確率的データストリームにおけるパターン照合結果のグループ化

杉浦　健人†　　佐々木勇和††　　石川　佳治†,†††

† 名古屋大学大学院情報科学研究科
†† 名古屋大学未来社会創造機構
††† 国立情報学研究所
E-mail: {sugiura, yuya}@db.ss.is.nagoya-u.ac.jp, ishikawa@is.nagoya-u.ac.jp

あらまし　近年，複数のイベントを組み合わせることでより高次のイベントを発見する複合イベント処理に注目が集まっている．中でも，複合イベントを発見する手法の一つであるパターン照合は，その有用さから広く研究が行われている．しかし，各イベントに生起確率が付与される確率的データストリームでは，パターン照合を行ったとき同じ時間帯に複数のマッチが重複して発見されるという問題がある．このように重複して存在するマッチは，いずれもその時間帯にパターンが存在した可能性を示しており，個々を区別することは必ずしも適当ではない．そこで本稿では，重複したマッチをグループとしてひとまとめにする手法を提案する．まず，グループの確率的な意味について考えるために，確率的データストリームに対してパターン照合を行った際の確率空間を定義する．その後，グループを生成するためのセマンティクスとして完全オーバラップと部分オーバラップを定義し，マッチの生成と同時にグループを生成する手法を提案する．最後に，トランスデューサを用いてグループの確率を効率的に計算する手法を提案し，評価実験により本手法の有効性を示す．

キーワード　複合イベント処理（CEP）, 確率的データストリーム, パターン照合, グループ化

1. はじめに

スマートフォンのように多数のセンサを持つ機器が普及したことで，得られたセンサ情報を有効活用しようとする動きが盛んである．特に，得られた一つ一つの情報をイベントとみなし，複数のイベントを組み合わせることで高次のイベントを発見する**複合イベント処理**（CEP）が注目されている．本稿では，CEP で用いられる手法の一つとして，その有用性から幅広く研究が行われている**パターン照合**に注目する [2,3,5,11–15,17,20]．

パターン照合に関する大部分の既存研究は入力となる情報の曖昧性を考慮していない．しかし情報源がセンサ機器などである場合，センサ自体の計測誤差や情報伝達時の欠落などによって，得られる情報は曖昧になる．このような曖昧さは確率によって表せるため，入力される情報は図 1 のような**確率的データストリーム**となる．確率的データストリームに対してパターン照合を行う場合，図 2 のように同じ時間帯に複数のマッチが発見される．これらのうち生起確率の小さいものは情報の曖昧さによって生成されたと考えられるため，既存手法ではそのようなものを除くことでユーザへ返すマッチを限定している [10]．

しかし，同じ時間帯で発見されるマッチはいずれもその時間帯に該当するパターンが存在した可能性を示し，個々を区別することは必ずしも適切ではない．例えば，図 2 のマッチはいずれも時区間 $[1:6]$ の間にパターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ が存在した可能性を示している．なお，本稿では $[t_s:t_e]$ で t_s から t_e までの閉区間を，$(t_s:t_e)$ で開区間を表す．時区間 $[1:6]$ に存在するマッチを一つにまとめ，時区間 $[1:6]$ の間にパターンに一致するイベントが生起した確率を考えると，その確率は約 0.94 となる．これは最も生起確率の高いマッチ m_2 と比べても非常に高く，同じ時間帯に存在するマッチをまとめることでより正確にパターンが発見できることを示している．そこで，本稿では同じ時間帯に存在する複数のマッチを一つの**グループ**にまとめる手法を提案し，グループの定義や生成方法について述べる．

時刻		1	2	3	4	5	6
イベント	a	1.0	0.3	0.1	0.1	0	0
	b	0	0.7	0.8	0.7	0.9	0
	c	0	0	0.1	0.2	0.1	1.0

図 1　確率的データストリーム

マッチ	時刻						確率
	1	2	3	4	5	6	
m_1	a	b	c				0.07
m_2	a	b	b	b	b	c	0.3528
m_3		a	b	b	c		0.0168
m_4				a	b	c	0.09

図 2　パターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ に対するパターン照合結果の一部

本稿の構成は以下の通りである．まず，グループの生成について厳密に議論するために，本稿で用いる仮定の説明や確率的データストリームに対してパターン照合を行った際の確率空間の定義など，議論のための準備を行う．次に，マッチをグループにまとめるためのセマンティクスを提案し，グループの定義を述べる．その後，グループの生成方法とグループ確率の効率的な計算方法を紹介する．最後に，グループの特徴を評価するために行った実験について説明し，本稿のまとめを述べる．

表 1　本稿で使用する記号の一覧

記号	意味
e_t	時刻 t の確率的イベント
D	イベントの離散的なドメイン
α_t	$e_t = \alpha$ $(\alpha \in D)$
PDS	確率的データストリーム
w	確率的データストリームにおける可能世界
$W_{[i:j]}$	時区間 $[i:j]$ の可能世界の全集合
p	問合せパターン
m	パターンに対応するマッチ
r	生成途中のマッチ
g	グループ
$T_{t-1,t}$	トランスデューサの遷移行列
V_t	トランスデューサの各状態にいる確率

2. 準　　備

本章では，確率的データストリームに対してパターン照合を行う際に用いる仮定の説明や，照合結果に対する確率空間の定義など，グループについて議論するための準備を行う．また，本稿で使用する記号の一覧を表 1 に示す．

本稿で用いる仮定．本稿では，確率的データストリームは以下の二つの性質を持つと仮定する．

（1） 各イベントは単位時間毎に生起し，生起した順にシステムに入力される．

（2） 異なる時刻に生起したイベントの生起確率は互いに独立である．

例えば図 1 は，ある単位時間に従って各イベントが生起している例である．なおこれ以降，ある時刻 t に対して 1 単位時間分進んだ時刻を $t+1$ で表す．

確率空間の定義．グループの確率を考えるために確率的データストリームにおける確率空間を定義する．まず，確率的データストリームの要素となる**確率的イベント**を次のように定める．

[定義 1] **確率的イベント** e_t は，イベントの各属性値 $\alpha \in D$ に対して生起確率 $P(e_t = \alpha)$ を持つ時刻 t のイベントである．ただし，D はイベントの離散的なドメインである．また，イベントの生起確率 $P(e_t = \alpha)$ は以下の式を満たす．

$$\forall \alpha \in D,\ 0 \leqq P(e_t = \alpha) \leqq 1$$

$$\sum_{\alpha \in D} P(e_t = \alpha) = 1 \qquad \square$$

そして，確率的イベントを用いて**確率的データストリーム**を以下のように定義する．

[定義 2] **確率的データストリーム** $PDS = \langle e_i, e_{i+1}, ..., e_j, ... \rangle$ は，確率的イベントの系列である． $\square$

