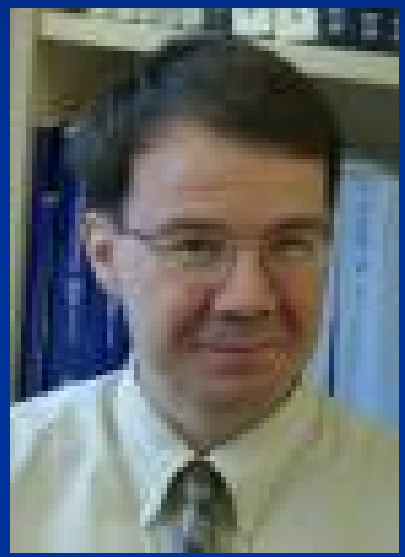
Discovery of frequent episodes in event sequences using WINEPI

Maurizio Atzori

University of Pisa

The authors



Heikki Mannila

Hannu Toivonen

Inkeri Verkamo

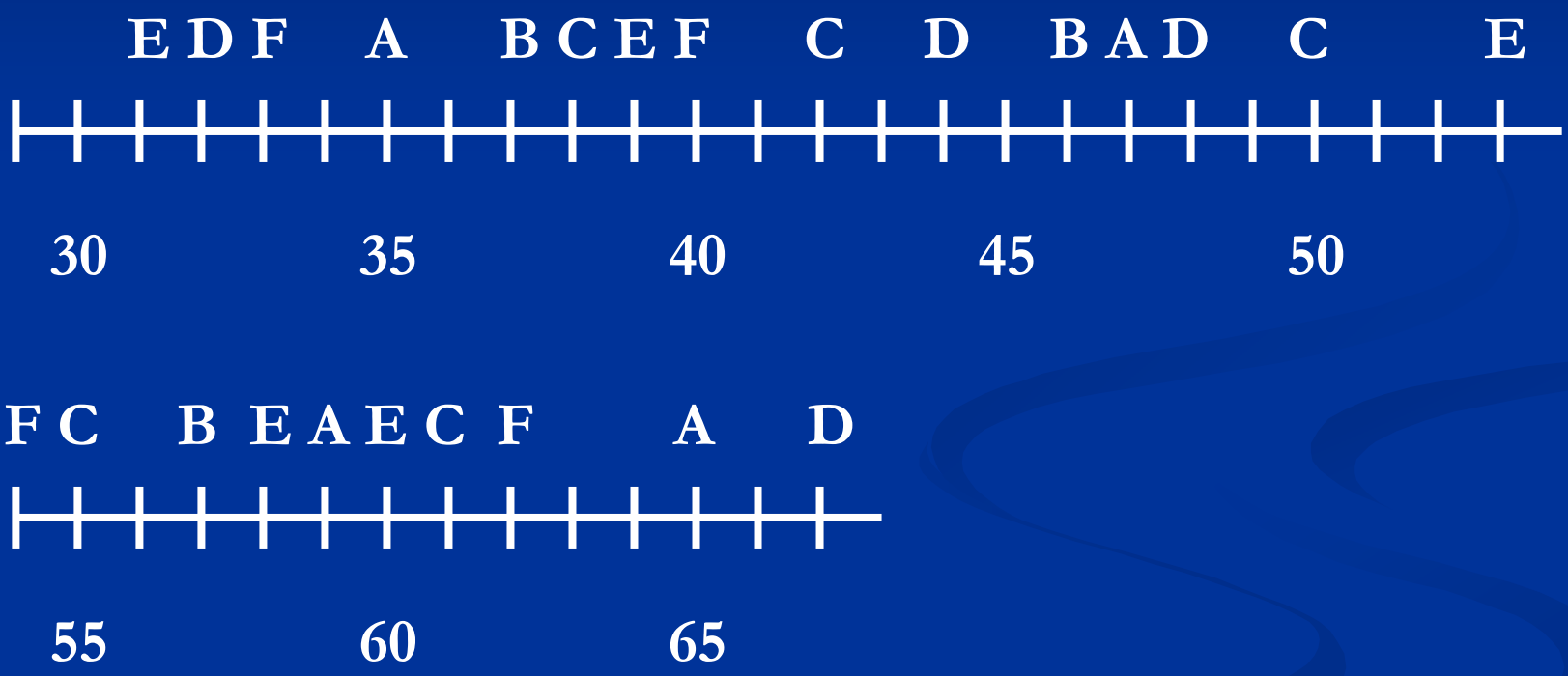
Index

- Definitions and examples
  - Event sequences
  - Episodes
  - Windows
- Algorithms (WINEPI)
  - Based on finding "incrementally" frequent episodes
    - Idea taken from data mining association rules (apriori)
- Conclusions

Definitions: event sequences

- An ***event*** is a pair (A,t) where
  - ***A*** $\in$ E, with E set of event types
  - ***t*** is the ***occurrence time*** of the event
- An ***event sequence s on E*** is a triple (s,T_s,T_e) where:
  - $s = <(A_1,t_1),(A_2,t_2), \ldots ,(A_n,t_n)>$
  - $t_i < t_{i+1} \quad \forall\, i = 1, \ldots , n-1$

