務プロセス管理 (BPM; Business Process Management) と並んで，業務ルール管理 (BRM; Business Rule Management) が重視されている．本研究は，業務プロセスにおいて「時間制約に関する業務ルール」をモデル検査手法を用いて検証することを目指すものである．

これまで，著者らは,「組織間にまたがるワークフロー」における組織間の協調と相互のやり取りを扱うこと，ならびに，アクター (実行者/サブシステム) を資源として扱うこと等を目的に，業務プロセスの表記法として「オブジェクト指向ペトリネット」を導入してきた．さらに，業務ルールを「ドメイン固有言語 (DSL; Domain Specific Language)」 [1] により表現し，それを統合することで，フローの構造変換やリソースの再配分，パラメータの外部化を実現してきた [5][6][7].

本報告では，業務プロセスにおいて時間制約により与えらる業務ルールについて，著者らが提案する業務プロセスの表記法に対して，時間制約を含めたモデル検査ツールに適用できるような変換操作を実施する方式の実現可能性の確認を試みる．

続く第 2 節で「時間制約に関する業務ルール」について，さらに，第 3 節で「オブジェクト指向ペトリネットによる業務プロセス表現」関して記述する．その後，第 4 節において，デンマークの Aalborg 大学とスウェーデンの Uppsala 大学の共同で開発されたモデル検査ツールの UPPAAL [8] の概略を紹介する．そして，業務プロセスの UPPAAL による検証を可能にするため，第 5 節においてペトリネットから時間オートマトンへの変換方法に関して述べた後で，第 6 節で適用事例を示す．

2. 時間制約により与えられる業務ルール

業務ルールとは「業務権限の範囲内のルール」であり，すべてのルールは「行動ルール (運用ルール)」か「定義づけのルール (構造ルール)」のどちらかに属す [2].「行動ルール」は，行為や行動に関する制約を表現することが多く，そこに必要であれば，判断や行動に関する評価をも含む．一方,「定義付けのルール」は概念を分類体系化して定義するものであり，それに基づいて「計算」を引き起こすが多い．

著者らは,「定義づけのルール」に関しては，DSL によって業務プロセスからの分離，外部化を実現している [7]．たとえば，運賃計算を行う業務プロセスにおいて，特定区間に対する運賃は，たとえば「東京名古屋間の運賃は，18000 円である」というルールによって定義できる．このような定義づけのルールは，条件と結果の決定表 (Decision Table) や決定木 (Decision Tree) を利用すると，理解しやすく管理しやすい形で表現（定義付け）できることがよくあり，多くの業務ルール管理システムで導入されている．そこで我々も，こうしたルールに対しては，一般的な if-then ルール形式の DSL 表現に加えて，決定表を導入することで，業務ルールというドメインの知識の共有の実現とともに，プロセスからルールを分離し，業務プロセス管理の効率化を実現してきた [7].

一方,「行動のルール」とは，たとえば「インターネットでチケットを予約した時点から 48 時間以内に購入を完了しなければならない」というような「当事者の行動を規制する制約」である．一般には，制約条件に加えて，業務プロセスがその制約を満たすために，追加的に実施する行動そのものも含めて扱うことがある．これまで，筆者らが取り組んできたフローの構造変換 [6] やリソースの再配分 [5] は，制約条件を充足することを実現するために業務プロセスの一部を加工する操作と位置づけることができる．これらは，制約条件の成立・不成立を評価する機構を含むことで有効に自動化することができる．すなわち「行動のルール」を有効に扱うには，制約条件と，それを評価する機構を提供することが必要である．

制約条件にはいろいろな種類のものが考えられるが，その中でも時間制約に関するものは重要である．人を含む業務プロセスを情報システム化し，自動運用するためには，タイムアウト処理は，ほぼ不可欠といえよう．そこで本報告では，時間制約に関する業務ルールを対象とし，著者らが取り組んできたオブジェクト指向ペトリネットによる業務プロセス表現に統合することを試みる [1].

