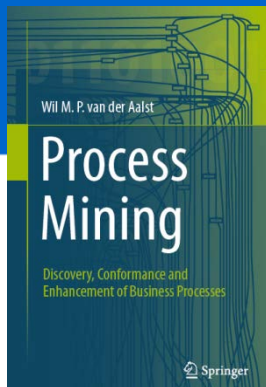


*Process Mining: Data Science in Action*

# Learning Transition Systems

prof.dr.ir. Wil van der Aalst  
[www.processmining.org](http://www.processmining.org)



**TU/e**

Technische Universiteit  
**Eindhoven**  
University of Technology

**Where innovation starts**

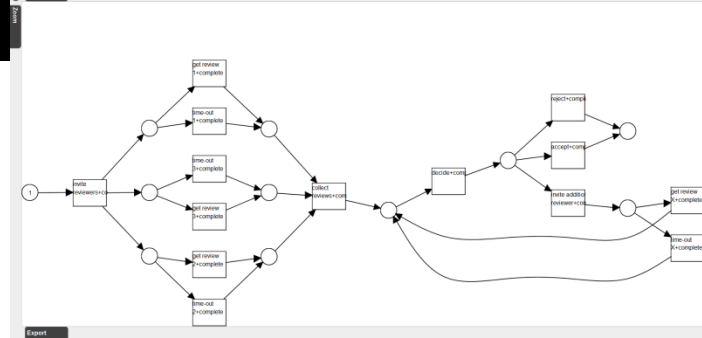
# Discovery approaches seen thus far

# Discovery approaches seen thus far

- Alpha algorithm

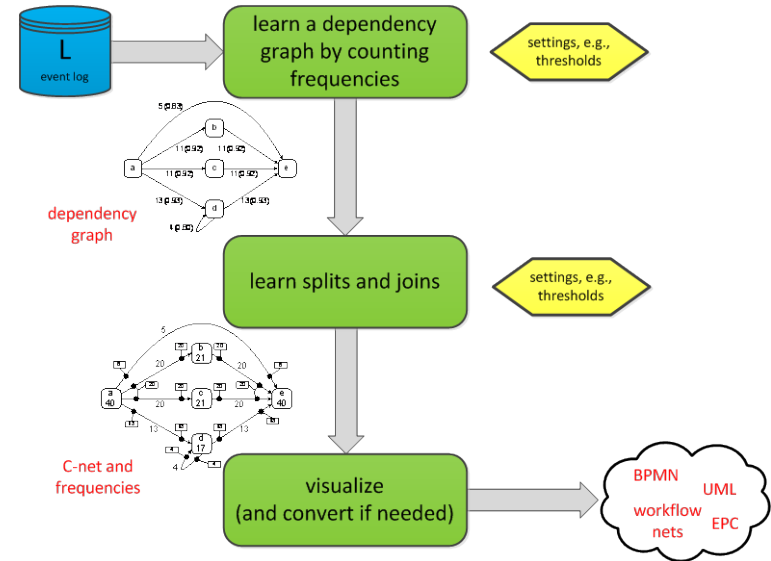
Let  $L$  be an event log over  $T$ .  $\alpha(L)$  is defined as follows.

- $T_L = \{t \in T \mid \exists \sigma \in L \ t \in \sigma\}$ ,
- $T_I = \{t \in T \mid \exists \sigma \in L \ t = \text{first}(\sigma)\}$ ,
- $T_O = \{t \in T \mid \exists \sigma \in L \ t = \text{last}(\sigma)\}$ ,
- $X_L = \{(A, B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall a \in A \forall b \in B \ a \rightarrow_L b \wedge \forall a_1, a_2 \in A \ a_1 \#_L a_2 \wedge \forall b_1, b_2 \in B \ b_1 \#_L b_2\}$ ,
- $Y_L = \{(A, B) \in X_L \mid \forall (A', B') \in X_L \ A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$ ,
- $P_L = \{p_{(A, B)} \mid (A, B) \in Y_L\} \cup \{i_L, o_L\}$ ,
- $F_L = \{(a, p_{(A, B)}) \mid (A, B) \in Y_L \wedge a \in A\} \cup \{(p_{(A, B)}, b) \mid (A, B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) \mid t \in T_I\} \cup \{(t, o_L) \mid t \in T_O\}$  and
- $\alpha(L) = (P_L, T_L, F_L)$ .