Example of event sequence



s = <(E,31), (D,32), (F,33), (A,35), (B,37) ... , (D,67)>

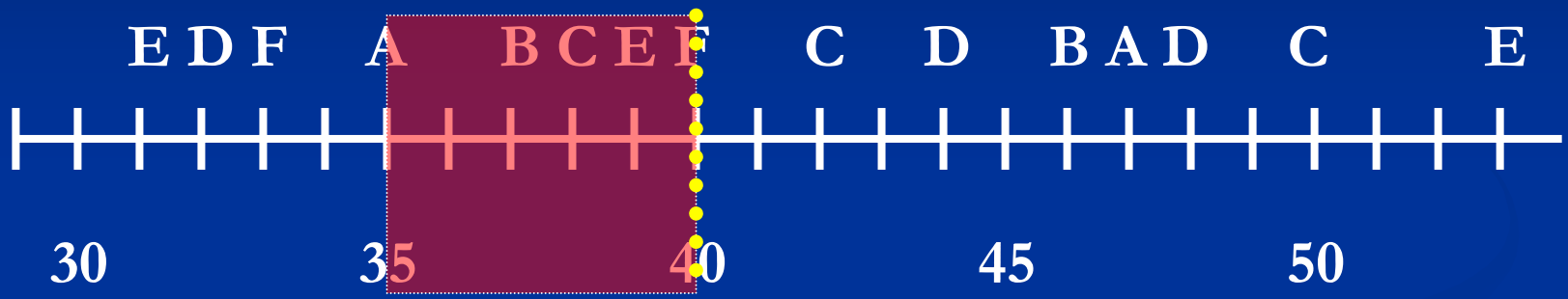
s = (s, 29, 68)

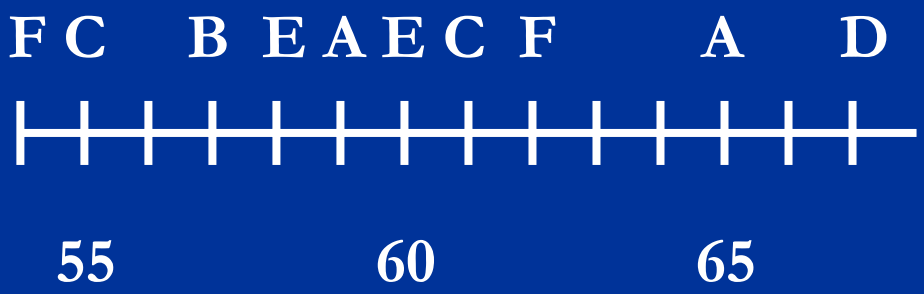
Definitions: windows®

- A ***window*** on event sequence **s**=(s, T_s, T_e) is an event sequence **w** = (w, t_s, t_e) where:
  - $t_s < T_e$, $t_e > T_s$
  - w consists of those pairs (A,t) from s where
    - $t_s \leq t < t_e$

- $t_e - t_s$ is called the ***width*** of the window **w**
  - width(**w**)

Example of window





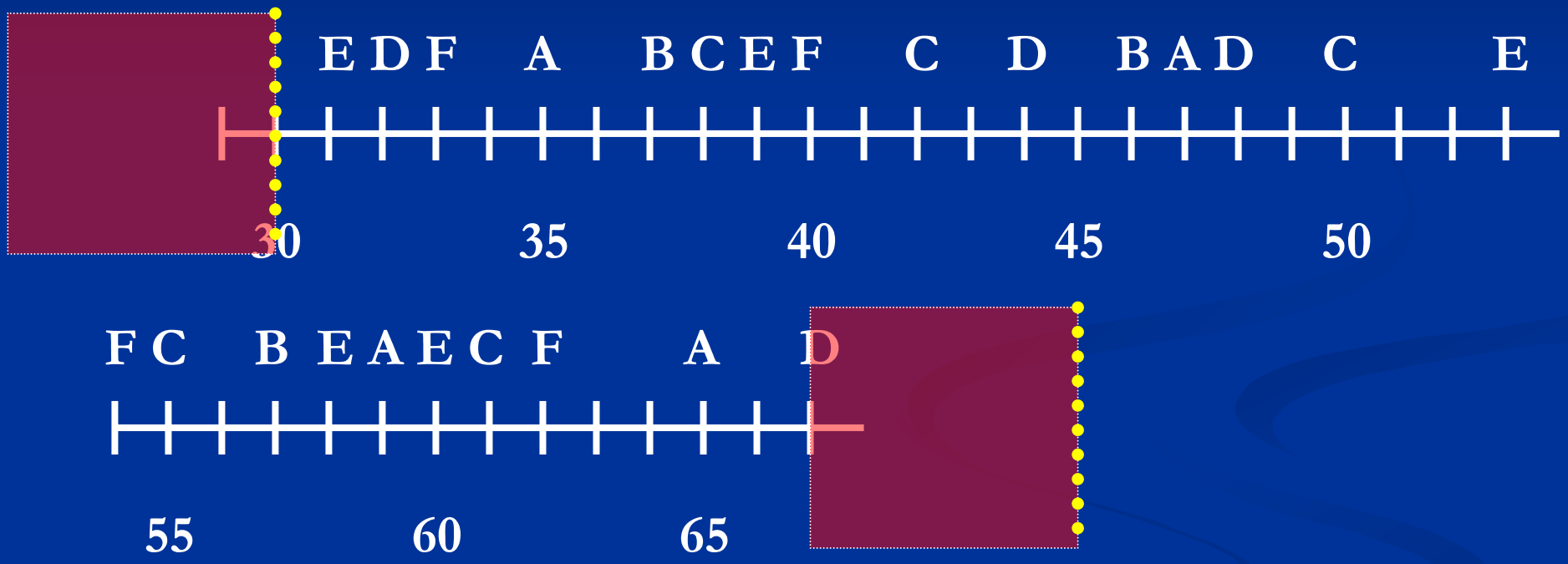
w = < (A,35), (B,37), (C,38), (E,39)>

w = (w, 35, 40)

Definitions: *W(.,.)*

- W(s,win) is the set of all windows w on **s** such that width(w) = win
- #W(s,win) = T_e- T_s +win-1
- In the previous figure, W(s,5):
  - The first window is (ϕ, 25, 30)
  - The last is (<(D,67)>, 67, 72)

Example of W(.,.)

