

オブジェクト中心ペトリネットの発見

Wil M. P. van der Aalst

Alessandro Berti

RWTH アーヘン大学プロセス&データサイエンス(PADS)、アーヘン、ドイツ
フラウンホーファー応用情報技術研究所(ドイツ・ザンクトアウグスティン)
wvdaalst,a.berti@pads.rwth-aachen.de

概要. イベントデータからペトリネットを発見する技術は、イベントごとに正確に 1 つのケース識別子を仮定している。そしてこれらのケース識別子はイベントの関連付けに使われ、その結果として発見されたペトリネットは個々のケースのライフサイクルを記述することを目的とする。しかし実際には、ケース概念は一つではなく、複数のケース概念が絡み合っている。例えば、イベントは、注文、アイテム、パッケージ、顧客、製品の混合物を指すことがある。パッケージは、複数のアイテム、複数の製品、1 つの注文、1 つの顧客を参照することができる。したがって、各イベントは、(単一のケース識別子ではなく)それぞれタイプを持つオブジェクトのコレクションを参照していると仮定する必要がある。このようなオブジェクト中心イベントログは、現実の情報システムにおけるデータにより近いものである。我々は、オブジェクト中心イベントログから、オブジェクトの種類に対応する場所と、異なる種類のオブジェクトの集合を消費・生成する可能性のある遷移を持つオブジェクト中心ペトリネットを発見することを目指す。オブジェクト中心ペトリネットは、異なるタイプのオブジェクト間の複雑な関係を可視化するものである。この論文では、PM4Py で実装された新しいプロセス発見アプローチについて説明する。これから示すように、必要に応じて特定の視点にドリルダウンできる全体論的なプロセスモデルを発見することは、実際に可能である。

キーワード: プロセスマイニング、ペトリネット、プロセス発見、多視点モデル

1. はじめに

「低レベル」な動作仕様から「高レベル」なプロセスモデルを合成することについては、これまで数十年にわたり、活発な研究が行われている。このような「高レベル」なプロセスモデルの例としては、(色付き)ペトリネット、BPMN モデル、ステートチャートなどがある。一方、合成の入力となる「低レベル」の動作仕様の例としては、遷移系、言語、部分順序、シナリオなどがある。ペトリネットの文脈では、「領域の理論」が非常に大きな影響力を持っている。領域は最初

の論文[35]で、初等的なネットと遷移系に対して導入された。その目的は、入力として使われる遷移系と同型の到達可能性グラフを持つペトリネットを作ることであった。この核となる考え方は、多くの方向で一般化されてきた。そしてその目的は、入力として用いた遷移系と同型の到達可能性グラフを持つペトリネットを作成することであった。これまで目標モデルの異なるクラスが研究されてきており[15, 16, 32, 43]、例えば、二相似な場所遷移(P/T)ネット[29]、円弧の重みを持つペトリネット[27, 26]、a/sync 接続を持つペトリネット[42]、 τ -ネット[43]、ゼロセーフネット[30]等である。これには一般に、入力として遷移系が用いられる。しかし、言語[17, 51, 18, 33, 20, 54]、部分オーダー/シナリオ[53, 19, 52]、または他の「低レベル」な動作仕様を入力とする様々な領域ベースのアプローチも存在する。

プロセスマイニングは統合の分野(特に言語ベース領域)と関係がある。しかし、その前提や目標は大きく異なる。古典的な統合のアプローチが、「低レベル」の動作仕様の動作をコンパクトに記述した「高レベル」のプロセスモデルを得ることを目的としているのに対し、プロセスマイニング技術はより難しい問題に直面している。プロセス探索の入力として使用されるイベントログには、通常、可能性のある動作のほんの一部しか含まれていない。イベントログに含まれるトレースは、その一例として見ることができる。ループがある場合、すべての可能なトレースを見ることは期待できない。モデルに並行処理がある場合、すべての可能なインターリーブを見ることは期待できない。モデルに複数の選択肢がある場合、すべての可能な組み合わせを目撃することは期待できない。このような設定に対して、領域ベースのアプローチを拡張する多くの試みがなされてきた[25, 17, 61, 63]。残念ながら、領域ベースの手法は計算が困難であり、モデルの過剰適合を招き、スキップや選択と同期の混合(例えば、OR-joins)のようなプロセス構造を発見することができないことが多い。そのため、より拡張的で堅牢な手法がいくつか開発されてきた。市販のツールでは、いわゆる **Directly Follows Graph(DFG)**の学習に頼ることが多いが、これは一般にプロセスモデルの適合度を低下させる[3]。アクティビティが順不同に現れると、ループが作成され、その結果、データによってサポートされていない繰り返しを示唆するスパゲッティのようなダイアグラムになる。プロセスモデルを学習する最先端の技術として、帰納的マイニング技術[45, 46]やいわゆるスプリットマイナー[14]が挙げられる。これらの技術は一般化し、並行性を明らかにすることができる。

本論文では、プロセスディスカバリーに焦点を当てる。しかし、従来のイベントログに対する新しい発見技術を提示するのではなく、オブジェクト中心イベントログ[4]から始める。従来のイベントログでは、各イベントは1つのアクティビティ、1つのタイムスタンプ、1つのケース(すなわち、プロセスインスタンス)に関連している。我々は、各イベントが1つのアクティビティと1つの時点を参照しているという仮定をまだ使用しているが、1つのケースという概念の存在は仮定していない。その代わり、イベントは任意の数のオブジェクトを参照することができ、これらのオブジェクトは異なるタイプである可能性があるのである。これにより、プロセスマイニングの適用範囲が飛躍的に拡大することとなり、これは、P/T(Place Transition)ネットから色付きペトリ(Colored Petri)ネットへの移行に相当する。オブジェクトは色付きのトークンとして見ることができ、オブジェクトのタイプは色の集合(すなわち場所のタイプ)として見る事ができる。

我々はオブジェクト中心イベントログに基づき、オブジェクト中心ペトリネットを自動的に発

見することを目的とする。このようなペトリネットは、イベントログ中のオブジェクトの種類を参照する型付けされた場所を持っている。色付きペトリネットと同様に、遷移は一回の実行中に一つの場所から複数のトークンを消費または生成することができる。本論文では、このようなネットを発見する最初の技術を提示する。関連する研究(セクション 9)では、オブジェクト中心行動制約(OCBC)モデル[49]、同期遷移システム[34, 62]、および作為的結果中心発見アプローチ[37, 38, 56]などの以前のアプローチとの関係を詳しく説明する。

図 1 は、そのアプローチを示したものである。オブジェクト中心イベントログは、あらゆる情報システムから抽出することができる[4]。そしてこれらのログは、今日の情報システムによって収集された実際のデータに近い中間フォーマットと見なすことができる。従来のイベントログ(例: XES ログ)とは異なり、1 つのイベントは複数のオブジェクトを参照することができ、1 つのケースに割り当てることを強制されるものではない。企業情報システム(EIS)、顧客関係管理(CRM)システム、医療情報システム(HIS)、e ラーニングシステム、生産システム、サプライチェーンシステムなどでは、通常、さまざまなオブジェクト(顧客、注文、患者、商品、支払いなどの)の情報が、互いに参照し合う複数のテーブルに格納されている。図 1 は、1 つの「荷物が届いた」イベントに関連するオブジェクトを示している。このイベントは、6 つの注文、9 つの商品、1 つのパッケージ、1 人の顧客、9 つの商品について言及しており、合計で 22,367 個のイベントが存在する。図 1 は、注文、商品、荷物に着目して発見されたオブジェクト中心ペトリネット(読めるようには意図されていない)である。場所と円弧は型付けされている。赤、緑、紫はそれぞれ注文、品目、パッケージを表す。

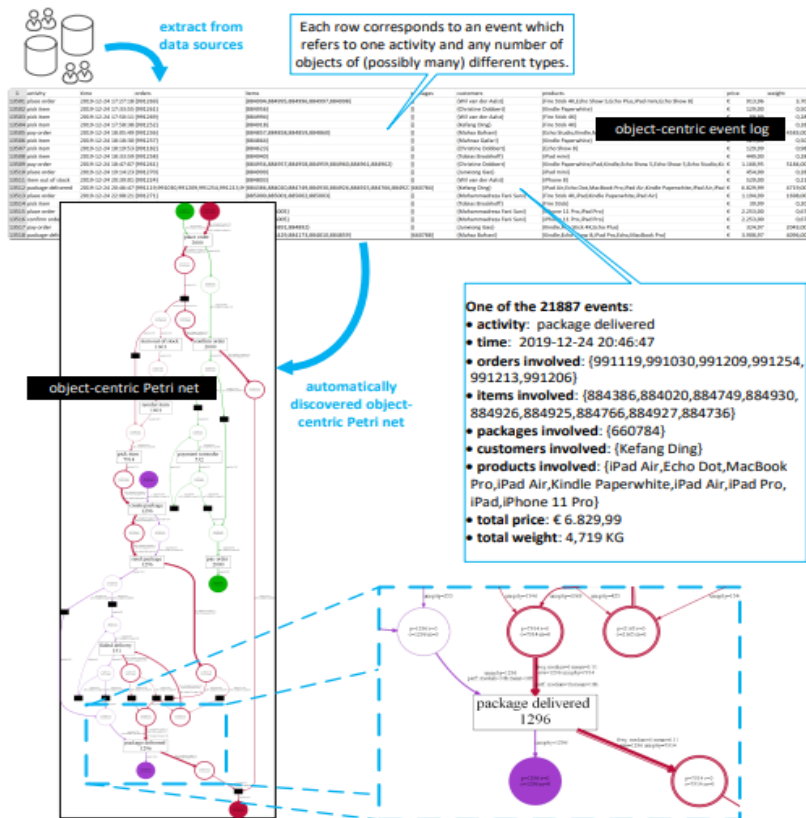


図 1. 本論文で紹介するアプローチの概要である。オブジェクト中心イベントログは、実際のデ

ータソースとシングルケースの概念を必要とする従来のイベントログの間の中間フォーマットとして使用される。この入力を用いて、複数のオブジェクトタイプを一つのモデルで記述できるオブジェクト中心ペトリネットを発見する。

従来のプロセスマイニングのアプローチと同様に、プロセスモデルをフィルタリングし、シームレスに簡略化することが可能である。また、特定のオブジェクトタイプにフォーカスすることで、従来のプロセスマイニング手法で分析可能なイベントログを作成することが出来る。

本論文の残りの部分は以下のように構成されている。第2節では、イベントログとプロセスモデルを紹介する。セクション3では、オブジェクト中心イベントログを紹介し、その動機付けを行う。このようなログが与えられた場合、まずセクション4で、単一のオブジェクトタイプに対するプロセスモデルを学習する技術について議論する。第5節では、オブジェクト中心ペトリネット、すなわち、オブジェクトの種類を参照する場所を持つペトリネットを紹介する。第6節では、本論文の主要な貢献である、オブジェクト中心イベントログからオブジェクト中心ペトリネットを学習するアプローチを紹介する。この発見技術は、**Python** で書かれたオープンソースのプロセスマイニングプラットフォームである **PM4Py** に実装されている。セクション7では実装を紹介し、セクション8ではアプローチの実現可能性を実証する。セクション9では、関連する仕事について議論する。セクション10では、本論文の最後にいくつかのコメントを述べる。

2. 前置き

まず、プロセスマイニングやアクセプタンスペトリネットに馴染みのない人のために、いくつかの前置きを紹介する。プロセスマイニングの入力はイベントログである。伝統的なイベントログは、イベントの関連付けに使われるケース概念によって提供される特定の角度からプロセスを見ることができる。このようなイベントログの各イベントは、(1)特定のプロセスインスタンス(ケースと呼ばれる)、(2)アクティビティ、および(3)タイムスタンプを参照する。リソース、人、コストなどを参照するイベント属性が追加されることもあるが、これは任意である。とはいえこのようなデータは、業務プロセスをサポートするあらゆる情報システムから、多少の努力で抽出することが可能である。プロセスマイニングは、これらのイベントデータを用いて、プロセスに関連する様々な質問に答えるものである。プロセスの発見、適合性チェック、モデルの強化、および運用サポートなどのプロセスマイニング技術は、パフォーマンスとコンプライアンスを向上させるために使用できる[2]。

イベントログの各イベントは、ケース、アクティビティ、タイムスタンプという3つの必須属性を持つ。このうちケースという概念は、同じ注文番号に対応するすべてのイベントを一緒にすること等、イベントをグループ化するために使われる。タイムスタンプはイベントの順序付けに使用され、ボトルネックや遅延の分析に使用すること等が可能である。例えば、コスト、リソース、場所など、多くの属性が追加される可能性がある。しかし、ほとんどのプロセス発見技術は、

まず、ケース内のアクティビティの順序のみが重要となるモデルを学習する。制御フローが明確になれば、モデル上でイベントログを再生することで、他の属性(例えば、時間)を追加することができる[2]。そこで、各ケースのアクティビティの順序のみを記録する、いわゆる「単純なイベントログ」を定義する。技術的には、イベントログはトレースの多重集合である。 $B \subseteq B(X) = X \rightarrow \mathbb{N}$ は、要素 $x \in X$ が $B(x)$ 回出現する X 上の多集合である。例えば、 $B = [a^5, b^2, c]$ において、 a は $B(a)=5$ 回、 b は 2 回、 c は 1 回出現する。

定義 2. 1. (簡易イベントログ)

アクティビティ名の母集団を $\mathbb{U}_{act}$ とする。トレース $\sigma \in \mathbb{U}_{act}^*$ はアクティビティの列である。 $L \in B(\mathbb{U}_{act}^*)$ はイベントログ、即ちトレースの多重集合である。 $\mathbb{U}_{SEL} = B(\mathbb{U}_{act}^*)$ は単純なイベントログの母集団である。

例えば、 $\mathbb{U}_{act} = \{po, pi, sh, in, sr, pa, co, \dots\}$ 、ここで、 po は発注アクティビティ、 pi はピックアップアクティビティ、 sh は出荷アクティビティ、 in は請求書送付アクティビティ、 sr はリマインダー送付アクティビティ、 pa は注文支払いアクティビティ、 co は完了マークアクティビティであることを示している。このようなコンパクトな表記法を用いて、3 つのトレース例： $\sigma_1 = hpo, in, pi, sr, sh, pa, coi$, $\sigma_2 = hpo, pi, sh, in, pa, coi$, $\sigma_3 = hpo, sr, pi, sr, pa, sh, coi$ を示す。当然ながら、複数のケースが同じトレースを持つことができる。イベントログ $L = [\sigma_1^{435}, \sigma_2^{366}, \sigma_3^{233}, \dots]$ において、上記の 3 つのトレースはそれぞれ 435、366、233 回出現する。このようなイベントログが与えられた場合、プロセス発見技術は、観測されたトレースを記述するプロセスモデルを学習することができる。例えば、最も頻度の高いトレースをモデル化し、頻度の低いトレースは抽象化することができる。図 2 は、イベントログ L から発見されたプロセスモデルである。

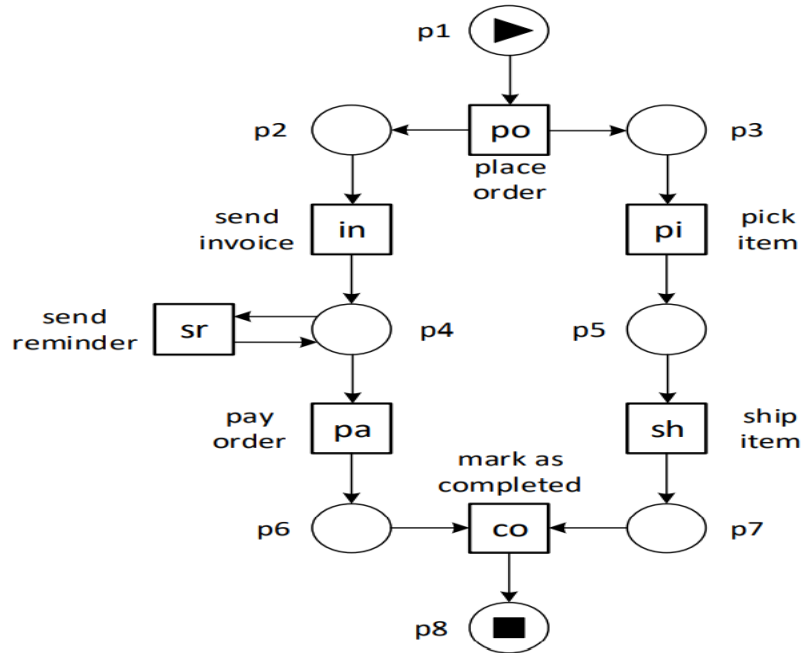


図 2. 8 つの場所と 7 つの遷移からなるアクセプタンスペトリネット。

図2の発見されたプロセスモデルは、遷移がラベル付けされたアクセプタンスペトリネットとして表現される。読者が標準的なペトリネットの表記に慣れていることを前提とするが、重要な概念を明示するために、いくつかの定義を示す。我々は、ラベリング関数と最終マークを持つペトリネットを用いる。これはプロセスマイニングからの要求によるものである。ラベリング関数は、スキップや重複するアクティビティをモデル化するために必要であり、最終マークは、トレースには開始と終了が定義されているため、必要である。

定義 2.2. (ラベル付きペトリネット)

ラベル付きペトリネットは、 P を場所の集合、 T を遷移の集合、 $P \cap T = \emptyset$ 、 $F \subseteq (P \times T) \cup (T \times P)$ をフロー関係、 $l \in T \rightarrow \mathbb{U}_{act}$ をラベル付け関数とするタプル $N = (P, T, F, l)$ である。

ペトリネットはノード $P \cup T$ とエッジ F からなる有向グラフを定義する。ペトリネットの状態はマーキングと呼ばれ、場所の多重集合 ($M \in B(P)$) である。ネット N のマーキング M において、入力場所 $\bullet t = \{p \in P \mid (p, t) \in F\}$ のそれぞれが少なくとも1つのトークンを含んでいれば、遷移 $t \in T$ は有効である。有効な遷移 t は実行される可能性がある、すなわち、入力場所 $\bullet t$ のそれぞれから1つのトークンが除去され、出力場所 $t \bullet = \{p \in P \mid (t, p) \in F\}$ のそれぞれについて1つのトークンが生成される。 $[p1]$ が図2のペトリネットの初期マーキングであるとする。この初期マーキングから到達可能なマーキングは、 $[p1]$ 、 $[p2, p3]$ 、 $[p4, p7]$ 、 $[p8]$ など11個である。

