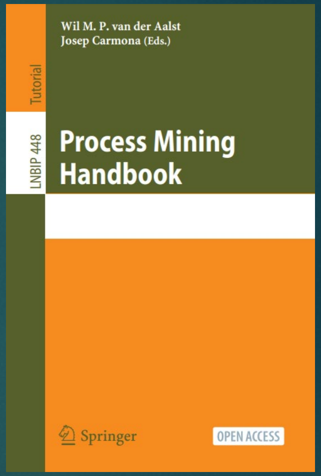
Process Mining Summer School, Aachen, 4-8 July 2022

#pmschool22

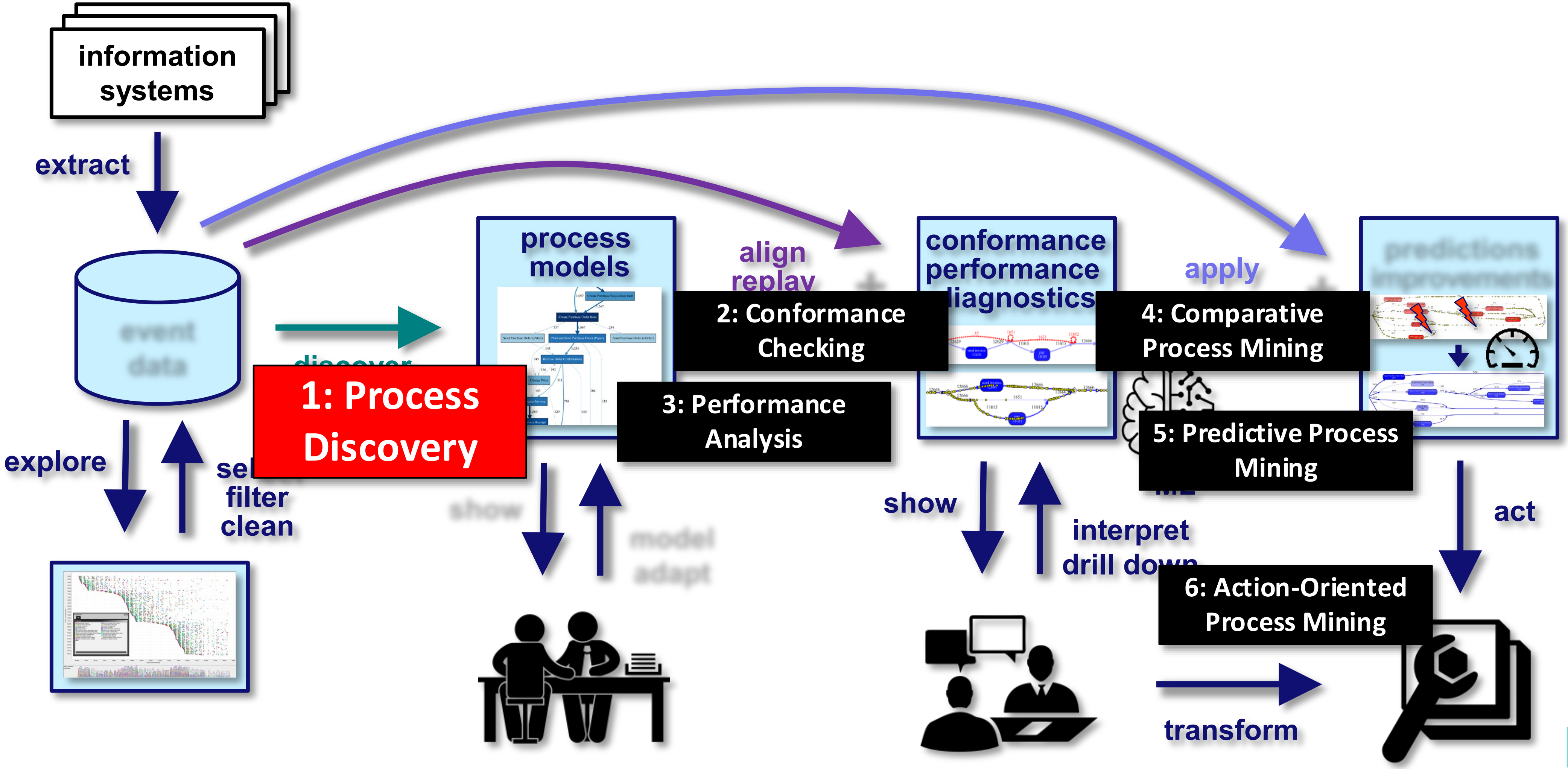
# Foundations of Process Discovery

WIL VAN DER AALST
PROCESS AND DATA SCIENCE @ RWTH AACHEN UNIVERSITY & CELONIS
www.vdaalst.com, @wvdaalst



# Recap: Six types of process mining

**In this lecture, we focus on process discovery**





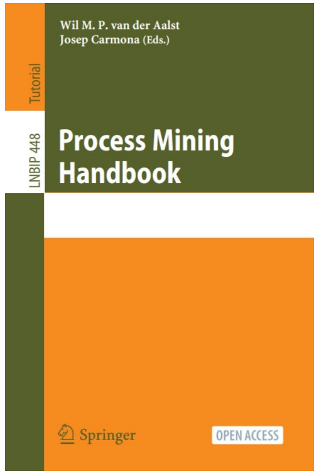

Chair of Process and Data Science

# Outline: Foundations of Process Discovery

## Baseline: Discovering DFG + filtering

| Bottom-up discovery | Top-down discovery |
| --- | --- |
| **Alpha algorithm** | **Inductive mining** |

At times, I refer to the formal definitions in the Chapter 2 to show that with the right tools one can be precise and compact.



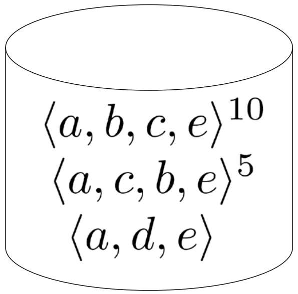

Chair of Process and Data Science

# Main idea of process discovery

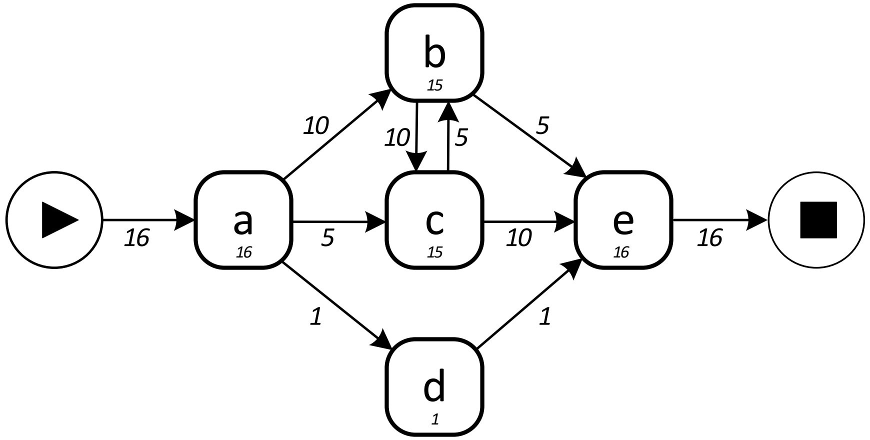


Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

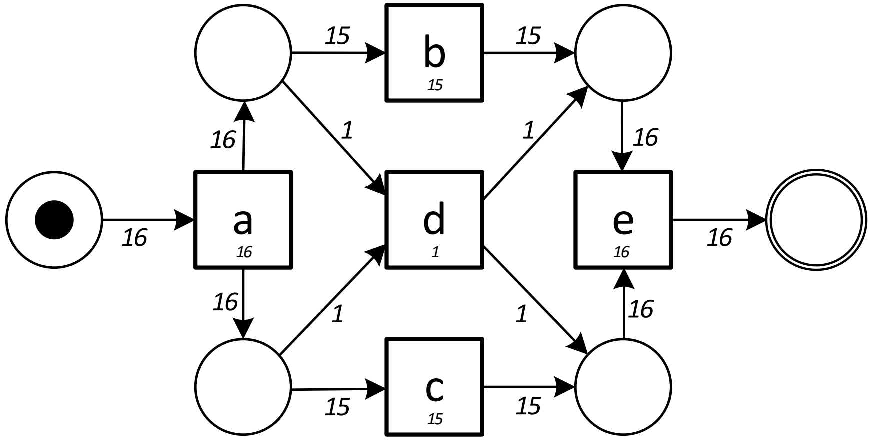
# The main idea (informal)



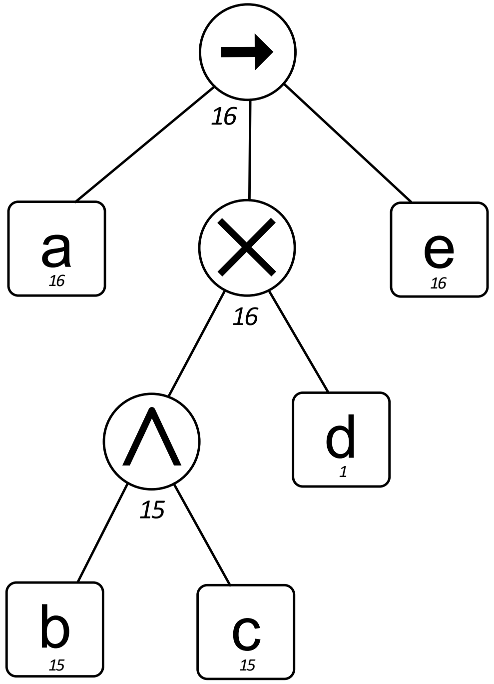

(a) Event log $L_1$




(b) Directly-Follows Graph (DFG): $M_1$




(c) Accepting Petri Net (APN): $M_2$

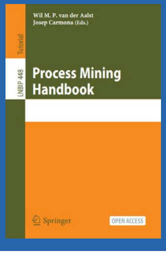



(d) Process Tree (PT): $M_3$



Chair of Process and Data Science

# The main idea (formal)




**Definition 1 (Event Log).** *$\mathcal{U}_{act}$ is the universe of activity names. A trace $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in \mathcal{U}_{act}{}^*$ is a sequence of activities. An event log $L \in \mathcal{B}(\mathcal{U}_{act}{}^*)$ is a multiset of traces.*

**Definition 2 (Process Model).** *$\mathcal{U}_M$ is the universe of process models. A process model $M \in \mathcal{U}_M$ defines a set of traces $lang(M) \subseteq \mathcal{U}_{act}{}^*$.*

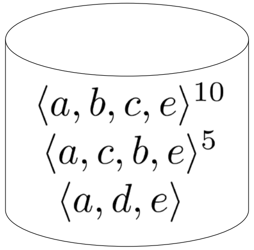
**Definition 3 (Process Discovery Algorithm).** *A process discovery algorithm is a function $disc \in \mathcal{B}(\mathcal{U}_{act}{}^*) \rightarrow \mathcal{U}_M$, i.e., based on a multiset of traces, a model is produced.*

Chair of Process and Data Science

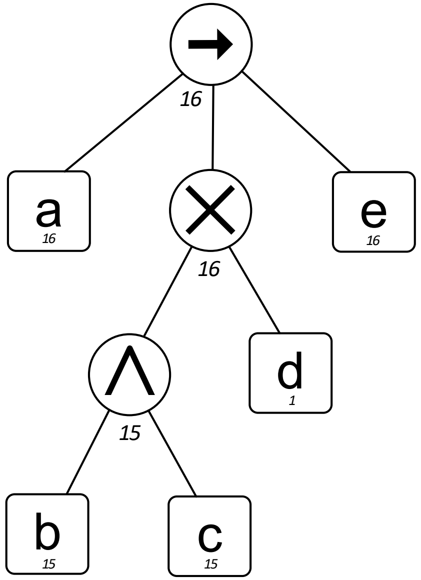
# Example

$$L_1 = [\langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{5}, \langle a,d,e\rangle] \in \mathcal{B}(\mathcal{U}_{act}{}^*)$$

$$lang(M_3) = \{\langle a,b,c,e\rangle, \langle a,c,b,e\rangle, \langle a,d,e\rangle\} \subseteq \mathcal{U}_{act}{}^*$$




Event log $L_1$




Process Tree (PT): $M_3$

Coincidence, model may allow for more or less than observed in the event log.



Chair of Process and Data Science

# How discover a process model?

- Base-line approach using Directly Follows Graphs (DFGs)
- Bottom-up discovery
  - Alpha algorithm
- Top-down discovery
  - Inductive Mining (IM) algorithm

**Definition 1 (Event Log).** *$\mathcal{U}_{act}$ is the universe of activity names. A trace $\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in {\mathcal{U}_{act}}^*$ is a sequence of activities. An event log $L \in \mathcal{B}({\mathcal{U}_{act}}^*)$ is a multiset of traces.*

**Definition 2 (Process Model).** *$\mathcal{U}_M$ is the universe of process models. A process model $M \in \mathcal{U}_M$ defines a set of traces $lang(M) \subseteq {\mathcal{U}_{act}}^*$.*

**Definition 3 (Process Discovery Algorithm).** *A process discovery algorithm is a function $disc \in \mathcal{B}({\mathcal{U}_{act}}^*) \to \mathcal{U}_M$, i.e., based on a multiset of traces, a model is produced.*



PADS Chair of Process and Data Science

# Baseline approach using DFGs



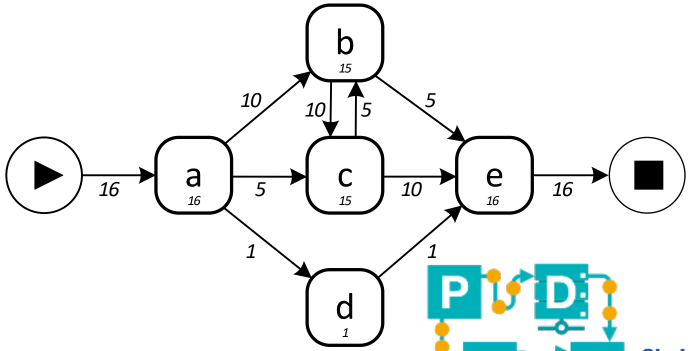
Chair of Process and Data Science

RWTH AACHEN UNIVERSITY

# Baseline approach: DFGs

**Definition 4 (Directly-Follows Graph).** *A Directly-Follows Graph (DFG) is a pair* $G = (A, F)$ *where* $A \subseteq \mathcal{U}_{act}$ *is a set of activities and* $F \in \mathcal{B}((A \times A) \cup (\{\blacktriangleright\} \times A) \cup (A \times \{\blacksquare\}) \cup (\{\blacktriangleright\} \times \{\blacksquare\}))$ *is a multiset of arcs.* $\blacktriangleright$ *is the start node and* $\blacksquare$ *is the end node* $(\{\blacktriangleright, \blacksquare\} \cap \mathcal{U}_{act} = \emptyset)$. $\mathcal{U}_G \subseteq \mathcal{U}_M$ *is the set of all DFGs.*

- **Graph with nodes representing activities and start ▶ and end ■.**
- **Behavior starts with dummy activity ▶ and ends with dummy activity ■. Node ▶ is a source node and ■ is a sink node.**
- **Arcs represent the directly-follows relation.**
- **Multisets to represent frequencies.**
- **Can be viewed as summary of the data!**

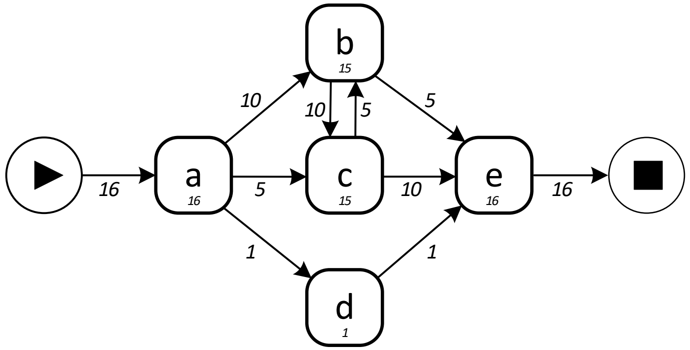



Chair of Process and Data Science

# Language of a DFG

**Definition 5 (Traces of a DFG).** *Let $G = (A, F) \in \mathcal{U}_G$ be a DFG. The set of possible traces described by $G$ is $lang(G) = \{\langle a_2, a_3, \ldots, a_{n-1}\rangle \mid a_1 = \blacktriangleright \land a_n = \blacksquare \land \forall_{1 \leq i < n}\ (a_i, a_{i+1}) \in F\}$.*

- **Possible traces: All paths possible according to the graph starting in node ► and ending in node ■.**
- **Recall: ► is a source node and ■ is a sink node.**





