時間オートマトンプロセスモデルにおけるコンフォーマンスチェッキング

伊藤 宗平　濱江 堅登

コンフォーマンスチェッキングとは，プロセスのモデル化された振る舞いと記録された振る舞いとの違いを定量的に評価するプロセス分析手法の一つである．従来のコンフォーマンスチェッキングでは，実行されたタスクの順序のみを用いて適合度を評価していたため，モデルで要求されたタスク完了時間に違反しているような実行に対する適合度を正しく評価することは不可能であった．本研究では，時間に関する制約を持つプロセスモデルを時間オートマトンとして与え，その振る舞いをアクティビティ名と時間の対の列として与えた場合に，振る舞いがモデルとどれだけ適合しているかを評価する手法を与える．また，適合度の計算アルゴリズムとして素朴な全探索によるものと A*アルゴリズムを用いた手法の二通りを実装し，評価する．

1　はじめに

プロセスは特定の目的を達成するために組織化されたアクティビティの集まりである．プロセスは至る所に現れ，特に業務プロセス（ビジネスプロセス）はその最たる例である．通常，組織はあまたの業務プロセスから構成されているため，その効率化は長年取り組まれてきた社会的な課題の 1 つである．

業務プロセスの設計，評価，改善を組織的に行うためにプロセスマイニング [12] と呼ばれる手法が注目を集めている．プロセスマイニングは，プロセスモデルやプロセスの実行記録（これを**イベントログ**という）を用い，様々な分析を行う手法の総称であり，プロセスに関するビッグデータの活用法として近年注目を集めている．

プロセスマイニングの手法の一つにコンフォーマンスチェッキングがある．これは，プロセスモデルが，実際の振る舞い（イベントログ）とどれだけ適合しているかを定量的に評価する手法である．現実世界においては，実際のプロセスの全ての情報をモデルに含めるることは一般に不可能であり，プロセスモデルは様々な抽象化を経て得られたものである．更に，モデル化の過程で誤りが混入することは珍しくない．またプロセス自身も環境の変化や技術の更新により変化していくものである．これらの理由から，プロセスモデルと実際のプロセスとの差異を定量的に評価する手法であるコンフォーマンスチェッキングは有用である．

このような重要性からコンフォーマンスチェッキングを計算機により自動的に行う手法がこれまでに提案されてきた [15] [17] [16] [9] [10] [14] [13] [1] [2] [3] [11]. しかしながら，これらの手法はいずれもアクティビティ名のみの情報からコンフォーマンスチェッキングを行うものであり，業務の手順がモデルと適合しているかどうかにのみ関心があった．すなわち，実際の振る舞いはアクティビティ名の列であり，それがアクティビティ名の依存関係を表したプロセスモデルにおいて再現できるか，再現できない場合はどれだけモデルと乖離しているかを，0 から 1 の間の適合度で表すものであった．しかしながら，一般にはイベントログは多くの情報を含んでおり，例えば各アクティビティのタイムスタンプやそのアクティビティ種別に応じた独自の属性（例えば「見積もりを送る」というア

---

*Conformance Checking on Timed Automaton Process Models

Sohei Ito, 長崎大学情報データ科学部, Dept. of Information and Data Sciences, Nagasaki University.
Kento Hamae, FCC テクノ, FCC techno.

クティビティならば「価格」という属性）を持っている．プロセスモデルがそれらの属性に依存して振る舞いを定めるモデルである場合は，アクティビティの順序だけでなくアクティビティが持つ属性値も考慮したコンフォーマンスチェッキングが可能であることが望ましい．例えばアクティビティの完了にかかった時間や消費された資源が定められた範囲の通りであるかどうかを確かめることは，実用上非常に重要である．

本研究では，時間属性を考慮したプロセスモデルとイベントログの間でコンフォーマンスチェッキングを行う手法を提案する．時間属性を選択した理由は，ほぼ全てのイベントログが時間属性を含むためである．プロセスモデルとしては，時間を自然に表現できる形式論である時間オートマトンを考える．ここではロケーションをイベントログのアクティビティと対応させる．イベントログはアクティビティと時間（そのアクティビティの完了時間，すなわちタイムスタンプとみなす）の対の列である．時間オートマトンの振る舞いはロケーションと時間の対の列（これを時間語という）であるため，これにより自然にイベントログが時間語の集合と対応することになる．

本手法ではアクティビティの順序だけを考慮した適合度 (時間適合度) を求め，そのうえで時間の適合度を評価する．順序適合度を求める手法としては，ペトリネットによるプロセスモデルに対するアライメントに基づくコンフォーマンスチェッキング手法 [3] を有限オートマトンモデルに修正し，それを用いる．これは，モデルとトレース (一つのプロセス実行列) とのマッチングを取るものである．時間適合度は，本研究が提案する評価法を与え，そのマッチングにおいて時間属性の条件がどれだけ満たされているかを評価する．

アライメントに基づく手法では，アクティビティ列をプロセスモデルに当てはめた場合，最も逸脱の少ない（最適な）モデルの実行列を探索することで，その列の適合度を求めている．一般にそのような最適なモデルの実行列は複数存在するが，順序だけを考えた場合いずれも適合度は等しい．しかしながら，時間属性の適合度も考慮した場合には，それら「最適なモデルの実行列」の間で適合度に差異が発生する．したがって，そのような適合度の中で最良の値を求めるには最適なモデルの実行列をすべて発見することが必要となる．しかしながら，モデルとイベントログが大きくなった場合，そのような全探索は実用的ではないことが予想されるため，本研究では，先行研究でも用いられていた A*アルゴリズムによる「最適なモデルの実行列」の探索アルゴリズムも実装し，全探索アルゴリズムとのパフォーマンスの違い，および求められた最良な適合度の比較を行う．

本稿の構成は以下の通りである．2 節では本手法を正確に述べるために必要な数学的準備を行う．3 節では時間オートマトンプロセスモデルに対するコンフォーマンスチェッキングにおける適合度の厳密な定義を与える．4 節では 3 節で与えた適合度の計算アルゴリズムについて述べる．5 節では本稿で提案するコンフォーマンスチェッキング手法の実装と実験結果について述べる．6 節では関連研究について述べる．7 節ではまとめと今後の課題について述べる．

2 準備

本節ではイベントログおよび時間オートマトンによるプロセスモデルの数学的定義を与える．

イベントログは，業務などを行った際に記録される業務に関する記録，業務履歴である．イベントログは複数の**ケース**から構成される．ケースは**イベント**の列である．イベントは**アクティビティ**と複数の**属性**から成る．イベントは必ず一つのケースにのみ出現する．すなわち同一のイベントが複数のケースに現れることはない．

イベントログは一般に表形式で表される．表 2 に仮想的な航空会社における航空券払戻プロセスのイベントログを示す．このイベントログには，ケース ID，イベント ID，タイムスタンプ，アクティビティ，担当者 ID などといった要素が記録されている．本研究ではイベントが持つ情報のうちアクティビティと時間属性のみを考慮するためその他の属性は用いないことに注意されたい．

数学的には，イベントログは以下のように定義される．

表 1　イベントログの例

ケース ID	イベント ID	タイムスタンプ	アクティビティ	担当者 ID	...
1	0001	2023-12-01 15:00:00	要求登録	BB33	
	0002	2023-12-02 13:15:00	詳細調査	AA01	
	0003	2023-12-02 18:30:05	航空券検査	AA02	
	0004	2023-12-03 16:14:32	決定	AA02	
	0005	2023-12-04 19:00:00	要求却下	AA03	
2	0006	2023-12-01 10:00:09	要求登録	BB19	
	0007	2023-12-01 14:45:00	航空券検査	AA01	
	0008	2023-12-01 17:30:30	簡易調査	AA01	
	0009	2023-12-01 18:00:37	決定	AA02	
	0010	2023-12-02 10:30:05	補償金支払い	AA03	
...					

