社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

トレースの例示による線形時相論理式の生成

小松　航汰†　　堀田　大貴†

† 茨城大学大学院理工学研究科, 日立市

あらまし　プロセスマイニングによるデータ分析を行うことで，ビジネスプロセスの効率的かつ網羅的な分析を支援することができる．プロセスマイニングの宣言的手法では，ユーザが線形時相論理（LTL）に基づいた正しい記述の論理式を与えることで，ビジネスプロセスにおける目的の性質を検証できる．しかし，多くのユーザにとって 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，検証したい性質を記述することは容易ではない．このようなユーザによるビジネスプロセスの分析は不適切な検証結果となる可能性がある．そのため，数学的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，本研究では充足可能性問題（SAT）や決定木学習 (DT) をベースとした手法を利用し，ビジネスプロセス上の時間的性質を XES 形式で保存されている一般的なイベントログの例から学習できるか検証した．また，本手法を用いて生成される論理式の表現能力についての調査も行った．本研究の妥当性を示すために，3 つの評価実験を行った．実験結果から本手法の一般的なイベントログに対する有効性と生成された論理式の表現能力の特徴を示すことができた．

キーワード　プロセスマイニング, ビジネスプロセス, 充足可能性問題, 決定木学習

Generating LTL formulas with Trace Examples

Kota KOMATSU† and Hiroki HORITA†

† Graduate School of Science and Engineering, Ibaraki University, Hitachi-shi

Abstract　In declarative process mining, a user can verify a desired property in a business process by providing an LTL formula with a correct description.However, it is difficult for many users to describe the properties they want to verify using mathematical notation based on LTL. Therefore, in this study, we use a method based on the satisfiability problem (SAT) and decision tree learning (DT) to verify whether temporal properties in business processes can be learned from general event log examples. The experimental results show the effectiveness of our method on general event logs and the characteristics of the learned properties.

Key words　Process Mining, Business Process, Satisfiability Problem, Decision Tree Learning

1. ま え が き

近年，情報技術の普及により，多くのデータが利用可能となっている．これらのデータを利用し，ビジネスプロセスの自動化や効率化に取り組む企業が増えてきている．プロセスマイニング [10] は，ビジネスプロセスの処理パターンをイベントログデータの蓄積により可視化し，改善点を具体的に特定することで業務効率化を支援するデータ分析手法である．このようなデータ分析に基づく手法を用いることで，データ分析に基づかない手法と比べ，より効率的かつ網羅的にビジネスプロセスの分析を支援することができる．

プロセスマイニングにおける宣言的手法は検証に基づく手法であり，高い複雑性と変動性を特徴とするプロセスにおいて本質的な性質に焦点を当てた分析を実現できる [5]．このようなビジネスプロセスの本質的な性質に焦点を当てた分析を実現するために，Declare などのモデリング言語を用いたプロセスマイニング技術が数種類提案されてる．

既存の宣言的手法は，ユーザがイベントログからビジネスプロセスの改善に必要な情報を抽出するために，正しい論理式の記述とその性質の理解を前提としている．しかし多くのユーザにとって，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容易ではない．このようなユーザによるビジネスプロセスの分析は不適切な検証結果となる可能性がある．そのため，プロセスマイニングにおける宣言的手法や論理式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．

この問題に対処するための手法として，教師あり機械学習手法の一つである決定木を用いてビジネスプロセスのイベントログから論理式を自動生成することで，検証したい性質を正しく記述する手法がある [9]．この手法を用いることで，LTL をベー

スとした数学的記法に精通していない者でも検証したい性質を論理式として記述することができる．しかし，この手法で生成した論理式は理想的な論理式と比較すると，目的の性質を簡潔に記述したものではない場合がある．