なお、ラベリング関数 l は部分的かつ非射影的であってもよい。つまり、複数の遷移が同じアクティビティを参照する可能性があり、アクティビティに対応しない「沈黙の」遷移が存在する可能性がある。ラベル付きペトリネットの実行シーケンスは、 l を用いて遷移をアクティビティにマッピングすることで得られる可視トレースに対応する。ラベルの付いていないトランジションを実行しても、トレースにアクティビティは追加されない。図2において、すべての遷移は可視的で一意であり、 σ_1 、 σ_2 、 σ_3 は可視トレースの例である(短い名前をアクティビティのラベルと仮定)。

プロセスマイニングでは、初期マーキングと最終マーキングを持つ、いわゆるアクセプタンスペトリネットに着目することが多い。これは、イベントログの生成に用いられたプロセスに対応する言語を定義するモデルを持ちたいからである。

定義 2.3. (アクセプティング・ペトリネット)

アクセプタンスペトリネットは、 $SN = (N, M_{init}, M_{final})$ の三組であり、 $N = (P, T, F, l)$ はラベル付きペトリネット、 $M_{init} \in B(P)$ は初期マーキング、 $M_{final} \in B(P)$ は最終マーキングである。 $\mathbb{U}_{APN}$ は、ペトリネットを受け入れる領域である。

図2では、初期マーク $M_{init} = [p1]$ と最終マーク $M_{final} = [p8]$ をスタートとストップの記号で表している。

定義 2.4. (アクセプタンスペトリネットの言語)

アクセプタンスペトリネット $SN = (N, M_{init}, M_{final})$ は、 M_{init} で始まり M_{final} で終わるすべての可視トレース(ラベルを持たない遷移の発生は無視)からなる言語 $\phi(SN)$ を定義する。

図 2 のアクセプタイズドペトリネットは、ループにより無限に見えるトレース sr を含む。このループがなければ、見えるトレースは 6 つである。

単純なイベントログとペトリネットを受け入れる基本設定を仮定すると、プロセス発見の概念を正式に定義することができる。任意のイベントログに対して、それに対応するプロセスモデルを構築したいと思う。

定義 2. 5. (プロセス発見技法)

発見技術 $disc$ は、単純なイベントログをアクセプタンスペトリネットにマッピングする関数、すなわち $disc \in \mathbb{U}_{SEL} \rightarrow \mathbb{U}_{APN}$ である。

プロセスマイニングを非常に難しくしているのは、イベントログには例の行動しか含まれていないことである。図 2 が実際のプロセスを表している場合、(ループのために)どのイベントログにもその全てのトレースが含まれないという問題がある。ループがない場合でも、選択肢の組み合わせや同時進行するアクティビティのインターリーブにより、現実のプロセスで可能なすべてのトレースを観察することは非常に困難である。一般に、可能なプロセスのほんの一部しか観測されない。さらに、イベントログには、プロセスモデルで終わるべきでないノイズや頻度の低い動作が含まれることがある。このため、再現性(フィットネスとも呼ばれる)、精度、汎化、単純化といった概念が必要になる[2]。これらは本論文の範囲外である。しかし、ここでは、適合性検査という概念を抽象的に定義する。

定義 2. 6. (適合性検査技術)

適合性検査技術 $conf$ は、イベントログと受理ペトリネットからなる組を適合性診断に対応付ける関数、すなわち $conf \in (\mathbb{U}_{SEL} \times \mathbb{U}_{APN}) \rightarrow \mathbb{U}_{diag}$ である。

$conf(L, SN) \in \mathbb{U}_{diag}$ は、想起、精度、汎化、単純化などに関する診断を提供します。例として、イベントログ中のトレースのうち、アクセプタンスペトリネットで再生可能な割合: $conf(L, SN) = |\{\sigma \in L \mid \sigma \in \phi(SN)\}| / |L|$ が挙げられる。図 2 に示す $L' = [\langle hpo, pi, sh, in, pa, co \rangle^8, \langle hpo, pi, sh, in, pa, co \rangle^2]$ と SN を考えると、この適合性の概念から $conf(L', SN) = 0.8$ である。その他にも多くの測定・診断が可能であるが、我々は $\mathbb{U}_{diag}$ を意図的に曖昧にしておく。

3. オブジェクト中心イベントログ

第 2 章では、明確なケース概念があることを前提に、プロセスマイニングの基本的な紹介を行った。本節では、多くのアプリケーションにおいて、この仮定が現実的でないことを示す(セクション 3. 1)。次に、オブジェクト中心イベントログの概念を定式化する(セクション 3. 2)。

3.1. ケース識別子が1つでない場合はどうするか？

多くのアプリケーションでは、複数のケース候補が存在し、同じプロセスに対して異なる見解が示される[4]。さらに、1つのイベントが異なるケースに関連することもある(収束)他、あるケースの中に、同じアクティビティが複数存在することもある(発散)。従来のプロセスモデルを作成するためには、イベントデータを「平坦化」する必要がある。通常、複数の選択肢があり、切断されたり、一貫性のない異なるビューが表示されたりする。

この問題を紹介するために、表1のようなイベントログを考えてみる。この表は、各注文が複数の品目に対応し、別々にピッキングされ、出荷されることを示している。これはより現実的な仮定である(ショップは顧客が1回の注文で複数の商品を購入することを許可する傾向がある)。

表1には、オーダー識別子とアイテム識別子の欄が存在する。オーダー99001は3品目(88124, 88125, 88126)、オーダー99002は2品目(88127, 88128)、オーダー99003は1品目(88129)、オーダー99004は5品目(88130, 88131, 88132, 88133, 88134)に相当している。ピッキングと出荷のアクティビティは、個々の品目に対して実行される。注文は、すべての品目のピッキングと出荷が完了し、注文自体の支払いが完了した時点で「完了」とマークされる。注文99001のイベント `place order` と `mark as completed` は、両方とも4つのオブジェクト(1つの注文と3つの品目)を参照していることには注意が必要である。後者の数は可変的である。例えば、オーダー99003の `place order` というイベントは、2つのオブジェクトだけを参照している。これは先に紹介したアクセプタンスペトリネットでは表現できない。遷移は異なるタイプのトークンを可変個消費し、生成する必要がある。そこで、我々はオブジェクト中心ペトリネットを使うことを提案する。なお、我々はこのようなネットを新しいモデリング言語として提案しているのではない。これは色付きペトリネットのサブクラスと見なすことができるが、我々の焦点は表1のデータを記述するモデルを学習することである。従って、我々はモデリング表記をそのようなデータに対して発見できるものに限定している。

図3は、表1のイベントデータから発見したいオブジェクト中心ペトリネットである。現在、注文に対応する場所(緑色)とアイテムに対応する場所(青色)の2種類がある。遷移は、それが参照するオブジェクトの種類によって色分けされている。遷移 `po` と `co` は2色であることに注意されたい。遷移は、ある場所から複数のトークンを消費したり、ある場所に対して複数のトークンを生成したりすることができる。複数のオブジェクトを消費または生成するイベントに関与する場所と円弧には、これを強調するために複合二重矢印が表示される。図3の遷移 `po` は、場所 `o1` から1つの注文オブジェクトと、場所 `i1` から可変個数のアイテムを消費する。`po` は、場所 `o2` に対して1つの注文オブジェクトを、場所 `i2` に対して可変個数のアイテムを生産する。図3の遷移 `pi` は、場所 `i2` から1つの品目オブジェクトを消費し、場所 `i3` に対して1つの品目オブジェクトを生成する。また、品目は個別に出荷される。しかし、図3の遷移 `co` は、場所 `o4` から1つの注文オブジェクトを消費し、場所 `i4` からの注文に対応するすべてのアイテムを消費する。

表 1. イベントログの断片。各行はイベントに対応し、おそらく複数のオブジェクト(すなわち、注文とアイテム)を参照している。

activity	timestamp	order	item
...	...	...	...
place order	25-11-2019:09.35	{99001}	{88124, 88125, 88126}
pick item	25-11-2019:10.35	$\emptyset$	{88126}
place order	25-11-2019:11.35	{99002}	{88127, 88128}
pick item	26-11-2019:010.25	$\emptyset$	{88124}
send invoice	27-11-2019:08.12	{99001}	$\emptyset$
send invoice	28-11-2019:09.35	{99002}	$\emptyset$
pick item	29-11-2019:09.35	$\emptyset$	{88127}
send reminder	29-11-2019:10.35	{99002}	$\emptyset$
pick item	29-11-2019:11.15	$\emptyset$	{88128}
ship item	29-11-2019:12.35	$\emptyset$	{88124}
pick item	29-11-2019:13.30	$\emptyset$	{88125}
send reminder	29-11-2019:14.35	{99001}	$\emptyset$
ship item	29-11-2019:15.15	$\emptyset$	{88125}
send reminder	29-11-2019:16.15	{99002}	$\emptyset$
ship item	29-11-2019:17.45	$\emptyset$	{88126}
ship item	29-11-2019:18.00	$\emptyset$	{88128}
send reminder	30-11-2019:09.35	{99002}	$\emptyset$
ship item	30-11-2019:10.05	$\emptyset$	{88127}
pay order	30-11-2019:11.45	{99002}	$\emptyset$
pay order	30-11-2019:12.55	{99001}	$\emptyset$
mark as completed	01-12-2019:09.35	{99001}	{88124, 88125, 88126}
place order	02-12-2019:10.40	{99003}	{88129}
mark as completed	04-12-2019:11.05	{99002}	{88127, 88128}
place order	06-12-2019:14.18	{99004}	{88130, 88131, 88132, 88133, 88134}
...	...	...	...

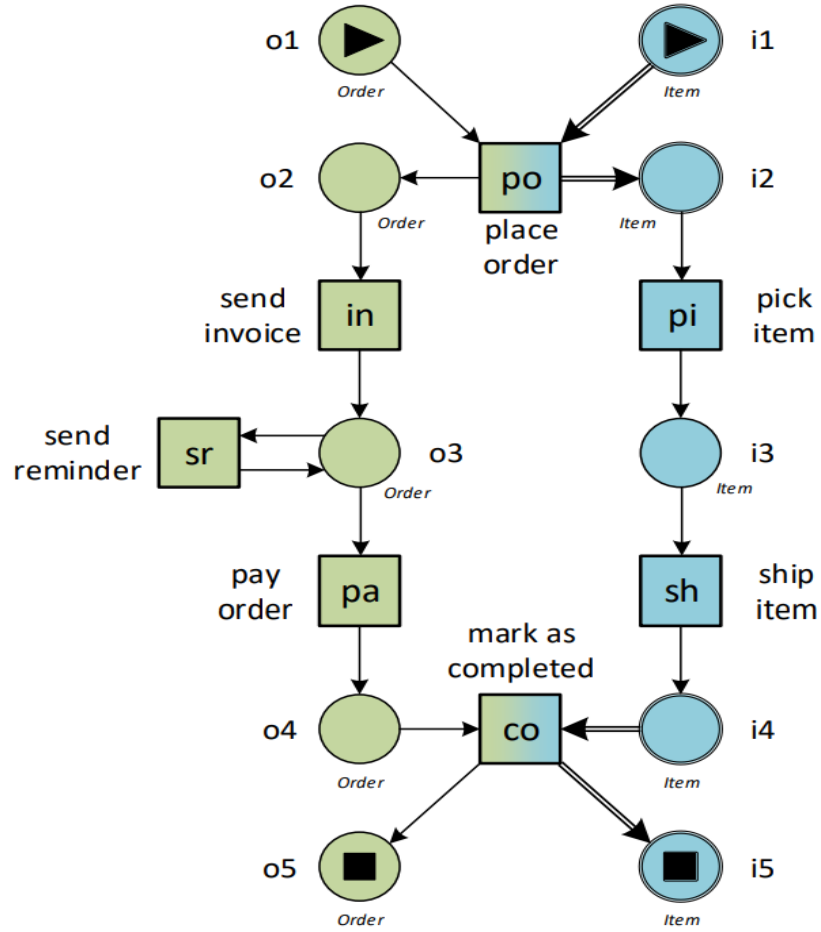


図 3. 2つのオブジェクトタイプを持つオブジェクト中心ペトリネットである、オーダーとアイテム。

表 2. 3 種類のオブジェクトを持つ単純なイベントログの小さな断片。

activity	timestamp	order	item	route
...	...	...	...	...
<i>place order</i>	25-11-2019:09.35	{99001}	{88124, 88125, 88126}	$\emptyset$
<i>place order</i>	25-11-2019:11.35	{99002}	{88127, 88128}	$\emptyset$
...	...	...	...	...
<i>start route</i>	25-11-2019:11.35	$\emptyset$	{88124, 88127}	{66222}
<i>end route</i>	25-11-2019:11.35	$\emptyset$	{88124, 88127}	{66222}
...	...	...	...	...
<i>start route</i>	25-11-2019:11.35	$\emptyset$	{88125, 88126, 88128}	{66223}
<i>end route</i>	25-11-2019:11.35	$\emptyset$	{88125, 88126, 88128}	{66223}
...	...	...	...	...
<i>mark as completed</i>	01-12-2019:09.35	{99001}	{88124, 88125, 88126}	$\emptyset$
<i>mark as completed</i>	04-12-2019:11.05	{99002}	{88127, 88128}	$\emptyset$
...	...	...	...	...

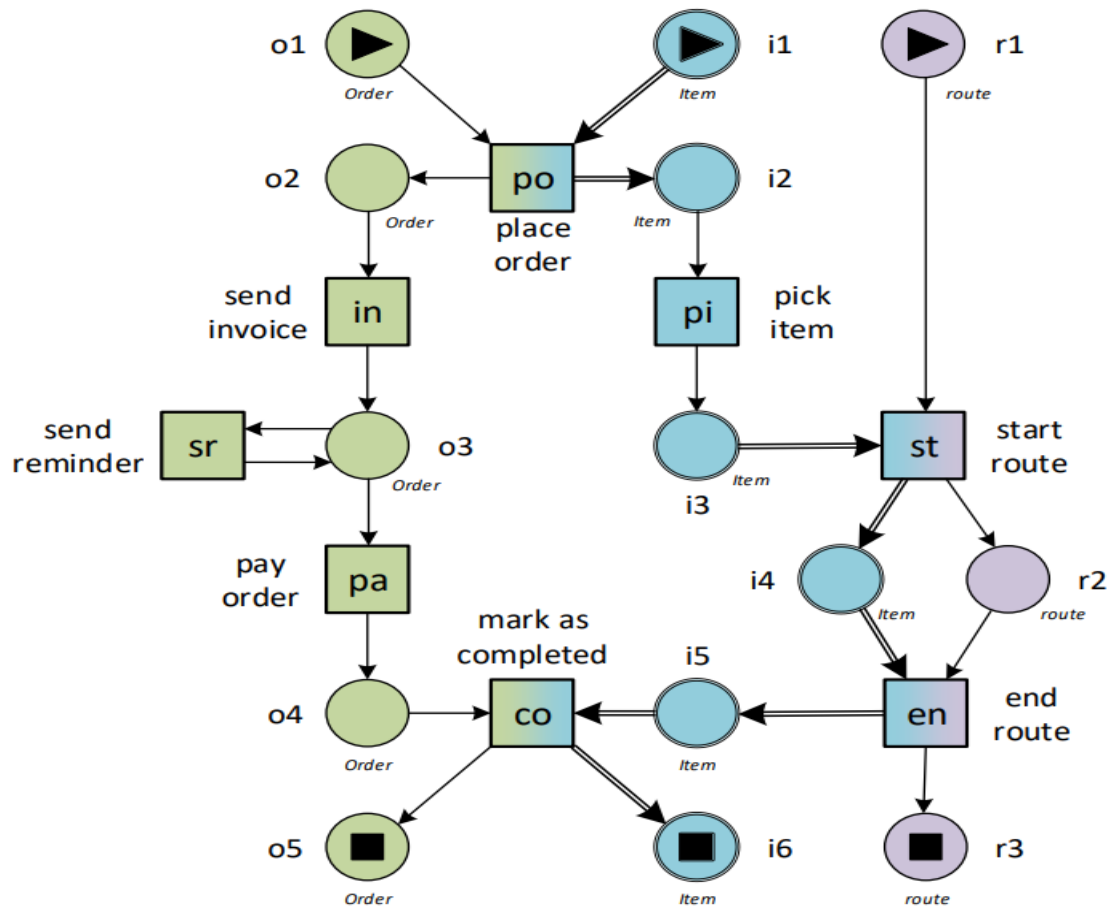


図 4. 3 つのオブジェクトタイプを持つオブジェクト中心ペトリネットである、オーダー、アイテム、ルート。

既存の探索技術では表 1 のイベントデータを扱うことはできないが、それでも注文と商品の間には 1 対多数の関係があるので、これは比較的単純なシナリオである。現実のアプリケーションでは、多数対多数の関係もありうる。これを説明するために、表 2 のイベントログに経路を追加した場合を考えてみよう。どのルートでも、複数の品物が配送される可能性がある。出荷アイテムアクティビティは、異なるオーダーからのアイテムを参照することができる開始ルートと終了ルートのアクティビティに置き換えられた。表 2 に示すように、ルート 66222 は、オーダー 99001 と 99002 に属する 2 つの品目(88124 と 88127)を参照している。ルート 66223 は、オーダー 99001 と 99002 に属する 3 つの品目(88125, 88126, 88128)を参照している。ここでも、単一案件の概念を前提とする従来のプロセスモデルではモデル化できないことは明らかである。

図 4 は、表 2 で参照したイベントログから発見されたオブジェクト中心ペトリネットである。このとき、注文場所(緑色)、品目場所(青色)、経路場所(紫色)の 3 種類の場所が存在する。図 4 の遷移 st は、場所 $i3$ から可変数のアイテム・オブジェクトを、場所 $r1$ から 1 つのルートオブジェクトを消費する。 st は、場所 $i4$ に対して可変数のアイテム・オブジェクトを、場所 $r2$ に対して 1 つのルートオブジェクトを生成する。遷移と場所の色分けと 2 種類の円弧は、イベントログで観察された動作を示している。

