決定木学習を利用したビジネスプロセス実行ログ検証のための論理式の生成

堀田　大貴[†a)]　平山　秀昭[†]　早瀬　健夫[†]　田原　康之[†]
大須賀昭彦[†]

Generating LTL Formula for Verification of Business Process Event Logs Using Decision Tree Learning

Hiroki HORITA[†a)], Hideaki HIRAYAMA[†], Takeo HAYASE[†], Yasuyuki TAHARA[†], and Akihiko OHSUGA[†]

あらまし　情報システムによって記録されたビジネスプロセスの実行履歴を分析することはプロセスマイニングと呼ばれ，実際に行われたビジネスプロセスの問題を把握して改善へ繋げるための重要な手段である．LTL checker は線形時相論理 (LTL) をベースにした形式的な言語を利用してビジネスプロセスにおいて成り立つべき性質を記述し，検証を行うためのツールであり，ビジネスプロセスの分析を行うための有力な手段として知られている．しかし，多くのビジネスアナリストは LTL のような数学的な記法に精通していないため，ビジネスプロセスにおいて検証したい性質を記述する際に，真に検証すべき性質を正確に記述することは困難である．論理式を誤って記述した場合は当然のことながら本来意図していた検証を行うことはできない．そこで本研究では教師あり機械学習手法の一種である決定木を用いてビジネスプロセス実行ログからイベントの実行順序関係に着目して抽出した特徴量に基づいて学習を行い，論理式を自動生成することで検証したい性質を記述する手法を提案する．本手法を用いることで，数学的な手法に精通していない者でも検証すべき性質を記述することができる．本手法の妥当性を示すために，電話修理プロセスに対し提案手法を適用し，有効性を確認した．

キーワード　ビジネスプロセス，検証，機械学習，プロセスマイニング

1. まえがき

情報システムを利用することでビジネスプロセスの実行や管理を効果的に行うことができるとともに，ビジネスプロセスの改善を目的とした分析に有用な多くの情報が記録される．このようなデータの分析はプロセスマイニング [1] と呼ばれ，近年注目されている．イベントログと呼ばれるビジネスプロセスの実行ログを分析することはモデル検査のような形式検証技術を用いてビジネスプロセス設計時にモデルを分析することとは異なり，ビジネスプロセスを実際に実行した後に記録したデータを分析するため，as-is 分析を行い，現場の実情を踏まえてビジネスプロセスを改善するための手段として有効である．

LTL checker [2] は線形時相論理 (LTL) [3] をベースに拡張した形式的な言語によってビジネスプロセスにおいて成り立つべき性質を記述し，ビジネスプロセスのイベントログ上で性質が成り立っているか検証するためのツールである．このようなツールを用いることで，組織の構成員にインタビューを行うよりも，より客観的に現状のビジネスプロセスで行われている物事を捉えるための支援を行うことができる．しかし，多くのビジネスアナリスト（ビジネスにおける問題を分析し，解決に導く役割をもつ者）にとって LTL をベースにした数学的な記法を理解し使いこなすことは困難であり，検証したい性質を正しく記述することは難しい．不適切な検証結果を用いたビジネスプロセスの分析は現状を適切に捉えたものにはならず，ビジネスプロセスを改善することに繋がらないどころか望ましく

† 電気通信大学大学院情報システム学研究科，調布市
Graduate School of Information Systems, The University of Electro-Communications, 1–5–1 Chofugaoka, Chofu-shi, 182–8585 Japan
a) E-mail: hiroki.horita.is@vc.ibaraki.ac.jp
DOI:10.14923/transinfj.2017PDP0020

ない結果を招く可能性もある．そのため，数学的な記法に精通していない者でも検証したい性質を記述するための手法が求められる．

そこで，本研究(注1)では，LTL checker を用いて検証するために必要な論理式を自動生成するための手法を提案する．そのためにまず，本研究ではイベントログにおいて記録されているビジネスプロセスのイベント実行順序関係に着目した．イベント実行順序関係とは，イベントログにおいて記録されたイベントの実行順序，すなわち ⟨A,B,C⟩ という順番でイベントが実行された場合における「イベント A はイベント B よりも先に実行された」,「イベント B はイベント C よりも先に実行された」などの情報である．本研究では，このような情報を教師あり機械学習手法の一つである決定木 [6] で学習可能な形式へ変換するためのアルゴリズムを提案しており，ビジネスプロセスにおけるある性質の真偽によってイベントログを分類する．また，決定木の分類ルールを木構造で表す性質を活かし，その決定木から LTL checker で利用可能な論理式を自動生成する手法を提案している．本手法を用いることで，数学に精通していない場合でも検証すべき論理式を記述することができる．本論文では，電話修理プロセスを例に本手法を適用し，生成した論理式を使用して検証を行った際に高い精度を示せることを確認した．