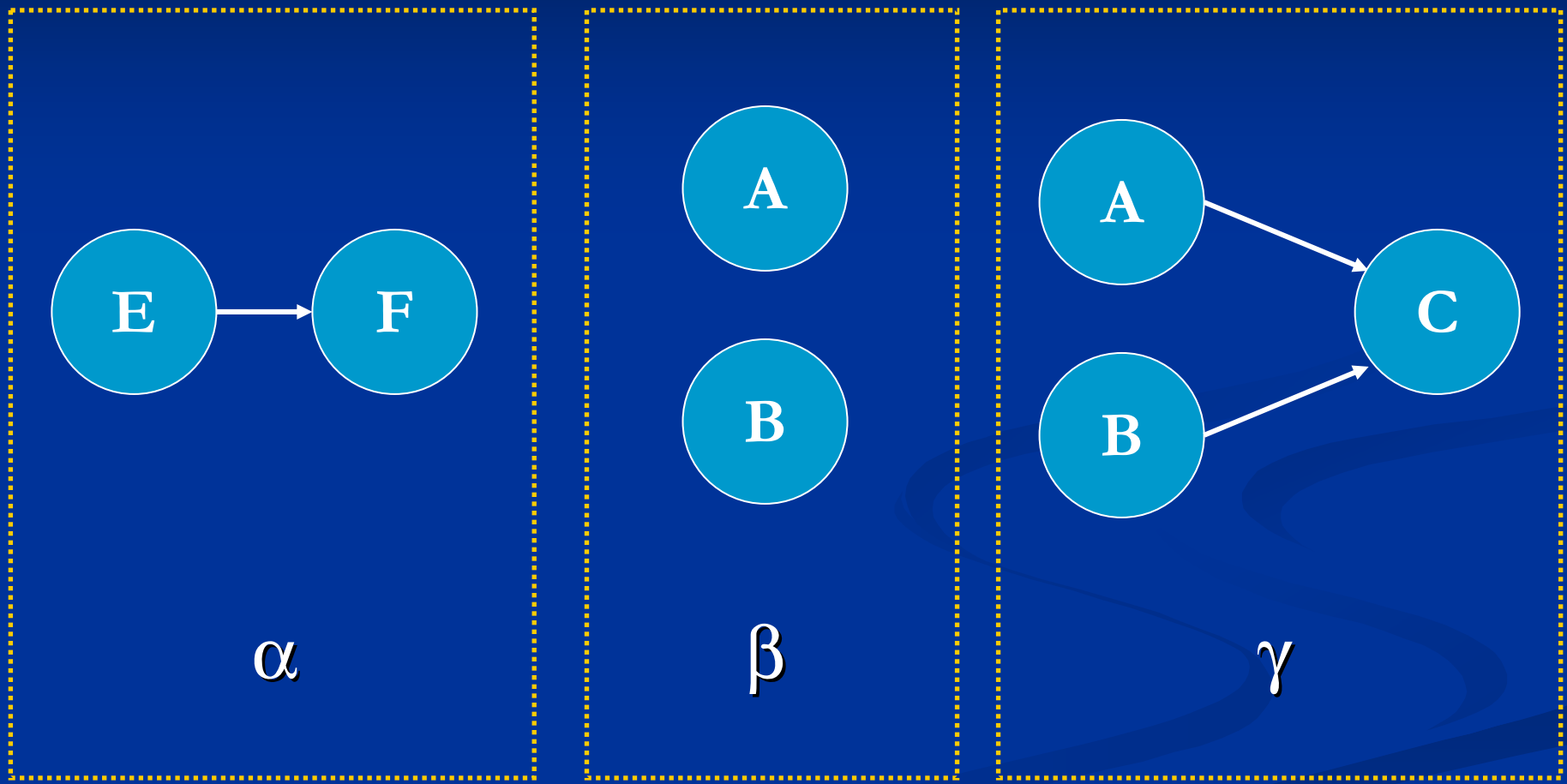


- W(s,5): The first window is (ϕ, 25, 30)
  The last is (<(D,67)>, 67, 72)

Definitions: episodes

- Informally:
  - An ***episode*** is a partially ordered collection of events occurring togheter
- Formally:
  - An episode α is a triple $(V, \leq, g)$ where:
    - V is a set of nodes, $\leq$ is a partial order on V
    - $g: V \rightarrow E$ is a mapping associating each node with an event type
  - The events in g(V) have to occur in the order described by $\leq$
  - The size of α, denoted by $|\alpha|$, is $|V|$

