Christian-Albrechts-Universität zu Kiel

Advanced Process Mining

Prof. Dr. Agnes Koschmider

Lecture 8: Event Log Clustering

Process Analytics

Lecture Overview

Christian-Albrechts-Universität zu Kiel

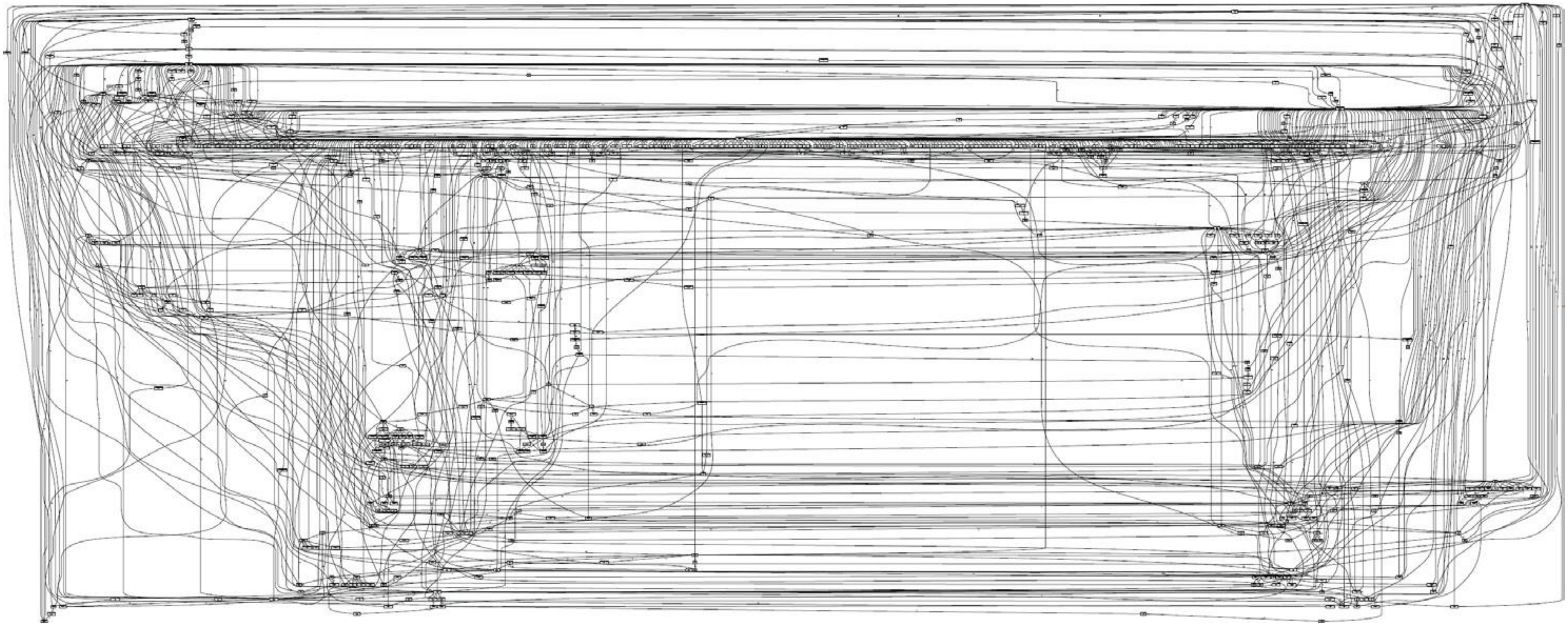
Motivation

- The major application of process mining
  - Discovery → extraction of abstract process knowledge from event logs

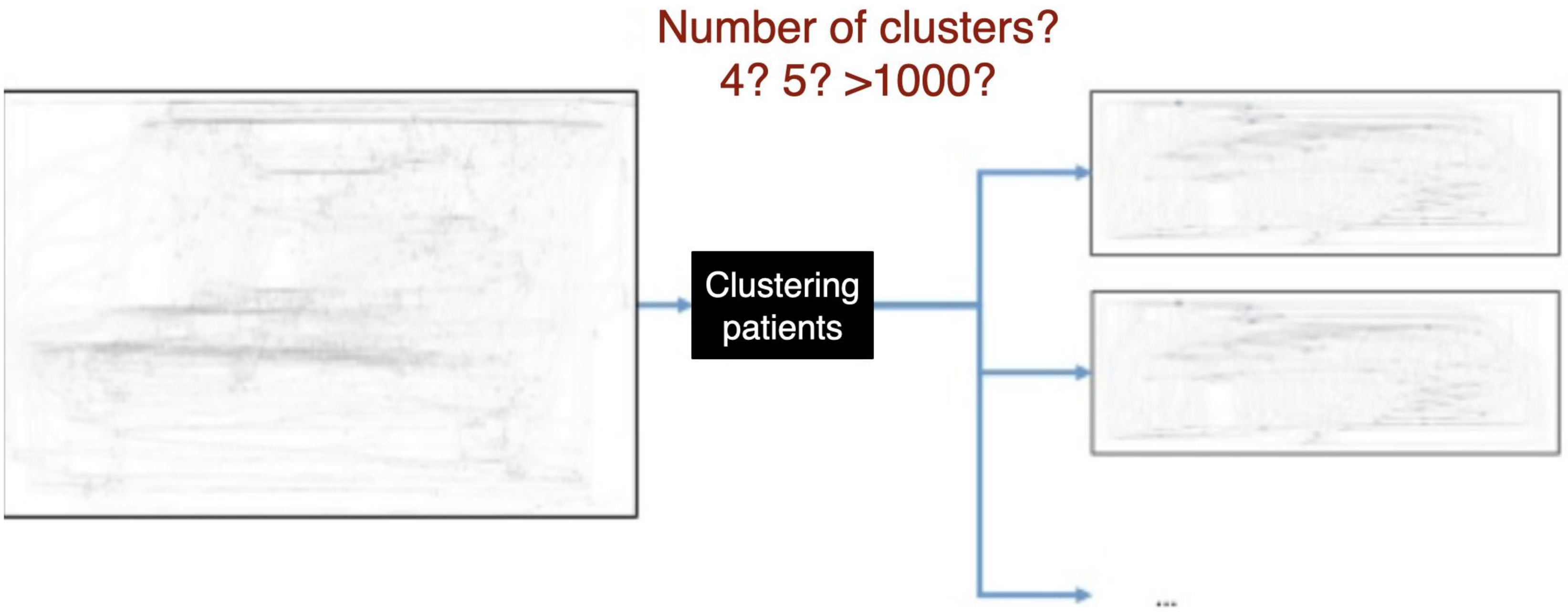
- Real-life business processes are <u>flexible</u>
  - Spaghetti model
  - Single cases differ significantly from one another = ‚Diversity'
  - Discovering actual process which is being executed is valuable.

- Solution for diversity of cases
  - Measure the similarity of cases and use the information to divide the set of cases into more homogeneous subsets
  - Trace clustering

Motivation for Clustering

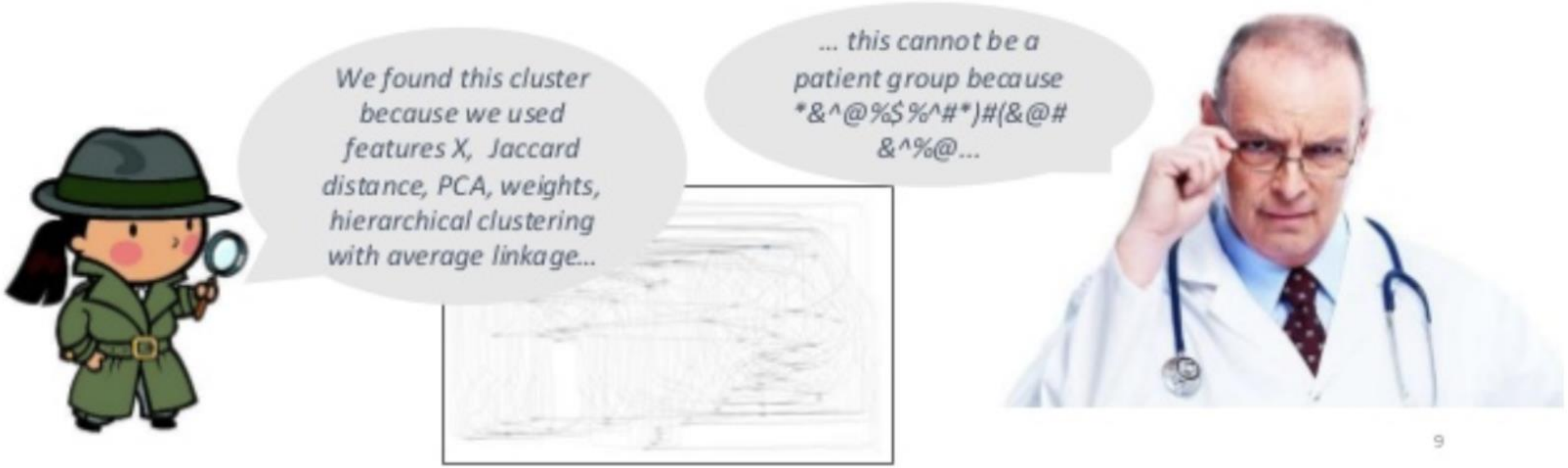


Challenge 1 – Unknown Number of Clusters



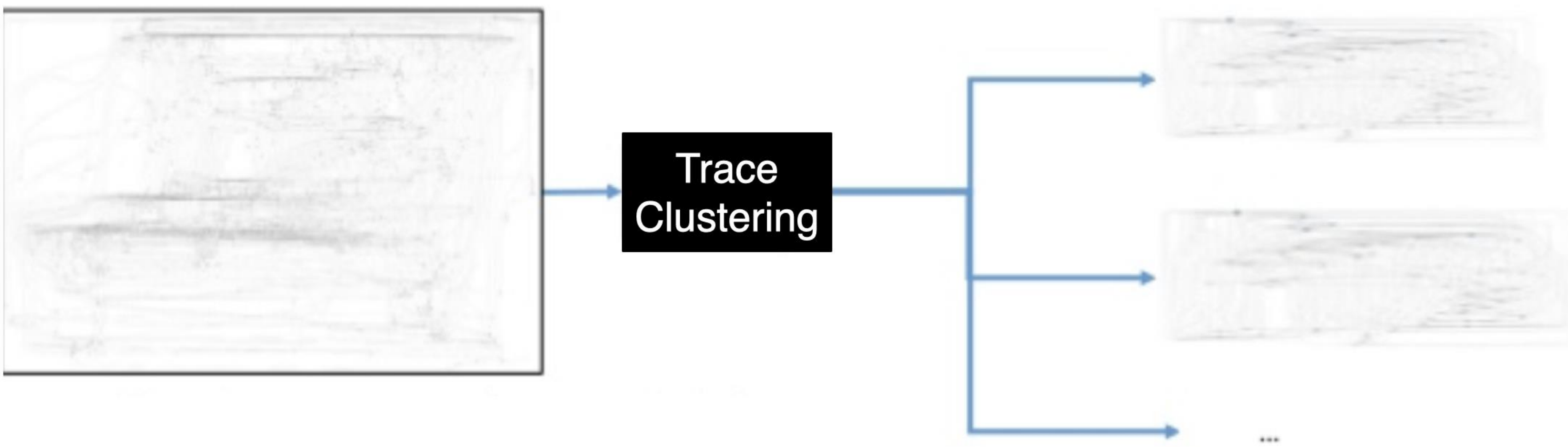
Challenge 2 - Quality of Clusters

- Highly dependent on domain/medical knowledge
- Difficult to convince or be used by domain experts