本論文の構成は下記のとおりである．まず，**2.** にてプロセスマイニング及びビジネスプロセスにおける検証の課題を説明する．**3.** で提案手法について述べる．**4.** で提案手法の有用性を示すための評価実験を行った結果を示す．**5.** では本研究の適用対象や限界について記述する．**6.** では本研究をまとめ，今後の展望を述べる．

2. プロセスマイニング及びビジネスプロセスにおける検証の課題

プロセスマイニング技術は情報システムにおいて記録されたイベントログと呼ばれるデータを分析し，プロセスを発見，監視，改善することに活用できる [1]．本論文では，ビジネスプロセスの検証を対象としているため，ビジネスプロセスにおける「モデル」と「データ」の検証について以下に記述する．

ビジネスプロセスにおけるモデルを用いる検証は主にビジネスプロセスの設計時において，ペトリネットや Business Process Modeling and Notation (BPMN) 等を用いて設計したビジネスプロセスモデルが仕様を満たしているか検証するために行われる．モデル検査はその有力な手段であり，設計したモデルが望ましい性質を満たしたものであるのか自動的かつ網羅的に検証することができる．

一方で，プロセスマイニングの一部であるビジネスプロセスのデータを用いる検証は，ビジネスプロセス実行後に蓄積されたデータが望ましい性質を満たしているか確認するために行われる．データを分析することで，モデルを対象とした検証では知ることができないビジネスプロセスの実態を把握し，問題点を踏まえたビジネスプロセスの改善に繋げることができる．

2.1 LTL checker

LTL checker [2] はイベントログを対象とした検証を行うための有力な手法であり，時間的に変化する性質を記述可能な LTL をベースとした記法を用いて成り立つべき性質を記述することで，検証対象となるビジネスプロセスインスタンスが望ましい性質を満たしているか自動的かつ網羅的に検証することができる．

本研究では LTL checker をビジネスプロセスの検証に用いるので，例を用いて説明する．表 1 はビジネスプロセスの実行ログを表している．それぞれのログは ID とトレースと呼ばれる実行されたイベントシーケンスにより表される．それぞれのイベントは一つのアルファベット (A, B, C, D, E) によって表現される．例えば，ID 1 はイベント A，B，C，D がそれぞれ順番に実行されたことを表す．表 2 はビジネスプロセスにおいて成り立つべき性質を表している．それぞれの性質は自然言語と線形時相論理によって記述されている．

LTL checker は表 1 における各トレースが表 2 における性質を満たすか検証することができる．これらを検証すると，ID1，2 は全ての性質を満たすが（この

表 1　イベントログの例
Table 1　Examples of event logs.

Trace　ID	Trace
1	ABCD
2	ABED
3	ABC

表 2　性質の例
Table 2　Examples of properties.

Property	Formal Property
A が実行されたらいつかは B が実行される	$A \Rightarrow \diamond B$
いつかは D または E が実行される	$\diamond(D \vee E)$

(注1)：本研究は [4], [5] を発展させたものである．

ことを真 (true) と呼ぶ)，ID 3 は「イベント D または E がトレースにおいていつかは実行される」という性質を満たさない（このことを偽 (false) と呼ぶ)．この性質を満たすためにはイベント D またはイベント E のいずれかが実行されている必要があるが，ID 3 ではいずれも実行されていない．

2.2 関連研究

ビジネスプロセスのイベントログの検証に関する研究は，本研究のようにイベントログにおいて特定の性質が満たされているか確認するための検証を行う研究と，イベントログとビジネスプロセスモデルの差分を検出する研究に大別できる．

前者に分類される研究として, [7] では，LTL checker を拡張した semantic LTL checker を提案している．これは意味的に等しい要素をオントロジーを用いて表現することで，より効率的に検証することができる．また, [8] では，決定木を用いたビジネスプロセス実行時における LTL で記述したゴールの達成確率を求める手法を提案している．これらの研究は本研究と同様にビジネスプロセスの実行ログに対して LTL ベースの言語によって記述した性質を検証するために用いられるが，検証すべき性質を自ら記述できることが前提となっており，ビジネスプロセスの分析を行う際に数学的な知識の不足等の理由により検証すべき性質を記述できないという問題を解決する手段とはなっていない．また, [9] では，Declare という宣言的にビジネスプロセスにおける性質を記述できるモデリング言語を用いて検証するための性質を記述できる. [10] では，Declare を自動構築する手法が提案されている．様々な性質がテンプレートとして用意されているが，一つのテンプレートを用いて検証したい性質を記述できない場合はビジネスアナリストがどのテンプレートを組み合わせれば適切な性質を記述できるのか判断できる必要がある．一方で，本研究ではビジネスアナリストがイベントログが満たすべき性質を満たしているのかを手動で分類することができれば，イベントログにおけるイベントの実行順序関係を利用した決定木学習を利用して自動でビジネスアナリストが検証に利用する性質を生成することができる．