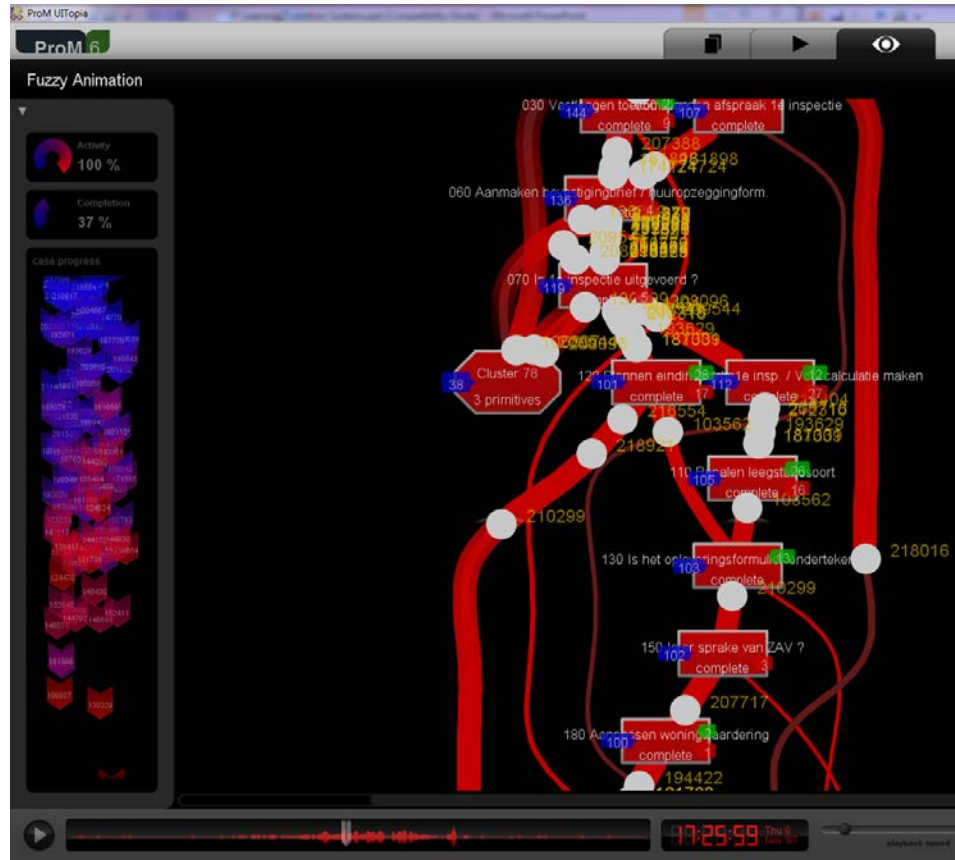
# Discovery approaches seen thus far

- Alpha algorithm
- Heuristic mining
  - dependency graph
  - C-net



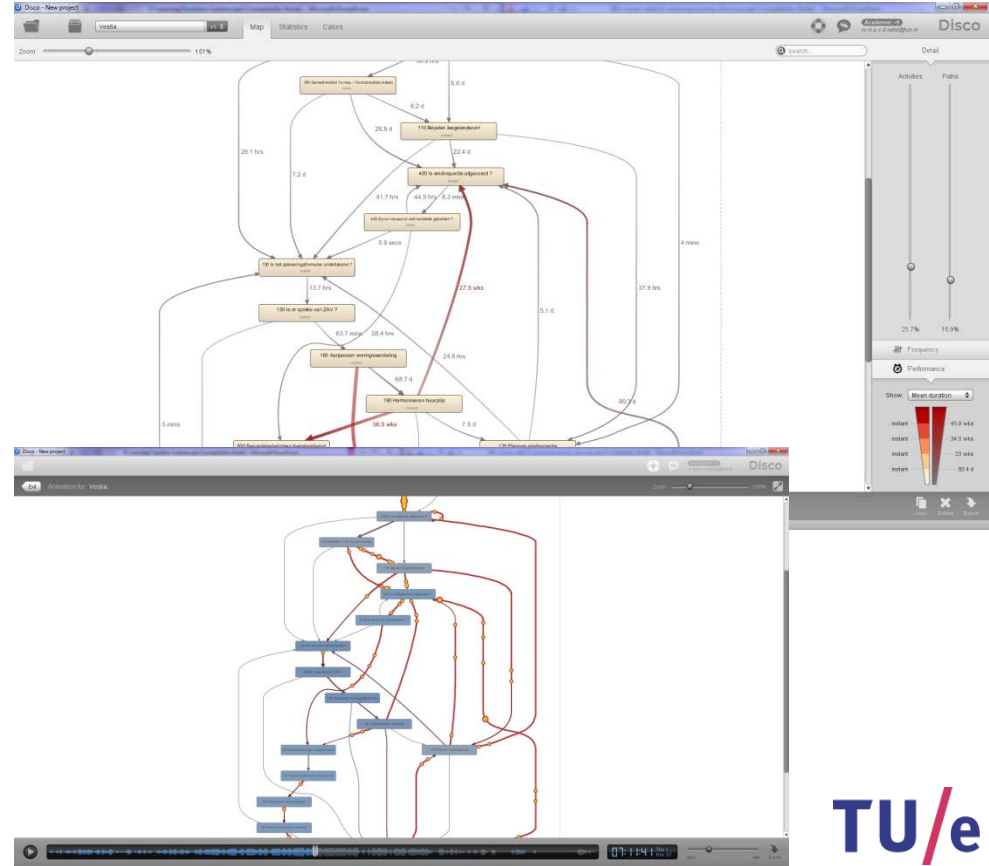
# Discovery approaches seen thus far

- **Alpha algorithm**
- **Heuristic mining**
- **Fuzzy miner  
(tool only)**

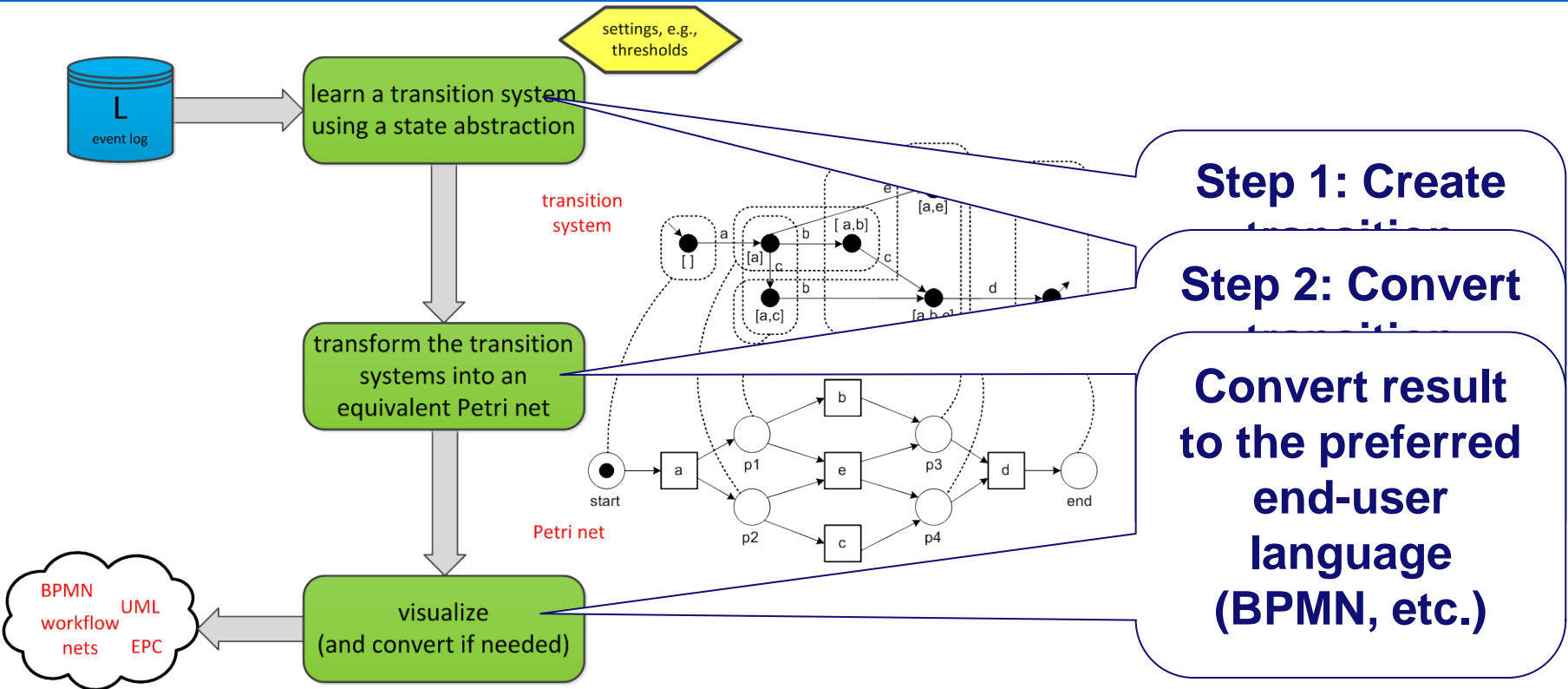


# Discovery approaches seen thus far

- **Alpha algorithm**
- **Heuristic mining**
- **Fuzzy miner**  
(tool only)
- **Disco**  
(tool only)

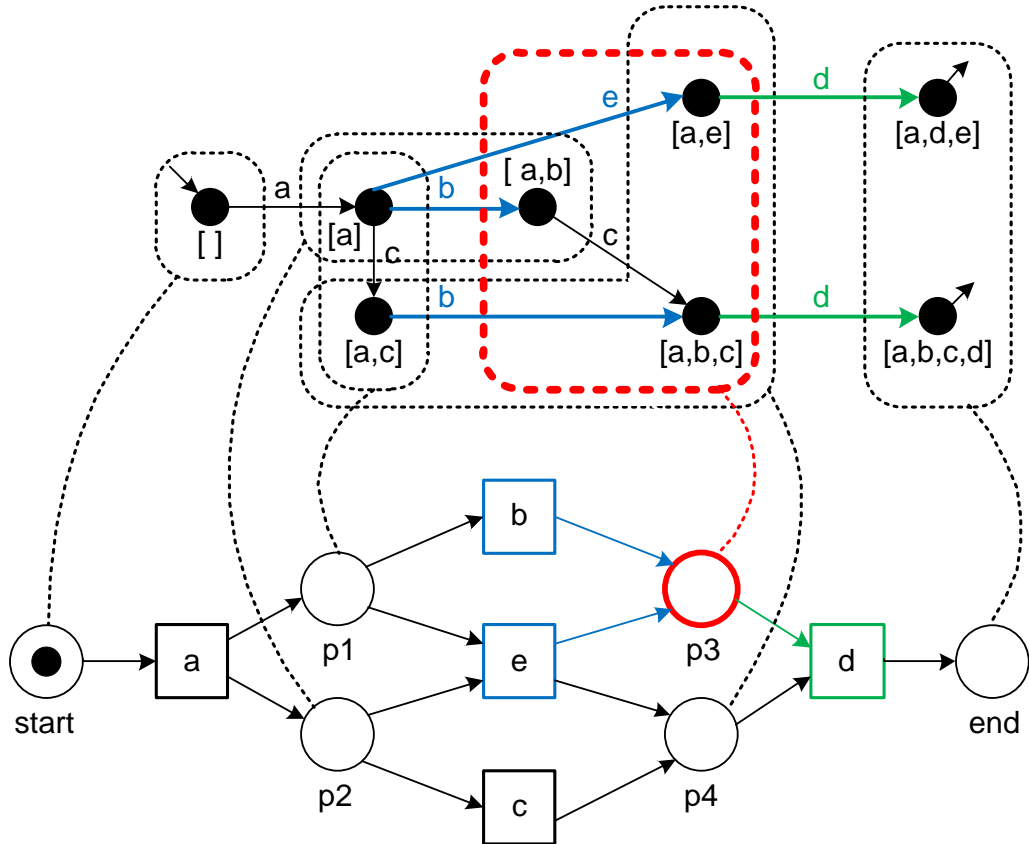


# Another two-phase approach (using state-based regions)



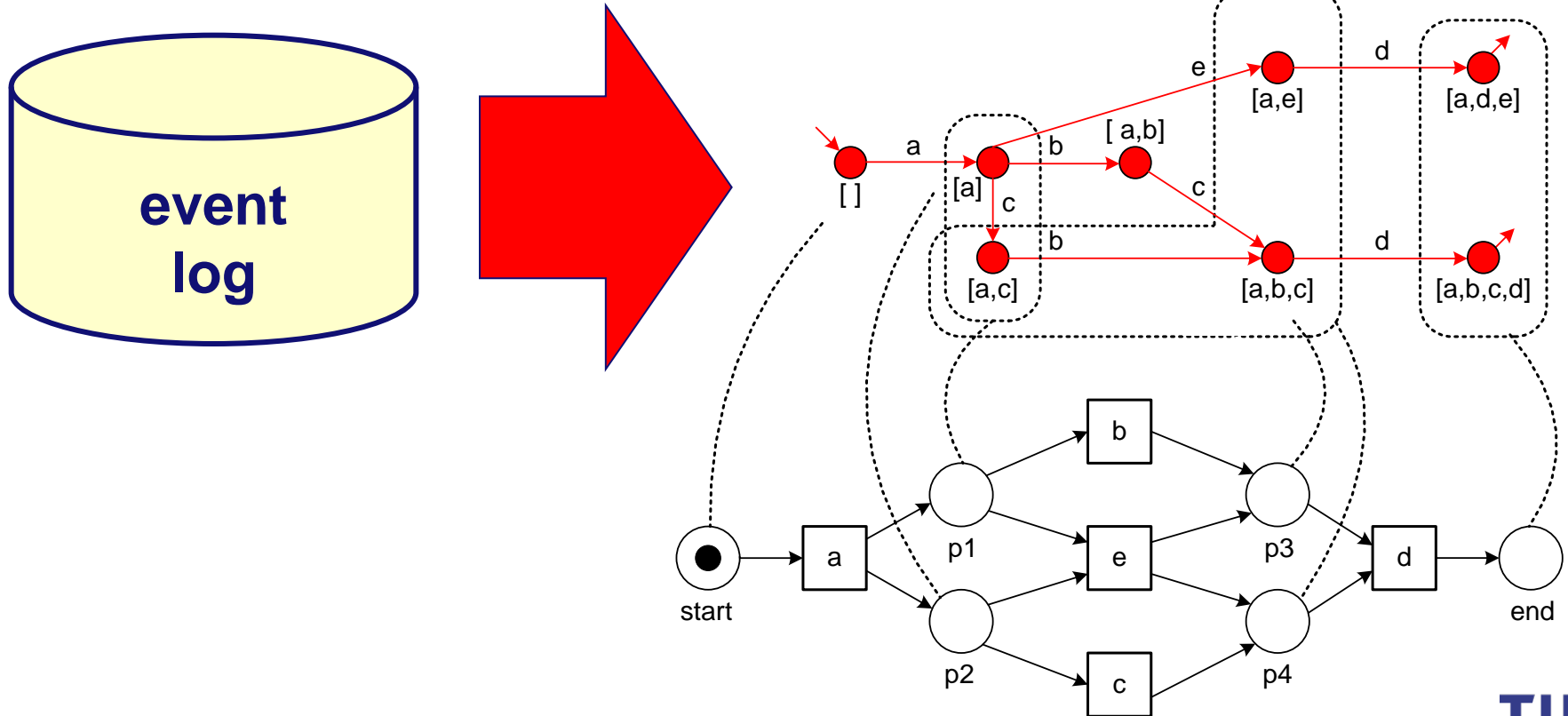
# Second step: State-based regions

It is all about  
discovering  
concurrency ...





# Today's focus

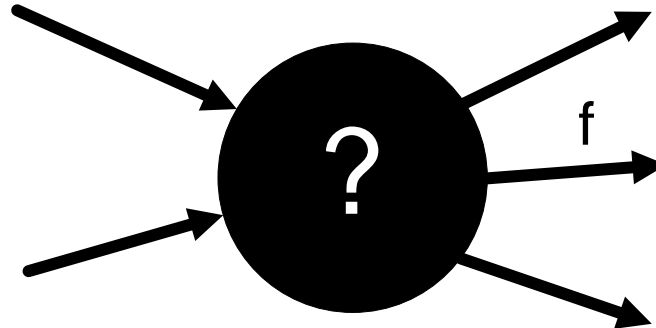


# Learning a transition system

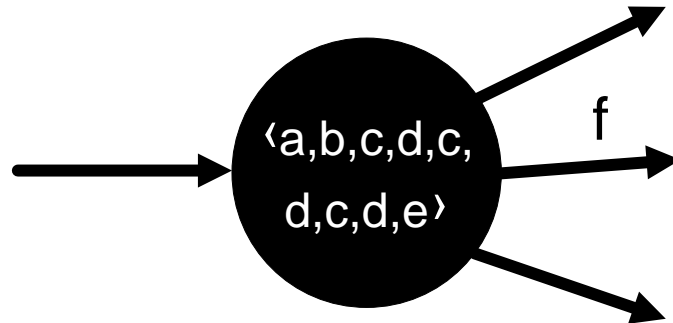
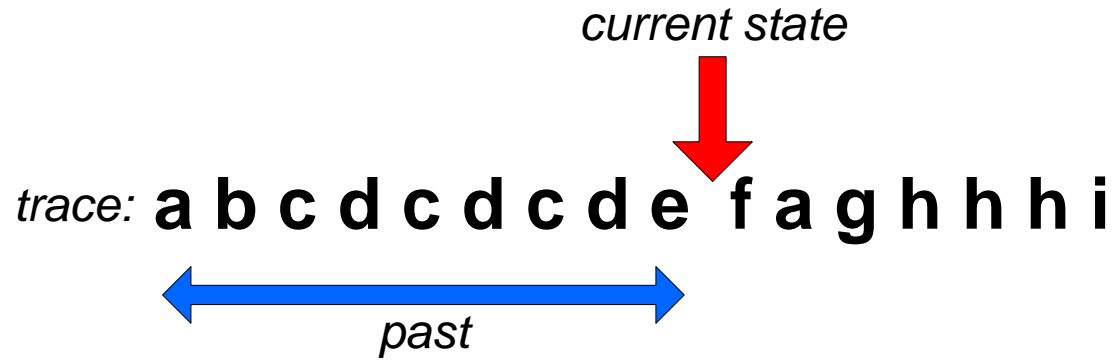
*current state*