目的の性質を示す論理式を自動生成する手法として，Neiderら [7] が提案した充足可能性問題（SAT）や決定木学習 (DT) をベースとした手法がある．この手法では，サンプル $\mathcal{S}$ という入力値を与えることで，その記述を満たす時間的性質を最小サイズの論理式として自動生成することができる．サンプルは，学習する時間的性質を満たすトレース群（正のトレース）と満たさないトレース群（負のトレース）から成る．しかし，この手法は XES 形式 [2] などで保存されている一般的なイベントログ [12] を例として与えることを考慮していない．

そこで，本研究では充足可能性問題（SAT）や決定木学習 (DT) をベースとした手法 [7] を利用し，一般的なイベントログの例からビジネスプロセス上の時間的性質を簡潔に記述できるか検証した．生成された論理式の表現能力は，ユーザに与えられた LTL のテンプレートに依存しないため，ほぼ無制限である．そのため，本手法を用いて生成される論理式の表現能力についての調査も行った．入力値として与えるサンプルにおけるトレース群の記述方法は一般的なイベントログの記述方法とは異なり，0 と 1 の数値列で記述されている．一般的なイベントログに対しての検証を行うために，イベントログをサンプルで用いる数値列形式に変換するツールを実装した．このツールを用いることでユーザは任意のイベントログからサンプルの選択を行い，SAT,DT ベースの手法で利用することができる．

本論文の構成は下記の通りである．まず，第 2 章では本論文に関連する基礎知識について説明する．第 3 章では本論文に関連する研究を紹介する．第 4 章では本研究における調査手法について述べる．第 5 章では調査手法に沿って行った評価実験の結果及び考察を示す．第 6 章では本論文をまとめ，今後の課題について述べる．

2. 関 連 知 識

本章では提案手法に関連する知識について説明する．

2.1 線形時相論理

線形時相論理（LTL）とは，時間に関する様相を持つ様相時相論理である．様相（modal）とは，「〜は必然的に真（true）」や「〜は可能である」といった必然性や可能性などを指し，時相論理（Temporal Logic）とは，時間との関連で問題を理解し表現するための規則と表記法の体系を指す． LTL ではある条件が最終的に真であるかなどの将来の出来事について論理式で表すことができる．

LTL では変項 $p_1, p_2, \ldots$ や一般的な論理作用素である $\neg, \vee, \wedge, \rightarrow$ に加えて，N(next)，G(globally)，F(finally)，U(until)，R(release) の時相様相作用素を使用する． N，G，F の 3 つの作用素は単項演算であるため，ϕ が論理式であれば，$N\phi$ も論理式である． U，R の 2 つの作用素は二項演算であるため，ϕ と ψ が論理式であれば，$\phi U \psi$ も論理式である．

表 1　イベントログの例

トレース ID	トレース
1	ACEF
2	ABCF
3	ADFE

表 2　発見できる制約（論理式）の例

論理式	性質
$\Diamond A$	いずれ A が実行される
$\Box\ (A \rightarrow \Diamond F)$	A が実行されたら，いずれ F が実行される

2.2 宣言的プロセスマイニング

宣言的プロセスマイニング [5] とは，宣言型モデリング言語に基づくプロセスマイニング技術を指す．具体的には，イベントログを入力として与えた場合，ユーザが必要とするタスク間の制約関係を出力として得ることができる．イベントログと発見できる制約（論理式）について，宣言的プロセスマイニングにおける標準的なツールである Declare [6] を用いて説明する．Declare とは，LTL に基づいた形式的な意味論を組み合わせた宣言型モデリング言語であり，Declare モデルはテンプレートに基づいた制約のセットで構成されている [6]． Declare のテンプレートは関係，否定，存在の 3 種類に分類され，ユーザは任意のテンプレートとパラメータを選択することで，イベントログにおけるテンプレートに沿った性質を特定することができる [4].