例えば，図 1 の確率的データストリームは，ドメインが $D = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ である確率的イベントの系列 $PDS = \langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle$ として表すことができる．なおこれ以降，時刻 t にイベントが $\alpha \in D$ であることを α_t で表す．

次に，確率的データストリームにおける可能世界を定義する．可能世界に基づいた確率計算は確率的データベースにおいて提案された手法で，確率的なデータの集合に対してそれがとりうるすべての可能性を列挙することで確率を考える [4]．時刻 t において生起確率が 0 より大きい属性の集合を D_t で表し，$W_{[t:t+1]} = D_t \times D_{t+1}$ で時刻 t と $t+1$ におけるイベントの系列の全集合を表すとして，確率的データストリームにおける可能世界を次のように定義する．

[定義 3] 有限長の確率的データストリーム $PDS = \langle e_i, e_{i+1}, ..., e_j \rangle$ が与えられたとき，PDS の可能世界 w の全集合は $W_{[i:j]} = D_i \times D_{i+1} \times ... \times D_j$ である．また，可能世界 $w = \langle \alpha_i, \alpha_{i+1}, ..., \alpha_j \rangle$ に対する確率は，$P(w) = \prod_{\alpha_t \in w} P(\alpha_t)$ で与える． $\square$

例えば図 1 の確率的データストリームの時区間 $[2:3]$ について考えると，$W_{[2:3]} = \{\mathsf{a}_2, \mathsf{b}_2\} \times \{\mathsf{a}_3, \mathsf{b}_3, \mathsf{c}_3\} = \{\langle \mathsf{a}_2, \mathsf{a}_3 \rangle, \langle \mathsf{a}_2, \mathsf{b}_3 \rangle, \langle \mathsf{a}_2, \mathsf{c}_3 \rangle, \langle \mathsf{b}_2, \mathsf{a}_3 \rangle, \langle \mathsf{b}_2, \mathsf{b}_3 \rangle, \langle \mathsf{b}_2, \mathsf{c}_3 \rangle\}$ である．また，可能世界 $\langle \mathsf{a}_2, \mathsf{b}_3 \rangle$ の確率は $P(\langle \mathsf{a}_2, \mathsf{b}_3 \rangle) = P(\mathsf{a}_2) \times P(\mathsf{b}_3) = 0.24$ である．

上述した可能世界を用いて，確率的データストリームにおける確率空間を次のように定義する．

[定義 4] 有限長の確率的データストリーム $PDS = \langle e_i, e_{i+1}, ..., e_j \rangle$ に対する確率空間は $(2^{W_{[i:j]}}, P)$ である．ただし，$2^{W_{[i:j]}}$ は $W_{[i:j]}$ のべき集合を，P は $x \in 2^{W_{[i:j]}}$ に対して $P(x) = \sum_{w \in x} P(w)$ で確率を与える関数である． $\square$

問合せパターンとマッチの定義．本稿では，問合せパターンは $p = \langle \mathsf{a}\ \mathsf{b}^+ \mathsf{c} \rangle$ のようにイベントの系列として入力されると想定する．なお，添字の $^+$ はイベントの繰返しを表すクリーネ閉包である．クリーネ閉包を使用したパターン，例えば $\langle \mathsf{a}\ \mathsf{b}^+ \mathsf{c} \rangle$ が与えられた場合，マッチとして $\langle \mathsf{a}_t, \mathsf{b}_{t+1}, \mathsf{c}_{t+2} \rangle, \langle \mathsf{a}_t, \mathsf{b}_{t+1}, \mathsf{b}_{t+2}, \mathsf{c}_{t+3} \rangle$ などが発見される．

また，現実に発生するイベントの記述を想定し，パターンの記述に対する以下の三つの制限を提案する．

（1） 最初と最後以外のイベントにクリーネ閉包を付与する．

（2） 最初と最後のイベントにはクリーネ閉包を付与しない．

（3） 同じイベントを連続で記述しない．

このような制限を用いる理由は， 現実に発生するイベントを考えた場合イベントがある単位時間にもれなく収まるとは考えづらいためである．例えば人の行動モニタリングであれば，歩く・走るなどの動作が常にある単位時間（一秒，五秒など）毎に切り替わるとは考えられない．つまり，ほとんどのイベントは複数の時刻にまたがって存在していると言える．このようなイベントの検出を考えた場合，イベントの複数回の生起を前提とする方が妥当であるため，パターンの記述ではクリーネ閉包の使用を基本とする（制限 1）．ただしパターンの最初と最後に関しては，それ以前ないし以降のイベントとの切れ目として考えられるため，クリーネ閉包は使用しない（制限 2）．また，クリーネ閉包の使用を基本とした場合同じイベントを連続して記述する意味が無いため，そのような記述を禁止する（制限 3）．

次に，本稿におけるマッチとその確率の定義を以下に示す．

[定義 5] マッチ m はユーザが指定したパターンに対応するイベントの系列である．確率空間 $(2^{W_{[i:j]}}, P)$ に対して，m を部分系列に含む可能世界の集合を $W_m \subseteqq W_{[i:j]}$ とするとき，マッチの確率は $P(m) = \sum_{w \in W_m} P(w)$ である． $\square$

例えば，図 1 における $W_{[1:4]}$ の可能世界で考えると，図 2 の

m_1 の確率は以下の可能世界の確率の和となる．

$w_1 = \langle \mathtt{a_1}, \mathtt{b_2}, \mathtt{c_3}, \mathtt{a_4} \rangle,\ P(w_1) = 0.007$

$w_2 = \langle \mathtt{a_1}, \mathtt{b_2}, \mathtt{c_3}, \mathtt{b_4} \rangle,\ P(w_2) = 0.049$

$w_3 = \langle \mathtt{a_1}, \mathtt{b_2}, \mathtt{c_3}, \mathtt{c_4} \rangle,\ P(w_3) = 0.014$

つまり，$P(m_1) = P(w_1) + P(w_2) + P(w_3) = 0.07$ である．

3. グループの定義

マッチを一つのグループにまとめること考えたとき，最も単純な方法はある幅の時間窓を設定し，時間窓の範囲内に存在するマッチをすべてひとまとめにすることである．例えば図 2 の場合，幅 6 単位時間の時間窓を設定すれば適当なグループが得られる．しかし，マッチの時間幅は入力によって大きく変化するため，静的な時間窓では適当なグループの取得は難しい．