3. オブジェクト指向ペトリネット

3.1. オブジェクト指向ペトリネットとは

本研究では，業務プロセスの表現としてオブジェクト指向ペトリネットを用いている．オブジェクト指向ペトリネットとはオブジェクト指向概念に基づいてモジュール性を取り入れたペトリネットの拡張モデルであり，多くのものが提案されている．著者らの研究プロジェクトでは，nets-in-nets 意味論に基づく参照ネット (Reference Net) [3] の考え方を採用している．

本研究で，オブジェクト指向ペトリネットを導入した理由には，以下のようなものがある:

(1) 組織間にまたがるワークフローにおける組織間の協調を扱うため
(2) アクター (実行者/サブシステム) を資源として扱うため

以下では，それらの目的に関連付けて，簡潔に位置づける．

3.1.1. 組織間にまたがるワークフロー

一つの業務フローは，一部門/一組織内の一つの業務内のローカルな，業務アクティビティ (作業，タスク)，ないし，呼び出される外部サービスの間の制御フロー (すなわち業務ロジック) を表現する．すなわち，アクティビティやサービスの**オーケストレーション (Orchestration)** を規定するものと位置づけられる．そうしたオーケストレーションに対し，業務間 (部門間/組織間) のメッセージのやり取りに規定するものを一般に**コレオグラフィ (Choreography)** と呼ぶことがある．この二つの区別は概念的には理解できるが，現実の業務を考えた場合には，境界を厳密に規定することは難しい．組織，部門，担当者というように細分化していくと，これらの境界をどこに設けるかは，ケースによって異なる．

3.1.2. アクターを資源として扱う

参照ネットにおいてはトークンには 2 通りある．一つは，通常の P/T ネット (Place/Transition ネット) におけるトークンである．単純トークン (Black Token) とよぶ．プレースに存在する単純トークンは，トークン数で示す．もう一つは，別のサブシステムを表現するサブネットへの参照 (reference) に相当する参照トークン (Reference Token) である．

プレースは単純トークン用の単純プレースと，参照トークン用の参照プレースに分類できる．

- 単純トークン (Black Token)[2]
- 参照トークン (Reference Token)

ここで，参照トークンが参照しているサブネットは，オブジェクト・ネットと呼ばれ，一つのオブジェクトとしてのアイデンティティをもつ．参照トークンも，単純トークンと同様，ネットワーク中を遷移することができるが，一つの参

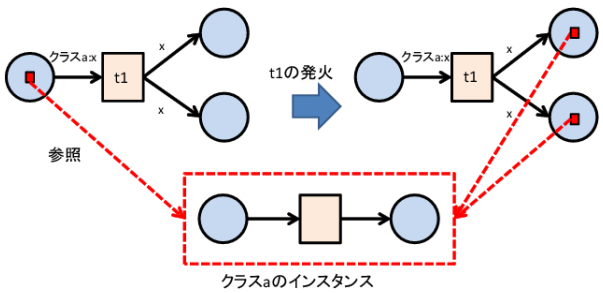

図 1: 参照トークンの例

[1]実際には，本研究プロジェクトの初期段階においても，モデル検査ツール SPIN を使った時間制約の導入には取り組んでいる [4]．しかし，本報告では，それを業務ルールの一部として再設計し，あらためて業務プロセスと業務ルールの統合という枠組で扱うことを目指している．

[2]慣習的にブラック・トークンと呼ばれるが，ここでは分かり易さのために単純トークンと呼ぶ

照トークンがトランジションの発火によってコピーされることもある．その際には，あくまで共通のオブジェクトネットへの参照がコピーされる点に注意されたい．

後述する (図 2 左) ように，各トランジションの発火の際には，そのトランジションと同期する (interaction 関係にある) ペトリネットにおいては，同名のメソッドが実行される．すなわち，そうした同期関係 (interaction 関係にある) ペトリネットが実行主体 (アクター) として位置づけられることになる．

