グラフ数え上げ手法を用いた 例からのビジネスプロセスモデリング

A Business Process Modeling and Refinement by Examples – A Graph Enumeration-based Approach –

飯島 正 †

Tadashi Iijima†

† 慶應義塾大学　理工学部

†Faculty of Science and Technology, Keio Univ.

要旨

本報告では，例からビジネスプロセスを定義すること (PMRbE; Process Modeling and Refinement by Examples) ことを提案するものであり，それを具体化する方法の一つとしてグラフ列挙手法によるアプローチの構想について述べるものである．

「例からのビジネスプロセス定義」とは，イベント列（トレース）の集合を与えると，その全てのイベント列をたどることのできるプロセス定義（モデル）を生成するものである．しかし，与えられたイベント列をたどることのできるプロセス定義（許容モデル）は，当然ながら複数存在する．その中で，もっとも妥当なモデルを選ばねばならない．そこで，そうしたプロセス定義（許容）を列挙して，まずは制約条件を満たすことで，フィルタリングし (generate and test)，さらに，アーク数最小化（ノード数最小化）等の基準により，適切なプロセス定義を選択する手法について述べる．

アーク数を最小化するために，エビデンスとなる「トレース例」を持たないアークは，仮説として，無効化される．実行トレース例を漸進的に増やていくと，新たに付け加えられた「トレース例」をたどることができるように，無効化されていたアークが有効化されるなど，モデル構造が変化していく．

本来，プロセスマイニングにおいては，与えられる「例」がログであることから，正例しか与えられない．しかし，本研究で提唱する「例からのビジネスプロセス定義」では，目的がインタラクティブなプロセス定義にあるので，正例以外の負例を与えたり，制約条件を付与したりしつつ，プロセス定義を行なっていく点で特徴的である．

また，その目的から，プロセスマイニングと異なり，大量のログデータから大規模なプロセスを生成するというよりは，比較的少量のトレース例を，ユースケース・モデリングと同じように，主系列と代替系列，例外系列を意識しつつ漸進的に投入しながら，対話的に操作してモデリングを行なう．そのため，ビジネスプロセスは，比較的粒度の小さいモジュールに分解されていて，協調しあうプロセス群として扱えることが望ましい．この観点から，筆者らの研究プロジェクトで，数年に渡り活用している，オブジェクト指向ペトリネットをベースとしたビジネス・プロセス表現を前提として考えている．

1. はじめに

プロセスマイニング [1] [2] は，ビジネスプロセスのイベントログから有益な情報を抽出する技術一般を指す．発見 (discovery)，適合性評価 (conformance)，強化 (enhancement) が，その重要なキーワードといえる.「発見」は，イベントログを入力として，それを満たすモデルを出力する技術を指し,「適合性評価」はモデルとイベントログを入力として，両者の適合性を判定する技術,「強化」は，やはりモデルとイベントログを入力として，より適合性の高いモデルを出力する技術といえる．

本報告は，イベントログに対応するような系列事例（example sequence もしくは sequence instance) を与えることで，それを満たすモデルを生成し，継続的に事例を与えることで，モデルを洗練化していく対話的な「例からのビジネス・プロセス・モデリング」(PMRbE; Process Modeling and Refinement by Examples) を提唱するものである．

そこには，プロセスマイニングの発見技術，適合性評価技術，強化技術がいずれも重要な位置を占める．しかし，プロセスマイニングとの大きな違いは，与えられる入力が，ログ（記録）データではないということである．ここでは，モデルを定義するために，あえて典型的な事例としてモデリング作業者が，考えている系列を与える．更に，それを満たすモデルを洗練化するために，更に系列事例を与えていく．

この手法は，現場のモデリング作業者が「現場の専門家」(domain expert) であって，必ずしも，モデリング技術自体の専門家ではないということを想定している.「現場の専門家」は，抽象化したモデルを定義することよりも，現実の事例を列挙していくことの方に長けているであろう，という想定に基づいている．ユースケース (シナリオ) に基づく開発手法や，シナリオ駆動型設計 (Scenario-driven Design) の一種とも言える．