そこで，マッチ同士の時間的なオーバラップに注目してグループに含めるか決定する手法を提案する．マッチが時間的にオーバラップするとは，図 2 における m_1, m_2 の時刻 $2, 3$ のように，二つのマッチがある時刻で互いに何らかのイベントを持つことである．これ以降，$ts\_overlap(m_i, m_j)$ を m_i, m_j が時間的にオーバラップすることを示す述語記号として扱う．

マッチの時間的なオーバラップに注目したとき，次の問題はどのようなマッチの集合をグループとするかである．本稿では，階層的クラスタリングにおける単連結法 (single-link method) と完全連結法 (complete-link method) の考えを参考にグループとなるマッチの集合を定義する．単連結法は，あるクラスタ内に存在するすべてのドキュメントが同じクラスタ内に少なくとも一つ類似なものを持つようクラスタリングする手法である．一方，完全連結法では，あるクラスタ内に存在するすべてのドキュメントが互いに類似となるようクラスタを生成する．以下では，各手法を参考に提案する部分オーバラップと完全オーバラップについて順に説明する．

3.1 部分オーバラップに基づく定義

単連結法に基づき，**部分オーバラップ**を次のように定義する．

[定義 6] マッチの集合 M が以下の式を満たすとき，M は**部分オーバラップ**の性質を持つ．

$$\forall m_i \in M, \exists m_j \in M, m_i \neq m_j \wedge ts\_overlap(m_i, m_j) \qquad \square$$

部分オーバラップは，グループ内のマッチが他のいずれかのマッチとオーバラップすることを保証する．例えば図 2 の場合，m_2, m_3 がすべてのマッチとオーバラップするため，四つのマッチすべてを含んだグループが生成される．

部分オーバラップにより生成されるグループは，言い換えれば，ある時区間に存在するマッチをすべてひとまとめにしたものである．先ほどの例の場合，生成されたグループは時区間 $[1:6]$ のマッチをひとまとめにしたものとみなせる．そこで，部分オーバラップに基づくグループを以下のように定義する．

[定義 7] グループは部分オーバラップの性質を持つ極大なマッチ集合である．グループは $g = (t_s, t_e)$ のように，グループの開始時刻，終了時刻の組で表す． $\square$

3.2 完全オーバラップに基づく定義

完全連結法に基づき，**完全オーバラップ**を次のように定める．

[定義 8] マッチの集合 M が以下の式を満たすとき，M は**完全オーバラップ**の性質を持つ．

$$\forall m_i, m_j \in M, ts\_overlap(m_i, m_j) \qquad \square$$

完全オーバラップは，グループ内のすべてのマッチが互いにオーバラップすることを保証する．例えば図 2 の場合，m_1, m_4 が互いにオーバラップしないため，$\{m_1, m_2, m_3\}$ と $\{m_2, m_3, m_4\}$ という二つのグループが生成される．

完全オーバラップによるグループは，部分オーバラップと違い開始時刻と終了時刻のみではグループを区別できないが，グループ内で初めてマッチを受理した時刻を用いることで区別できる．先ほどの例の場合，各グループの開始時刻と終了時刻は同じであるが，$\{m_1, m_2, m_3\}$ は時刻 3 に，$\{m_2, m_3, m_4\}$ は時刻 5 に初めてマッチを受理したという点で異なる．これは，マッチを初めて受理した時刻がそのグループを構成するマッチの開始時刻と終了時刻の区間を示しているためである．上述の例であれば，$\{m_1, m_2, m_3\}$ が時刻 3 に初めてマッチを受理したことが，このグループのマッチの開始・終了時刻がそれぞれ $[1:3], [3:6]$ の区間に存在することを示している．したがって，完全オーバラップに基づくグループを以下のように定義する．

[定義 9] グループは完全オーバラップの性質を持つ極大なマッチ集合である．グループは $g = (t_s, t_f, t_e)$ のように，グループの開始時刻，マッチの初受理時刻，グループの終了時刻の組で表す． $\square$

3.3 グループの生起確率の定義

定義 5 に示した通り，マッチの確率はマッチを部分系列に含む可能世界の確率の和である．したがって，マッチの集合であるグループの確率は，各マッチを部分系列に含むすべての可能世界の和として考えるのが自然である．グループの確率の具体的な定義を以下に示す．

[定義 10] グループ g の確率を以下の式で与える．なお，$W_g = \bigcup_{m_i \in g} W_{m_i}$ であり，W_{m_i} は m_i を部分系列に含む可能世界の集合を示す．

$$P(g) = \sum_{w \in W_g} P(w) \qquad \square$$

ただし，グループの確率を求めるための計算量が大きい点に注意する．グループの時間幅を $|g| = t_e - t_s + 1$ とするとき，確率計算のための計算量，すなわち計算に必要な可能世界の総数は $O(|D|^{|g|})$ である．この数はグループの時間幅に対して指数関数的に増加するが，グループの時間幅はクリーネ閉包によるイベントの繰り返しで容易に増大するため，すべての可能世界を列挙する方法では効率が悪い．そこでこの問題に対応するために，グループ確率の効率的な計算方法を 5. で提案する．

4. グループの生成アルゴリズム

本章ではグループの生成方法について述べる．まず，部分オーバラップと完全オーバラップを使用する場合の生成方法についてそれぞれ説明し，その後閾値を用いたグループの効率的な生成について述べる．なおこれ以降，パターンの前半部分と一致する系列を**ラン**と呼ぶ．例えば，パターン $\langle \mathtt{a\ b\ c} \rangle$ に対するランは $\langle \mathtt{a_t} \rangle, \langle \mathtt{a_t}, \mathtt{b_{t+1}} \rangle$ である．

```
Input: PDS                                          ▷ 確率的データストリーム
1: G ← ∅                                            ▷ グループの候補の集合
2: for all e_t ∈ PDS do
3:     e_t を用いて既存のすべてのランを更新，破棄する
4:     r_t ← ⟨e_t⟩                                  ▷ 現在時刻 t から開始するラン
5:     for all g_i ∈ G do                           ▷ 添字は生成された順番を表す
6:         if g_i が現在時刻 t にマッチを受理した then
7:             for all g_j ∈ G (j > i) do
8:                 g_i ← g_i ∪ g_j                  ▷ オーバラップするグループを統合
9:                 G ← G \ {g_j}
10:            end for
11:            g_i ← g_i ∪ {r_t}
12:        end if
13:        if g_i がランを持たない then
14:            g_i の生起確率を計算し，g_i を出力
15:            G ← G \ {g_i}
16:        end if
17:    end for
18:    if r_t がどのグループにも追加されていない then
19:        g ← {r_t}
20:        G ← G ∪ {g}
21:    end if
22: end for
```