定義 1 (イベントログ) アクティビティの集合を A とする．このとき，$L_A = (E, C, \alpha, \beta, \tau, \succ)$ はアクティビティの集合 A に関する**イベントログ**である．ここで E は**イベント**の有限集合，C は**ケース**の有限集合，$\alpha : E \to A$ はイベントに**アクティビティ**を割り当てる関数，$\beta : E \to C$ はイベントをケースに関連付ける関数，$\tau : E \to \mathbb{R}_{\geq 0}$ はイベントの時間属性の値を返す関数，$\succ \subseteq E \times E$ は E 上の狭義全順序 (全順序から反射律を除いたもの) である．

プロセスモデルとは業務手順などを仮想的にモデル化したものである．モデル化の形式論としては，ペトリネットモデルや有限オートマトンモデルなどが通常用いられる．本研究では，時間オートマトン (時間の条件を持つ有限オートマトン) で表されるプロセスモデルを扱う．

定義 2 (時間オートマトンによるプロセスモデル) 時間オートマトンによるプロセスモデルは 7 項組 $(L, l_0, z, A, C, T, \gamma)$ である．ここで，

- L はロケーションの集合
- $l_0 \in L$ は初期ロケーション
- $z \in L$ は最終ロケーション
- A はアクティビティの有限集合
- C はクロック変数 $C = \{t_1, t_2, \dots\}$ の集合
- $T \subseteq L \times B(C) \times 2^C \times L$ は遷移の集合
- $B(C)$ はクロック制約の集合で $t_i \bowtie d_i$ の形の論理積であり，$t_i \in C, \bowtie \in \{\leq, <, =, \geq, >\}, d_i \in \mathbb{R}_{\geq 0}$
- $\gamma : L \to A$ はロケーションに対応するアクティビティを割り当てる単射

である．遷移 $(l_1, g, \{t_1, t_2\}, l_2)$ は，ロケーション l_1 から l_2 への遷移を表しており，その遷移が可能になる条件 (ガード) が論理式 g であり，この遷移が起きた場合クロック変数 t_1, t_2 の値がリセットされることを表している．

注釈. 本研究における時間オートマトンでは，遷移の際のクロックのリセットは重要な意味を持たない．これは，時間オートマトンプロセスモデルを実行することを考えているわけではなく，イベントログに記録された時間属性の値がモデルのガードを満たしているかどうかにのみ関心があるためである．インバリアントについては単純化のため考慮していないが，ガードと同様に扱うことが可能である．また，遷移ではなくロケーションがアクティビティに対応している理由は，そのアクティビティの完了にかかる時間をそのロケーションにおけるディレイ遷移として自然に表せるためである．

図 1 に時間オートマトンによるプロセスモデルの例を示す．

時間オートマトンの意味論は，ロケーションとクロックの付値の変化列として定められる．

$L = \{s_1, s_2, s_3, s_4\}$, $l_0 = s_1$, $z = s_4$, $A = \{a, b, c, d\}$, $C = \{t\}$, $T = \{(s_1, 5 < t < 10, \{t\}, s_2), (s_2, 10 < t < 20, \{t\}, s_3), (s_3, 15 < t < 25, \{t\}, s_2), (s_3, 10 < t < 15, \{t\}, s_4)\}$, $\gamma(s_1) = a$, $\gamma(s_2) = b$, $\gamma(s_3) = c$, $\gamma(s_4) = d$.

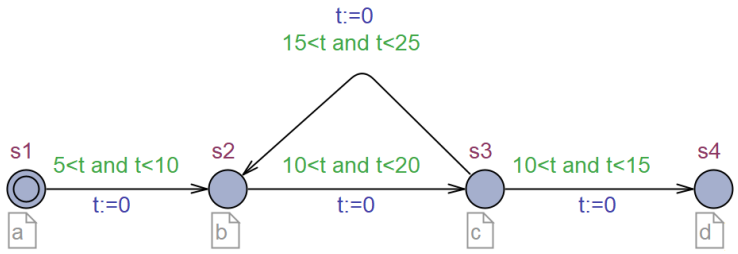

図 1　時間オートマトンによるプロセスモデルの例

定義 3 (時間オートマトンの意味論) 時間オートマトン $(L, l_0, z, A, C, T, \gamma)$ の意味論は以下のラベル付き遷移系 $(\mathcal{S}, s_0, \to)$ により与えられる．

- $\mathcal{S} \subseteq L \times (C \to \mathbb{R}_{\geq 0})$ は状態の集合
- $s_0 = (l_0, s_0)$ は初期状態
- $\to \subseteq \mathcal{S} \times \mathbb{R}_{\geq 0} \cup \{\tau\} \times \mathcal{S}$ は以下を満たす遷移である (τ はアクション遷移を表すタグである)．
  1. $(l, u) \xrightarrow{d} (l, u + d)$, ただし $d \in \mathbb{R}_{\geq 0}$
  2. $(l, u) \xrightarrow{\tau} (l', u)$, ただし $e = (l, g, r, l') \in E$ が存在し，$u \models g, u' = u[r \mapsto 0]$ である．ここで，$u \models g$ で，クロック付値 u がガード g を満たすことを表し，$u[r \mapsto 0]$ は r 中のクロック変数に対し 0 を返し，その他のクロック変数に対しては u と同じ値を返す関数である．

状態 (s, u) の列を**時間語**と呼ぶ．遷移系 $(\mathcal{S}, s_0, \to)$ はこの遷移系が含む時間語の集合を表しているとみなすことができる．時間オートマトンの**実行系列**は，対応する遷移系に含まれる時間語である．

図 1 の時間オートマトンはクロック変数は t のみであるため，遷移系 $(\mathcal{S}, s_0, \to)$ において「ロケーション l に滞在し，クロック t の値が x である」という状態を単純に (l, x) と書くことにする．初期状態を $(s_1, 0)$ とすると，時間語 $(s_1, 0) \xrightarrow{6.2} (s_1, 6.2) \xrightarrow{\tau} (s_2, 0) \xrightarrow{13.8} (s_2, 13.8) \xrightarrow{\tau} (s_3, 0) \xrightarrow{11.1} (s_3, 11.1) \xrightarrow{\tau} (s_4, 0)$ は，このオートマトンの一つの実行系列である．

3　時間オートマトンプロセスモデルのコンフォーマンスチェッキング

コンフォーマンスチェッキングとは，イベントログとプロセスモデルとの差異を評価する手法である．その差異は，0 から 1 の値で表される適合度によって評価される．イベントログとプロセスモデルの内容が全く一致していなければ適合度は 0 であり，完全に一致しているなら 1 と評価する．また適合度が 0 より大きく 1 より小さい場合は，次の 2 つの解釈が考えられる．

1. プロセスモデル側に問題があるという解釈．実際の業務内容を正しくモデル化できていない，つまりプロセスモデルに改善の余地があるという解釈である．
2. イベントログ側に問題があるという解釈．イベントログとして記録された内容に何かしらの違反が含まれる，つまりプロセスモデルの想定通りに業務が行われていない．この場合は非効率化している業務内容の発見などにつながる可能性がある．

本研究では，時間オートマトンプロセスモデルと，時間属性を持つイベント列に対し，それらの間の適合度を 0 から 1 の間の実数で評価する手法を与える．この手順は以下の通りである．

1. 時間オートマトンのうち，時間に関する制約を除いた有限オートマトンと，時間属性を持つイベント列のうちアクティビティだけを取り出した列（ケース）の間で最適なアライメント（モデルとケースのマッチング）を求め，その適合度（**順序適合度**と呼ぶ）を求める．
2. 1 で得られた最適アライメントに対し，時間オー

トマトン上で対応する遷移を辿ったときに時間属性に関する制約がどれだけ満たされているか（**時間適合度**）を求める．

3. 1 で得られた順序適合度と 2 で得られた時間適合度の平均を，そのイベント列の適合度とする．

1 の有限オートマトンとケースの間の順序適合度は，ペトリネットモデルにおけるアライメントによる適合度の定義 [3] を有限オートマトンモデルに適応させることで定義する．2 の時間適合度については，クロックの制約を区間制約としたときに，その区間からどれだけ外れているかを定量的に評価する式を与えることで定義する．

3.1 順序適合度

本節では有限オートマトンモデルにおけるケース（イベント列）のアライメントに基づく適合度（順序適合度）について述べる．なお，ここではイベントの情報はアクティビティのみを考えるため，ケースはアクティビティの列と同一視してよい．

順序適合度を定めるための準備として，有限オートマトンモデルのインスタンスという概念を導入する．これは，有限オートマトンモデルから分岐やループの繰り返しを解いて得られる直線的な有限オートマトンのことである．よって，一つのインスタンスはモデルの一つの実行例を表している．

定義 4 アクティビティ集合 A 上の有限オートマトンモデルを $M = (L, l_0, z, A, T, \gamma)$ とする（これは，定義 2 からクロックを除き，遷移関係からガードとリセットを取り除いたものである）．有限オートマトンモデル M の**インスタンス** $I = (LI, TI, \rho)$ は以下のように定義される．

- LI はロケーションインスタンスの集合
- $TI \subseteq LI \times LI$ は遷移インスタンスの集合
- $\rho : LI \to L$ は状態インスタンスを M の状態 L に割り当てる関数

ここで，I は以下を満たす．

1. すべての $(x, y) \in TI$ について，$(\rho(x), \rho(y)) \in T$
2. $\forall x \in LI.|\mathsf{succ}(x)| \leq 1$
3. $\forall x \in LI.|\mathsf{pred}(x)| = 0$ ならば $\rho(x) = l_0$

ここで $\mathsf{succ}(x) = \{x' \mid (x, x') \in T\}$，$\mathsf{pred}(x) = \{x' \mid (x', x) \in T\}$ である．

図 2 の有限オートマトンモデルに対し，そのインスタンスの一つを図 3 に示す．このインスタンスは，モデルにおいてアクティビティを a, c, d, e, d, e, f と実行したときの実行列に対応している．