3.2. オブジェクト指向ペトリネット OPeN

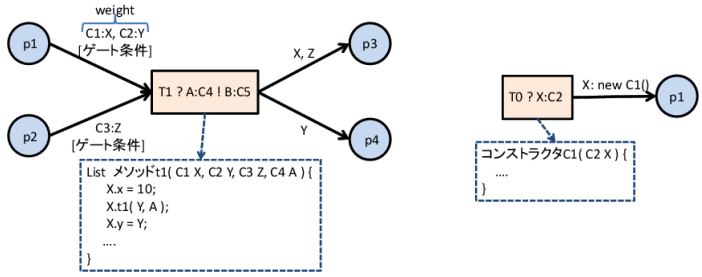

図 2: OPeN におけるアークとトランジション

以下では，具体的なオブジェクト指向ペトリネットのモデルとして，著者らの研究プロジェクトで継続的に開発を行ってきたオブジェクト指向ペトリネット OPeN ファミリー (the Object-oriented Petri Net family) の中でも，特に人と人の連携して行う協調作業や，SOA におけるサービス間連携を記述するためのビジネスモデル記述 (ワークフロー) に特化した OPeN/WF を用いている．

OPeN では，個々のオブジェクト・ネットは，クラス記述から生成されるインスタンスに相当する．クラス記述は，オブジェクトを定義するものであり，opn(もしくは従来からの慣習から pnmlx) という拡張子を持つ XML ファイルとして記述する．クラスが定義するオブジェクト・ネットは，そのオブジェクトの振る舞い (ライフサイクル) を記述する一つのペトリネット，そのオブジェクトが持つプロパティ (インスタンス変数) 群と，ライフサイクルを遷移する際に使われるメソッド群である．メソッドは，主に，トランジションの発火条件であるゲート条件の記述や，トランジション発火時に起動されてプロパティの値の更新に使われる．図 2 に OPeN におけるアークとトランジションのアノテーションを示す．

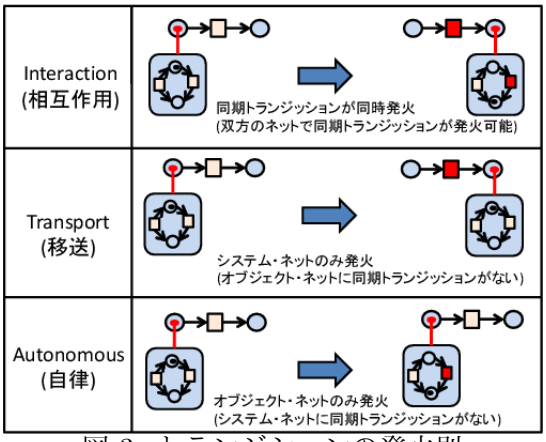

図 3: トランジションの発火則

トランジション t_1 の発火によって，同じ名前をもつメソッド t1 が起動される [3]．現時点のプロトタイプ仕様では，プロパティの型やメソッドの定義のための言語仕様は，厳密には定義していないが，図 2 に準拠し，基本的なリフレクション機能を備えた一般的なオブジェクト指向言語 (たとえば Java や Scala) のクラス定義 (ネットと同名のクラス定義) に外部化しており，相互に対応付けている．

ここでは，全体的な統制を記述する方を EOS の慣習に従いシステム・ネットと呼び，統制されたサブオブジェクトの側をオブジェクト・ネットと呼んで説明することにする [4]．

システム・ネットとオブジェクト・ネットの間では，一部のトランジションの発火が同期的に行われる (それによって，システム・ネットはオブジェクト・ネットの動作をオーケストレートする)．システム・ネットとオブジェクト・ネットの同期関係は，双方のトランジションの対の集合として表現される．システム・ネットにおいて，あるトランジション T が発火するためには，以下の三条件が成り立たねばならない [5]．