後者に分類される研究として, [11] 等ではイベントログとビジネスプロセスモデルとの差異を検出する適合性検査という手法に関する研究があげられる．これらの研究を用いることでイベントログを構成する各トレースがビジネスプロセスモデルのどの部分で再現できないかを把握することができるとともに，与えらえたモデルがイベントログに見られる挙動をどのくらいよく許容するかを判断する尺度であるフィットネスを計算することができる．これらはイベントログとビジネスプロセスモデルの全体的な差異を把握するためには有効だが，LTL checker を用いて行われる検証のように，特定の性質に着目した検証を行うことには不向きである．

3. 提案手法

本章では，図 1 の流れに沿って提案手法について説明する．図 1 では，角丸四角形は提案手法における各タスクに入出力される要素を表しており，だ円形は要素を処理する自動で行うタスクを表しており，人型のマークが付いただ円形は人手で行うタスクを表している．イベントログの検証を行うための論理式を生成するために，図 1 におけるタスクが行う処理と入出力について，**3.1** から **3.3** において説明する．

3.1 LTL による検証及び検証結果の確認

このプロセスでは，まず検証したい性質を人手で記述し，LTL checker を用いてイベントログを検証する（図 1 中，上部の LTL による検証）．その後，出力された真のトレースと偽のトレースの内容をトレースの種類ごとに数件から十数件ほど人手で確認し，問題がない場合は適切な性質が記述されていると判断し，このプロセスは終了となる．問題が見つかった場合，すなわち出力された結果が本来記述したかった性質を用いて検証した結果になっていない場合は，この問題を解決するためにイベント実行順序関係に着目した特徴量抽出を行う（**3.2**）．

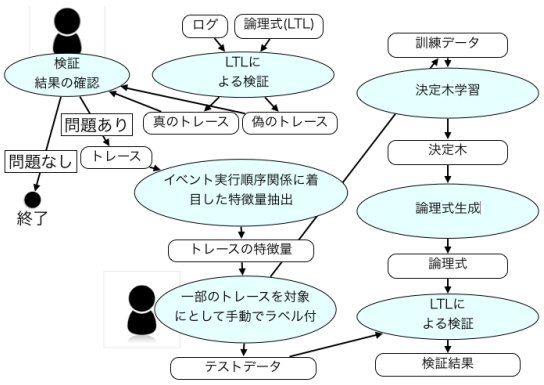

図 1 提案手法の概要
Fig. 1 Summary of our approach.

3.2 イベント実行順序関係に着目した特徴量抽出と決定木学習

本節は図 1 における「イベント実行順序関係に着目した特徴量抽出」,「決定木学習」に対応する．決定木学習を行うことで，望ましい性質を満たすための条件を抽出し，望ましい性質を反映した論理式の生成に活用することができる．決定木学習は教師あり学習技術であるため，予測モデルを構築するためには訓練データが必要となる．訓練データとしては検証したい性質が真となるトレースの一部と偽となるトレースの一部を使用し，人手で各トレースに対して検証したい性質を真に満たしているかどうかを表したクラスラベルを付ける．決定木学習のためには Classification And Regression Tree (CART) [6] というアルゴリズムを用いる．

各トレースの特徴量は各トレースを構成する各イベントの実行順序関係に着目して抽出する．ビジネスプロセスのトレースは実行されたイベントが順番に並んでいる配列型のデータであり，多くの機械学習アルゴリズムが利用するベクトル形式のデータではない．本研究で利用する決定木である CART もベクトル形式のデータを用いて学習を行う．それゆえ，配列型のデータとして表現されているトレースをベクトル形式のデータへ各トレースのイベント実行順序関係に着目して変換する．各トレースは複数のイベント実行順序関係へ変換され，各トレースにおけるイベント実行順序関係の集合が特徴集合となる．学習に使用するそれぞれのトレースはクラスラベルをもつ．それゆえ，各トレースはイベント順序に関する特徴とクラスラベルをもっている．各イベント名は一つのアルファベット（A, B, C 等）に置き換えられる．

表 3 は特徴ベクトルの例である．イベントログの特徴集合は全種類のイベント実行順序関係であり，特徴集合を構成する元の総数は全てのトレースにおけるイベント集合から二つ取る順列を計算することで得られる．これは以下の式により記述できる．