表 1 はビジネスプロセスのイベントログを表している．イベントログは ID とトレースと呼ばれる実行された一連のイベント群により表される．それぞれのイベントはアルファベット（A，B，C，D，E，F）によって表される．例えば，トレース ID が 1 のトレースはイベント A，C，E，F がそれぞれ順番に実行されたことを表している．表 1 のイベントログに対して Declare を用いた場合に発見できる制約の例とその性質を表 2 に示す．「$\Diamond A$」は existence のテンプレート，「$\Box\ (A \rightarrow \Diamond F)$」は response のテンプレートをそれぞれユーザが指定した場合に発見できる制約である．

3. 関 連 研 究

本章では本研究に関連する研究を紹介する．

3.1 SAT,DT ベースで論理式を生成する手法

分散システムのデバッグ，マルウェアやウイルスなどのリバーズエンジニアリングなど複雑なシステムの挙動を理解することは，適切なツールによるサポート無しでは困難である．そこで，Neider ら [7] はユーザが複雑なシステムの動的な挙動を理解するために，LTL 式を学習する 2 つの新しいアルゴリズムを提案した．第 1 の学習アルゴリズムは充足可能性問題（SAT）をベースとしたアルゴリズムであり，第 2 の学習アルゴリズムは SAT ベースのアルゴリズムを基に考案した決定木学習（DT）ベースのアルゴリズムである．これらのアルゴリズムはサンプル $\mathcal{S}$ を入力値とする．サンプルは，学習する LTL 式の性質を満たすトレース群（正のトレース）と満たさないトレース群（負のトレース）から成る．詳細なサンプルの構成については，第

表 3　検証する性質の例

論理式	性質
$\Diamond A$	いずれ A が実行される
$\Box\ (A \rightarrow \Diamond C)$	A が実行されたら，いずれ C が実行される

4 章において説明する．

SAT ベースの学習アルゴリズムは SAT ソルバを用いて，与えられたサンプルの記述を満たすような LTL 式を構成するアルゴリズムである．SAT ベースの学習アルゴリズムの特徴として，最小サイズの論理式を学習できること，先験的に与えられたテンプレートのセットに依存しないことが挙げられる．

DT ベースの学習アルゴリズムは SAT ベースの学習アルゴリズムをサンプルの小さな部分集合に対して実行し，各部分集合から得られた LTL 式を特徴量として決定木を生成，生成した決定木から解となる LTL 式を得るアルゴリズムである．DT ベースの学習アルゴリズムは SAT ベースの学習アルゴリズムと同様な特徴を示す．

これらの学習アルゴリズムは目的の論理式を自動生成する入力値をベンチマークとして形成するために，大幅な時間を費やしている [7]．また，入力値を生成するためにはユーザが LTL をベースとした数学的記法に精通していることが前提である [7]．そのため，多くのユーザは Neider らの提案手法において，XES 形式などの一般的なイベントログを用いてデータ分析を行うことが困難である．

3.2　トレース抽出技術

LTL Checker [1] はイベントログを対象とした検証を行うための 1 つの手法であり，LTL をベースとした記法を用いてユーザが目的とする時間的に変化する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目的の性質を満たしているか自動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 3 はビジネスプロセスにおいてユーザの目的とする性質を表している．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る．検証対象となるビジネスインスタンスは第 2 章で紹介した表 1 を用いる．

LTL Checker は表 1 における各トレースが表 3 における性質を満たすか検証することができる．実際に検証した結果，ID が 1，2 のトレースは全ての性質を満たすが，ID が 3 のトレースは「A が実行されたら，いずれ C が実行される」という性質を満たさない．ID が 1，2 のトレースのように，検証する全ての性質を満たしたトレースを真（true）のトレースと呼び，ID が 3 のトレースのように，検証する一部の性質を満たさないトレースを偽（false）のトレースと呼ぶ．

4.　本研究における調査手法

本章では，既存研究である SAT，DT ベースの手法 [7] を一般的なイベントログに対して実行し，生成できる論理式の表現能力について調査する方法を説明する．本手法の概要を図 1 に示す．図 1 では，角丸四角形は提案手法における入出力の要素を表しており，楕円形は要素を処理するタスクを表している．それぞれのタスクは 1 から 4 の順序で実行する．