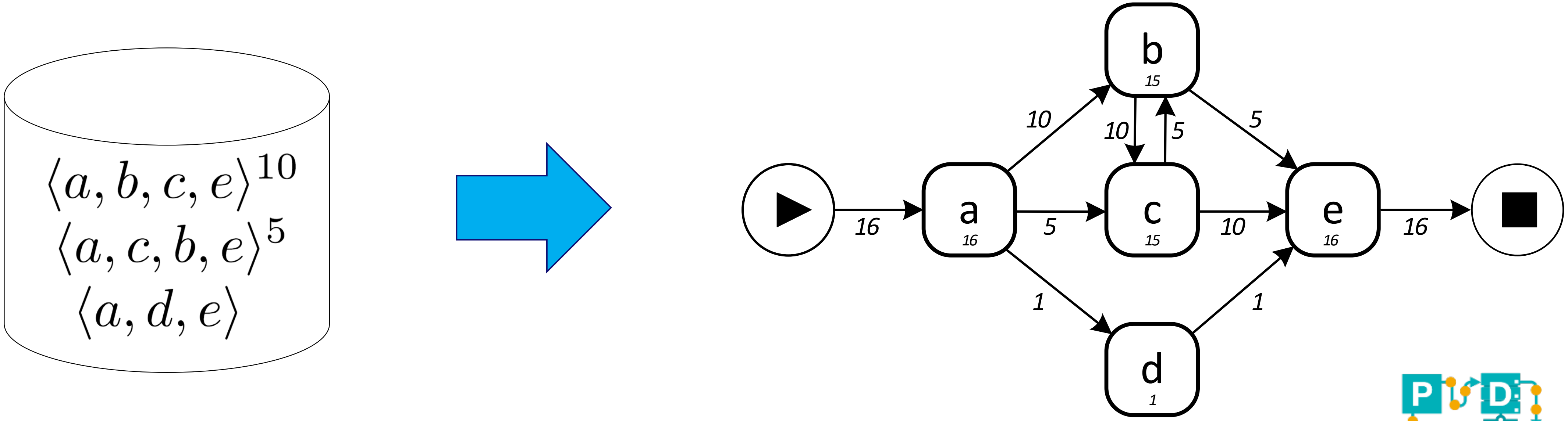

Chair of Process and Data Science

# Baseline discovery

## Your first discovery algorithm in just two lines of mathematics

**Definition 6 (Baseline Discovery Algorithm).** *Let* $L \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$ *be an event log.* $disc_{DFG}(L) = (A, F)$ *is the DFG based on* $L$ *with:*

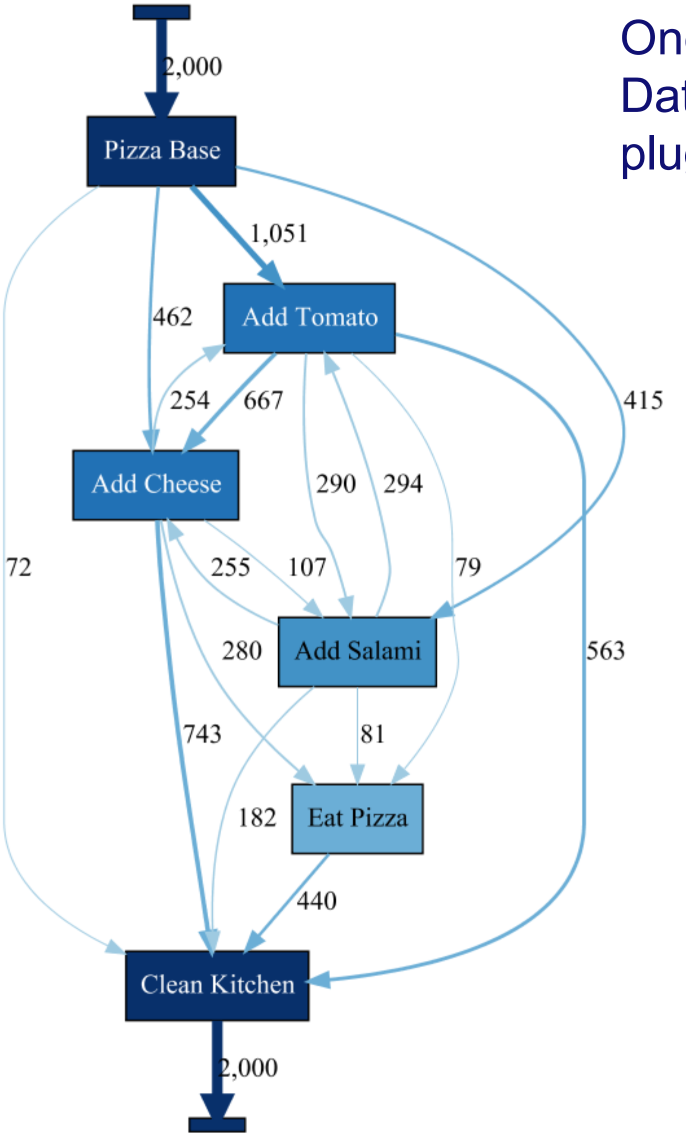
- $A = \{a \in \sigma \mid \sigma \in L\}$ *and*
- $F = [(\sigma_i, \sigma_{i+1}) \mid \sigma \in L' \wedge 1 \leq i < |\sigma|]$ *with* $L' = [\langle \blacktriangleright \rangle \cdot \sigma \cdot \langle \blacksquare \rangle \mid \sigma \in L]$.






Chair of Process and Data Science

# DFG discovery in ProM

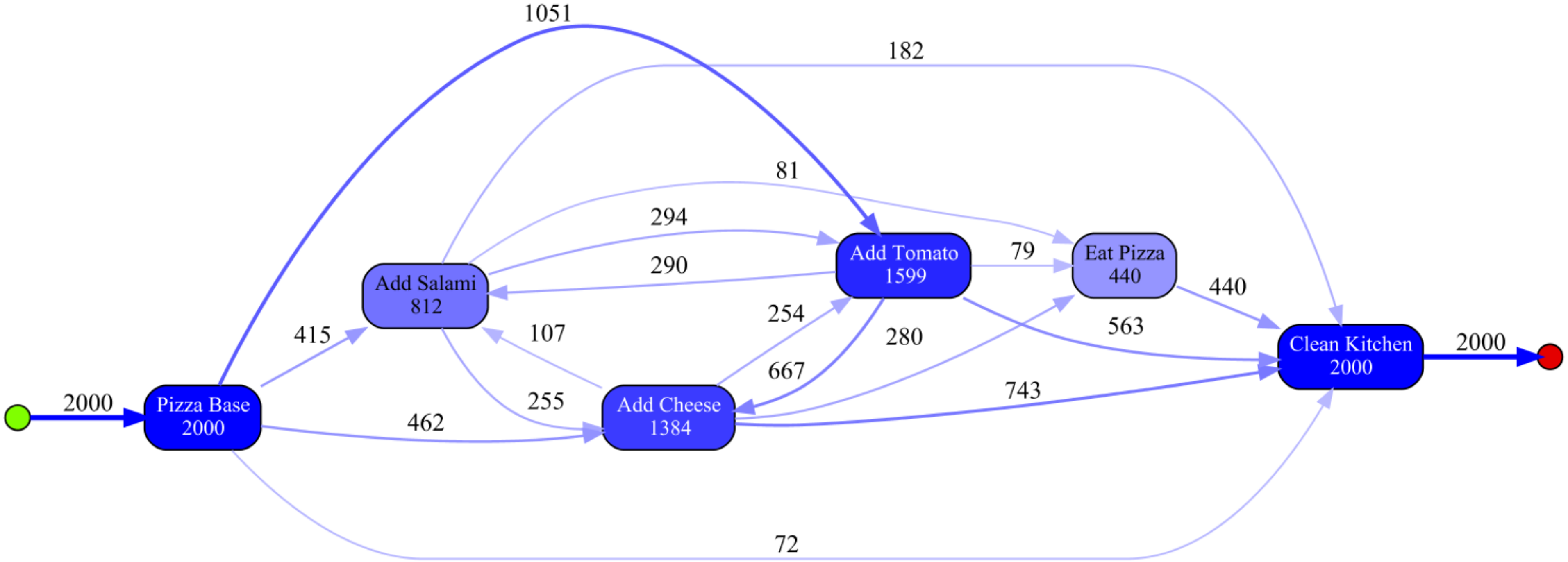

One of the views of the Data-aware heuristic miner plug-in (Felix Mannhardt)

ProM process mining workbench

Chair of Process and Data Science

# DFG discovery in ProM

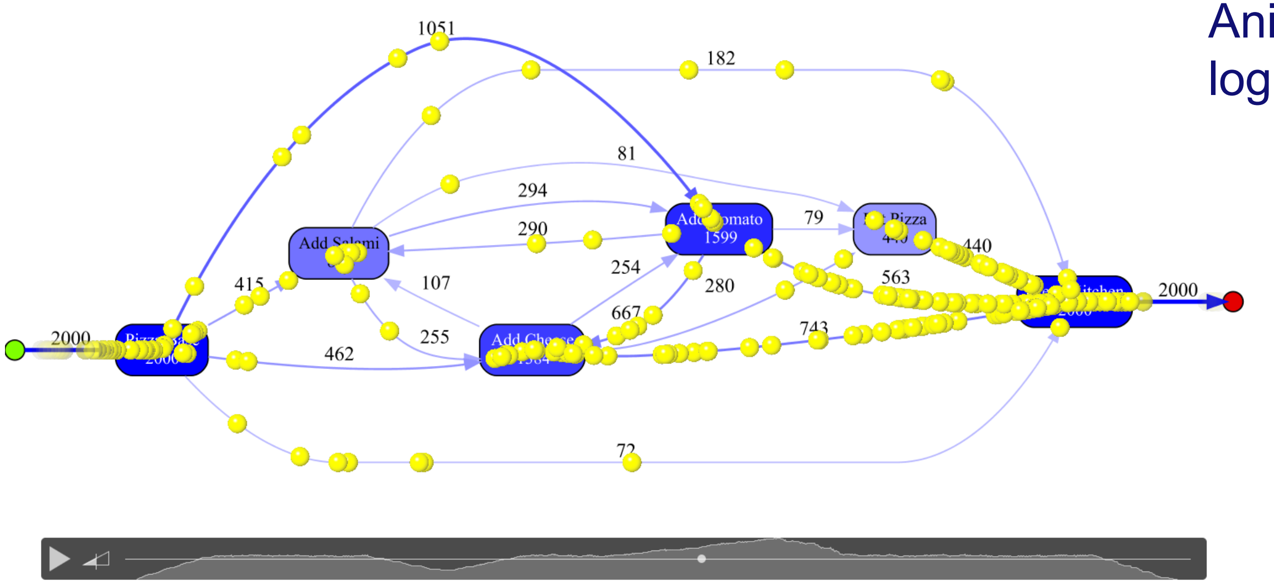

Directly-follows visual miner (Sander Leemans)



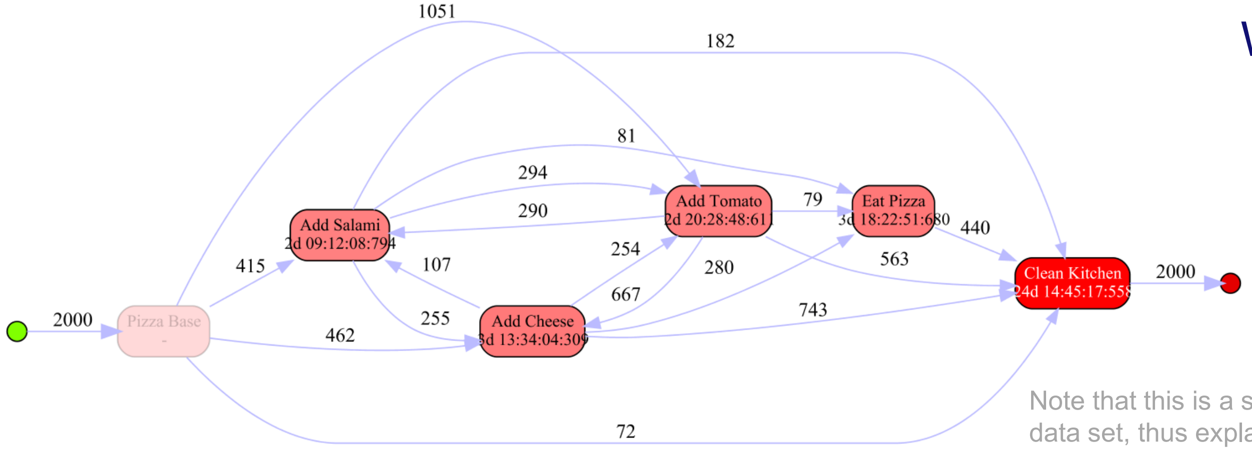

ProM
process mining workbench

Chair of Process and Data Science

# DFG discovery in ProM


Animation of the event log on top of the model



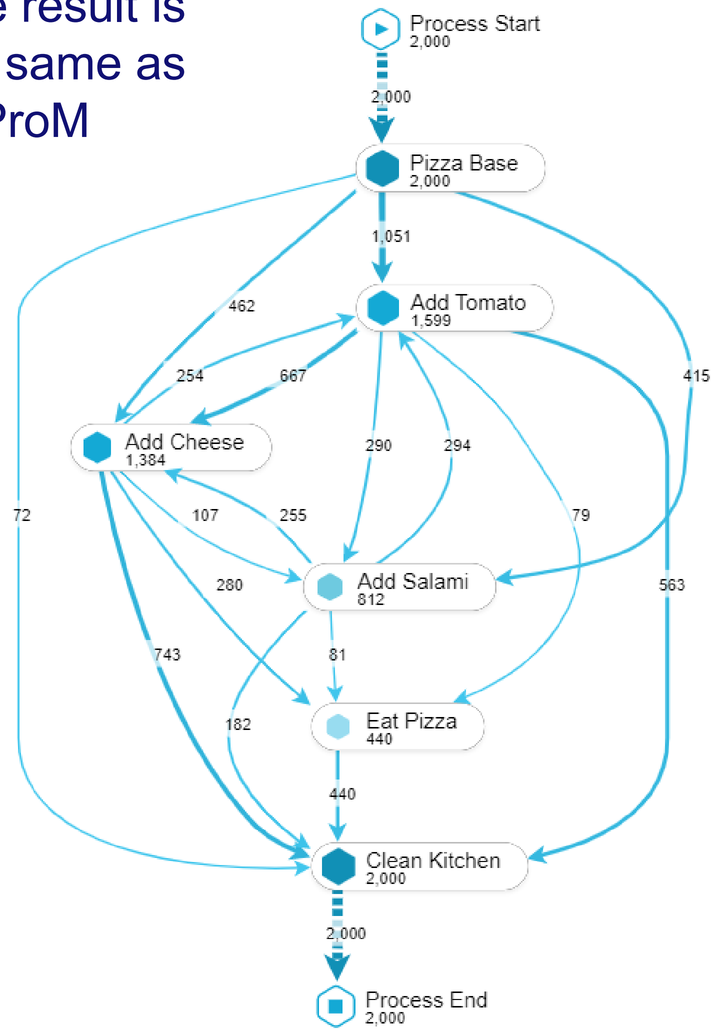

Waiting times

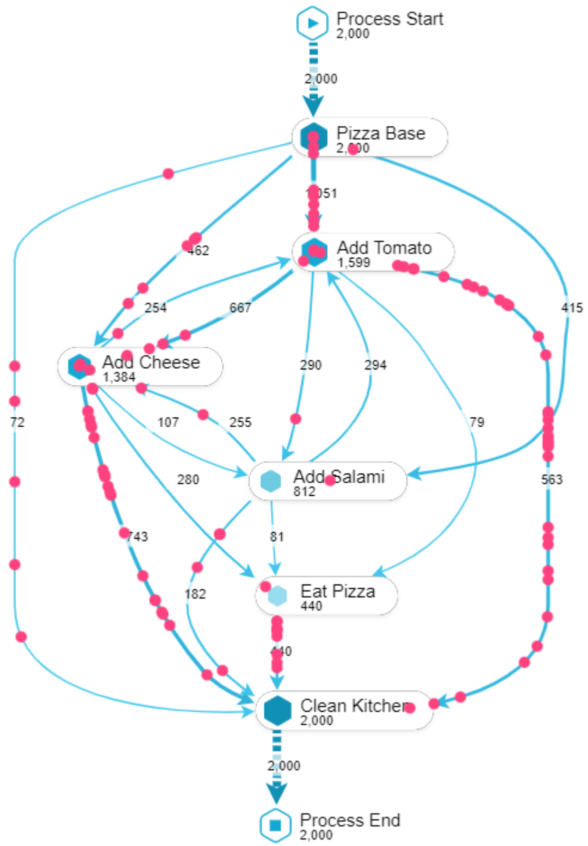
Note that this is a synthetic data set, thus explaining the long delays.