問題は、既存のプロセスマイニング技術は、各イベントが正確に 1 つのケースを参照する「フラット化されたイベントログ」を想定していることである。しかし、我々は、図 4 のようなプロセスモデルが欲しいと考えている。SAP、Oracle、Microsoft などの ERP システムにプロセスマイニングを適用すると、すぐに本節で述べたような問題に遭遇する。

3.2. オブジェクト中心イベントログの形式化

表 1 および表 2 は、発見のための入力として使用するデータの種類を示したものである。このようなデータは、情報システムの実データ(例えば、リレーショナルデータベースの複数のテーブル)と eXtensible Event Stream(XES)[40]形式で保存された従来のイベントデータの中間的なものである。XES はイベントごとに 1 つのケース識別子を必要とするが、我々のフォーマットはイベントごとに異なるタイプのオブジェクトをいくつでもサポートする。オブジェクト中心イベントログを定義するために、まず、残りの部分で使用されるいくつかの母集団を定義する([4]に基づいている)。

定義 3.1. (母集団)

本論文で使用する母集団を以下のように定義する：

- U_{ei} はイベント識別子の母集団である。
- U_{act} はアクティビティ名(アクセプタンスペトリネットの遷移のラベル付けにも使われる)の母集団である。

- $\mathbb{U}_{time}$ はタイムスタンプの母集団である。
- $\mathbb{U}_{ot}$ はオブジェクトタイプ(クラスとも呼ばれる)の母集団である。
- $\mathbb{U}_{oi}$ はオブジェクト識別子(実体ともいう)の母集団である。
- $type \in \mathbb{U}_{oi} \rightarrow \mathbb{U}_{ot}$ は、各オブジェクト識別子に正確に1つの型を割り当てている。
- $\mathbb{U}_{omap} = \{omap \in \mathbb{U}_{ot} \rightarrow P(\mathbb{U}_{oi}) \mid \forall ot \in \text{dom}(omap) \forall oi \in \text{omap}(ot) type(oi) = ot\}$ は、タイプごとにどのオブジェクト識別子が含まれるかを示す全オブジェクトマッピングの母集団である¹。
- $\mathbb{U}_{att}$ は属性名の母集団である。
- $\mathbb{U}_{val}$ は属性値の母集団である。
- $\mathbb{U}_{vmap} = \mathbb{U}_{att} \rightarrow \mathbb{U}_{val}$ は値の割り当ての母集団²、および
- $\mathbb{U}_{event} = \mathbb{U}_{ei} \times \mathbb{U}_{act} \times \mathbb{U}_{time} \times \mathbb{U}_{omap} \times \mathbb{U}_{vmap}$ はイベントの母集団である。

イベント $e = (ei, act, time, omap, vmap) \in \mathbb{U}_{event}$ は、一意のイベント識別子 ei 、対応する活動 act 、イベントのタイムスタンプ $time$ 、オブジェクト参照と属性値それぞれのための二つのマッピング $omap$ と $vmap$ で特徴づけられる。

定義 3. 2. (イベントプロジェクション)

$e = (ei, act, time, omap, vmap) \in \mathbb{U}_{event}$, $\pi_{ei}(e) = ei$, $\pi_{act}(e) = act$, $\pi_{time}(e) = time$, $\pi_{omap}(e) = omap$, $\pi_{vmap}(e) = vmap$ が与えられたとき、 $\pi_{ei}(e) = ei$, $\pi_{time}(e) = omap$, $\pi_{omap}(e) = omap$, $\pi_{vmap}(e) = vmap$ となる。

$\pi_{omap}(e) \in \mathbb{U}_{ot} \rightarrow P(\mathbb{U}_{oi})$ は、イベント e のオブジェクト識別子の集合にオブジェクトタイプのサブセットをマッピングする。さらに、 $\pi_{act}(e) = \text{place order}$ 、 $\pi_{time}(e) = 25-11-2019:09.35$ となり、表 2 には属性値が記載されていないため、 $\text{dom}(\pi_{vmap}(e)) = \emptyset$ になる。イベントのコストが 30 ユーロ、場所が Aachen であれば、 $\pi_{vmap}(e)(\text{cost}) = 30$ 、 $\pi_{vmap}(e)(\text{location}) = \text{Aachen}$ となる。

オブジェクト中心イベントログは、部分的に順序付けられたイベントの集合体である。イベント識別子は一意であり、二つのイベントが同じイベント識別子を持つことはない。

¹ $P(\mathbb{U}_{oi})$ はオブジェクト識別子のユニバースのべき集合であり、オブジェクトの型はオブジェクト識別子の集合にマッピングされる。 $omap \in \mathbb{U}_{ot} \rightarrow P(\mathbb{U}_{oi})$ は部分関数である。 $ot \notin \text{dom}(omap)$ ならば、 $omap(ot) = \emptyset$ と仮定する。

² $\mathbb{U}_{att} \rightarrow \mathbb{U}_{val}$ は、属性名の部分集合を対応する値にマッピングするすべての部分関数の集合である。

定義 3.3. (オブジェクト中心イベントログ)

$L = (E, \leq_E)$ は、 $E \subseteq \mathbb{U}_{event}$ 、 $\leq_E \subseteq E \times E$ で、以下のようなイベントログである。

- $\leq_E$ は部分順序(反射的、反対称的、他動的)を定義し、

- $\forall e_1, e_2 \in E \pi_{ei}(e_1) = \pi_{ei}(e_2) \Rightarrow e_1 = e_2$, かつ

- $\forall e_1, e_2 \in E e_1 \leq_E e_2 \Rightarrow \pi_{time}(e_1) \leq \pi_{time}(e_2)$ となる。

定義 3.3 では、イベントログの部分的な順序付けが可能である。しかし、実際には、イベントはタイムスタンプに基づいて順序付けされ、2 つのイベントが同じタイムスタンプを持つとき、何らかの順序を仮定するなど、総体的な順序を使用することが多い。以前使用していた表形式(例: 表 2)でも、合計順を使用せざるを得なかった。しかし、因果関係を考慮したプロセス発見技術が存在する[4, 55]。これらは、このような部分的な順序を利用することができる。

4. 単一オブジェクト型に対するペトリネットの発見

オブジェクト中心イベントログは、各イベントが正確に 1 つのケース識別子を持つという、従来のイベントログの概念を一般化する。このようなログは、任意のイベント $e \in E$ に対して $|\pi_{omap}(e)(case)| = 1$ であるような特別なオブジェクトタイプ $case \in \mathbb{U}_{ot}$ を用いて模倣できる。従来のプロセスマイニング技術はこれを前提としているので、イベントデータを「平坦化」して可変数のオブジェクトを参照するイベントを持つ古典的イベントログに変換することはよくあることである。特定のオブジェクトの種類をケース識別子と仮定する。あるイベントがそのタイプの複数のオブジェクトを持つ場合、各オブジェクトに対して 1 つのイベントを作成するだけでよい。もし、あるイベントがそのタイプのオブジェクトを持たなければ、単にイベントを省略する。もしイベントが選択されたタイプのオブジェクトを正確に 1 つだけ持っていれば、そのイベントを保持する。これは次のように定式化できる。

定義 4.1. (イベントログの平坦化)

$L = (E, \leq_E)$ をオブジェクト中心イベントログ、 $ot \in \mathbb{U}_{ot}$ を格概念として機能するオブジェクト型とすると、 $L = (E, \leq_E)$ はオブジェクト中心イベントログである。平坦化されたイベントログは $L^{ot} = (E^{ot}, \leq_E^{ot})$ で³ :

- $e_i =$ 任意の $e \in E$ と $I \in \pi_{omap}(e)(ot)$ に対する $((\pi_{ei}(e), i), \pi_{act}(e), \pi_{time}(e), \pi_{omap}(e) \oplus (case, \{i\}), \pi_{vmap}(e))$ と

³ $f' = f \oplus (x, y)$ は $z \in \text{dom}(f) \setminus \{x\}$ に対して $\text{dom}(f') = \text{dom}(f) \cup \{x\}$ 、 $f'(x) = y$ 、 $f'(z) = f(z)$ となるような関数である。

- $E^{ot} = \{e_i \mid e \in E \wedge i \in \pi_{omap}(e)(ot)\}$ 、および
- $\leq_E^{ot} = \{(e'_i, e''_j) \in E^{ot} \times E^{ot} \mid e' \in E \wedge i \in \pi_{omap}(e')(ot) \wedge e'' \in E \wedge j \in \pi_{omap}(e'')(ot) \wedge e' \leq_E e'' \wedge (e' = e'' \Rightarrow i = j)\}$ とした。

平坦化されたイベントログは、イベントを削除し、複製してもイベントログのままである。

補題 4.2. $L=(E, \leq_E)$ をオブジェクト中心イベントログとし、 $ot \in \mathcal{U}_{ot}$ をケース概念として機能するオブジェクト型とする。平坦化されたイベントログ $L^{ot} = (E^{ot}, \leq_E^{ot})$ は、確かに定義 3.3 で定義されたようにイベントログである。

証明：

$\leq_E^{ot}$ は部分順序を定義する。任意の $e_i \in E^{ot}$ に対して、 $e_i \leq_E^{ot} e_i$ (反射的)。 $e'_i \leq_E^{ot} e'_j$ と $e'_j \leq_E^{ot} e'_i$ ならば、 $e' = e''$ かつ $i = j$ であり、したがって $e'_i = e'_j$ でもある (反対称的)。 $e'_i \leq_E^{ot} e'_j$ と $e'_j \leq_E^{ot} e'_k$ ならば、 $e' \leq_E e''$ 、 $e'' \leq_E e'''$ 、 $(e' = e'' \Rightarrow i = j)$ 、 $(e'' = e''' \Rightarrow j = k)$ となる。したがって、 $e' \leq_E e'''$ となる ($\leq_E$ は推移的)。 $e' \neq e'''$ ならば、 $\leq_E^{ot}$ の定義により、 $e'_i \leq_E^{ot} e'_k$ となる。 $e' = e'''$ ならば、 $e' = e''$ 、 $e'' = e'''$ となる。したがって、 $i = j$ 、 $j = k$ (前述)、 $i = k$ となり、ここでも $e'_i \leq_E^{ot} e'_k$ (推移的) と結論される。 $\pi_{ei}(e'_i) = \pi_{ei}(e'_j)$ ならば、 $(e', i) = (e'', j)$ でイベント識別子は一意となる。もし、 $e'_i \leq_E^{ot} e'_j$ ならば、 $e' \leq_E e''$ となる。したがって、 $\pi_{time}(e'_i) = \pi_{time}(e') \leq \pi_{time}(e'') = \pi_{time}(e'_j)$ であり、時間は逆行できないことがわかる。

表 2 には 8 つの事象が示されている (残りは省略)。この 8 つのイベントのみからなるログを $L = (E, \leq_E)$ とする。平坦化されたイベントログ $L^{Order} = (E^{Order}, \leq_E^{Order})$ は 4 つのイベントを持つ (表 2 の真ん中の 4 つのイベントは削除される)。平坦化イベントログ $L^{Item} = (E^{Item}, \leq_E^{Item})$ は、元のイベントが全て 2~3 回複製されるため、20 イベントとなる。平坦化されたイベントログ $L^{Route} = (E^{Route}, \leq_E^{Route})$ は、4 つのイベントを持つ。

ここで $L = (E, \leq_E)$ を、オブジェクト型 ot を使って平坦化し、イベントログ $L^{ot} = (E^{ot}, \leq_E^{ot})$ を作成したと仮定する。この場合、ケース概念を選択した従来のイベントログができ、既存のプロセスマイニング技術をすべて適用することができる。しかし、ケース概念として ot を用いてイベントログを平坦化すると、以下のような問題が発生する可能性がある。

- 欠陥：元のイベントログのうち、平坦化後のイベントログに対応するイベントがないものは、データセットから消滅する (すなわち、 $\pi_{omap}(e)(ot) = \emptyset$)。
- 収束：選択されたタイプの複数のオブジェクトを参照するイベントが複製され、意図しない重複が発生する可能性がある (つまり、 $|\pi_{omap}(e)(ot)| \geq 2$)。

- ・発散：格概念として選択されていないタイプの異なる対象を参照する事象は、因果関係があるとみなされる。例えば、2つの事象が同じ順序だが異なる時間を指している場合や、2つの事象が同じ経路だが異なる項目を指している場合などである。

定義 4.3. (欠損、収束、発散)

$L = (E, \leq_E)$ をオブジェクト中心イベントログ、 $L^{ot} = (E^{ot}, \leq_E^{ot})$ をオブジェクトタイプ $ot \in \mathbb{U}_{ot}$ に基づいて平坦化されたイベントログとする。イベント $e(E)$ は、 $\pi_{omap}(e)(ot) = \emptyset$ (すなわち、 ot をケース概念として使用する場合、イベントは無視される) の場合、欠陥問題を有する。イベント $e \in E$ は、 $|\pi_{omap}(e)(ot)| \geq 2$ であれば収束問題がある(つまり、 ot を格概念として用いる場合にイベントが意図せずに複製される)。イベント $e \in E$ は、 $\pi_{omap}(e)(ot) \neq \emptyset$ となるような別のイベント $e' \in E$ とオブジェクトタイプ $ot' \in \mathbb{U}_{ot}$ のような $\pi_{omap}(e)(ot) \neq \emptyset$ 、 $\pi_{omap}(e')(ot) \neq \emptyset$ 、 $\pi_{omap}(e)(ot') \neq \emptyset$ 、 $\pi_{omap}(e')(ot') \neq \emptyset$ 、 $\pi_{omap}(e)(ot) = \pi_{omap}(e')(ot)$ 、 $\pi_{omap}(e)(ot') \neq \pi_{omap}(e')(ot')$ が存在する場合、発散問題が発生する。

発散の場合、二つの事象 e と e' 、二つの候補格概念 ot と ot' があり、両方の事象が両方のオブジェクト型のオブジェクトを指し、事象は ot については「同意する」が ot' については「同意しない」ことに注意されたい。

表 2 に示した 8 つのイベントをもう一度考えてみよう。イベントログを平坦化するためのオブジェクトタイプとして **Order** や **Route** を取ると、半分のイベントが消滅する(欠陥)。イベントログを平坦化するオブジェクトタイプを **Item** とした場合、最初のイベントは 3 つの **place order** イベントに、2 番目のイベントは 2 つの **place order** イベントに置き換わる。このように複製されたイベントは一度しか発生しないため、誤解を招く恐れがある(収束)。発散を説明するために、1 つの注文が 10 個のアイテムからなり、イベントログを平坦化するためにオブジェクトタイプ **Order** が使用されていると仮定する。そうすると、ある順番で実行されるピックイベントが 10 個あることになる。これらは独立しているが、因果関係があるように見える(同じケース)ので、ほとんどの発見アルゴリズムではループが発生する。

図 5 は、平坦化された 3 つのイベントログから発見された 3 つのプロセスモデル： $L^{Order} = (E^{Order}, \leq_E^{Order})$ 、 $L^{Item} = (E^{Item}, \leq_E^{Item})$ 、 $L^{Route} = (E^{Route}, \leq_E^{Route})$ である。例えば、真ん中のアクセプタンスペトリネットは、 L^{Item} 、すなわち、オブジェクト型 **Item** を使って平坦化された元のイベントログを基に発見された。まず、100 個の注文があり、1 個の注文につき平均 5 個の品物があると仮定すると、これは、500 個のアイテムがあることを意味する。次に各ルートは平均して 50 個の商品を配送する必要がある、全部で 10 ルートあると仮定し、この数字を図 5 に示す。3 つのペトリネットは合理的に見えるが、実際には「適合」していない(対応するアクティビティの頻度が異なっている)。例えば、左のモデル(注文)では、場所注文のアクティビティが 100 回実行され、中央のモデル(品目)では、同じアクティビティが 500 回実行されている(要因 5)。右のモデル(ルート)では、ルート開始アクティビティが 10 回実行され、真ん中のモデル(アイテム)では、同じアクティビティが 500 回実行される(要因 50)。このようなミスマッチが収束の問題を

示している。アクティビティの頻度が元のプロセスモデルの頻度と一致しないので、真ん中のアクセプタンスペトリネットは間違っていると主張することができる。

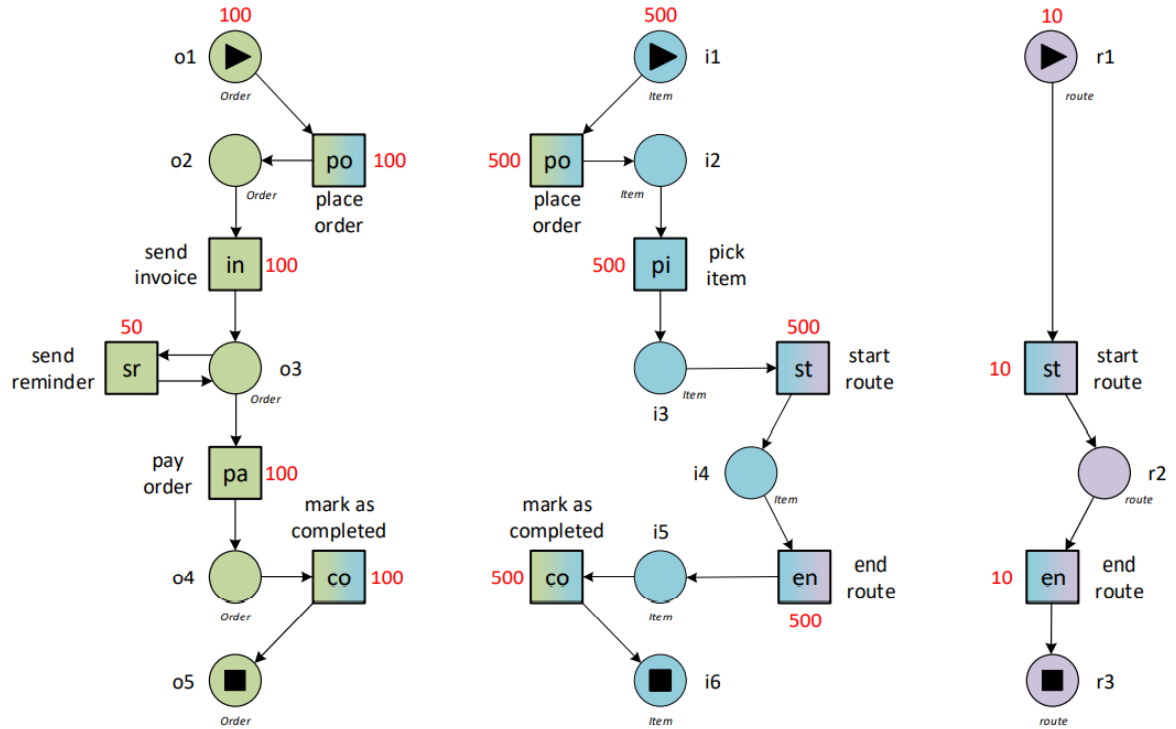


図 5. 平坦化された 3 つのイベントログに対して発見された 3 つのアクセプタンスペトリネット : $(E^{Order}, \leq_E^{Order})$ (左)、 $(E^{Item}, \leq_E^{Item})$ (中)、 $(E^{Route}, \leq_E^{Route})$ (右)である。赤の数字は、円弧ごとに生成または消費されたトークンの総数である。