表 3 特徴量としてイベント順序関係をもったトレースの例
Table 3 Examples with event order relations of traces.

trace ID	trace	AB	AC	AE	AF	...	label
1	ABCEF	1	1	1	1	...	true
2	ABCD	1	0	1	0	...	false
3	ABBA	2	0	0	0	...	false

α : 全トレースにおけるイベントの組

特徴の総数 $= P(\alpha, 2)$

P(n, r) は n(n−1)(n−2)...(n−r+1) のように計算できる．次に，各トレースから特徴量を抽出する方法を説明する．例えば，表 3 においてトレース 1 では「ABCEF」という順番でイベントが実行されている．それぞれのアルファベットはイベントログにおいて記録されたイベントを表している．各トレースがもつ特徴の値がそれぞれ計算される．トレース 1 は幾つかのイベント実行順序関係をもっており，(AB, AC, AE, AF, BC, BE 等．これらは Algorithm 1 において FeatureListOfEachLog1 として記述される) これらは Algorithm 1 における「step 1 : 各トレースからイベント順序関係を抽出する」によって出力される．

Algorithm 1 の内容を説明する．このアルゴリズムは各トレースである log_i を入力とする．各 log_i において，$event_{ij}$ はイベント名とトレースにおけるイベントの場所を表す．例えば，$event_{i0}$ は log_i の最初に実行されたイベントであることを表す．各 $event_{ij}$ はトレースにおいて $event_{ij}$ の後に実行される $event_{ik}$ とペアにされる．この処理は 8 行目に書かれている．$event_{ij} + event_{ik}$ は $event_{ij}$ と $event_{ik}$ ペアとして統合することを表し，そのトレースにおいて両イベントがこの順番で実行されたという意味である．もしそのペアが特徴集合を構成するいずれかの元に等しければ，該当する値がインクリメントされる．全てのトレースに対して同様の手続きを行い，決定木学習のためのベクトル形式のデータを得て決定木を構築する．図 2 のフローチャートはこの流れを表している．

構築した決定木はクラスラベルが付けられていない各トレースの真偽を予測することができる．図 3 は決定木の例であり，この木は 2 回分岐している．根ノー

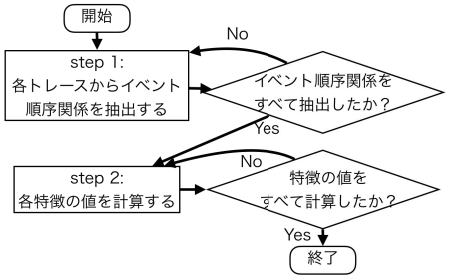

図 2 トレースの特徴量を得るためのフローチャート
Fig. 2 Flowchart for extracting features of a trace.

Algorithm 1 各トレースからの特徴量の抽出

```
Input:
  log_0, ..., log_n : 配列として表現されるトレース群
  log_i = ⟨event_i0, ..., event_im⟩: トレースはイベントの集まりによって構成される
  AllFeatures : すべてのイベント順序関係の集合
for i = 0 to n do
  //step 1: 各トレースからイベント順序関係を抽出する
  FeatureListOfLog_i = []
  Results = [] : 各ログの特徴を表すベクトルデータ
  for j = 0 to m − 1 do
    for k = j + 1 to m do
      FeatureListOfLog_i.append(event_ij+event_ik)
    end for
  end for
  //step 2:各特徴の値を計算する
  for L = 0 to last do
    for O = 1 to last do
      if AllFeatures_L == FeatureListOfLog_O then
        ValueOfFeatureListOfLog_O += 1
      end if
    end for
  end for
  Results.extend(FeatureListOfEachLog_O)
end for
Output:Results
```

ドでは分解条件は EH≦0.5 であり，これはもしあるトレースにおいて E→H の順でイベントが実行されたら False に分岐し（図 3 中右側），そうでないなら True（図 3 中左側）に分岐するということを表す．samples は決定木の構築に用いたトレースの数である．value は samples の true または false を表す各トレース数，左側の数値は false の数，右側の数値は true の数を表す．class は value における false と true の大小関係によって決定される．同様のことがより下のノードにおいても実行される．一つのクラスは各パスによって学習された結果であり，クラスラベルが 1 の場合と 0 の場合の両方の数は 50 (50, 48 + 2) である．これらの数字は図 3 におけるオレンジ色の葉ノードから 48,2 が得られる．また，青色の葉ノードから 50 が得られる．

3.3 LTL 式の生成

3.2 において記述した手法によって構築された決定木は論理式の生成に利用される．これはよりユーザの意図を正確に反映した論理式を生成するために行われ，より正確な検証に寄与する．