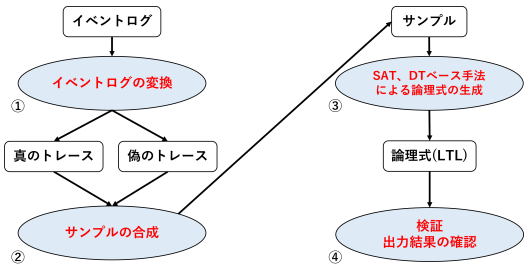

図 1　本研究における調査手法の概要

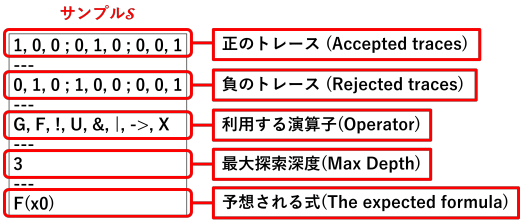

図 2　サンプル $\mathcal{S}$ の構成

4.1　イベントログの変換

SAT，DT ベースで論理式を生成する手法 [7] では，入力値としてサンプル $\mathcal{S}$ を使用するため，検証で使用するイベントログをサンプルに変換する必要がある．具合的なサンプルの構成を図 2 として示す．正のトレースは生成する論理式の性質を満たすトレース群である．負のトレースは生成する論理式の性質に違反するトレース群である．利用する演算子は論理式を生成する際に利用できる演算子である．最大探索深度は SAT ベースのアルゴリズムにおいて論理式を生成するための最大の探索深度（式のサイズ）である．予想される式では正と負のトレースから生成できる論理式のヒントを与えることができる．サンプルは正のトレースと負のトレースが記述されていれば入力値として利用できるため，「- - -」で区切られた 5 つの要素をすべて満たす必要はない．本研究では，サンプルのパラメータ指定により生成できる論理式への影響が未知のため，正と負のトレース以外は基本的にデフォルト値を利用する．

正と負のトレースはともに「,」でイベントの真偽が区切られており，「1」の場合が真，「0」の場合が偽であり，「;」でタイムスタンプが区切られている．トレースの各タイムスタンプにおいて，実行されたイベントは真（1）であり，実行されなかったイベントは偽（0）である．1 個のトレースは改行によって区別することができ，複数のトレースを正と負のトレースとして扱うことができる．つまり，XES 形式で保存されているイベントログを 0 と 1 の配列にトレース単位で変換する必要がある．

本研究では，イベントログをトレース単位で 0 と 1 の配列に変換するツールの実装を行い，そのツールを用いてサンプルの生成を行った．このツールではイベントログ全体で利用されるクティビティに識別子（ID）を割り当て，割り当てた ID に基づいて真（1）か偽（0）か判別する．さらに，イベントの実行順序をタイムスタンプと捉え，「;」で区別する．例えば，「イベン

ト 1 → イベント 2」のようなイベントの遷移に従ったサンプルの記述は「イベント 1 が真（1，0）; イベント 2 が真（0，1）」である．また，「;」で区切られたタイムスタンプの 1 区間において，真（1）となるイベント 1 つのみであり，それ以外のイベントは全て偽（0）であるものとする．なお，このツールで変換したトレースは全て有限である．

4.2 サンプルの選択

本研究では，サンプルを選択する際に 2 つの手法を用いる．1 つは真と偽のトレースを全て利用してサンプルを選択する手法であり，もう 1 つはユーザが任意で真と偽のトレースを選択し，元のトレース数より少ないトレース数でサンプルを選択する手法である．前者の手法を実現するために，正と負のトレースに該当するイベントログを指定することで，サンプルを選択するツールを実装した。最初に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．次に出力された真のトレース，偽のトレースをそれぞれ保存する．最後に実装したツールを利用することで，LTL Checker で分類した真と偽のトレースを 0 と 1 の配列と対応付けてサンプルを選択することができる．この 2 つの合成手法を用いることで，サンプルを Neider ら [7] が提案したツールに与えたときに生成される論理式の表現能力の調査が可能である．