有限オートマトンのインスタンスが，ケースのプレフィックスにマッチしているとき，それをマッチングインスタンスと呼び以下のように定義される．

定義 5 (マッチングインスタンス) アクティビティ集合を A とするイベントログ $L_A = (E, C, \alpha, \beta, \tau, \succ)$ のうち，ケース ID が ℓ のケースを E_ℓ とする．E' を E_ℓ のプレフィックスとする．$M = (L, l_0, z, A, T, \gamma)$ を有限オートマトンプロセスモデルとする．$I = (LI, TI, \rho)$ を M のインスタンスとする．$\mu : E' \rightharpoonup LI$ を，イベントを状態インスタンスに割り当てる部分関数とし，定義域 $\mathsf{dom}(\mu) \subseteq E'$ から値域 $\mathsf{rng}(\mu) \subseteq LI$ への双射を誘導するとする．

このとき，以下の 2 つの条件を満たすとき，I は μ によって E' とマッチしているといい，I を μ による E' の**マッチングインスタンス**と呼ぶ．

1. すべての $e_1, e_2 \in \mathsf{dom}(\mu)$ に対し，I 中で $\mu(e_1)$ から $\mu(e_2)$ にパスがあるならば $e_1 \succ e_2$
2. すべての $e \in \mathsf{dom}(\mu)$ に対し，$\rho(\mu(e)) \in \{s \in L \mid \gamma(s) = \alpha(e)\}$

条件 1 はプレフィックス中のイベントの順序がインスタンス中でも保存されていることを表し，条件 2 はプレフィックスの中で μ によってマッピングされた各イベントが，このイベントが表すアクティビティに対応するロケーションにマッピングされていることを表す．

μ により E' とマッチする任意のマッチングインスタンスを対 $(I_{E'}, \mu)$ で表し（$I_{E'}$ はオートマトンインスタンス），このような対全体の集合を $\mathcal{J}_{E'}$ で表す．

図 3 に示したオートマトンインスタンスを I_1 とし，プレフィックス $E' = \langle e_1, e_4, e_5, e'_4, e'_5, e_6 \rangle$ を考える．ここで $\alpha(e_1) = a$, $\alpha(e_4) = \alpha(e'_4) = d$, $\alpha(e_5) = \alpha(e'_5) = e$, $\alpha(e_6) = f$ とする．すなわち，こ

$L = \{s_1, s_2, s_3, s_4, s_5, s_6\}$, $l_0 = s_1$, $z = s_6$, $A = \{a, b, c, d, e, f\}$,
$T = \{(s_1, s_2), (s_1, s_3), (s_2, s_4), (s_3, s_4), (s_4, s_5), (s_5, s_4), (s_5, s_6)\}$,
$\gamma(s_1) = a$, $\gamma(s_2) = b$, $\gamma(s_3) = c$, $\gamma(s_4) = d$,$\gamma(s_5) = e$,$\gamma(s_6) = f$.

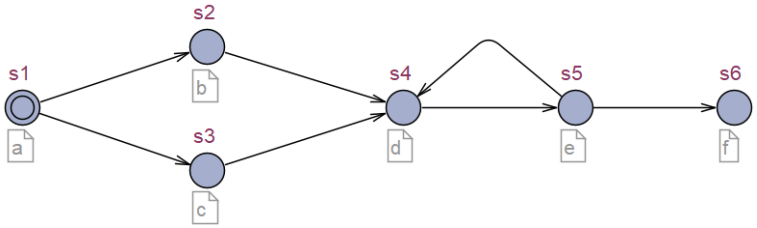

図 2　有限オートマトンによるプロセスモデルの例

$LI = \{s'_1, s'_3, s'_4, s''_4, s'_5, s''_5, s'_6\}$, $TI = \{(s'_1, s'_3), (s'_3, s'_4), (s'_4, s'_5), (s'_5, s''_4), (s''_4, s''_5), (s''_5, s'_6)\}$,
$\rho(s'_1) = s_1$, $\rho(s'_3) = s_3$, $\rho(s'_4) = \rho(s''_4) = s_4$, $\rho(s'_5) = \rho(s''_5) = s_5$, $\rho(s'_6) = s_6$

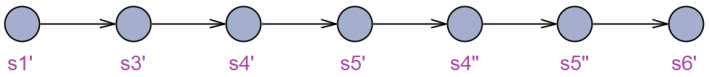

図 3　インスタンスの例

のケースは図 2 のモデルにおいて，アクティビティを a, d, e, d, e, f の順で実行している．いま，写像 μ_1 を，$\mu_1(e_1) = s'_1$, $\mu_1(e_4) = s'_4$, $\mu_1(e_5) = s'_5$, $\mu_1(e_4) = s''_4$, $\mu_1(e_5) = s''_5$, $\mu_1(e_6) = s'_6$ とすると，(I_1, μ_1) はプレフィックス E' のマッチングインスタンスである（図 4）．

図 4 のマッチングインスタンスには，s'_3 というロケーションインスタンスが存在する．これは $\rho(s'_3) = s_3$ で，$\gamma(s_3) = c$ であるため，プロセスモデルにおいて 2 番目に実行されるアクティビティが c であることに対応する．しかし，E' には対応するアクティビティを持つイベントは存在しない．すなわち，図 2 のプロセスモデルでは a の次に c というアクティビティを実行することを想定していたが，現実に実行されたケース E' では c というアクティビティがスキップされてしまっていたことがわかる．

このように，現実に実行されたケースと，本来のプロセスモデルにおいて実行されるべき列（オートマトンインスタンス）との間の対応を，マッチングインスタンスにより表すことができる．このようなケースとモデルの差異を，順序適合度として定量的に評価する．

定義 6（マッチングインスタンスの順序適合度） アクティビティ A 上の有限オートマトンプロセスモデルを $M = (L, l_0, z, A, T, \gamma)$，$A$ 上のイベントログを $L_A = (E, C, \alpha, \beta, \tau, \succ)$，$c \in C$ をケース，E_ℓ をケース ℓ のイベント列とする．M のオートマトンインスタンスを $I = (LI, TI, \rho)$ とし，(I, μ) を I と E_ℓ のマッチングインスタンスとする．$LI_s = LI - \mathsf{rng}(\mu)$ とし，$E_i = E_\ell - \mathsf{dom}(\mu)$ とする．また，$\kappa^s : A \to \mathbb{N}_{>0}$, $\kappa^i : A \to \mathbb{N}_{>0}$ をそれぞれアクティビティのスキップコスト，挿入コストを返す関数とする．このとき，マッチングインスタンス (I, μ) の**順序適合度** f_{order} は以下の式で定義される．

$$f_{order} = 1 - \frac{\mathsf{cost}}{\mathsf{base}}$$

ここで，

$$\mathsf{cost} = \sum_{s' \in LI_s} \kappa^s(\gamma(\rho(s'))) + \sum_{e \in E_i} \kappa^i(\alpha(e)),$$

$$\mathsf{base} = \min_{\sigma \in \mathsf{run}(M)} \sum_{s \in \sigma} \kappa^s(\gamma(s)) + \sum_{e \in E_\ell} \kappa^i(\alpha(e))$$

であり，$\mathsf{run}(M) \subseteq L^*$ は M の実行列の集合である．列 $\sigma = \langle s_0, s_1, ..., s_n \rangle$ が M の実行列であるとは，$s_0 = l_0$ かつ $s_n = z$ かつすべての $i \in \{0, ..., n-1\}$

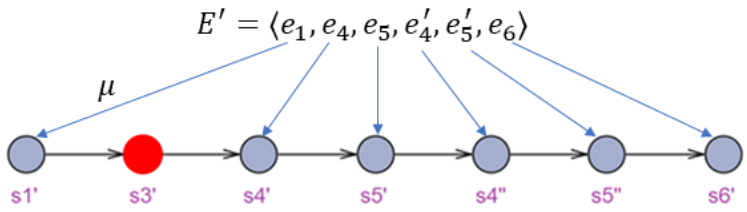

図 4　マッチングインスタンスの例

に対し, $(s_i, s_{i+1}) \in E$ であることをいう.

この定義において，LI_s はモデルでは実行されるべきだったのにケースの実行 E_ℓ においてスキップされたイベントの集合を表し，E_i はモデルでは実行されるべきではなかったのにケースの実行 E_ℓ において実行されてしまった（挿入された）イベントの集合を表す．このように，モデルの進行とケースの進行が対応しない動作をミスアライメントという．それぞれのイベントにはスキップされた場合，挿入された場合のペナルティを評価するためのコストが κ^s, κ^i により定められている．この式の分母の値は，モデルの実行列のうち最小のコストと E_ℓ 中の全てのイベントが挿入されたとみなしたときのコストの和である．これはいわば可能なアライメントのうち，最悪の場合のコストである．これを基準値とし，その基準値に対しどれだけのミスアライメントが発生したか (分子の値) を比率として求め，1 からその値を引いたものが順序適合度である．スキップあるいは挿入されたイベントがなければ適合度は 1 となり，ケースとモデルがまったく一致していない場合には 0 となる．したがって，適合度は必ず 0 から 1 の間の値を取る [2] [4].