図 3　部分オーバラップに基づくグループの生成アルゴリズム

4.1　部分オーバラップの場合

部分オーバラップでグループを生成する際のポイントは，部分オーバラップの性質を必ず満たすラン，マッチのみをグループにまとめることである．例として図 2 の m_1, m_3, m_4 を用いて説明する．時刻 4 のイベントまで入力された場合を考えると，$m_1 = \langle \mathtt{a}_1, \mathtt{b}_2, \mathtt{c}_3 \rangle, m_3 = \langle \mathtt{a}_2, \mathtt{b}_3, \mathtt{b}_4 \rangle$ は一つのグループ g にまとめることができるが，$m_4 = \langle \mathtt{a}_4 \rangle$ は g に追加することができない．なぜなら，m_4 とオーバラップする m_3 が時刻 4 の時点ではまだ受理されておらず，将来的に破棄される可能性があるためである．したがって，時刻 5 で m_3 が受理され m_4 とのオーバラップが確定するまで，m_4 を g に加えることはできない．

図 3 に部分オーバラップによるグループの生成アルゴリズムを示す．オーバラップが確定するのはマッチを受理したときであるため，マッチを受理するたびにオーバラップが確定したものを同じグループにまとめていく．

実際に，図 1 の確率的データストリームと閾値 0.05 を用いて，パターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ のグループを生成する際の例で説明する．まず時刻 1 では，その時刻から開始するラン $\langle \mathtt{a}_1 \rangle$ を生成する（4 行目）．時刻 1 の段階ではまだグループの集合 G は空集合であるため，5〜17 行目の処理は行われない．生成したランがどのグループにも追加されていないため，グループ $g_1 = \{\langle \mathtt{a}_1 \rangle\}$ を生成し G へ追加する（18〜21 行目）．時刻 2 では，まずイベント e_2 を用いて $g_1 = \{\langle \mathtt{a}_1, \mathtt{b}_2 \rangle\}$ に更新し，新しいラン $\langle \mathtt{a}_2 \rangle$ を生成する（3，4 行目）．この時点で g_1 はランしか持たないため，6〜16 行目の処理は行われない．そして時刻 1 同様新しく生成したランがどのグループにも追加されていないため，新しいグループ $g_2 = \{\langle \mathtt{a}_2 \rangle\}$ を生成し G に加える（18〜21 行目）．時刻 3 では，ランの更新によって g_1 がマッチ $\langle \mathtt{a}_1, \mathtt{b}_2, \mathtt{c}_3 \rangle$ を受理するため，g_1 よりも後に生成されたグループ g_2 を g_1 に統合

```
Input: PDS                                          ▷ 確率的データストリーム
1: R ← ∅                                            ▷ ランの集合
2: G ← ∅                                            ▷ グループの候補の集合
3: for all e_t ∈ PDS do
4:     e_t を用いて既存のすべてのランを更新，破棄する
5:     R ← R ∪ {⟨e_t⟩}                              ▷ 新しいランの追加
6:     if R がマッチを持つ then
7:         g_copy ← R                               ▷ 複製を生成し G へ追加
8:         G ← G ∪ {g_copy}
9:         R からマッチを除く
10:    end if
11:    for all g ∈ G do
12:        if g がランを持たない then
13:            if g が出力されたグループの部分集合でない then
14:                g の生起確率を計算し，g を出力
15:            end if
16:            G ← G \ {g}
17:        end if
18:    end for
19: end for
```

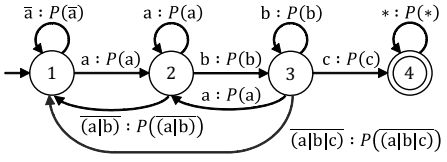
図 4　完全オーバラップに基づくグループの生成アルゴリズム

し，新しいラン $\langle \mathtt{a}_3 \rangle$ を g_1 に加える（6〜12 行目）．なお，マッチを受理したグループ g_i は，g_i の開始時刻から現在時刻 t までに生成されたすべてのラン，マッチとオーバラップする．そのため，g_i より後に生成されたグループ g_j と結合しても結合後のグループは必ず部分オーバラップの性質を満たす．時刻 4 以降も同様にランの更新とグループの結合を繰り返し，すべてのマッチが受理されランが無くなる時刻 6 にグループ $g_1 = (1, 6)$ を出力する（13〜16 行目）．

4.2　完全オーバラップの場合

完全オーバラップでグループを生成する際のポイントは，3.2 で述べたように，マッチを初めて受理した時刻によってグループが区別できるという点である．図 4 に完全オーバラップを用いてグループを生成するアルゴリズムを示す．マッチの初受理時刻によってグループの区別が行えるため，完全オーバラップの性質を満たすランをまとめつつ，マッチを受理するたびに新しいグループの候補を生成していく．

実際に，図 1 の確率的データストリームと閾値 0.05 を用いて，パターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ のグループを生成する際の例で説明する．まず，グループの候補を生成するために，ランだけの集合 $R = \{\langle \mathtt{a}_1 \rangle\}$ を作る（5 行目）．なお，すべてのランは現在時刻 t において必ずオーバラップするため，ランの集合である R は必ず完全オーバラップの性質を満たす．時刻 1 ではまだマッチ，グループの候補共に存在しないため，6 から 18 行目までは実行されない．次に時刻 2 では 4, 5 行目によって $R = \{\langle \mathtt{a}_1, \mathtt{b}_2 \rangle, \langle \mathtt{a}_2 \rangle\}$ に更新され，時刻 1 同様マッチが受理されないためそれ以降の処理は行われない．時刻 3 ではマッチ $\langle \mathtt{a}_1, \mathtt{b}_2, \mathtt{c}_3 \rangle$ が受理されるため，初受理時刻 3 のグループとして g_1 を生成しグループの候補の集合 G に加える（6〜8 行目）．なおこれ以降，g_1 は 4 行目でランの更新のみが行われる点に注意する．その後，初受理時刻が異なるグループを生成するために R からすべてのマッチを除く（9 行目）．また，時刻 4 でも新たにマッチ $\langle \mathtt{a}_1, \mathtt{b}_2, \mathtt{b}_3, \mathtt{c}_4 \rangle$ が受理され，初受理時刻 4 のグループ g_2 が生成される．以降