Chair of Process and Data Science

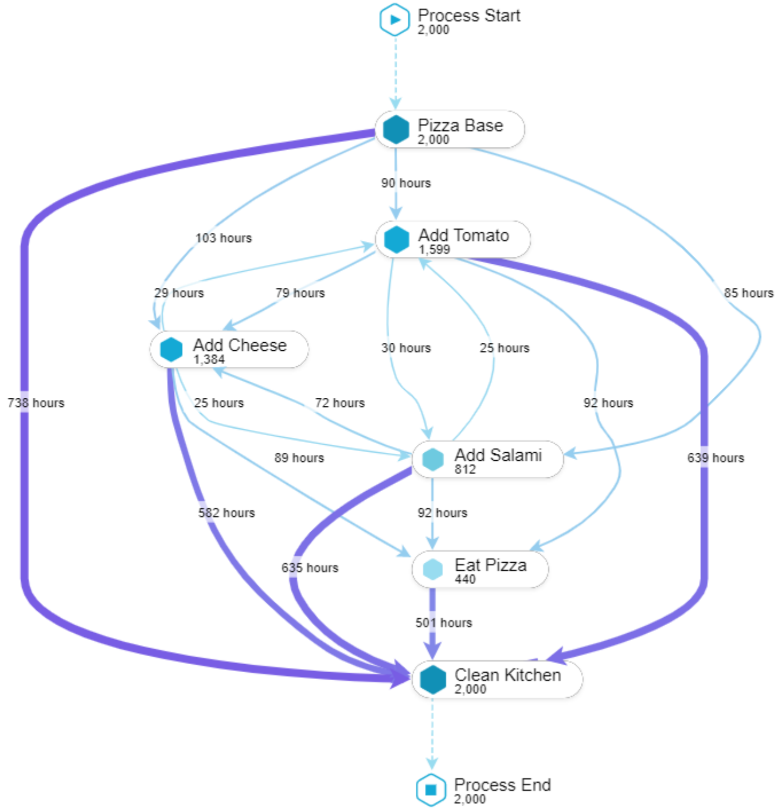
# DFG Discovery in Celonis

The result is the same as in ProM




animation



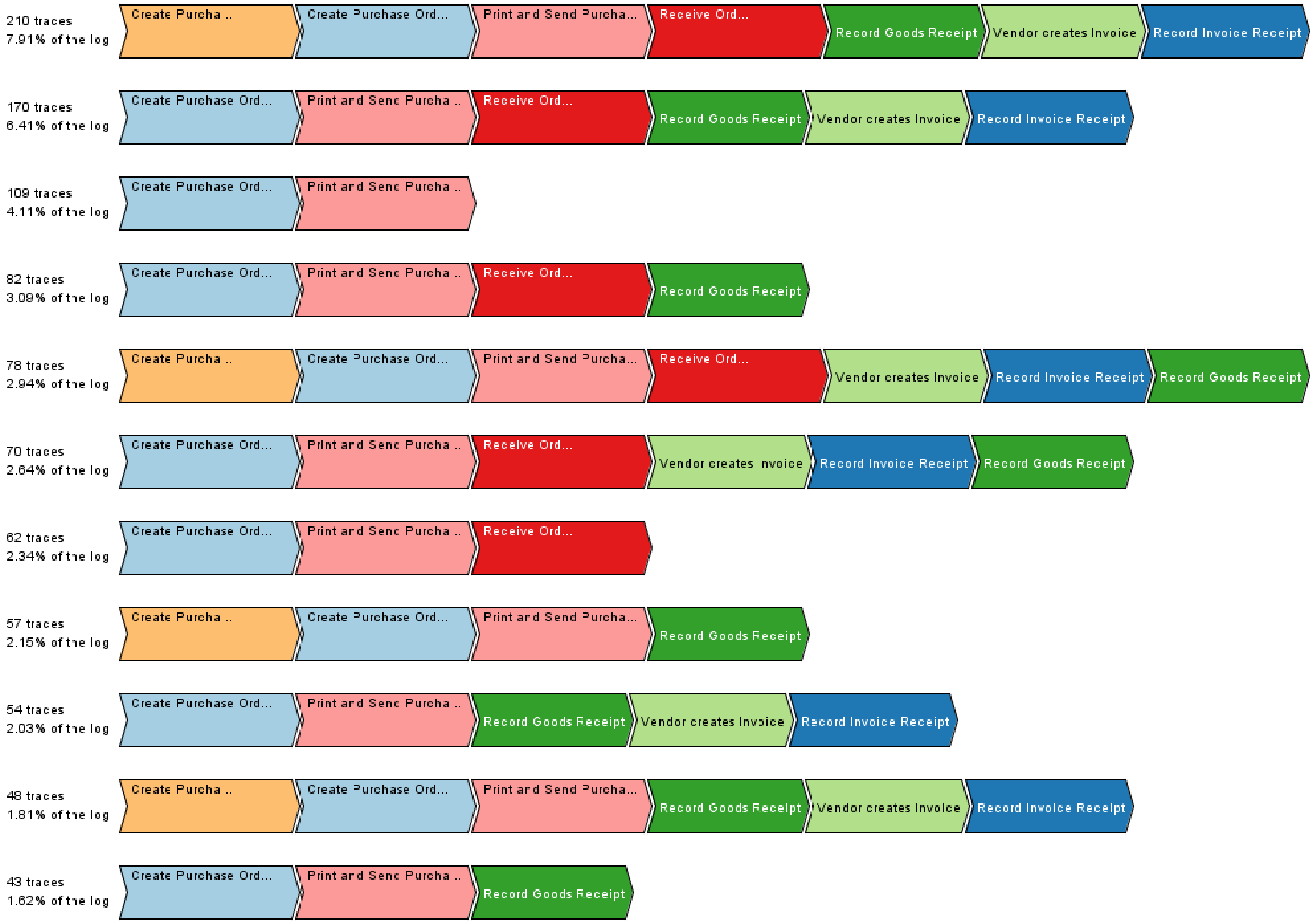

times

Note that this is a synthetic data set, thus explaining the long delays.

celonis

PADS Chair of Process and Data Science

# What if we get Spaghetti instead of Lasagna?

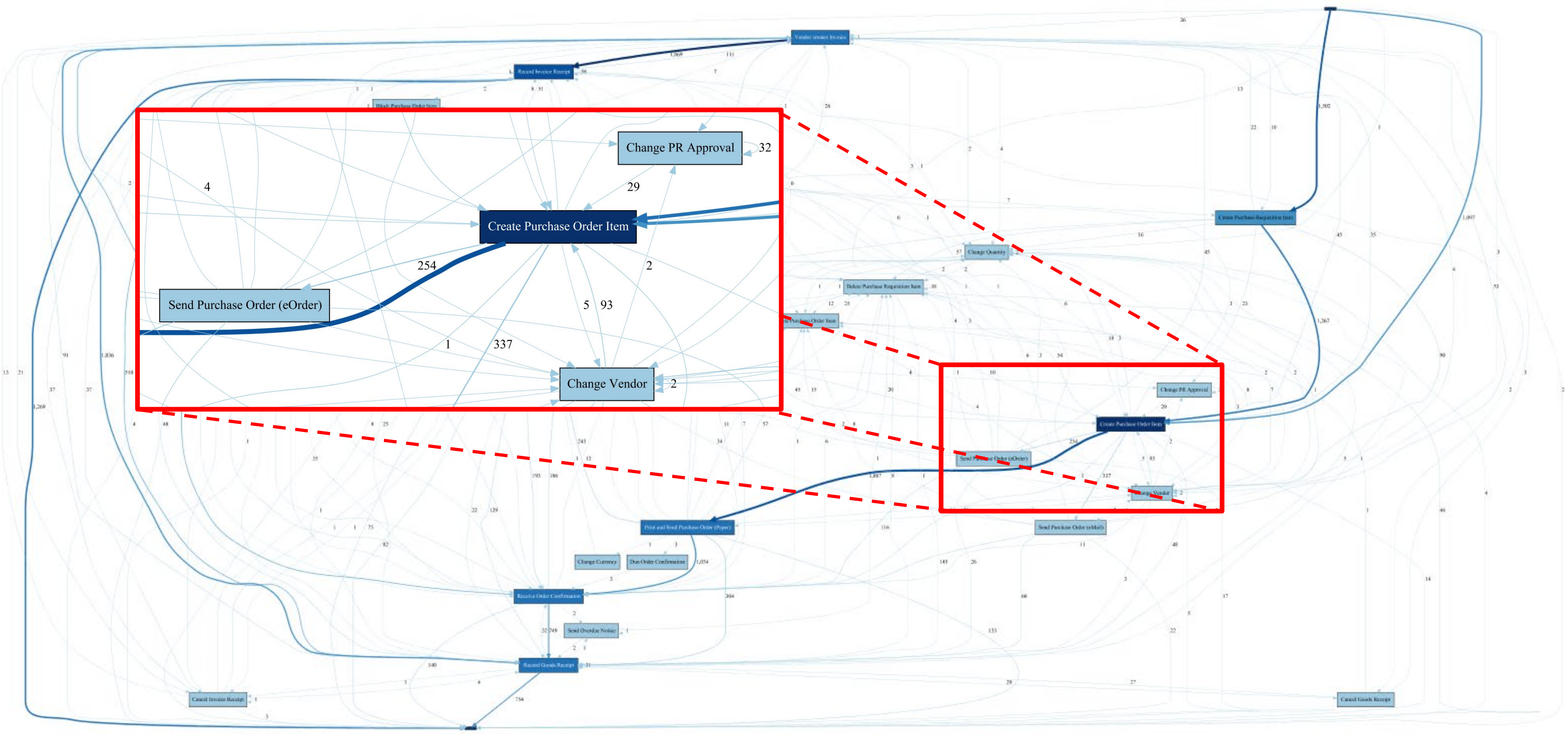



- **Purchase to Pay (P2P).**
- **2654 cases**
- **16226 events**
- **685 variants**
- **24 unique activities**

**Still relatively simple, but …**

Chair of Process and Data Science

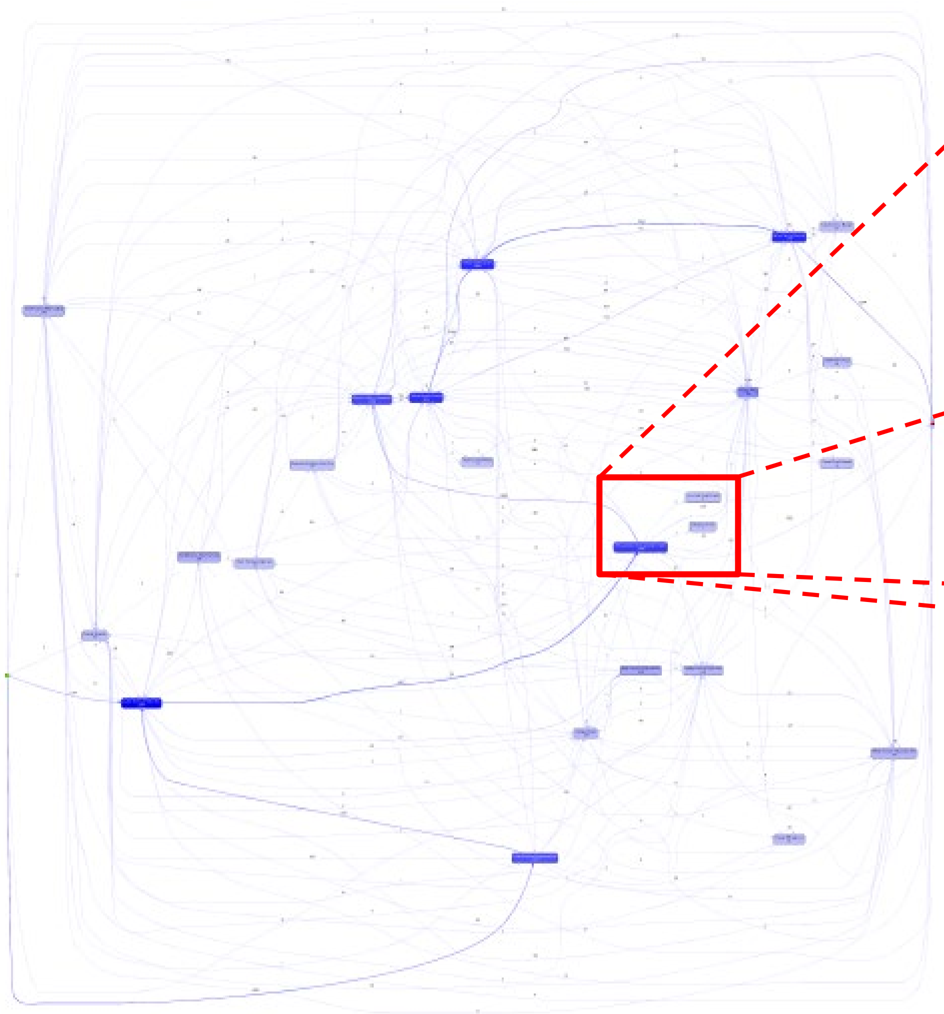
# What if we get Spaghetti instead of Lasagna?