4.3 SAT,DT ベースの手法による論理式の生成

4.2 節において，選択したサンプルを Neider ら [7] が提案したツールに与えることで，サンプルの性質を満たす論理式を自動生成できる．生成された論理式は，自動生成できる解の候補のうち，最も小さいサイズ（探索深度）のものである．そのため，目的の性質を示す論理式に近いサイズの式を得るためにパラメータ操作をする必要がある． DT ベースの手法では，ツールを実行するたびに構成される決定木がわずかに変化するため，実行ごとに生成される論理式が異なる場合がある．既存研究では，DT ベースの手法が SAT ベースの手法より性能面で優れているとされていた [7] が，SAT ベースの手法ではパラメータ操作により得られる解への影響が大きいため，本研究では両手法を利用する．したがって，本研究では SAT ベースの手法では最小の探索深度を変化させながら実行し，DT ベースの手法では同一条件で数回実行する．

4.4 出力結果の確認

4.3 節において生成された論理式の記録，分析を人手で行う必要がある．

4.4.1 生成された論理式の性質

数理論理学において，論理の体系化には「意味論（セマンティクス）」と「構文論（シンタクス）」の 2 つの立場がある．意味論では，論理式に対する適当な解釈に注目し，構文論では，論理式の構文上の構造に注目する．つまり，論理式の解釈を明確にすることで，論理式の性質を意味的に理解することができる．恒等式や真理値表を利用することで論理式の性質を解釈できるが，最終的には分析者の判断により論理式の意味的性質を解釈しなければならない．そのため，本研究では次節（4.4.2）で示す手法を用いてトレースに基づいた論理式の性質を検証する．

4.4.2 生成された論理式による検証

本手法では，元のトレース数より少ないトレース数によって生成できる論理式を LTL Checker の入力値とし，どのように元のイベントログを検証できるか調査する．これは，LTL ベースの数学的記法に精通していない者がサンプルを選択する際に少ないトレース数のほうがツールを容易に扱えるという仮定に基づく．また，業務ログなどの大規模なイベントログにおいては，少ないトレース数で目的の性質を発見できれば，ツールの実行時間の削減が望める．したがって，「検証したい性質で分類した結果」と「元のトレース数より少ないトレース数で生成した論理式が分類した結果」を比較して，トレースをどの程度同じように分類しているかを検証する．

5. 実　　験

本章では実験の目的と概要，実験で用いるツール及びイベントログについて説明し，実験結果を示し，最後に考察を行う．

5.1 実験の目的と概要

LTL ベースの数学的な記法に精通していない者が検証したい性質を簡潔に記述するために，SAT，DT ベースで論理式を生成する手法 [7] が一般的なイベントログに対して利用できることを示す必要がある．具体的には， XES 形式で保存されているイベントログを用いて論理式を自動生成することが可能であるかを示す．また，生成された論理式がどのような性質を示すかも調査する．Neider ら [7] が提案した手法が一般的なイベントログに対して利用できることや生成できる論理式の表現能力を示すために，以下のような未知の課題を解く必要がある．

RQ1　LTL テンプレートの論理式を生成できるか？

RQ2　どれくらい少ないトレース数で目的の性質を示す論理式を生成できるか？

RQ3　少ないトレース数で生成した論理式は、元のトレースをどのように分類するのか？

RQ1 から RQ3 に解答するために，本研究では第 4 章の調査手法に沿った 3 つの実験を行った．

5.2 実験で用いるツール及びイベントログ

本実験では ProM(Process Mining Framework) [11] と SAT，DT ベースで論理式を学習するツール[1]，さらに本研究で実装したツールを利用した．ProM のプラグインである LTL Checker でイベントログを検証し，変換ツールを用いてサンプルを選択，選択したサンプルを入力値として SAT，DT ベースで論理式を生成する実験を行った．