∗ : 任意のイベント

図 5　パターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ のトランスデューサ

も同様にランの更新とグループの生成を行い，グループがランを持たなくなる時刻 6 で $g_1 = (1, 3, 6), g_2 = (1, 4, 6)$ を出力する (12〜17 行目)．

4.3　生起確率の閾値によるマッチの枝刈り

マッチを枝刈りしない場合，確率的データストリーム中に存在するすべてのマッチをグループにまとめることになるが，現実にはこのような処理が困難な場合もある．例として，パターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ からマッチを生成する場合を考える．曖昧な確率的データストリーム，具体的にはすべてのイベントの生起確率が常に 0 より大きいような場合では，確率的データストリームが続く限り $\mathtt{b}^{+}$ による繰返しが行われるためいつまでも受理されないランが発生する．このようなランを含むグループはいつまでも出力されないだけでなく，本来含むべきでないマッチとまでオーバラップしグループに含んでしまう可能性がある．

また，そもそもグループの生成に必要でないマッチも存在する．例えば図 2 のマッチから部分オーバラップによりグループを生成すると $g = (1, 6)$ が得られるが，グループの開始・終了時刻が m_2 のものと一致しているため，m_2 の情報さえあればこのグループは生成できる．完全オーバラップを用いる場合でも，m_2 が m_3 を時間的に包含しているため，グループを生成する際には m_1, m_2, m_4 さえあれば十分である．

以上のことから，グループの生成にすべてのマッチを利用するよりも，生起確率に基いてマッチを選別し生成に利用する方が適当だと考えられる．そこで，マッチの生起確率に閾値を設け，閾値以上のマッチのみを用いたグループの生成を提案する．なおランに対しても閾値を用いた破棄を行うことで，確率の低いマッチの早期破棄による処理速度の向上も期待できる．

5.　グループ確率の効率的な計算手法

本章では，グループ確率の効率的な計算手法を提案する．まず本手法で用いるトランスデューサとその生成方法について説明し，その後部分オーバラップと完全オーバラップの場合についてそれぞれ述べる．

5.1　トランスデューサの利用と生成

3.3 で述べたように，可能世界をすべて列挙してグループ確率を計算する方法は効率が悪い．そこで，**トランスデューサ**を用いた効率的な計算手法を提案する．トランスデューサは入力を受理するか確認するのと同時に出力を求めるオートマトンである．例えば，図 5 はパターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ で生成したグループの確率を計算するためのトランスデューサである．このトランスデューサはマッチを部分系列に持つすべての可能世界を受理するよう作成されており，同時に各可能世界の確率を計算する．このトランスデューサを用いることで，マッチを部分系列に含

Input: パターン p
Output: トランスデューサの状態と遷移の集合
1: 状態 $s[1], s[2]$ を生成し，遷移 $(s[1], s[2], p_1), (s[1], s[1], \overline{p_1})$ を追加
2: $i \leftarrow 2, j \leftarrow 1$　▷ 照合位置を初期化
3: **while** $i \leqq |p|$ **do**
4:　状態 $s[i+1]$ を生成し，遷移 $(s[i], s[i+1], p_i), (s[i+1], s[i+1], p_i)$ を追加
5:　**if** $p_i = p_j$ **then**　▷ p_i がパターン前半の繰返しである場合
6:　$j \leftarrow j + 1$　▷ 照合箇所を進める
7:　**else**　▷ p_i がパターン前半の繰返しでない場合
8:　遷移 $(s[i], s[j+1], p_j)$ を追加　▷ 繰り返しの箇所まで戻る
9:　**if** $p_i = p_1$ **then** $j \leftarrow 2$ **else** $j \leftarrow 1$ **end if**
10:　**end if**
11:　**if** $s[i]$ が p_1 による遷移を持たない **then**
12:　遷移 $(s[i], s[2], p_1)$ を追加
13:　**end if**
14:　$s[i]$ の既存の遷移に一致しないとき $s[1]$ へ移る遷移を追加
15:　$i \leftarrow i + 1$
16: **end while**
17: 受理状態の遷移をすべて削除し，任意のイベントで受理状態自身に移る遷移を追加

図 6　トランスデューサの生成アルゴリズム

む可能世界の確率の総和を計算するという問題を，トランスデューサの受理状態に到達する確率を求めるという問題に変換して考えられる．

図 6 にトランスデューサの生成アルゴリズムを示す．なお，トランスデューサの状態を配列 s で，状態 $s[i]$ から $s[j]$ にパターンの k 番目のイベント p_k で遷移することを $(s[i], s[j], p_k)$ の組で，パターンを構成するイベントの数を $|p|$ で表す．トランスデューサの生成では，$\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{a}^{+}\mathtt{b}^{+}\mathtt{c} \rangle$ における $\langle \mathtt{a}\ \mathtt{b} \rangle$ のように，パターン内部での繰り返しに注意する必要がある．このようなパターンでは $\langle \mathtt{a}, \mathtt{b}, \mathtt{a}, \mathtt{b}, \mathtt{a} \rangle$ のように単純に破棄してはならない可能世界が存在するためである．そこで，クヌース・モーリス・プラット法の失敗関数 [7] を参考に，パターン自身に対して照合を行うことでパターンの繰り返しを検出し生成に利用する．このアルゴリズムにおいて，j はパターンの繰り返しとなるイベントがいくつ存在するかを表す．つまり，パターンの繰り返しに応じて j を加算し，繰り返しが終了した時点での j の値を利用して状態の適当な遷移先を決定する．

図 6 の手続きで生成したトランスデューサを用いてグループの確率を計算する．以下では，具体的な計算方法を部分オーバラップと完全オーバラップの場合に分け順に説明する．

5.2　部分オーバラップの場合

3.1 で述べたように，部分オーバラップによって生成されたグループ $g = (t_s, t_e)$ は，開始時刻 t_s から終了時刻 t_e の間にあるすべてのマッチをひとまとめにしている．つまり，時区間 $[t_s : t_e]$ の間でマッチを部分系列として含む可能世界の確率の総和がグループの確率となる．したがって，グループ確率 $P(g)$ は時区間 $[t_s : t_e]$ において，上述したトランスデューサの受理状態に到達する確率である．

トランスデューサの受理状態に到達する確率は，遷移行列を用いて計算する．例えば，次の式は図 5 のトランスデューサに対応する遷移行列である．

時刻	0	1	2	3	4	5	6
$V_t[1]$	1.0	0	0	0.03	0.05	0.06	0.06
$V_t[2]$	0	1.0	0.3	0.10	0.09	0	0
$V_t[3]$	0	0	0.7	0.80	0.63	0.65	0
$V_t[4]$	0	0	0	0.07	0.23	0.29	0.94