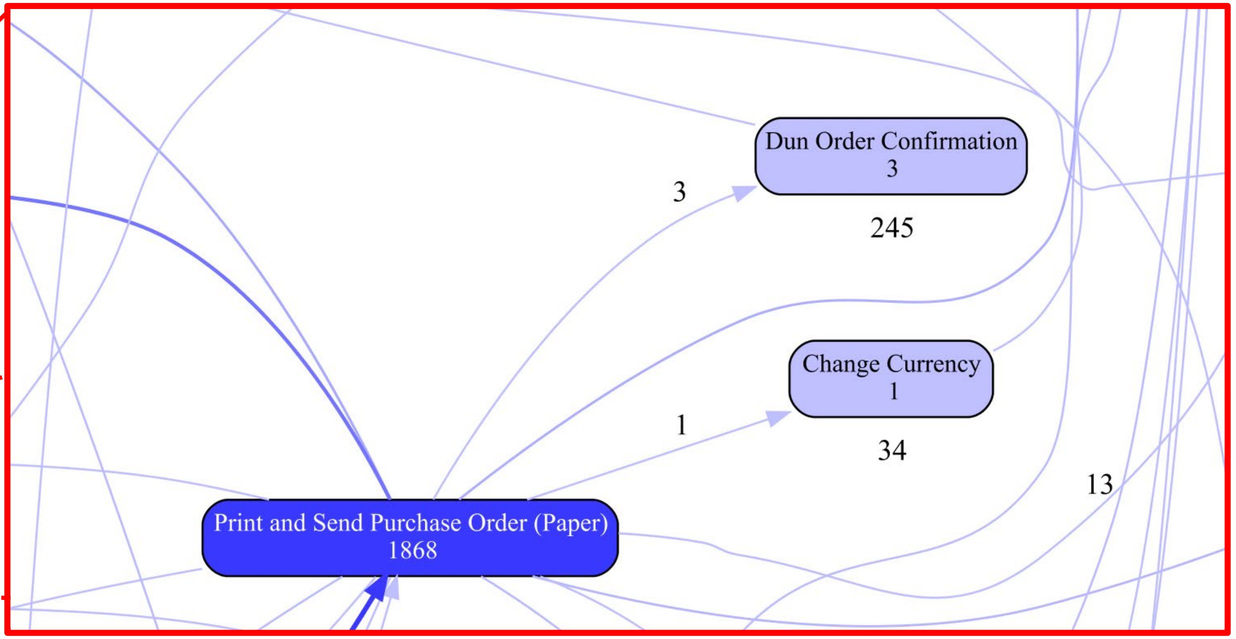



Data-aware heuristic miner plug-in

Chair of Process and Data Science

# What if we get Spaghetti instead of Lasagna?



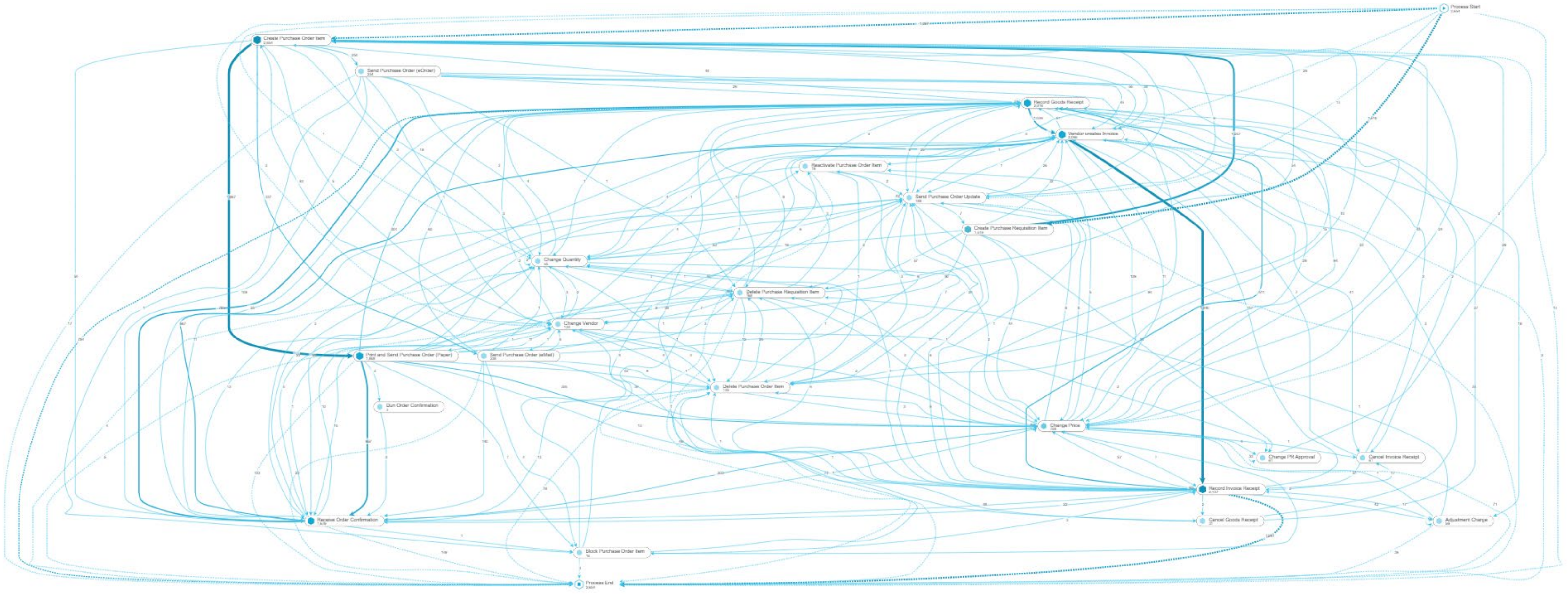



Directly-follows visual miner



Chair of Process and Data Science

# What if we get Spaghetti instead of Lasagna?




celonis

Celonis process explorer

Chair of Process and Data Science

# Filtering

Chair of Process and Data Science

RWTH AACHEN UNIVERSITY

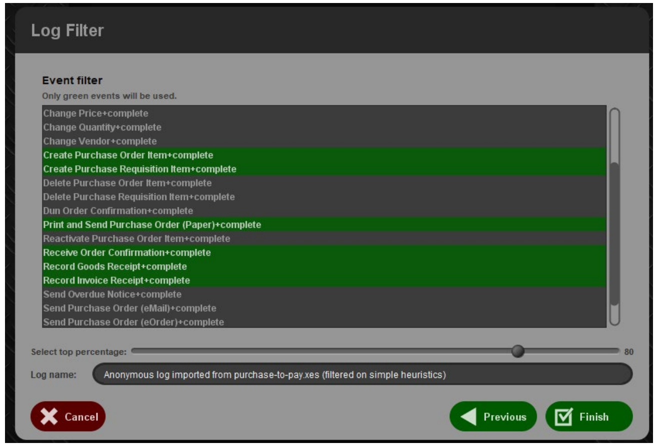
# Filtering

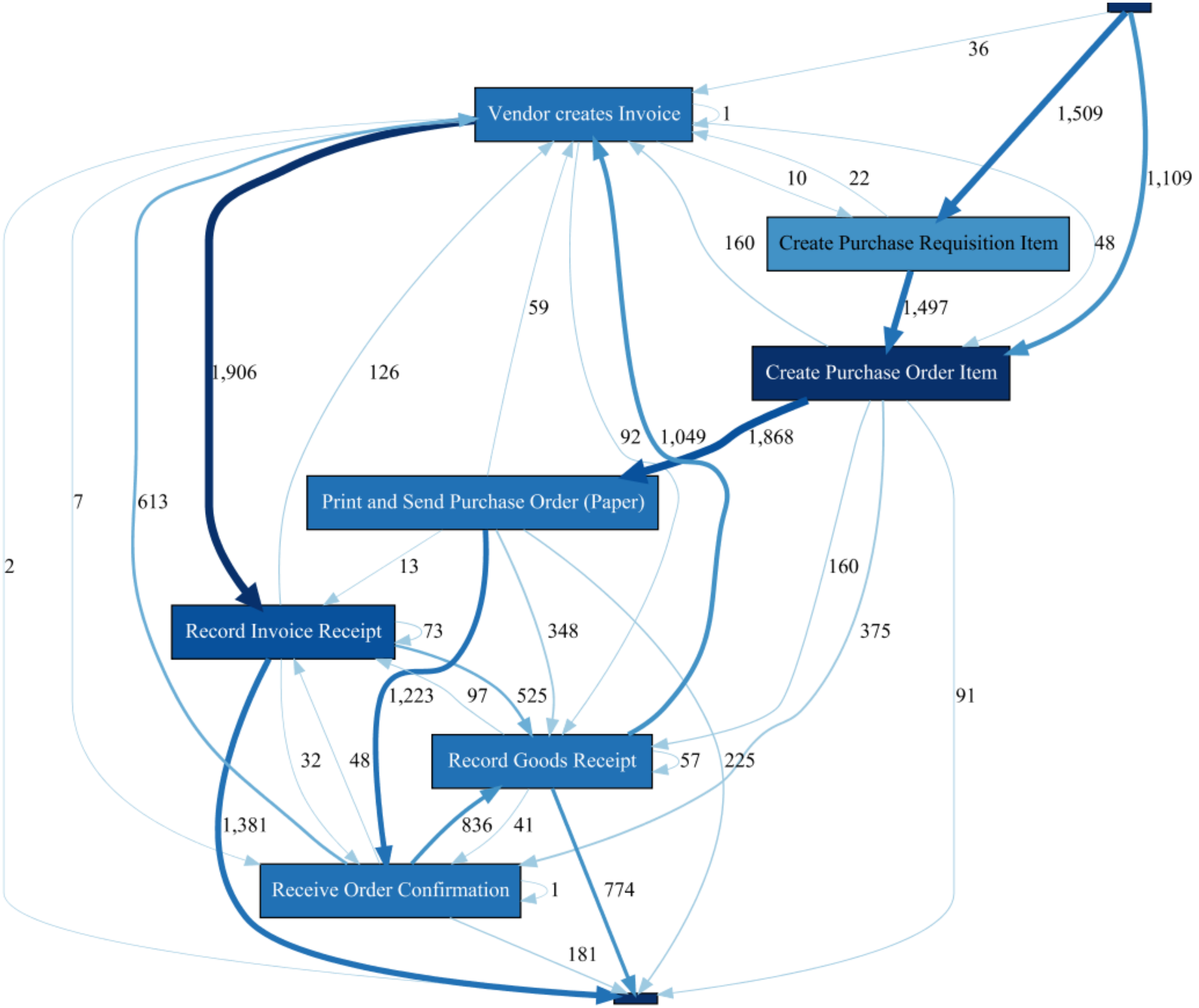
- **Activity-based filtering**: Rank the activities (e.g., based on frequency) and remove lower-ranked activities completely from your data.
- **Variant-based filtering**: Rank the variants (e.g., based on frequency) and remove lower-ranked variants. A variant is simply a sequence of activities and may occur multiple times.
- **Arc-based filtering** (not recommended!): Delete arcs in the DFG (e.g., based on frequency).



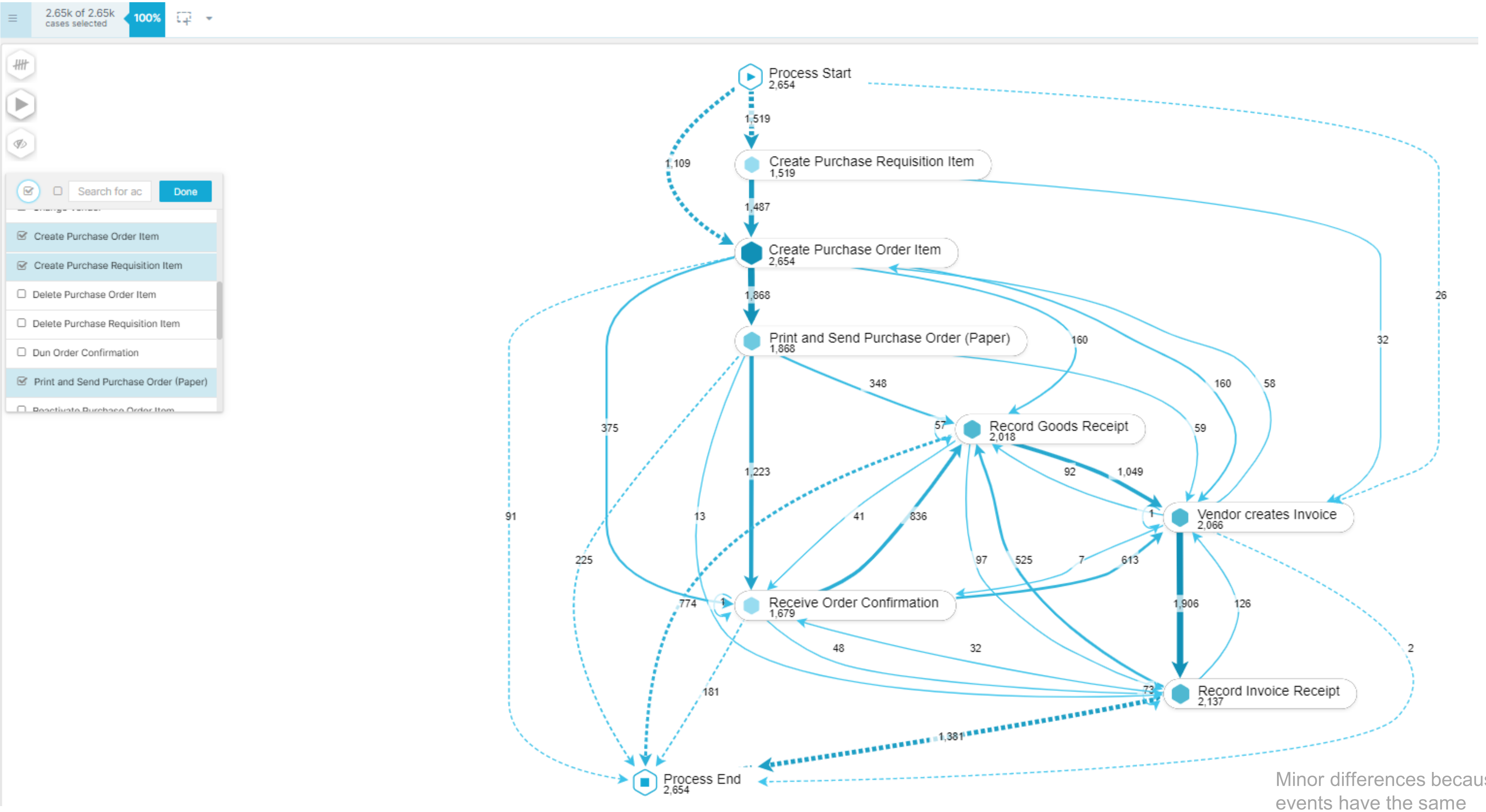
Chair of Process and Data Science

# Activity-based filtering (top 7 of 24 activities)



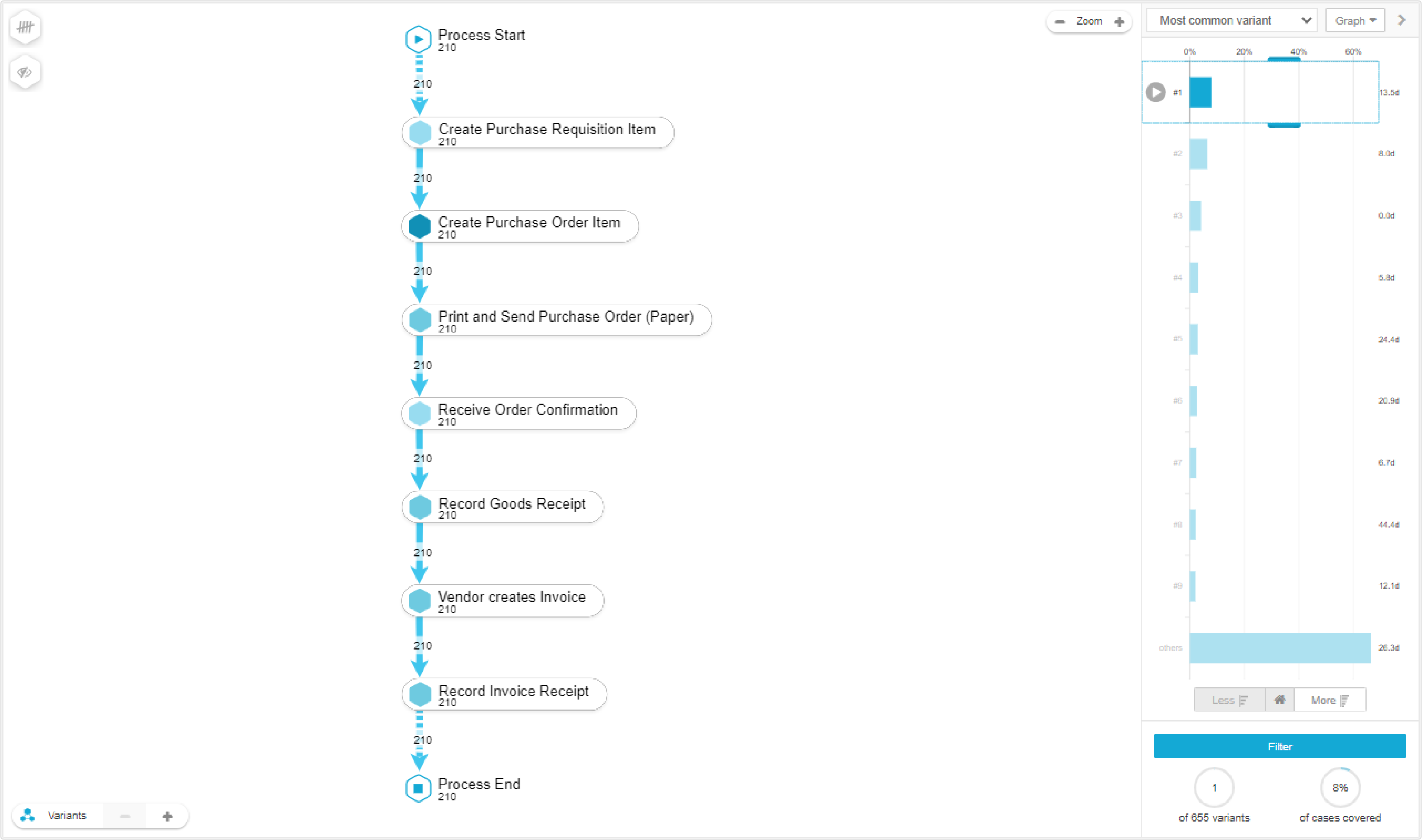



# Activity-based filtering (top 7 of 24 activities)




Minor differences because events have the same timestamp (date only)

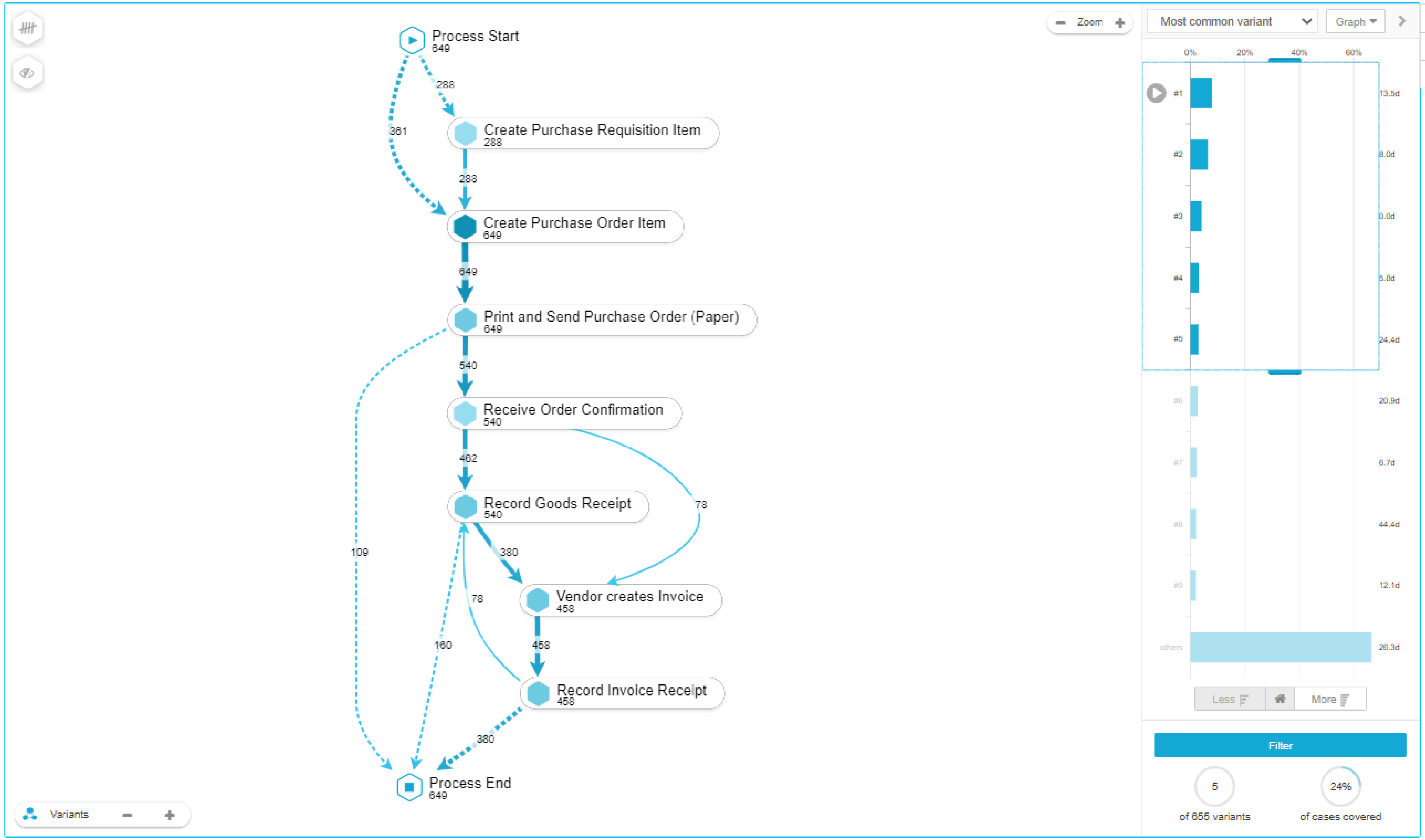
# Variant-based filtering (most frequent variant only)




celonis

Chair of Process and Data Science

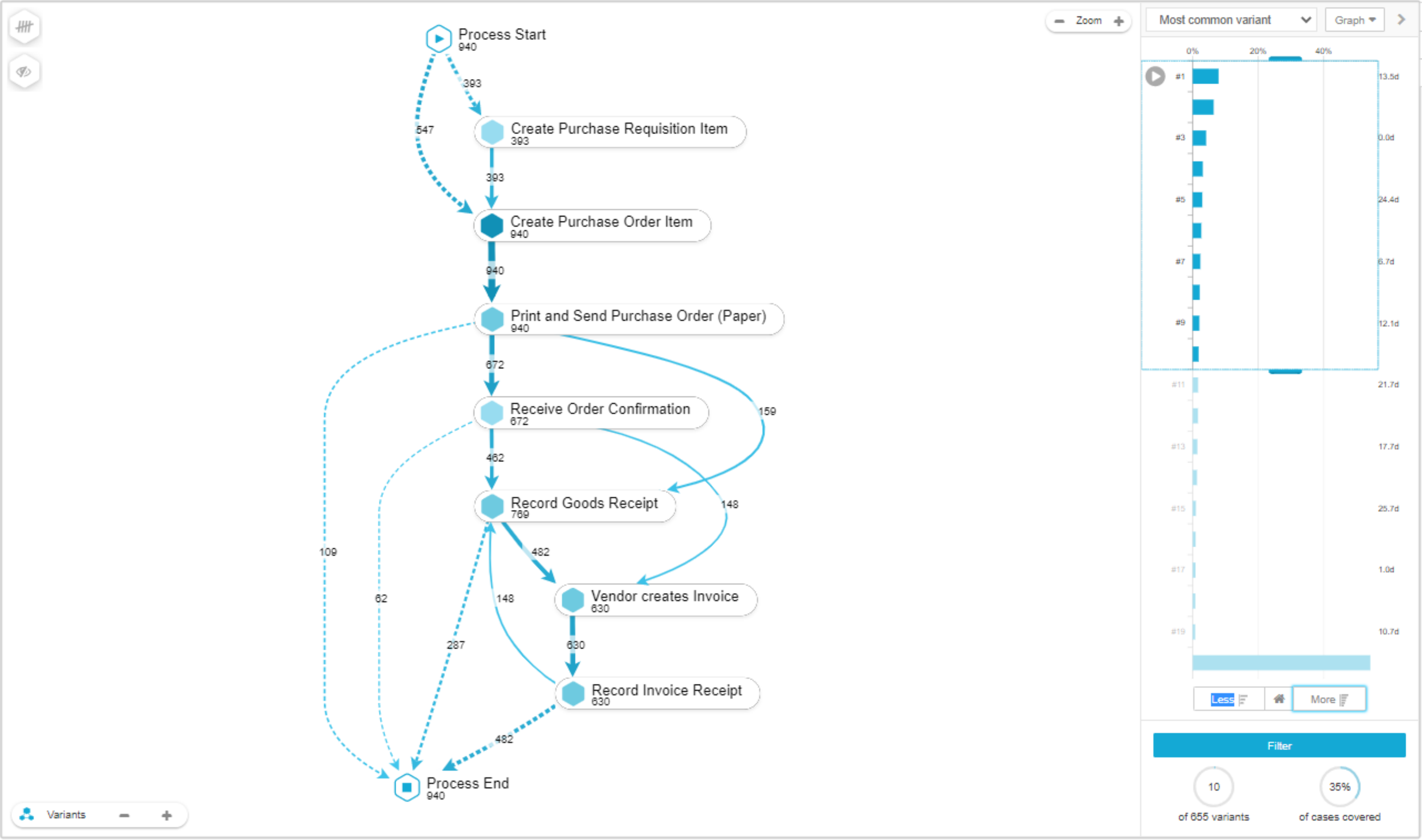
# Variant-based filtering (top 5 variants)