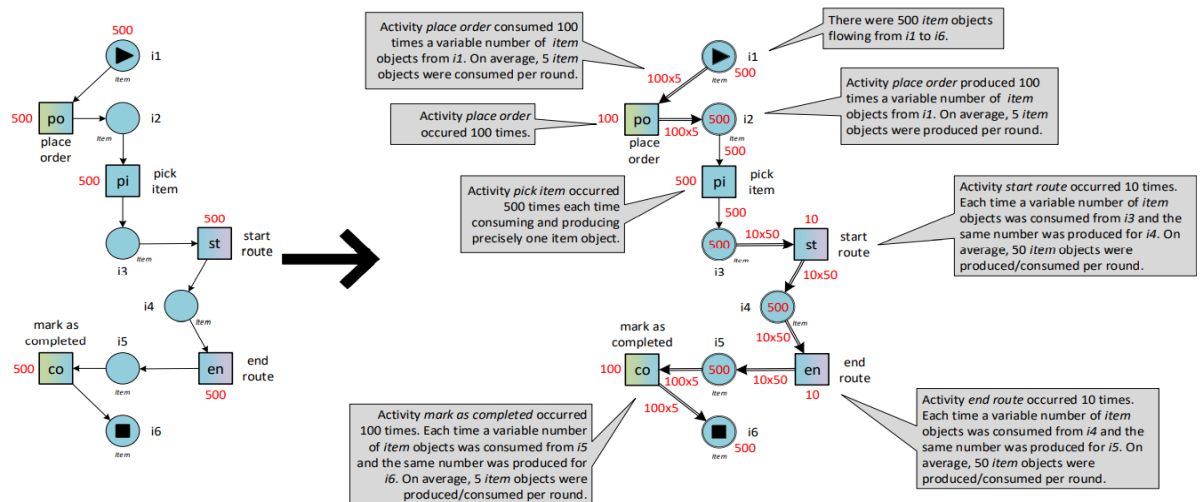


図 6. 左のモデルは、 $L^{Item} = (E^{Item}, \leq_E^{Item})$ について発見されたものである。平坦化されている

ため、アクティビティの頻度は正しくない。しかし、どの遷移の発生が各イベントに属するかは分かっているので、それらを再グループ化することができる。これを利用して、発生を統合することで、右のようなプロセスモデルを導くことができる。

図 6 は、可変的な円弧、すなわち、1 ステップで複数のトークンを消費または生成するために使用できる円弧を使用して、不正なアクティビティ頻度の問題を解決する方法をスケッチしたものである。このような「多重集合円弧」は色付きペトリネットでも可能である[12, 41]。あるイベントが「平坦モデル」(例えば図 5)を生成するために複製されたとき、対応する遷移の発生を、複数のトークンを消費したり生成したりすることができる 1 つの遷移の発生に統合することができる。例えば、遷移 `place order` を参照されたい。左のアクセプタンスペトリネットでは、平坦化されたイベントログ L^{Item} を再生すると、遷移 `place order` が 500 回発生する。しかし、我々はどの遷移と一緒に発生するのかを正確に知っている。

このことを利用して、1 ステップで可変数のトークンを消費・生成する遷移を再構成することができる。このような場合、500 回の発生が 100 回の発生に統合され、平均して 1 つの円弧につき 5 個のトークンが消費され、生成されることになる。これを示すために、 100×5 というタグのついた複合二重矢印を使用する。次に、遷移開始経路を考える。左のモデルでは、遷移開始ルートは 500 回発生する。しかし、この 500 回の遷移のうち、どれが 10 個のルートに属するかは正確に分かっている。このような低レベルの遷移発生は、1 ステップで可変数のトークンを消費・生成する高レベルの遷移発生に統合することができる。これは、遷移開始ルートでは、平均して 1 円弧あたり 50 トークンを消費して生成する 10 回の発生があることを意味する。このことを示すために、再び複合二重矢印を使用するが、今度は 10×50 というタグを付ける。また、`pick item` の出現頻度だけが平坦化によって変化しなかった。このため、対応する円弧も変化していない。

図 6 は、1 つのオブジェクトタイプについて、各遷移の頻度がイベントログの実際の対応するイベントの数と一致するペトリネットを作成する方法を示したものである。これらのモデルは、次に示すように、異なるオブジェクトタイプを示すより全体的なプロセスモデルに統合することができます。

5. オブジェクト中心ペトリネット

前節で示したように、我々は異なるオブジェクトタイプを区別できる必要があり、1 つのイベント(すなわち遷移の発生)が可変数のオブジェクトを含む場合がある(例えば、1 つの注文は任意の数の項目を持つ場合がある)。このようなプロセスをモデル化する明白な方法は、場所が異なるタイプを持つことができる色付きペトリネットを使用することである[12, 41]。図 7 は、前述の 100 オーダー、500 アイテム、10 経路のシナリオをシミュレートしているときの CPN Tools のスクリーンショットである。オーダー、アイテム、ルートというカラーセットが場所のタイプに

使用されており、**or** のタグが付いた 10 個の円弧は、1 つのオーダーを生成または消費する。タグ **it** を持つ 2 つの円弧は、1 つの品目を生成または消費する。**rt** というタグを持つ 4 つの円弧は、1 つのルートを生成または消費する。タグ **its** を持つ 8 つの円弧があり、これは **Items** タイプの変数、つまりアイテムのリストである。これらは、可変数のアイテム・オブジェクトを消費するか、生成する。4 つの条件式は、オーダー、ルート、およびアイテムの間の対応を決定する。例えば、transition place **order** の $[its = oi(or)]$ という条件式は、特定の順序 **or** に関与するアイテム **its** の集合を指定する。同じ条件式は、遷移の完了マークにも使われる。トランジション「**start route**」と「**end route**」は条件式 $[its = ri(rt)]$ を使って、ルート **rt** に関係するアイテムを決定する。

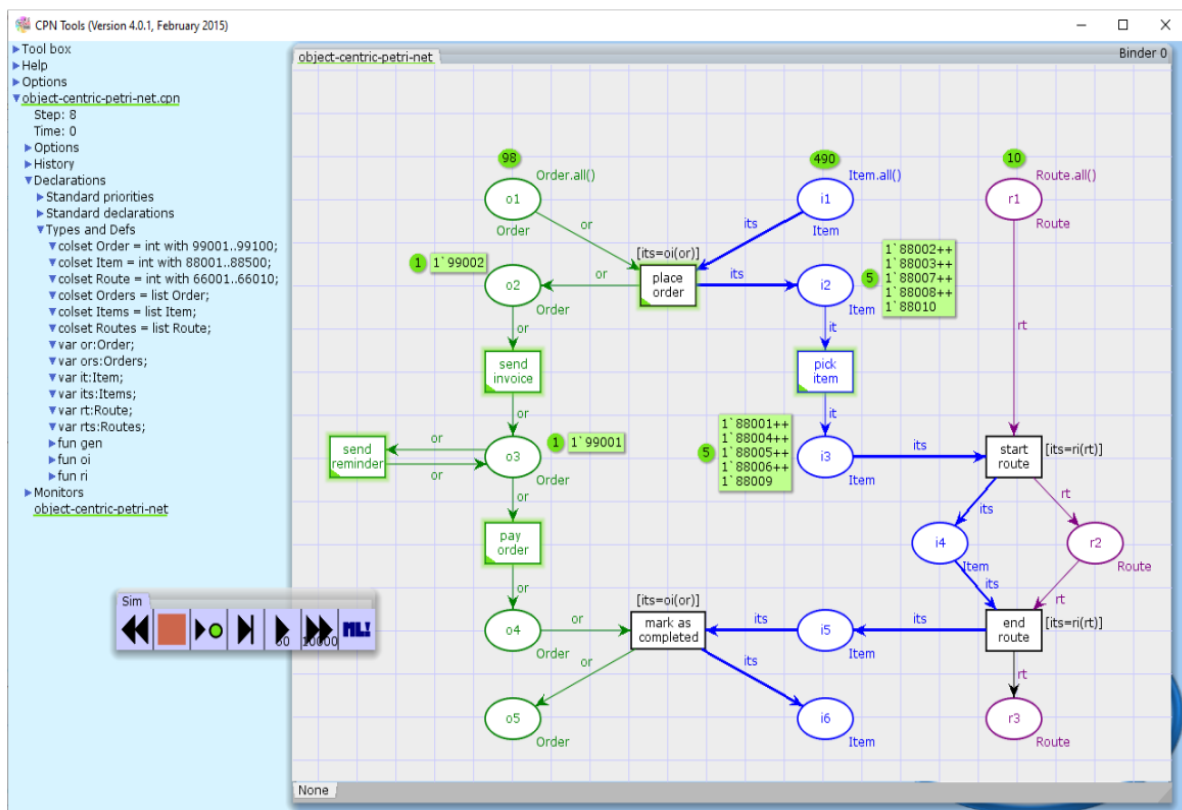


図 7. 表 2 のイベントデータから発見された図 4 のプロセスをモデル化した CPN Tools[12, 41] の色付きペトリネット。

図 7 は、複数のオブジェクトが関与するプロセスを色付きペトリネット(またはその関連形式)でモデル化できることを示している。しかし、(オブジェクト中心の)イベントログから任意の色のペトリネットを発見することは不可能である。イベントログの情報に対応した表現バイアスが必要である。そこで、我々は特定のタイプの色付きペトリネットを発見することを目的とする。また、問題を単純化するために、異なるオブジェクト間のマッチング(すなわち、図 7 の条件式)を抽象化する。これにより、オブジェクト中心ペトリネットと呼ばれる、より具体的でより抽象

的な表現を用いることができる⁴。

定義 5.1. (オブジェクト中心ペトリネット)

ここで、 $N = (P, T, F, l)$ はラベル付きペトリネット、 $pt \in P \rightarrow \mathbb{U}_{ot}$ は場所をオブジェクト型に写すもの、 $F_{var} \subseteq F$ は可変円弧の部分集合である。

図 4 はオブジェクト中心ペトリネット： $P = \{o1, \dots, o5, i1, \dots, i6, r1, r2, r3\}$ 、 $T = \{po, in, pi, \dots\}$ 、 $F = \{(o1, po), (i1, po), (po, o2), (po, i2), \dots\}$ 、 $l(po)$ =注文、 $l(in)$ =請求書の送信などや $pt(o1)$ = Order、 $pt(i1)$ = Item、 $pt(r1)$ = Route など、 $F_{var} = \{(i1, po), (po, i2), \dots\}$ である。図 4 のグラフ表記は、オブジェクト中心ペトリネットを完全に定義していることに注意されたい。

定義 5.2. (整形式)

$ON = (N, pt, F_{var})$ を $N = (P, T, F, l)$ のオブジェクト中心ペトリネットとする。ここで次のような表記がある。

- $pl(t) = \bullet t \cup t \bullet$ は $t \in T$ の入出力場所、 $pl_{var}(t) = \{p \in P \mid \{(p, t) \mid (t, p)\} \cap F_{var} \neq \emptyset\}$ は変数円弧で接続された入出力場所、 $pl_{nv}(t) = \{p \in P \mid \{(p, t), (t, p)\} \cap (F \setminus F_{var}) = \emptyset\}$ は非可変円弧で接続された場所である。
- $tpl(t) = \{pt(p) \mid p \in pl(t)\}$ 、 $tpl_{var}(t) = \{pt(p) \mid p \in pl_{var}(t)\}$ 、 $tpl_{nv}(t) = \{pt(p) \mid p \in pl_{nv}(t)\}$ は対応する場所のタイプである。

ON は、各遷移 $t \in T$ ： $tpl_{var}(t) \cap tpl_{nv}(t) = \emptyset$ のとき、整形式である。

よくできたオブジェクト中心ペトリネットでは、円弧は可変性について「同意」するはずである、すなわちオブジェクトタイプと遷移の組み合わせは、可変円弧か通常円弧を持つが、その両方は持たないということである。例えば、 $(i1, po) \in F_{var}$ もまた $(po, i2) \in F_{var}$ である。なぜなら $(o1, po) \notin F_{var}$ もまた $(po, o2) \notin F_{var}$ であるから。この仮定は、オブジェクト中心イベントログを見るときに合理的である。イベント e とオブジェクトタイプ ot につき、 $\pi_{omap}(e)(ot)$ が与えられる。したがって、遷移 t ごとに同じ型 ot のオブジェクトの異なるセットを考慮することは意味がない。残りの部分では、我々は(明示的にこれを表明することなく)良好なオブジェクト中心ペトリネットに自分自身を制限する。

(p, oi) で示されるトークンは、場所 p に存在し、オブジェクト oi を参照する。マーキングとは、このようなトークンの多重集合のことである。マーキング $[(p1, 666), (p2, 666), (p2, 555), (p3, 555)]$ には 4 つのトークンがある(場所 $p2$ にはオブジェクト 555 と 666 を指すトークンが 2 つあ

⁴ オブジェクト・ペトリネットに似た用語は、Rudiger Valk, Charles Lakos, Jinzhong Niu, Li-Chi Wang, Daniel Moldt らによってすでに使用されていた。私たちのネットは異なるので、用語のオーバーロードは避けられないことに注意されたい。

る)。

定義 5.3. (マーキング)

$ON = (N, pt, F_{var})$ を $N = (P, T, F, l)$ のオブジェクト中心ペトリネットとする。 $Q_{ON} = \{(p, oi) \in P \times \mathbb{U}_{oi} \mid \text{type}(oi) = \text{pt}(p)\}$ は可能なトークンの集合である。 ON のマーキング M はトークンの多集合、すなわち、 $M \in B(Q_{ON})$ である。

オブジェクト中心ペトリネットの意味論を記述するために、色付きペトリネットにおける結合の概念と同様に、結合の概念を使用する。ただし、現在、結合はイベントログ中の対応するイベントのオブジェクト参照を参照する。結合 (t, b) は遷移 t と、オブジェクト型の部分集合をオブジェクト識別子の集合にマッピングする関数 b を指す。オブジェクトタイプのサブセットは周囲の場所のオブジェクトタイプに対応する(すなわち、 $\text{tpl}(t)$)。さらに、非可変円弧に対して、結合は正確に 1 つのオブジェクトを選択すべきである(すなわち、 $ot \in \text{tpl}_{nv}(t)$ に対して $|b(ot)| = 1$)。遷移 t とその入力場所 p の 1 つ(すなわち、 $p \in \bullet t$) を考える。もし t が結合 (t, b) で発生すれば、 $\text{pt}(p) \in \text{dom}(b)$ とオブジェクト $b(\text{pt}(p))$ は入力場所 p から削除される。もし p が t の出力場所($p \in t \bullet$) であれば、オブジェクト $b(\text{pt}(p))$ は出力場所 p に加えられる。したがって結合 (t, b) は新しい印を完全に決定する。

定義 5.4. (結合実行)

$ON = (N, pt, F_{var})$ を $N = (P, T, F, l)$ とするオブジェクト中心ペトリネットとする。 $B = \{(t, b) \in T \times \mathbb{U}_{omap} \mid \text{dom}(b) = \text{tpl}(t) \wedge \forall_{ot \in \text{tpl}_{nv}(t)} |b(ot)| = 1\}$ はすべての可能な結合のセットである。 $(t, b) \in B$ は結合で、入力場所からの選択オブジェクトを消費し出力場所(両方とも b で指定)の対応オブジェクトを生成する移行 t の実行に相当する。 $\text{cons}(t, b) = \{(p, oi) \in Q_{ON} \mid p \in \bullet t \wedge oi \in b(\text{pt}(p))\}$ は、結合 (t, b) が与えられたときに消費されるトークンの多集合である。 $\text{prod}(t, b) = \{(p, oi) \in Q_{ON} \mid p \in t \bullet \wedge oi \in b(\text{pt}(p))\}$ は、結合 (t, b) が与えられたときに生成されるべきトークンのマルチセットである。結合 (t, b) は、 $\text{cons}(t, b) \leq M$ であれば、マーキング $M \in B(Q_{ON})$ において有効であり、マーキング M において有効な結合 (t, b) の発生は、新しいマーキング $M' = M - \text{cons}(t, b) + \text{prod}(t, b)$ につながる⁵。これを $M \xrightarrow{(t,b)} M'$ と表記する。

$M \xrightarrow{(t,b)} M'$ は、マーキング M において結合 (t, b) が有効であり、この結合の発生が新しいマーキング M' につながることを意味する。また、有効な結合のシーケンス $\sigma = \langle (t_1, b_1), (t_2, b_2), \dots, (t_n, b_n) \rangle \in B^*$ すなわち $M_0 \xrightarrow{(t_1, b_1)} M_1 \xrightarrow{(t_2, b_2)} M_2 \xrightarrow{(t_3, b_3)} \dots \xrightarrow{(t_n, b_n)} M_n$ となるような有効な結合の列を持つことも可能である。すなわち、 M_0 から M_n に n ステップで到達することが可能である。これを $M \xrightarrow{\sigma} M'$ とする。また、遷移名に対応するアクティビティ名に 1 で対応させることも可能であり、いわゆる可視的な結合シーケンス $\sigma_v = \langle (l(t_1), b_1), (l(t_2), b_2), \dots, (l(t_n), b_n)) \rangle$ につながる(ここで、 t_i にラベ

⁵ 和(+)、差(-)、包含($\leq$)は多集合に対して通常の方法で定義され、例えば $[a, b] + [b, c] = [a, b^2, c]$ 、 $[a, b^2, c] - [b, c] = [a, b]$ 、 $[a, b] \leq [a, b^2, c]$ のように定義される。