図 7　ベクトル V_t の更新（小数点第二位まで表示）

$$T_{t-1,t} = \begin{bmatrix} P(\overline{\mathtt{a}_t}) & P(\mathtt{a}_t) & 0 & 0 \\ P\left(\overline{\mathtt{a}|\mathtt{b}_t}\right) & P(\mathtt{a}_t) & P(\mathtt{b}_t) & 0 \\ P\left(\overline{\mathtt{a}|\mathtt{b}|\mathtt{c}_t}\right) & P(\mathtt{a}_t) & P(\mathtt{b}_t) & P(\mathtt{c}_t) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

各行が時刻 $t-1$ におけるトランスデューサの状態を，各列が時刻 t で遷移した先の状態を表す．時刻 t にトランスデューサの各状態に到達する確率をベクトル V_t で表すとき，V_t の確率は 1 単位時刻前の確率 V_{t-1} と遷移行列 $T_{t-1,t}$ を用いて次の式で計算できる．

$$V_t = V_{t-1} \times T_{t-1,t}$$

したがって，時区間 $[t_s : t_e]$ において最終的にトランスデューサの各状態に到達する確率 V_{t_e} は次の式で求められる．ただし，V_{t_s-1} は $V_{t_s-1}[0] = 1$，$V_{t_s-1}[i] = 0\ (i > 0)$ で初期化する．

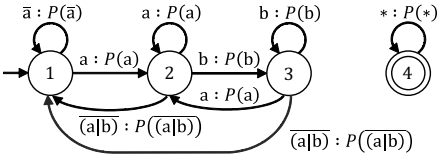
$$V_{t_e} = V_{t_s-1} \times \prod_{t=t_s}^{t_e} T_{t-1,t}$$

グループ確率はこのベクトル V_{t_e} のうち，受理状態にあたる要素の値となる．実際に，図 1 の確率的データストリームに対してパターン $\langle \mathtt{a}\ \mathtt{b}^{+}\mathtt{c} \rangle$ で生成したグループ $g = (1, 6)$ の確率を計算する様子を図 7 に示す．このグループの終了時刻は 6 であるため，グループ確率 $P(g)$ は約 0.94 である．

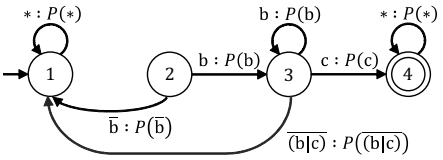
5.3　完全オーバラップの場合

完全オーバラップの場合，部分オーバラップと同じ方法ではグループ確率を正確に計算できない．3.2 で述べたように，完全オーバラップでは開始時刻と終了時刻が同じでもグループを構成するマッチが異なる場合がある．部分オーバラップと同じ方法を用いるとこのようなグループを構成するマッチの違いを汲み取れず，誤ったグループ確率を計算してしまう．そこで，トランスデューサを追加することで正確なグループ確率を求める方法を提案する．完全オーバラップにおいて誤った確率を計算してしまう理由は，トランスデューサがそのグループに含まれていないマッチも受理してしまうことである．3.2 で述べた通り，グループ $g = (t_s, t_f, t_e)$ を構成するマッチは開始時刻が $[t_s : t_f]$，終了時刻が $[t_f : t_e]$ の間にある．つまり，$[t_s : t_f)$ 及び $(t_f : t_e]$ の区間に存在するマッチを受理しないトランスデューサを追加で用いれば，正確な確率が計算できる．

実際に，図 5 のトランスデューサに加えて使用する二つのトランスデューサを図 8 に示す．(a) はグループの開始時刻 t_s から初受理時刻 t_f の直前まで使用するトランスデューサで，図 5 のトランスデューサから受理状態に移る遷移を除くことで作成できる．受理状態に移る遷移を削除したことで，時区間 $[t_s : t_f)$ のマッチを受理しないようになっている．一方，(b) は初受理時刻 t_f の直後から終了時刻 t_e まで使用するトランスデューサであり，図 5 のトランスデューサから状態 2 に移る遷移を除くことで作成できる．状態 2 に移る遷移を削除することで，時区間 $(t_f : t_e]$ のマッチを生成しないようになっている．



(a) マッチ受理前に使用するトランスデューサ



(b) マッチ受理後に使用するトランスデューサ

図 8　追加で使用するトランスデューサ

表 2　実 験 環 境

OS	Windows 7 Professional 64 bit
CPU	Intel Core i7-2600 3.40 GHz
メモリ	4.0 GB
Java 仮想マシン（JVM）	JVM 1.5
JVM へのメモリ割当て	1.5GB

基本的な確率の計算方法は部分オーバラップの場合と同じであり，トランスデューサの遷移行列を利用する．ただし，上述の通りグループの時刻に応じて使用するトランスデューサを変更する．図 5 のトランスデューサの遷移行列を T，図 8 の (a) と (b) の遷移行列をそれぞれ T^a, T^b とする．このとき，グループ $g = (t_s, t_f, t_e)$ の確率ベクトル V_{t_e} は以下の式で計算できる．なお，V_{t_s-1} は部分オーバラップの場合と同様に初期化する．

$$V_{t_e} = V_{t_s-1} \times \prod_{t=t_s}^{t_f-1} T^a_{t-1,t} \times T_{t_f-1,t_f} \times \prod_{t=t_f+1}^{t_e} T^b_{t-1,t}$$

実際に，図 2 のマッチを完全オーバラップでまとめた際に得られるグループ $g_1 = (1, 3, 6), g_2 = (1, 5, 6)$ について考えると，g_1, g_2 の確率ベクトル V_{g_1}, V_{g_2} はそれぞれ次の式で求められる．

$$V_{g_1} = V_0 \times T^a_{0,1} \times T^a_{1,2} \times T_{2,3} \times T^b_{3,4} \times T^b_{4,5} \times T^b_{5,6}$$
$$V_{g_2} = V_0 \times T^a_{0,1} \times T^a_{1,2} \times T^a_{2,3} \times T^a_{3,4} \times T_{4,5} \times T^b_{5,6}$$

6.　評 価 実 験

本章では実験により提案手法の有効性を評価する．実験で使用するシステムは，SASE プロジェクトで作成されたシステム SASE+ [1] を拡張することで実装した．なお，すべての測定値は表 2 に示す構成のコンピュータ上で計測した．