典型的な系列事例からモデルを生成したあと，例外的な系列事例を与えていくことで，モデルを強化することができる.「例からのビジネス・プロセス・モデリング」支援システムを構築した場合，そのシ

ステムは，モデリング作業者と対話的に機能する．支援システムは，与えられた系列事例集合を満たすモデル群を生成し，それを何らかの基準に基づいて優先順位付けをして提示する．現場のモデリング作業者は，それを見ながら，さらに改善 (refinement) していく．

本報告は，上記のようなモデリング支援アプローチを提案するにあたり，その実現方法としてグラフ列挙手法を取り上げる．グラフ列挙問題とは，与えられた条件を満たす部分グラフを全て列挙する問題であり，そのための手法 (ZDD を用いたフロンティア法) が Graphillion ライブラリとして公開されている．このライブラリを利用することによって，条件を満たすプロセスモデルを生成し，洗練化を行なう．

以下では，第二節で「例からのビジネス・プロセス・モデリング」(PMRbE; Process Modeling and Refinement by Examples) の考え方について述べた後，続く第三節では,「例からのビジネス・プロセス・モデリング」の実現可能性を検討するために試作してきた，系列データとオブジェクト指向ペトリネットによるプロセス記述との相互変換ツールについて紹介する．現場の専門家にとっても馴染み深いと考えられる表計算ソフトで作成した系列事例をオブジェクト指向ペトリネットに変換し，系列事例に手を加えるとそれをペトリネットに反映させるものである．しかし，現時点の試作段階では，系列からのループ検出には，極単純なヒューリスティックスを取り入れた一般性の低いアドホックな手法しか実装していない．オブジェクト指向ペトリネットでは，プロセスはオブジェクトという単位でモジュール化される．プロセスはアクタや資源に相当するオブジェクト毎に作成され，それらが同期しながら連携動作する．第四節では，それを踏まえてグラフ列挙手法によるアプローチの構想について述べる．

2. 例からのビジネス・プロセス・モデリング

プロセスマイニングは，ビジネスプロセスのイベントログから有益な情報を抽出する技術一般を指す．その概要は，IEEE プロセスマイニングタスクフォース (`http://www.win.tue.nl/ieeetfpm/`) から各国語に翻訳されて公開されている「プロセスマイニングマニフェスト」 [1] でうかがい知ることができる．

発見 (discovery)，適合性評価 (conformance)，強化 (enhancement) が，その重要なキーワードといえる.「発見」は，イベントログを入力として，それを満たすモデルを出力する技術を指し,「適合性評価」はモデルとイベントログを入力として，両者の適合性を判定する技術,「強化」は，やはりモデルとイベントログを入力として，より適合性の高いモデルを出力する技術といえる．今後，データマイニングの対象として，多様なセンサーによる大量のデータの活用が益々発展し，それが，ヒューマンアクティビティを含む情報システムの改善につながることが期待できる．

本報告で提唱する「例からのビジネス・プロセス・モデリング」(PMRbE; Process Modeling and Refinement by Examples) は，このプロセスマイニングに使われる技術を利用する．但し，実際のシステム運用の記録であるログデータを入力とするのではない．モデリング作業者が，モデリングしようとしているビジネスプロセスで，発生しうるイベント系列を想定し，それを列挙して与えていくことで，プロセスを定義していくものである．

ここで，モデリング作業者としては，プロセスモデル定義に精通している技術者ではなく，ペトリネットや BPMN といったプロセス記述言語に慣れていない「現場の専門家 (domain expert)」を想定している．そのため，現場の用語で具体的な事例系列を挙げていくことはできても，プロセス記述言語でモデル化することは得意としていない．

