Mining Frequent Elliptic Episodes from Event Sequences

Takashi Katoh[1] and Kouichi Hirata[2]

[1] Graduate School of Computer Science and Systems Engineering
[2] Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
{f673024t,hirata}@ai.kyutech.ac.jp
Tel: +81-948-29-7622, Fax: +81-948-29-7601

Abstract. In this paper, we formulate an episode in episode mining as an *acyclic transitive labeled digraph*. Also we introduce an *elliptic* episode $s \mapsto [S_1, \dots, S_n] \mapsto t$ such that s and t are event types, $[S_1, \dots, S_n]$ is a sequence of serial episodes, and every pair of S_i and S_j shares no same event type. Then, we show that every elliptic episode is *constructible from serial episodes*. Next, in this paper, we design the algorithm FreqEllip to extract all of the *frequent elliptic episodes* from a given event sequence.

1 Introduction

It is one of the important tasks for data mining to discover frequent patterns from time-related data. Mannila *et al.* [6] have introduced a method for such a task called *episode mining* to discover frequent *episodes* as patterns in an *event sequence*. By combining a *parallel episode* as a set of event types and a *serial episode* as a sequence of event types, the episodes are formulated as *directed acyclic graphs* of events of which edges specify the temporal precedent-subsequent relation. The episode mining has been developed by introducing the specific form of episodes for every target area together with efficient algorithms [2–5].

In order to focus on the direct *causality* in episode mining, Katoh *et al.* have introduced a *sectorial episode* [3, 5] of the form $E \mapsto e$ and a *diamond episode* [4] of the form $f \mapsto E \mapsto e$, where e and f are event types and E is a parallel episode, that is, a set of event types. Intuitively, a sectorial episode $E \mapsto e$ means that every event in E is followed by e, and a diamond episode $f \mapsto E \mapsto e$ means that every event in E is followed by e and follows f.

Note that, in the previous works [3–5], we have designed the algorithms to extract frequent sectorial (*resp.*, diamond) episodes from the information of occurrences of serial episodes with length 2 (*resp.*, 3) in an event sequence. In both cases, it is not necessary to consider the duplicated occurrences of the same event type in E, because E is a parallel episode. On the other hand, when we extend the form of episodes with allowing the duplicated occurrences of event types and give a method to construct such episodes from the information of occurrences of serial episodes, it is necessary to formulate an episode and an event sequence possible to deal with the duplicated occurrences of event types in them carefully.

Then, in this paper, we deal with an episode and an event sequence as an *acyclic transitive labeled digraph* (*ATL-digraph*, for short). Also we say that an ATL-digraph $D = (V, A)$ is *parallel-free* if, for every pair $(u, v) \in V \times V$ of vertices such that $u \neq v$ but u and v have the same label, it holds that either $(u, v) \in A$ or $(v, u) \in A$. Hence, we formulate an episode and an event sequence as a parallel-free ATL-digraph of which label is an event type.

Under the above formulation, we introduce a *constructible episode from serial episodes*, that is, from the information of occurrences of serial episodes in an event sequence. An episode D is *constructible from serial episodes* if every parallel-free ATL-digraph containing all of the serial episodes in D always embeds D. Also, in this paper, we newly introduce an *elliptic episode*, as the extension of a diamond episode [4], of the form $s \mapsto [S_1, \dots, S_n] \mapsto t$, where s and t are event types, $[S_1, \dots, S_n]$ is a sequence of serial episodes and every pair of S_i and S_j shares no same event type. The last requirement guarantees that an elliptic episode is parallel-free. Then, we show that every elliptic episode is constructible from serial episodes.

Next, we design the algorithm FreqEllip to extract all of the *frequent elliptic episodes* from an event sequence, by extending the algorithm FreqDmd [4]. The algorithm FreqEllip first calls the algorithm NewBitSerl, which is an extension of the algorithm BitSerl [4]. While the algorithm BitSerl constructs the set of bit vectors representing the occurrences of all serial episodes with length 3, the algorithm NewBitSerl constructs the set of bit vectors representing the occurrences of all serial episodes within given length. Then, the algorithm FreqEllip extracts the set of all frequent elliptic episodes of the form $s \mapsto [S_1, \ldots, S_n] \mapsto t$, by regarding a serial episode $s_1 \mapsto [S_i] \mapsto s_2$ as an item S_i and by calling the modified algorithm of AprioriTid [1] to preserve that an elliptic episode is parallel-free.

Finally, in order to extract frequent elliptic episodes concerned with *changes for drug resistance*, we apply the algorithm FreqEllip to bacterial culture data. By fixing the detected bacterium and the sample, we extract frequent elliptic episodes of the form $s \mapsto [S_1, \ldots, S_n] \mapsto t$ such that s is of the form $\mathtt{Ant}_0\mathtt{=S}$ (S means "susceptibility"), t is of the form $\mathtt{Ant}_0\mathtt{=R}$ (R means "resistant"), and every S_i is a serial episode of the form $(\mathtt{Ant}_i\mathtt{=S})(\mathtt{Ant}_i\mathtt{=R})$, where $\mathtt{Ant}_i$ $(0 \leq i \leq n)$ is a mutually distinct antibiotic.