実験ではトレース数が 100 個，イベント数が 2297 個の合成データを利用した．これはウェブ上でだれでも入手可能である ProM のチュートリアルで扱う演習ファイルである[2]．

5.3 実 験 結 果

5.3.1 実　験　1

本実験では，線形時相論理（LTL）のテンプレートとして，宣

(注1)：https://github.com/ivan-gavran/samples2LTL
(注2)：http://promtools.org/doku.php

表 4　実験 1 の実行結果

名称	生成された式	生成結果
existence(A)	$\Diamond A$	○
absence(A)	$\neg(\Diamond A)$, $\neg(\Box A)$	○
responded existence(A,B)	$\Diamond B$	×
co-existence(A,B)	$\Diamond BUA$, $\Diamond A$	○
response(A,B)	$\Diamond AUB$	○
precedence(A,B)	$\Diamond B \rightarrow \Diamond A$	×
succession(A,B)	—	T
chain response(A,B)	$\Diamond(A \wedge \bigcirc B)$	○
chain precedence(A,B)	$\bigcirc(\bigcirc C)$	×
chain succession(A,B)	—	T
not co-existence(A,B)	$\Diamond C \rightarrow \Diamond D$	×
not succession(A,B)	$\Diamond C \rightarrow \Diamond D$	×
not chain succession(A,B)	$\neg(\Diamond AUB)$	○

言的プロセスマイニングの一般的な手法である Declare のテンプレート [6] を復号できるか検証した．ここでの「復号」は，同一の性質を持つ論理式であるかを真理値表や恒等式をもとに，分析者によって意味論的に判断することである．本実験では，「init」と 3 つの「alternate」を除いた Declare テンプレートを対象とした．また，SAT ベースの手法では最小の探索深度を 1 から 5 の値で変化させながら実行し，DT ベースの手法では同一条件で 3 回実行した．それぞれの実行は実行時間 30 分でタイムアウトとした．

実行結果を表 4 に示す．生成結果は，復号が成功した場合は「○」，失敗した場合は「×」，タイムアウトとなった場合は「T」で表している．今回の実験では 13 種類のテンプレートの復号を検証し，6 種類のテンプレートの生成に成功した．また，5 種類のテンプレートは生成に失敗し，2 種類のテンプレートは実行時間が 30 分を越えたため，タイムアウトとなった．生成された論理式はテンプレートの構文式と一致するものがほぼなかった．

5.3.2 実　験　2

本実験は，元のトレース数より少ないトレース数から成るサンプルを使用し，論理式の自動生成を行った．目的の性質を示す論理式が生成されるまで，トレースを一定個数ずつ増加させて実行した．この実験では，サンプルのトレース数が全体のトレース数の 5 分 1 に達した時点で目的の性質を表現できないものと考え，実行を終了した．トレース数の増加幅は正と負のトレース 1 個ずつとし，計 2 個ずつトレースを増加させながら繰り返し実行を行った．検証する性質は実験 1 において生成結果が「○」であったテンプレートに対してにのみ実行を行った．また実験 1 と同様に，SAT ベースの手法では最小の探索深度を 1 から 5 の値で変化させながら実行し，DT ベースの手法では同一条件で 3 回実行した．

実行結果を表 5 に示す．トレース数は目的の性質を発見した際のサンプルに含まれる正と負のトレースの合計数を表す．結果として，response 以外の 5 つのテンプレートは，元のトレース数より少ないトレース数で選択したサンプルを使用して目的の性質を示す論理式の自動生成が可能であった．