イベント系列の事例は，モデリング作業者が典型的な系列であると考えるものから，順に継続的に与えていく．それに対し，モデリング支援システム（現在，試作の途中段階）は，そうした系列集合を満たすプロセスモデル群を生成する．もちろん，そうしたモデル群は一意に定まらないが，何らかの基準に基づいて，その中の少数のみ提示する．モデリング作業者は，それを元に，さらに系列事例を与えていくことで，対話的に洗練化する．典型的な系列事例に基づいて基本モデルを構築し，例外的な事例や特殊な事例を追加していくことで，モデルを洗練化することができる．

系列事例からモデルを出力するだけではなく，モデルから系列（パス）を生成し，それをモデリング作業者が確認するといった相互の変換も可能であると考えている．一旦，構築されたモデルに対し，事例系列に手を加える操作（イベントの変更，イベントの挿入，削除など）を反映させることも想定して

いる．こうしたアプローチは,「現場の専門家」にとっても扱いやすいものであると考えている．

3. 系列事例とオブジェクト指向ペトリネットの相互変換ツール

「例からのビジネス・プロセス・モデリング」の実現可能性を検討するために，系列データとオブジェクト指向ペトリネットによるプロセス記述との間の，簡易的な相互変換ツールを試作している (図 1).「例からのビジネス・プロセス・モデリング」の方法論の構築を進めるにあたり，小規模な評価実験から有効性の検討を並行して実施するためである．

このツールでは,「現場の専門家」にとっても馴染み深いと考えられる表計算ソフトをつかって系列事例を記述すると，その系列事例を満たすオブジェクト指向ペトリネット [3] に変換することができる．但し，本稿執筆時点の試作段階では，系列事例データからの，分岐やループ検出には，極めて単純なヒューリスティックスに基づく一般性の低いアドホックな手法しか実装しておらず，十分ではない．

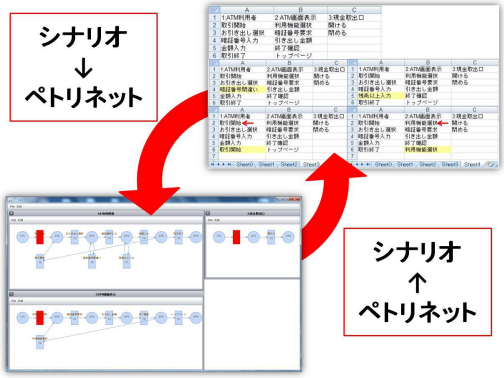


図 1: 相互変換ツール

また，系列事例に手を加えるとそれをペトリネットに反映させ，逆に，ペトリネットに手を加えると系列事例にその変更を反映させることで,「現場の専門家」にもモデルの修正変更を容易にすることを目指しているが，系列中のイベント名の変更，イベントの削除，イベントの追加など，簡単な修正変更にしか対応していない．

ここで，ビジネス・プロセスの表現には，オブジェクト指向ペトリネットを使っており，各プロセスモデルは，アクタや資源を表す単位（オブジェクト）にモジュール化され，互いに同期して振る舞う．モジュール化しているため，各オブジェクト毎のモデルは比較的，小さなサイズにとどまることが期待される．オブジェクト指向ペトリネットに関しては，付録に記載するが，現在のツールで扱っているペトリネットは，その非常に限られたサブクラスに相当する．トランジッションの発火をイベントに対応づけているので，イベント名とトランジッション名は同じである．また，オブジェクト間の同期制御を行なうペトリネット（システムネット）を 1 つに限定し，その配下に複数のオブジェクトネットをもつ二階層に限定したモデルであり，さらにオブジェクトネットの構造はほぼ状態遷移図に対応するものに限られている．

4. グラフ列挙手法を使った対話的プロセスモデリングのアプローチ

4.1. 例からのプロセスモデリング

前節で紹介したツールでは，系列事例集合からモデル構築の機能は，かなり限定された範囲にとどまっている．一般的にいえば，系列事例集合からのモデル生成の機能は，プロセスマイニングにおける「発見 (discovery)」に相当する．プロセス発見の方法には，α-アルゴリズム [5]，その欠点を解消する拡張である α^+-アルゴリズム [6]，遺伝的アルゴリズム手法，領域理論に基づく手法 (状態に基づく領域手法，言語に基づく領域手法)，帰納論理プログラミングによる手法，ヒューリスティックス・マイニング手法などがある [2].