*trace:* **a b c d c d c d e f a g h h h i**



# Learning a Transition System

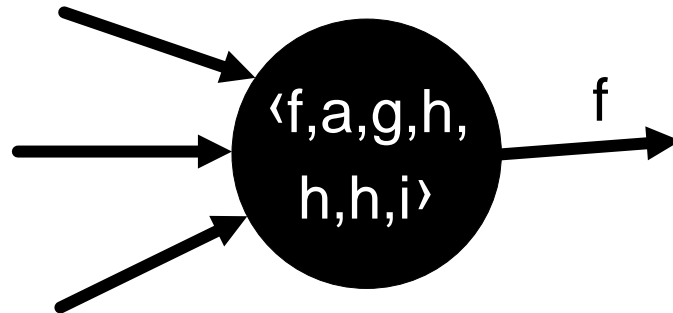
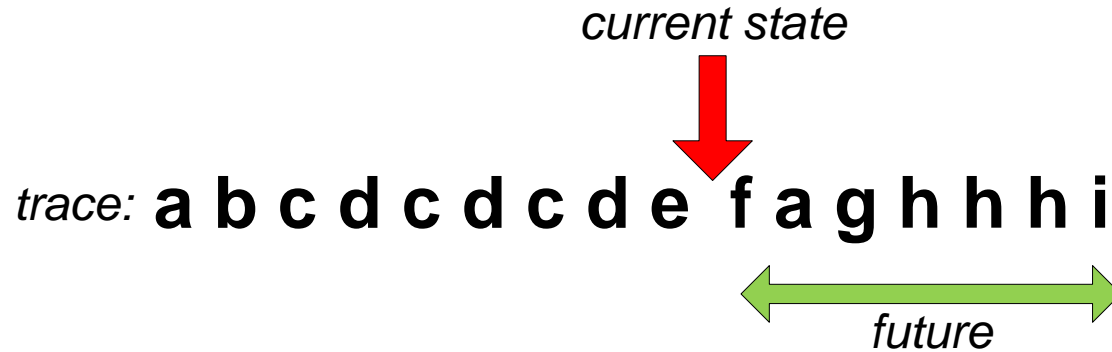


# Learning a Transition System

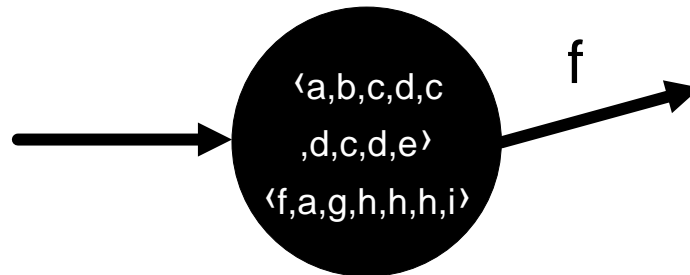
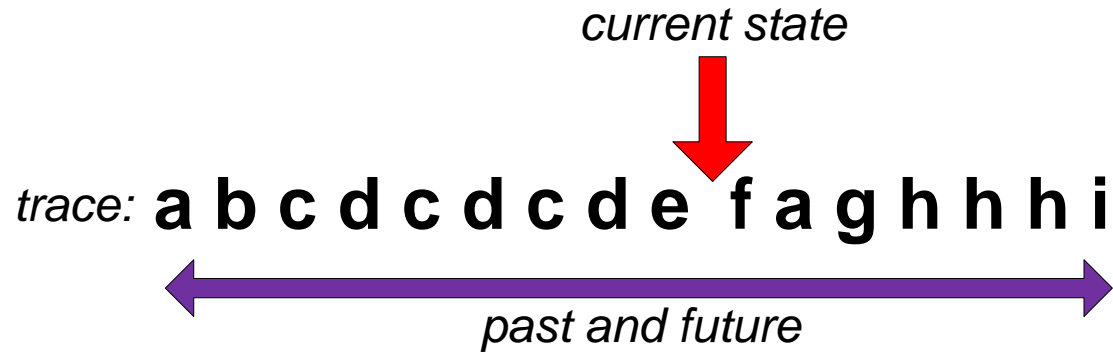
*current state*

*trace:* **a b c d c d c d e f a g h h h i**

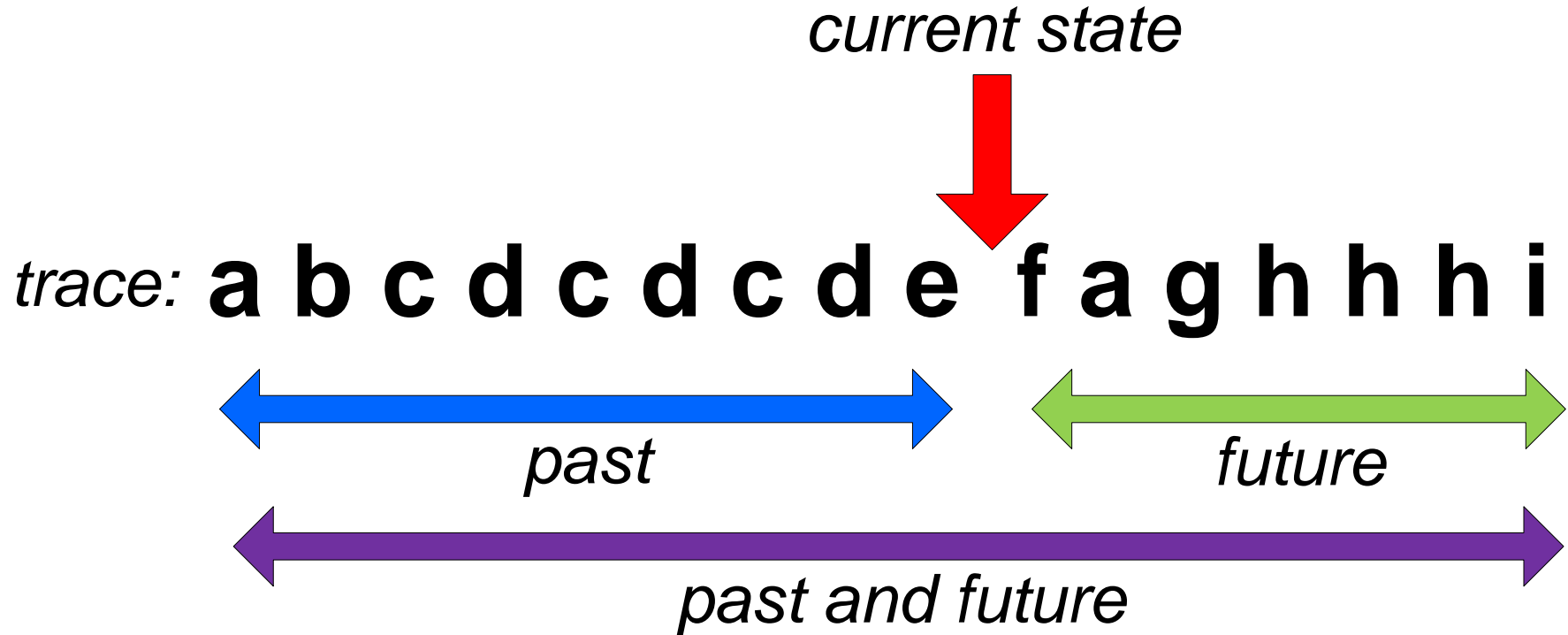
*future*



# Learning a Transition System

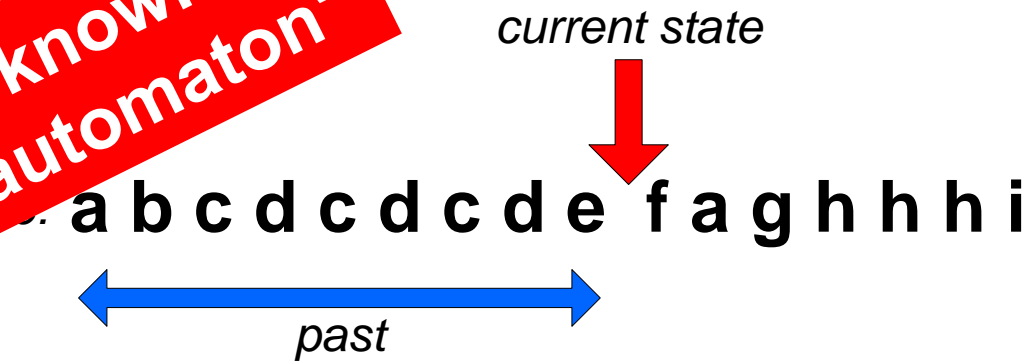


# Learning a Transition System

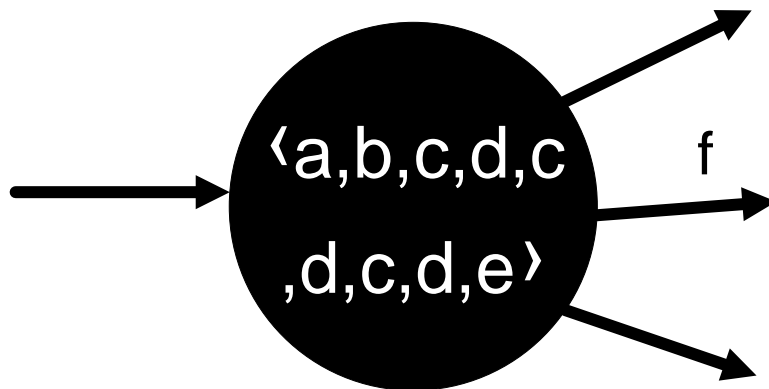


# Sequence selection based on past

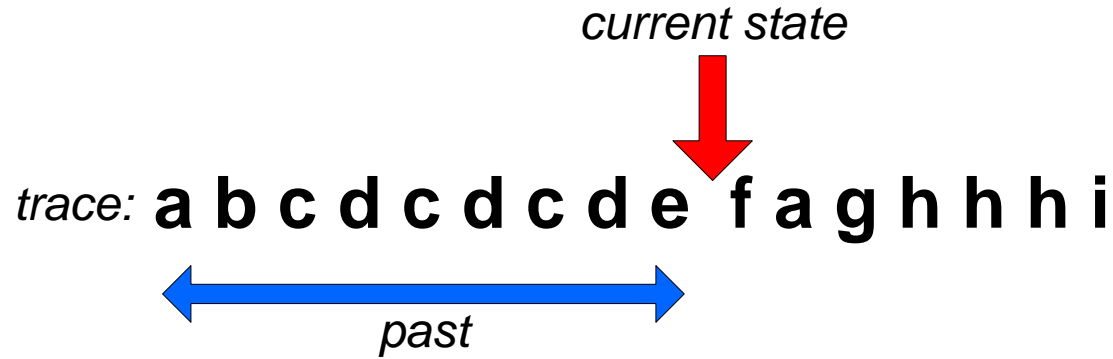
resulting transition system also known as the "prefix automaton"