Trace Clustering

Christian-Albrechts-Universität zu Kiel



- Handle very complex event data
- Handle unknown number of clusters
- Incorporate and leverage domain knowledge

Running Example

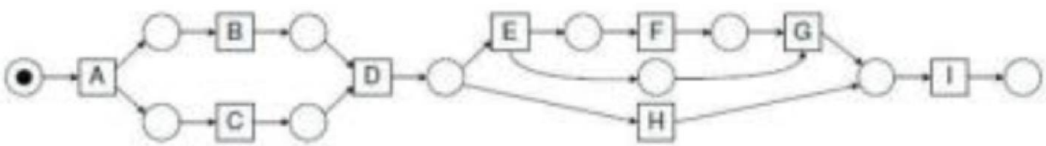


Fig. 1. The example process model

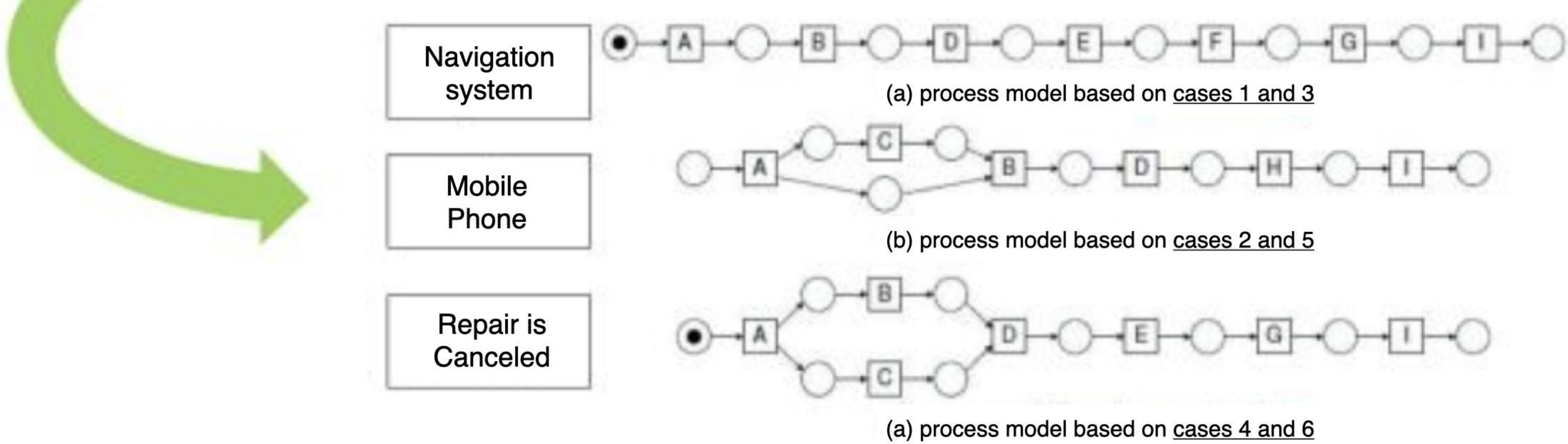


Fig. 2. The derived process models from three groups

- Trace clustering can support the identification of process variants corresponding to homogenous subsets of cases

Trace Profiles(1)

Christian-Albrechts-Universität zu Kiel

- In the trace clustering approach, each case is characterized by a defined set of items, i.e., specific features which can be extracted from the corresponding trace.
- Items for comparing traces are organized in trace profiles, each addressing a specific perspective of the log

Trace Profiles(2)

- Profile
  - A set of related items which describe the trace from a specific perspective

- Every item is a metric → we can consider a profile with n items to be a function, which assigns to a trace a vector ($i_1, i_2, \ldots i_n$)

- Profiling a log can be described as measuring a set of traces with a number of profiles, resulting in an aggregate vector
  - Resulting vectors can subsequently be used to calculate the distance between any two traces, using a *distance metric*

Trace Profiles(3)

Christian-Albrechts-Universität zu Kiel

Case ID	log events
1	(A,John) , (B,Mike) , (D,Sue) , (E,Pete) , (F,Mike) , (G,Jane) , (I,Sue)
2	(A,John) , (B,Fred) , (C,John) , (D,Clare) , (E,Robert) , (G,Mona) , (I,Clare)
3	(A,John) , (B,Pete) , (D,Sue) , (E,Mike) , (F,Pete) , (G,Jane) , (I,Sue)
4	(A,John) , (C,John) , (B,Fred) , (D,Clare) , (H,Clare) , (I,Clare)
5	(A,John) , (C,John) , (B,Fred) , (D,Clare) , (H,Clare) , (I,Clare)
6	(A,John) , (B,Mike) , (D,Sue) , (H,Sue) , (I,Sue)

Table 1. Example process logs (A: Receive a item and repair request, B: Check the item, C: Check the warranty, D: Notify the customer, E: Repair the item, F: Test the repaired product, G: Issue payment, H: send the cancellation letter, I: Return the item

Table 2. Activity and originator profiles for the example log from Table 1.

Case ID	Activity Profile									Originator Profile								
	A	B	C	D	E	F	G	H	I	John	Mike	Sue	Pete	Jane	Fred	Clare	Robert	Mona
1	1	1	0	1	1	1	1	0	1	1	2	2	1	1	0	0	0	0
2	1	1	1	1	1	0	1	0	1	2	0	0	0	0	1	2	1	1
3	1	1	0	1	1	1	1	0	1	1	1	2	2	1	0	0	0	0
4	1	1	1	1	0	0	0	1	1	2	0	0	0	0	1	3	0	0
5	1	1	1	1	1	0	1	1	0	2	0	0	0	0	1	2	2	0
6	1	1	0	1	0	0	0	1	1	1	1	3	0	0	0	0	0	0

Clustering Methods – Distance Measures

Christian-Albrechts-Universität zu Kiel

- Distance Measures
  → To calculate the similarity between cases

- Three distance measures
  - Euclidean distance(c_j,c_k) = $\sqrt{\sum_{l=1}^{n} |i_{jl} - i_{kl}|^2}$
  - Hamming distance(c_j,c_k) = $\sum_{l=1}^{n} \delta(i_{jl}, i_{kl})/n$

  $$\text{where } \delta(x,y) = \begin{cases} 0 \text{ if } (x > 0 \ \wedge \ y > 0) \vee (x = y = 0) \\ 1 \text{ otherwise} \end{cases}$$

  - Jaccard distance(c_j,c_k) = $1 - (\sum_{l=1}^{n} i_{jl} i_{kl}) / (\sum_{l=1}^{n} i_{jl}^2 + \sum_{l=1}^{n} i_{kl}^2 - \sum_{l=1}^{n} i_{jl} i_{kl})$

→ *n*: the number of items extracted from the process log
→ Case c_j: corresponds to the vector (i_{j1}, i_{j2}, … i_{jn})
→ i_{jk}: the number of appearance of item *k* in the case *j*

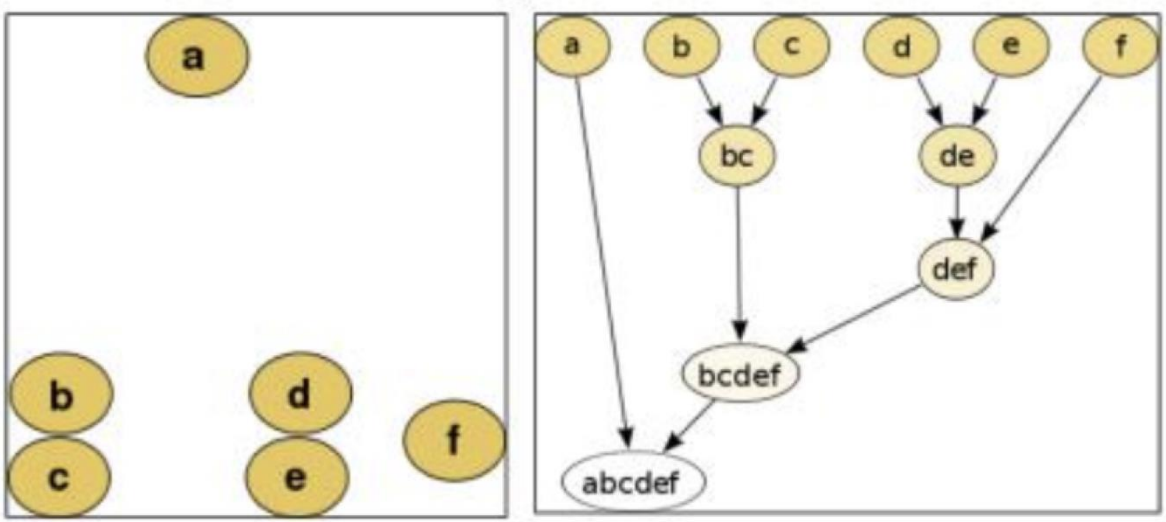
K-means

Christian-Albrechts-Universität zu Kiel

- K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data(i.e., data without defined categories or groups)
- The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*(user defined)
- The algorithm works iteratively to assign each data point to one K groups based on the features that are provided
- Data points are clustered based on feature similarity(distance)