ルがない場合は $(l(t_i), b_i)$ は省略される)。なお、可視化された結合シーケンスでは、サイレントステップ(ラベルのない遷移)は表示されず、重複したアクティビティ(同じラベルを持つ2つの遷移)は区別できない。

定義 5.4 は、非可変円弧に対して正確に1つのトークンが消費/生成されること以外、結合にいかなる制約も加えていないことに注意すべきである。図7の色付きペトリでは、アイテムを特定の順序や経路にリンクするための条件式を持つ4つの遷移がある。これは定義5.1において意図的に抽象化され、初期および最終マークを持ついわゆるアクセプタンスオブジェクト中心ペトリネットの発見を可能にするものである。

定義 5.5. (アクセプタンスオブジェクト中心ペトリネット)

アクセプタンスオブジェクト中心ペトリネットとは、整形されたオブジェクト中心ペトリネット $ON = (N, pt, F_{var})$ 、初期マーキング $M_{init} \in B(Q_{ON})$ 、最終マーキング $M_{final} \in B(Q_{ON})$ からなるタプル $AN = (ON, M_{init}, M_{final})$ のことである。

可視的な結合シーケンスの概念を用いると、初期マーキングから最終マーキングに至るすべての動作について推論することができる。

定義 5.6. (オブジェクト中心ペトリネットの言語)

アクセプタンスオブジェクト中心ペトリネット $AN = (ON, M_{init}, M_{final})$ は、 M_{init} で始まり、 M_{final} で終わるすべての可視的結合系列からなる言語 $\phi(AN) = \{\sigma_v \mid M_{init} \xrightarrow{\sigma} M_{final}\}$ を定義する。

オブジェクト中心ペトリネットの動作は、意図的に「未定義」であることに注意されたい。型式とカーディナリティの制約があるだけである。従って、異なる型のオブジェクトは無関係である。図7の色付きペトリネットと比較すると、我々のプロセスモデルは、異なる型のオブジェクトを関連付けるために条件式を使用しない。条件式は、識別子によってのみ特徴付けられる異なる型のオブジェクトを結合することに注意されたい。識別子だけを使ってしまうと、モデルが過剰適合してしまう。オーダー99001がアイテム88124、88125、88126で構成されていることを伝えるルールを見つけるにはどうすればよい？これはデータに含まれるものであり、正確で明示的なルールでは扱えない。結論で述べたように、これは今後の研究課題である(第10節参照)。

6. オブジェクト中心ペトリネットの発見

まず、オブジェクト中心イベントログからオブジェクト中心ペトリネットを学習するための一般的なアプローチを紹介する。次に、モデルの性能関連のアノテーション、モデルビュー、および、これらの結果を従来のプロセスマイニング技術と組み合わせる方法について述べる。

6.1. 一般的なアプローチ

オブジェクト中心イベントログ $L = (E, \leq_E)$ (定義 3.3) が与えられたとき、我々はアクセプタンスオブジェクト中心ペトリネット $AN = (ON, M_{init}, M_{final})$ (定義 5.5) を発見したい。我々は、特定の発見アルゴリズムを定義するのではなく、既存のプロセス発見技法を活用した一般的なアプローチを提示する。

- **ステップ 1:** オブジェクト中心イベントログ $L = (E, \leq_E)$ が与えられたとき、イベントログに現れるオブジェクトの種類 $OT \subseteq \mathbb{U}_{ot}$ を特定する。そして、各オブジェクト型 $ot \in OT$ に対して、平坦化イベントログ $L^{ot} = (E^{ot}, \leq_E^{ot})$ を作成する。
- **ステップ 2:** フラット化されたイベントログ L^{ot} を用いて、各オブジェクトタイプ $ot \in OT$ に対して $N^{ot} = (P^{ot}, T^{ot}, F^{ot}, l^{ot})$ を持つアクセプタンスペトリネット $SN^{ot} = (N^{ot}, M_{init}^{ot}, M_{final}^{ot})$ を発見する。この目的のために、従来のあらゆる発見技術を使用することができる。唯一の前提は、ラベルの重複がないこと、すなわち、ラベリング関数 l^{ot} が射影的であることである。ただし、沈黙する遷移を許容する、すなわち、 l^{ot} は部分的であってもよい。
- **ステップ 3:** アクセプタンスペトリネットをペトリネット N に結合する。名前の被り为了避免のために、まず、場所名と、異なるネットの中の沈黙する遷移の名前が異なることを確認する。また、同じラベルを持つ遷移は同じ名前であることを確認する(これはラベリング関数が帰納的であるため可能)。名前を変更した後、以下のような $P = \bigcup_{ot \in OT} P^{ot}$ 、 $T = \bigcup_{ot \in OT} T^{ot}$ 、 $F = \bigcup_{ot \in OT} F^{ot}$ 、 $l = \bigcup_{ot \in OT} l^{ot}$ を伴う全体的なラベル付きペトリネット $N = (P, T, F, l)$ を作成する。
- **ステップ 4:** 統合ペトリネット N の場所にオブジェクトタイプを割り当てる: $p \in P^{ot}$ と $ot \in OT$ に対して $pt(p) = ot$ とする。これは、異なるオブジェクトタイプの場所が不連続であるため、可能である。
- **ステップ 5:** 変数円弧 $F_{var} \subseteq F$ を特定する。これは、様々な方法(例えば、リプレイ結果を使用したり、平坦化プロセスを診断したり)で決定することが可能である。目標は、複数のトークンを消費または生成する必要がある円弧を特定することである。例としては、 τ は閾値(例えば、0.98)である $F_{var} = \{(p, t) \in F \cap (P \times T) \mid \text{score}(l(t), pt(p)) < \tau\} \cup \{(t, p) \in (T \times P) \mid \text{score}(l(t), pt(p)) < \tau\}$ と、 $\text{score}(\text{act}, ot) = |\{e \in E \mid \pi_{act}(e) = \text{act}\}| / |\{e \in E \mid \pi_{omap}(e) = \text{act}\}|$ が、タイプ ot の一つの対象を正確に参照する act イベントの断片であるスコア $\in (\mathbb{U}_{act} \times \mathbb{U}_{ot}) \rightarrow [0, 1]$ が挙げられる。
- **ステップ 6:** 前の 3 つのステップを組み合わせることで、オブジェクト中心ペトリネット $ON = (N, pt, F_{var})$ を作成することができる。初期マークと最終マークは、対応するオブジェク

トのそれぞれについて、アクセプタンスペトリネットのマークを複製することによって得られる。 M_{init} は $[(p, oi) \in Q_{ON} \mid \exists_{ot \in OT} p \in M_{init}^{ot} \wedge \exists_{e \in E} oi \in \pi_{omap}(e)(pt(p))]$ であり、 M_{final} は $[(p, oi) \in Q_{ON} \mid \exists_{ot \in OT} p \in M_{final}^{ot} \wedge \exists_{e \in E} oi \in \pi_{omap}(e)(pt(p))]$ である。

• **ステップ 7:** アクセプタンスオブジェクト中心ペトリネット $AN=(ON, M_{init}, M_{final})$ を返す。

上記のアプローチには 2 つのパラメータがある。それは(1)ステップ 2 で使用する発見技術と、(2)ステップ 5 での変数円弧の選択(例えば、閾値 τ や関数スコア)である。ステップ 2 については、重複ラベルのないペトリネットを生成する任意の発見技術を用いることができる(例えば、ラベル分割を行わない領域ベースの技術や帰納的マイニング技術など)。ステップ 5 で説明したスコアリング関数は単なる一例である。関数 $score(act, ot)$ は act イベントのうち、 ot 型のオブジェクトを正確に 1 つだけ参照している割合をカウントする。これがかなり低い場合(閾値 τ 以下)、対応する円弧は可変であるとみなされる(すなわち、これらの円弧は任意の数のトークンを消費/生成することができる)。 F_{var} の選択は遷移と場所のタイプにのみ依存するため、このアプローチは常に整形されたオブジェクト中心ペトリネットを返す。

6.2. アノテーション、ビュー、および抽出

本論文の主な新規性は、複数のオブジェクトタイプを持つ単一のプロセスモデルを発見し、イベントデータにおける複数の一対多および多対一の関係を捕捉できるようにしたことである。これに基づいて、従来のプロセスマイニングの多くのアイデアは、このより現実的な設定に変換することが可能である。本節では、そのいくつかを簡単に説明する。

プロセスモデルに頻度情報や時間情報をアノテーションするのは簡単である。例えば、Figure 6 の右側は既に様々な頻度を示しているが、我々の実装はより多くの診断を提供する。

- **遷移アノテーション:** 遷移の頻度は、オブジェクト中心イベントログで対応するアクティビティが発生した頻度を示している。また、対応するイベントに関与したオブジェクトに関する統計情報を追加することも可能である(たとえば、特定のタイプのオブジェクトが平均何個関与したかなど)。トランザクション情報(開始と完了)がある場合、対応するアクティビティの継続時間に関する情報(平均、中央値、分散、最小、最大など)を表示することも可能である。
- **場所アノテーション:** 各場所で消費されたトークンと生成されたトークンの数を表示することができる。これらのトークンはオブジェクトに対応する。したがって、その場所を訪れたユニークなオブジェクトの数と、オブジェクトごとの平均訪問回数を示すことも可能である。トークンが生成された瞬間と消費された瞬間の時間差を取ることで、タイミング情報(平均、

中央値、分散、最小、最大など)を表示することができる。コンプライアンスチェックの場合、トークンの欠落や残りを表示することも可能である(実装を参照)。

- **円弧アノテーション**: 円弧には、可変円弧 F_{var} と非可変円弧 $F \setminus F_{var}$ の 2 種類がある。どちらも頻度と時間の情報をアノテーションすることが可能である。また、可変円弧については、遷移発生ごとに生成／消費されるトークン数の統計を表示することができる。図 6 を参照すると、可変円弧に対するアノテーションは平均値を示している。例えば、 100×5 というアノテーションは、トークンの多重集合が円弧に沿って 100 回移動し、この多重集合の平均サイズが 5 であることを示し、500 個のオブジェクトが円弧に沿って移動したことを示している。

アノテーションの追加に加え、オブジェクトの種類の選択・非選択も可能である。6.1 節で説明したアプローチでは、まずイベントログに現れるオブジェクトの種類 $OT \subseteq U_{ot}$ を特定する。しかし、任意の空でない部分集合 $OT' \subseteq OT$ を取ることができる。例えば、オブジェクトタイプ **Order** を省き、**Item** と **Route** のみを使用することが可能である。こうすることで、簡略化したビューを作成することができる他、すべてを頻度ベースのフィルタリングと組み合わせることもできる。つまり、スライダーを追加して、頻度の低いアクティビティや円弧をシームレスに削除することが可能である。

ほとんどのプロセスマイニング技術はオブジェクト中心イベントログを扱えないため、古典的なイベントログを生成し、従来の技術を適用できることは貴重である。オブジェクト中心ペトリネットによって提供される全体的なビューは、1 つのオブジェクトタイプに焦点を当てたより詳細な分析のための出発点として機能する。定義 4.1 では、イベントログを平坦化することが容易であることを既に示した。また、オブジェクトの種類の組み合わせをケース識別子とすることも可能である。これは、ビューやインタラクティブなフィルタリングと組み合わせることができる。もちろん、このような結果を解釈する際には、常に十分な注意が必要である。先に述べた収束と発散の問題により、結果は誤解を招く可能性がある。しかし、全体的にオブジェクト中心ペトリネットを受け入れることで、誤解を避けることができる。

7. オブジェクト中心ペトリネットのためのツールサポート

説明したコンセプトとテクニックは、完全に実装されている。このセクションでは、実装、サポートされる機能、および性能の評価について説明する。

7.1. 実装

この論文で紹介した発見アプローチ(トークン・ベースの再生によるパフォーマンスとコンフォーマンスの解析を含む)をサポートするために、PM4Py を、Python ライブラリ PM4PyMDL を

追加して拡張した⁶。このツールは、Python Package Installer(PIP)を使ってインストールすることができる(`pipinstallpm4pymdl` コマンドを使用する)。PM4Py-MDL は、オブジェクト中心ペトリネットを発見することに加え、多次元直接追従グラフを発見することができる[21, 23]。

我々の実装は、この論文で説明したアプローチに従っている。オブジェクト中心ペトリネットの発見は、単一のオブジェクトタイプに対するペトリネットの発見に基づいている。そして、これらのペトリネットは併合され、アノテーションが付けられる。個々のオブジェクトタイプに対するペトリネットの発見には、帰納的マイナー直接追従プロセス発見アルゴリズム[47]を適用して、健全なワークフローネットを得ることができる。しかし、アクセプタンスペトリネットを生成する任意の発見技法を使用することもできる。

場所と円弧上のパフォーマンスを注釈するために、[22]で説明されたトークン・ベースの再生アプローチが使用される。このアプローチは、アプローチ[59]を改善し、実装はかなり高速化されている。各場所について、生成された p 、消費された c 、残りの r 、および欠落した m のトークンの数が計算され、表示される。これらの値は、平坦化されたイベントログ L^{ot} と、各オブジェクトタイプ ot のペトリネット SN^{ot} を受け入れて、「トークン・ゲームをする」ことによって得られる。これは、各場所が正確に 1 つのタイプを持っているために可能である。 p と c は、生成されたトークンと消費されたトークンの数である(場所ごとに報告される)。トークンの数 m は、ログが、出力遷移が発生したことを示唆しているにもかかわらず、トークンが場所に存在しない状況を意味する。残存トークン数 r は、イベントログを再生した後に残ったトークンの数である。トークンに基づく再生手法は、沈黙遷移や重複遷移(すなわち、ラベリング関数 l が部分的または非射影的な場合)にも対応することが可能である。詳細は[59, 22]を参照されたい。

性能に関するアノテーションには、トークンの生産時間と消費時間の差に基づく遅延のセットが使用される。これらの測定値に基づいて、最小、最大、平均、分散などを算出することができる。

遷移に関するアノテーションは、イベントログから直接(つまり、イベントログを再生することなく)導出される。このようにして、遷移の頻度、関与したオブジェクトの平均数、ユニークなオブジェクトの数をモデルに追加することが可能である。

7.2. ツールの機能

図 8 は、私たちの PM4Py-MDL ツールの 2 つのスクリーンショットである。以下の機能がサポートされている。

- オブジェクト中心イベントログのインポートとエクスポートを様々なフォーマットで行うことができる。現在サポートされているフォーマットは、Multi-Dimensional Logs(MDL)、Parquet、XOC(OCBC モデルに接続されたフォーマット)である。

⁶ ソフトウェアは、www.pm4py.org および <https://github.com/Javert899/pm4py-mdl.git> からダウンロードすることができる。

- オブジェクト中心なプロセスディスカバリーアプローチを幅広くサポートする。また、本稿で紹介したオブジェクト中心ペトリネットに次いで、いくつかのターゲットフォーマットが存在する。また、MVP(Multiple View Point)モデルもサポートされている。これらは、基本的に色付き円弧を持つ **Directly Follows Graphs**[3]である[21, 23]を参照されたい。この論文で紹介したアプローチは、異なる低レベルの発見技術と組み合わせることができる。例では、**Inductive Miner Directly-Follows process discovery** アルゴリズム[47]を使用する。
- 発見プロセスに影響を与える様々な閾値を設定することが可能である。例えば、アクティビティとパスの最小出現数を設定できる。また、オブジェクトの種類ごとに、その種類で考慮されるアクティビティを指定することも可能である。
- 生のイベントデータを探索するためのいくつかの方法が提供されている(例: タイプごとの関連オブジェクト数の統計や、イベントの時間的分布など)。これらのアノテーションは、場所、遷移、および円弧に付けることができる。
- パフォーマンスとコンフォーマンスの解析のために、トークン・ベースの再生がサポートされている。これにより、ボトルネックや逸脱した動作を特定することができる。
- アクティビティ、パス、タイプごとの関連オブジェクトの数に基づいてフィルタリングすることが可能である。また、タイムフレームや属性に基づくフィルタリングもサポートされている。
- イベントグラフに基づくクラスタリングやイベントの相関関係もサポートされている。
- 任意の時点で、オブジェクトの種類を選択することにより、オブジェクト中心イベントログを従来のイベントログに平坦化することが可能である。得られたイベントログは、従来のプロセスマイニングの手法で分析することができる。

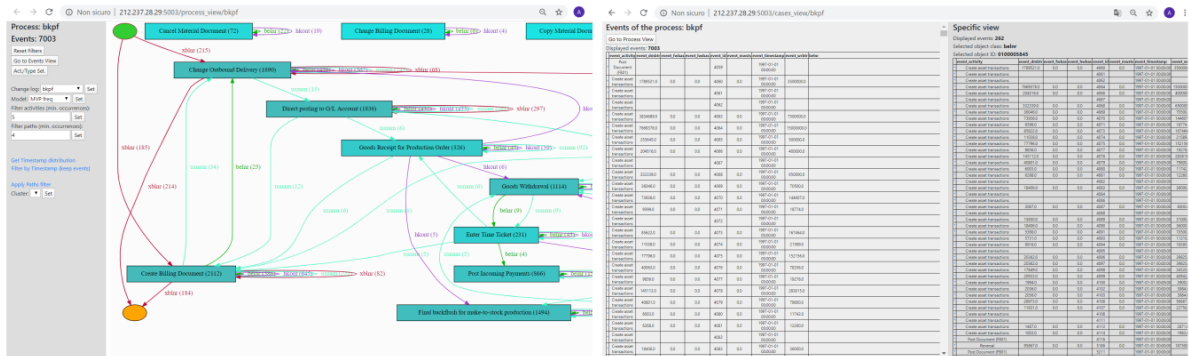


図 8. PM4Py-MDL ライブラリが提供する機能をサポートする Web インターフェイス。主なコンポーネントは、プロセスの発見(左)とイベントの探索(右)である。

ウェブインターフェイスは、主にプロセスディスカバリーとイベントエクスプローラーの 2 つで構成されている(図 8 参照)。可視化は、双方向的である。ノードをクリックすることで、アクティビティのイベントに関する統計情報を確認したり、フィルタリングオプションを設定したりすることが可能である。イベント探索は、ログのイベントを双方向的に表示する。関連するオブジェクトと対話し、オブジェクトに関連するすべてのイベントを別のパネルに表示することが可能である。このようにして、オブジェクトのライフサイクルを理解することが容易になる。次に、この手法と実装のスケーラビリティを評価する。