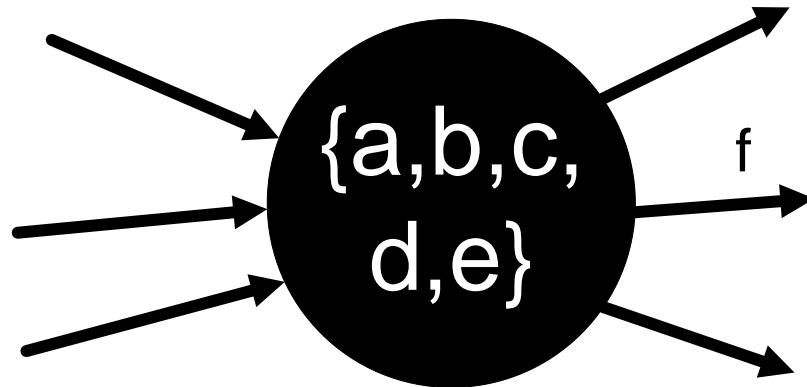
order and frequency matter



# Set abstraction based on past

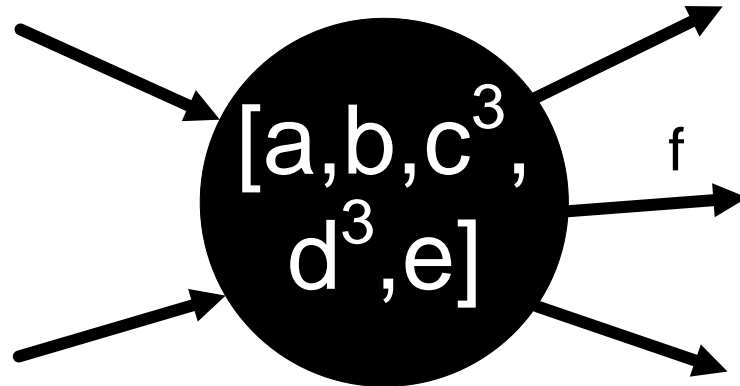
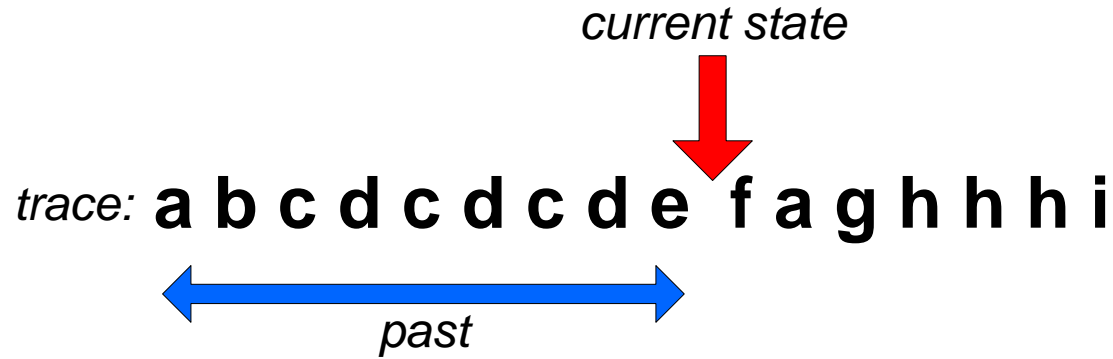


order and  
frequency  
don't matter





# Multiset abstraction based on past



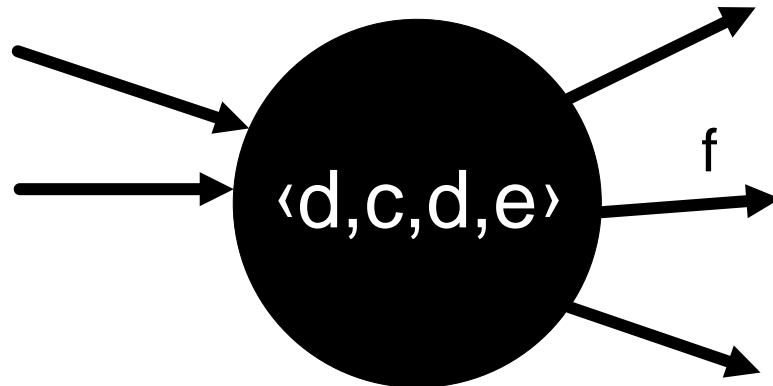
only  
frequency  
matters

# Sequence abstraction based on last 4 events

*current state*

*trace:* **a b c d c d c d e f a g h h h i**

*past 4 events*



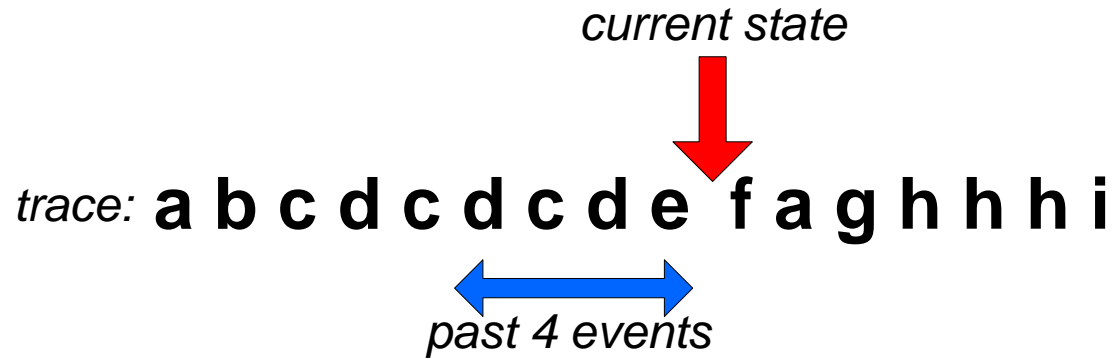
***k*-tail**

# Set abstraction based on last 4 events

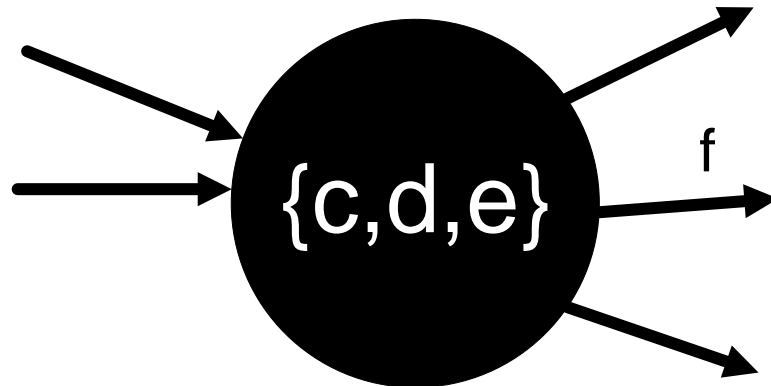
*current state*

*trace:* **a b c d c d c d e f a g h h h i**

*past 4 events*



The diagram shows a sequence of events: a, b, c, d, c, d, c, d, e, f, a, g, h, h, h, i. A red arrow points to the 'e' event, labeled 'current state'. A blue double-headed arrow spans the last four events 'c, d, c, d', labeled 'past 4 events'.

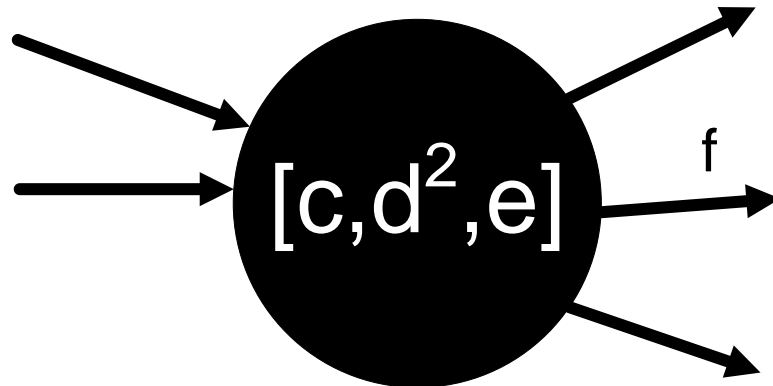
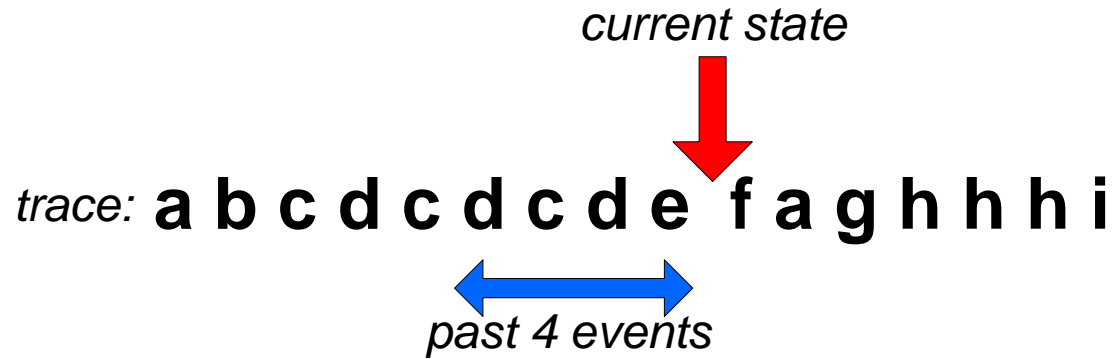