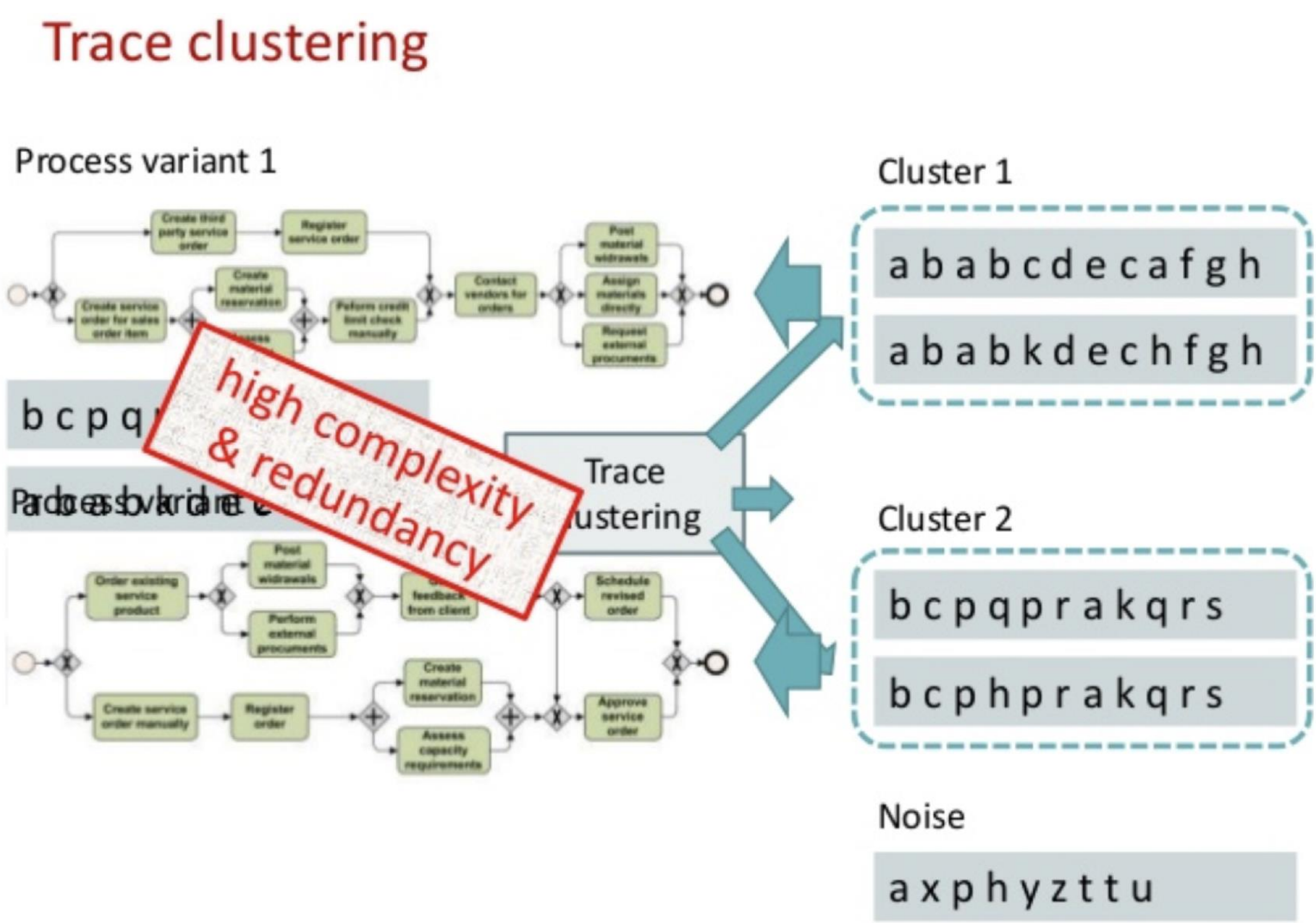
Agglomerative Clustering

- Agglomerative hierarchical clustering
  - Gradually generate clusters by merging nearest traces
  - Smaller clusters are merged into large ones
  - Example: we have six elements {a} {b} {c} {d} {e} and {f}. The first step is to determine which elements to merge in a cluster. Usually, we want to take the two closest elements, according to the chosen distance.