Example of episodes



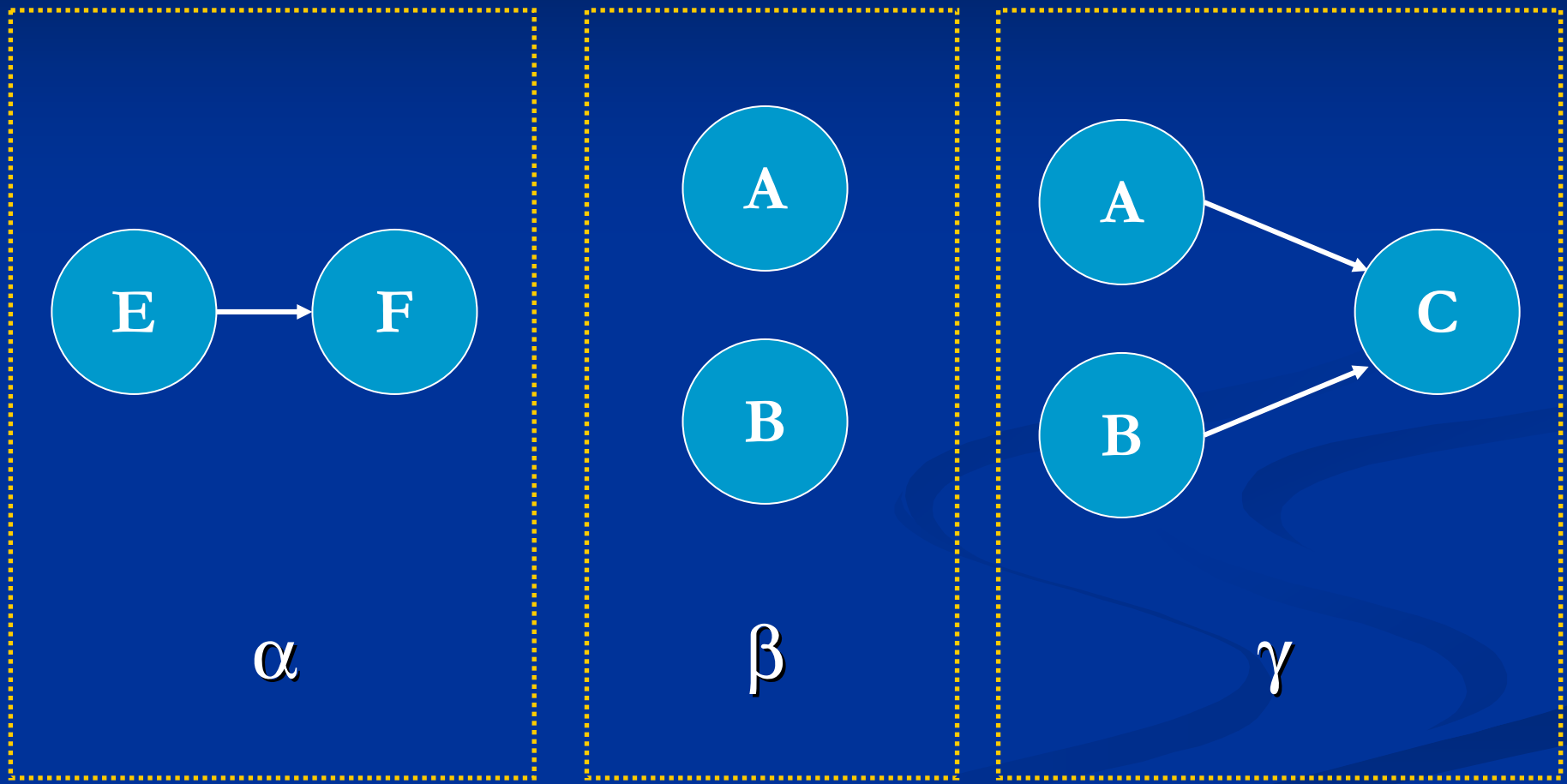
Some kinds of episodes

- Parallel episodes
  - The partial order $\leq$ is trivial
- Serial episodes
  - The $\leq$ relation is a total order
- Injective episodes
  - The mapping g is an injection (i.e. no event type occurs twice in the episode)

Definitions: subepisodes

- An episode $\beta=(V', \leq', g')$ is a ***subepisode*** of $\alpha = (V, \leq, g)$, denoted by $\beta \leq \alpha$, if
  - Exist an injective mapping $f: V' \rightarrow V$ such that
    - $g'(v) = g(f(v)) \ \forall \ v \in V'$
    - $\forall \ v,w \in V'. \ \ v \leq' w \Rightarrow f(v) \leq f(w)$
- Informally:
  - $\beta \leq \alpha \Leftrightarrow$ the graph representing β is a subgraph of α