今回は，目的が対話的なビジネスプロセスのモデリング支援にあるので，それに向いた手法を検討している．現在検討中の手法の一つが，グラフ列挙アルゴリズム [4] の利用である．グラフ列挙は，与えられたグラフから，条件を満たす部分グラスを抽出することを意味し，実行系列（path）からプロセスモデルを生成するプロセスマイニングの逆操作に相当する．

ここでは，簡単のために，ペトリネットに基づくプロセス表現へ変換する前段階として，イベントの発火系列を表現する遷移系までを取り扱う．イベントフローグラフは，イベントをノードとし有向アークでつないだ有向グラフとなる（唯一の開始イベントと，唯一の終了イベントは，必要に応じて補うも

のとする）．すなわち，本報告では，ログデータからイベントフローグラフを生成する段階までを論じることとする（言い換えれば，イベントフローグラフからペトリネットに基づくビジネスプロセス表現への変換部分に関しては省略する）．

まず，今作ろうとしているプロセスモデルを包含するようなイベントフローグラフ（proto-model もしくは proto-graph と呼ぶ）を与える．たとえば，proto-model として，ログに含まれる全てのイベントをそれぞれノードとし，各ノードを全結合した有向グラフを作ったとする．その部分グラフの集合から与えられイベント系列をたどるイベントフローグラフを列挙することで部分集合を作成すると，解となるイベントフローグラフは，その中に含まれる．最初に与えた proto-model に依存するが，そのような部分グラフは多数存在する．

4.2.proto-model の構成方法

proto-model の構成方法には，いろいろな手法が考えられる．今回は，proto-model の生成に，素朴な α-アルゴリズムを基礎として用いることを試みた．ログは，トレース（イベント系列）のマルチ集合で表現できる．たとえば，ログ $L_1 = \{abcd^3, acbd^2\}$ が与えられた例をもって，α-アルゴリズムの前半（イベントフローグラフの生成まで）を紹介する．

- (1) α-アルゴリズム [5] では，ログに含まれている実行系列（トレース）をもとに，イベント間の直接的な基本順序関係 (Log-based ordering relation) $\succ_W$,$\rightarrow_W$,$\#_W$,$||_W$ を抽出する．
  - (1-1) まず，トレース中のイベント直接先行関係 $\succ_W$ を抽出する．イベント x とイベント y の間の直接先行関係 $\succ_W$ は，以下の関係を意味する．

$$\text{直接先行関係}\quad x \succ_W y \iff x \text{ が } y \text{ に直接先行するトレースが存在する} \tag{1}$$

    * $\{a \succ_{L_1} b, b \succ_{L_1} c, c \succ_{L_1} d, a \succ_{L_1} c, c \succ_{L_1} b, b \succ_{L_1} d, a \succ_{L_1} e, e \succ_{L_1} d\}$
  - (1-2) そこから，単方向順序関係 $\rightarrow_W$，非直接先行関係 $\#_W$，並行関係 $||_W$ を導出する．

$$\text{単方向順序関係}\quad x \rightarrow_W y \iff (x \succ_W y) \wedge (y \not\succ_W x) \tag{2}$$
$$\text{直接関係未定義}\quad x \#_W y \iff (x \not\succ_W y) \wedge (y \not\succ_W x) \tag{3}$$
$$\text{並行関係}\quad x ||_W y \iff (x \succ_W y) \wedge (y \succ_W x) \tag{4}$$

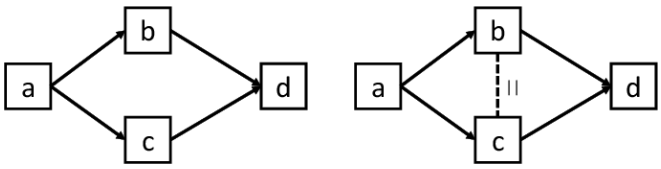
- (2) footprint 表を構成する (表 1).
- (3) この footprint 表から，基本的なイベントフローグラフを作成を作成することができる (図 2).