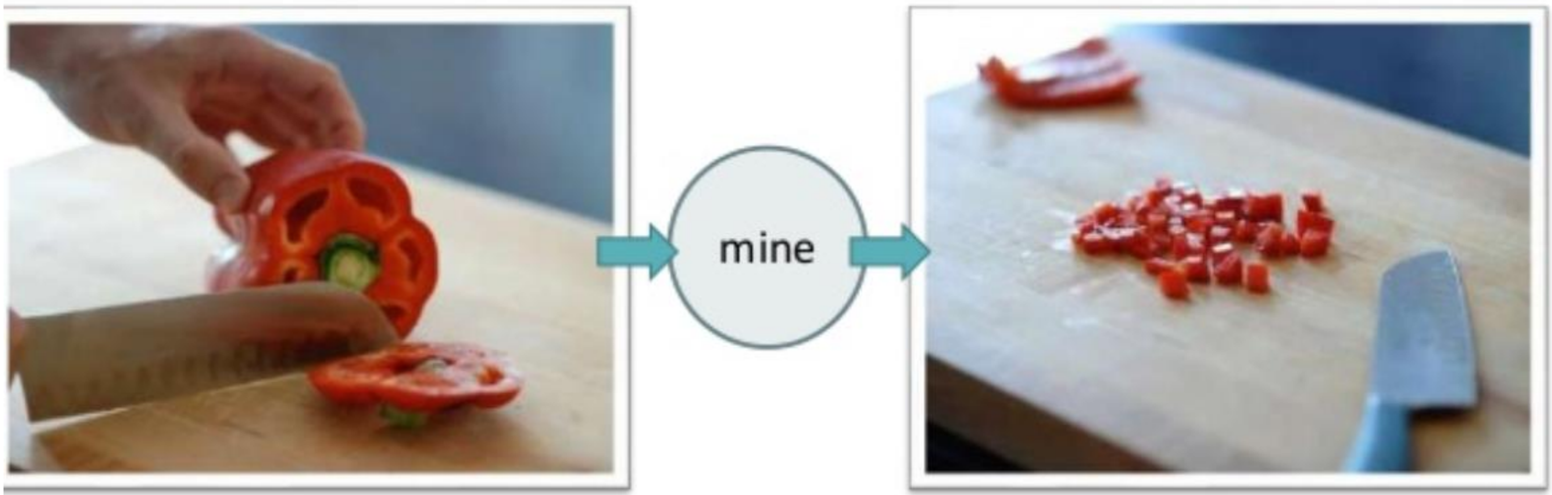


Bottlenecks of Trace Clustering

Christian-Albrechts-Universität zu Kiel



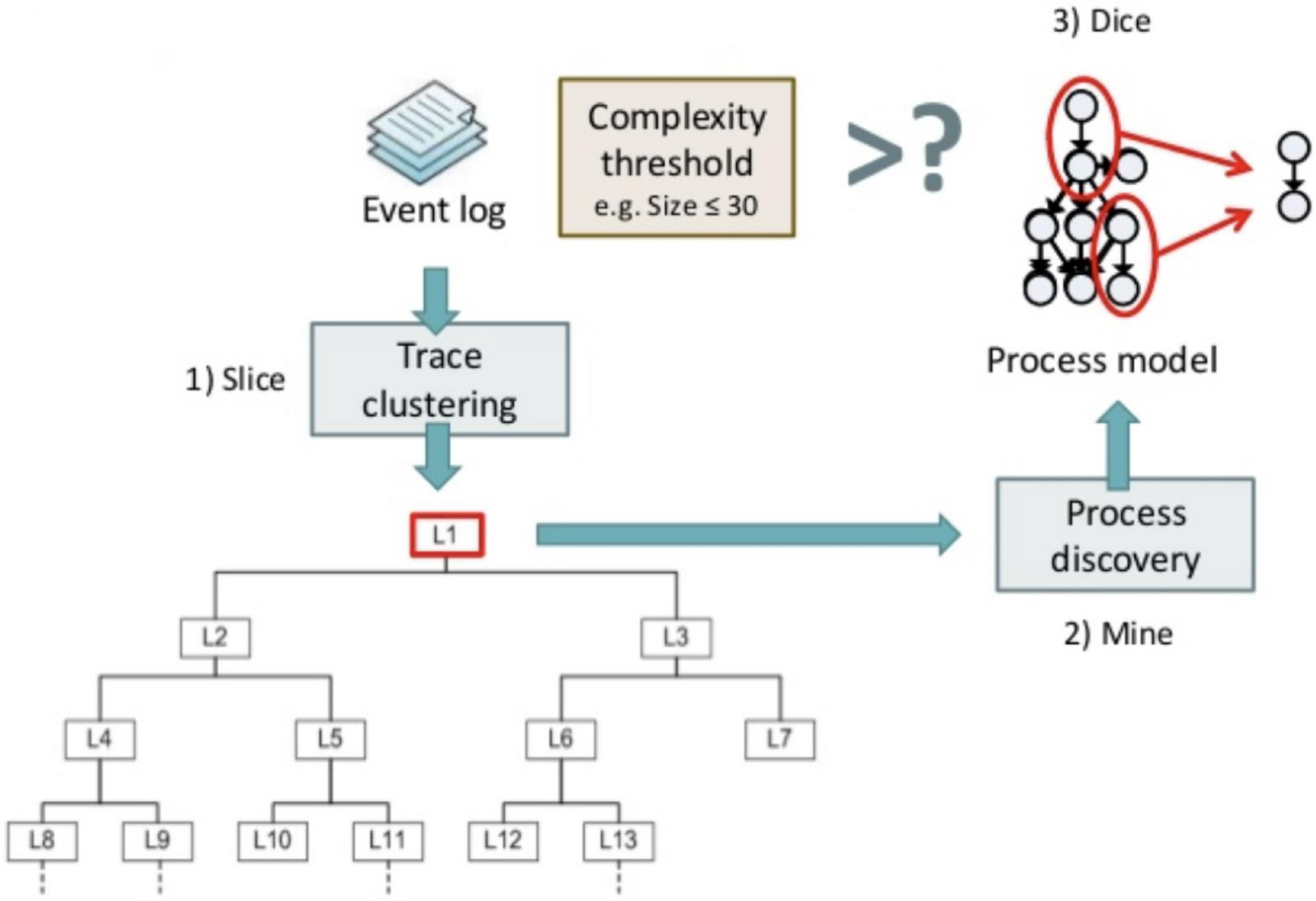
Slice, Mine & Dice



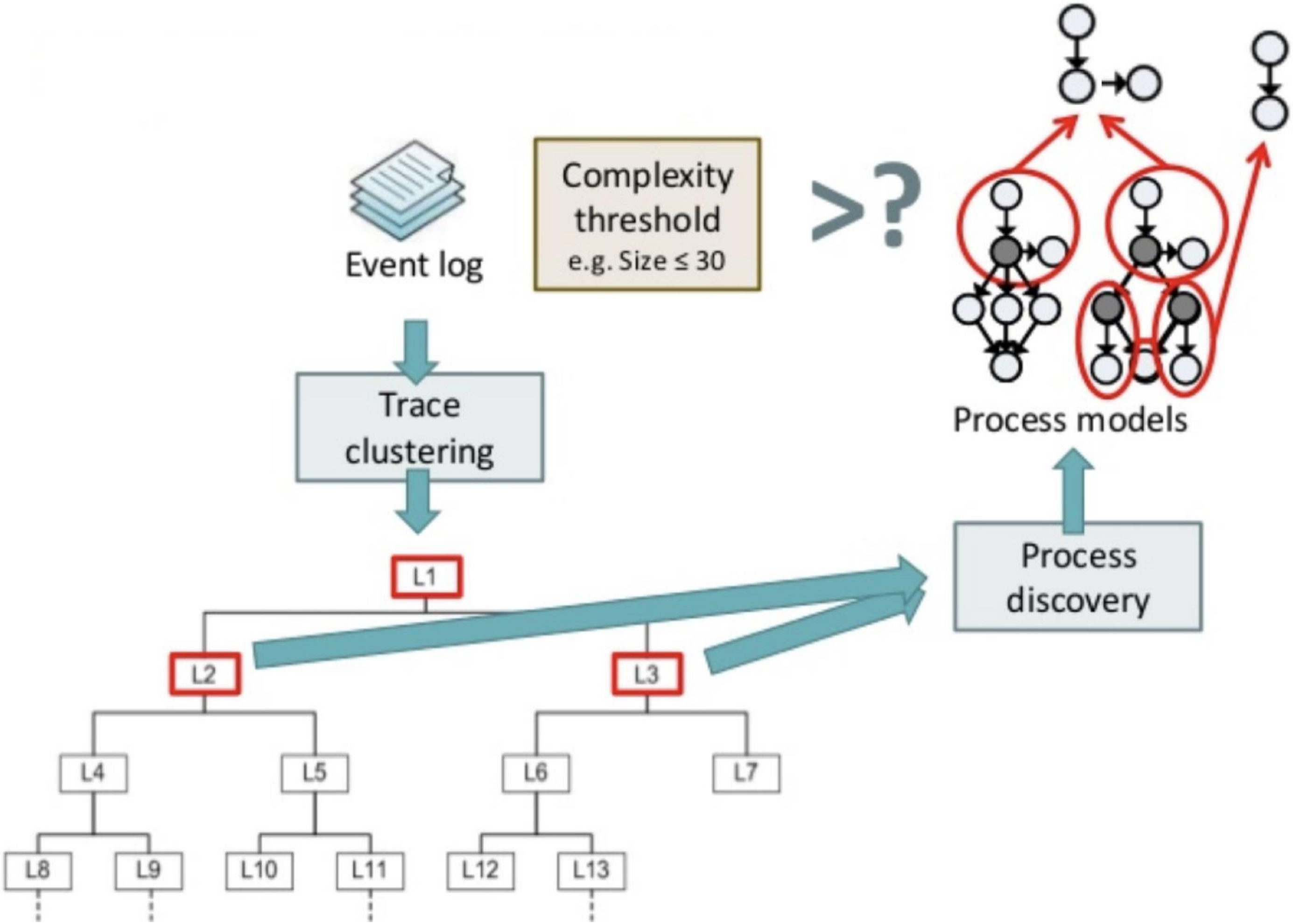
slice the log horizontally
per variant

dice the discovered models
hierarchically

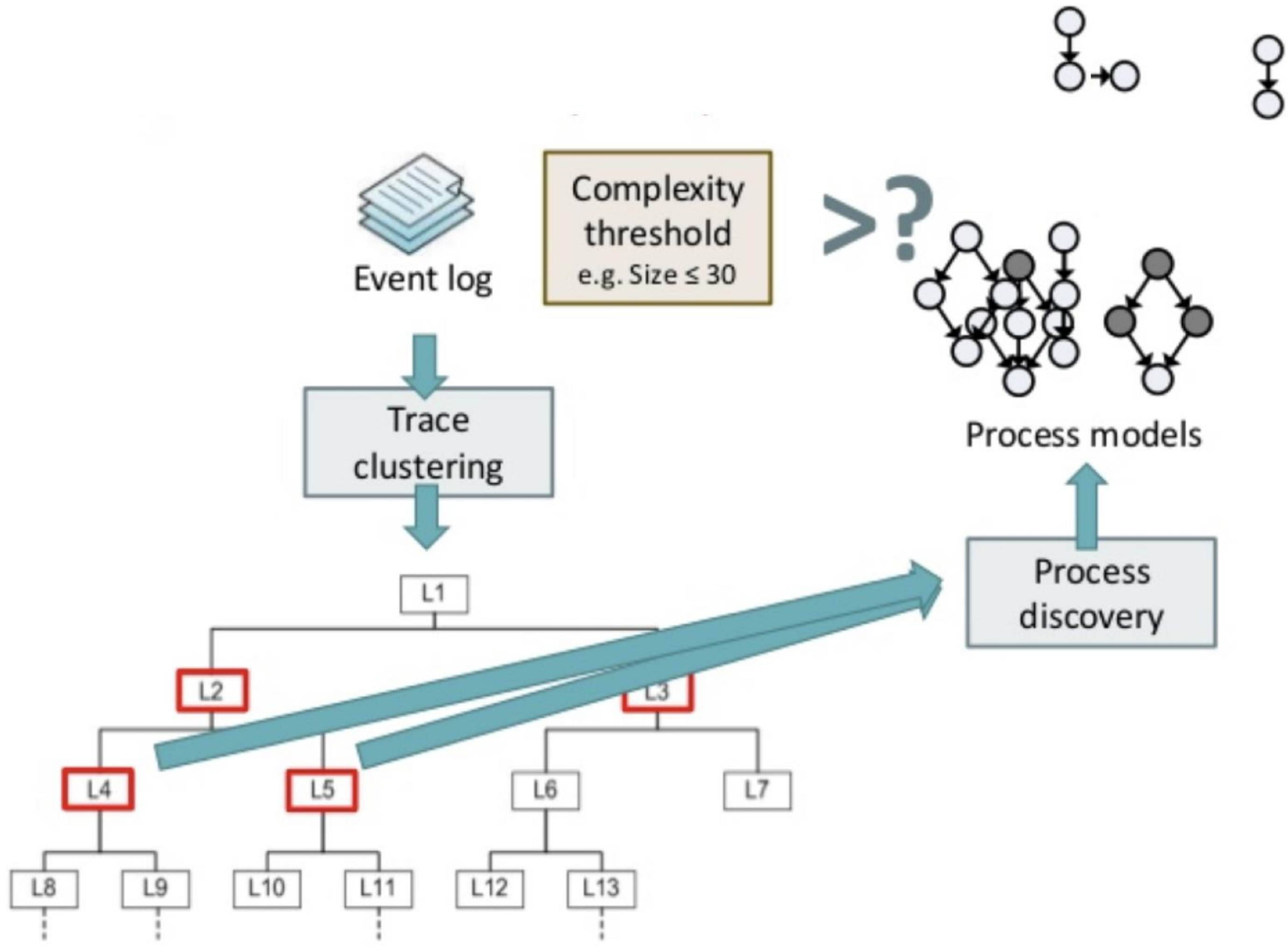
Slice, Mine and Dice (1)



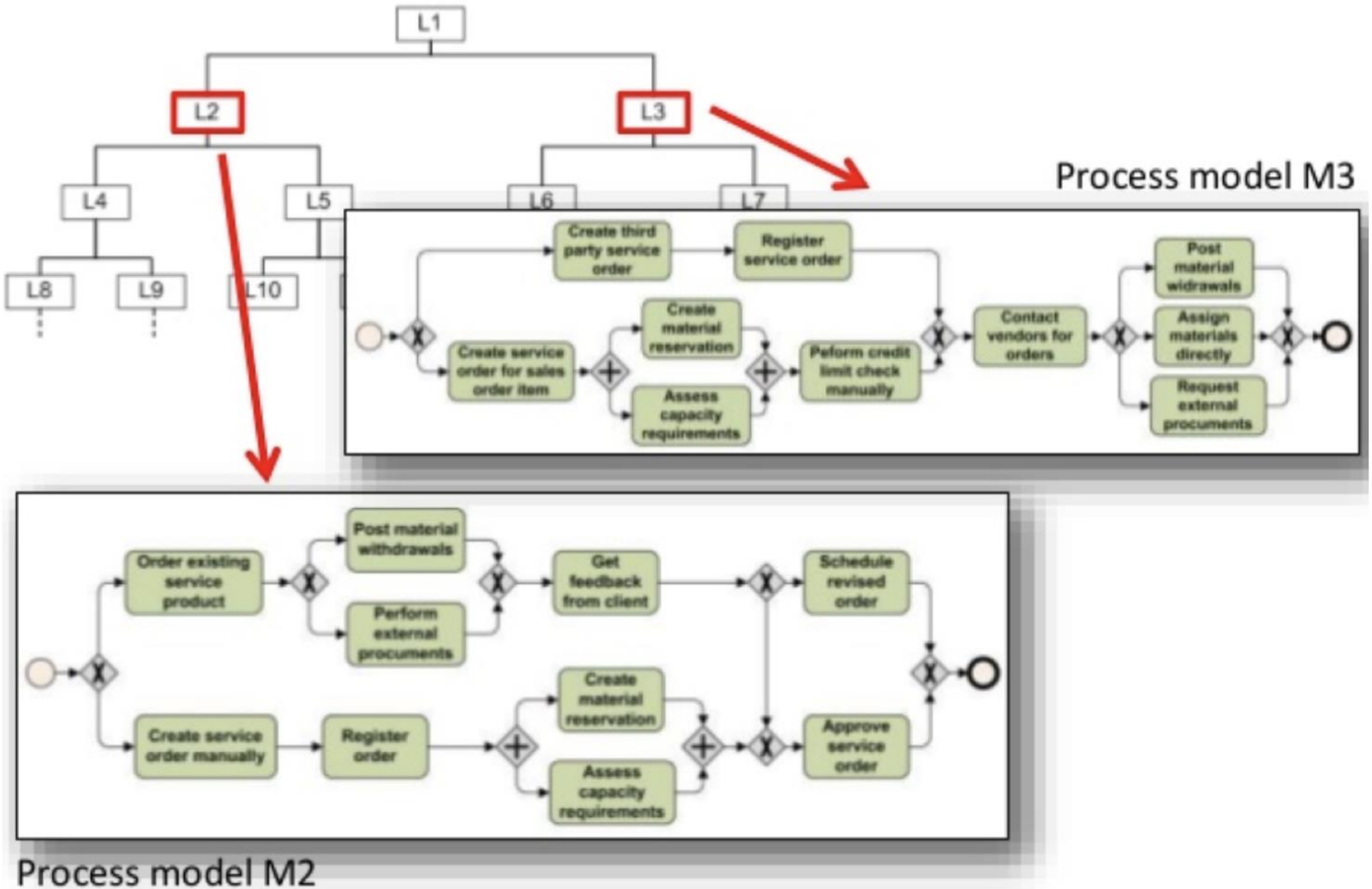
Slice, Mine and Dice (2)



Slice, Mine and Dice (3)



Slice, Mine and Dice (4)