まず，実験で使用したデータセットについて説明する．実験は実データと人工データの両方を用いて行った．実データには，Lahar プロジェクトで公開されている屋内位置の確率的データストリームを使用した．このデータは屋内構造をグラフで表しており，被験者が各ノードにいる確率と実際にその被験者がい

表 3　実験で使用するパラメータ

パラメータ	値
o：オーバラップのタイプ	$comp$：完全オーバラップ $sing$：部分オーバラップ
c：グループ確率の計算方法	$proposed$：提案手法 $naïve$：素朴な手法
$\|g\|$：グループの時間幅	$\{3,4,5,6,7,8,9,10\}$
θ：マッチ確率の閾値	$\{0.001, 0.005, 0.01, 0.02, ..., 0.1\}$

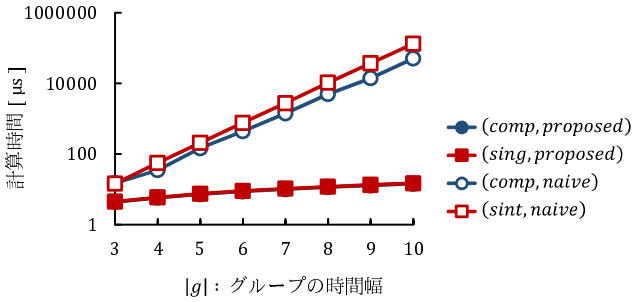


図 9　グループの時間幅に対するグループ確率の計算時間

た正解ノードの情報を一秒毎に持つ．被験者はある部屋への入退出と廊下の移動を繰り返し行うよう指定されており，合計 9 回部屋への入退出を行っている．そのため，実験ではこの部屋の入退出を $\langle \texttt{Door Room}^+ \texttt{ Door} \rangle$ として表し，入力パターンとした．また，人工データには以下の手順で生成したイベント数 100,000 の確率的データストリームを使用する．

（1） 非確率的データストリーム $\langle \alpha_1, ...\alpha_{100000} \rangle$ を生成 $(\alpha_t \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\})$.

（2） 各時刻 t で生起しなかったイベント $(\forall \alpha'_t \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\} \backslash \{\alpha_t\})$ に対して，$[0, 0.1]$ の範囲でランダムに生起確率を付与．

（3） 残りの生起確率 $\left(1 - \sum_{\alpha'_t \in \{\texttt{a},\texttt{b},\texttt{c},\texttt{d}\} \backslash \{\alpha_t\}} P(\alpha'_t)\right)$ を α_t の生起確率とする．

なお手順 1 の段階で，入力として与えるパターン $\langle \texttt{a b}^+\texttt{c} \rangle$ に対応するマッチが生起するようストリームを生成している．

次に，実験で使用するパラメータを表 3 に示す．グループ確率の計算方法において，提案手法は 5. で述べたトランスデューサを用いた手法を，素朴な手法は 3.3 で述べた可能世界を列挙する方法を用いる．以降，使用するパラメータのうちオーバラップの種類とグループ確率の計算方法を組 (o, c) で表す．例えば，完全オーバラップと提案手法を使用する場合は $(comp, proposed)$ で表す．なお，マッチ確率の閾値のうち 0.02 から 0.09 の間は 0.01 ずつ値を増加させている．

6.1　処理速度の評価

各パラメータを変化させた際の処理速度を評価する．なお，本節の実験では入力に人工データとパターン $\langle \texttt{a b}^+\texttt{c} \rangle$ を使用する．まず，グループの時間幅に対するグループ確率の計算時間を図 9 に示す．この結果は確率の閾値として $\theta = 0.01$ を使用したものであるが，他の値を使用した場合でも同じ傾向が得られる．結果から，提案手法の計算時間が線形に増加しているのに対し，素朴な手法は指数関数的に増加していることがわかる．これは，3.3 で述べた通り，素朴な手法ではグループ確率の計算に必要な可能世界の個数がグループの時間幅に対して指数関数的に増加するためである．一方提案手法では，グループの時間幅の大きさは行列の乗算を行う回数となるため，計算時間の増加は線形で済む．

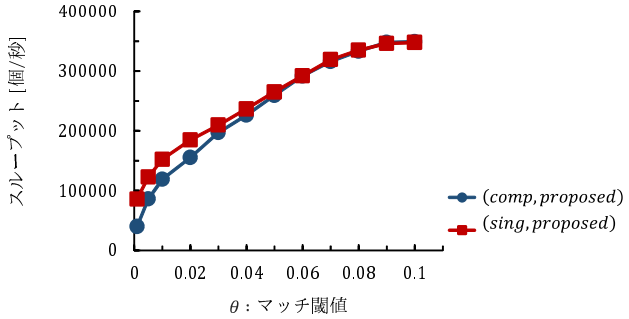


図 10　マッチ閾値に対するスループット

表 4　部分オーバラップによるグループの生成結果

正解マッチ	正解マッチを含むグループ	グループの生起確率	マッチの最大生起確率
$(26, 60)$	$(25, 72)$	0.072	0.002
$(114, 130)$	$(98, 145)$	0.955	0.073
$(197, 214)$	$(178, 234)$	0.093	0.003
$(286, 326)$	$(279, 358)$	0.999	0.077
$(406, 418)$	$(403, 431)$	0.177	0.015
$(484, 501)$	$(476, 515)$	0.996	0.168
$(639, 660)$	$(638, 672)$	0.325	0.032
$(715, 727)$	$(701, 745)$	0.125	0.008
$(773, 783)$	$(766, 795)$	0.527	0.037

次に，マッチ確率の閾値を変化させた際のスループット (一秒あたりに処理可能なイベント数) を図 10 に示す．なお，素朴な手法ではほとんどイベントを処理できないため結果から省略している．実験結果から，マッチ閾値を大きくするにつれスループットも増加することがわかる．これは，マッチ閾値の増加による確率の小さいランやマッチの枝刈りが原因である．生成されるランやマッチの数が減った場合，グループにランを追加する処理や生成されるグループの数が少なくなるため，このように処理が高速化される．ただし，大き過ぎるマッチ閾値を設定した場合グループを生成するためのマッチが無くなることも考えられるため，入力となる確率的データストリームを考慮したうえで適当なマッチ閾値を選ぶ必要がある．

6.2　処理結果の評価

提案手法によって生成されたグループを評価する．この実験では入力に実データとパターン $\langle \texttt{Door Room}^+ \texttt{ Door} \rangle$ を，マッチ確率の閾値として 0.0001 を用いる．また，グループの生成には部分オーバラップを使用した．