(a) システム・ネットにおいて，単純トークン B_i に関して発火条件を満たしていること，

[3]後述するオブジェクト指向ペトリネットの発火則「Interaction(相互作用)」により，t_1 と同期するトランジション t_{s1} が，このペトリネットのサブネット中に存在し，かつそれが発火可能であれば，その t_{s1} も同時に発火する．t_{s1} が発火可能でなければ t_1 も発火できない．トランジションの入力アークがメソッドの入力パラメータに対応する．

[4]階層的なシステムの場合には，ここでいうオブジェクト・ネットもまた，より下位のサブペトリネットからみれば相対的にシステム・ネットに相当する．OPeN は EOS の概念を多階層に拡張しているが，混乱のない限り，最上位ネットに限らず，隣り合う階層のネット間の関係を，システムネットとオブジェクト・ネットというように称することとする

[5]但し，OPeN の場合には，二階層の EOS ではなく，多階層を許容しているので，条件 (c) は再帰的に，より下位層にあるオブジェクト・ネットに対して適用されていくことになる．

(b) システム・ネットにおいて，参照トークン R_j に関して発火条件を満たしていること，
(c) T の発火に寄与している参照トークン R_j が参照しているオブジェクト・ネット中で，T と同期関係にあるトランジションが発火可能であること．

こうした発火のメカニズムから，nets-within-nets 意味論 (ないし参照意味論)，およびその拡張に基づくオブジェクト指向ペトリネットにおいては，(a) interaction(相互作用), (b) transport(移送), (c) autonomous(自律)，という 3 通りの発火則が考えられる．本報告では，図 3 を載せるに留め詳細は省く (必要があれば [3][7] 等を参照してほしい)．

4. モデル検査手法

システム構築においては，そのシステムが「必要な処理を実行し，なおかつ安全に動作する」という信頼性が求められる．その信頼性を確認する方法としてはテストがある．しかし，単体テストのためにも，少なくともテスト対象のコンポーネントは完成させる必要があり，さらに，それらを組み合わせた統合テストも必要である．また，テストケースの網羅性も留意しなければならない．

そこで，システムのモデルを設計した時点で，そのシステム上で起こり得る状態に関して,「システムが満たすべき動作を完了する」あるいは「満たすべきでない状態にならない」といった検証項目を網羅的に確認できるモデル検査手法が注目されてきた．モデル検査を行う検証ツールは様々あるが，今回は，時間オートマトンによるモデルを検証する UPPAAL [8] を採用する．

5. 有界ペトリネットから有限オートマトンへの変換

ペトリネットにより記述した業務プロセスの時間制約を含めたモデル検証を行うために，ペトリネットから UPPAAL で検証できる時間オートマトンへ変換を行う．

まず，業務プロセス中の時間表現は捨象して，単なるペトリネットを対象に考える．但し，トークン数が無限に増え続けるプレースを含む非有界ペトリネットは対象としない．ペトリネットではシステムの状態は，全てのプレース上におけるトークン数をベクトルで表現したマーキングにより表される．有界ペトリネットをオートマトンに変換するためには，このマーキングを一つの「状態」と見なし，マーキングの変化の要因となるトランジションの発火を「状態間の遷移」として対応付ければよい[6]．すなわち，ペトリネットから全ての取り得るマーキングとその際の発火可能トランジションを調べ上げることにより，たとえば，図 9 のペトリネットを，図 10 のように変換することができる．