表 1: ログ L_1 の footprint 表

	a	b	c	d
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$||_{L_1}$	$\rightarrow_{L_1}$
c	$\leftarrow_{L_1}$	$||_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$

このアルゴリズムでは，長さ 1 ならびに 2 の「短い」ループを生成できない (長さ 3 以上は問題ない) ことが知られていて，その問題に対処する拡張も提案されている [6]．ログ $L_2 = \{abcd, acbd, abcbd, abcbcd\}$ が与えられたとすると，このモデルには長さ 2 のループが含まれていると判断することが妥当となる．

α-アルゴリズムの拡張である α^+-アルゴリズムでは，長さ 1 ならびに 2 の「短い」ループについてのみ特別に前処理ならびに後処理で対応する．長さ 1 のループは，トレース中に同じイベントが連続すればすぐ分かるので，前処理として，一旦，それを一つのイベントへ縮退させておき，最終的に，そのイベントを意味するノードに自己ループを付与すればよい．長さ 2 のループに関する問題は，並行関係 $x||_W y$



L_1 に対する妥当なイベントフローグラフの一つ　　並行関係 $||_{L_1}$ を明示したイベントフローグラフ

図 2: L_1 に対するイベントフローグラフ

長さ 1 のループを持つイベントフローグラフの例　　長さ 2 のループを持つイベントフローグラフの例

図 3: 「短い」ループを持つイベントフローグラフの例

と，双方向にに付与された単方向順序関係 $(x \rightarrow_W y) \wedge (y \rightarrow_W x)$ とが区別できないことに起因している．そこで，長さ 2 のループを識別するための関係 $\diamond_W$ を導入し，イベント間の直接的な基本順序関係を再定義する．まず，長さ 2 のループの存在を以下の関係で表現する．

$$x \triangle_W y \iff \text{ログ } W \text{ 中に } xyx \text{ という部分系列を持つトレースが存在する} \tag{5}$$

$$x \diamond_W y \iff (x \triangle_W y) \wedge (y \triangle_W x) \tag{6}$$

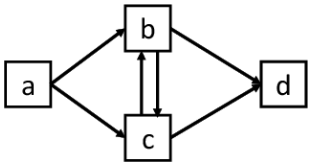
この関係を用いて，長さ 2 のループを，並行関係 $x||_W y$ から除外する．

$$x \rightarrow_W y \iff (x \succ_W y) \wedge (y \not\succ_W x \vee x \triangle_W y) \tag{7}$$

$$x \#_W y \iff (x \not\succ_W y) \wedge (y \not\succ_W x) \tag{8}$$

$$x||_W y \iff (x \succ_W y) \wedge (y \succ_W x) \wedge x \not\diamond_W y \tag{9}$$

これにより，長さ 2 のループが含まれたイベントフローグラフを得ることができる．



L_2 に対する妥当なイベントフローグラフの一つ

図 4: L_2 に対するイベントフローグラフ

4.3.proto-model の操作

こうして，α-アルゴリズムもしくはその拡張である α^+-アルゴリズムに基づいて構築した proto-model に対し，対話的なツールによって漸進的に，

- 直接先行関係 $x \succ_W y$ が提示されていないイベント対 x, y を footprint 表から選び，
- 直接先行関係 $x \succ_W y$ を追加する

操作を行なうことにより，生成されるモデル（イベントフローグラフ）が変化する．これを繰り返しながら，モデルを洗練化していく．その際，モデルの操作前後で，BDD/ZDD を利用して，系列データ集合の順序関係を満たす部分グラフだけを列挙させて差分を提示することで，利用者がモデルを理解することを支援する．

proto-model の操作には，イベント間関係の追加以外に，イベント間関係の削除，イベントの追加，イベントの削除がある（イベント名の変更もあるが，これは，モデルに対し構造的には変化をもたらさない）．イベントの追加/削除は，これまでに入力している例全体に影響をおよぼすものであり，ツール支援が不可欠と考えているが，具体的な支援方法に関しては，今後の課題とする．