次に，決定木を論理式へ変換する方法を説明する．イベントログのイベント実行順序関係によって順次分岐していく前節の方法で構築された決定木を使用する．この決定木では，クラスラベルが 1 である（真である）

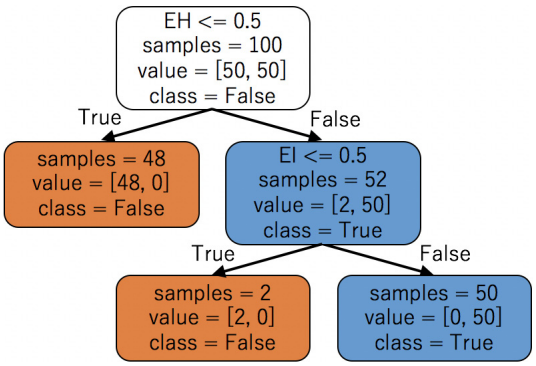

図 3 トレースのイベント実行順序関係に基づいて構築した決定木

Fig. 3 Decision tree based on event order relations of traces.

葉ノードに注目し，根ノードから葉ノードへ至る全ての条件分岐の連言をとる．条件分岐は図 3 の EH≦0.5 のように表現され，これはイベント E が実行された後にイベント H が実行されることを表す．もしこれが真ならば，論理式は ◇(E ∧ ◇H) となり，偽であれば，¬◇(E ∧ ◇H) となる．加えて，全てのパスの選言は ∨(a pass (class label = 1)) として表される．その結果，論理式はクラスラベルが 1 となる，すなわち望ましい性質を満たすための条件を全て網羅できる．以上のことから，決定木において，クラスラベルが 1 となる部分のみが論理式へ変換される．決定木から構築される論理式の一般形は (1) のとおりである．

$$\vee\{\wedge(\text{a pass(根ノードから class label} = 1 \text{ となる葉ノードへのパス))}\} \quad (1)$$

次に生成された論理式の一般形を図 4 を例に用いて説明する．図 4 は決定木であり，各アルファベットはノードの名前を表す．黒いノードは真のクラスラベルを表す．ノードに接続された左右のエッジは左側が始点となったノードの条件が True であることを表し，右側が始点となったノードの条件が False であることを表す．また，ノード A，B の下部に記述された True は決定木によって分類を行うとき，True に分類されることを表している．それゆえ，新しい論理式を作るためにパス (A → B → E)，パス (A → C → G) が使用される．A,B と A,C はそれぞれ連言によって接続される．真になるためのパスは二つであるため，これらのパスは選言によって接続される．このように，次の論理式が図 4 の論理式から構築される．

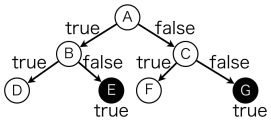

図 4　決定木の例
Fig. 4　Example of decision tree.

$$(A \wedge \neg B) \vee (\neg A \wedge \neg C) \tag{2}$$

この論理式を検証に用いることで，初期論理式を用いるよりも，よりユーザの真の意図を反映した結果が得られる．このフェイズは図 1 における下部の「LTL による検証」に対応する．

4. 評　　価

この章では，提案手法の有効性をケーススタディを行うことと，他手法と比較することによって示す．データセットと対応する論理式は **4.1** において説明する．**4.2** では提案手法により決定木を構築し，新たな論理式を生成する．**4.3** では人手で記述した論理式による検証，決定木による予測，生成した論理式による検証，他手法を用いた場合の正解率を比較する．

4.1　電話修理プロセス

我々はウェブ上で誰でも入手可能であり，幾つかの研究における評価で使用された実績のある電話修理プロセスのログ(注1)を評価のために用いる．このログは 1104 件のトレースにおける 12 種類のイベントから 11855 個のイベントが記録されており，各トレースは電話修理のプロセスを表している（登録，故障原因分析，修理結果の確認，各案件のアーカイブなどのイベントにより構成される）．

評価には電話修理プロセスに関する 4 種類の正しい論理式と間違った論理式のペアを用いた．次の論理式はその一部である．

correct : ◇ (activity==Repair(Complex)start
∧ ◇ (activity==Repair(Complex)complete
∧ ◇ (activity==Inform User
∧ ◇ (activity==Archive Repair)))) (3)

incorrect : ◇ (activity==Repair(Complex)start)
∧ ◇ (activity==Repair(Complex)complete)
∧ ◇ (activity==Inform User) (4)