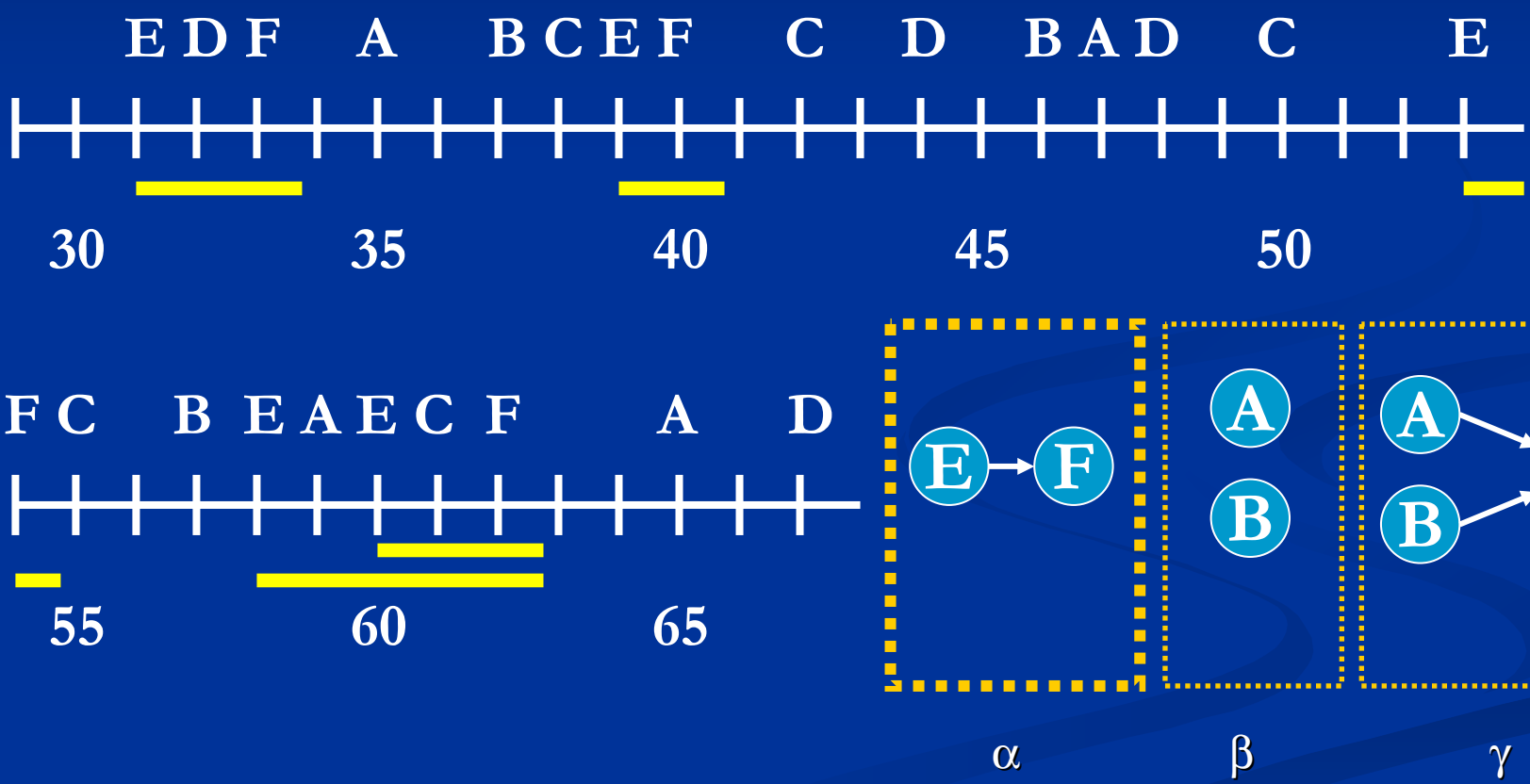
Example of episodes



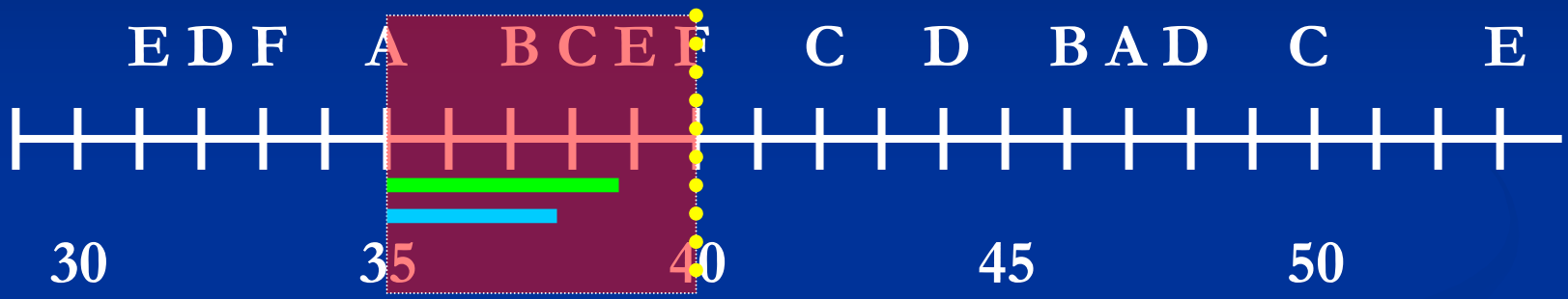
Definitions: occurring

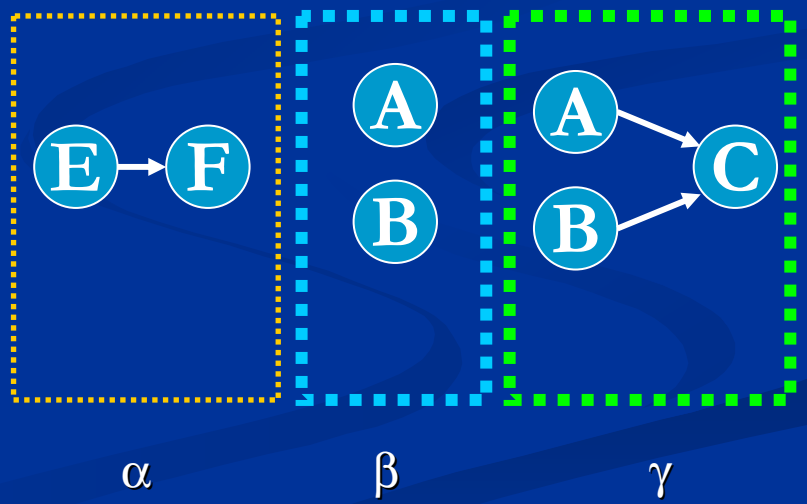
- An episode $\alpha = (V, \leq, g)$ ***occurs*** in an event sequence $s = \langle(A_1,t_1),(A_2,t_2), \ldots ,(A_n,t_n)\rangle$ if:
  - $\exists$ injective mapping $h: V \rightarrow \{1,\ldots,n\}$
    - $g(x) = A_{h(x)} \quad \forall\, x \in V$
    - $x \neq y$ and $x \leq y \Rightarrow t_{h(x)} < t_{h(y)}$

Example of occurring episodes in an event sequence



Example of occurring episodes in a window





Definitions: frequency

- The ***frequency*** of an episode is the fraction of windows in which the episode occurs

$$fr(\alpha,s,win) = \frac{|\{w \in W(s,win) \mid \alpha \text{ occurs in } w\}|}{|W(s,win)|}$$

Given a ***frequency threshold*** *min_fr*, an episode α is ***frequent*** if $fr(\alpha,s,win) \geq min\_fr$

The task

- We are interested to discover all frequent episodes from a given class $\mathcal{E}$ of episodes