3.2　時間適合度

この節では，時間属性を持つイベントの列であるケースと時間オートマトンプロセスモデルとの間の時間適合度の定義を与える．本定義では，ケースのアクティビティ列と時間オートマトンプロセスモデルにおける有限オートマトン部との間のマッチングインスタンスが与えられたとき，そのインスタンスがどれだけ時間の制約を満たしているかを定量的に評価する．

ここでは議論を単純化するため，時間オートマトンプロセスモデルが持つクロックはただ一つ (t とする) とし，ロケーション遷移間のガードは $l < t < u$ の形のみとする．$g = \mathsf{true}$ の場合には $l = 0, u = \infty$ であるとみなす．さらに，$l = \underline{g}$, $u = \overline{g}$ と書く．

定義 7 (マッチングインスタンスの時間適合度) アクティビティ A 上の時間オートマトンプロセスモデルを $M = (L, l_0, z, A, C, T, \gamma)$, $M_1 = (L, l_0, z, A, T, \gamma)$ を M から時間の情報を除いて得られた有限オートマトンプロセスモデルとする．$L_A = (E, C, \alpha, \beta, \tau, \succ)$ をイベントログ，ケース ID が ℓ のケースを E_ℓ とする．$I = (LI, TI, \rho)$ をオートマトンインスタンス，(I, μ) を E_ℓ と M_1 のマッチングインスタンスとする．$\mathsf{last}(E_\ell)$ を E_ℓ の最後のイベントを取り出す関数とし，$D = \mathsf{dom}(\mu) - \{\mathsf{last}(E_\ell)\}$ とする．遷移 $(p, g, r, q) \in T$ に対し，$g = G(p, q)$ と表すこととする．また，イベント e に対し，$\rho(\mu(e))$ を $\mathsf{loc}(e)$ と表す（マッチングインスタンスにおいてイベント e の実行がモデルのロケーション $\mathsf{loc}(e)$ のアクティビティの実行と解釈されていることを表す）．$s' \in LI$ に対し，$\mathsf{next}(s')$ で $(s', t') \in TI$ なるただ一つの t' を表す．このとき，(I, μ) の M に対する時間適合度は以下の式で定義される．

$$f_{time} = \frac{1}{|D|} \sum_{e \in D} \frac{f(e)}{F(e)}$$

ここで

$$f(e) = \overline{G(\mathsf{loc}(e), \mathsf{loc}(\mathsf{next}(e)))} - \underline{G(\mathsf{loc}(e), \mathsf{loc}(\mathsf{next}(e)))}$$

$$F(e) = \max(\tau(e), \overline{G(\mathsf{loc}(e), \mathsf{loc}(\mathsf{next}(e)))}) - \min(\tau(e), \underline{G(\mathsf{loc}(e), \mathsf{loc}(\mathsf{next}(e)))})$$

この定義の直感的な意味を述べる．ケースのイベント e の実行があるロケーション p の実行に対応し，その遷移のガードが $l < t < u$ であるとする（p から

の遷移が複数あり得るが，マッチングインスタンスにおいて次に遷移するイベント $\mathsf{next}(e)$ より特定可能なことに注意されたい）．このとき，e に記録された時刻が $\tau(e)$ である．$l < \tau(e) < u$ であれば，この遷移のガードを満たしている．このとき $f(e) = u - l$, $F(e) = \max(\tau(e), u) - \min(\tau(e), l) = u - l$ となり，$\frac{f(e)}{F(e)} = 1$，すなわちこの実行は完全に適合している．一方，もし $\tau(e) \leq l$ であれば， $F(e) = u - \tau(e)$ となり，$\frac{f(e)}{F(e)} = \frac{u-l}{u-\tau(e)}$ となる．$\tau(e)$ が l よりも小さければ小さいほどこの値は小さくなり，適合度が小さくなる．例えば $l = 10$, $u = 20$ のとき，$\tau(e) = 5$ なら $\frac{20-10}{20-5} = \frac{10}{15} = \frac{2}{3}$ だが，$\tau(e) = 0$ なら $\frac{20-10}{20-0} = \frac{10}{20} = \frac{1}{2}$ となる．逆に，もし $u \leq \tau(e)$ であれば，$F(e) = \tau(e) - l$ となり，$\frac{f(e)}{F(e)} = \frac{u-l}{\tau(e)-l}$ となる．$\tau(e)$ が u よりも大きければ大きいほどこの値は小さくなり，適合度が小さくなる．

このように，イベント実行ごとの時間制約の適合度の平均値を求めたものがマッチングインスタンスの時間適合度 f_{time} である．

以下に時間適合度の計算例を示す．まず，時間オートマトンプロセスモデルは図 1 に示したものとし，ケース $E = \langle (a, 10), (b, 15), (c, 25), (b, 35), (d, 0) \rangle$ とする．またマッチングインスタンス $((LI, TI, \rho), \mu)$ を以下の通りとする．$LI = \{a', b', c', b'', c'', d'\}$, $TI = \{(a', b'), (b', c'), (c', b''), (b'', c''), (c'', d')\}$, $\rho(a') = s_1$, $\rho(b') = \rho(b'') = s_2$, $\rho(c') = \rho(c'') = s_3$, $\rho(d') = s_4$. $\mu((a, 10)) = a', \mu((b, 15)) = b'$, $\mu((c, 25)) = c'$, $\mu((b, 35)) = b''$, $\mu((d, 0)) = d'$. これより，LI 中の c'' に対応するケース中のアクティビティは存在しないことが分かる．すなわち，ケースにおいてアクティビティ c が一回スキップされている．このとき，$D = \mathsf{dom}(\mu) - \{\mathsf{last}(E)\} = \{(a, 10), (b, 15), (c, 25), (b, 35)\}$ となる．この関係を図 5 に示す．

この例に対し，定義 7 を用いて時間適合度を計算すると以下の通りである．

$$\begin{aligned}&\frac{1}{4}\left(\frac{\max(5,10)-\min(5,10)}{\max(10,10)-\min(10,5)}\right.\\&+\frac{\max(10,20)-\min(10,20)}{\max(15,20)-\min(15,10)}\\&+\frac{\max(15,25)-\min(15,25)}{\max(25,25)-\min(25,15)}\\&\left.+\frac{\max(10,20)-\min(10,20)}{\max(35,20)-\min(35,10)}\right)\\=&\frac{1}{4}(1+1+1+0.4)=0.85\end{aligned}$$

注釈. この例からわかるように，ミスアライメントとなるアクティビティについては，時間適合度の計算においてその属性値が使用されることはない．これは，そのようなイベントについては順序適合度ですでに適切なペナルティが課されているためという意味論的な理由と，対応する遷移が存在しないため妥当な計算方法が与えられないという技術的な理由による．ミスアライメントとなったアクティビティの扱い方は，Leoni らによるモデルが定める属性値とイベントの持つ属性値の不一致まで考慮したコンフォーマンスチェッキング手法 [6] と同様である．また，タイムスタンプがガードの境界値を取る場合，そのイベントに関する時間適合度は計算上 1 となる．実際，先の例でイベント $(a, 10)$ においては，その部分の適合度は $\frac{\max(5,10)-\min(5,10)}{\max(10,10)-\min(10,5)} = 1$ となる．この場合，論理的にはガードを満たしていないため偽である．しかし，現実的には属性値がちょうど境界値を取る確率はほぼ無視してよく，また直感的には $t = 10$ の場合と $t = 10 + 10^{-100}$ の場合ではどちらも違反であってもほぼ同等の適合度であると考えられるため，本研究では境界値においては適合度は 1 を取るものと定めた．

3.3 最終的な適合度

マッチングインスタンスの最終的な適合度は，そのマッチングインスタンスに対する順序適合度 f_{order} と時間適合度 f_{time} の加重平均である．

定義 8 (マッチングインスタンスの適合度) アクティビティ A 上の時間オートマトンプロセスモデルを $M = (L, l_0, z, A, C, T, \gamma)$, $M_1 = (L, l_0, z, A, T, \gamma)$ を M から時間の情報を除いて得られた有限オートマトンプロセスモデルとする．$L_A = (E, C, \alpha, \beta, \tau, \succ)$ をイベントログ，ケース ID が ℓ のケース E_ℓ とする．