Correct はユーザの意図を正確に反映した望ましい性質を表し，Incorrect はユーザの意図を正確に反映していない論理式であることを表す．本来検証に使用すべき論理式を正確に記述できていないという状況は不十分なドメイン知識や数理論理学の知識をもっている場合に起こり得る．両方の論理式は似ているが，これらはそれぞれ異なる意味をもつため，両方の検証結果は異なる結果を示す．Correct の論理式はイベント “Repair (Complex) start”(A) が実行されたらそのうちイベント “Repair (Complex) complete”(B) が実行され，いつかイベント “Inform User”(C) が実行され，いつかイベント “Archive Repair”(D) が実行されるという論理式であり，そのためこれらの四つのイベントについて許容される順番は一つしか存在しない（ABCD がこの順番で実行される）．ここで，許容されるとはすなわち検証した結果，真に分類されることである．一方で Incorrect の論理式はイベント ABC がそれぞれ実行されるというものである．この論理式はこれら三つのイベントについて実行されるべき順番を規定していないため，ABC,BAC,CBA 等様々な順番が許容される．更に Incorrect の論理式ではイベント D の実行については記述されていない．つまり，Incorrect の論理式にはイベント実行順序に関する性質と条件の見落としという二つの問題がある．これらの論理式は LTL checker を用いて検証される．これは図 1 において左上部に記述された「LTL による検証」に該当する．

4.2　構築された決定木と論理式

決定木を構築するために，イベントログを訓練データとテストデータに分割する．訓練データは真のログと偽のログからランダムでそれぞれ 50 件選択され，クラスラベルを手動で付ける．これらの訓練データを用いて決定木が CART アルゴリズム [6] によって scikit-learn [12] を用いて構築される．図 3 は訓練データから構築された決定木である．この決定木は **3.2** において説明されている．

次に，**3.2** において説明された決定木から構築された論理式を示す．この決定木から生成された論理式は (5) である．次節では，この論理式が十分であることを評価する．

(◇ ((activity == Repair(Complex)complete
∧ ◇ (activity==InformUser)))
∧ ◇ ((activity == Repair(Complex)complete
∧ ◇ (activity==ArchiveRepair)) (5)

(注1)：http://www.processmining.org/logs/start

4.3 提案手法と他手法の比較

この節では図 1 における我々の提案手法における各フェーズ間での比較及び他手法との比較を行う．表 4 は人手で記述した論理式を用いた検証結果である．表 5 は決定木を用いた場合の予測結果及び，生成した論理式を用いて検証した結果である．両者は同一の結果であった．本研究の結果である表 5 は表 4 よりも良い結果を示した．図 5 は人手で記述した論理式を用いた検証，決定木による予測，生成した論理式による検証，他手法の結果を比較したものである．他手法は我々が行った研究 [13] であり，決定木を予測のために使っているが，学習に用いる特徴集合が本論文の手法とは異なる．その特徴集合は適合性検査を行い，それぞれのイベント（invisible transition を含む）が各トレースにおいて実行されたか否かを表すものである．比較結果は正解率 (TP + TN) / (FP + FN + TP + TN) を比較したものである．その結果，本論文の手法（イベント実行順序関係に着目した決定木による予測，生成した論理式による検証）の正解率は人手で記述した論理式による検証や他手法を用いた場合よりも高いことが分かった．提案手法による検証結果は精度が 100%になっていないが，これは学習データに偏りがあり条件を網羅できなかったためである．精度を上げたい場合は学習データの数を増やすことが解決策として考えられる．このように特徴量としてイベントの実行順序関係を用いた決定木による予測や決定木の構造に基づく論理式の生成は有効であることが確認できた．

表 4　人手で記述した論理式を用いた検証結果
Table 4　Verification result using manually written logical formula.

		actual	
		T	F
classified	T	226	431
	F	0	447

表 5　決定木による予測結果及び，提案手法により生成した論理式を用いた検証結果
Table 5　Prediction result using decision tree and verification result using logical formula generated by our proposed approach.

		actual	
		T	F
classified	T	226	17
	F	0	861

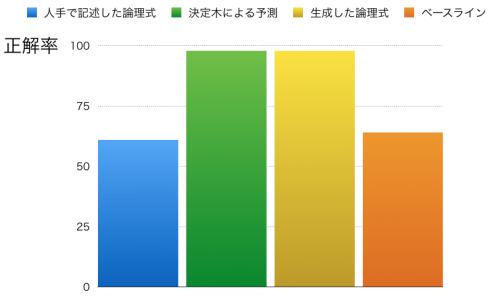

図 5　各ステップにおける比較と他手法との比較
Fig. 5　Comparison of a result of each step and the other approach.