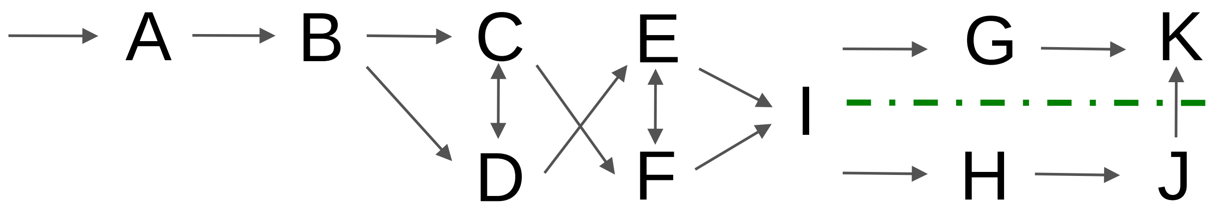
Process tree

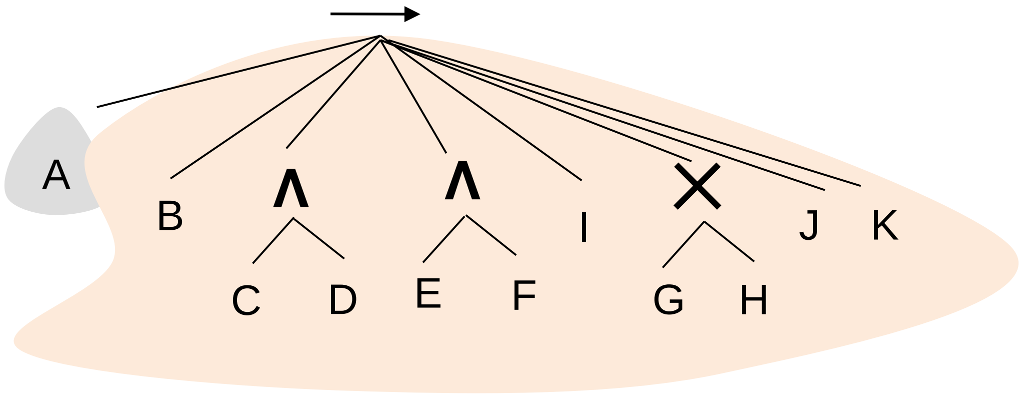
- A process tree is a directed connected graph without cycles.
- A node V in the graph is either a *branch node* or a *leaf*.
- Each leaf node represents an activity from the collection of activities A.
- Each branch node, or operator node:
  - $\rightarrow$: sequence
  - $\wedge$: parallel
  - $\underline{\times}$: choice
  - ↻ : iteration

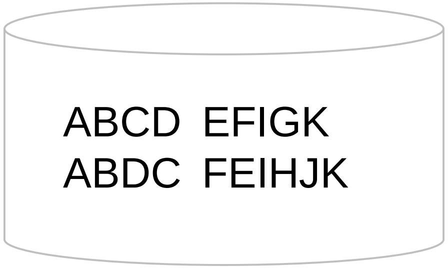
A Generic Algorithm for Discovering Process Trees
(Buijs, van Dogen, van der Aalst, 2012)

How to derive a process tree from an event log?

Christian-Albrechts-Universität zu Kiel

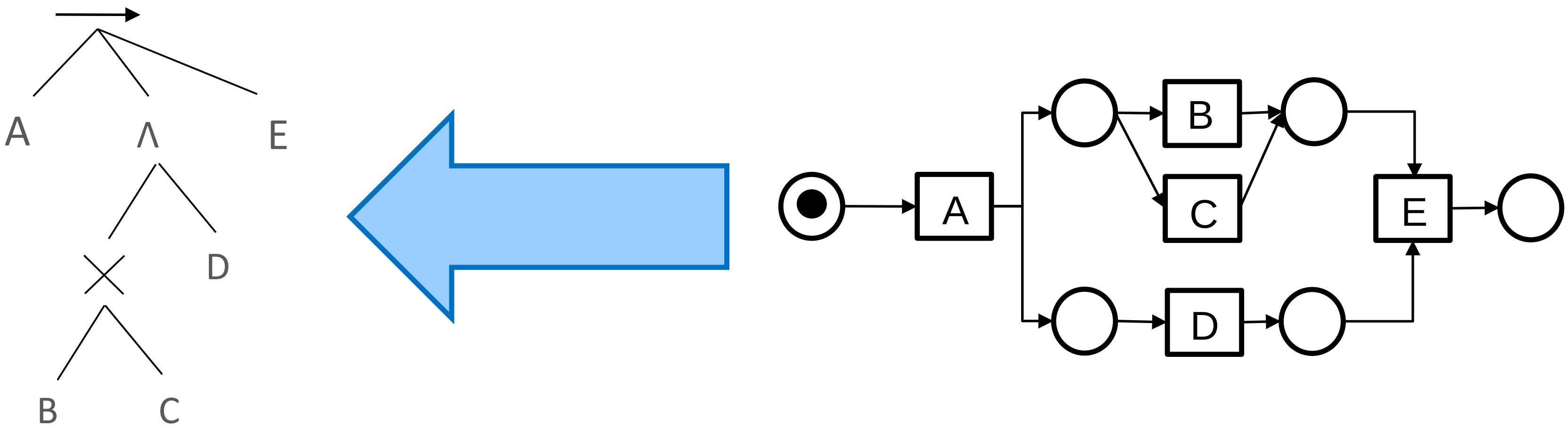




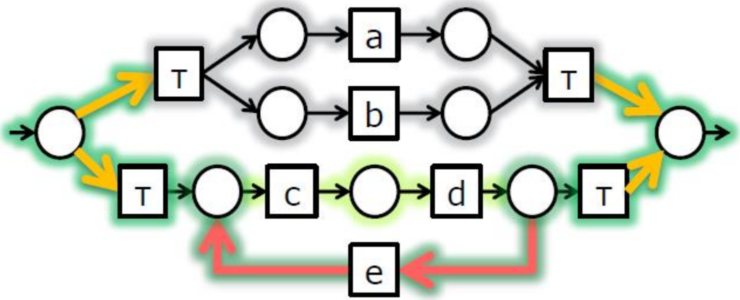


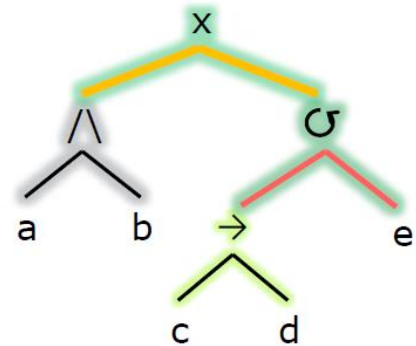
Derive a process tree from a graphical representation?

Christian-Albrechts-Universität zu Kiel



Extract process trees from event logs



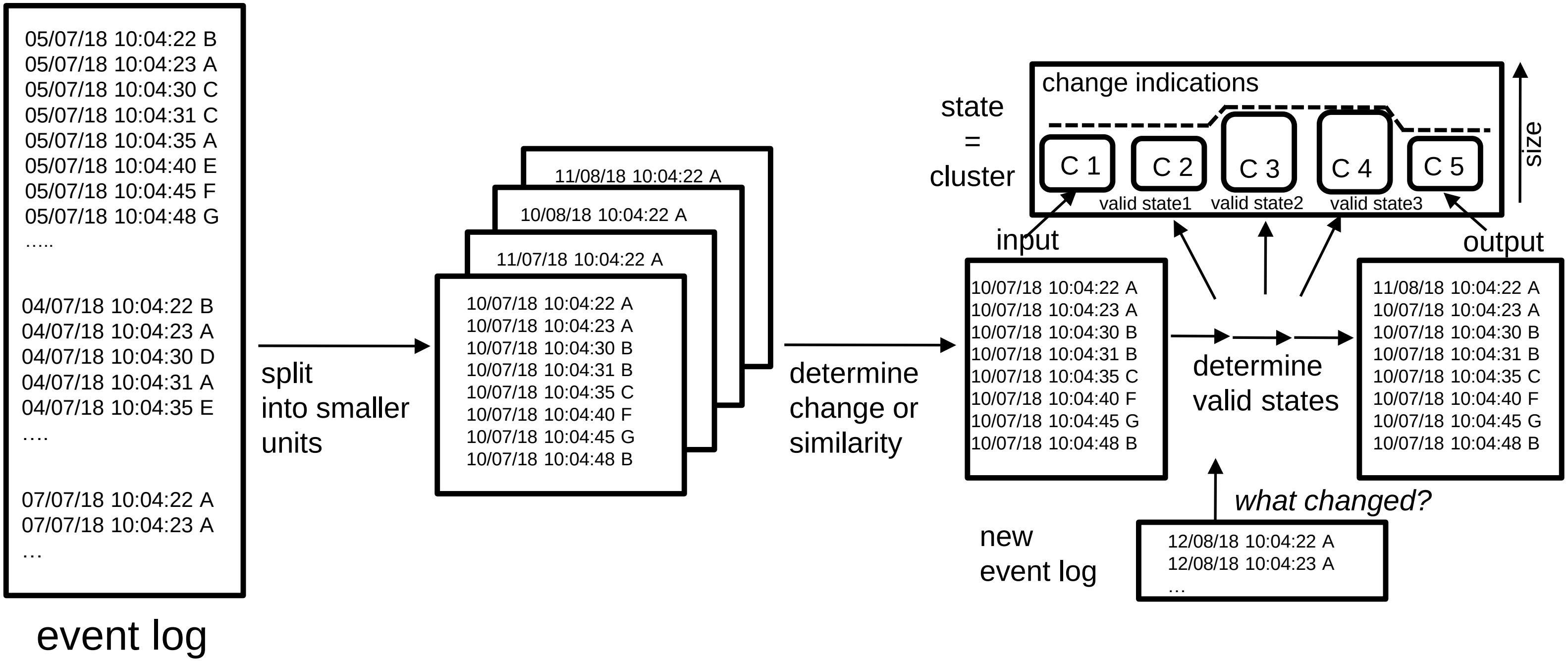


{<c,d,e,c,d>,
<c,d,e,c,d>,
<c,d>,
<c,d,e,c,d,e,c,d>
<c,d,e,c,d,e,c,d,e,c,d>}

- Process trees reduce the complexity
- Efficient techniques for process tree parsing

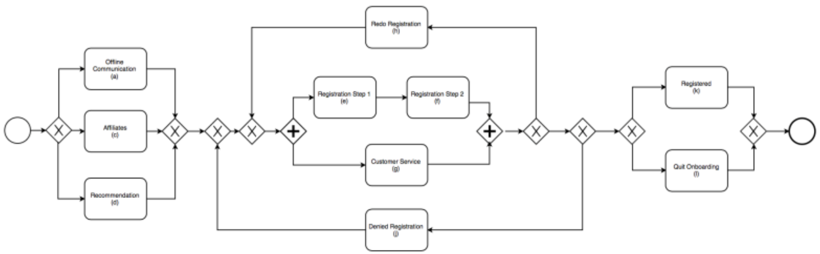
Extracting information from data: structure event log