2 Episodes as Acyclic Transitive Labeled Digraphs

As similar as [6], we assume that an event has an associated time of occurrence as a natural number. Formally, let $\mathcal{E}$ be a set of *event types*. Then, a pair (e, t) is called an *event*, where $e \in \mathcal{E}$ and t is a natural number which is the (*occurrence*) *time* of the event. In the following, for a set $E \subseteq \mathcal{E}$ of event types, we denote $\{(e, t) \mid e \in \mathcal{E}\}$ by (E, t), and also call it by an *event* again. Furthermore, we denote a set $\{e_1, \ldots, e_m\} \subseteq \mathcal{E}$ of event types by a string $e_1 \cdots e_m$.

An *event sequence* $\mathcal{S}$ on $\mathcal{E}$ is a triple $\langle S, T_s, T_e \rangle$, where $S = \langle (E_1, t_1), \ldots, (E_n, t_n) \rangle$ is an ordered sequence of events satisfying the following conditions.

$$E_i \subseteq \mathcal{E}\ (1 \leq i \leq n),\ t_i < t_{i+1}\ (1 \leq i \leq n-1), \text{ and } T_s \leq t_i < T_e\ (1 \leq i \leq n).$$

In particular, T_s and T_e are called the *starting* time and the *ending* time of $\mathcal{S}$. We denote $T_e - T_s$ by $l_{\mathcal{S}}$.

A *window* in an event sequence $\mathcal{S} = (S, T_s, T_e)$ is an event sequence $W = (w, t_s, t_e)$ such that $t_s < T_e$, $t_e > T_s$ and w consists of all of the events (e, t) in S where $t_s \leq t < t_e$. The time span $t_e - t_s$ is called the *width* of the window. We call a window with width k in $\mathcal{S}$ a *k-window* and denote the k-window $(w, t, t+k)$ of $\mathcal{S}$ starting from t by $w(\mathcal{S}, t, k)$.

Mannila *et al.* [6] have formulated an episode as an acyclic labeled digraph. On the other hand, in this paper, we formulate an episode as an acyclic *transitive* labeled digraph. Note that the transitivity will have the important property for the construction of episodes. Then, we prepare some notions for digraphs necessary for discussion bellow.

A *digraph* (or a *directed graph*) $D = (V, A)$ consists of a finite, nonempty set V of *vertices* and a (possibly empty) set A of ordered pairs of distinct vertices. We sometimes denote V by $V(D)$ and A by $A(D)$. An element of A is called an *arc*. We denote $|V|$ by $|D|$.

For two digraphs $D_1 = (V_1, A_1)$ and $D_2 = (V_2, A_2)$, we denote a digraph $(V_1 \cup V_2, A_1 \cup A_2)$ by $D_1 \cup D_2$. For a digraph $D = (V, A)$ and $U \subseteq V$, we denote a digraph $(V - U, A - \{(v, u) \in A \mid v \in U \text{ or } u \in U\})$ by $D - U$.

For an arc $(u, v) \in A$, u is said to be *adjacent to* v and v is *adjacent from* u. For a digraph $D = (V, A)$ and a vertex $v \in V$, the *outdegree* of v in D, denoted by $od_D(v)$, is the number of vertices adjacent from v in D and the *indegree* of v in D, denoted by $id_D(v)$, is the number of vertices adjacent to v in D.

Let D be a digraph (V, A). Then, a *walk* in D is an alternating sequence $W = v_0 a_1 v_1 \cdots a_n v_n$ of vertices and arcs, beginning and ending with vertices, such that $a_i = (v_{i-1}, v_i)$ for $1 \leq i \leq n$, and refer to W as a v_0-v_n walk. Also a *path* in D is a walk in which no vertex is repeated. For vertices u and v in V, u is *accessible to* v (in D) if there exists a u-v walk in D. A digraph D is *acyclic* if there exists no v-v walk in D. Also a digraph D is *transitive* if, for every $u, v, w \in V$, it holds that $(u, w) \in A$ whenever $(u, v) \in A$ and $(v, w) \in A$. Furthermore, for a set L of labels, a digraph D is *labeled* (by L) if every vertex $v \in V$ has a label $l(v) \in L$. We call an acyclic transitive labeled digraph an *ATL-digraph*.

A digraph $D_1 = (V_1, A_1)$ is *embedded into* a digraph $D_2 = (V_2, A_2)$ *as labeled digraphs* (or D_2 *embeds* D_1), denoted by $D_1 \sqsubseteq D_2$, if there exists an injection from V_1 to V_2 such that $(\varphi(u), \varphi(v)) \in A_2$ whenever $(u, v) \in A_1$, and $l(v) = l(\varphi(v))$ for every $v \in V_1$.