7.3. アプローチと実装のスケーラビリティ

このサブセクションの目的は、PM4Py-MDL ツールで実装されたオブジェクト中心ペトリネットの発見に関する拡張性を分析することである。我々は、オブジェクト中心ペトリネットの発見が、拡張性があることを期待する。なぜなら、関与するステップは、平坦化されたログに対するプロセス発見アルゴリズムの適用を除いて、最大でも線形複雑度を有するからである。さらに、我々はイベントログに対して線形である発見技術もサポートする(アクティビティの有限個が与えられた場合)。

拡張性を分析するために、この論文の他の部分でも使用されている「実行例」のオブジェクト中心イベントログのバリエーションを使用している。ここでは、次の 3 つの異なる設定が検討された。

1. イベントログのイベント数から見たアルゴリズムの実行時間(ユニークアクティビティの数とイベントごとのオブジェクトの数は一定に保ったまま)。
2. イベントログのユニークアクティビティの数から見たアルゴリズムの実行時間(イベントの数とイベントごとのオブジェクトの数を一定に保ちながら)。
3. イベントごとのオブジェクトの数から見たアルゴリズムの実行時間(ユニークアクティビティの数とイベントの数を一定に保ちながら)。

7.3.1. イベント数の増加

図 9 と図 10(a)は、全体の実行時間とオブジェクト中心イベントログのイベント数との関係を

示している。図 10(a)は、イベント数と実行時間の間に直線的な関係があることを示している。最初の"running-example"ログには 22、367 のイベントが含まれている。異なるサイズのサブセットは、ユニークなアクティビティのセットが一定のままであるように取られる(我々は、より少ない順序を考慮するだけである)。つまり、より短い期間でプロセスを観察するのである。ログ全体の分析には 1 分も要しない。これは、比較的小さなイベントログには長いと思われるかもしれないが、発見に必要な時間は 1 秒未満である。図 9 は、解析時間を 6 つの要素に分割したものである。

- すべてのイベントログのログ平坦化処理に必要な時間(**Log Fl.**)。
- プロセス探索に要する時間(**Disc.**)。本実験では、帰納的マイニングを使用する。
- トークン・ベースのリプレイ処理に要する時間(**TR**)。
- トークン・ベースリプレイの結果を基にしたパフォーマンスアノテーションの計算に要する時間(**Perf.**)。
- ログからの追加統計量の算出に要する時間(**Stats**)。
- 可視化に要する時間(**Vis.**)。

図 9 から明らかなように、トークンによる再生操作(**TR**)とパフォーマンスアノテーションの計算(**Perf.**)に多くの時間が費やされていることがわかる。前者は制御フローの変種ごとに行われ(同じ問題を繰り返し解くのを避けるため)、後者はオブジェクトごとに行われる。このため、**TR** よりも **Perf.**の方が、時間がかかる。また、イベントログの前処理(**Log Fl.**)にもかなりの時間がかかる。興味深いことに、発見そのものは他のコンポーネントと比較して非常に高速である。

図 9 と図 10(a)は、我々のアプローチの特徴が、古典的なイベントログのプロセスマイニングと類似していることを示している。イベントログを再生して適合性や性能の統計情報を収集することは、帰納的マイニングなどの技術を用いてプロセスモデルを発見するよりも多くの時間を要する。これは、シングルケースの概念を用いた従来のプロセスマイニング技術にも当てはまる。

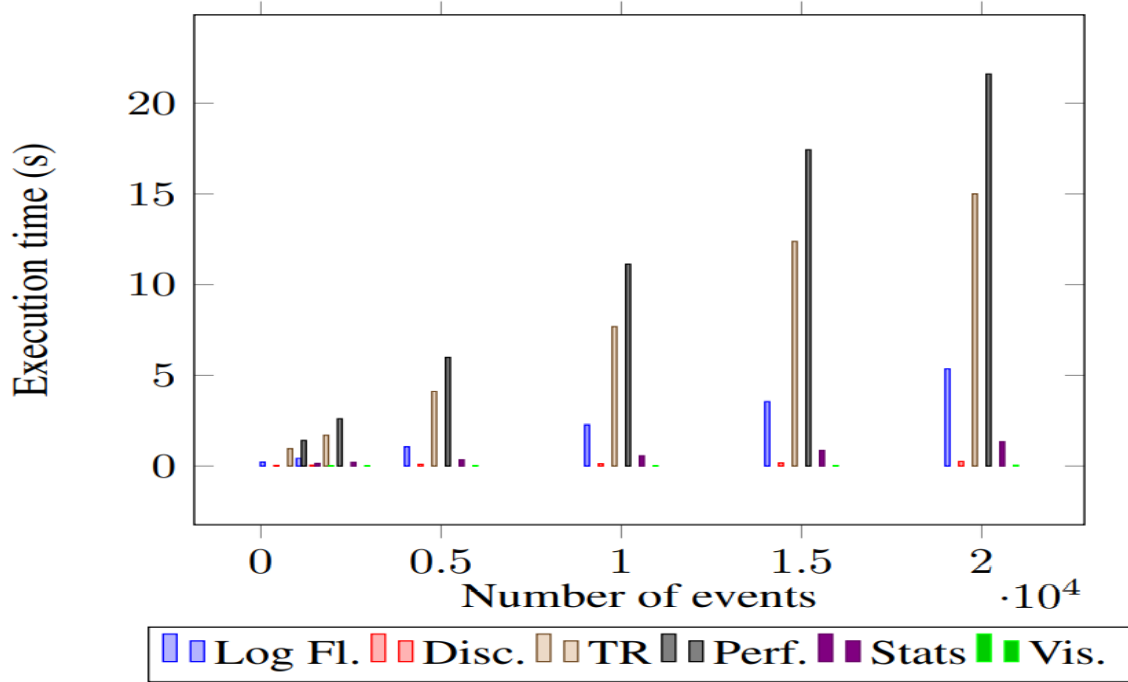


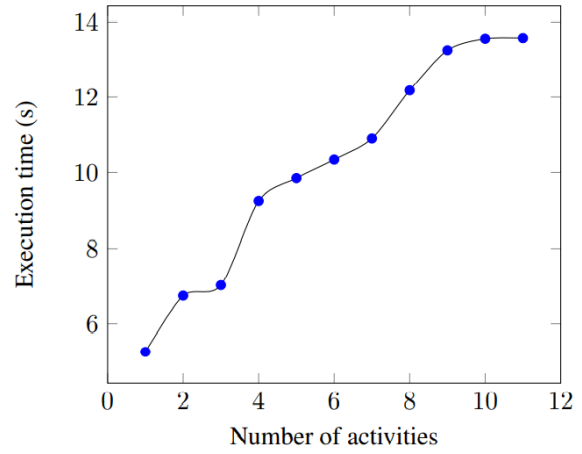
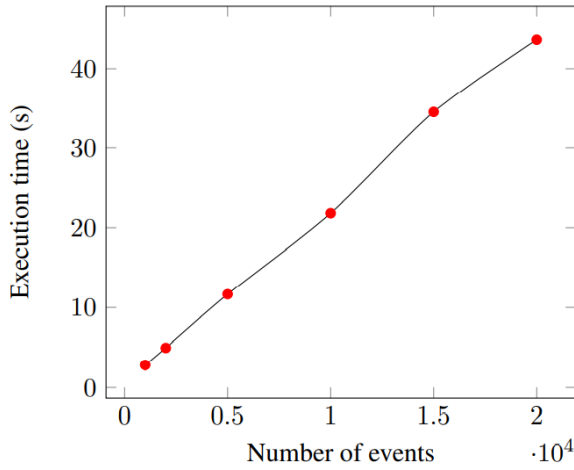
図 9. ログのイベント数を増加させた場合のアプローチ全体の実行時間の詳細分析。測定値は、5つのサブログについてグループ化されている。グループ内の列は、イベントログの平坦化(Log Fl.)、発見(Disc.)、トークン・ベーススリプレイ(TR)、計算性能注釈(Perf.)、計算統計(Stats)、可視化(Vis.)を示している。

7.3.2. アクティビティ数の増加

図 10(b)は、ユニークアクティビティの数を増やした場合の実行時間である。そして使用したイベントログは、イベント数を一定に保ったまま、アクティビティフィルタリングで作成したものである。表 7.3.2 にアクティビティ数、イベント数、全体の所要時間を示す。k 行目では、最も頻度の高い k 個のアクティビティを保持し、さらにイベントログをフィルタリングして、正確に 8159 個のイベントを持つようにした。全体的な計算時間の増加は、最もコストのかかる操作がトークン・ベースの再生とパフォーマンスアノテーションの計算であり、これらの操作の複雑さがトレースの平均的な長さに伴って直線的に増加するという事実によって説明される 7。

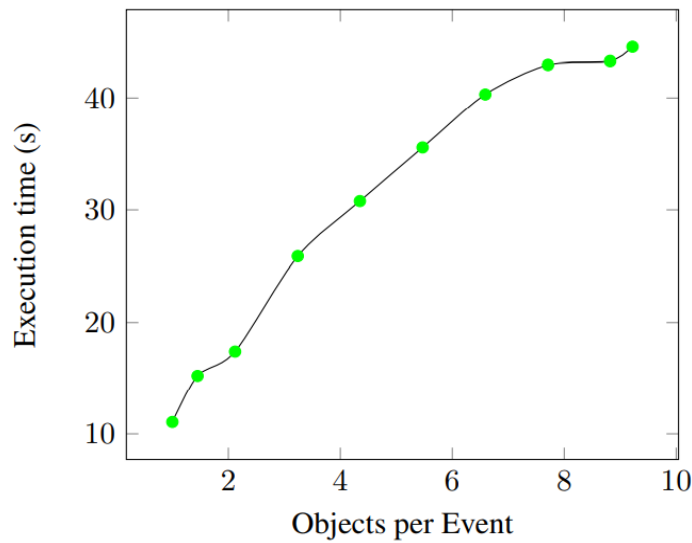
表 3. ユニークアクティビティの数を増やしながら実行した場合の実行時間。

Number of Activities	Number of events	Execution time
1	8159	5.26
2	8159	6.74
3	8159	7.02
4	8159	9.25
5	8159	9.85
6	8159	10.34
7	8159	10.90
8	8159	12.18
9	8159	13.26
10	8159	13.56
11	8159	13.58



(a) イベント数を増やした場合の実行時間

(b) アクティビティ数を増やした場合の実行時間



c) 1 イベントあたりのオブジェクト数を増やしながらの実行時間

図 10. オブジェクト中心ペトリネット発見アルゴリズムの拡張性評価。イベント数、アクティビティ数、イベント毎のオブジェクト数を変化させた場合の全体時間(再生とアノテーションを含む)を異なるグラフで示す。

7.3.3. イベントあたりの関連オブジェクト数の増加

図 10(c)は、1 イベントあたりの関連オブジェクト数を増加させた場合の実行時間である。このような設定を分析するために、イベント数と異なるアクティビティの数が変わらないように、「running-example」イベントログの異なるサブセットを作成した。関連オブジェクトのセットは、各イベントに対して少なくとも 1 つの関連オブジェクト(種類を問わず)が残るように選択されている。解析中は各オブジェクトに対してイベントが複製されるため、線形関係は予想通りであった。また、全体の解析時間とオブジェクトの種類数には線形関係があることが実験により確認されている(図示はしていない)。

全体として、非常に有望な結果である。このアプローチは、イベントの数、ユニークなアクティビティの数、オブジェクトの数に対して線形にスケールする。発見時間は、適合性チェックや性能分析に必要な時間に比べれば、ごくわずかである。したがって、このアプローチは実世界のイベントデータに適用することができる。

表 4. 22367 のイベントと 5 つのオブジェクトタイプ(注文、アイテム、製品、顧客、パッケージ)を持つ、より大きなオブジェクト中心イベントログの断片。2000 種類のオーダー、8159 個のアイテム、20 個の商品、17 個の顧客、そして 1325 個のパッケージが存在する。表は、いくつかのサンプルイベントと最初の 3 つのオブジェクトタイプを示している。

event_activity	event_timestamp	orders	items	products
place order	2019-05-20 09:07:47	['990001']	['880001', '880002', '880003', '880004']	['Echo Show 8', 'Fire Stick 4K', 'Echo', 'Echo...']
place order	2019-05-20 10:35:21	['990002']	['880005', '880006', '880007', '880008']	['iPad', 'Kindle', 'iPad Air', 'MacBook Air']
pick item	2019-05-20 10:38:17	['990002']	['880006']	['Kindle']
confirm order	2019-05-20 11:13:54	['990001']	['880001', '880002', '880003', '880004']	['Echo Show 8', 'Fire Stick 4K', 'Echo', 'Echo...']
pick item	2019-05-20 11:20:13	['990001']	['880002']	['Fire Stick 4K']
place order	2019-05-20 12:30:30	['990003']	['880009', '880010', '880011', '880012']	['iPad Air', 'iPhone 11', 'Fire Stick', 'iPhon...']
confirm order	2019-05-20 12:34:16	['990003']	['880009', '880010', '880011', '880012']	['iPad Air', 'iPhone 11', 'Fire Stick', 'iPhon...']
item out of stock	2019-05-20 13:54:37	['990001']	['880004']	['Echo Studio']
place order	2019-05-20 14:20:47	['990004']	['880013', '880014']	['Echo Studio', 'Echo Show 8']
item out of stock	2019-05-20 15:19:49	['990003']	['880009']	['iPad Air']
place order	2019-05-20 16:01:22	['990005']	['880015', '880016']	['iPad Pro', 'iPad Air']
pick item	2019-05-20 16:56:02	['990004']	['880014']	['Echo Show 8']
pick item	2019-05-20 17:08:25	['990002']	['880008']	['MacBook Air']
place order	2019-05-20 17:22:31	['990006']	['880017', '880018', '880019']	['Echo Show 8', 'Fire Stick 4K', 'iPhone X']
pick item	2019-05-20 17:51:15	['990003']	['880011']	['Fire Stick']
pick item	2019-05-20 18:15:00	['990002']	['880007']	['iPad Air']
confirm order	2019-05-20 18:36:37	['990004']	['880013', '880014']	['Echo Studio', 'Echo Show 8']
place order	2019-05-20 19:04:49	['990007']	['880020', '880021', '880022']	['Echo Show 8', 'Echo Dot', 'Kindle Paperwhite']
...	...	...	...	...

8. 応用例

アプローチとそれに対応する PM4Py-MDL の実装の実現可能性を説明するために、冒頭で簡単に述べた大きな例を使用してみよう(図 1 参照)。CSV 形式のオブジェクト中心イベントログは、https://github.com/Javert899/pm4py-mdl/blob/master/example_logs/mdl/mdl-running-example.mdから取得できる。ログの小さな断片は、3つの選択されたオブジェクトタイプを示し、表 4 で可視化されている。これは、以前に使用した小さな例の拡張版と考えることができ、合計で 2367 個のイベントがある。オブジェクトの種類は、注文、アイテム、商品、顧客、パッケージの 5 つである。イベントログには、2000 種類のオーダー、8159 個のアイテム、20 個の商品、17 個の顧客、1325 個のパッケージに関する情報が含まれている。したがって、1 つのオーダーの平均アイテム数は 4.08 であり、1 つのパッケージの平均アイテム数は 6.16 である。

特定の「アクティビティ-オブジェクトタイプ」(a, ot)の組み合わせをフィルタリングすることができる。これは、アクティビティ a とオブジェクトタイプ ot に関連するオブジェクトを除去することに相当する。表 4 では、すべてのアクティビティについて、顧客とパッケージに関連するすべてのオブジェクトを削除している。これは、顧客とパッケージの情報を持つ列を削除することに帰着する。また、特定のアクティビティに関連する行を削除することも可能である。しかし、特定の「アクティビティ-オブジェクトタイプ」の組み合わせを残すような、より細かいフィルタリングを行うことも可能である。

表 5. 最初の 2 列は、分析に使用した「アクティビティ-オブジェクトタイプ」の組み合わせを示す。例えば、「place order イベントは、product と customer に関する情報も持っていたが、これらのオブジェクトタイプは削除された」、「failed delivery イベントは、order、product、customer に関する情報も持っていたが、これらのオブジェクトタイプは削除された」などである。最後の 3 つの列は、元のイベントログの「アクティビティ-オブジェクトタイプ」の組み合わせの統計を示す(オブジェクトタイプの注文、アイテム、パッケージのみ)。オブジェクトの最小数/オブジェクトの平均数/オブジェクトの最大数の 3 つの値が報告される。

Activity	Retained object types	Orders per event	Items per event	Packages per event
place order	orders, items	1 / 1.00 / 1	1 / 4.08 / 15	0 / 0.00 / 0
confirm order	orders, items	1 / 1.00 / 1	1 / 4.08 / 15	0 / 0.00 / 0
item out of stock	items	1 / 1.00 / 1	1 / 1.00 / 1	0 / 0.00 / 0
reorder item	items	1 / 1.00 / 1	1 / 1.00 / 1	0 / 0.00 / 0
pick item	items	0 / 0.00 / 0	1 / 1.00 / 1	0 / 0.00 / 0
payment reminder	orders	1 / 1.00 / 1	1 / 4.18 / 14	0 / 0.00 / 0
pay order	orders	1 / 1.00 / 1	1 / 4.08 / 15	0 / 0.00 / 0
create package	items, packages	1 / 3.32 / 9	1 / 6.16 / 22	1 / 1.00 / 1
send package	items, packages	1 / 3.32 / 9	1 / 6.16 / 22	1 / 1.00 / 1
failed delivery	items, packages	1 / 3.21 / 8	1 / 5.95 / 18	1 / 1.00 / 1
package delivered	items, packages	1 / 3.31 / 9	1 / 6.16 / 22	1 / 1.00 / 1

表 5 は、オリジナルのオブジェクト中心イベントログの 22367 イベントの統計情報の例である。各アクティビティについて、オブジェクトの最小数、オブジェクトの平均数、および所定のタイプのオブジェクトの最大数を示している。例えば、発注イベントは常に正確に 1 つの発注オブジェクトと可変数のアイテム・オブジェクト(最小値=1、平均値=4.08、最大値=15)を指し、パッケージ送信イベントは常に正確に 1 つのパッケージ・オブジェクト、可変数のアイテム・オブジェクト(最小値=1、平均値=6.16、最大値=22)、可変数の発注オブジェクト(最小値=1、平均値=3.32、最大値=9)を指す。