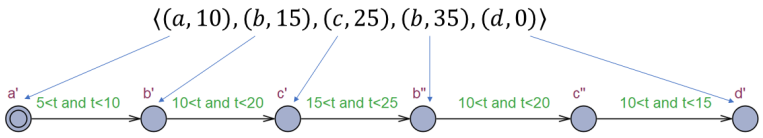

図 5　図 1 のマッチングインスタンスの例．マッチングインスタンスにガードは存在しないが，理解の補助のため表示．

(I, μ) を E_ℓ と M_1 のマッチングインスタンスとする．定義 6 によるマッチングインスタンス (I, μ) の順序適合度を f_{order}，定義 7 によるマッチングインスタンス (I, μ) の時間適合度を f_{time} とする．このとき，マッチングインスタンス (I, μ) の**適合度**は以下の式で定義される．

$$fitness = \frac{f_{order} + f_{time}}{2}$$

4　適合度の計算

前節ではマッチングインスタンスに対する適合度を与えた．しかし，ケースに対するマッチングインスタンスは一般に複数存在する．例えばプロセスモデルにループがある場合，可能なオートマトンインスタンスはループ内のアクティビティを繰り返した回数毎に存在する．例えば図 1 において，アクティビティを a, b, c, d と実行した場合と a, b, c, b, c, d と実行した場合，a, b, c, b, c, b, c, d と実行した場合，のようにいくつもオートマトンインスタンスが考えられる．したがって一つのケースに対しても複数のマッチングインスタンスが存在する．その中で，適合度の計算に用いるものは**最適な**マッチングインスタンスである必要がある．例えばケースが a, b, d となっていた場合，これに最も近い図 1 のモデルの実行列は a, b, c, d であるため，a, b, c, b, c, d のような実行列を用いて適合度を評価するのは不適切である．

この節ではケースとプロセスモデルとの間の適合度の計算法について述べる．特に，ケースとプロセスモデルの順序適合度は**最適なマッチングインスタンス**を用いるため，この定義を与える．また，最適なマッチングインスタンスを効率的に求めるために A* アルゴリズムを利用する方法について示す．

4.1　最適マッチングインスタンス

定義 6 において，マッチングインスタンスの適合度を定義した．ここではマッチングインスタンスのコストを定義し，それに基づき最適マッチングインスタンスを定義する．

定義 9（マッチングインスタンスのコスト，最適マッチングインスタンス） アクティビティ A 上の有限オートマトンプロセスモデルを $M = (L, l_0, z, A, T, \gamma)$，$A$ 上のイベントログを $L_A = (E, C, \alpha, \beta, \tau, \succ)$，$\ell \in C$ をケース，E_ℓ をケース ℓ のイベント列とする．E' を E_ℓ のプレフィックスとする．また，$\kappa^s : A \to \mathbb{N}_{>0}$，$\kappa^i : A \to \mathbb{N}_{>0}$ をそれぞれアクティビティのスキップコスト，挿入コストを返す関数とする．マッチングインスタンスの**コスト関数** $\delta_n : \mathcal{J}_{E'} \to \mathbb{N}$ は以下の式で与えられる．

$$\delta_n((I_{E'}, \mu)) = \sum_{s_i \in LI_s} \kappa^s(\gamma(\rho(s_i))) + \sum_{e \in E_i} \kappa^i(\alpha(e))$$

ここで，$I_{E'} = (LI, TI, \rho)$，$LI_s = LI - \mathsf{rng}(\mu)$，$E_i = E' - \mathsf{dom}(\mu)$ である．

プレフィックス E' の M に対する**最適マッチングインスタンス**は，$\mathcal{J}_{E'}$ の要素の中で最小のコストを持つマッチングインスタンスである．

最適マッチングインスタンスのコストは定義 6 の式中の分子として表れている．この式の分母はマッチングインスタンスに関わらず定数であるため，最適マッチングインスタンスの順序適合度が最大の適合度となる．

以上により，**ケース E_ℓ のプロセスモデル M に対する順序適合度**は，$\mathcal{J}_{E_\ell}$ 中の最適マッチングインスタンスの順序適合度 (定義 6) と定める．

4.2 最適マッチングインスタンス探索

ケースとプロセスモデルが与えられたとき，そのケースの順序適合度の計算のために最適マッチングインスタンスを求める必要がある．そのために，マッチングインスタンスを節点とする探索空間グラフを定義する．まず，マッチングインスタンス間の遷移関係と距離を定義する．

以下ではオートマトンインスタンス $I = (LI, TI, \rho)$ と $S \subseteq LI$ に対し，$I \cap S$ で，オートマトンインスタンス $(LI \cap S, TI \restriction_S, \rho \restriction_S)$ を表すものとする．ここで $TI \restriction_S = \{(l_1, l_2) \in TI \mid l_1 \in S \wedge l_2 \in S\}$, $\rho \restriction_S$ は ρ の定義域を S に制限した関数である．

定義 10 (マッチングインスタンス間の遷移関係) アクティビティ A 上の有限オートマトンプロセスモデルを $M = (L, l_0, z, A, T, \gamma)$，$A$ 上のイベントログを $L_A = (E, C, \alpha, \beta, \tau, \succ)$，$\ell \in C$ をケース，E_ℓ をケース ℓ のイベント列とする．E_1, E_2 を E_ℓ のプレフィックスとする．$I_1 = (LI_1, TI_1, \rho_1)$, $I_2 = (LI_2, TI_2, \rho_2)$ を M のオートマトンインスタンスとし，$(I_1, \mu_1) \in \mathcal{J}_{E_1}$, $(I_2, \mu_2) \in \mathcal{J}_{E_2}$ とする．このとき，$(I_1, \mu_1) \blacktriangleright (I_2, \mu_2)$ であるのは，以下の三つの条件のうちいずれかを満たすとき，またそのときに限る．

- $E_2 = E_1 \cdot e$ かつ $s_i \in LI_2 - LI_1$ が存在し，$LI_2 = LI_1 \cup \{s_i\}$, $\mu_2(e) = s_i$, $\forall e' \in E_1.\mu_1(e') = \mu_2(e')$, $I_1 = I_2 \cap LI_1$ が成り立つ．すなわち，イベント e が E_1 の末尾に付け加えられたものが E_2 となっており，E_2 のマッチングインスタンス I_2 ではそのイベントに対応するプロセスモデル中の実行が存在する（ケースの進行とモデルの進行が対応している）．
- $E_1 = E_2$ かつ $s_i \in LI_2 - LI_1$ が存在し，$LI_2 = LI_1 \cup \{s_i\}$, $\forall e' \in E_2.\mu_1(e') = \mu_2(e')$, $I_1 = I_2 \cap LI_1$ が成り立つ．すなわち，イベントの一つ (s_i) が状態インスタンス I_2 の末尾のみに追加されている（ケース E_2 ではプロセスモデルのイベントがスキップされた）．
- $E_1 = E_2 \cdot e$ かつ $\forall e' \in E_1.\mu_1(e') = \mu_2(e')$, さらに $\mu_1(e)$ が未定義である．すなわち，イベント e が E_1 の末尾に付け加えられたものが E_2 となっており，E_2 のマッチングインスタンス I_2 ではそのイベントに対応するプロセスモデル中の実行が存在しない（ケース E_2 にイベントが挿入された）．

$(I_1, \mu_1) \blacktriangleright (I_2, \mu_2)$ である場合，マッチングインスタンス (I_1, μ_1) において，ケースとプロセスモデルがともにあるイベント e を実行したか，プロセスモデルのみがイベントを実行してケースはそのままであるか（スキップ），ケースのみがイベントを実行してプロセスモデルは実行しなかったか（挿入），の 3 つの場合のいずれかの状況が (I_2, μ_2) であることを示している．

このようなマッチングインスタンス間の遷移関係に対し，その距離を次のように定義する（定義上は任意のマッチングインスタンス間に距離が定まる）．

定義 11 (マッチングインスタンス間の距離) アクティビティ A 上の有限オートマトンプロセスモデルを $M = (L, l_0, z, A, T, \gamma)$，$A$ 上のイベントログを $L_A = (E, C, \alpha, \beta, \tau, \succ)$，$\ell \in C$ をケース，E_ℓ をケース ℓ のイベント列とする．E_1, E_2 を E_ℓ のプレフィックスとする．I_1, I_2 を M のオートマトンインスタンスとし，$(I_1, \mu_1) \in \mathcal{J}_{E_1}$, $(I_2, \mu_2) \in \mathcal{J}_{E_2}$ とする．このとき，(I_1, μ_1) と (I_2, μ_2) の距離 $\delta((I_1, \mu_1), (I_2, \mu_2))$ を次のように定義する．

$$\delta((I_1, \mu_1), (I_2, \mu_2)) = \delta_n((I_2, \mu_2)) - \delta_n((I_1, \mu_1)) + |E_2| - |E_1|$$

これらの定義をもとに，マッチングインスタンスの探索空間グラフを定義する．ここで，二つの列 σ, σ' に対し，σ が σ' のプレフィックスであるという関係を $\sigma < \sigma'$ と表す．

定義 12 (探索空間グラフ) アクティビティ A 上の有限オートマトンプロセスモデルを $M = (L, l_0, z, A, T, \gamma)$，$A$ 上のイベントログを $L_A = (E, C, \alpha, \beta, \tau, \succ)$，$\ell \in C$ をケース，E_ℓ をケース ℓ のイベント列とする．このとき，E_ℓ の M に対するマッチングインスタンスの探索空間グラフ $G = (V, W, \zeta)$