# Multiset abstraction based on last 4 events

*current state*

*trace:* **a b c d c d c d e f a g h h h i**

*past 4 events*

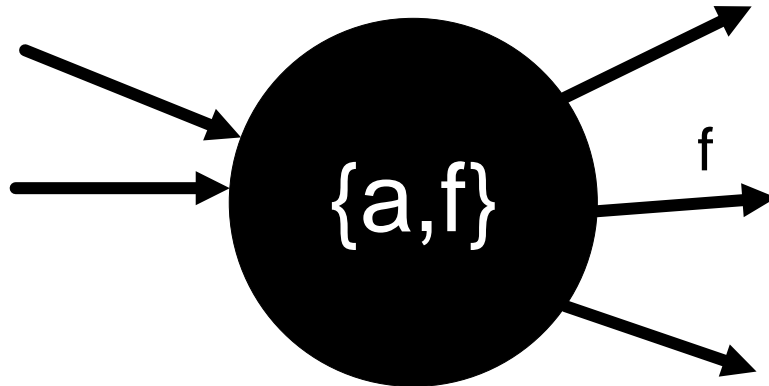


# Set abstraction based on the next 2 events

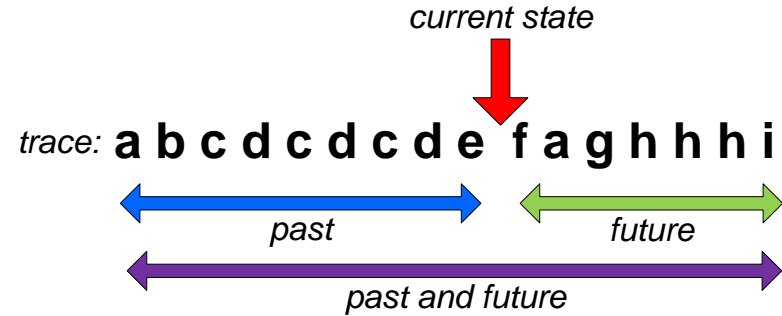
*current state*

*trace:* **a b c d c d c d e f a g h h h i**

*next two events*



# Summary

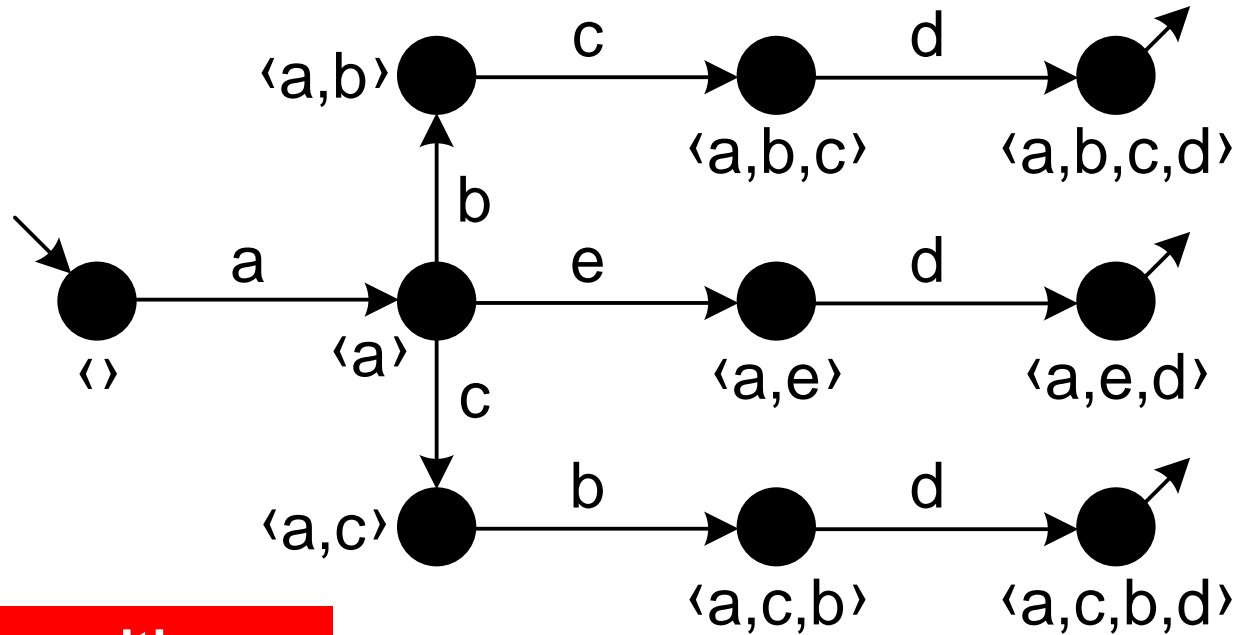


- Position in trace determines the current state.
- State based on **past**, **future**, or **past+future**.
- Sequence, multiset, set abstraction.
- Limited horizon to abstract further (e.g., *k*-tail).

# Example

$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

# Past without abstraction

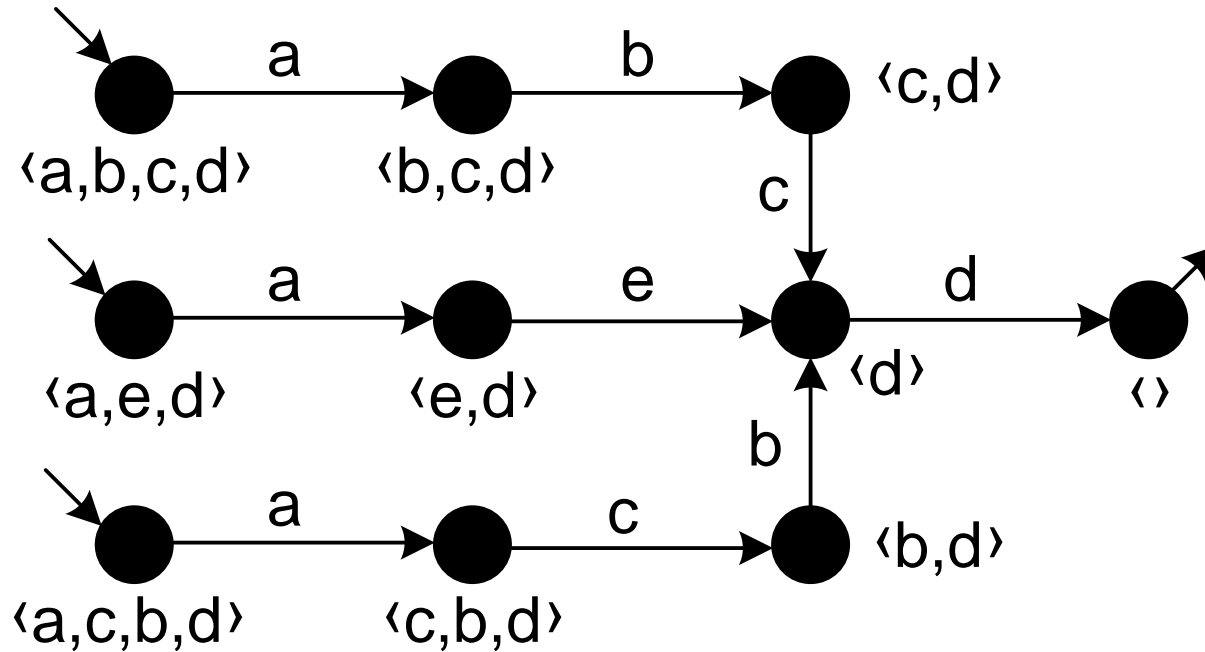


resulting transition  
system also known as  
the "prefix automaton"

$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

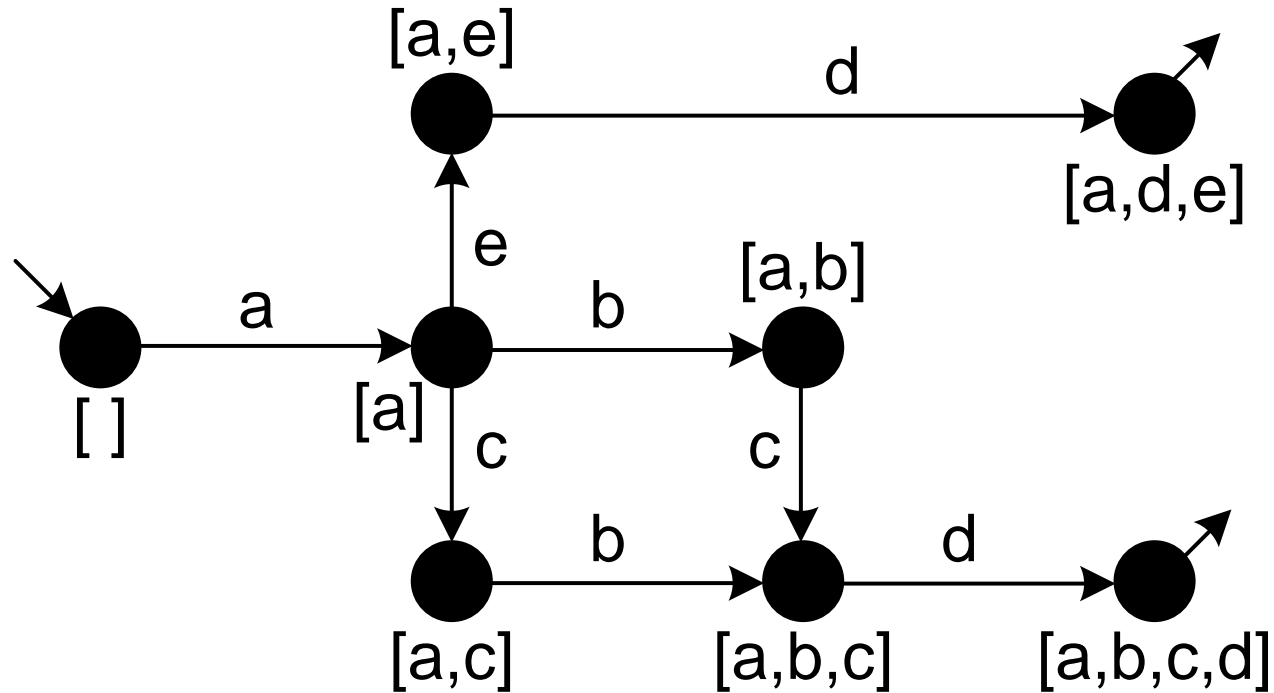


# Future without abstraction



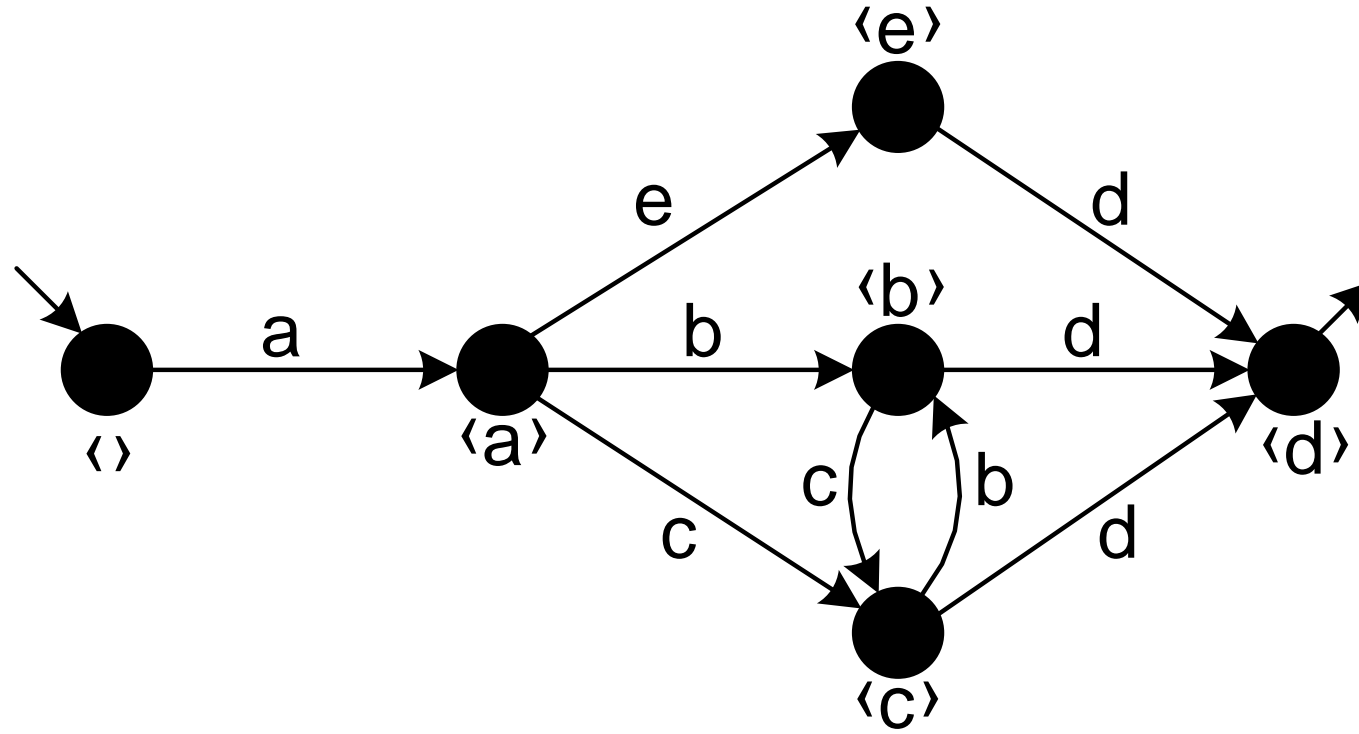
$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