Chair of Process and Data Science

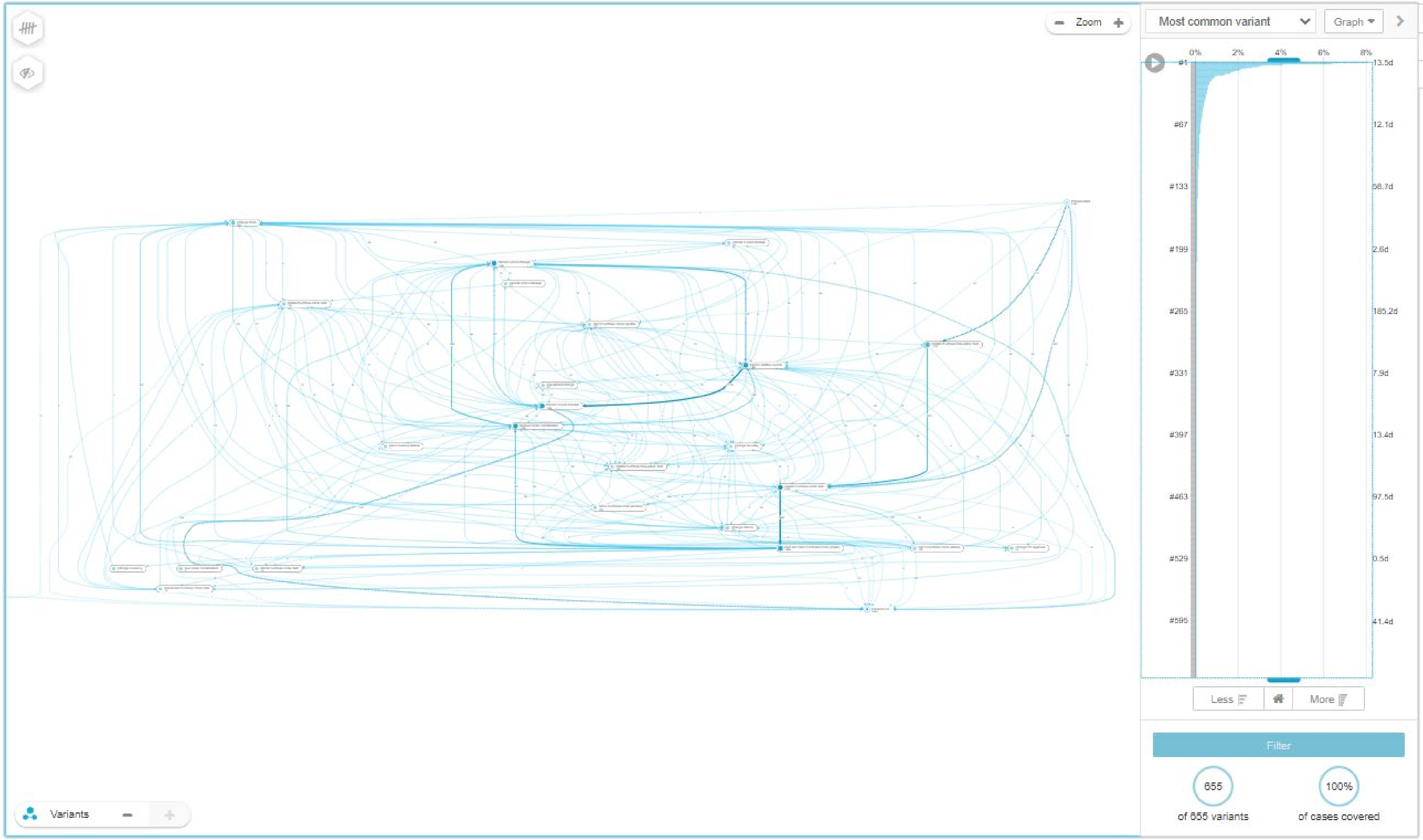
# Variant-based filtering (top 10 variants)




celonis

Chair of Process and Data Science

# Variant-based filtering (all 655 variants)





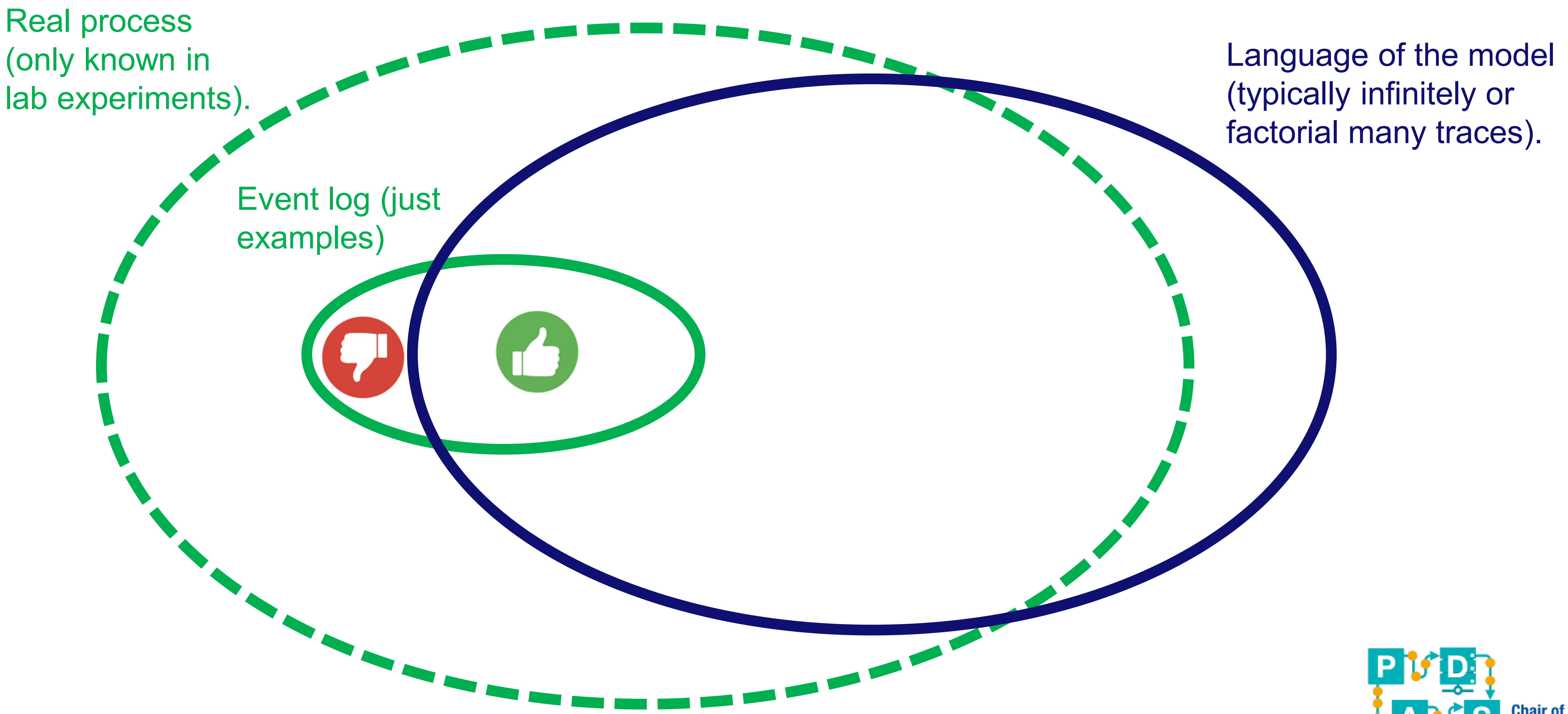

Chair of Process and Data Science

# Challenges

# Challenges

- If the model allows for a loop, we have infinitely many possible traces. This can never be observed!
- The event log just shows examples, the fact that something did not happen does not mean it cannot.
- We do not have negative traces, i.e., it is not a classification problem.
- Hence, precision and recall cannot be defined in the usual manner.

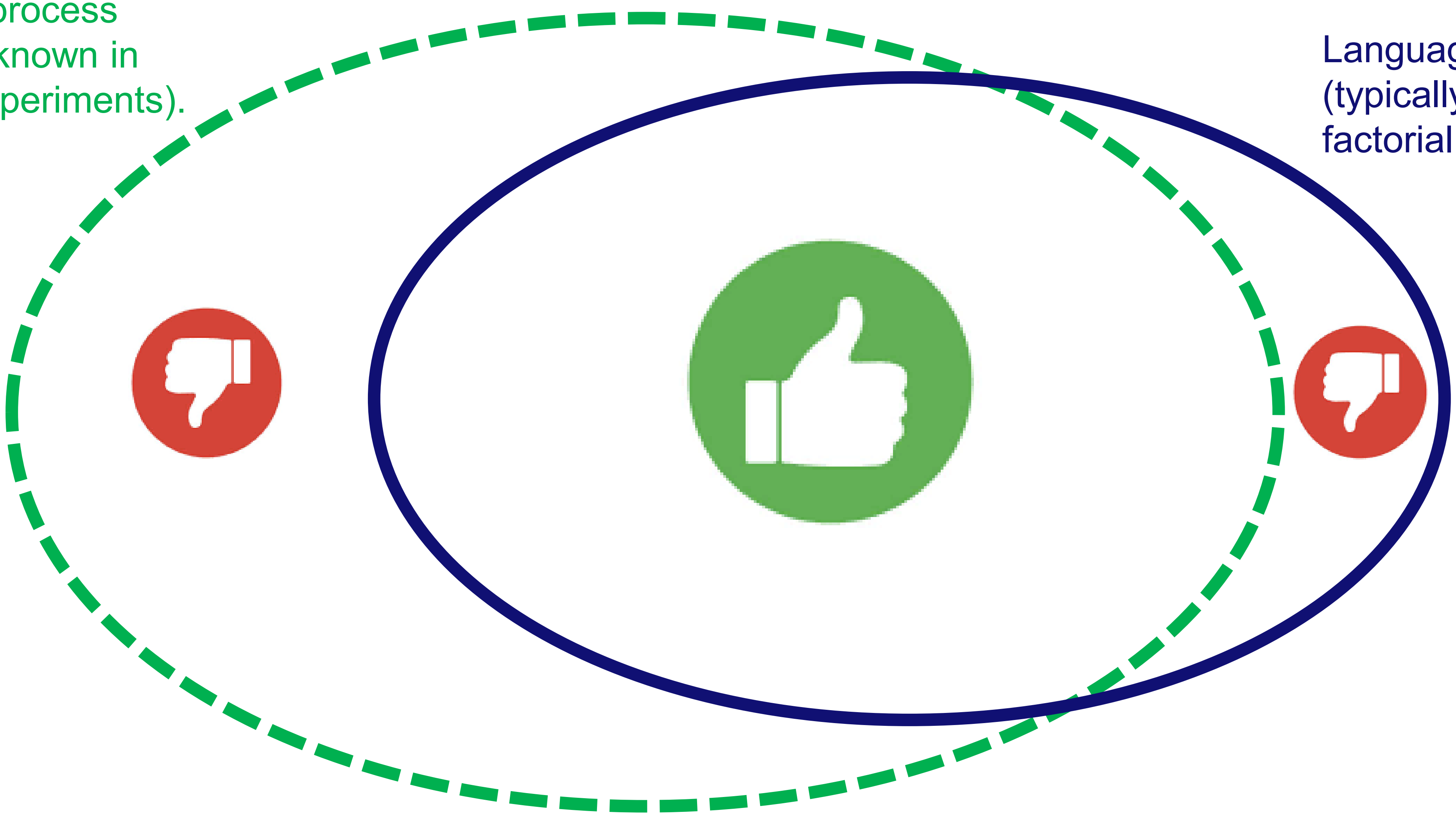
# Visualizing the challenges






Chair of Process and Data Science

# What we would like to know, but cannot know



Real process (only known in lab experiments).

Language of the model (typically infinitely or factorial many traces).

# Therefore, there are many approximations (often using proxies)

See later lectures!

- **Replay fitness** (using the fraction of fitting traces on the event log, token-based, or alignment based).
- **Precision** (e.g., escaping edges).
- **Simplicity** (e.g., number of arcs).
- **Generalization** (e.g., likelihood that the next trace will fit given some assumptions about the distribution).

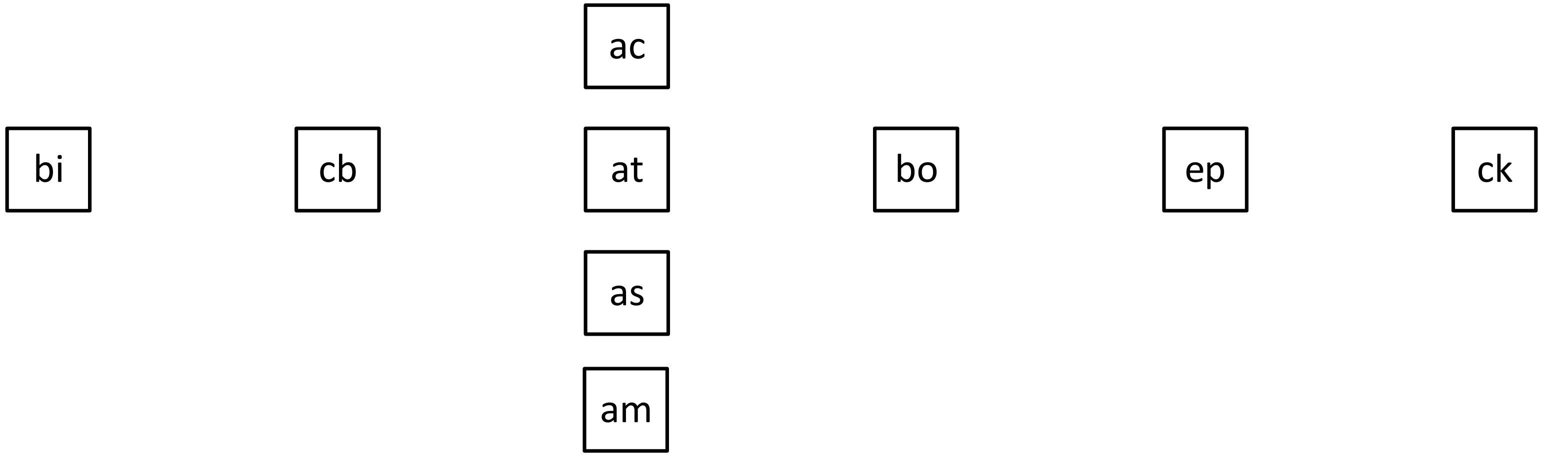
**Check out stochastic conformance checking!**

Sander Leemans, Wil van der Aalst, Tobias Brockhoff, Artem Polyvyanyy: Stochastic process mining: Earth movers' stochastic conformance. Inf. Syst. 102: 101724 (2021)

# Bottom-up discovery

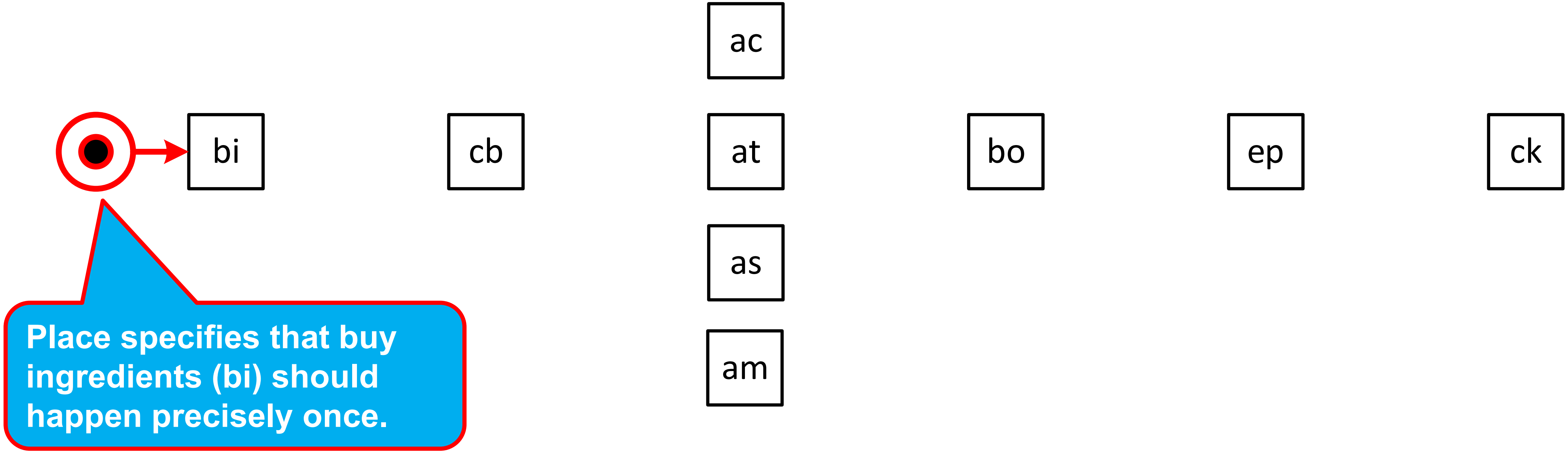
# Bottom-up discovery

- **Assume that anything is possible.**
- **Start adding constraints supported by the data.**
- **A Petri net place is a constraint.**
- **Accepting Petri-nets are surprisingly declarative.**




Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).