は以下のように定義される．

- V はノードの集合であり，$V = \bigcup_{E' < E_\ell} \mathcal{J}_{E'}$
- W はエッジの集合であり，$W = \{(v_1, v_2) \in V \times V \mid v_1 \blacktriangleright v_2\}$
- ζ はエッジの重みであり，すべての $(v_1, v_2) \in W$ に対し，$\zeta((v_1, v_2)) = \delta(v_1, v_2)$

最適マッチングインスタンス探索は以下の問題である．

定義 13 (最適マッチングインスタンス探索問題) ケース E_ℓ の $M = (L, l_0, z, A, T, \gamma)$ に対するマッチングインスタンスの探索空間グラフ $G = (V, W, \zeta)$ において，開始ノードを $v_{src} = ((\emptyset, \emptyset, \rho_0), \mu_0) \in \mathcal{J}_{\langle\rangle}$ (ρ_0, μ_0 は空写像), 目標ノードの集合を $V_{trg} = \{((LI, TI, \rho), \mu) \in \mathcal{J}_{E_\ell} \mid \rho(\mathsf{last}((LI, TI, \rho)) = z\}$ とするとき，グラフ G において v_{src} から V_{trg} までの最短経路 $v_{src}, ..., v_{goal}$ を求める．ここでオートマトンインスタンス I に対して $\mathsf{last}(I)$ は I 中で $\mathsf{succ}(l) = \emptyset$ であるようなロケーション l である．

注釈. 目標ノードの集合の定義は Adryansiah らの定義[3] では，オートマトンインスタンスは最終ノードまで到達しないものも目標ノードとして認めていた．この場合，ケースがモデルの途中までを正しく実行できていた場合には適合度は 1 となる．一方，本研究における目標ノードの定義では，オートマトンインスタンスが最終ノードを含むもののみとしている．これにより，ケースがモデルの途中までを実行していた場合には適合度は 1 未満となる．イベントログにおけるケースとは，プロセスの最初から最後までの一つの実行であると通常みなされるため，本定義の方がより適合度の定義として自然である．

定理 1 定義 13 の探索空間グラフにおける最短経路を $v_{src}, ..., v_{goal}$ とすると, v_{goal} は最適マッチングインスタンスである．

証明 探索空間グラフ中の v_{src} から V_{trg} への経路全体の集合を Π とする．また，$\pi \in \Pi$ を $\pi = \langle (I_1, \mu_1), ..., (I_n, \mu_n)\rangle, (I_i, \mu_i) \in \mathcal{J}_{E_i}$ とする．すると，$\delta((I_i, \mu_i), (I_{i+1}, \mu_{i+1})) = \delta_n((I_{i+1}, \mu_{i+1})) - \delta_n((I_i, \mu_i)) + |E_{i+1}| - |E_i|$ より，この経路の距離 $\mathsf{len}(\pi)$ は $\sum_{k=1}^{n-1} \delta((I_k, \mu_k), (I_{k+1}, \mu_{k+1})) = \delta_n((I_n, \mu_n)) - \delta_n((I_1, \mu_1)) + |E_n| - |E_1|$ が成り立つ．ここで，$(I_1, \mu_1) = v_{src}$ より，$\delta((I_1, \mu_1)) = 0 = |E_1|$ である．また，$E_n = E_\ell$ である．したがって，$\mathsf{len}(\pi) = \delta_n((I_n, \mu_n)) + |E_n| = \delta_n(\mathsf{last}(\pi)) + |E_\ell|$ である．これより，$\arg\min_{\pi \in \Pi} \mathsf{len}(\pi) = \arg\min_{\pi \in \Pi}(\delta_n(\mathsf{last}(\pi)) + |E_\ell|) = \arg\min_{\pi \in \Pi} \delta_n(\mathsf{last}(\pi))$ ($|E_\ell|$ は定数のため)．これより，最短経路となる π の最後のインスタンス $\mathsf{last}(\pi) = v_{goal}$ において，$\delta_n(v_{goal})$，すなわち定義 6 の適合度の式の分子が最小となるため，適合度は最大となる．これより v_{goal} は最適マッチングインスタンスである． □

4.3 探索アルゴリズム

順序適合度だけを考えるのであれば，最適マッチングインスタンスをひとつ求めるだけでよいが，本研究では，得られた最適マッチングインスタンスに対しさらに時間適合度を評価する必要がある．その結果，最適マッチングインスタンスであっても，総合的な適合度は異なるものが存在する可能性がある．よって定義 8 における最良の適合度を持つインスタンスを発見するためには，最適マッチングインスタンスをすべて求める必要がある．

注釈. あらゆるマッチングインスタンスの中で定義 8 における最良の（時間適合度も合わせた）適合度を持つインスタンスは，必ずしも最適マッチングインスタンスから得られるとは限らない．しかし，本研究では，ケースの順序適合度を最適なアライメントを持つマッチングインスタンスの順序適合度と定めているため，そのようなマッチングインスタンスは除外される．

探索空間グラフの節点集合は一般に無限集合であり，モデルにループがある場合任意回のループ繰り返し（スキップ）が可能となるため最適なマッチングインスタンスの探索が停止しない可能性がある．しかし，本研究ではすべてのイベントのスキップコスト (関数 κ_s で定まる) は正であると仮定しているため，幅優先探索などにより最適なマッチングインスタンスの探索は有限時間で停止することが保証できる．

そこで，一つの探索アルゴリズムとして，幅優先探

索による最適なマッチングインスタンスの**全探索アルゴリズム**が考えられる．

一方，全探索アルゴリズムの実行は非効率的であるため，ケースやプロセスモデルが大きくなる場合，合理的な時間で探索を終えることが困難になることが予想される．そこで，A^***アルゴリズム**を用いて最適なマッチングインスタンスの探索を行うことが考えられる．A*アルゴリズムによる探索では，可能なすべての最適なマッチングインスタンスが発見される保証はないため,（時間適合度も合わせた）最良の適合度を持つマッチングインスタンスが得られるとは限らない．そのため，効率的だが近似的なアルゴリズムとなる．

A*アルゴリズムを最適なマッチングインスタンスの探索に適用するためには，ヒューリスティック関数を適切に定める必要がある．本研究では定義 12 の探索空間グラフ (V, W, ζ) において，ヒューリスティック関数 $h : V \to \mathbb{N}$ を，$v \in \mathcal{J}_{E'}$ に対し $h(v) = |E_\ell| - |E'|$ と定める．$h(v)$ は任意のノード v から V_{trg} 中のノードへのコストを過小評価する関数となっている．すなわち，すべての $v_{trg} \in V_{trg}$ に対し，$h(v) \leq \zeta(v, v_{trg})$ が成り立つ [3]．このヒューリスティック関数をもとに評価関数 f を，$f(v) = g(v) + h(v)$ と定める．ここで $g(v)$ は v_{src} から探索中のノード v までの最小コストを返す関数である．

アルゴリズム 1 に A*アルゴリズムによる最適マッチングインスタンス探索アルゴリズムを示す．

A*アルゴリズムが確かに目標ノードの一つを発見することを保証するには，以下の 3 つを証明する必要がある．(1) 少なくとも一つの目標ノードが到達可能である，(2) ヒューリスティック関数が目標ノードまでの距離の過小評価となっている，(3) 評価関数は単調増加である．これらの事実は文献 [3] と同様に証明可能である．

5 実験と評価

この節ではいくつかの時間オートマトンプロセスモデルとケースに対する，本提案手法によるコンフォーマンスチェッキングの実行結果について述べる．その際，最適マッチングインスタンスの探索に A*アルゴリズムを用いた場合と，全探索で求めた場合のそれぞれについて，所要時間と適合度の値の違い，および探索されたノード数と探索に要した時間について比較する．

5.1 実験

探索アルゴリズムは Python で実装した．この実装においては時間オートマトンは時間オートマトンのモデル検査器 UPPAAL[†1] [8] で用いられる .xml 形式ファイルで与え，ケースはアクティビティと数値の対の列としてテキストファイルで与えることとした．また，各アクティビティのスキップコスト，挿入コストはどれも 1 であるとした．実験には AMD Ryzen 7 3800XT 3.9GHz 32GB メモリの計算機を用いた．

ここでは三つの時間オートマトンモデルを考える．

一番目のモデルは一つのループのみをもつ単純な時間オートマトンモデルである（図 6）．このモデルに対しケース $\langle (a, 10), (b, 15), (c, 25), (b, 15), (d, 0) \rangle$ を与えたときの実験結果を表 2 に示す．

表の「最適インスタンス」は，発見された最適マッチングインスタンスを表し，f_{order} はその順序適合度を，f_{time} は時間適合度を，*fitness* は総合適合度を表している．「探索ノード数」は探索空間グラフにおいて最適マッチングインスタンスを発見するまでに探索したノード数を，「探索時間」は最適マッチングインスタンスの探索に要した時間を表す．