In this paper, we formulate an *episode* as an ATL-digraph of which label is an event type. Mannila *et al.* [6] have introduced a *serial episode* as a sequence of event types. In this paper, we formulate a serial episode as an ATL-digraph $S = (\{v_1, \ldots, v_n\}, \{(v_i, v_j) \mid 1 \leq i < j \leq n\})$ such that $l(v_i) = a_i$. We sometimes identify a serial episode S with the label sequence $a_1 \cdots a_n$ of S. For an episode D, we denote the set of all serial episodes embedded into D by $se(D)$.

Furthermore, as same as episodes, we also formulate an event sequence $\mathcal{S}$ as an ATL-digraph $D(\mathcal{S}) = (V, A)$ satisfying the following conditions.

1. For every event $(e, t) \in \mathcal{S}$, there exists a vertex $v_{e,t} \in V$ such that $l(v_{e,t}) = e$.
2. For every pair $((e, t), (e', t')) \in \mathcal{S} \times \mathcal{S}$ of events, $(v_{e,t}, v_{e',t'}) \in A$ iff $t < t'$.

Hence, we say that an episode D *occurs* in an event sequence $\mathcal{S}$ if $D \sqsubseteq D(\mathcal{S})$.

Let $\mathcal{S}$ be an event sequence and k a natural number. Then, we denote the set of all k-windows by $W(\mathcal{S}, k)$. Also, for an episode D, we denote the set of all k-windows such that D occurs in $\mathcal{S}$ by $W(D, \mathcal{S}, k)$. Furthermore, the *frequency* $freq_{\mathcal{S},k}(D)$ and the *support* $supp_{\mathcal{S},k}(D)$ of D in $\mathcal{S}$ w.r.t. k are defined as follows.

$$freq_{\mathcal{S},k}(D) = |W(D, \mathcal{S}, k)|, \qquad supp_{\mathcal{S},k}(D) = \frac{freq_{\mathcal{S},k}(D)}{|W(\mathcal{S}, k)|}.$$

3 Elliptic Episodes Constructible from Serial Episodes

In this section, we formulate that an episode is *constructible from serial episodes*, that is, from the information of occurrences of serial episodes in a window, and introduce an *elliptic* episode as an extension of a diamond episode [4]. Then, we show that every elliptic episode is constructible from serial episodes.

An ATL-digraph $D = (V, A)$ is *parallel-free* if, for every pair $(u, v) \in V \times V$ of vertices such that $u \neq v$ and $l(u) = l(v)$, it holds that either $(u, v) \in A$ or $(v, u) \in A$.

Definition 1. An episode D is *constructible from serial episodes* if it holds that $D \sqsubseteq W$ for every parallel-free ATL-digraph W such that $se(D) \subseteq se(W)$.

Definition 1 requires that, for an episode D and an ATL-digraph W, every serial episode in W is corresponding to exactly one serial episode in D. Hence, by regarding W as a window, Definition 1 claims that a window W contains the information of occurrences of serial episodes in D.

Example 1. Let D be an episode and W an ATL-digraph described as Figure 1, where the transitive arcs are omitted. Then, it holds that $se(D) = se(W) = \{a, b, c, ab, bb, bc, abc, bbc, abbc\}$, and also that W is parallel-free. However, it is obvious that $D \not\sqsubseteq W$. Hence, D is *not* constructible from serial episodes.

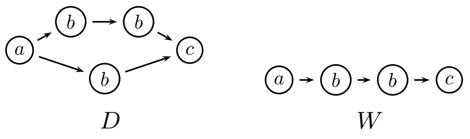

Fig. 1. An episode D and an ATL-digraph W in Example 1.

Definition 2. For $1 \leq i \leq n$, let S_i be a serial episode such that $S_1 \cup \cdots \cup S_n$ is parallel-free, and suppose that $V = V(S_1 \cup \cdots \cup S_n)$ and $A = A(S_1 \cup \cdots \cup S_n)$. Also let v and u be new vertices not in V such that $l(v) = s$ and $l(u) = t$. Then, the following ATL-digraph D is called an *elliptic episode*.

$$D = (V \cup \{v, u\}, A \cup \{(v, w), (w, u) \mid w \in V\}).$$

We call event types s and t *source* and *sink*, respectively. We denote such an elliptic episode by $s \mapsto [S_1, \ldots, S_n] \mapsto t$. The *length* of an elliptic episode $D = s \mapsto [S_1, \ldots, S_n] \mapsto t$ is defined as $\max\{|S_i| \mid 1 \leq i \leq n\} + 2$ and denoted by $len(D)$.