5. おわりに

本報告では,「例からのビジネス・プロセス・モデリング」(PMRbE; Process Modeling and Refinement by Examples) を提唱した．この方法論を定式化する前に，その実現可能性を検討するために，系列事例データの集合と，オブジェクト指向ペトリネットによる簡単なプロセス記述 (モデル）との相互変換を行なうツールを試作している．しかし，モデル生成の機能は，ごく簡単なヒューリスティックスに基づいた限定的なものにとどまっている．

そこで，グラフ列挙アルゴリズムを利用して，より一般的なモデルを生成して，モデリング作業者との間で対話的にモデルを洗練化していくモデル構築支援方法のアプローチを構想した．そこでのグラフ列挙アルゴリズムの役割は，モデルを含むであろうグラフ構造を与えたときに，その部分グラフのうち，モデルが満たすべき条件を備えたものを列挙することである．最終的なモデル構築にはオブジェクト指向ペトリネットを想定しており，オブジェクト指向に基づくモジュール化によって，個々のモデルサイズは比較的小さいことを想定しているとはいえ，高速なグラフ列挙アルゴリズムを必要としており，footprint の網羅的な表による表現にくわえて，BDD(Binary Decision Diagram), ZDD(zero-suppressed decision diagram）ないし ZBDD(zero-suppressed binary decision diagram) の利用を検討している．BDD, ZDD(ないし ZBDD) のソフトウェア実装としては，Python 用の graphillion[1] [4] や Java 用の JavaBDD[2]，JDD[3] などがある．これらライブラリを利用して，対話的なモデリングツールを構築することが次の課題である．

謝辞

本研究は，オブジェクト指向ペトリネットによるビジネスプロセスとセキュリティに関する研究プロジェクトの一環である．同プロジェクトに参加し，その一部としてビジネスプロセスエンジンやビジネスルールの DSL 表現の実装作業も進めてくれている秦 良平，城戸 聡（慶應義塾大学・大学院・理工学研究科・修士課程），ならびに，卒業生の小形 達也 (現在，三菱東京 UFJ 銀行), 渡邊 泰成 (現在，日本 IBM) の各氏に謝意を表する．特に，第 3 節で示したツールの試作は，小形 達也氏が在学中に卒業研究の一部として担当してくれたものである．

参考文献

[1] W. van der Aalst et al.(IEEE プロセス マイニングタスク フォース): “Process Mining Manifesto,” 日本語版 http://www.win.tue.nl/ieeetfpm/lib/exe/fetch.php?media=shared:pmm-japanse-v1.pdf, 初版　Business Process Management Workshops Lecture Notes in Business Information Processing Volume 99, 2012, pp 169-194.

[2] Wil M. P. van der Aalst: “Process Mining: Discovery, Conformance and Enhancement of Business Processes,” Springer, 2011

[3] Rüdiger Valk : “Object Petri nets - Using nets-within-nets paradigm,” LNCS 3098, pp.819–848, Springer, 2004.

[4] ERATO 湊離散構造処理系プロジェクト, 湊真一 (編): 「超高速グラフ列挙アルゴリズム– 〈フカシギの数え方〉が拓く，組合せ問題への新アプローチ」, 森北出版, 2015.

[5] W.M.P. van der Aalst and A.J.M.M. Weijter and L. Maruster : “Workflow Mining: Discovering process models from event logs,” IEEE Transactions on Knowledge and Data Engineering, Vol.16, 2003.

[6] by A. K. A. de Medeiros , B. F. van Dongen , W. M. P. van der Aalst , and A. J. M. M. Weijters “Process mining: Extending the α-algorithm to mine short loops,” BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven, 2004.

[1] https://github.com/takemaru/graphillion

[2] http://javabdd.sourceforge.net/

[3] https://bitbucket.org/vahidi/jdd/wiki/Home