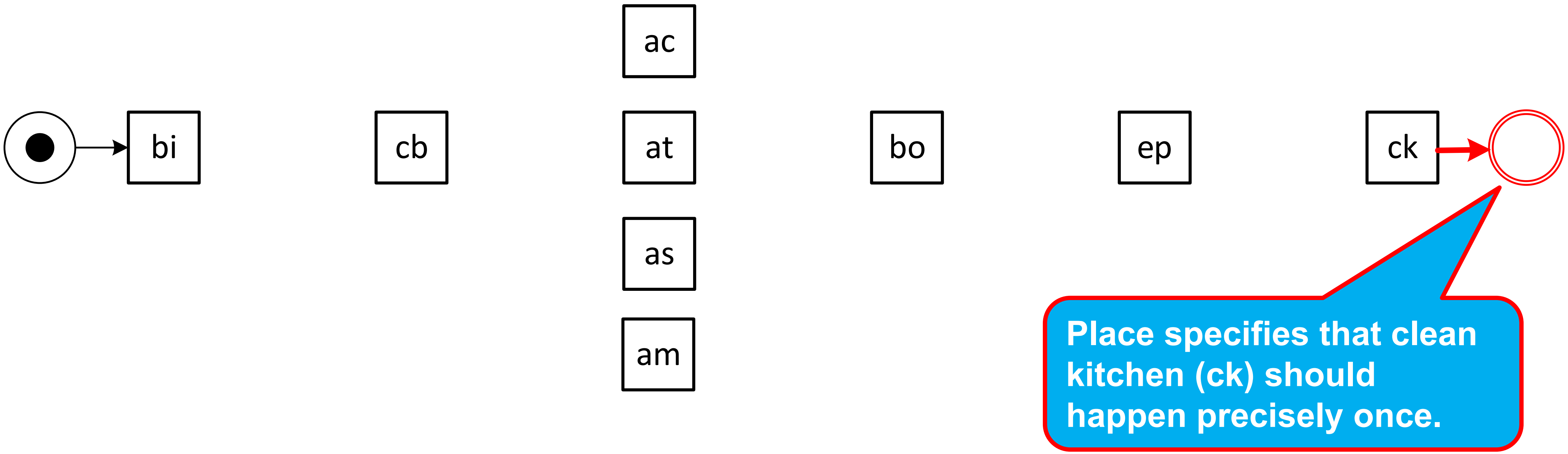
# Places as constraints




An accepting Petri net has an initial and final marking (here the final marking is [ ], i.e., no tokens).

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).

# Places as constraints

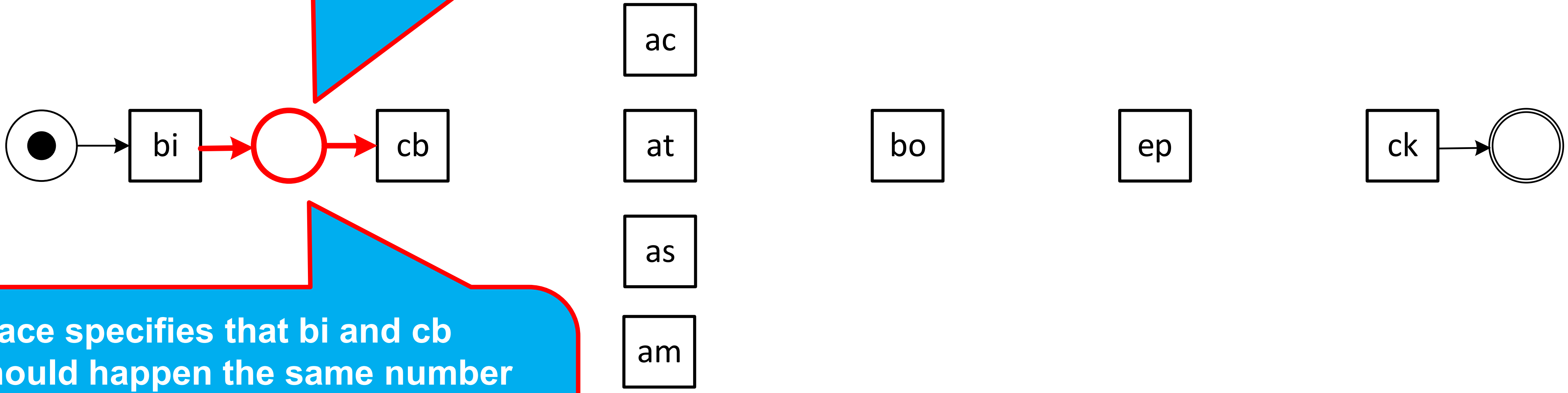



An accepting Petri net has an initial and final marking (here the final marking has one token in sink place).

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).

# Places as constraints

- **#bi = #cb at the end of each case**
- **#bi ≥ #cb at any point in time**




**Place specifies that bi and cb should happen the same number of times. Moreover, at any stage the number of occurrences of cb should not exceed the number of occurrences of bi.**

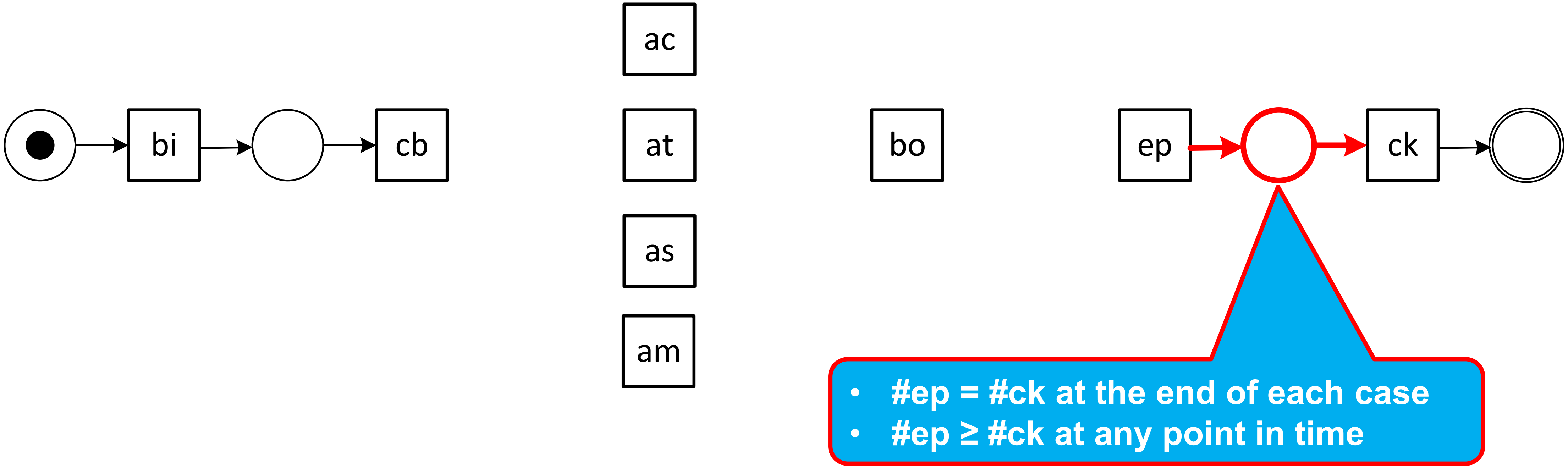
A place defines a local constraint.

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).



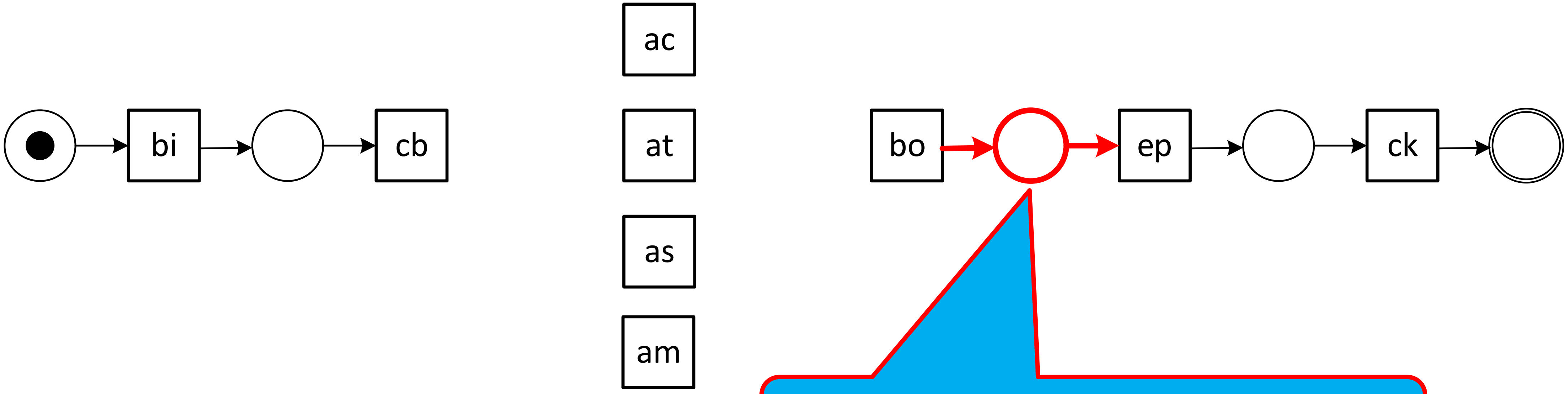
Chair of Process and Data Science

# Places as constraints




Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).

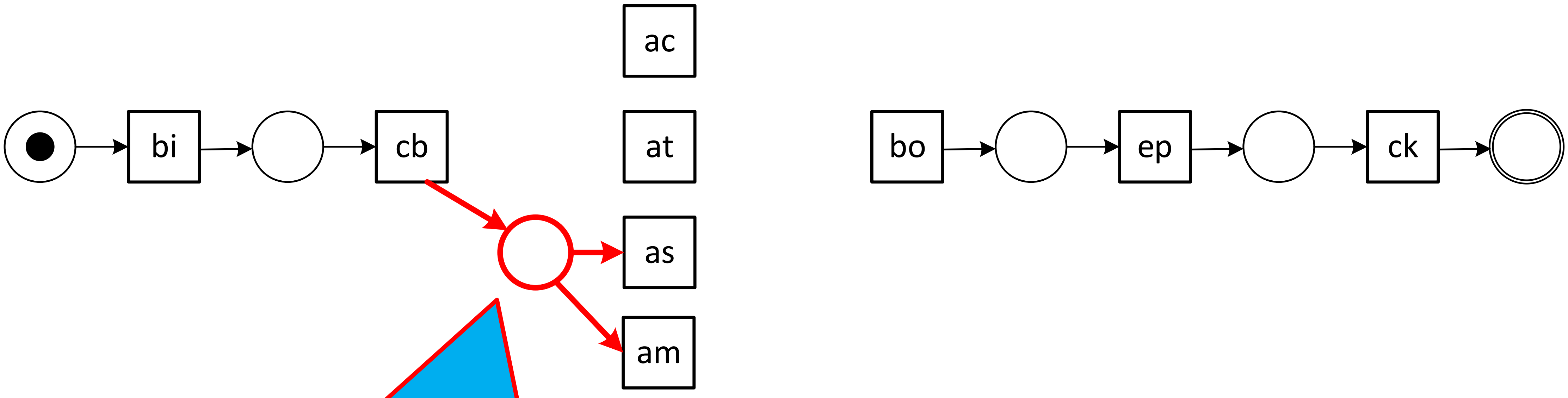
# Places as constraints




- **#bo= #ep at the end of each case**
- **#bo ≥ #ep at any point in time**

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).

# Places as constraints



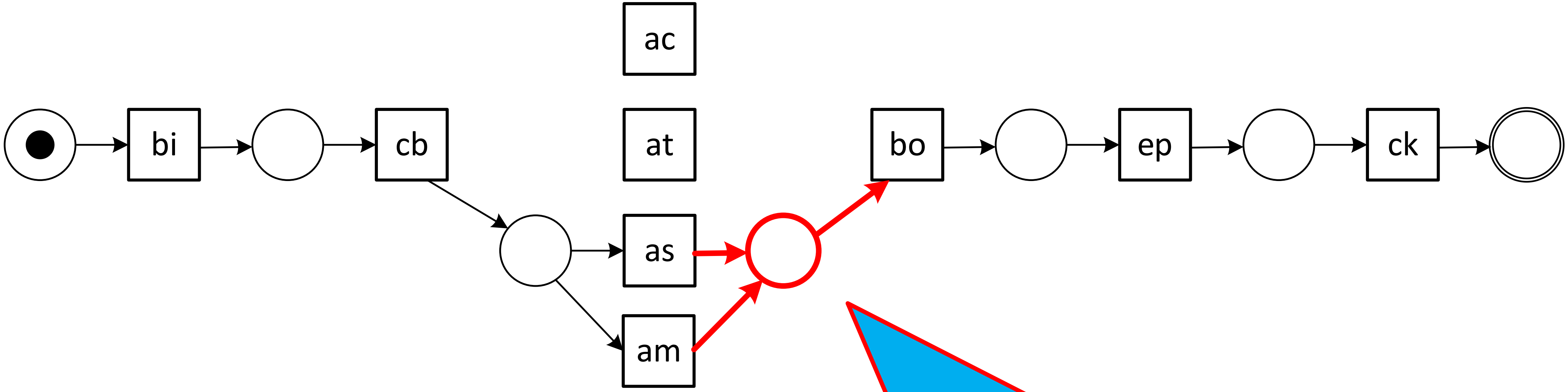

- #cb = #as+#am at the end of each case
- #cb ≥ #as+#am at any point in time

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).



Chair of Process and Data Science

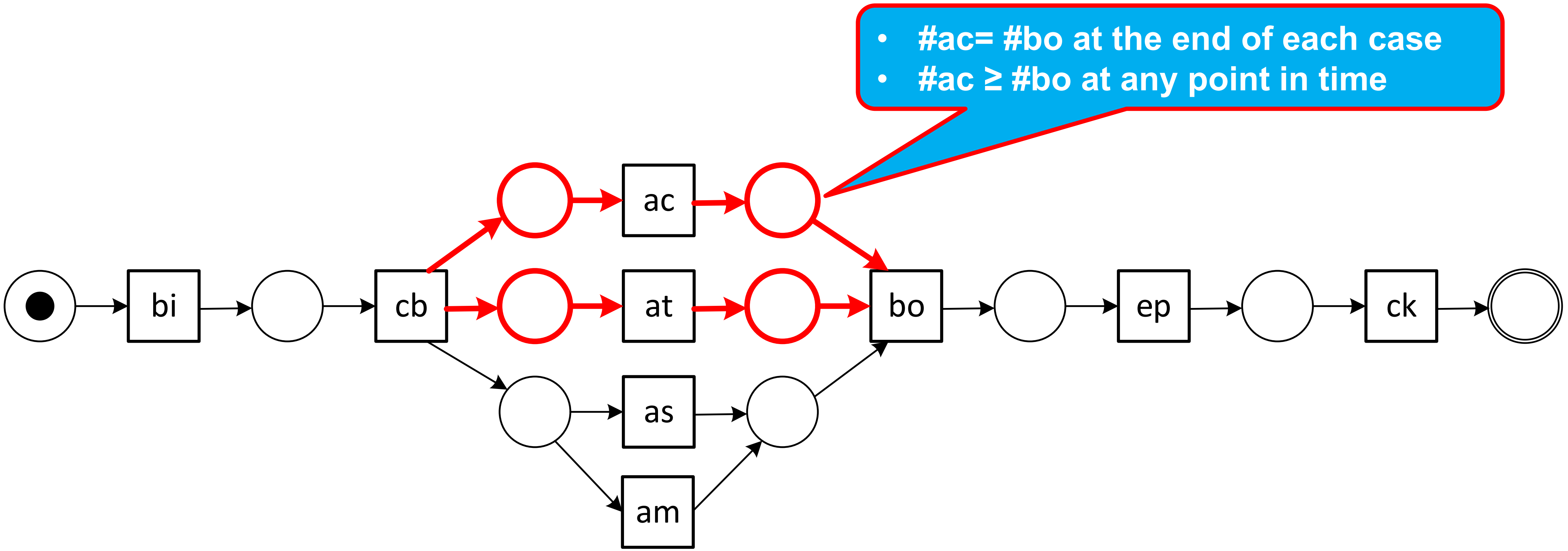
# Places as constraints




- **#as+#am = #bo at the end of each case**
- **#as+#am ≥ #bo any point in time**

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).