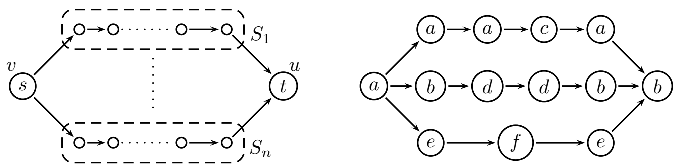

Fig. 2. An intuitive figure of an elliptic episode (left) and an elliptic episode $a \mapsto [aaca, bddb, efe] \mapsto b$ (right).

Figure 2 (left) describes an intuitive figure of an elliptic episode $s \mapsto [S_1, \ldots, S_n] \mapsto t$. Also Figure 2 (right) describes the elliptic episode $a \mapsto [aaca, bddb, efe] \mapsto b$. Here, we denote $l(v) = e$ by an vertex ⓔ in Figure 2. Also, while their digraphs are transitive, their transitive arcs are omitted in Figure 2.

Lemma 1 (Anti-monotonicity of elliptic episodes). *Let S_i be serial episodes such that $S_1 \cup \cdots \cup S_{n-1} \cup S_n$ is parallel-free. If $s \mapsto [S_1, \ldots, S_{n-1}, S_n] \mapsto t$ is frequent, then so is $s \mapsto [S_1, \ldots, S_{n-1}] \mapsto t$.*

Proof. It is obvious by the definition of elliptic episodes and their frequency. □

Theorem 1. *Every elliptic episode is constructible from serial episodes.*

Proof. We show the statement by induction on n for an elliptic episode $D = s \mapsto [S_1, \ldots, S_n] \mapsto t$.

If $n = 1$, then D is of the form $s \mapsto [S_1] \mapsto t$, which is just a serial episode. Then, for every parallel-free ATL-digraph W such that $se(D) \subseteq se(W)$, it holds that $D \sqsubseteq W$.

Suppose that the statement holds for $D_1 = s \mapsto [S_1, \ldots, S_{n-1}] \mapsto t$, and consider the elliptic episode $D = s \mapsto [S_1, \ldots, S_n] \mapsto t$.

Since D is an elliptic episode, D is parallel-free, so no label in S_n occurs in $S_1 \cup \cdots \cup S_{n-1}$. By induction hypothesis, for every W_1 such that $se(D_1) \subseteq se(W_1)$, it holds that $D_1 \sqsubseteq W_1$. Also consider a serial episode $D_2 = s \mapsto [S_n] \mapsto t$. Then, by induction hypothesis, for every W_2 such that $se(D_2) \subseteq se(W_2)$, it holds that $D_2 \sqsubseteq W_2$.

Since D is parallel-free, it holds that $se(D) = se(D_1) \cup se(D_2)$ and $se(D_1) \cap se(D_2) = \emptyset$. Then, for a parallel-free ATL-digraph $W = (V, A)$, it holds that $se(D) \subseteq se(W)$ iff $W_1 \cup W_2 \subseteq W$. Hence, it is sufficient to show that $D \sqsubseteq W$.

For $i = 1, 2$, let v_i and u_i be vertices in $V(D_i)$ such that $id_{D_i}(v_i) = 0$ and $od_{D_i}(u_i) = 0$. It holds that $l(v_1) = l(v_2)$ and $l(u_1) = l(u_2)$. Also, for $1 \leq j \leq n$, let w_j and x_j be vertices in $V(S_j)$ such that $id_{S_j}(w_j) = 0$ and $od_{S_j}(x_j) = 0$. Since $D_1 \sqsubseteq W$ and $D_2 \sqsubseteq W$, there are four cases of embedding of D_1 and D_2 into W; (1) $v_1 = v_2$ and $u_1 = u_2$, (2) $v_1 \neq v_2$ and $u_1 = u_2$, (3) $v_1 = v_2$ and $u_1 \neq u_2$ and (4) $v_1 \neq v_2$ and $u_1 \neq u_2$.

For the case (1), it holds that $D_1 \cup D_2 = D$, then it is obvious that $D \sqsubseteq W$.

Consider the case (2). Since W is a parallel-free ATL-digraph and $l(v_1) = l(v_2)$, there exists an arc either (2a) $(v_1, v_2) \in A$ or (2b) $(v_2, v_1) \in A$. See Figure 3 (2a) and (2b).

For the case (2a), since $(v_2, w_n) \in A$ and W is transitive, there exists an arc $(v_1, w_n) \in A$. Hence, it holds that $D \sqsubseteq W - \{v_2\}$, that is, $D \sqsubseteq W$. On the other hand, for the case (2b), since $(v_1, w_j) \in A$ $(1 \leq j \leq n-1)$ and W is transitive, there exists an arc $(v_2, w_j) \in A$ for $1 \leq j \leq n-1$. Hence, it holds that $D \sqsubseteq W - \{v_1\}$, that is, $D \sqsubseteq W$.