# Past with multiset abstraction



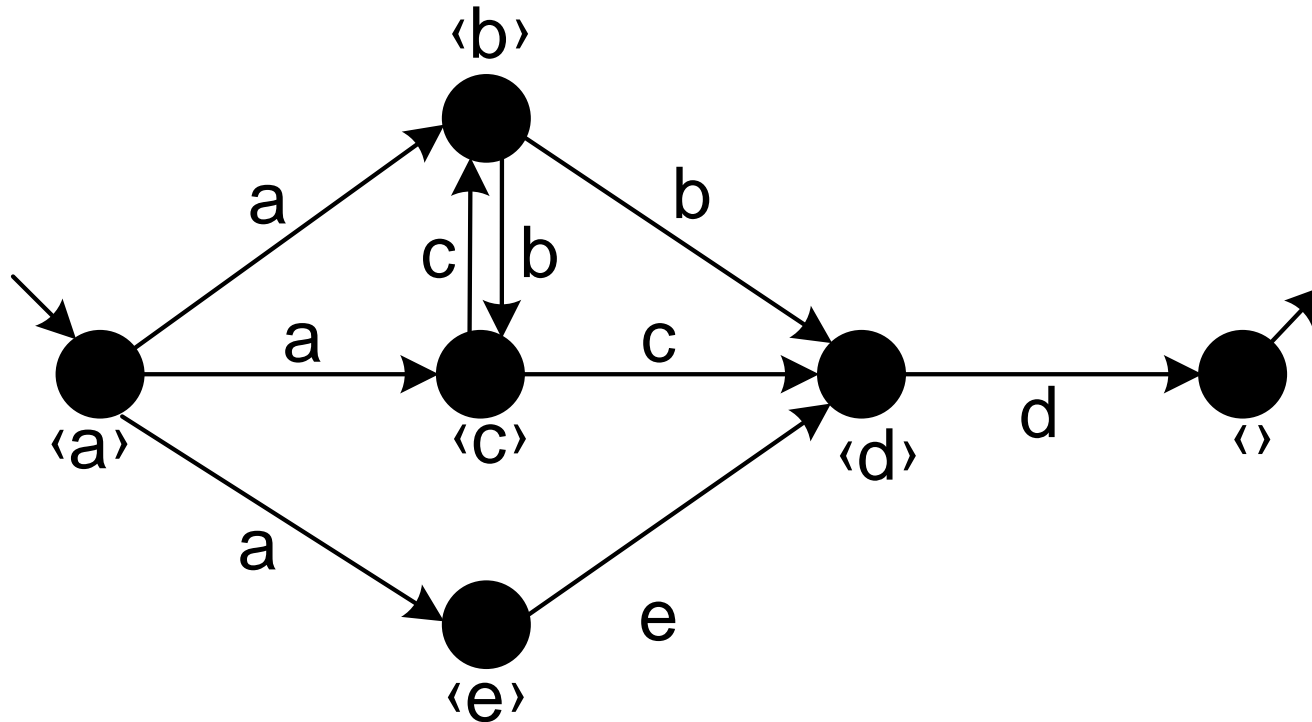
$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

# Only last event matters for state



$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

# Only next event matters for state

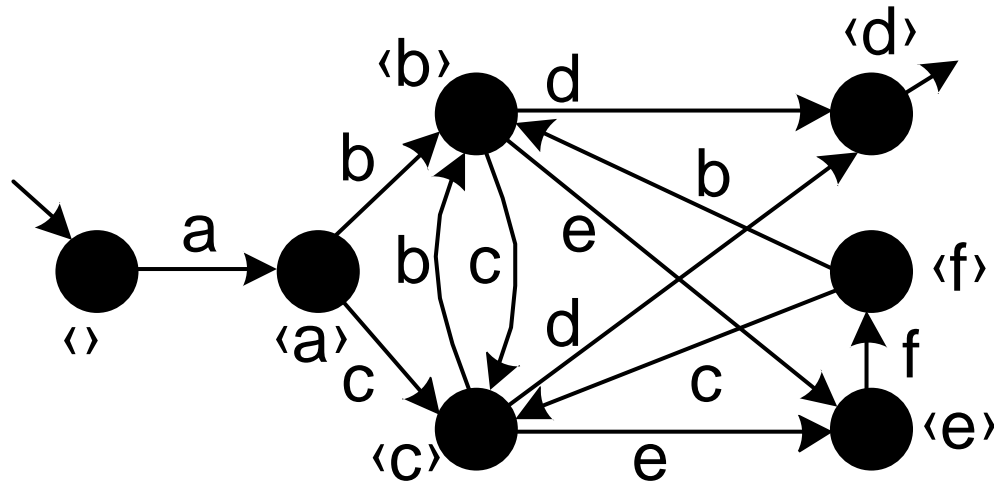


$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

# Question:

## What kind of abstraction was used?

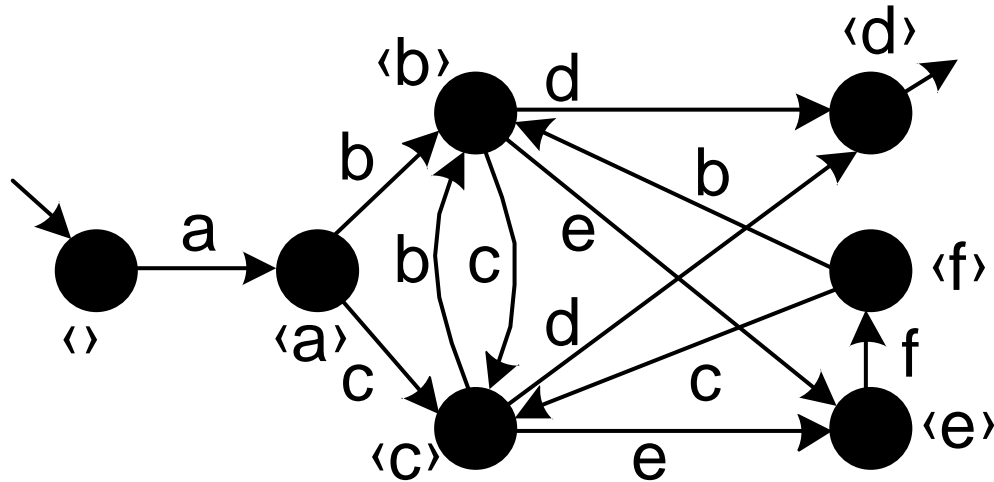
$$L_2 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle, \\ \langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle]$$



# Answer:

## Only last event matters

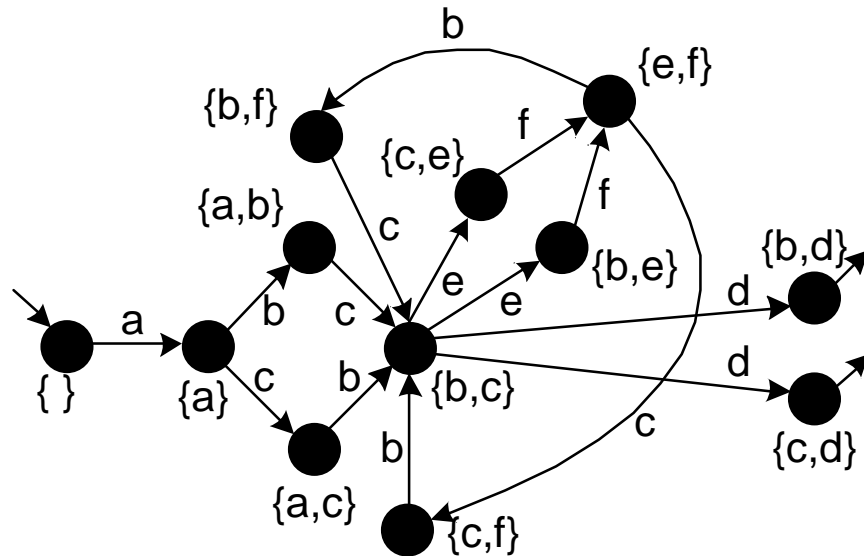
$$L_2 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle, \\ \langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle]$$



# Question:

## What kind of abstraction was used?

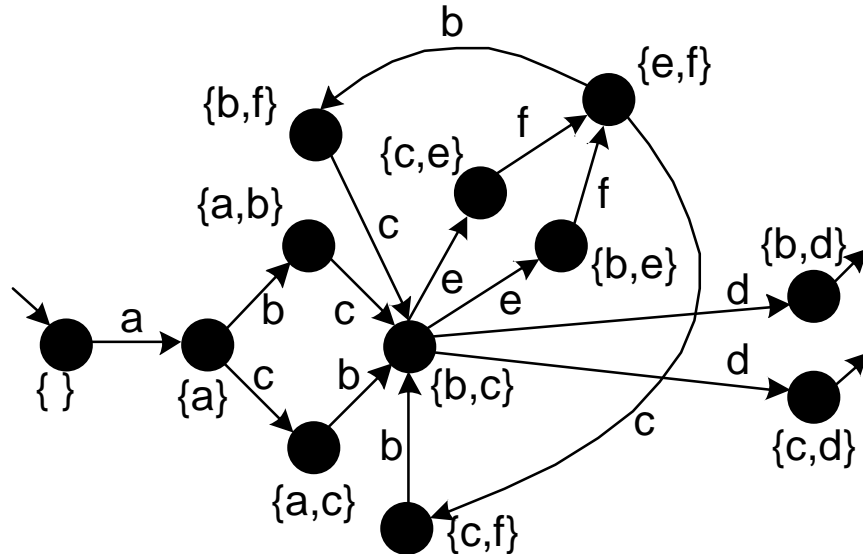
$$L_2 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle, \\ \langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle]$$



# Answer:

## Only last two events matters (set abstraction)

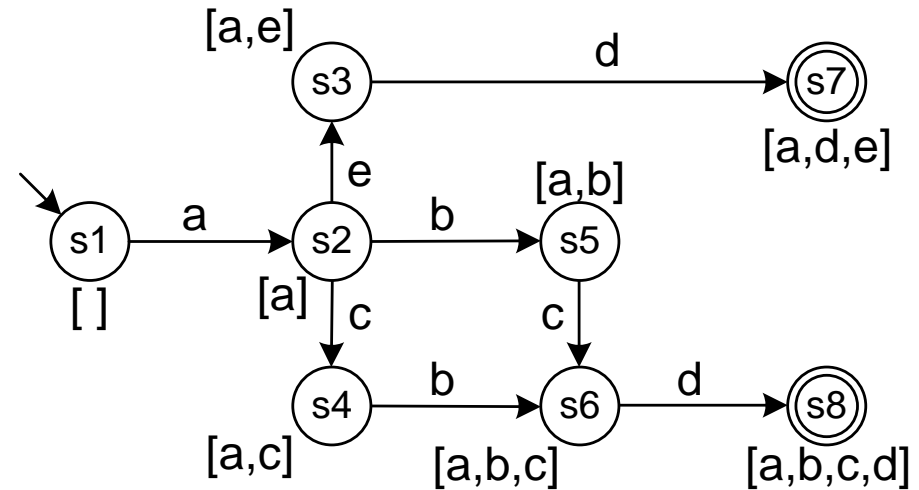
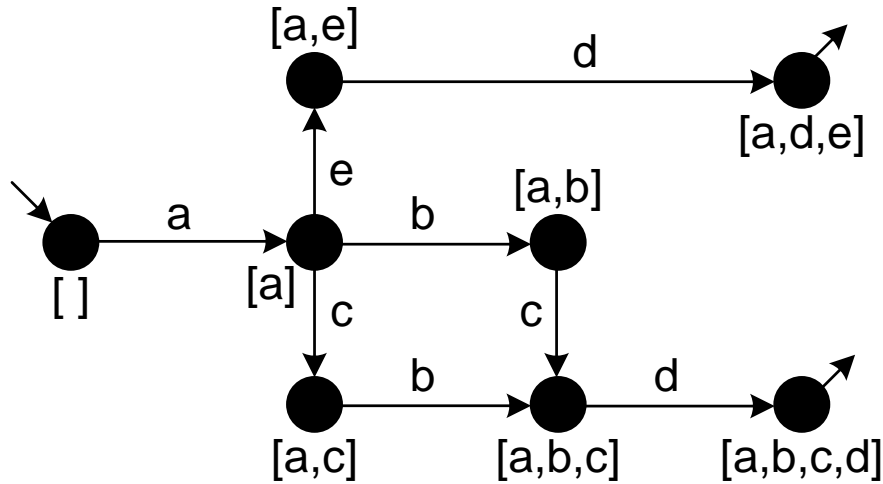
$$L_2 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle, \\ \langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle]$$