  - The class could be, e.g., all parallel episodes or all serial episodes

- $\mathbf{F}(\mathbf{s}, \text{win}, \text{min\_fr})$ is the collection of frequent episodes with respect to s, win and min_fr

Algorithm 1: finding rules

Input: A set E of event types, an event sequence s over E, a set $\mathcal{E}$ of episodes, a window width win, a frequency threshold min_fr, and a confidence threshold min_conf.

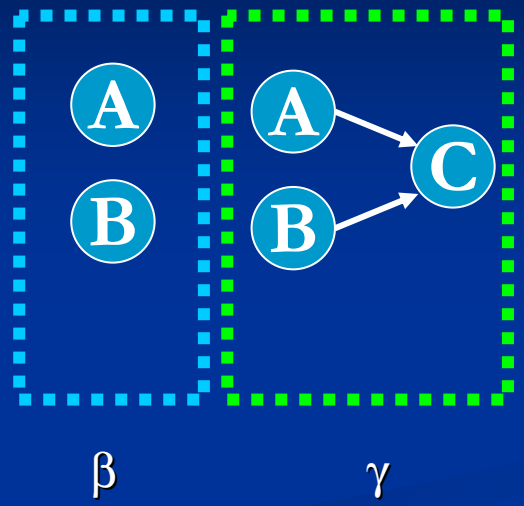
Output: The episode rules that hold in s with respect to win, min_fr, and min_conf.

Method:

1. *`/* Find frequent episodes (Algorithm 2): */`*
2. compute F(s; win; min_fr);
3. *`/* Generate rules: */`*
4. for all $\alpha \in$ F(s; win; min_fr) do
5. for all $\beta < \alpha$ do
6. if fr(α)/fr(β) >= min_conf then
7. output the rule $\beta \rightarrow \alpha$ and the confidence fr(α)/fr(β);

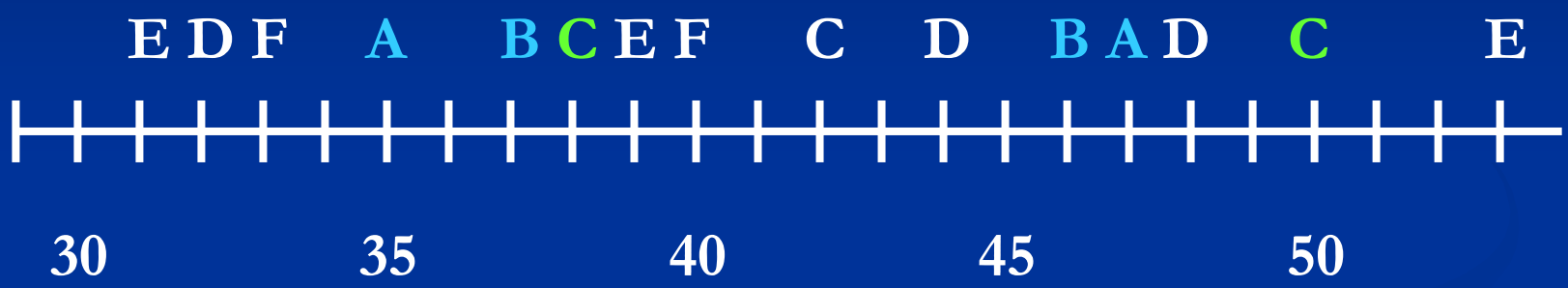
Example of rule

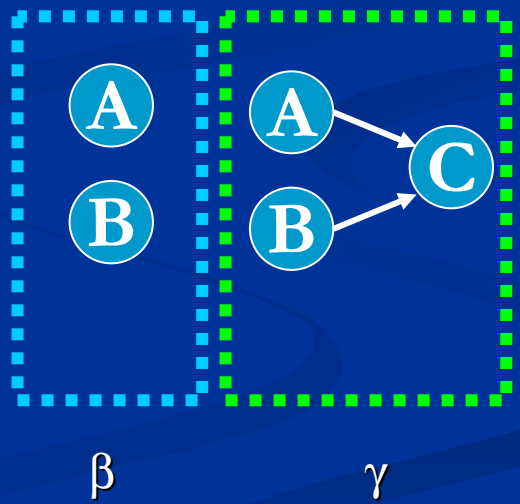
- Given:
  - $\beta < \gamma$
  - $fr(\beta) = 4.2\ \%$
  - $fr(\gamma) = 4.0\ \%$