Consider the case (3). Since W is a parallel-free ATL-digraph and $l(u_1) = l(u_2)$, there exists an arc either (3a) $(u_1, u_2) \in A$ or (3b) $(u_2, u_1) \in A$. See Figure 3 (3a) and (3b).

For the case (3a), since $(x_j, u_1) \in A$ $(1 \leq j \leq n-1)$ and W is transitive, there exists an arc $(x_j, u_2) \in A$ for $1 \leq j \leq n-1$. Hence, it holds that $D \sqsubseteq W - \{u_1\}$, that is, $D \sqsubseteq W$. On the other hand, for the case (3b), since $(x_n, u_2) \in A$ and W is transitive, there exists an arc $(x_n, u_1) \in A$. Hence, it holds that $D \sqsubseteq W - \{u_2\}$, that is, $D \sqsubseteq W$.

For the case (4), we can prove $D \sqsubseteq W$ by combining the cases (2) and (3). □

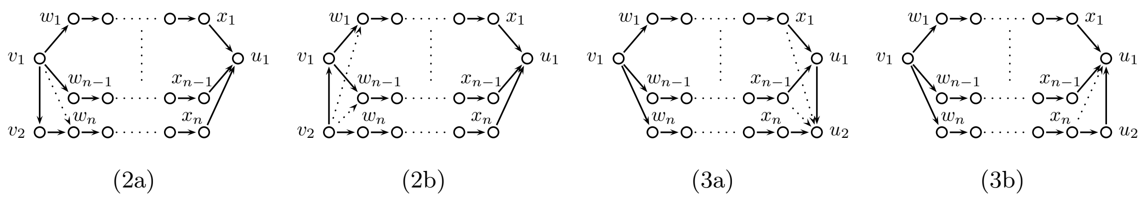

Fig. 3. A parallel-free ATL-digraph W in the proof of Theorem 1.

4 Algorithm to Extract Frequent Elliptic Episodes

In this section, we design the algorithm to extract all of the frequent elliptic episodes from an event sequence $\mathcal{S}$, the minimum support σ, the width k of windows, and the maximum length n of elliptic episodes. If it is necessary to extract frequent elliptic episodes with arbitrary length, then set n to k.

The algorithm NewBitSerl, described as Figure 4, computes a bit vector of the occurrences of all serial episodes within the length n, by scanning an event sequence. Here, for a bit vector $v \in \{0,1\}^*$, $|v|$

denotes the length of v, and v_i denotes the i-th bit of v for $1 \leq i \leq |v|$, that is, $v = v_1 \cdots v_{|v|}$. Then, $\mathit{sft}(v)$ is $v_2 \cdots v_{|v|}0$. For $v, w \in \{0,1\}^*$, $v \circ w$ denotes the concatenation of v and w. Furthermore, for $v, w \in \{0,1\}^*$ such that $|v| = |w|$, $v \wedge w$ and $v \vee w$ are bit-wise logical conjunction and bit-wise logical disjunction, respectively. In particular, $0(k)$ and $1(k)$ denote the vectors $0 \cdots 0$ and $1 \cdots 1$ with length k, respectively. For a set $\mathcal{E}$ of event types, let $\mathcal{E}^{\leq n} = \{s \in \mathcal{E}^* \mid |s| \leq n\}$.

```
procedure NewBitSerl($S, T_s, T_e, k, \mathcal{E}, n$) /* $\langle S, T_s, T_e \rangle$: event sequence */
/* $k$: the width of windows, $\mathcal{E}$: event types, $n$: the maximum length of elliptic episodes */
/* initialization, where $B[e] = B[e][T_s] \cdots B[e][T_e - 1]$ */
for $t = T_s$ to $T_e - 1$ do
        foreach $e \in \mathcal{E}$ do $B[e][t] \leftarrow 0$;
/* transforming an event sequence to a set of bit vectors */
for $t = T_s$ to $T_e - 1$ do
        foreach $e \in \mathcal{E}$ s.t. $(e, t) \in S$ do $B[e][t] \leftarrow 1$;
/* constructing a bit vector of a serial episode $e_1 \mapsto \cdots \mapsto e_n$ */
$l \leftarrow T_e - T_s + k - 1$; $T \leftarrow \emptyset$;
foreach $s \in \mathcal{E}^{\leq n}$ do begin /* $|V[\cdot]| = |W[\cdot]| = l$ */
        for $i = 1$ to $|s|$ do
                $V[i] \leftarrow 0(k-1) \circ B[s[i]]$;  $W[i] \leftarrow 0(l)$;
        $W[1] \leftarrow V[1]$;
(*)     for $d = 1$ to $k - 1$ do begin
(*)             for $m = |s|$ downto 2 do
(*)                     $W[m] \leftarrow \mathit{sft}(W[m]) \vee (V[|s| - m + 1] \wedge \mathit{sft}(W[m-1]))$;
(*)             $W[1] \leftarrow \mathit{sft}(W[1]) \vee V[|s|]$;
(*)     end /* for */
        $T \leftarrow T \cup \{W[|s|]\}$; /* $W[|s|] = W[|e_1 \mapsto \cdots \mapsto e_{|s|}|]$ */
end /* foreach */
return $T$;
```

Fig. 4. The algorithm NewBitSerl.