この例では、表 5 に示した「アクティビティとオブジェクトの種類」の組み合わせについて検討した。すなわちオブジェクトタイプはオーダー、アイテム、パッケージを残し、アクティビティはすべて残すが、いくつかのアクティビティについては関連性の低いオブジェクトタイプを削除する。表 4 のイベントログと表 5 の「アクティビティ・オブジェクトタイプ」の組み合わせから始めると、我々の発見アプローチは図 11 に示すオブジェクト中心ペトリネットを返す。

全体の図がほとんど読めなくなっている。しかし、これまで述べてきたフィルタリングアプローチを使い、シームレスにモデルを単純化することができる(例えば、頻度の低いアクティビティを削除したり、オブジェクトタイプを少なく選択したりする)。さらに、モデルのさまざまな側面を拡大することが可能である。

図 12 は、図 11 の大きなオブジェクト中心ペトリネットの断片を示したものである。緑色のソース・プレイスはオーダー型である。赤のソース・プレイスはアイテム型である。活動場所順序は 2000 回発生し、緑色の場所から正確に 1 つのトークンを消費し、赤色の場所から可変数のトークンを消費する。複合二重矢印はこれを反映しており、銘文は平均して 4.08 個のアイテム・オブジェクトが消費されたことを示している。

図 13 は、別の断片を示す。パッケージの配送アクティビティは、パッケージとアイテムの両方のライフサイクルの最終アクティビティである。2 つの複合二重矢印は、可変数のアイテム・オブジェクトが消費され、生産されることを表している。納品されたパッケージが消費・生産するアイテム・オブジェクトの平均数は 6.16 個である。円弧のタグは、パッケージの前のアクティビティからこのアクティビティまでの平均時間が 18 時間であることを示している。アイテムに関する前のアクティビティからこのアクティビティまでの平均時間は 9 時間である。

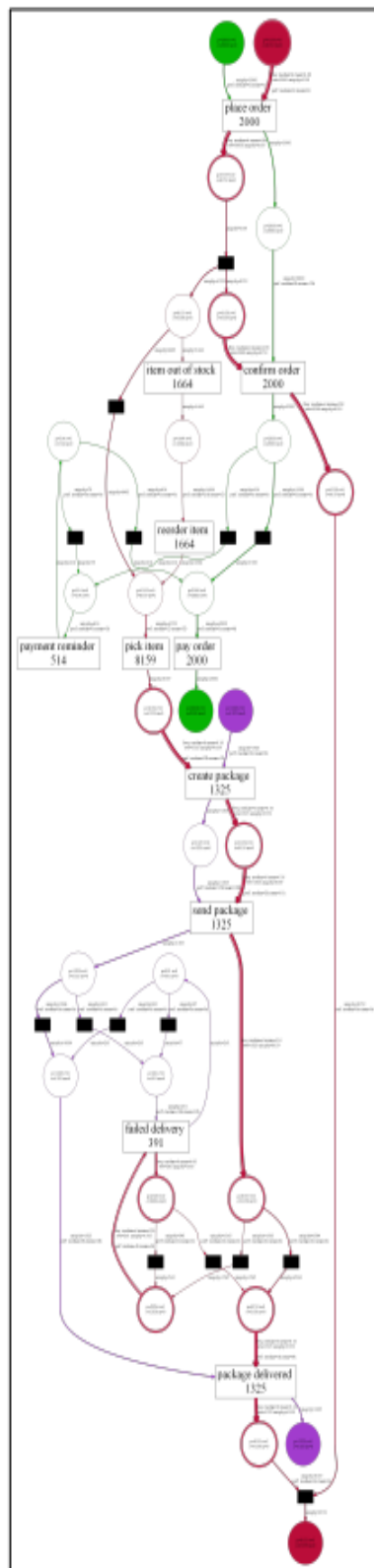


図 11. 受注(緑)、商品(赤)、パッケージ(紫)の 3 つのオブジェクトタイプを考慮し、例のログから発見されたオブジェクト中心ペトリネット。

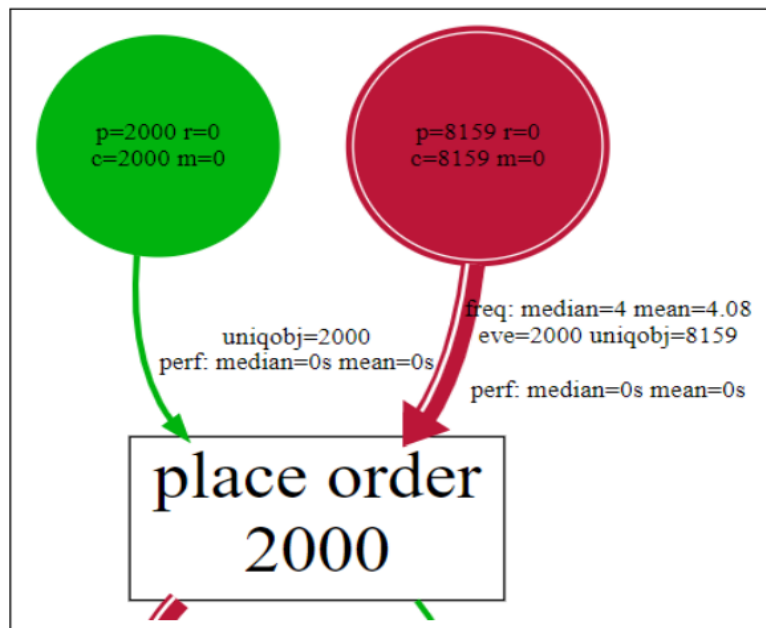


図 12. 発注活動を示すモデルの断片。ログには 2000 件の注文があり、発注はそれぞれ 1 回ずつ発生する。2000 個の注文には 8159 個のアイテムが含まれる。診断の結果、平均して 4.08 個のアイテム・オブジェクトが赤いタイプのアイテムの場所から消費されていることがわかる。

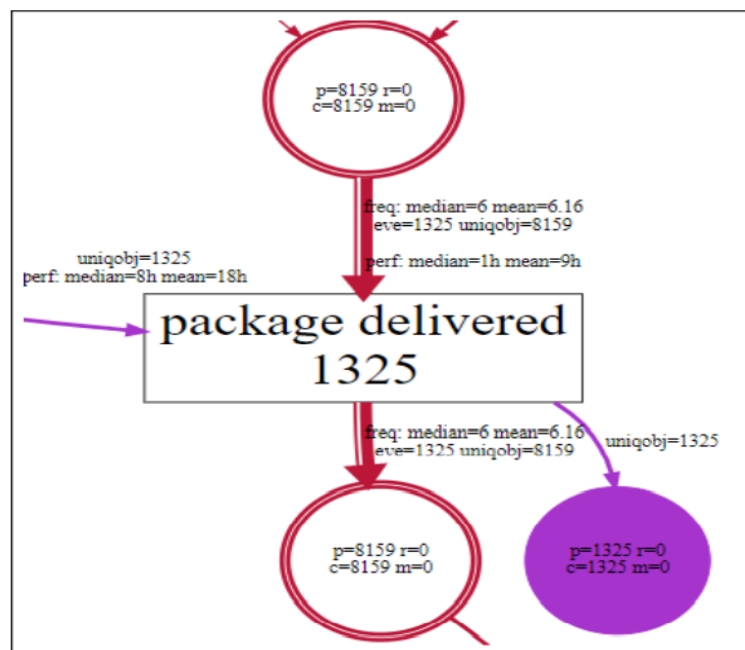


図 13. パッケージの配送アクティビティを示すモデルの断片。このアクティビティは、複数のアイテムで構成されるパッケージが正常に配送されたことに対応している。1325 個の荷物に分散された 8159 個のユニークなアイテムがある。すべてのパッケージは、配送されたパッケージ

図 11 のすべての場所にはリプレイ情報も表示されている。 p はその場所で生成されたトークンの数、 c は消費されたトークンの数、 m は不足したトークンの数、 r は残りのトークンの数である(7.1 節参照)。この例では、このモデルは完全に適合する。したがって、すべての場所について、 $m = 0$ および $r = 0$ となる。

発見されたオブジェクト中心ペトリネットから、よりシンプルなビューを生成し、従来のイベントログを作成し、さらなる分析のために従来のプロセスマイニング技術を展開することも可能である。さらに、この例では、従来のアプローチでは発見できなかった多くの知見が得られている。イベントログを平坦化することで、異なるタイプのオブジェクト間の関係が失われてしまう。さらに、異なるタイプのオブジェクトを調べようとすると、存在しないループや誤解を招く頻度／時間の診断が行われることになる。

このセクションでは、オブジェクトの種類ごとに個別のプロセスモデルを使用する従来のモデルと比較して、本アプローチを使用する主な利点を強調した。要約すると

- 図 11(および対応するモデルの断片)は、プロセス全体と異なるオブジェクトタイプ間の相互作用の概要を示している。各オブジェクトタイプを個別のケース概念として考える場合、これらの相互作用を示さない、複数の切断されたモデルが得られる。
- 欠損、収束、発散の問題は回避される(定義 4.3 参照)。すべての事象は一度だけ正確に考慮される。すなわち、事象が消滅したり、意図せずに複製されたりすることはない。さらに、発散による人為的なループも回避される。
- トークン・ベースの再生により、パフォーマンスとコンフォーマンスの情報を 1 つの全体的なモデルに投影することができる。統計情報のほとんどは、フラット化されたプロセスモデルでは見ることができない(例えば、アクティビティに関与するオブジェクトの平均数など)。

9. 関連作品

ここでは、従来のシングルケースの概念を用いたプロセスマイニング手法、複数のオブジェクトタイプを扱うモデリングアプローチ、複数のオブジェクトタイプを扱うプロセスマイニングアプローチについて説明する。

9.1. シングルケース概念を用いた従来のプロセスマイニング技術

冒頭で、シングルケース概念を用いた古典的なイベントログに基づくいくつかのプロセス発見アプローチに言及した[2]。これらの手法の多くは、古典的なペトリネット(例えば、場所遷移ネット)を発見し、例えば、領域ベースのアプローチは場所を導出するために使用することができる[15,

16, 17, 18, 20, 25, 26, 27, 29, 30, 33, 42, 43, 51, 52, 53, 54, 61, 63]。領域ベースのプロセス発見技術は、イベントログからプロセスモデルを導出するための全てのアプローチの一部分に過ぎない。プロセスモデルを学習する最新の技術としては、帰納的マイニング技術[45, 46]や、いわゆるスプリットマイナー[14]などが挙げられる。商用システムでは、Directly Follows Graph(DFG)を用いることが多いが、[3]で説明したような明らかな制約がある。上記のアプローチはすべて単一事例を想定している。これは、ワークフローネット[1, 8]やプロセスツリー[44]からビジネスプロセスモデリング表記法(BPMN)モデル[58]やイベント駆動型プロセスチェーン(EPC)[60]まで、単一ケースの概念を前提とした従来のプロセスモデルと整合している。

9.2. 複数のオブジェクト型を利用したモデリング技法

プロセスモデルの多くはシングルケースの概念を用いているが、この方法では多くのプロセスを適切に捉えることができないという問題が早くから指摘されていた。IBM の FlowMark システムでは、サブケースで構成されるケースを扱うために、いわゆる「バンドル」概念をすでにサポートしていた[39]。これは、世紀の変わり目に特定されたワークフローパターンのカテゴリである、マルチインスタンスパターンに関連している[9]。この問題を解決しようとした最初のプロセスモデリング表記法のひとつが、いわゆるプロクレットである[6, 7]。プロクレットは、軽量で相互作用するワークフロープロセスである。相互作用を第一級オブジェクトに昇格させることによって、プロクレットを使用して、より自然な方法で複雑なワークフローをモデル化することができる。

その後、人工物中心のモデリング表記法[24, 28, 50, 57]など、他のアプローチも登場した。プロセスのインスタンスが一对多数または多数対一で相互作用する場合に発生する課題の最新概説は[36]を参照されたい。

9.3. 複数のオブジェクトタイプを利用したプロセスマイニング技術

収束・発散するインスタンスを持つ相互作用するプロセスに関する研究のほとんどは、新しいモデリング表記法の開発とそのプロセスの実装をサポートすることに焦点を当てたものである。プロセスマイニングの文脈でこの問題に焦点を当てたアプローチはごくわずかである。SAP、Oracle、Microsoft、およびその他のエンタープライズソフトウェアベンダーの ERP システムにプロセスマイニングを適用すると、すぐにこの問題に遭遇するので、これは驚くべきことである。この問題は、従来のプロセスモデルを作成するためにイベントデータを「平坦化」する必要性を議論した[2]のセクション 5.5 でも提起されている。

[31]では、ソースシステムから「非平坦」なイベントデータを抽出し、従来のプロセスマイニングに対応させるための技術が説明されている。eXtensible Event Stream(XES)フォーマット[40]は、イベントデータを保存するための公式の IEEE 標準であり、多くのプロセスマイニング

ベンダーによってサポートされている。XES は、イベントを関連付けるためにケース概念を必要とする。標準的な IEEE XES フォーマット[40]の次に、1 対多数や多数対多数の関係を持つオブジェクト中心のデータ(例えば、データベーステーブル)を扱うために、eXtensible Object-Centric(XOC)[48]などの新しいストレージフォーマットが提案されている。XOC 形式は、多次元データの平坦化を避けるために、大文字・小文字の区別を必要としない。XOC ログは、データベースの進化に対応するイベントとともに正確に保存することができるが、XOC のログは非常に大きくなりやすいという欠点がある。

[37, 38, 56]のアプローチは、各プロセスが独自のケース識別子を使用する、相互作用するプロセスに焦点を当てたものである。[56]では、相互作用する人工物は ERP システムから発見される。[37]では、従来の適合性検査が、相互作用する人工物の適合性を検査するために適用された。

主な課題の 1 つは、人工物のモデルが複雑になり、理解が難しくなる傾向があることである。この問題に取り組む試みとして、Van Eck らは、複数の視点があり、それぞれが単純な遷移系でモデル化された、より単純な設定を使用している[34, 62]。これらは *artifact centric process model* とも呼ばれるが、[24, 28, 37, 38, 50, 57, 56]で用いられているモデルよりも単純なものである。ケースの状態は、観点ごとに 1 つの状態に分解され、モデル全体が単純化される。サブステート間の関係は、明示的な因果関係制約ではなく、相関関係として捉えられる。同時並行性は、異なる観点の間にのみ存在し、個々の観点の中には存在しない。最近の拡張では、各視点は複数回インスタンス化することが可能である。つまり、人工物タイプ間の多数対多数の関係を可視化することができる[62]。

上記の手法では、プロセス全体が 1 つの図で可視化されるのではなく、異なる(サブ)ケースの概念を用いて相互に接続された図の集合体として表示されるという欠点がある。いわゆるオブジェクト中心動作制約(OCBC)モデルはこの問題に対処し、データの観点も単一のダイアグラムに組み込んでいる[5, 10, 13, 49]。OCBC モデルは、データモデルを行動的な観点で拡張したものである。データモデルでは、多数対多数や 1 対多数の関係を容易に扱うことができる。これを利用して、異なるタイプのインスタンス間の複雑な相互作用もモデル化できるプロセスモデルを作成する。古典的な複数インスタンスの問題は、データモデルをイベント相関に使用することで回避される。アクティビティはデータの視点に関連し、Declare[11]のような宣言型言語から着想を得た順序制約を持っている。LTL ベースの制約の代わりに、より単純な多重度制約が用いられる。OCBC モデルのために、いくつかの発見技術が開発されている[49]。また、このようなモデルの適合性やプロジェクトの性能情報を確認することも可能である。OCBC モデルは、動作とデータの間を忠実に記述し、すべての情報を 1 つの統合されたダイアグラムで捉えることができるという点で魅力的であるが、OCBC モデルは複雑すぎる傾向があり、対応する発見技術や適合性チェック技術はあまり拡張的ではない。

OCBC モデルの複雑さと拡張性の問題から、MVP(Multiple ViewPoint)モデル(以前は StarStar モデル)と呼ばれるモデルが開発された[21, 23]。MVP モデルは、リレーショナルデータベースに格納されたデータから学習される。まず、従来のデータベースのリレーションとタイムスタンプをもとに、イベントとオブジェクトを関連付ける E2O グラフと呼ばれるものを構築

する。E2O グラフを基に、イベントを、オブジェクトを介して関係付ける E2E マルチグラフを学習する。最後に、アクティビティを関連付ける A2A マルチグラフが学習される。A2A グラフはアクティビティ間の関係を示し、各関係は入力として使われたオブジェクトクラスの 1 つに基づいている。これはシンプルで拡張的であるため、非常に有望なアプローチである。オブジェクト中心ペトリネットを発見するアプローチは、[21, 23]の研究の継続と見ることができる。

イベントデータの収束と分岐に関する問題は商用ベンダーも認識しているが、成果物中心モデル、OCBC(Object Centric Behavioral Constraint)モデル、MVP(Multiple ViewPoint)モデルのような概念を実際にサポートしているところはない。しかし、商用システムで実装された最初の試みがいくつかある。その一例が Celonis で、サブインスタンス間の並行処理がグループに変換される「スパゲッティ的」モデルを避けるために、二次的なケース識別子の使用をサポートしている。Celonis の直行グラフは、サブインスタンス間の相互作用を考慮しないため、よりシンプルなモデルを実現する。もう一つの例は、myInvenio でサポートされているマルチレベルのディスカバリー技術である。得られたモデルは、異なるアクティビティが異なるケース概念に対応する(ただし、アクティビティごとに 1 つのケース概念)簡略化された MVP モデルと見なすことができる。このアプローチの問題点は、現実には、同じイベントが複数のケース概念に対応することがあり、特にダイアグラムに示される頻度に影響するため、1 つを選択すると誤解を招くことが多いということである。