表 5　実験 2 の実行結果

名称	生成された式	トレース数	生成結果
existence(A)	$\Diamond A$	6	○
absence(A)	$\Box(\neg A)$	4	○
co-existence(A,B)	$\Diamond BUA$	8	○
response(A,B)	$\Diamond(C \vee B)$	20	×
chain response(A,B)	$\Box(A \rightarrow \bigcirc B)$	14	○
not chain succession(A,B)	$\neg(\Diamond AUB)$	16	○

表 6　response に関する一致率の検証結果

検証する式	真のトレース数	偽のトレース数	一致率
$\Box(A \rightarrow \Diamond B)$	84	16	100%
$\Diamond C$	56(56)	44(16)	72%
$\Diamond(C \vee D)$	83(76)	17(9)	85%
$\Diamond(B \vee C)$	84(84)	16(16)	100%

表 7　co-existence に関する一致率の検証結果

検証する式	真のトレース数	偽のトレース数	一致率
$\Diamond A \leftrightarrow \Diamond B$	39	61	100%
$\Diamond A$	49(39)	51(51)	90%
$\Diamond BUD$	39(12)	61(34)	46%
$\Diamond EUA$	39(39)	61(61)	100%

5.3.3 実　験　3

実験 2 の response と co-existence の実行結果を用いて，少ないトレース数で生成した論理式は，元のトレースをどのように分類するのかを検証した．具体的には，合成データの全てのトレースに対して各テンプレートで検証した結果と生成された論理式で検証した結果を比較し，真偽の分類結果が一致しているトレースを確認した．分類結果が一致しているトレース数を全体のトレース数 (今回は 100) で割った結果を「一致率」として定量的に評価を行った． response に関する実行における各論理式の検証結果を表 6，co-existence に関する実行における各論理式の検証結果を表 7 に示す．真のトレース数は検証する論理式が真であるトレースの数を表し，偽のトレース数は検証する論理式が偽であるトレースの数を表す．丸括弧内に記述されている数値は分類結果が一致しているトレース数を表している．

結果として，response，co-existence に関する実行で生成された論理式は高い一致率を示した．さらに，response に関する実行では「$\Diamond(B \vee C)$」，co-existence に関する実行では「$\Diamond EUA$」の一致率が 100% を示した．

5.4 考　　察

5.4.1 RQ1

実験 1 の結果 (表 4) より，LTL のテンプレートの復号は困難であると言える．特に，「precedence」の性質を示す論理式の生成に失敗，またはタイムアウトとなっている．このように目的の性質を示さない原因として，正のトレースに含まれる一部のトレースが空虚 (vacuity) に満たされている [3] ことが考えられる．例えば，「$A \rightarrow B$」のような含意（→）を含む論理式を用いてイベントログを検証したとき，イベント A が存在しないトレースは全て真であるとみなされる．この場合，イベント A が

存在しないトレースに対して論理式が空虚に満たされていると言う．したがって，本実験では空虚性の検出を考慮せず，サンプルの合成を行ってしまったため，自動生成される論理式の表現幅が広がってしまったと考えられる．これは，本実験のように目的の性質を復号する際に大きな影響を受けるが，任意に与えたトレースから未知の論理式を生成する場合はほぼ影響がないと考えられる．

5.4.2 RQ2

実験 2 の結果 (表 5) より，元のトレース数の 5 分の 1 以下のトレース数から成るサンプルを利用して論理式の自動生成を行った場合でも，目的の性質を示す論理式の生成が可能であると言える．実験 2 では response 以外のテンプレートに関する実行おいて，目的の性質を示す論理式を生成することができた．このような結果は，5.4.1 節と同様に正のトレースに含まれる一部のトレースが空虚に満たされていることが原因であると考えられる．さらに，本実験では正と負のトレースをランダムに選択してサンプルを合成しているため，トレースが一定個数に達するまで空虚に満たされているトレースのみを選択する可能性がある．実験 2 の response に関する実行では，サンプルに含まれるトレースの総数が 8 に達するまで目的の性質を示すトレースを含まなかった．また，ユーザによる検証するイベントの指定も論理式の自動生成に大きな影響を与えていると考えられる．本研究では，合成データのワークフロー [8] を参考に目的の性質を示す実行順序のイベントを指定したが，指定するイベントによっては目的の性質を示す検証を行うことができない．