二番目のモデルはループ内に分岐を含むモデルである（図 7）．このモデルに対しケース $\langle (a, 5), (b, 10), (c, 5), (e, 15), (b, 10), (e, 15) \rangle$ を与えたときの実験結果を表 3 に示す．

三番目の時間オートマトンモデルは分岐の後にループを含むモデルである（図 8）．このモデルに対しケース $\langle (a, 10), (d, 15), (e, 10), (d, 10), (f, 0) \rangle$ を与えたときの実験結果を表 4 に示す．このケースでは分岐の中のアクティビティ (b と c) がスキップされていることに注意されたい．

†1 https://uppaal.org/

アルゴリズム 1 A*アルゴリズムによる最適マッチングインスタンス探索

Input: オートマトンモデル $M = (L, l_0, z, A, T, \gamma)$, ケース E

Output: 最適インスタンス

$n \leftarrow \{EI : [], LI : [], \rho : [], \mu : [], f : []\}$ # 初期ノード．EI は E のプレフィックス
$Nodes \leftarrow \{n\}$;
$OI \leftarrow \{\}$; # 最適インスタンスの集合
while $Nodes \neq \emptyset$ **do**
  $n = Nodes.\mathsf{pop}()$; $m \leftarrow \min\{k \mid k \leftarrow x.f, x \in Nodes\}$;
  if $n.f > m$ **then**
    continue while # 評価関数による枝刈り
  end if
  if $\exists x \in \mathsf{next}(\rho(\mathsf{last}(n.LI))).\gamma(x) = E[\mathsf{len}(n.EI) + 1]$ **then**
    let x be such location # γ は単射なので x は一意
    $y \leftarrow \mathsf{newlocinstance}()$; $n' \leftarrow n.\mathsf{copy}()$;
    $n'.EI.\mathsf{push}(E[\mathsf{len}(n.EI) + 1])$; $n'.LI.\mathsf{push}(y)$; $n'.\rho.\mathsf{push}((y, x))$; $n'.\mu.\mathsf{push}((E[\mathsf{len}(n.EI) + 1], y))$;
    $n'.f \leftarrow n.f + \delta_n(n') - \delta_n(n) + \mathsf{len}(n'.EI) - \mathsf{len}(n.EI)$;
    $Nodes.\mathsf{push}(n')$; # モデルとケースの実行に対応
  end if
  for each $x \in \mathsf{next}(\rho(\mathsf{last}(n.LI)))$ **do**
    $y \leftarrow \mathsf{newlocinstance}()$; $n' \leftarrow n.\mathsf{copy}()$;
    $n'.LI.\mathsf{push}(y)$; $n'.\rho.\mathsf{push}((y, x))$;
    $n'.f \leftarrow n.f + \delta_n(n') - \delta_n(n) + \mathsf{len}(n'.EI) - \mathsf{len}(n.EI)$;
    $Nodes.\mathsf{push}(n')$; # モデルのみの実行に対応
  end for
  $n' \leftarrow n.\mathsf{copy}()$;
  $n'.EI.\mathsf{push}(E[\mathsf{len}(n.EI) + 1])$;
  $n'.f \leftarrow n.f + \delta_n(n') - \delta_n(n) + \mathsf{len}(n'.EI) - \mathsf{len}(n.EI)$;
  $Nodes.\mathsf{push}(n')$; # ケースのみの実行に対応
  for each $n \in Nodes$ **do**
    if $n.EI = E \wedge \rho(\mathsf{last}(n.LI)) = z$ **then**
      $OI.\mathsf{push}(n)$;
      $Nodes.\mathsf{remove}(n)$;
    end if
  end for
end while
return OI

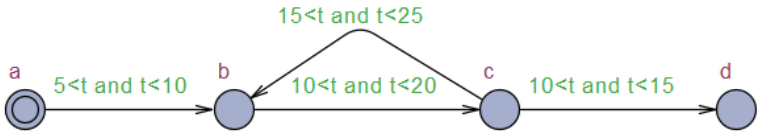

図 6　一番目の時間オートマトンモデル．ロケーション名はアクティビティ名を示している．

表 2　図 6 のモデルにケース $\langle (a,10),(b,15),(c,25),\ (b,15),(d,0)\rangle$ を与えたときの実験結果．

アルゴリズム	最適インスタンス	f_{order}	f_{time}	$fitness$	探索ノード数	探索時間
A*	$\langle a,b,c,b,c,d\rangle$	0.889	1.000	**0.945**	20	2.292×10^{-4}
全探索	$\langle a,b,c,d\rangle$	0.889	0.778	0.834	1138	1.555×10^{-2}
	$\langle a,b,c,b,c,d\rangle$	0.889	1.000	**0.945**		

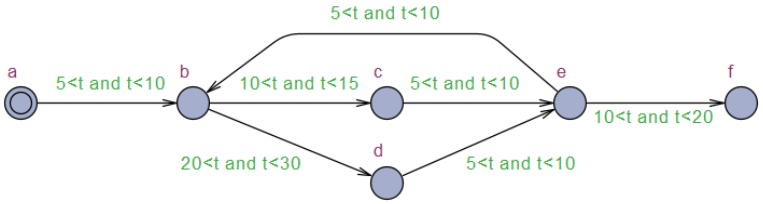

図 7　二番目の時間オートマトンモデル．ロケーション名はアクティビティ名を示している．

表 3　図 7 のモデルにケース $\langle (a,5),(b,10),(c,5),\ (e,15),(b,10),(e,15)\rangle$ を与えたときの実験結果．

アルゴリズム	最適インスタンス	f_{order}	f_{time}	$fitness$	探索ノード数	探索時間
A*	$\langle a,b,c,e,b,c,e,f\rangle$	0.909	0.917	**0.913**	38	5.703×10^{-4}
全探索	$\langle a,b,c,e,b,c,e,f\rangle$	0.909	0.917	**0.913**	12441	1.784×10^{-1}
	$\langle a,b,c,e,b,d,e,f\rangle$	0.909	0.833	0.871		

5.2　評価

三つの実験の結果を見ると，全探索アルゴリズムにより発見されたそれぞれの最適インスタンスに対して順序適合度はどれも同じ値であるが，時間適合度は最適インスタンスごとに異なる値となっている．すなわち，最適マッチングインスタンスであっても，時間適合度の差が生じることが実験によって確かめられた．さらに，図 8 のモデルの実験結果に注目すると，A*アルゴリズムで形成された最適インスタンスの最終適合度は 0.784 であるのに対して，全探索で形成された最適インスタンスの中で最終適合度が一番高いものは 0.859 であることが分かる．これにより，A*アルゴリズムでは複数存在する最適インスタンスの中から，最も現実に適合している実行経路を発見できるわけではないことが明らかになった．順序適合度だけでケースの適合度を判定すると，複数の最適インスタンスが存在するときに適合度が全て同じ値であるため，現実に起きた実行列と最も近いモデルの実行経路を特定するのは困難であるが，時間適合度の値も考慮することによってそのような実行経路を特定できるという利点が示された．今回の実験では，アクティビティの順序と時間の二つの要素を扱ったが，これら以外も考慮することによって複数の最適インスタンスに対して，より精度の高い適合度を評価することが可能になると考えられる．

図 8 のモデルの実験についてもう少し詳しく分析する．このモデルとケースでは，時間属性を考慮しない場合にはアクティビティ b をスキップしたと解釈し

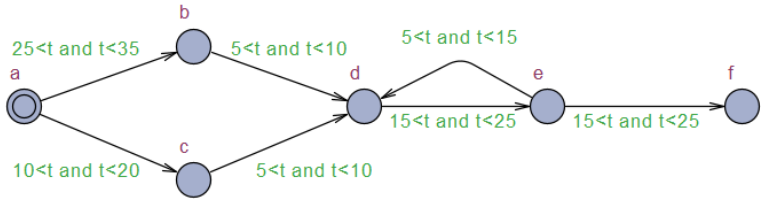

図 8　三番目の時間オートマトンモデル．ロケーション名はアクティビティ名を示している．

表 4　図 8 のモデルにケース $\langle (a,10),(d,15),(e,10),\ (d,10),(f,0)\rangle$ を与えた時の実験結果．

アルゴリズム	最適インスタンス	f_{order}	f_{time}	$fitness$	探索ノード数	探索時間
A*	$\langle a,b,d,e,d,e,f\rangle$	0.800	0.767	0.784	42	5.653×10^{-4}
全探索	$\langle a,b,d,e,f\rangle$	0.800	0.689	0.744	3491	4.822×10^{-2}
	$\langle a,c,d,e,f\rangle$	0.800	0.889	0.845		
	$\langle a,b,d,e,d,e,f\rangle$	0.800	0.767	0.784		
	$\langle a,c,d,e,d,e,f\rangle$	0.800	0.917	**0.859**		