表 4 に実験結果を示す．なお，(t_s, t_e) はマッチないしグループの開始・終了時刻の組を，マッチの最大確率はグループに含まれるマッチのうち最も生起確率の大きいマッチの確率を示す．また，時刻は計測開始時刻を 1 として表す．結果から，マッチをグループにまとめることで生起確率が大幅に増加していることがわかる．マッチ確率は各イベントの生起確率の乗算で求められるため，マッチはイベント数つまり時区間の幅が大きくな

るほど生起確率が小さくなる．一方グループは，時区間の幅が大きくなるほど含まれるマッチの数も多くなるため，正規確率も大幅に増加する．

また，結果から得られたグループの生起確率の差が大きいこともわかる．これは入力データの持つ曖昧性が原因である．例えば，一行目のグループの生起確率は 0.072 と最も小さいが，入力の時点で Door ∨ Room の確率は 0.4 程度しかない．これは，測定に利用した RFID リーダがすべて廊下に設置されていることで，被験者が部屋の中に入ると測定誤差が大きくなるためである．二行目のグループのように生起確率が大きい場合，入力時点で Door ∨ Room の確率が 0.9 ほどある正確なデータが入力されている．つまり，グループはマッチをまとめることで生起確率を高められるが，その結果は元々の曖昧さに大きく影響される．

7. 関 連 研 究

入力となるデータに曖昧性が存在しない非確率的データストリームに対してパターン照合を行う研究は数多く提案されている [1–3, 5, 6, 11–15, 17–20]．本稿の実験で拡張した SASE+ も，非確率的データストリームに対するパターン照合を扱う SASE プロジェクト [1, 6, 18–20] で作成された CEP システムである．ただし，SASE プロジェクトを含め，これらの手法は確率的データストリームには対応していない．

確率的データストリームでのパターン照合を扱った研究はいくつか存在する [8–10, 16]．Lahar プロジェクト [8, 9, 16] は，異なる時刻に生起したイベントの生起確率の相関に注目し，各イベントがマルコフ過程に基づいた条件付き確率を持つ確率的データストリームを扱っている．Lahar プロジェクトではマッチの確率計算にオートマトンを用いており，暗黙的にマッチの統合を行っている．しかし，それらは本稿で提案した完全オーバラップにより生成されるグループの一部として考えられるため，マッチの統合に関して提案手法は Lahar プロジェクトを包含している．

[10] では，確率的データストリームから生起確率が上位 k 個のマッチを効率的に発見する手法を提案している．ただし，確率的データストリームが入力として与えられるのではなく，システムが持つエラーモデルを用いて決定的データストリームを確率的データストリームに変換し処理を行っている．また，発見したマッチの確率を統合する処理もオプションとして述べているが，同時刻に発見されたマッチをまとめるだけであり，本稿で提案したグループほど柔軟性の高い処理はできない．

8. お わ り に

本稿では，確率的データストリームでのパターン照合におけるマッチのグループ化手法について提案した．マッチをグループにまとめるための指針として完全オーバラップと部分オーバラップを定義し，これらを用いたグループの定義を提案した．また，各オーバラップを使用した場合のグループの生成アルゴリズムと，トランスデューサを用いることで効率良くグループの確率を計算する手法を提案した．加えて，人工データと実データによる評価実験を行い，本手法の有効性を確認した．今後の課題としては，グループにまとめるための新しい指針の考案や，異なる時刻のイベントの生起確率が相関を持つ確率的データストリームへの拡張，より詳細な問合せ条件の記述ができる言語の考案などが挙げられる．

文　　献

[1] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *Proc. ACM SIGMOD*, pages 147–160, 2008.

[2] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1(1):66–77, 2008.

[3] B. Chandramouli, J. Goldstein, and D. Maier. High-performance dynamic pattern matching over disordered streams. *Proc. VLDB Endow.*, 3(1-2):220–231, 2010.

[4] N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[5] A. Demers, J. Gehrke, and B. P. Cayuga: A general purpose event monitoring system. In *Proc. CIDR*, pages 412–422, 2007.

[6] D. Gyllstrom, J. Agrawal, Y. Diao, and N. Immerman. On supporting Kleene closure over event streams. In *Proc. ICDE*, pages 1391–1393, 2008.

[7] D. Knuth, J. Morris, Jr., and V. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[8] J. Letchner, C. R, M. Balazinska, and M. Philipose. Access methods for Markovian streams. In *Proc. ICDE*, pages 246–257, 2009.

[9] J. Letchner, C. R, M. Balazinska, and M. Philipose. Approximation trade-offs in Markovian stream processing: An empirical study. In *Proc. ICDE*, pages 936–939, 2010.

[10] Z. Li, T. Ge, and C. X. Chen. ε-matching: Event processing over noisy sequences in real time. In *Proc. ACM SIGMOD*, pages 601–612, 2013.

[11] M. Liu, D. Golovnya, E. A. Rundensteiner, and K. T. Claypool. Sequence pattern query processing over out-of-order event streams. In *Proc. ICDE*, pages 784–795, 2009.

[12] A. Majumder, R. Rastogi, and S. Vanama. Scalable regular expression matching on data streams. In *Proc. ACM SIGMOD*, pages 161–172, 2008.

[13] Y. Mei and S. Madden. ZStream: A cost-based query processor for adaptively detecting composite events. In *Proc. ACM SIGMOD*, pages 193–206, 2009.

[14] B. Mozafari, K. Zeng, and C. Zaniolo. High-performance complex event processing over XML streams. In *Proc. ACM SIGMOD*, pages 253–264, 2012.

[15] Y. Qi, L. Cao, M. Ray, and E. A. Rundensteiner. Complex event analytics: Online aggregation of stream sequence patterns. In *Proc. ACM SIGMOD*, pages 229–240, 2014.

[16] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu. Event queries on correlated probabilistic streams. In *Proc. ACM SIGMOD*, pages 715–728, 2008.

[17] L. Woods, J. Teubner, and G. Alonso. Complex event detection at wire speed with FPGAs. *Proc. VLDB Endow.*, 3(1-2):660–669, 2010.

[18] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proc. ACM SIGMOD*, pages 407–418, 2006.

[19] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. *Proc. VLDB Endow.*, 3(1-2):244–255, 2010.

[20] H. Zhang, Y. Diao, and N. Immerman. On complexity and optimization of expensive queries in complex event processing. In *Proc. ACM SIGMOD*, pages 217–228, 2014.