# Places as constraints

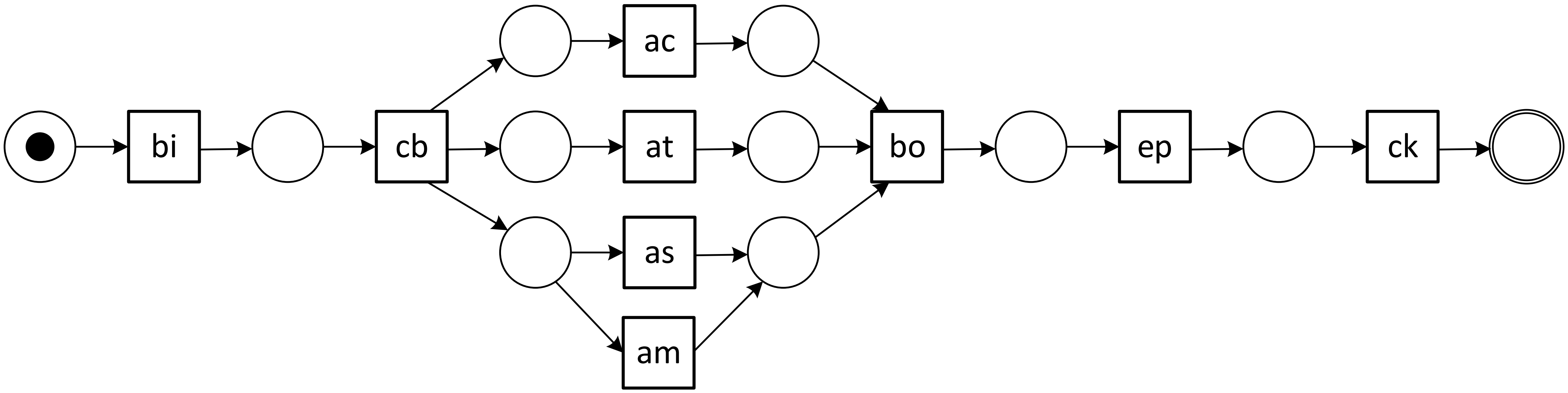



Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).



Chair of Process and Data Science

# Final accepting Petri net




Also every intermediate model was an accepting Petri net!

Bottom-up process discover uses this locality principle!

Using short names: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).



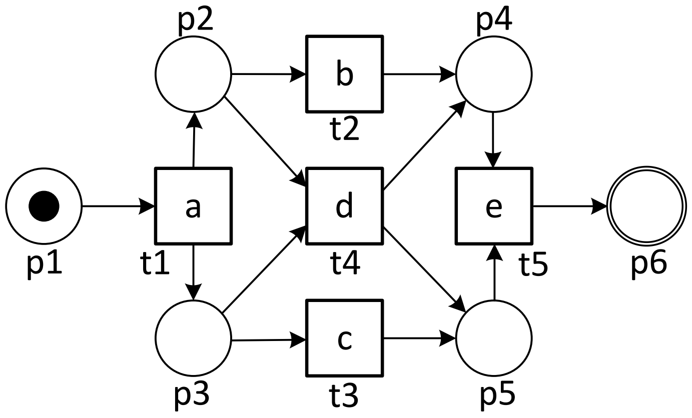
Chair of Process and Data Science

# Accepting Petri Nets

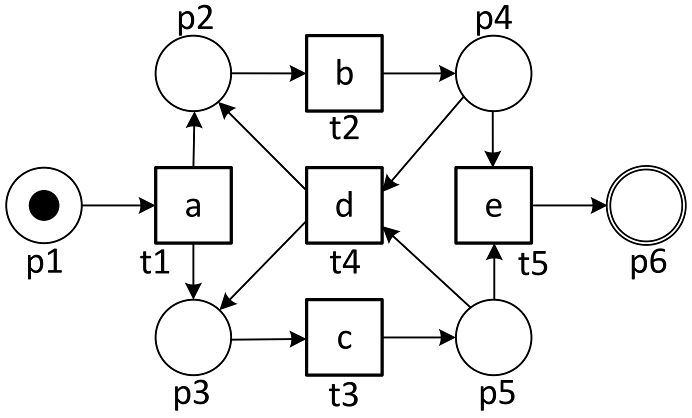


Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

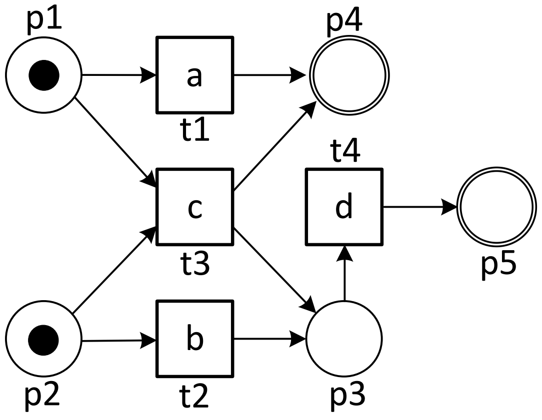
# Examples of accepting Petri nets



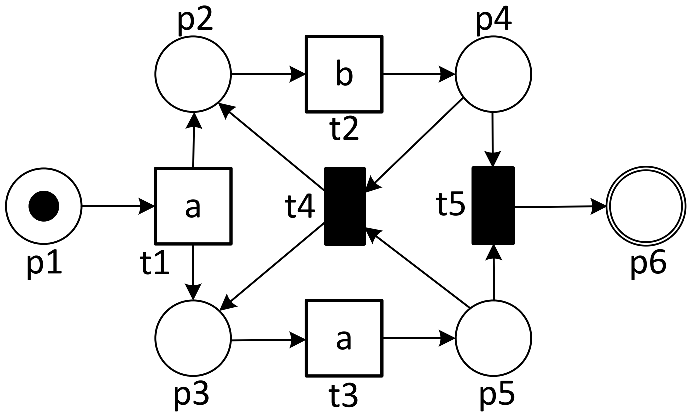

(a) $AN_1 = (N_1,[p1],[p6])$




(b) $AN_2 = (N_2,[p1],[p6])$




(c) $AN_3 = (N_3,[p1,p2],[p4,p5])$




(d) $AN_4 = (N_4,[p1],[p6])$

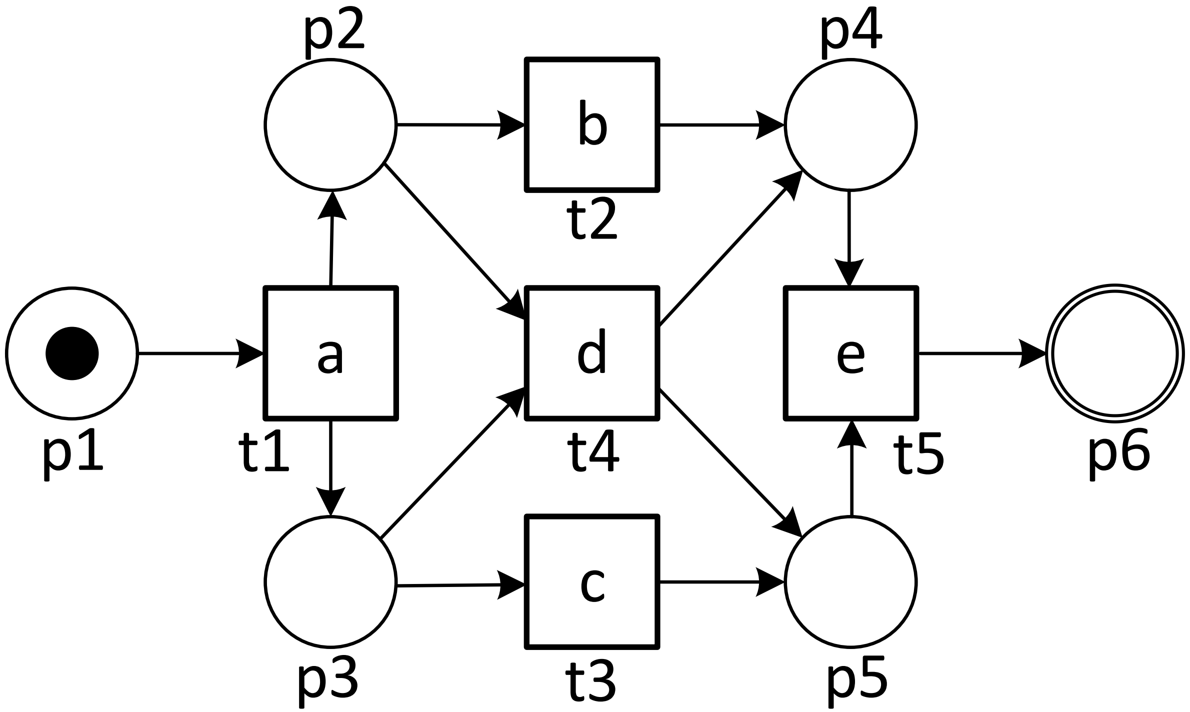
**Definition 14 (Accepting Petri Net).** *An accepting Petri net is a triplet* $AN = (N, M_{init}, M_{final})$ *where* $N = (P, T, F, l)$ *is a labeled Petri net,* $M_{init} \in \mathcal{B}(P)$ *is the initial marking, and* $M_{final} \in \mathcal{B}(P)$ *is the final marking.* $\mathcal{U}_{AN} \subseteq \mathcal{U}_M$ *is the set of accepting Petri nets.*

- **Initial and final marking.**
- **Labeled transitions to refer to activities.**
- **Allows for transitions with the same or no label.**



Chair of Process and Data Science

# Example of an accepting Petri net and its language (1/4)



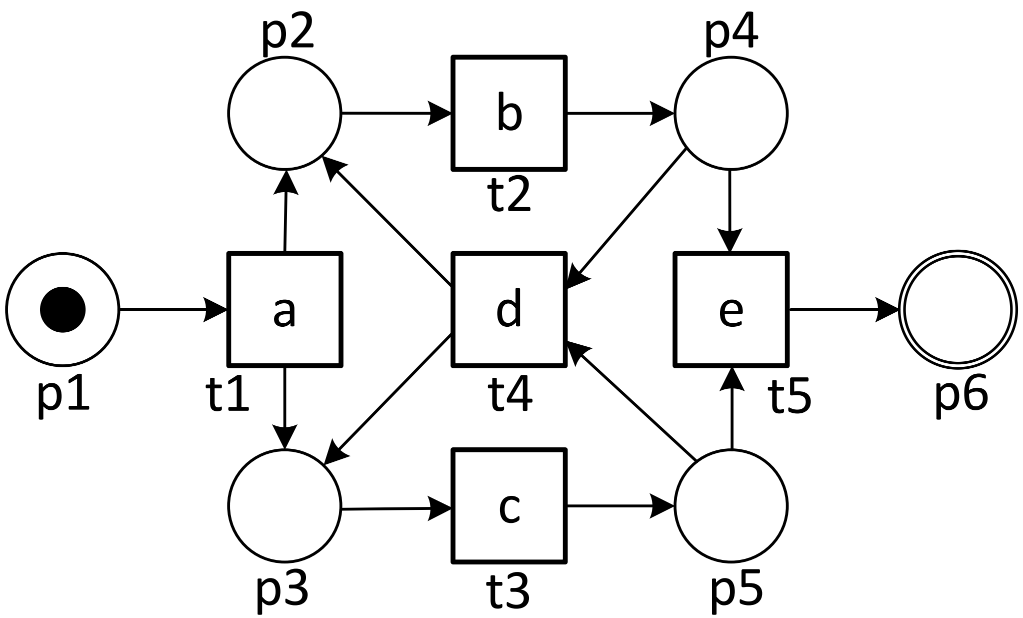

(a) $AN_1 = (N_1,[p1],[p6])$

$$lang(AN_1) = \{\langle a, b, c, e\rangle, \langle a, c, b, e\rangle, \langle a, d, e\rangle\}$$



Chair of Process and Data Science

# Example of an accepting Petri net and its language (2/4)



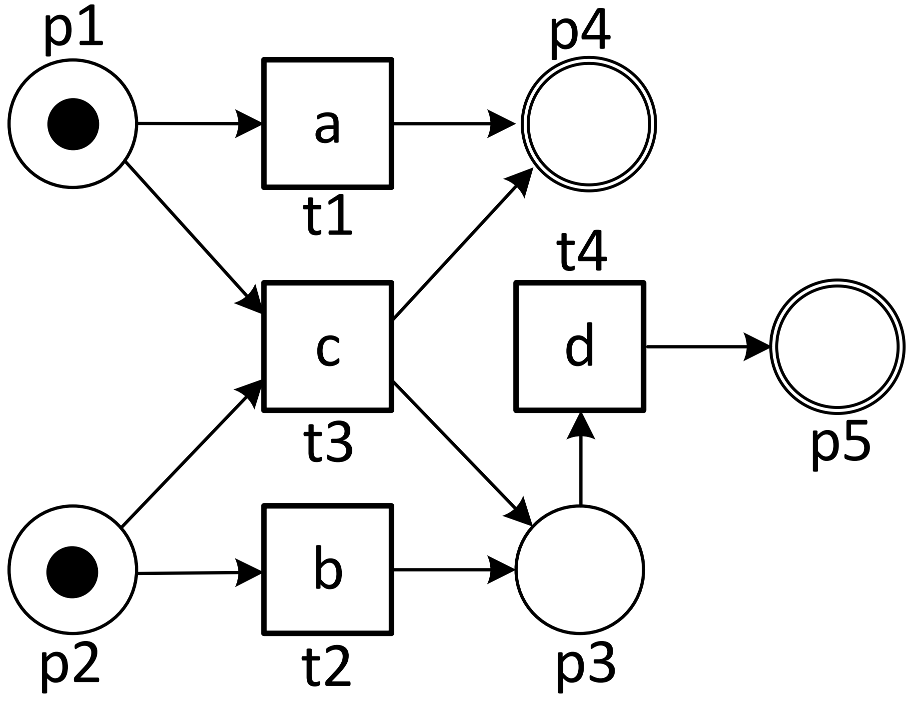

(b) $AN_2 = (N_2,[p1],[p6])$

$$lang(AN_2) = \{\langle a,b,c,e\rangle, \langle a,c,b,e\rangle, \langle a,b,c,d,b,c,e\rangle, \langle a,c,b,d,b,c,e\rangle, \ldots, \langle a,c,b,d,b,c,d,c,b,d,c,b,e\rangle, \ldots\}$$



Chair of Process and Data Science

# Example of an accepting Petri net and its language (3/4)

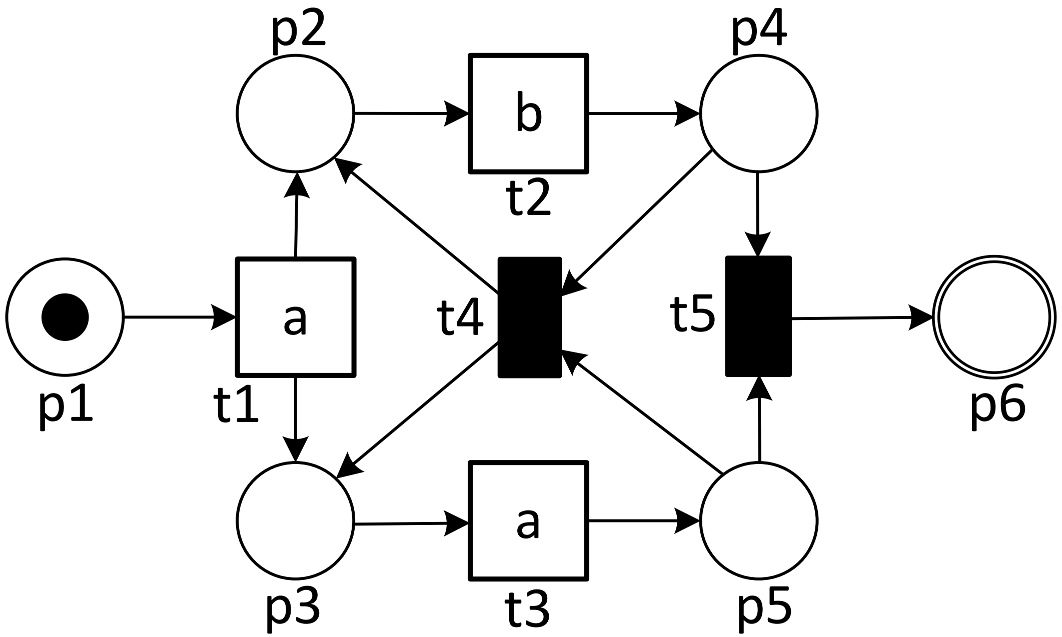



(c) $AN_3 = (N_3,[p1,p2],[p4,p5])$

Initial and final markings may refer to multiple tokens and places.

$$lang(AN_3) = \{\langle a,b,d\rangle, \langle b,a,d\rangle, \langle b,d,a\rangle, \langle c,d\rangle\}$$

# Example accepting Petri net and its language (4/4)




(d) $AN_4 = (N_4,[p1],[p6])$

Two transitions have the same label and two are silent.

$$lang(AN_4) = \{\langle a,b,a\rangle, \langle a,a,b\rangle, \langle a,b,a,b,a\rangle, \langle a,a,b,b,a\rangle, \ldots, \langle a,a,b,b,a,a,b,a,b\rangle, \ldots\}$$

# Accepting Petri nets & process mining

- A lot of powerful analysis techniques exist for accepting Petri nets.
- For example, alignments are based on this.
- We can map the relevant subsets of BPMN, process trees, etc. onto accepting Petri nets.
- No need to restrict to workflow nets or transition with unique visible labels.
- Surprisingly declarative!!