The difference between the algorithm BitSerl in [4] and the algorithm NewBitSerl in this paper is that, while BitSerl finds just bit vectors of serial episode with the length 3, NewBitSerl finds bit vectors of serial episode within the length n, by using the for-loop signed by (*). In this part, by repeating the shift operation at $k - 1$ times, we find the bit vector of a serial episode $e_1 \mapsto \cdots \mapsto e_n$. By using the following lemma, we can show the correctness of the algorithm NewBitSerl as similar as one of the algorithm BitSerl [4].

Lemma 2 (Katoh *et al.* **[4]).** *Let $\mathcal{S}$ be an event sequence $\langle S, T_s, T_e \rangle$. Then, a serial episode $e_1 \mapsto \cdots \mapsto e_n$ occurs in a window $w(\mathcal{S}, t, k)$ if and only if one of the following two statements holds.*

1. *A serial episode $e_1 \mapsto \cdots \mapsto e_n$ occurs in a window $w(\mathcal{S}, t+1, k-1)$.*
2. *An event e_1 occurs in a window $w(\mathcal{S}, t, 1)$ and a serial episode $e_2 \mapsto \cdots \mapsto e_n$ occurs in a window $w(\mathcal{S}, t+1, k-1)$.*

After constructing the set of bit vectors representing the occurrences of all serial episodes within the length n by the algorithm NewBitSerl, we design the algorithm FreqEllip to extract all of the frequent elliptic episodes under the minimum support σ as Figure 5. The algorithm FreqEllip is based on the anti-monotonicity of elliptic episodes (Lemma 1) and the constructibility of elliptic episodes from serial episodes (Theorem 1).

While the algorithm FreqEllip is an extension of the algorithm FreqDmd [4], the main difference between the algorithms FreqDmd and FreqEllip is that, while we can apply the algorithm AprioriTid [1] directly in the algorithm FreqDmd, it is necessary to improve the algorithm AprioriTid in the algorithm FreqEllip, because it is necessary to check whether or not every pair of serial episodes in the set obtained from the algorithm NewBitSerl is parallel-free. Then, we adopt the algorithm PlfrFq instead of AprioriTid.

In the algorithm PlfrFq, U is a bit vector of the occurrences of each element of I in an event sequence and M is a bit vector representing whether or not each element of I contains an event type in $\mathcal{E}$. Then, by using M, the algorithm PlfrFq checks whether or not the set I of serial episodes is

```
procedure FreqEllip($S, T_s, T_e, k, \mathcal{E}, n, \sigma$)
/* $\langle S, T_s, T_e\rangle$: event sequence, $k$: the width of windows, $n$: maximum length of elliptic episodes */
/* $\mathcal{E}$: event types, $\sigma$: the minimum support */
$T \leftarrow$NewBitSerl($S, T_s, T_e, k, \mathcal{E}, n$);  $D \leftarrow \emptyset$;
foreach $(s, t) \in \mathcal{E} \times \mathcal{E}$ do begin
    $\mathcal{I} \leftarrow \emptyset$;
    foreach $W[s \mapsto S \mapsto t] \in T$ do begin /* $S$: serial episode */
        $\mathcal{I} \leftarrow \mathcal{I} \cup \{S\}$;
        foreach $e \in \mathcal{E}$ do $N[e] \leftarrow 0$;
        foreach $e \in S$ do $N[e] \leftarrow 1$;
        $M[S] \leftarrow N$;  $U[S] \leftarrow W[s \mapsto S \mapsto t]$; /* $S$ is regarded as an item */
    end /* foreach */
    foreach $I \subseteq \mathcal{I}$ do /* $I$ is the set of serial episodes */
        if PlfrFq($I, U, M, \sigma, T_e - T_s + k - 1, |\mathcal{E}|$) $= \top$ then $D \leftarrow D \cup \{s \mapsto [I] \mapsto t\}$;
end /* foreach */
return $D$;

procedure PlfrFq($I, U, M, \sigma, l, m$)
/* $I$: set of serial episodes, $U, M$: sets of bit vectors, $\sigma$: the minimum support, $l, m$: natural numbers */
$V \leftarrow 1(l)$;  $N \leftarrow 0(m)$;
foreach $S \in I$ do begin
    $V \leftarrow V \wedge U[S]$;
    if $N \wedge M[S] \neq 0(m)$ then return $\perp$; else $N \leftarrow N \vee M[S]$;
end /* foreach */
$freq \leftarrow count(V)$; /* $count(V)$ returns the number of 1 in $V$*/
if $freq/l < \sigma$ then return $\perp$;  else return $\top$;
```

Fig. 5. The algorithm FreqEllip.

parallel-free. Furthermore, if so, then the algorithm PlfrFq checks whether or not every element in I is frequent, by storing the bit vector of I in V and by counting the number of the occurrence of 1 in V.