た場合でも c をスキップしたと解釈した場合でも適合度は変わらない（スキップコストがどちらも等しいため）．しかし，時間属性を考慮すると a の完了時刻は 10 であるため，c をスキップしたと解釈した場合に時間制約の違反は発生しないため，より高い適合度が得られる．このことは表 4 の結果からもわかり，全探索により適合度を求めた場合には，アクティビティ a の後に c を実行したと解釈した場合 (2 番目と 4 番目のマッチングインスタンス) の方が，アクティビティ a の後に b を実行したと解釈した場合 (1 番目と 3 番目のマッチングインスタンス) よりも高い時間適合度の値が得られていることからわかる．

コンフォーマンスチェッキングにはプロセスモデルが規範的でありケース (現実に起きたこと) に誤りがあったという解釈と，現実に起こったケースが正当でありプロセスモデル側が不完全という二つの解釈が可能である．前者の解釈では，適合度が最も高いプロセスモデル中の実行経路が，現実に起きたケースが意図した実行であったことになる．後者の解釈では，最も適合度の高い経路にプロセスモデルを改善する鍵があることを示唆していると考えられる．いずれの場合でも，順序適合度だけでは特定が難しかった最適な経路をこの実験に限れば一つに絞り込めているため，現実に起きたケースのどこに誤りがあったかの分析や，現実と乖離しているモデル個所の特定とモデル改善などを効果的に行うことが可能となった．

探索にかかる所要時間については，A*アルゴリズムの方が全探索に対して一番目の実験では約 68 倍，二番目の実験では約 313 倍，三番目の実験では約 85 倍速いことが分かった．現実の業務プロセスは，今回の実験例よりも遥かに複雑であることの方が多いため，A*アルゴリズムのように効率的かつ，時間適合度も考慮したうえで最良のマッチングインスタンスを発見する近似アルゴリズムが必要になる．最適マッチングインスタンスの探索問題は NP 困難であることが示されている [5] ため，モデルの規模が大きくなった場合何らかの近似的な探索法が不可欠である．

6　関連研究

1 節で述べた通り，ほとんどのコンフォーマンスチェッキング手法はアクティビティの順序に関するものであるが，属性値やタイムスタンプに関するコンフォーマンスチェッキングあるいはアライメントの研究が皆無なわけではない．

Leoni らはイベントの属性に着目したコンフォーマンスチェッキング手法を提案した [6]．この研究では，プロセスモデルは変数への代入文をもつ（右辺は定数のみ）．アクティビティの実行の際にその代入文が実

行され変数の値が更新される．ケースとモデルとのアライメントは通常通り定まるが，ミスアライメントでないアクティビティの中で属性値の割り当てが異なるものの個数を数え，その個数をもとに適合度が評価される．すなわち，このコンフォーマンスチェッキング手法は，属性値の不一致の個数のみを用いている．一方，我々の提案手法では単に不一致の有無だけでなく，その不一致の度合まで加味して適合度を評価している．

Giacomo らは時間付きトレースの有限列上の Metric Temporal Logic (MTL_f) に対するアライメントを提案した[7]．これは，時間付きトレースと MTL の論理式が与えられたとき，式を満たさないトレースに対しイベント (アクティビティとタイムスタンプの対) の追加や削除で式を満たすように編集して得られるトレースのうち，編集コストが最小となるものを求める．この研究では適合度の計算は対象としていないが，もし適合度を定めるならば必然的に挿入と削除の個数で評価することになるため，本質的に順序適合度となる．したがってタイムスタンプがどれだけ制約を満たしていないかの度合の評価はできない．また，この手法は非初等的な計算量 (NONELEMENTARY) となっており，実用が困難である．

7 まとめと今後の課題

本稿では，時間オートマトンによるプロセスモデルに対し，時間属性を考慮した上でイベントログの適合度を評価するコンフォーマンスチェッキング手法を提案した．従来のアクティビティの順序にもとづくコンフォーマンスチェッキング手法を拡張し，得られたマッチングインスタンスにおいて，各イベントのタイムスタンプと対応するモデル上の遷移の時間制約を比較し，その制約が定める区間からどれだけそのタイムスタンプが逸脱しているかを評価する式を与え，時間適合度を定義した．この評価方法に基づくコンフォーマンスチェッキング手法を A*アルゴリズムと全探索の二通りのアルゴリズムにより実装し，単純なモデルとケースを用いて実験を行った．その結果，得られた複数の（順序適合度の意味で）最適インスタンスに対しても，時間適合度を評価することにより適合度に差異が生じることが確認された．また，時間適合度も含めた総合的な適合度を用いることにより，その中で最もケースの実行に近いモデルの実行経路を特定することが可能であることが示された．

今後の課題としては，最適インスタンスを形成するアルゴリズムの改善が考えられる．今回の実験では全探索によって複数の最適インスタンスを形成したが，実際の業務モデルでは探索するノード数が増大し，探索コストが膨大にかかる可能性が考えられるので，より効率的かつ網羅的に探索できるアルゴリズムの考案が望まれる．

謝辞 この研究は JSPS 科研費 JP21K11756 の助成を受けています．

参 考 文 献

[1] Adriansyah, A.: *Aligning Observed and Modeld Behavior*, PhD Thesis, Technische Universiteit Eindhoven, 2014.

[2] Adriansyah, A., Sidorova, N., and van Dongen, B. F.: Cost-based fitness in conformance checking, *11th International Conference on Application of Concurrency to System Design, ACSD 2011, Newcastle Upon Tyne, UK, June 20-24, 2011*, 2011, pp. 57–66.

[3] Adriansyah, A., van der Aalst, W. M. P., and van Dongen, B. F.: Conformance checking using cost-based fitness analysis, *15th IEEE International Enterprise Computing Conference, EDOC 2011, Helsinki, Finland, August 29-Spetember 2, 2011*, 2011, pp. 55–64.

[4] Carmona, J., van Dongen, B., Solti, A., and Weidlich, M.: *Conformance Checking*, Springer, 2018.

[5] de Leoni, M. and van der Aalst, W. M. P.: Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming, *Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*, Daniel, F., Wang, J., and Weber, B.(eds.), Lecture Notes in Computer Science, Vol. 8094, Springer, 2013, pp. 113–129.

[6] de Leoni, M., van der Aalst, W. M. P., and van Dongen, B. F.: Data- and Resource-Aware Conformance Checking of Business Processes, *Business Information Systems - 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings*, Abramowicz, W., Kriksciuniene, D., and Sakalauskas, V.(eds.), Lecture Notes in Business Information Processing, Vol. 117, Springer, 2012, pp. 48–59.

[7] Giacomo, G. D., Murano, A., Patrizi, F., and

Perelli, G.: Timed Trace Alignment with Metric Temporal Logic over Finite Traces, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, Bienvenu, M., Lakemeyer, G., and Erdem, E.(eds.), 2021, pp. 227–236.

[8] Larsen, K. G., Petterson, P., and Yi, W.: UPPAAL in a nutshell, *International Journal on Software Tools for Technology Transfer*, Vol. 1, No. 1/2(1997), pp. 134–152.

[9] Polyvyanyy, A., Weidlich, M., Conforti, R., Rosa, M. L., and ter Hofstede, A. H.: The 4C spectrum of fundamental behavioral relations for concurrent systems, *32th International Conference in Application and Theory of Petri Nets and Concurrency, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Volume 8489 of Lecture Notes in Computer Science*, Springer, 2014, pp. 210–232.

[10] Rozinat, A.: *Process Mining Conformance and Extension*, PhD Thesis, Technische Universiteit Eindhoven, 2010.

[11] van der Aalst, W. M. P., Adriansyah, A., and van Dongen, B. F.: Replaying history on process models for conformance checking and performance analysis, *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, Vol. 2, No. 2(2012), pp. 182–192.

[12] van der Aalst, W. M.: *Process Mining: Data Science in Action*, Springer, 2016.

[13] vanden Broucke, S. K. L. M., Munoz-Gama, J., Carmona, J., Baesens, B., and Vantheienen, J.: Event-based real-time decomposed conformance analysis, *On the Move to Meaningful Internet Systems, OTM 2014, Amantea, Italy, October 27-31, 2014*, 2014, pp. 345–363.

[14] vanden Broucke, S. K. L. M., Weerdt, J. D., Vantheienen, J., and Baesens, B.: Determining process model precision and generalization with weighted artificial negative events, *IEEE Trans. Knowl. Data Eng.*, Vol. 26, No. 8(2014), pp. 1877–1889.

[15] Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., and Weske, M.: Process compliance analysis based on behavioural profiles, *Inf. Syst.*, Vol. 36, No. 7(2011), pp. 1009–1025.

[16] Weidlich, M., Polyvyanyy, A., Mendling, J., and Weske, M.: Causal behavioural profiles – Efficient computation, applications, and evaluation, *Fundam. Inform.*, Vol. 113, No. 3–4(2011), pp. 399–435.

[17] Zha, H., Wang, J., Wen, L., Wang, C., and Sun, J.: A workflow net similarity measure based on transition adjacency relations, *Computers in Industry*, Vol. 61, No. 5(2010), pp. 463–471.