5.4.3 RQ3

実験 3 の結果 (表 6，表 7) より，少ないトレースで生成した論理式は，元のトレースを Declare のテンプレートで検証したときと近い分類結果を示した．特に，一致率が 100% となる論理式はテンプレ―トと同じ分類結果を示すが，性質の異なる論理式であった．したがって，少ないトレース数によるサンプルを用いた論理式の生成でも元のトレースを特徴づけるような論理式を生成する可能性が高いといえる．

6. おわりに

本論文では，既存研究である SAT，DT ベースで論理式を生成する手法 [7] を利用し，一般的なイベントログの例から生成できる論理式の表現能力の調査を行った．3 つの実験より，一般的なイベントログに対して SAT，DT ベースの手法を利用した場合，トレースの空虚性を考慮すべきであるとわかった．さらに，全体のトレース数よりも少ないトレース数から成るサンプルを使用しても，元のトレースを特徴づけるような論理式を生成できる可能性が高いとわかった．本手法は LTL ベースの数学的記法に精通していない者が目的の性質を示す論理式を記述するための手法の 1 つであり，本研究により適した形式にツールの改良を行えば，宣言的手法に基づいたビジネスプロセスの分析に新たな洞察を与えることができると予測できる．

今後は，少ないトレース数で論理式を生成する際に利用するサンプルの選択手法の確立や本手法におけるサンプルの構成要素が生成される論理式に与える影響を評価すべきである．

文　献

[1] H.T. de Beer, P.C.W. van den Brand. *The LTL checker plugins: A reference manual*, 2004.

[2] Christian W. Gunther and Eric Verbeek. Xes - standard definition. Technical report, BPMcenter. org, 2014.

[3] Orna Kupferman, Moshe Y.Vardi. Vacuity detection in temporal model checking. In *International Journal on Software Tools for Technology Transfer*, pp. 224–233. Springer, 2003.

[4] F.M. Maggi. Declarative process mining with the declare component of prom. Technical report, BPM(Demos), 2013.

[5] F.M. Maggi. Declarative process mining. In *Encyclopedia of Big Data Technologies*. Springer, 2018.

[6] F.M. Maggi, Arjan J. Mooij, Wil M.P. van der Aalst. User-guided discovery of declarative process models. In *2011 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 192–199. IEEE, 2011.

[7] Daniel Neider, Ivan Gavran. Learning linear temporal properties. In *Formal Methods in Computer-Aided Design 2018*, pp. 1–10. IEEE, 2018.

[8] W.M.P. van der AalstH, B.F.van Dongen T. de Beer. Process mining and verification of properties: An approach based on temporal logic. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pp. 130–147. Springer, 2005.

[9] 堀田大貴, 平山秀昭, 早瀬健夫, 田原康之, 大須賀昭彦. 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. 電子情報通信学会論文誌 D J101-D(3), pp. 530–538. 電子情報通信学会, 2018.

[10] 飯島正, 田端啓一, 斎藤忍. プロセスマイニング・サーベイ (第 01 回: 概要と基本概念). 情報システム学会誌, Vol. 11, No. 2, pp. 20–53, 2016.

[11] 飯島正, 田端啓一, 斎藤忍. プロセスマイニング・サーベイ (第 02 回: ツール). 情報システム学会誌, Vol. 12, No. 1, pp. 1–25, 2016.

[12] 飯島正, 田端啓一, 斎藤忍. プロセスマイニング・サーベイ (第 03 回: データ). 情報システム学会誌, Vol. 12, No. 1, pp. 26–41, 2016.