Hence, after checking the set $I \subseteq \mathcal{I}$ of serial episodes is parallel-free and frequent by the algorithm PlfrFq, the algorithm FreqEllip constructs an elliptic episode $s \mapsto [I] \mapsto t$ from I under fixing a source s and a sink t, and adds it to D.

As similar as the algorithm FreqDmd [4], the following theorem holds for the algorithm FreqEllip.

Theorem 2. *The algorithm* FreqEllip *extracts all of the frequent elliptic episodes from an event sequence by scanning it just once.*

5 Empirical Results

In this section, by applying the algorithm FreqEllip to bacterial culture data of Osaka Prefectural General Medical Center, we extract frequent elliptic episodes concerned with *changes for drug resistance*.

By fixing the category of bacteria and the sample, we connect data of every patient with the span of 30 days, which is the width of windows. Then, by focusing just on the sensitivity of antibiotics, we extract frequent elliptic episodes of the form $D = s \mapsto [S_1, \ldots, S_n] \mapsto t$ such that $len(D) = 4$, s is of the form $\texttt{Ant}_0$=S (S means "susceptibility"), t is of the form $\texttt{Ant}_0$=R (R means "resistant"), and every S_i is a serial episode of the form ($\texttt{Ant}_i$=S)($\texttt{Ant}_i$=R), where $\texttt{Ant}_i$ $(0 \leq i \leq n)$ is a mutually distinct antibiotic. Here, the minimum support is 0.02%.

Figure 6 describes the result applying the algorithm FreqEllip to bacterial culture data. The computer environment is: CPU is AMD Athlon(TM) 64 Processor 3000+1 1.8GHz and RAM is 2GB.

Figure 7 describes the most frequent elliptic episode $s \mapsto [S_1, \ldots, S_n] \mapsto t$ with the largest n for every category of bacteria and every sample. Here, the subscribe denotes its frequency, and Ant in S_i denotes a serial episode (Ant=S)(Ant=R). Also antibiotics are benzilpenicillin (PcB), synthetic penicillins (PcS), augmentin (Aug), anti-pseudomonas penicillin (PcAP), 1st generation cephems (Cep1), 2nd generation cephems (Cep2), 3rd generation cephems (Cep3), anti-pseudomonas cephems (CepAP), aminoglycosides (AG), macrolides (ML), tetracyclines (TC), carbapenems (CBP) and vancomycin (VCM).

Hence, in the case that the category of bacteria is "Anaerobes" and the sample is "catheter/others," we succeed to extract elliptic episodes more frequent than others, while the number of extracted elliptic

category of bacteria	sample	#records	#patients	#windows	#episodes	time
Stapylococci	catheter/others	775	249	41521	58	0.6594
	respiratory organs	1014	311	46062	189	0.4813
Entercocci	catheter/others	206	70	9318	4	0.1281
Enteric bacteria	catheter/others	441	133	14379	644	0.2219
	urinary/genital organs	304	120	17756	20	0.2219
	respiratory organs	987	308	39917	36	0.5391
Glucose-nonfermentative gram-negative bacteria	catheter/others	238	81	12585	19	0.1906
	respiratory organs	1030	302	48195	12	0.6469
Anaerobes	catheter/others	763	218	19032	113	0.1703

Fig. 6. The empirical results.