4.4 その他のケーススタディ

本節では他のケーススタディの結果を説明する．(6) の論理式は使用した正確な論理式である．これは先ほど記述した正確な論理式と似ているが，“Archive Repair”が記述されていないという違いがある．それゆえ，これはより単純な例題であるといえる．一方で，不正確な論理式は (4) に記述したものと同じである．その結果は決定木による予測と新たに生成した論理式を用いた両方において完全な分類であることを示した．それゆえ，シンプルな論理式を用いた場合は本論文の手法はより正確に分類できることが分かった．また，このほかに行った例においても同様に **4.2** の例題と同等の高い正解率を示す論理式を生成できることが確認できた．

$$\boldsymbol{correct}: \Diamond(\text{activity==Repair(Complex)start} \wedge \Diamond(\text{activity==Repair(Complex)complete} \wedge \Diamond(\text{activity==Inform User}))) \quad (6)$$

5. 議論，限界

本論文においては，イベントログにおけるイベント実行順序関係に着目した学習を行うために決定木を使用した．決定木と同様に，サポートベクターマシン (SVM) やナイーブベイズ等の機械学習アルゴリズムを用いた場合も学習することができる．特にカーネル法 [14] と SVM は構造データを学習するための重要な方法である．カーネル法は構造データの種類（木，グラフ等）に応じたカーネルを用いることで明示的に特徴集合を構築することなく学習することができる．同一のデータに対してこれらのアルゴリズムを適用した結果，特に SVM（精度：97.9%（小数点第 2 位切り上げ））は決定木（98.5%（小数点第 2 位切り上げ））と同等の精度を示すことが確認できたが，これらは検証したい性質を記述した論理式を生成するという目的に

は不十分である．なぜなら，これらの手法は決定木のような木構造をもっておらず，論理式を生成できないからである．我々の手法では決定木の木構造によって表されたビジネスプロセスにおけるイベントの実行順序関係に関する条件を利用して論理式を生成している．

本手法によって生成した論理式を用いた検証は評価の結果，98.5%という高い精度でイベントログを分類できることが確認できた．しかし，生成できた論理式は理想的な論理式と比較すると望ましい性質を簡潔に記述したものにはなっていない場合がある．そのため，より適切な論理式を生成するための手法は今後の課題である．

また，本手法で生成できる論理式によって検証できる範囲には制限がある．本手法ではイベントログにおけるイベント実行順序関係に着目して学習するための特徴集合を構築した．そのため，生成できる性質はシーケンスフローに関するものに限定される．ビジネスプロセスに関係する性質にはイベントを実行した時間に関する性質やイベントを実行した主体に関する性質など様々な属性に関するものがあるため，提案手法の適用範囲を拡大することは今後の課題である．

本論文においてはスケーラビリティについて定量的な評価は行っていない．しかし，ビジネスプロセスに関するイベントログはビッグデータと呼ばれるような非常に大規模なデータとはなりづらい．4TU.Datacentrum(注2)ではイベントログの実データが公開されているが，大きいサイズのものでもトレース数にして 10 数万ほどのサイズであり，イベントの種類も限られているため，実用上利用できないほどの学習時間はかからないと考えられる．

6. むすび

本論文では，ビジネスプロセスに関する実行ログのイベント実行順序関係に着目し，決定木学習を行うことで，ビジネスプロセスにおいて検証したい性質を自動生成する手法を提案した．電話修理プロセスのイベントログを題材に提案手法を適用し，高い精度で分類可能な論理式を生成できることを示した．今後は，より正確で簡潔な論理式を生成する方法やイベント実行した日時やイベントを実行した主体等の属性に関する性質を取り扱えるように本手法を拡張することである．

(注2)：http://data.4tu.nl/repository/collection:event_logs_real

文　献

[1] W.M.P. van. der. Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.

[2] W.M.P. van. der. Aalst, H.T. de. Beer, and Boudewijn F. van Dongen, “Process mining and verification of properties: An approach based on temporal logic,” On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, pp.130–147, 2005.

[3] A. Pnueli, “The temporal logic of programs,” In Proc. 18th Annual Symposium on Foundations of Computer Science 1977, pp.46–57, IEEE Computer Society 1977.

[4] H. Horita, H. Hirayama, T. Hayase, Y. Tahara, and A. Ohsuga, “Process Mining Approach Based on Partial Structures of Event Logs and Decision Tree Learning,” Proc. 5th IIAI International Congress on Advanced Applied Informatics, (AAI’16), pp.113–118, 2016.

[5] H. Horita, H. Hirayama, T. Hayase, Y. Tahara, and A. Ohsuga, Computer and Information Science – Studies in Computational Intelligence 656– (chapter 7), pp.89–103, Splinger, 2016.

[6] W. Loh, “Classification and regression trees,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol.1, no.1, 2011.