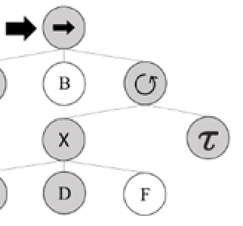
C | A | U
Christian-Albrechts-Universität zu Kiel



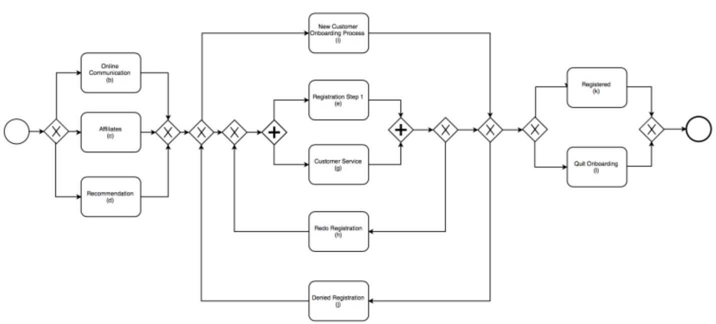
Source: A. Koschmider, D. Siqueira Vidal Moreira: Change Detection in Event Logs by Clustering. OTM Conferences 2018: 643-660

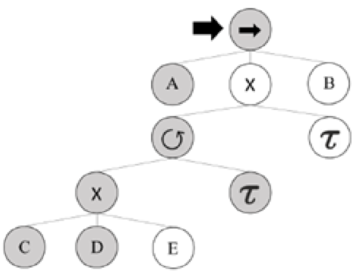
Step 1: define input & output





(seq, (A, B,(xl, (x, C, D, E))), tau))

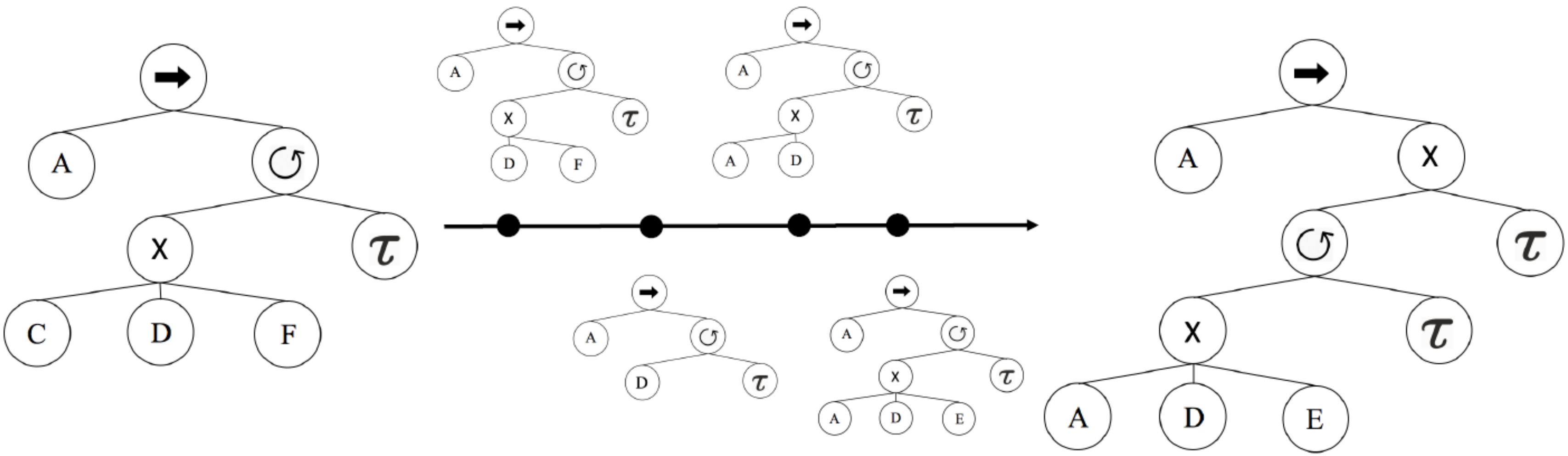




(seq, (A, (x, (xl, (x, C, D, E)), tau), tau), B)))

Apply Morphing

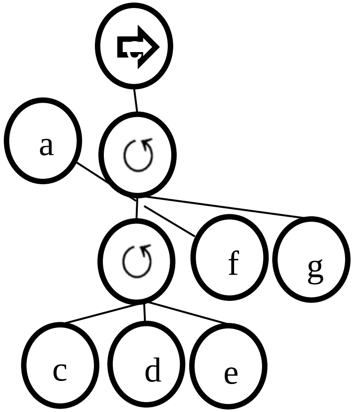
C | A | U
Christian-Albrechts-Universität zu Kiel

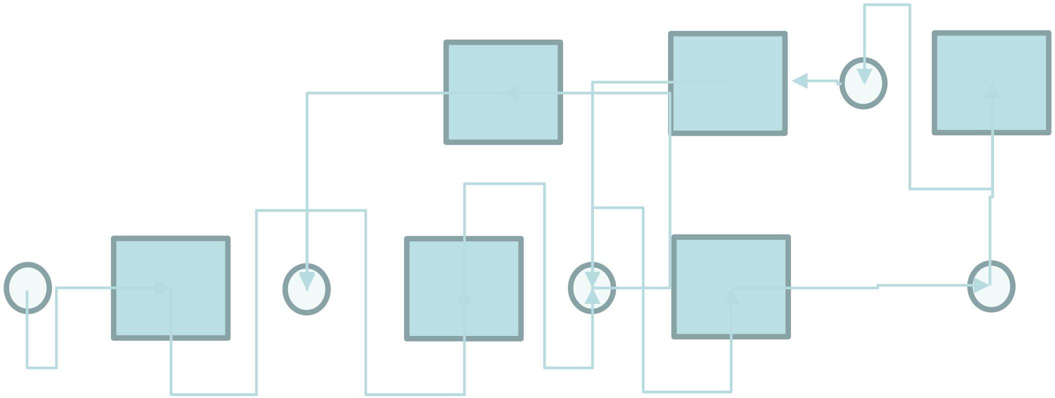


Valid state

Christian-Albrechts-Universität zu Kiel

- Process tree, represented either as behavior-oriented trace or graphically as process tree is valid if each branch is syntactically correct (i.e. complete).

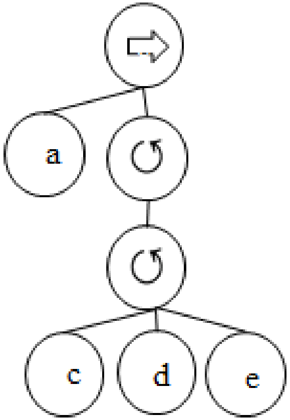


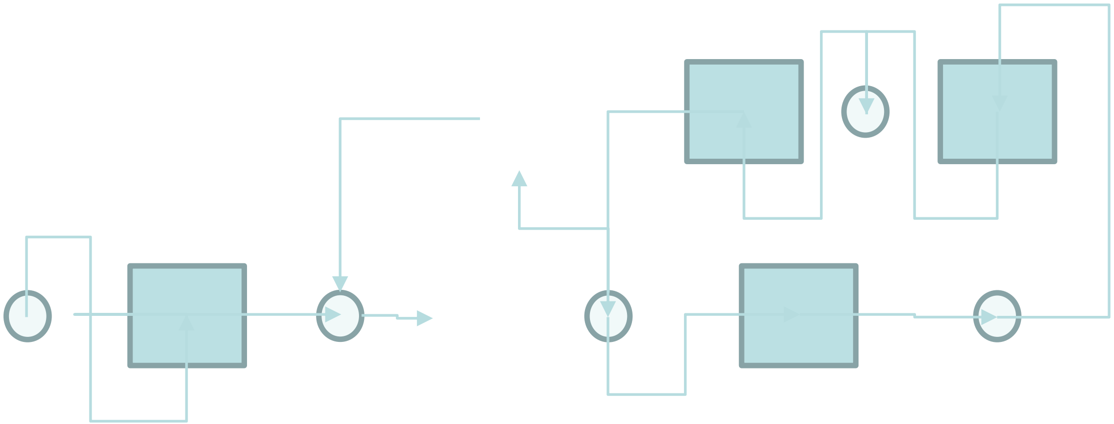


Invalid state

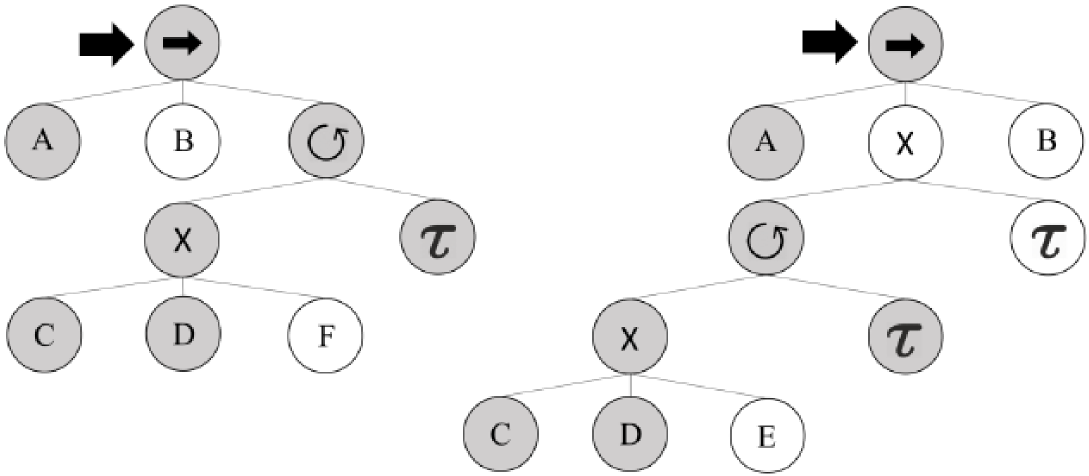
Christian-Albrechts-Universität zu Kiel

- A process tree is invalid if branching activities cannot be mapped to a syntactically correct Petri net