- Then:
  - If we have A and B in any order then there is a chance of 95% (4.0/4.2) that C follows in the same window
    - A and B → C (95%)

Example of rule (2)





The idea of WINEPI

- If an episode α is frequent in an event sequence s, then all subepisodes $\beta \leq \alpha$ are frequent

- This is used during the generation of candidate episodes, from small to big episodes

Algorithm 2: compute F(.,.)

Input: A set E of event types, an event sequence s over E, set ε of episodes, a window width win, and a frequency threshold min_fr.

Output: The collection F(s; win; min_fr) of frequent episodes.

Method:

1. compute $C_1 := \{\alpha \in \varepsilon \mid |\alpha| = 1\}$;
2. l := 1;
3. while $C_l \neq \phi$ do
4. /* *Database pass (Algorithms 4 and 5):* */
5. compute $F_l := \{\alpha \in C_l \mid fr(\alpha; s; win) \geq min\_fr\}$;
6. l := l + 1;
7. /* *Candidate generation (Algorithm 3):* */
8. compute $C_l := \{\alpha \in \varepsilon \mid |\alpha| = l$ and for all $\beta \in \varepsilon$ such that $\beta < \alpha$ and $|\beta| < l$ we have $\beta \in F_{|\beta|}\}$
9. for all l do output F l ;

Generation of candidate episodes

- C_l is the set containing candidate episodes of length l
  - Generation step
- F_l is the set containing all frequent episodes of length l
  - "Pruning" step

Algorithm 3: candidate episodes generation

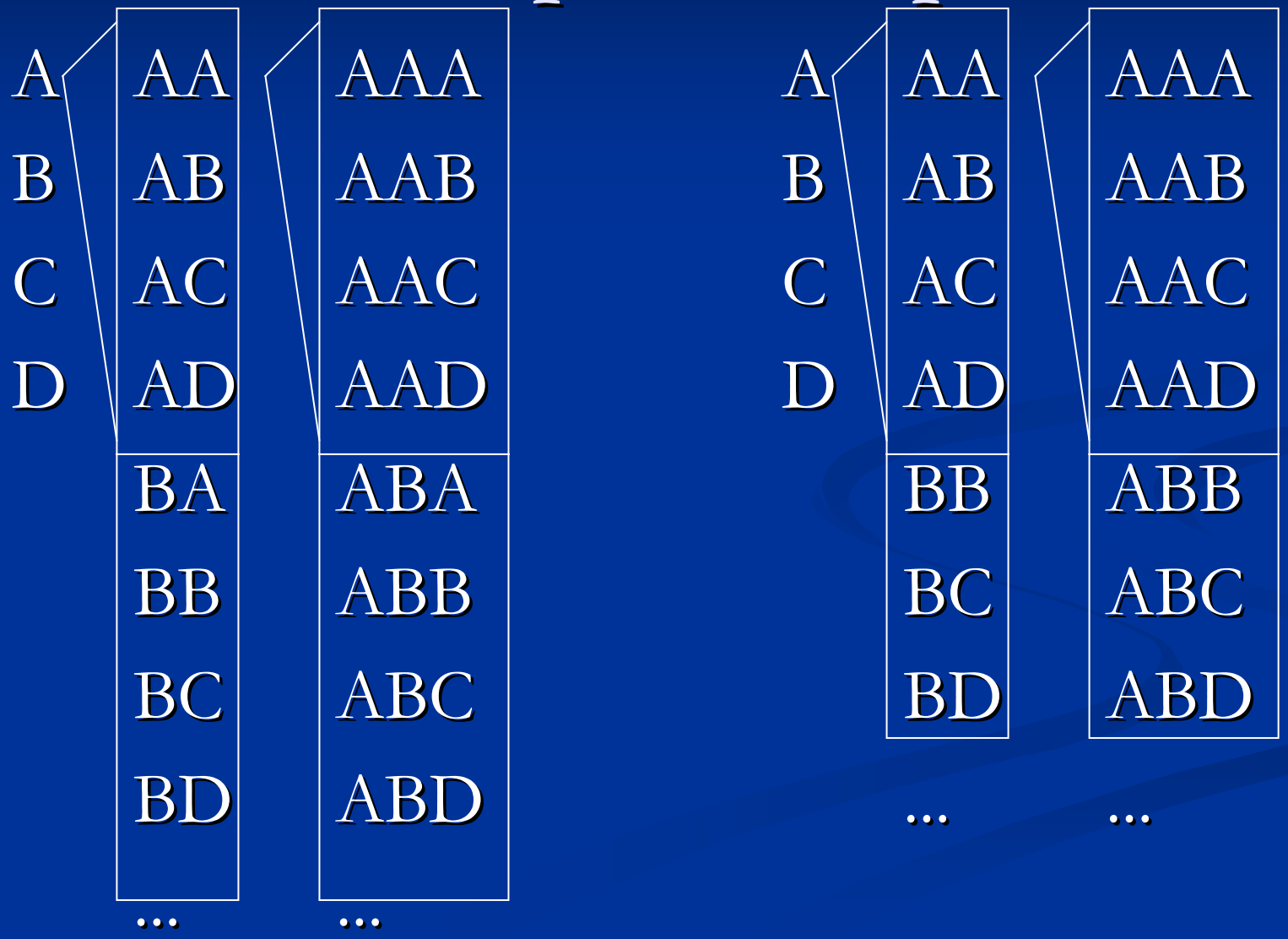
Input: A sorted array F_l of frequent parallel episodes of size l.