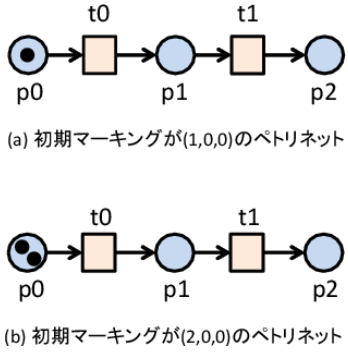

図 4: 変換前のペトリネット

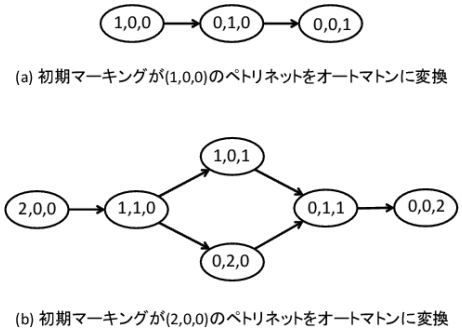

図 5: 変換後のオートマトン

但し，トークンが単純トークン（ブラックトークン）であれば，この通りでよいが，オブジェクト指向ペトリネットのリファレンストークンの場合，マーキングは単にトークン数で表すことはできず，どのオブジェクトネットへの参照（オブジェクト識別子）であるかを区別する必要がある．そのため状態数は組み合わせ的に増えることがある．たとえば，図 9(b) のペトリネットにおけるトークンは，ブラックトークンなので図 10(b) のオートマトンへ変換できるが，図 6 左のオブジェクト指向ペトリネットの場合，トークンは参照トークンなので，変換後のペトリネットは図 6 右となり識別される状態の個数が増えてしまう[7]．

[6] これはペトリネットの解析手法の一つである可達木 (Reachability Tree) ないし，それをグラフ化した可達グラフに相当する．可達木は，初期マーキングを根とし，その後の発火によるマーキングを子として木を成長させて，全ての取り得るマーキングを調べ上げたものである（可達グラフは共通の状態をまとめたグラフ）

[7] 論文執筆時に実装が完了している変換ツールは，この参照（オブジェクト識別子）には対応していない

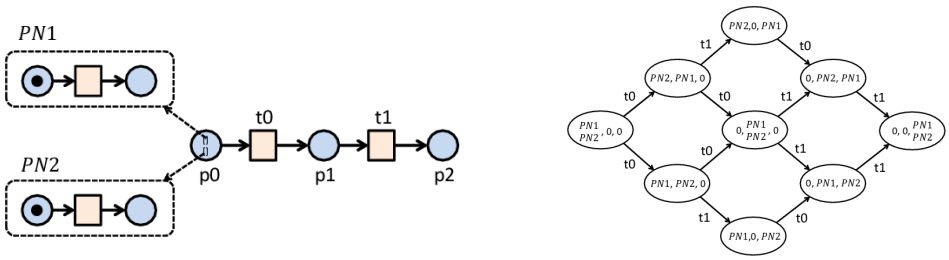

図 6: 参照トークンのオブジェクト識別性を考慮したオートマトン (可達グラフ) 変換

6. 適用事例

オブジェクト指向ペトリネットで表現した業務プロセスに時間表現を加え，UPPAAL でモデル検査を行う例を示す．ある会社における社員の出張旅費申請のプロセスを挙げる．このプロセスにおいてオブジェクトとして表現するものを社員，領収書，旅費申請書，財務部を定義した．それぞれのプロセスをペトリネットで図 7 のように表わすものとする．