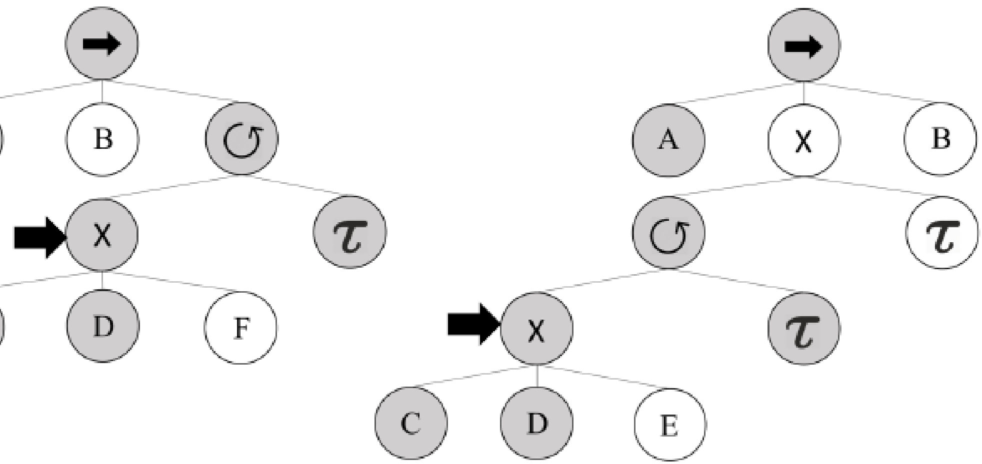
Apply morphing algorithm(1)



INPUT	OUTPUT
[seq, A, B,[xl, [x, C, D, F], tau]]	[seq, A, [x, [xl, [x, C, D, E], tau], tau], B]
[[xl, [x, C, D, F], tau]]	[x, [xl, [x, C, D, E], tau], tau]
[xl, [x, C, D, F], tau]	[xl, [x, C, D, E], tau],
[x, C, D, F]	[x, C, D, E]

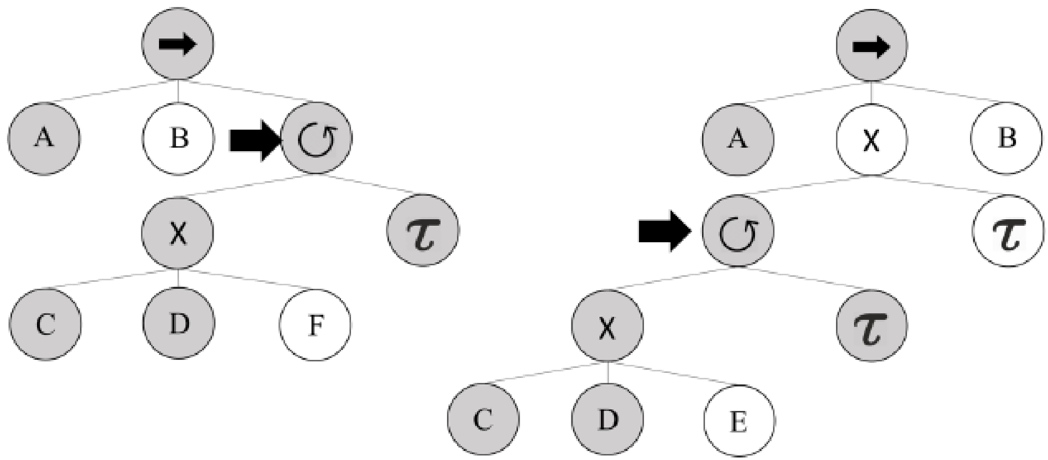
Apply morphing algorithm(2)

Christian-Albrechts-Universität zu Kiel



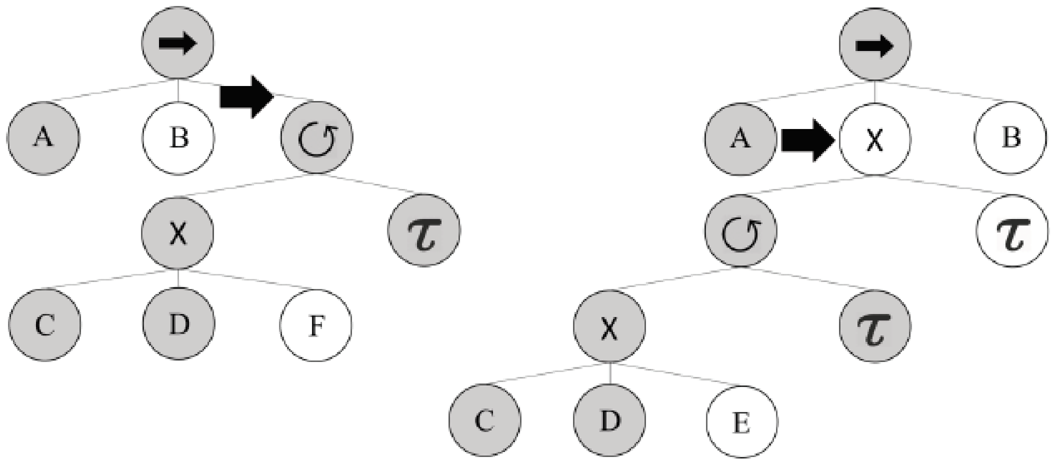
INPUT	OUTPUT
[seq, A, B,[xl, [x, C, D, F], tau]]	[seq, A, [x, [xl, [x, C, D, E], tau], tau], B]
[[xl, [x, C, D, F], tau]]	[x, [xl, [x, C, D, E], tau], tau]
[xl, [x, C, D, F], tau]	[xl, [x, C, D, E], tau],
[x, C, D, F]	[x, C, D, E]

Apply morphing algorithm(3)



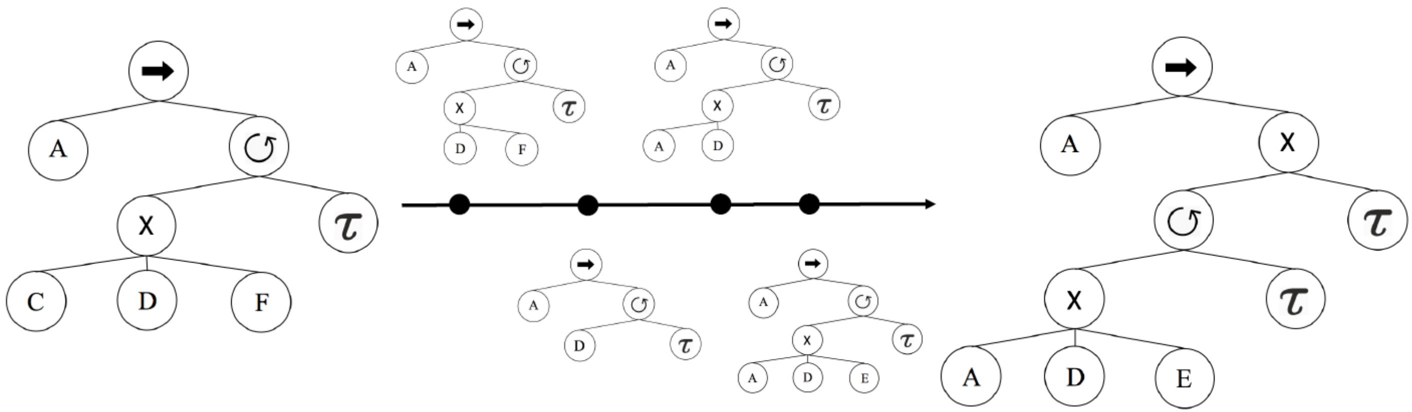
INPUT	OUTPUT
[seq, A, B,[xl, [x, C, D, F], tau]]	[seq, A, [x, [xl, [x, C, D, E], tau], tau], B]
[[xl, [x, C, D, F], tau]]	[x, [xl, [x, C, D, E], tau], tau]
[xl, [x, C, D, F], tau]	[xl, [x, C, D, E], tau],
[x, C, D, F]	[x, C, D, E]

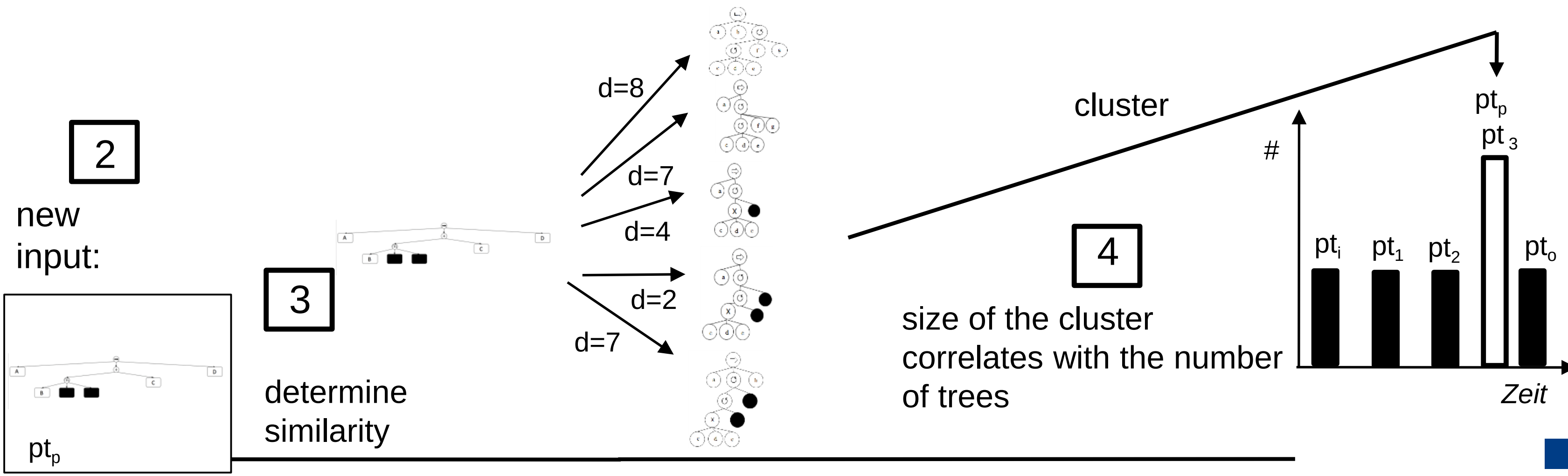
Apply morphing algorithm(4)



INPUT	OUTPUT
[seq, A, B,[xl, [x, C, D, F], tau]]	[seq, A, [x, [xl, [x, C, D, E], tau], tau], B]
[[xl, [x, C, D, F], tau]]	[x, [xl, [x, C, D, E], tau], tau]
[xl, [x, C, D, F], tau]	[xl, [x, C, D, E], tau],
[x, C, D, F]	[x, C, D, E]

Cluster




2
new input:
pt_p
3
determine similarity
d=8
d=7
d=4
d=2
d=7
cluster
4
size of the cluster correlates with the number of trees
#
pt_p
pt_3
pt_i
pt_1
pt_2
pt_o
Zeit