[7] A.K.A. de. Medeiros, W.M.P. van. der. Aalst, and C. Pedrinaci, “Semantic process mining tools: Core building blocks,” Proc. 16th European Conference on Information Systems, (ECIS’08), pp.1953–1964, 2008.

[8] F.M. Maggi, C.D. Francescomarino, M. Dumas, and C. Ghidini, “Predictive Monitoring of Business Processes,” Proc. 24th International Conference on Advanced Information Systems Engineering, (CAiSE’14), pp.457–472, 2014.

[9] W.M.P. van. der. Aalst, M. Pesic, and H. Schonenberg, “Declarative workflows: Balancing between flexibility and support,” Computer Science - Research and Devel- opment, vol.23, no.2, pp.99–113, 2009.

[10] F.M. Maggi, R.P.J.C. Bose, and W.M.P. van. der. Aalst, “Efficient discovery of understandable declarative process models from event logs,” Proc. 24th International Conference on Advanced Information Systems Engineering, (CAiSE’12), pp.270–285, 2012.

[11] A. Rozinat and W.M.P. van. der. Aalst, “Conformance checking of processes based on monitoring real behavior,” Inf. Syst., vol.33, no.1, pp.64–95, 2008.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., “Scikit-learn: Machine learning in python,” J. Mach. Learn. Res., pp.2825–

2830, Oct. 2011.
[13] H. Horita, H. Hirayama, Y. Tahara, and A. Ohsuga, “Goal achievement analysis based on LTL checking and decision tree for improvements of PAIS,” Proc. 31st Annual ACM Symposium on Applied Computing, (SAC’16), pp.1214–1218, 2016.
[14] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.

（平成 29 年 6 月 7 日受付，10 月 2 日再受付，
12 月 4 日早期公開）

堀田　大貴

1989 年生．2017 年電気通信大学大学院情報システム学研究科社会知能情報学専攻博士後期課程修了．博士（工学）．同年，茨城大学工学部情報工学科助教．ソフトウェア工学，ビジネスプロセスマネジメントの研究に従事．情報処理学会会員．

平山　秀昭

1958 年生．1981 年慶應義塾大学工学部管理工学科卒．同年（株）東芝入社．2001 年電気通信大学大学院情報システム学研究科博士後期課程了．2003 年より東芝ソリューション（株），2015 年より東芝ソリューション販売（株）に所属．2014 年より電気通信大学協力研究員．2015 年より東京電機大学情報環境学部非常勤講師．博士（工学）（電気通信大学）．並列分散処理，ソフトウェア工学等の研究に従事．情報処理学会，電子情報通信学会会員．

早瀬　健夫

1967 年生．1990 年 3 月上智大学理工学部機械工学科卒業．1992 年 3 月同大学大学院理工学研究科機械工学専攻修士課程修了．同年 4 月（株）東芝入社．現在，同社ストレージ＆デバイスソリューション社半導体研究開発センター ソフトウェアソリューション技術開発部に所属．2016 年より電気通信大学協力研究員．博士（工学）（上智大学）．ソフトウェア工学の研究に従事．ACM，IEEE，電気学会，情報処理学会，人工知能学会各会員．

田原　康之

1966 年生．1991 年東京大学大学院理学系研究科数学専攻修士課程修了．同年（株）東芝入社．1993～1996 年情報処理振興事業協会に出向．1996～1997 年英国 City 大学客員研究員．1997～1998 年英国 Imperial College 客員研究員．2003 年国立情報学研究所入所．2008 年より電気通信大学准教授．博士（情報科学）（早稲田大学）．エージェント技術，及びソフトウェア工学などの研究に従事．情報処理学会，日本ソフトウェア科学会会員．

大須賀昭彦　（正員）

1981 年上智大学理工学部数学科卒．同年（株）東芝入社．同社研究開発センター，ソフトウェア技術センター等に所属．1985～1989 年（財）新世代コンピュータ技術開発機構 (ICOT) 出向．2007 年より電気通信大学．現在，同大学大学院情報理工学研究科教授．2017 年より同大学大学院情報システム学研究科研究科長併任．2012 年より国立情報学研究所客員教授兼任．工学博士（早稲田大学）．情報処理学会フェロー．ソフトウェア工学，エージェント，人工知能の研究に従事．1986 年度，2016 年度情報処理学会論文賞，2013 年度人工知能学会研究会優秀賞，2014 年度同学会功労賞受賞．IEEE Computer Society Japan Chapter Chair，人工知能学会理事，日本ソフトウェア科学会理事，同学会監事等を歴任．情報処理学会，電子情報通信学会，人工知能学会，日本ソフトウェア科学会，電気学会，IEEE Computer Society 各会員．