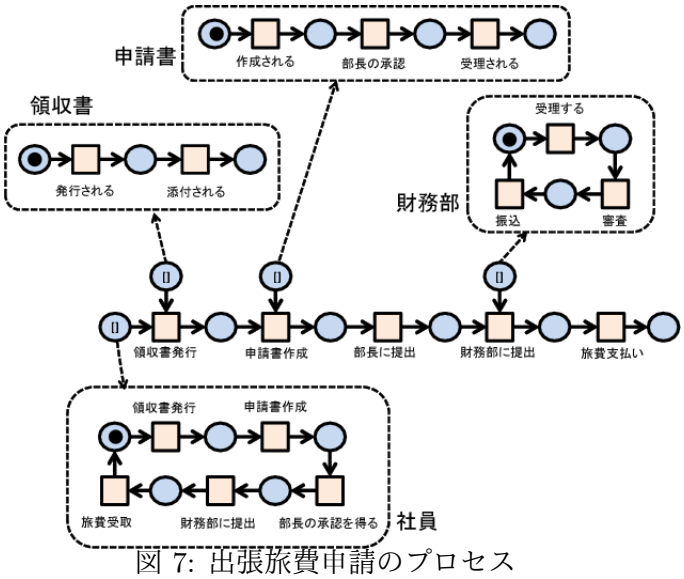

図 7: 出張旅費申請のプロセス

この例題のプロセスの概要を説明する．まず (1) 社員が出張において領収書を入手する．次に (2) 社員は出張旅費申請書を記入し，申請書に領収書を添付する．(3) 必要事項を記入した申請書は，部長に提出して承認を得る．このプロセスにおいては，部長が行うタスクは申請書の承認のみであり，状態の変化を持たないので，プロセス表現は省略している．(4) 社員が部長の承認済みの出張旅費申請書を財務部に提出すると，財務部はそれを審査し，受理すれば社員に旅費を支払う (社員は受け取る) ことでプロセスが完了する．

このオブジェクト指向ペトリネットで表現された業務プロセスのアーク上に時間表現を加え，時間オートマトンに変換することで，UPPAAL で検証できる入力データとなりうる．その際，個々のオブジェクト指向ペトリネットを時間オートマトンに変換した上で，一つの時間オートマトンに合成するか，オブジェクト指向ペトリネットを一つの時間ペトリネットに合成した上で，時間オートマトンへ変換するか，という 2 通りの方針が考えられる．本論文執筆時点では，オートマトンの合成ツールは実装中であるため，オブジェクト指向ペトリネットを手作業で一つのペトリネットに合成し，時間オートマトンに変換することで，本方式の有用性を確認した．

その合成ペトリネットが図 8 である．

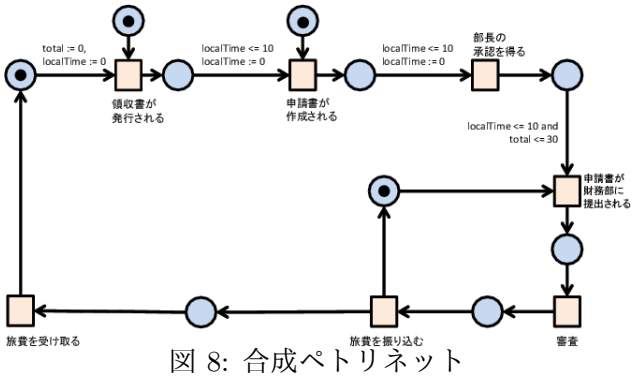

図 8: 合成ペトリネット

さらに，以下のような時間制約に関する業務ルールを定義し，ペトリネットに付与した時間表現の下，これが成立することを検証する．この例題は，全体の流れを確認するためのものであるため，極めて単純なものに過ぎないが，ペトリネットのアーク上に，そこを通過する参照トークンに対しての時間制約等を記述する [8].

[8] その参照先オブジェクトが内部に持つクロックへの操作とその値を下にする条件式

- 社員は部長の承認済みの出張旅費申請書を領収書が発行された日から 30 日以内に財務部に提出しなければならない．

このルールは次のような検証式で定義することが出来る．

- A[] (total > 30 imply not (Process.marking_0011000001 or Process.marking_0000100001 or Process.marking_0000010001))

この検証式は，すべての実行系列で常に括弧内の特性が成り立つことを表しており，成立すべき特性は,「出張旅費申請プロセスの実行に要する時間の累計 (totaltime) が，プロセスの完了に至る前に 30 日を超過することはない」である．検証式中の状態表現はペトリネットのマーキングをもとにした表現なので，利用者にとってわかりやすいものではない．本論文執筆時点では未着手であるが，利用者にとって理解しやすい表現で検証式を与えると，それを UPPAAL における検証に利用できる表現に変換するツールも構築予定である．