category of bacteria	sample	the most frequent elliptic episode
Stapylococci	catheter/others	$(\mathtt{Cep1{=}S})\mapsto[\mathtt{ML,\ PcS,\ RFPFOM,\ TC}]\mapsto(\mathtt{Cep1{=}R})_{0.036\%}$ $(\mathtt{VCM{=}S})\mapsto[\mathtt{ML,\ PcS,\ RFPFOM,\ TC}]\mapsto(\mathtt{VCM{=}R})_{0.036\%}$
	respiratory organs	$(\mathtt{PcB{=}S})\mapsto[\mathtt{CBP,\ Cep1,\ ML,\ TC}]\mapsto(\mathtt{PcB{=}R})_{0.052\%}$ $(\mathtt{PcS{=}S})\mapsto[\mathtt{CBP,\ Cep1,\ ML,\ TC}]\mapsto(\mathtt{PcS{=}R})_{0.052\%}$
Entercocci	catheter/others	$(\mathtt{VCM{=}S})\mapsto[\mathtt{AG,ML}]\mapsto(\mathtt{VCM{=}R})_{0.160\%}$
Enteric bacteria	catheter/others	$(\mathtt{CBP{=}S})\mapsto[\mathtt{Aug,\ Cep1,\ Cep2,\ Cep3,\ CepAP,\ PcAP,\ TC}]\mapsto(\mathtt{CBP{=}R})_{0.090\%}$ $(\mathtt{Cep3{=}S})\mapsto[\mathtt{Aug,\ CBP,\ Cep1,\ Cep2,\ CepAP,\ PcAP,\ TC}]\mapsto(\mathtt{Cep3{=}R})_{0.090\%}$ $(\mathtt{CepAP{=}S})\mapsto[\mathtt{Aug,\ CBP,\ Cep1,\ Cep2,\ Cep3,\ PcAP,\ TC}]\mapsto(\mathtt{CepAP{=}R})_{0.090\%}$ $(\mathtt{PcAP{=}S})\mapsto[\mathtt{Aug,\ CBP,\ Cep1,\ Cep2,\ Cep3,\ CepAP,\ TC}]\mapsto(\mathtt{PcAP{=}R})_{0.090\%}$ $(\mathtt{TC{=}S})\mapsto[\mathtt{Aug,\ CBP,\ Cep1,\ Cep2,\ Cep3,\ CepAP,\ PcAP}]\mapsto(\mathtt{TC{=}R})_{0.090\%}$
	urinary/genital organs	$(\mathtt{Cep2{=}S})\mapsto[\mathtt{Aug,\ Cep1,\ PcS}]\mapsto(\mathtt{Cep2{=}R})_{0.028\%}$ $(\mathtt{TC{=}S})\mapsto[\mathtt{Aug,\ Cep1,\ PcS}]\mapsto(\mathtt{TC{=}R})_{0.028\%}$
	respiratory organs	$(\mathtt{TC{=}S})\mapsto[\mathtt{Aug,\ Cep1}]\mapsto(\mathtt{TC{=}R})_{0.047\%}$
Glucose-nonfermentative gram-negative bacteria	catheter/others	$(\mathtt{CBP{=}S})\mapsto[\mathtt{Aug,\ Cep1,\ PcS,\ TC}]\mapsto(\mathtt{CBP{=}R})_{0.079\%}$
	respiratory organs	$(\mathtt{CBP{=}S})\mapsto[\mathtt{CepAP,\ PcAP}]\mapsto(\mathtt{CBP{=}R})_{0.033\%}$
Anaerobes	catheter/others	$(\mathtt{Cep3{=}S})\mapsto[\mathtt{ML,\ PcAP}]\mapsto(\mathtt{Cep3{=}R})_{0.220\%}$

Fig. 7. The most frequent elliptic episodes that are not serial.

episodes in the case that the category of bacteria is "Enteric bacteria" and the sample is "catheter/others" is largest. Furthermore, the changes for drug resistant of `Aug` in "Enteric bacteria" and `CBP` in "Glucose-nonfermentative gram-negative bacteria" in Figure 7 are interesting from medical viewpoints.

6 Conclusion

In this paper, after formulating an episode and an event sequence as a parallel-free ATL-digraph, we have introduced a *constructible episode from serial episodes* and an *elliptic episode*, and shown that every elliptic episode is constructible from serial episodes. Then, we have designed the algorithm FreqEllip to extract all of the frequent elliptic episodes from an event sequence. Finally, we have applied the algorithm FreqEllip to bacterial culture data in order to extract frequent elliptic episodes concerned with *changes for drug resistance*.

As a future work, it remains open whether or not every parallel-free episode is constructible from serial episodes. Also it is a future work to investigate the empirical results from the medical viewpoints.

References

1. R. Agrawal, R. Srikant: *Fast algorithms for mining association rules in large databases*, Proc. 20th VLDB, 487–499, 1994.
2. C. Bettini, S. Wang, S. Jajodia, J.-L. Lin: *Discovering frequent event patterns with multiple granularities in time sequences*, IEEE Trans. Knowledge and Data Engineering **10**, 222–237, 1998.
3. T. Katoh, K. Hirata, M. Harao: *Mining sectorial episodes from event sequences*, Proc. 10th DS, LNAI **4265**, 137–145, 2006.
4. T. Katoh, K. Hirata, M. Harao: *Mining frequent diamond episodes from event sequences*, Proc. 4th MDAI, LNAI, 2007 (to appear). Also available at `http://www.dumbo.ai.kyutech.ac.jp/hirata/MDAI2007.pdf`.
5. T. Katoh, K. Hirata, M. Harao, S. Yokoyama, K. Matsuoka: *Extraction of sectorial episodes representing changes for drug resistant and replacements of bacteria*, Proc. CME'07, 2007 (to appear).
6. H. Mannila, H. Toivonen, A. I. Verkamo: *Discovery of frequent episodes in event sequences*, Data Mining and Knowledge Discovery **1**, 259–289, 1997.