**Last two events  
are considered  
without looking  
at the order.**



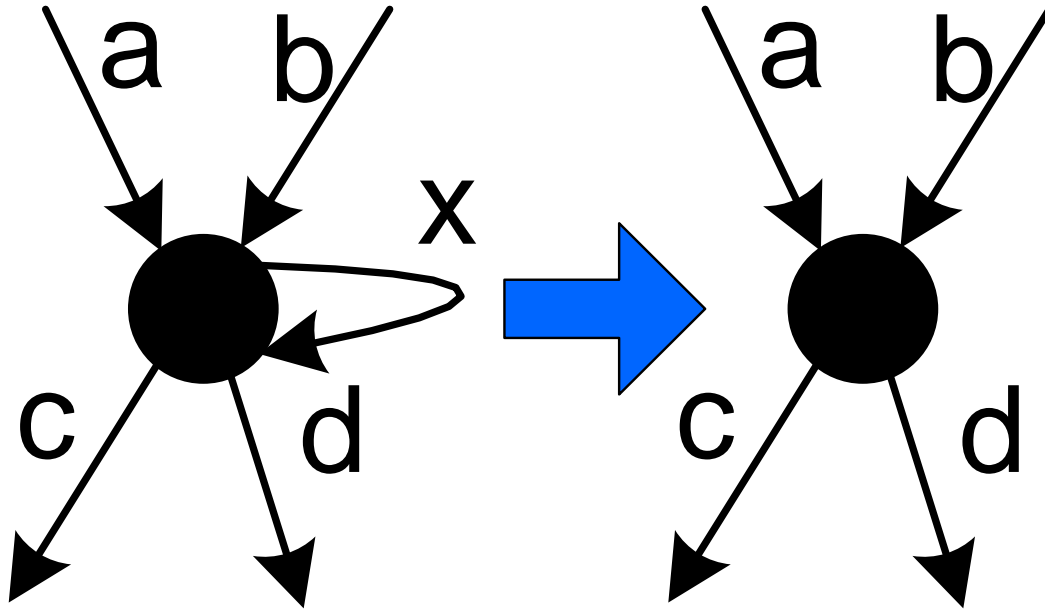
# Remark about notation



# Extensions and variations

- Events may have **other attributes** that can be used:
  - States based on resources or data elements rather than activities.
  - Transition names based on resources.
- Filtering of **infrequent paths**.
- Filtering of **infrequent activities**.

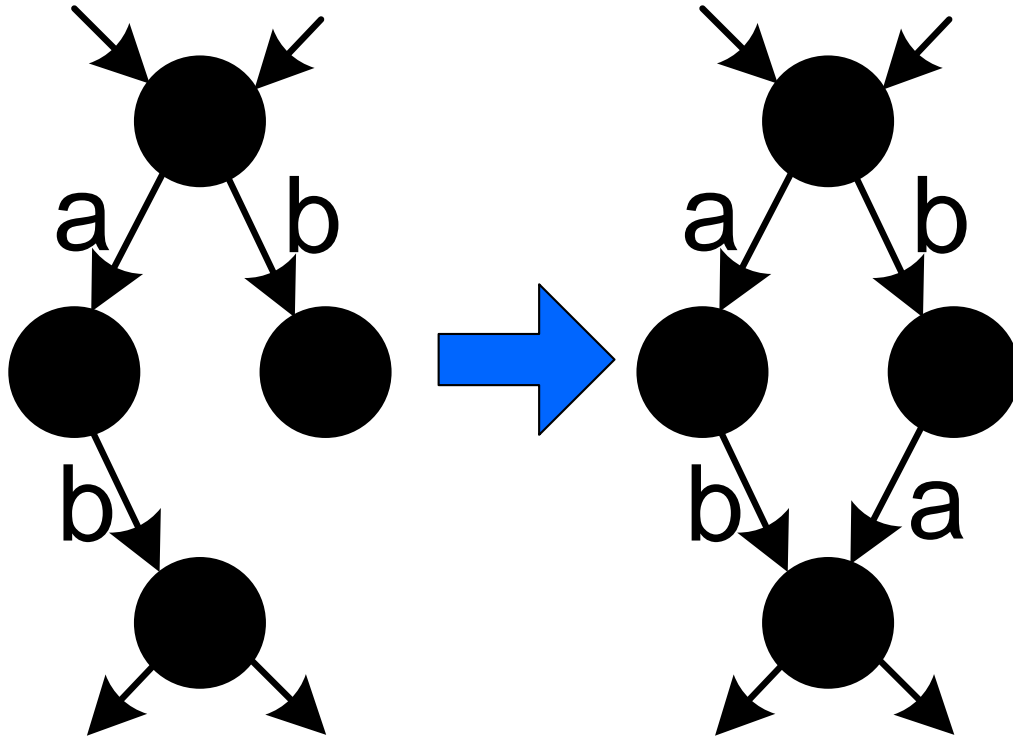
# Postprocessing of transition system



**removing  
self loops**

**(will jeopardize  
fitness)**

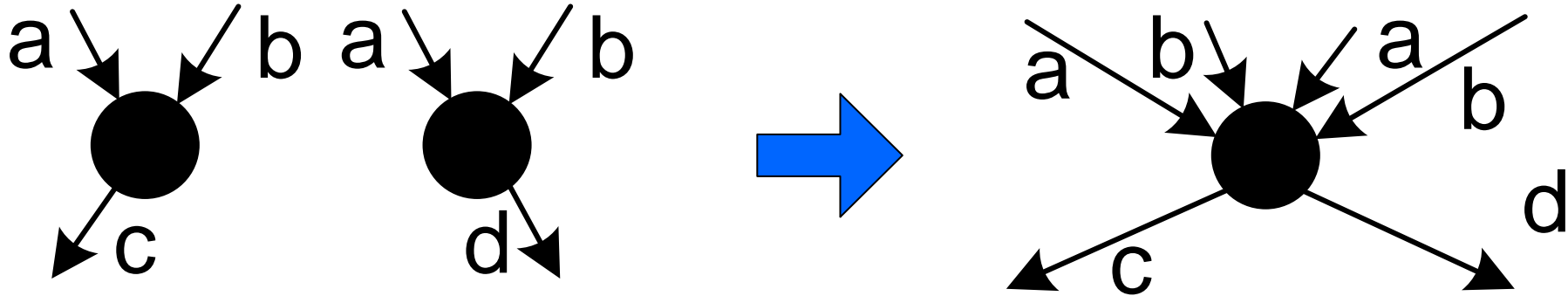
# Postprocessing of transition system



**improve  
diamond  
structure**

**(for missing  
interleavings)**

# Postprocessing of transition system



**merge similar states**  
(e.g., based on inputs)

# Mine Transition System plug-in of ProM

The image displays the ProM 6.10.0 User Interface (UITopia) with the 'Mine Transition System' plug-in selected in the 'Actions' panel. The 'Input' panel shows 'L1.xml' and an 'Event Log' icon. The 'Output' panel is empty. The 'TS Miner' configuration window is open, showing the 'Configure key classifiers' section. This section has two columns: 'Select backward keys' and 'Select forward keys'. Both columns list 'MXML Legacy Classifier', 'Event Name', and 'Resource'. The 'Event Name' classifier is selected in the backward keys list. At the bottom of the configuration window, there is a checkbox for 'Select key data attributes' and buttons for 'Cancel', 'Previous', and 'Next'. The 'Next' button is highlighted. The background shows the 'Introduction' screen of the 'TS Miner' plug-in.

ProM 6.10.0

Actions

Input

L1.xml  
Event Log

Filter: [icon] [icon] [search...]

Actions

- Mine for a Reassignment Social Network  
M. Song (m.song@unist.ac.kr)
- Mine for a Similar-Task Social Network  
M. Song (m.song@unist.ac.kr)
- Mine for a Subcontracting Social Network  
M. Song (m.song@unist.ac.kr)
- Mine for a Working-Together Social Network  
M. Song (m.song@unist.ac.kr)
- Mine Heuristic Net using Genetic Miner  
A.K. Alves de Medeiros (c.c.alves@tue.nl)
- Mine SPD Model  
B.F. van Dongen (b.f.v.dongen@tue.nl)
- Mine Transition System**  
H.M.W. Verbeek (h.m.w.verbeek@tue.nl)
- Rename/Merge Events  
J. Claes (jan.claes@ugent.be)
- Replay a Case on Causal net for Conformance Analysis  
Arya Adrianyah (a.adrianyah@tue.nl)
- Replay a Case on CPF Model for Conformance Analysis  
H.V. Nguyen (h.v.nguyen@student.tue.nl)
- Replay a Case on Petri Net for Conformance Analysis  
Arya Adrianyah (a.adrianyah@tue.nl)

Reset Start

Output

TS Miner

Introduction

Activity

TS Miner

Configure key classifiers

Select backward keys

- MXML Legacy Classifier
- Event Name
- Resource

Select forward keys

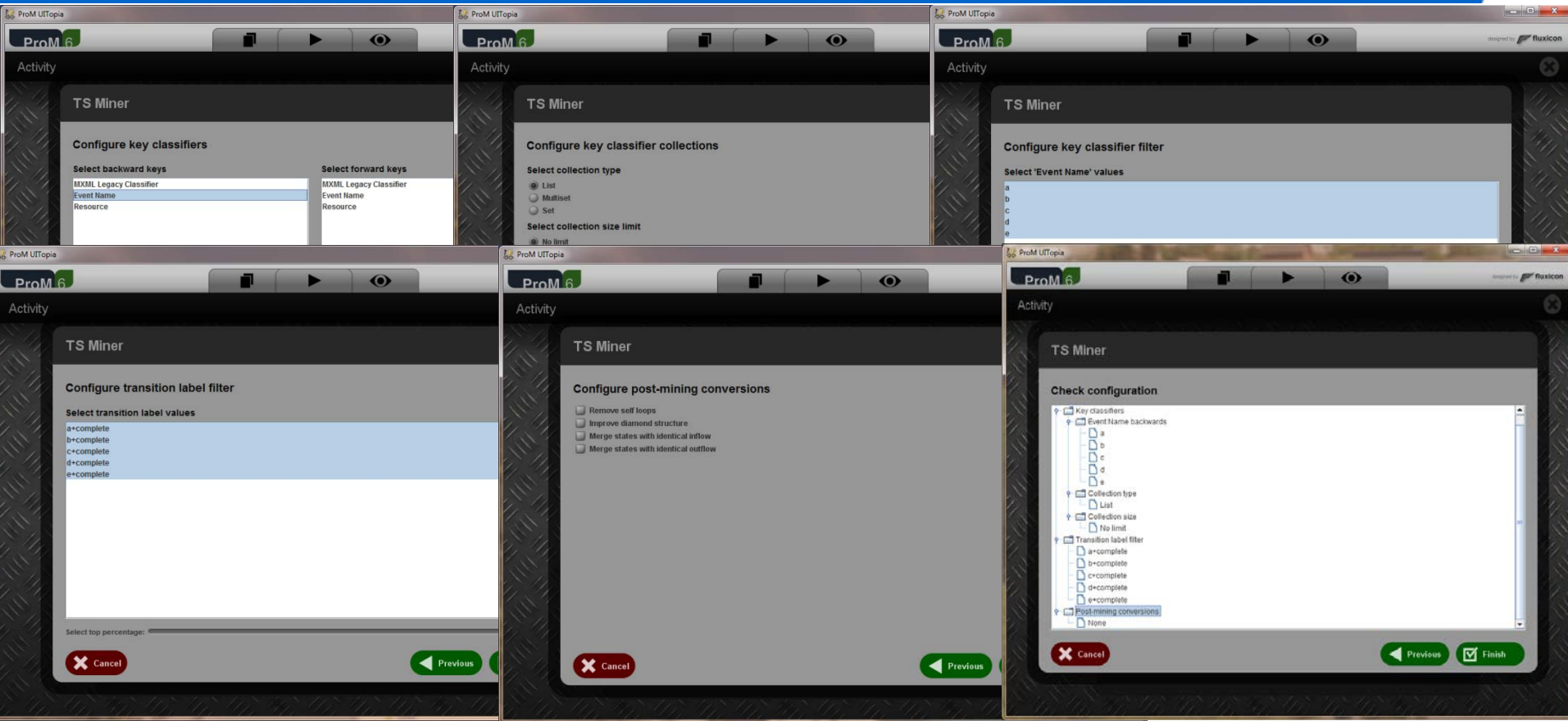
- MXML Legacy Classifier
- Event Name
- Resource