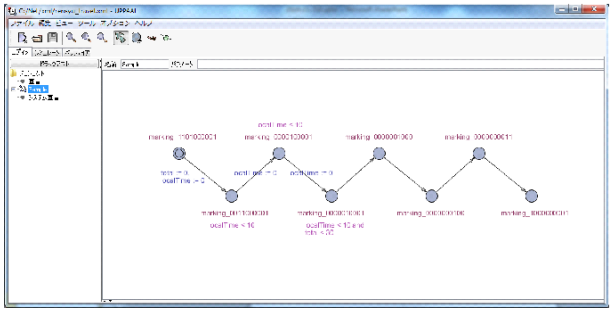

図 9: UPPAAL 上での時間オートマトン表現

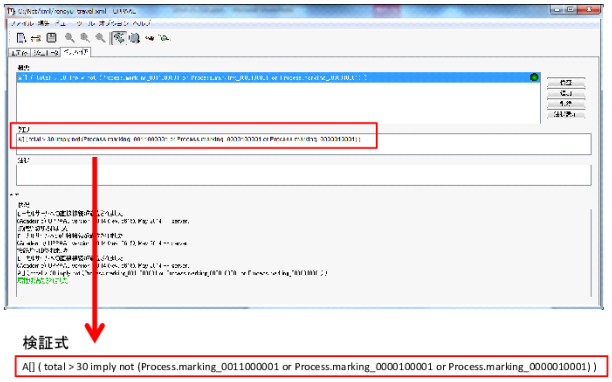

図 10: 検証式と検証の様子

7. まとめ

オブジェクト指向ペトリネットによる業務プロセス表現に，時間制約による業務ルールの表現を導入するために，ペトリネットを時間オートマトンに変換してモデル検査を実施することを試みることで，この方式の有用性を確かめた．今回は方式の実現可能性確認を目的としたため，一部に未だ自動変換ができていない部分は手作業で補ったが，引き続き変換ツールの自動化を進めていく．

謝辞

本研究は，当研究室・Net 班 (業務プロセスグループ) による蓄積の上に成り立っている．同班の卒業生 (金子良太 (2013 年度修士了), 片山 輝彦 (2012 年度修士了)，高橋 貴大 (2012 年度学部卒)，新間 理貴 (2010 年度修士了)，孫 騰涛 (2009 年度修士了)，塚原 浩太 (2009 年度学部卒)) 各位に感謝します．

参考文献

[1] Debasish Ghosh, (監訳) 佐藤 竜一: “実践プログラミング DSL ～ ドメイン特化言語の設計と実装のノウハウ,” 翔泳社, 2012.
[2] ロナルド・G・ロス, “アジャイル経営のためのビジネスルールマネジメント入門,” 日経 BP 社, 2013 年.
[3] Rüdiger Valk: “Object Petri nets? Using the nets-within-nets paradigm, ” LNCS 3098, pp.819–848, Spriner, 2004.
[4] 孫 騰涛，塚原 浩太，飯島 正: “オブジェクト指向ペトリネットによるワークフローのモデル化と分析,” 第 5 回 全国大会・研究発表大会, 情報システム学会, 2009.
[5] 飯島 正, 片山 輝彦, 金子 良太, 高橋 貴大: “オブジェクト指向論理ペトリネットを使った業務プロセス/業務ルール管理,” 第 8 回 全国大会・研究発表大会, 情報システム学会, 2012.
[6] 飯島 正: “アスペクト指向ワークフロー変換 ～ オブジェクト指向ペトリネットによるワークフロー表現への適用 ～,” 技術報告 (知能ソフトウェア工学研究会), Vol.112, Vol.165, pp.1-6, 電子情報通信学会, 2012.
[7] 飯島 正, 秦 良平, 金子 良太: “オブジェクト指向ペトリネットとルールに基づく業務プロセスの理解支援,” 第 9 回 全国大会・研究発表大会, 情報システム学会, 2013.
[8] “UPPAAL,” http://www.uppaal.org/