Output: A sorted array (C_{l+1}) of candidate parallel episodes of size $l + 1$.

Algorithm 3: two steps

- Build a potential candidate α as a combination of 2 episodes from the same block
- Build and test subepisodes β that do not contain $\alpha[y]$

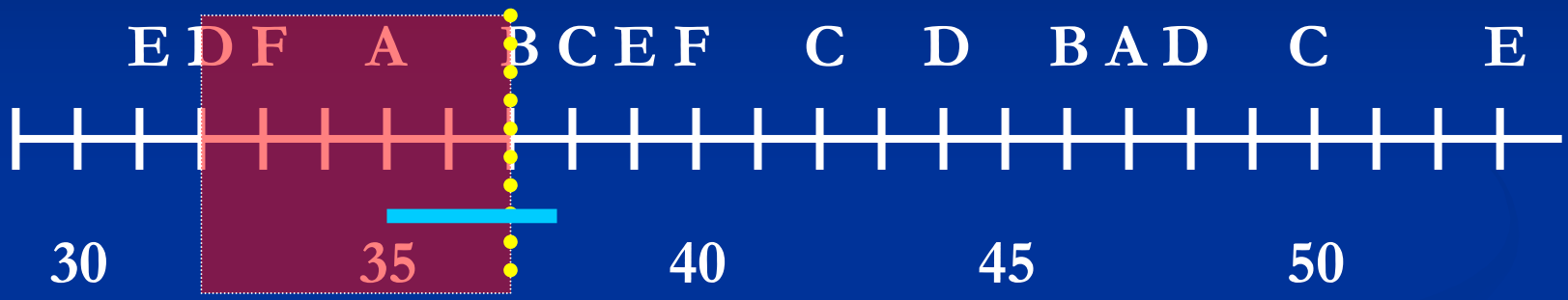
Blocks
serials / parallel episodes

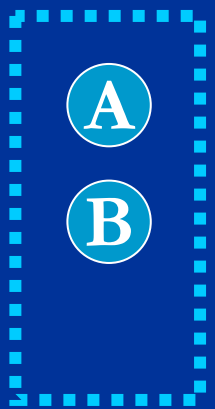


Algorithm 4: recognizing parallel episodes

1. Initialization
2. Recognition
    1. Bring in new events to the window
    2. Drop out old events from the window

Algorithm 4: example





α

D, F, A.count = 1

Bring in

B.count:=B.count+1

Look at Contains(B,1) → If α.event_count =|α|
α.inwin=start

Drop out

Look at Contains(D,1) → If α.event_count =|α|
α.freq_count += start-α.inwin

D.count:= D.count-1

Algorithm 5: recognizing serial episodes

- The basic idea is to have an automaton for each episode

- Data structures:

(α,x) episode α is waiting for xth event

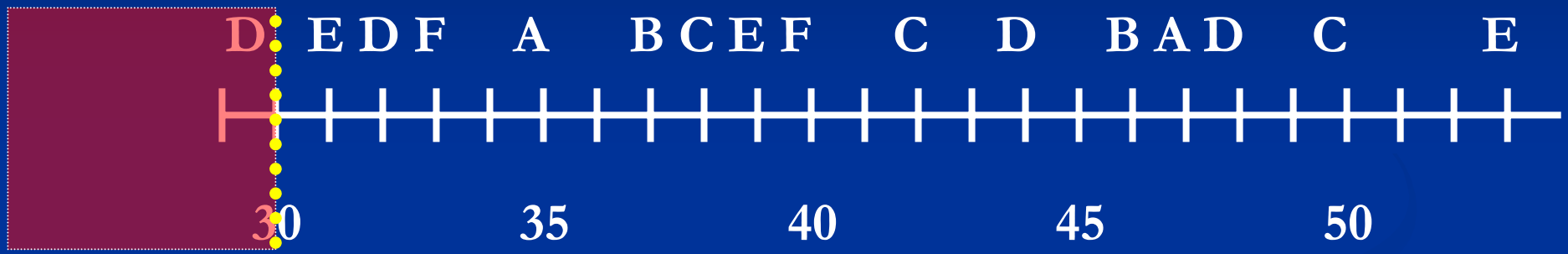
Wait(A) set of episodes waiting for A

α.initialized[i] the latest initalization time of an automaton that has reached its ith state

transitions set of transitions (α,x,t)

episode α got its xth event at time t

Algorithm 5: example



Look at Wait(D)

transitions U= {(α,1,start+win-1)}

Complexity

- Algorithm 3 $O(l^2 |F_1| \log |F_1|)$
- Algorithm 4 $O((n+l^2)|C|)$

  In practice, $l << n$
- Algorithm 5 $O(n|C|l)$

Example of episodes