近年のプロセスマイニングの進展にもかかわらず、複数のプロセスインスタンスが相互に作用する問題は十分に解決されていない。その問題の一つは、XES で見られる「平坦化」されたイベントデータを超える標準化されたイベントデータがないことである。そのため、プロセスマイニングの競技会では、古典的なイベントログに焦点が当てられる傾向にある。以前の論文[4, 21, 23]で、我々はすでにオブジェクト中心なプロセスマイニングの必要性を強調した。本論文では、オブジェクト中心イベントログからオブジェクト中心ペトリネットを学習するための、具体的で、同時に汎用的な発見アプローチを提供した。

10. まとめ

データモデルやデータベーススキーマを見ると、プロセスに関連する異なるタイプのオブジェクトの間に一对多数や多数対一の関係が存在することが多い。主流のプロセスモデリングやプロセスマイニングのアプローチは、特定のケースの概念を強制するため、モデラーやアナリストは特定の視点を選択することを余儀なくされる。この問題は、異なるケース概念をカバーするために複数のイベントログを抽出し、ケース概念ごとに 1 つのモデルを検討することで部分的に解決することができる。しかし、異なるタイプのオブジェクト間の相互作用を示す、より全体的な 1 つのプロセスモデルを持つことが望ましい。さらに、分析のために 1 つまたは複数の特定のケース概念を選択する必要があるため、本論文で議論した発散と収束の問題が発生する。

そこで、本稿では、情報システム内の実データと従来のイベントログ(XES ベースなど)の中間の表現として、オブジェクト中心イベントログを用いる。オブジェクト中心イベントログは、ケ

ースの概念に依存しない。その代わり、イベントはオブジェクトの任意のセットを参照することができる。一つのイベントは、異なるタイプの複数のオブジェクトを参照することができる。入力形式が異なるだけでなく、対象言語もオブジェクト中心ペトリネットを使用している。このペトリネットは色付きペトリネットの限定版で、場所は型付けされ、トークンはオブジェクトを指し、遷移はアクティビティに対応する。他の主流の表記法とは異なり、遷移は異なるタイプのオブジェクトを可変個消費・生成することができる。我々は、オブジェクト中心イベントログからオブジェクト中心ペトリネットを発見するための、具体的かつ汎用的なアプローチを提示した。このアプローチは **PM4Py** で実装され、様々なアプリケーションにより、このアプローチが新しい洞察を提供し、高い拡張性(オブジェクト、オブジェクトタイプ、イベント、アクティビティの数において線形)を持つことが示されている。したがって、このアイデアは商用ツールに直接実装可能であり、既存のソフトウェアは大規模な組織における実際のイベントデータの分析に使用することが可能である。

本論文は、オブジェクト中心イベントログからオブジェクト中心ペトリネットを学習することを目的とした最初の論文である。我々の発見は、さらなる研究のための多くの機会を示している。これらは以下の通りである。

- オブジェクト中心ペトリネットとオブジェクト中心イベントログに基づく適合性検査技術の開発を目指す。イベントログがプロセスモデル上で再生可能かどうかのチェックの次に、多重度を用いた外れ値の検出が興味深い。現在の実装では、既に欠損トークンや残存トークンを報告しているが、これらは平坦化されたイベントログに基づいている。
- 本アプローチは汎用的であり、平坦化されたイベントログに対して独立して動作する異なるプロセス発見アルゴリズム(帰納的マイニング、リージョンベース技術など)を組み込むことができる。その結果は、オブジェクト中心ペトリネットに折り畳まれる。異なるアプローチを比較し、より統合的なアプローチを開発することは興味深い(例えば、最初にあるオブジェクトタイプのプロセスモデルを発見し、その後、他のオブジェクトタイプを反復的に追加する)。さらに、専用の品質指標(例：複雑度、精度)があると良いだろう。
- 現在のオブジェクト中心ペトリネットは、実際の動作の「過度な近似」と見なすことができる。モデルをより精密にする方法(例えば、ガードを自動的に検出したり、分割と結合を関連付けたり)を考えるのは興味深いことである。例えば、表 4 において、完了とマークされた遷移は、遷移の場所順の発生に先に関与したオブジェクトの同じ集合に結合するはずである。同様に、我々はモデルに確率論(例えば、注文の項目数に対する確率分布)を追加したい。
- 現在のオブジェクト中心イベントログは、オブジェクトの識別子のみを含み、オブジェクトのプロパティは含まれていない。オブジェクトの識別子が患者や顧客を指す場合、その年齢、体重、住所、収入などを知ることはできない。オブジェクトの属性がないため、自動的に「過大な推定」が行われる。そこで、オブジェクト中心の拡張イベントログを開発中である。

- また、遷移、場所、円弧にタイミングや頻度診断を加えるだけでなく、より高度な性能解析のあり方を検討していく予定である。異なるオブジェクトの種類は、お互いにどのような影響を及ぼすのだろうか？オブジェクト間の相互作用の解析の次に、CPN Tools との連携による「what if」解析(例えば、改善されたプロセスでのイベントログの再生など)をより良くサポートしたいと考えている。
- また、オブジェクトを中心としたイベントログの包括的で一般に利用可能なセットを作成することを目指す。

謝辞 : AlexandervonHumboldt(AvH)Stiftung による研究支援に感謝する。

参考文献

- [1] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1):21–66, 1998.
- [2] W.M.P. van der Aalst. Process Mining: Data Science in Action. Springer-Verlag, Berlin, 2016.
- [3] W.M.P. van der Aalst. A Practitioner’s Guide to Process Mining: Limitations of the Directly-Follows Graph. In International Conference on Enterprise Information Systems (Centeris 2019), volume 164 of Procedia Computer Science, pages 321–328. Elsevier, 2019.
- [4] W.M.P. van der Aalst. Object-Centric Process Mining: Dealing With Divergence and Convergence in Event Data. In P.C. Olveczky and G. Sala “un, editors, “Software Engineering and Formal Methods (SEFM 2019), volume 11724 of Lecture Notes in Computer Science, pages 3–25. Springer-Verlag, Berlin, 2019.
- [5] W.M.P. van der Aalst, A. Artale, M. Montali, and S. Tritini. Object-Centric Behavioral Constraints: Integrating Data and Declarative Process Modelling. In Proceedings of the 30th International Workshop on Description Logics (DL 2017), volume 1879 of CEUR Workshop Proceedings. CEUR-WS.org, 2017.
- [6] W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Workflow Modeling using Procllets. In O. Etzion and P. Scheuermann, editors, 7th International Conference on Cooperative Information Systems (CoopIS 2000), volume 1901 of Lecture Notes in Computer

Science, pages 198–209. Springer-Verlag, Berlin, 2000.

[7] W.M.P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *International Journal of Cooperative Information Systems*, 10(4):443–482, 2001.

[8] W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.

[9] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[10] W.M.P. van der Aalst, G. Li, and M. Montali. Object-Centric Behavioral Constraints. *CoRR*, abs/1703.05740, 2017.

[11] W.M.P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - Research and Development*, 23(2):99–113, 2009.

[12] W.M.P. van der Aalst and C. Stahl. *Modeling Business Processes: A Petri Net Oriented Approach*. MIT Press, Cambridge, MA, 2011.

[13] A. Artale, D. Calvanese, M. Montali, and W.M.P. van der Aalst. Enriching Data Models with Behavioral Constraints. In S. Borgo, editor, *Ontology Makes Sense (Essays in honor of Nicola Guarino)*, pages 257–277. IOS Press, 2019.

[14] A. Augusto, R. Conforti, M. Marlon, M. La Rosa, and A. Polyvyanyy. Split Miner: Automated Discovery of Accurate and Simple Business Process Models from Event Logs. *Knowledge Information Systems*, 59(2):251–284, May 2019.

[15] E. Badouel, L. Bernardinello, and P. Darondeau. *Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series*. Springer-Verlag, Berlin, 2015.

[16] E. Badouel and P. Darondeau. Theory of Regions. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer-Verlag, Berlin, 1998.

- [17] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Process Mining Based on Regions of Languages. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 375–383. Springer-Verlag, Berlin, 2007.
- [18] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri Nets from Finite Partial Languages. *Fundamenta Informaticae*, 88(4):437–468, 2008.
- [19] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri Nets from Scenarios with VipTool. In *Applications and Theory of Petri Nets (Petri Nets 2008)*, volume 5062 of *Lecture Notes in Computer Science*, pages 388–398. Springer-Verlag, Berlin, 2008.
- [20] R. Bergenthum, J. Desel, S. Mauser, and R. Lorenz. Synthesis of Petri Nets from Term Based Representations of Infinite Partial Languages. *Fundamenta Informaticae*, 95(1):187–217, 2009.
- [21] A. Berti and W.M.P. van der Aalst. StarStar Models: Using Events at Database Level for Process Analysis. In P. Ceravolo, M. van Keulen, and M.T. Gomez Lopez, editors, *International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018)*, volume 2270 of *CEUR Workshop Proceedings*, pages 60–64. CEUR-WS.org, 2018.
- [22] A. Berti and W.M.P. van der Aalst. Reviving Token-based Replay: Increasing Speed While Improving Diagnostics. In *Proceedings of the International Workshop on Algorithms and Theories for the Analysis of Event Data (ATAED 2019)*, volume 2371 of *CEUR Workshop Proceedings*, pages 87–103. CEUR-WS.org, 2019.
- [23] A. Berti and W.M.P. van der Aalst. Discovering Multiple Viewpoint Models from Relational Databases. In P. Ceravolo, M. van Keulen, and M.T. Gomez Lopez, editors, *Postproceedings International Symposium on Data-driven Process Discovery and Analysis*, volume 379 of *Lecture Notes in Business Information Processing*, pages 24–51. Springer-Verlag, Berlin, 2020.
- [24] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards Formal Analysis of Artifact-Centric Business Process Models. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 288–304. Springer-Verlag, Berlin, 2007.
- [25] J. Carmona, J. Cortadella, and M. Kishinevsky. A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In *Business Process Management (BPM 2008)*, pages 358–373,

2008.

[26] J. Carmona, J. Cortadella, and M. Kishinevsky. New Region-Based Algorithms for Deriving Bounded Petri Nets. *IEEE Transactions on Computers*, 59(3):371–384, 2010.

[27] J. Carmona, J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. A Symbolic Algorithm for the Synthesis of Bounded Petri Nets. In *Applications and Theory of Petri Nets (Petri Nets 2008)*, pages 92–111, 2008.

[28] D. Cohn and R. Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *IEEE Data Engineering Bulletin*, 32(3):3–9, 2009.

[29] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers*, 47(8):859–882, August 1998.

[30] P. Darondeau. On the Synthesis of Zero-Safe Nets. In *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 364–378. Springer-Verlag, Berlin, 2008.

[31] E. González López de Murillas, H.A. Reijers, and W.M.P. van der Aalst. Connecting Databases with Process Mining: A Meta Model and Toolset. In R. Schmidt, W. Guedria, I. Bider, and S. Guerreiro, editors, *Enterprise, Business-Process and Information Systems Modeling (BPMDS 2015)*, volume 248 of *Lecture Notes in Business Information Processing*, pages 231–249. Springer-Verlag, Berlin, 2016.

[32] J. Desel and W. Reisig. The Synthesis Problem of Petri Nets. *Acta Informatica*, 33(4):297–315, 1996.

[33] B.F. van Dongen, J. Desel, and W.M.P. van der Aalst. Aggregating Causal Runs into Workflow Nets. In K. Jensen, W.M.P. van der Aalst, M. Ajmone Marsan, G. Franceschinis, J. Kleijn, and L.M. Kristensen, editors, *Transactions on Petri Nets and Other Models of Concurrency (ToPNoC VI)*, volume 7400 of *Lecture Notes in Computer Science*, pages 334–363. Springer-Verlag, Berlin, 2012.

[34] M.L. van Eck, N. Sidorova, and W.M.P. van der Aalst. Guided Interaction Exploration in Artifact-centric Process Models. In *IEEE Conference on Business Informatics (CBI 2017)*, pages 109–118. IEEE Computer Society, 2017.

- [35] A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.
- [36] D. Fahland. Describing Behavior of Processes with Many-to-Many Interactions. In S. Donatelli and S. Haar, editors, *Applications and Theory of Petri Nets 2019*, volume 11522 of *Lecture Notes in Computer Science*, pages 3–24. Springer-Verlag, Berlin, 2019.
- [37] D. Fahland, M. De Leoni, B. van Dongen, and W.M.P. van der Aalst. Behavioral Conformance of Artifact-Centric Process Models. In A. Abramowicz, editor, *Business Information Systems (BIS 2011)*, volume 87 of *Lecture Notes in Business Information Processing*, pages 37–49. Springer-Verlag, Berlin, 2011.
- [38] D. Fahland, M. De Leoni, B. van Dongen, and W.M.P. van der Aalst. Many-to-Many: Some Observations on Interactions in Artifact Choreographies. In D. Eichhorn, A. Koschmider, and H. Zhang, editors, *Proceedings of the 3rd Central-European Workshop on Services and their Composition (ZEUS 2011)*, *CEUR Workshop Proceedings*, pages 9–15. CEUR-WS.org, 2011.
- [39] IBM. IBM MQSeries Workflow - Getting Started With Buildtime. IBM Deutschland Entwicklung GmbH, Boeblingen, Germany, 1999.
- [40] IEEE Task Force on Process Mining. XES Standard Definition. www.xes-standard.org, 2013.
- [41] K. Jensen and L.M. Kristensen. *Coloured Petri Nets*. Springer-Verlag, Berlin, 2009.
- [42] J. Kleijn, M.Koutny, and M. Pietkiewicz-Koutny. Regions of Petri nets with a/sync connections. *Theoretical Computer Science*, 454:189–198, 2012.
- [43] J. Kleijn, M.Koutny, M. Pietkiewicz-Koutny, and G. Rozenberg. Applying Regions. *Theoretical Computer Science*, 658:205–215, 2017.
- [44] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-structured Process Models from Event Logs: A Constructive Approach. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer-Verlag, Berlin, 2013.
- [45] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In N. Lohmann, M. Song,

and P. Wohed, editors, Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013), volume 171 of Lecture Notes in Business Information Processing, pages 66–78. Springer-Verlag, Berlin, 2014.

[46] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery with Guarantees. In K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, and Q. Ma, editors, Enterprise, Business-Process and Information Systems Modeling (BPMDS 2015), volume 214 of Lecture Notes in Business Information Processing, pages 85–101. Springer-Verlag, Berlin, 2015.

[47] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, 17(2):599–631, 2018.

[48] G. Li, E. Gonzalez Lopez de Murillas, R. Medeiros de Carvalho, and W.M.P. van der Aalst. Extracting Object-Centric Event Logs to Support Process Mining on Databases. In J. Mendling and H. Mouratidis, editors, Information Systems in the Big Data Era, CAiSE Forum 2018, volume 317 of Lecture Notes in Business Information Processing, pages 182–199. Springer-Verlag, Berlin, 2018.

[49] G. Li, R. Medeiros de Carvalho, and W.M.P. van der Aalst. Automatic Discovery of Object-Centric Behavioral Constraint Models. In W. Abramowicz, editor, Business Information Systems (BIS 2017), volume 288 of Lecture Notes in Business Information Processing, pages 43–58. Springer-Verlag, Berlin, 2017.

[50] N. Lohmann. Compliance by Design for Artifact-Centric Business Processes. In S. Rinderle, F. Toumani, and K. Wolf, editors, Business Process Management (BPM 2011), volume 6896 of Lecture Notes in Computer Science, pages 99–115. Springer-Verlag, Berlin, 2011.

[51] R. Lorenz, R. Bergenthum, J. Desel, and S. Mauser. Synthesis of Petri Nets from Finite Partial Languages. In T. Basten, G. Juhas, and S.K. Shukla, editors, International Conference on Application of Concurrency to System Design (ACSD 2007), pages 157–166. IEEE Computer Society, 2007.

[52] R. Lorenz, J. Desel, and G. Juhas. Models from Scenarios. In K. Jensen, W.M.P. van der Aalst, G. Balbo, M. Koutny, and K. Wolf, editors, Transactions on Petri Nets and Other Models of Concurrency (ToPNoC VII), volume 7480 of Lecture Notes in Computer Science, pages 314–371. Springer-Verlag, Berlin, 2013.

- [53] R. Lorenz and G. Juhas. Towards Synthesis of Petri Nets from Scenarios. In S. Donatelli and P.S. Thiagarajan, editors, *Application and Theory of Petri Nets 2006*, volume 4024 of *Lecture Notes in Computer Science*, pages 302–321. Springer-Verlag, Berlin, 2006.
- [54] R. Lorenz and G. Juhas. How to Synthesize Nets from Languages: A Survey. In S.G. Henderson, B. Biller, M. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, editors, *Proceedings of the Wintersimulation Conference (WSC 2007)*, pages 637–647. IEEE Computer Society, 2007.
- [55] X. Lu, D. Fahland, and W.M.P. van der Aalst. Conformance Checking Based on Partially Ordered Event Data. In F. Fournier and J. Mendling, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2014)*, volume 202 of *Lecture Notes in Business Information Processing*, pages 75–88. Springer-Verlag, Berlin, 2015.
- [56] X. Lu, M. Nagelkerke, D. van de Wiel, and D. Fahland. Discovering Interacting Artifacts from ERP Systems. *IEEE Transactions on Services Computing*, 8(6):861–873, 2015.
- [57] A. Nigam and N.S. Caswell. Business artifacts: An Approach to Operational Specification. *IBM Systems Journal*, 42(3):428–445, 2003.
- [58] OMG. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03, 2011.
- [59] A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
- [60] A.W. Scheer. *Business Process Engineering: Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.
- [61] M. Sole and J. Carmona. Process Mining from a Basis of State Regions. In J. Lilius and W. Penczek, editors, *Applications and Theory of Petri Nets 2010*, volume 6128 of *Lecture Notes in Computer Science*, pages 226–245. Springer-Verlag, Berlin, 2010.
- [62] M.L. van Eck, N. Sidorova, and W.M.P. van der Aalst. Multi-instance Mining: Discovering Synchronisation in Artifact-Centric Processes. In F. Daniel, Q.Z. Sheng, and H. Motahari, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2018)*, volume 342 of *Lecture Notes in Business Information*

Processing, pages 18–30. Springer-Verlag, Berlin, 2018.

[63] S.J. van Zelst, B.F. van Dongen, W.M.P. van der Aalst, and H.M.W Verbeek. Discovering Workflow Nets Using Integer Linear Programming. *Computing*, 100(5):529–556, 2018.