☐ Select key data attributes

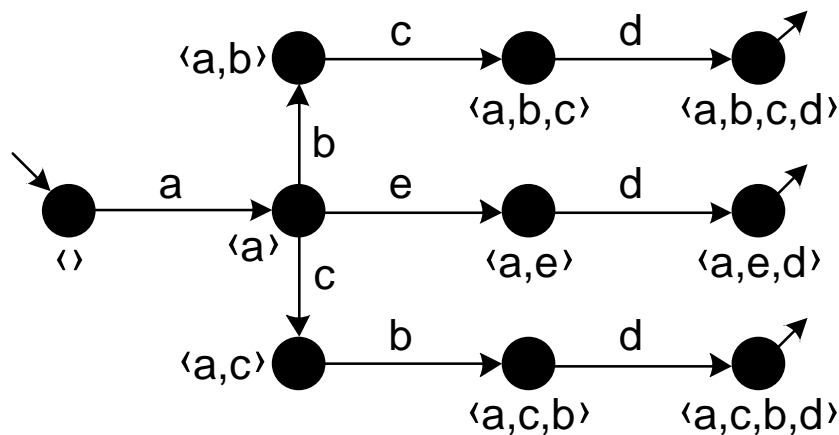
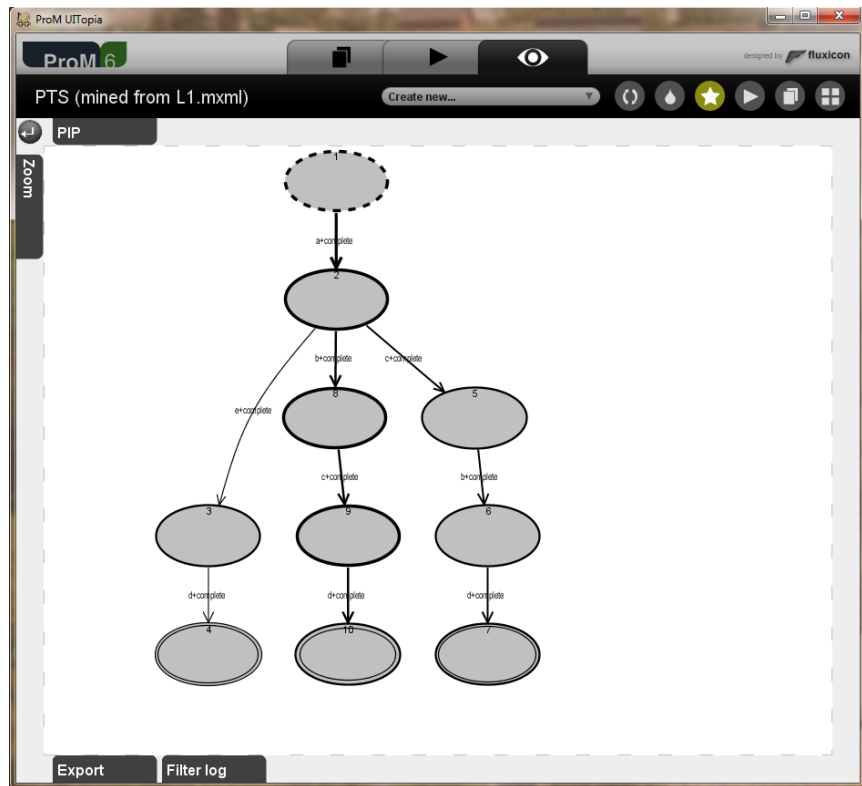
Cancel Previous Next

TU/e

# Highly configurable

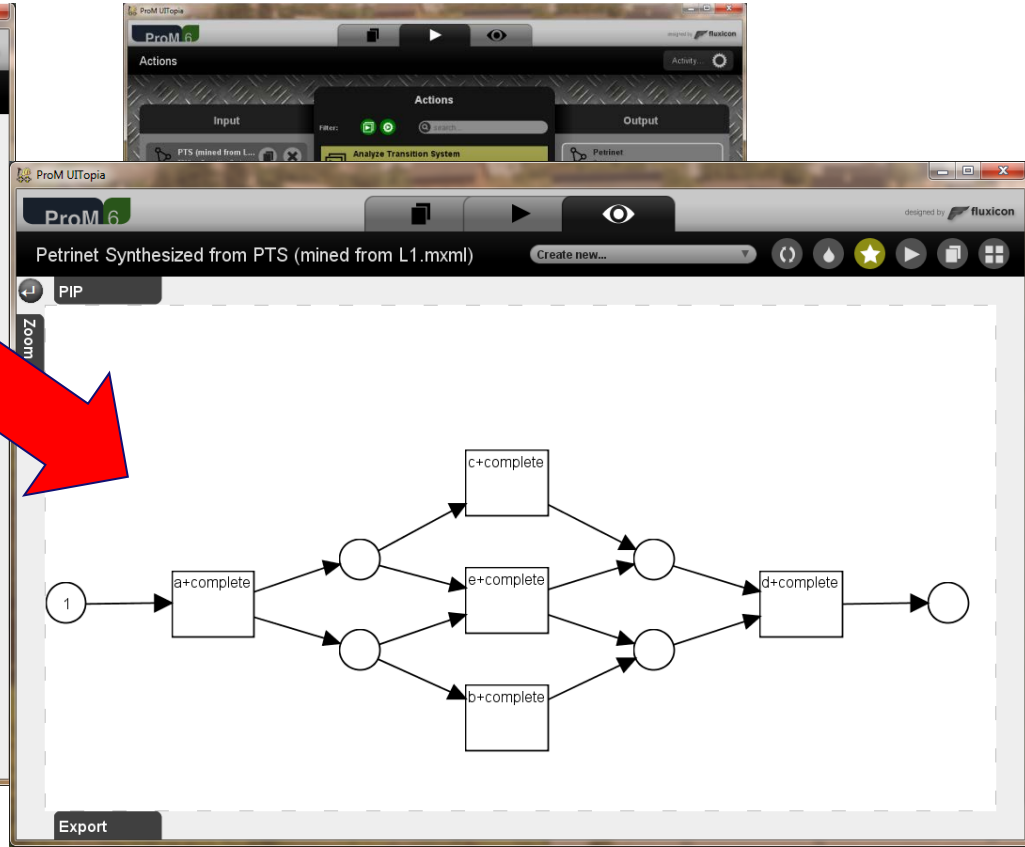
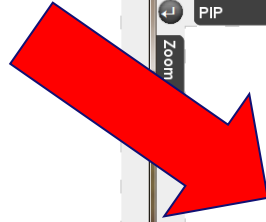
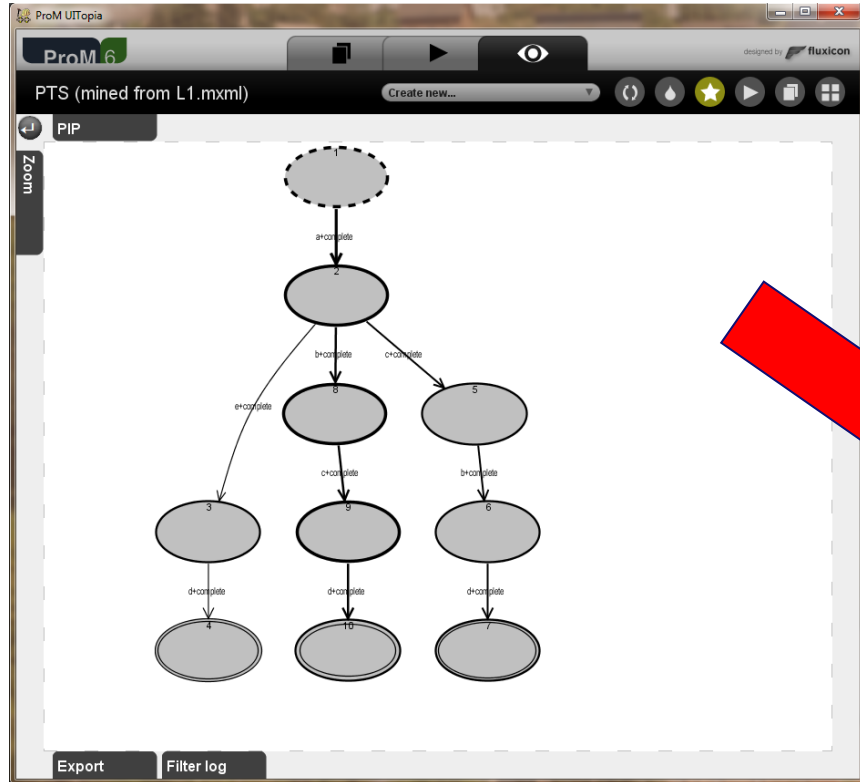


# Output





# Next: state-based regions



### *Part I: Preliminaries*

**Chapter 1**  
Introduction

**Chapter 2**  
Process Modeling and  
Analysis

**Chapter 3**  
Data Mining

### *Part III: Beyond Process Discovery*

**Chapter 7**  
Conformance  
Checking

**Chapter 8**  
Mining Additional  
Perspectives

**Chapter 9**  
Operational Support

### *Part II: From Event Logs to Process Models*

**Chapter 4**  
Getting the Data

**Chapter 5**  
Process Discovery: An  
Introduction

**Chapter 6**  
Advanced Process  
Discovery Techniques

### *Part IV: Putting Process Mining to Work*

**Chapter 10**  
Tool Support

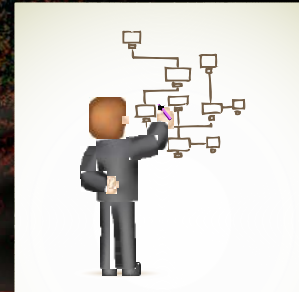
**Chapter 11**  
Analyzing “Lasagna  
Processes”

**Chapter 12**  
Analyzing “Spaghetti  
Processes”

### *Part V: Reflection*

**Chapter 13**  
Cartography and  
Navigation

**Chapter 14**  
Epilogue



Wil M. P. van der Aalst

# Process Mining

Discovery, Conformance and  
Enhancement of Business Processes

 Springer