Determine Similarity

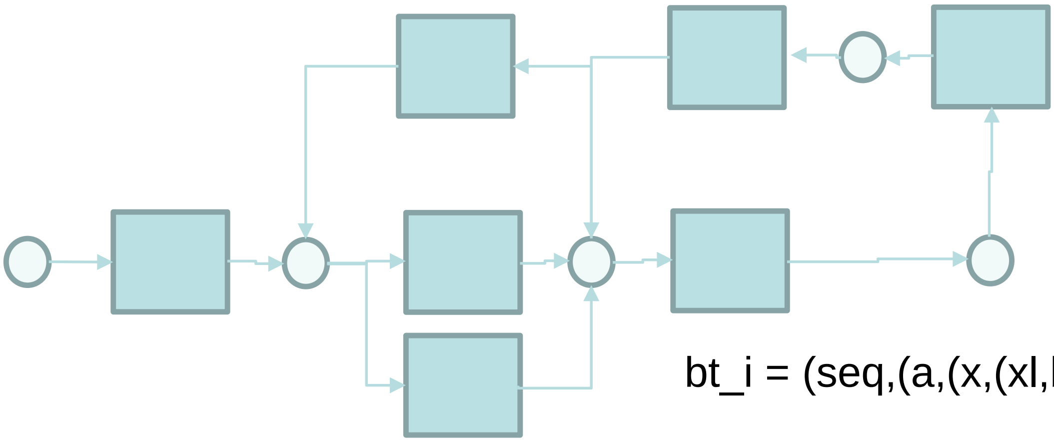
Christian-Albrechts-Universität zu Kiel

Levenshtein Distanz

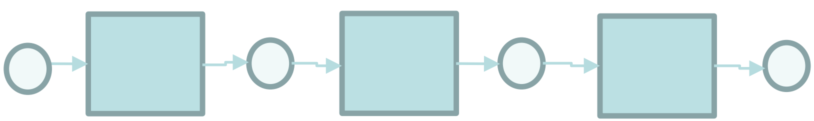
Trace 1 = (seq,(A,(x,(xl,C,C),D)),E)

Trace 2 = (seq,(A,B,C,D,E))

d=2



bt_i = (seq,(a,(x,(xl,b,c),d)),e)



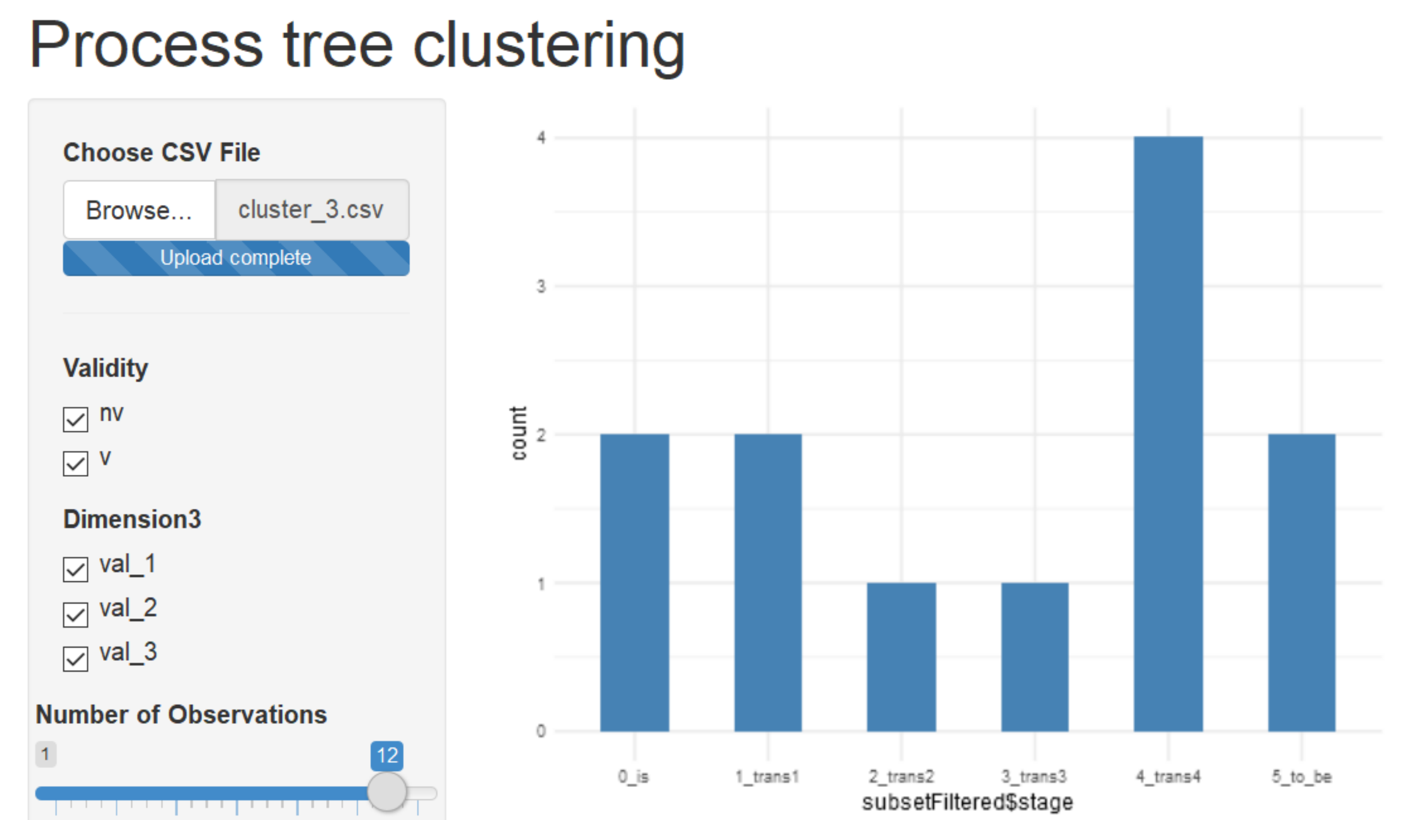
bt_o = (seq,(a,b,c,d,e))

Change analysis(1)

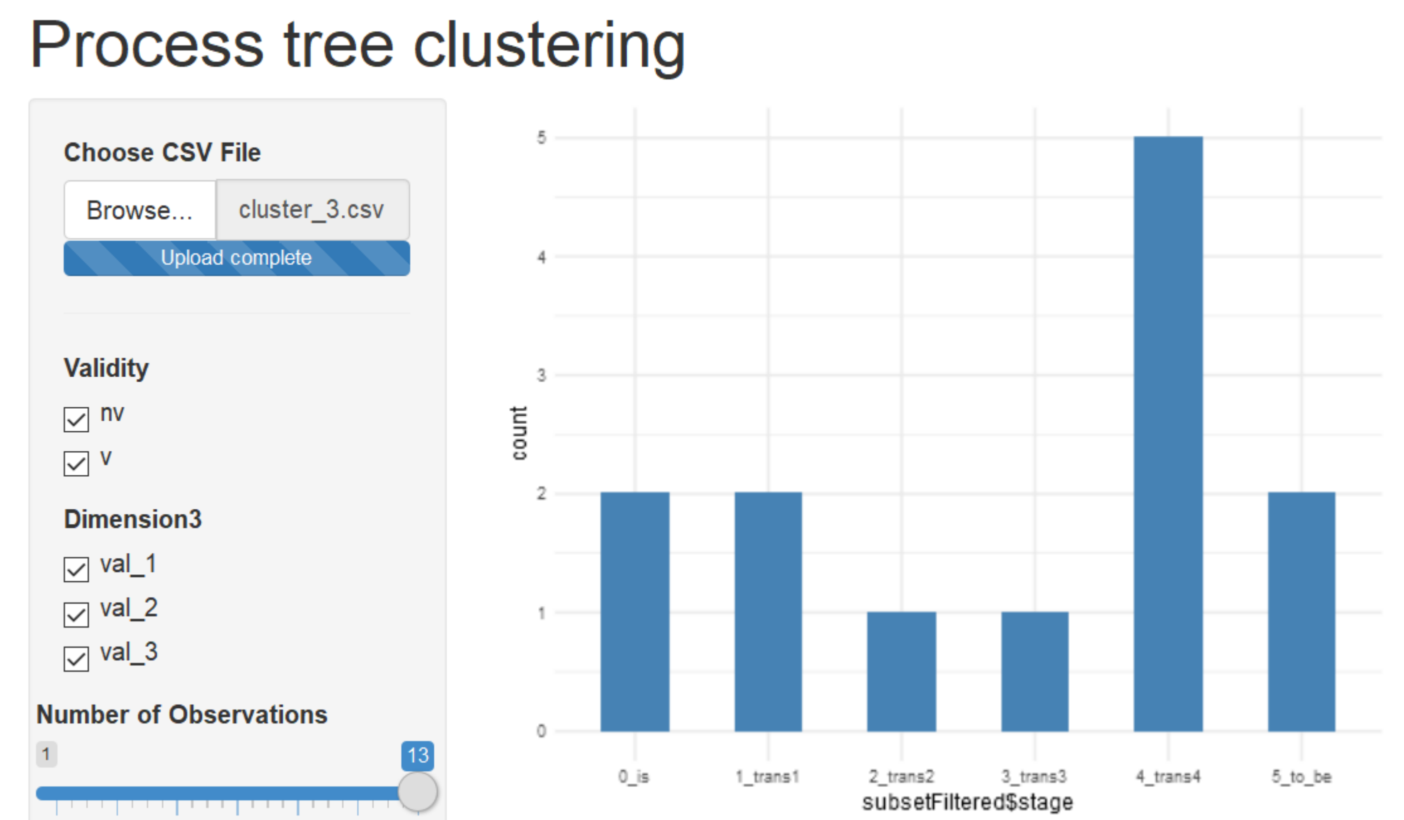
Change analysis(2)

Christian-Albrechts-Universität zu Kiel

Change analysis(3)

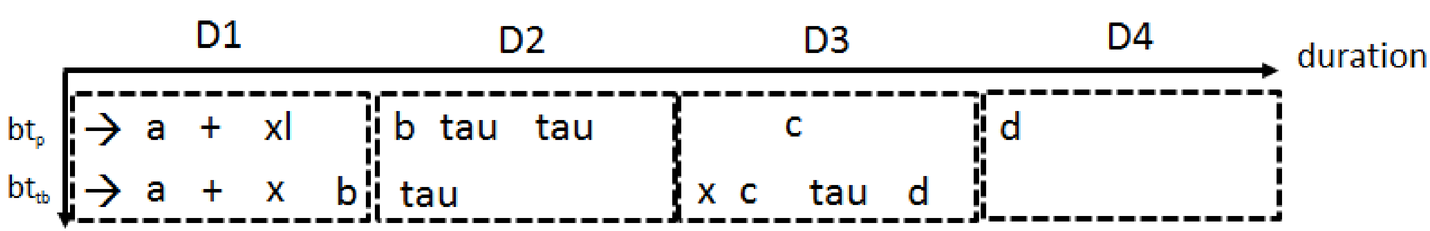


Change analysis (4)



Change analysis & explanation

Christian-Albrechts-Universität zu Kiel



Predictive Analysis

Christian-Albrechts-Universität zu Kiel

Process tree clustering