# Alpha Algorithm

# Just eight lines of mathematics, based on the DFG created before

**Definition 22 (Alpha Algorithm).** *The alpha algorithm* $disc_{alpha} \in \mathcal{B}(\mathcal{U}_{act}{}^*) \rightarrow \mathcal{U}_{AN}$ *returns an accepting Petri net* $disc_{alpha}(L)$ *for any event log* $L \in \mathcal{B}(\mathcal{U}_{act}{}^*)$. *Let* $A = act(L)$ *and* $fp(L) = fp(disc_{DFG}(L))$ *the footprint of event log* $L$. *This allows us to write* $a_1 \rightarrow_L a_2$ *if* $fp(L)((a_1, a_2)) = \rightarrow$ *and* $a_1 \#_L a_2$ *if* $fp(L)((a_1, a_2)) = \#$ *for any* $a_1, a_2 \in A' = A \cup \{\blacktriangleright, \blacksquare\}$.

1. $Cnd = \{(A_1, A_2) \mid A_1 \subseteq A' \wedge A_1 \neq \emptyset \wedge A_2 \subseteq A' \wedge A_2 \neq \emptyset \wedge \forall_{a_1 \in A_1} \forall_{a_2 \in A_2}\ a_1 \rightarrow_L a_2 \wedge \forall_{a_1, a_2 \in A_1}\ a_1 \#_L a_2 \wedge \forall_{a_1, a_2 \in A_2}\ a_1 \#_L a_2\}$ *are the candidate places,*
2. $Sel = \{(A_1, A_2) \in Cnd \mid \forall_{(A_1', A_2') \in Cnd}\ A_1 \subseteq A_1' \wedge A_2 \subseteq A_2' \implies (A_1, A_2) = (A_1', A_2')\}$ *are the selected maximal places,*
3. $P = \{p_{(A_1, A_2)} \mid (A_1, A_2) \in Sel\} \cup \{p_\blacktriangleright, p_\blacksquare\}$ *is the set of all places,*
4. $T = \{t_a \mid a \in A'\}$ *is the set of transitions,*
5. $F = \{(t_a, p_{(A_1, A_2)}) \mid (A_1, A_2) \in Sel \wedge a \in A_1\} \cup \{(p_{(A_1, A_2)}, t_a) \mid (A_1, A_2) \in Sel \wedge a \in A_2\} \cup \{(p_\blacktriangleright, t_\blacktriangleright), (t_\blacksquare, p_\blacksquare)\}$ *is the set of arcs,*
6. $l = \{(t_a, a) \mid a \in A\}$ *is the labeling function,*
7. $M_{init} = [p_\blacktriangleright]$ *is the initial marking,* $M_{final} = [p_\blacksquare]$ *is the final marking, and*
8. $disc_{alpha}(L) = ((P, T, F, l), M_{init}, M_{final})$ *is the discovered accepting Petri net.*

The presentation is different from the original algorithm, but in essence it is the same.

- We add an artificial start and end to overcome the usual problems.
- Also it builds on the DFG and any tool can produce this!
- We can filter before.



Chair of Process and Data Science

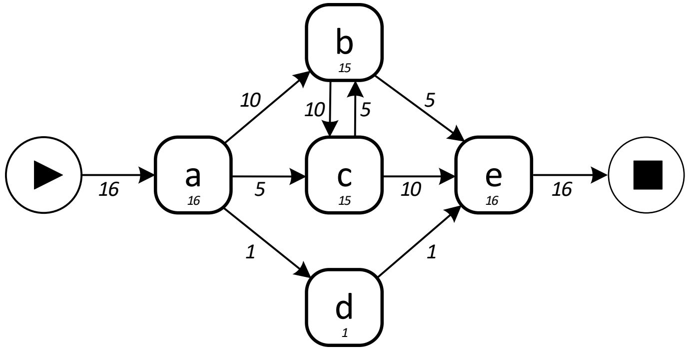
# Remember DFGs

**Definition 6 (Baseline Discovery Algorithm).** *Let* $L \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$ *be an event log.* $disc_{DFG}(L) = (A, F)$ *is the DFG based on* $L$ *with:*

- $A = \{a \in \sigma \mid \sigma \in L\}$ *and*
- $F = [(\sigma_i, \sigma_{i+1}) \mid \sigma \in L' \wedge 1 \leq i < |\sigma|]$ *with* $L' = [\langle \blacktriangleright \rangle \cdot \sigma \cdot \langle \blacksquare \rangle \mid \sigma \in L]$.

- **Graph with nodes representing activities and start ► and end ■.**
- **Behavior starts with dummy activity ►and ends with dummy activity ■. Node ► is a source node and ■ is a sink node.**

Can be filtered using one of the three approaches.



# Two relations based on the DFG

- $a_1 \rightarrow_L \ a_2$ means that $a_1$ is connected to $a_2$ in the DFG but not the other way around, i.e., a one-directional arc.
- $a_1 \#_L \ a_2$ means that $a_1$ is not connected to $a_2$ and $a_2$ is not connected to $a_1$.
- Note that notation also applies to start ▶ and end ■.
- $A$ is the set of activities and $A' = A \cup \{▶,■\}$ includes the start and end node.
- $A' = A \cup \{▶,■\}$, $\rightarrow_L$ and $\#_L$ are all we use!



Chair of Process and Data Science

# Step 1: Create candidate places

1. $Cnd = \{(A_1, A_2) \mid A_1 \subseteq A' \wedge A_1 \neq \emptyset \wedge A_2 \subseteq A' \wedge A_2 \neq \emptyset \wedge \forall_{a_1 \in A_1} \forall_{a_2 \in A_2}\, a_1 \rightarrow_L a_2 \wedge \forall_{a_1, a_2 \in A_1}\, a_1 \#_L a_2 \wedge \forall_{a_1, a_2 \in A_2}\, a_1 \#_L a_2\}$ *are the candidate places;*

$\boldsymbol{a_1 \rightarrow_L a_2}$ means that $\boldsymbol{a_1}$ is connected to $\boldsymbol{a_2}$ in the DFG but not the other way around, i.e., a one-directional arc.

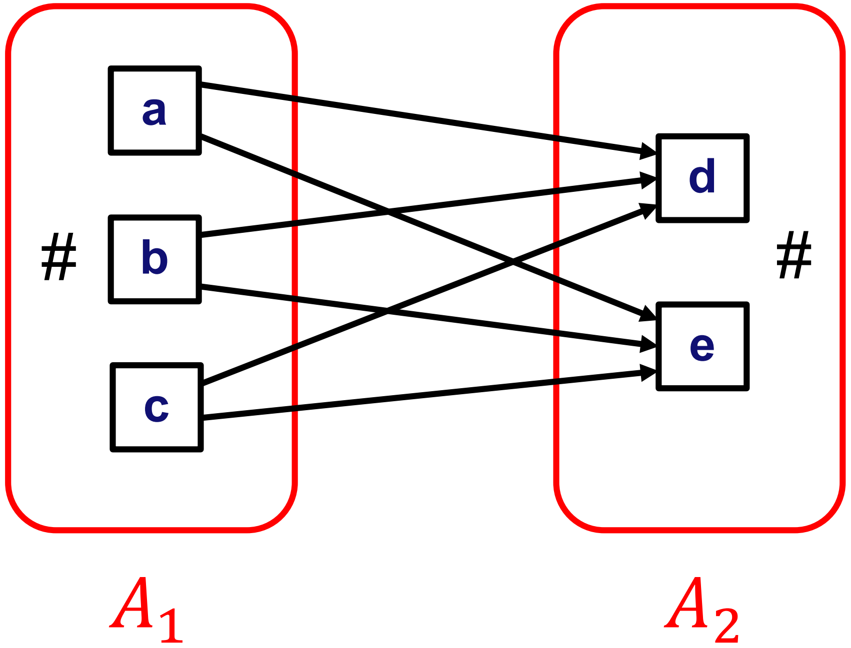
$\boldsymbol{a_1 \#_L a_2}$ means that $\boldsymbol{a_1}$ is not connected to $\boldsymbol{a_2}$ and $\boldsymbol{a_2}$ is not connected to $\boldsymbol{a_1}$.

$\boldsymbol{A}$ is the set of activities and $\boldsymbol{A' = A \cup \{▶, ■\}}$ includes the start and end node.

Chair of Process and Data Science

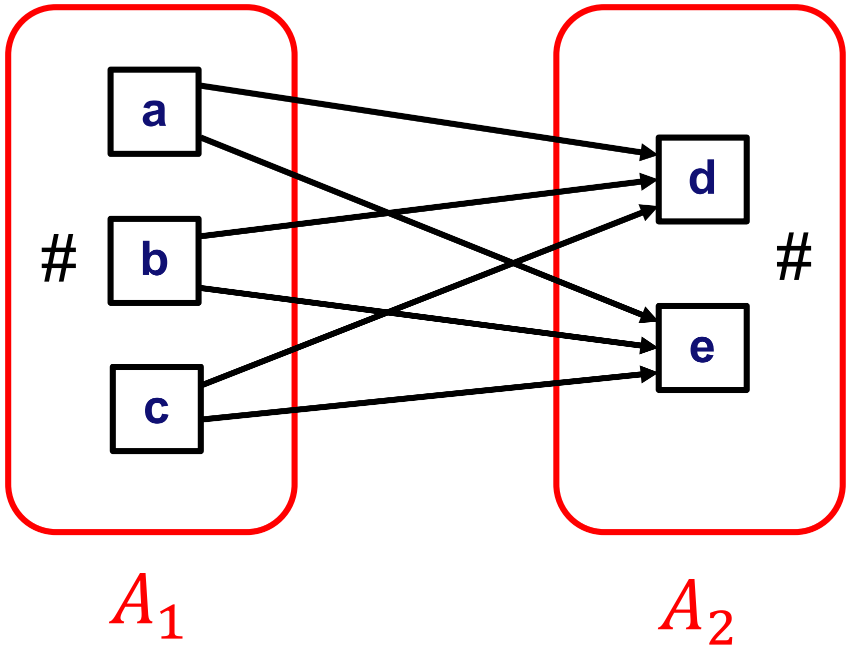
# Step 1: Create candidate places

1. $Cnd = \{(A_1, A_2) \mid A_1 \subseteq A' \land A_1 \neq \emptyset \land A_2 \subseteq A' \land A_2 \neq \emptyset \land \forall_{a_1 \in A_1} \forall_{a_2 \in A_2}\, a_1 \rightarrow_L a_2 \land \forall_{a_1, a_2 \in A_1}\, a_1 \#_L a_2 \land \forall_{a_1, a_2 \in A_2}\, a_1 \#_L a_2\}$ *are the candidate places,*

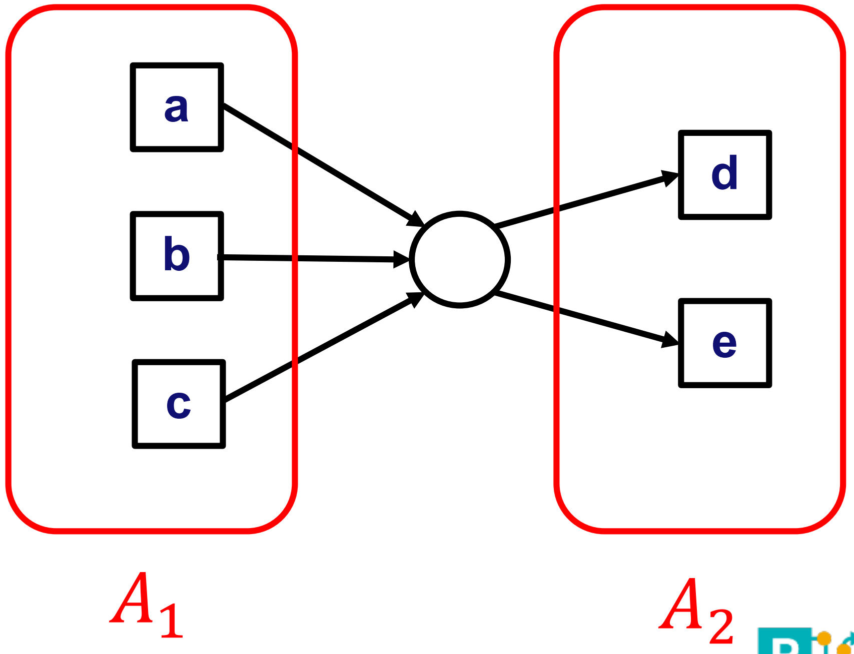


# Step 1: Create candidate places

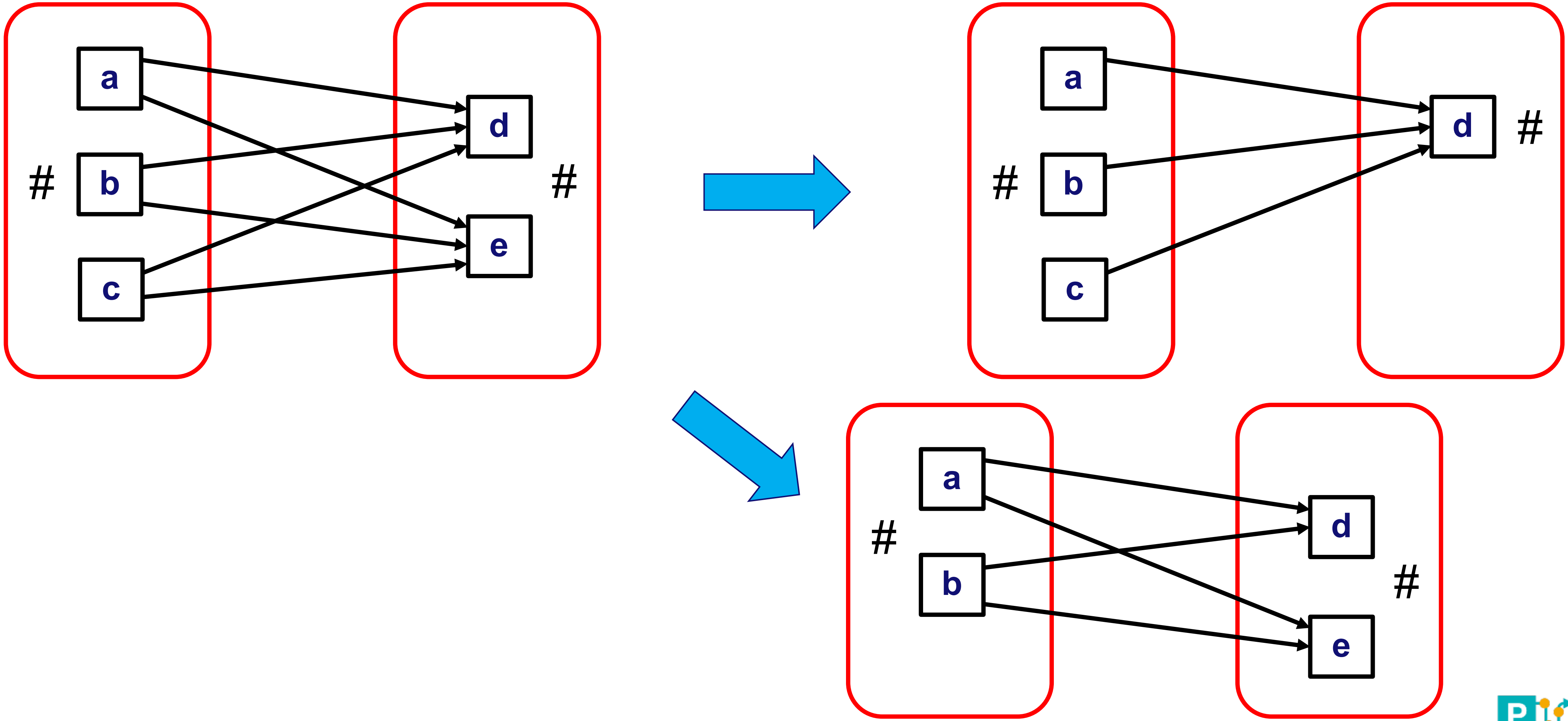
1. $Cnd = \{(A_1, A_2) \mid A_1 \subseteq A' \wedge A_1 \neq \emptyset \wedge A_2 \subseteq A' \wedge A_2 \neq \emptyset \wedge \forall_{a_1 \in A_1} \forall_{a_2 \in A_2}\; a_1 \rightarrow_L a_2 \wedge \forall_{a_1, a_2 \in A_1}\; a_1 \#_L a_2 \wedge \forall_{a_1, a_2 \in A_2}\; a_1 \#_L a_2\}$ *are the candidate places,*



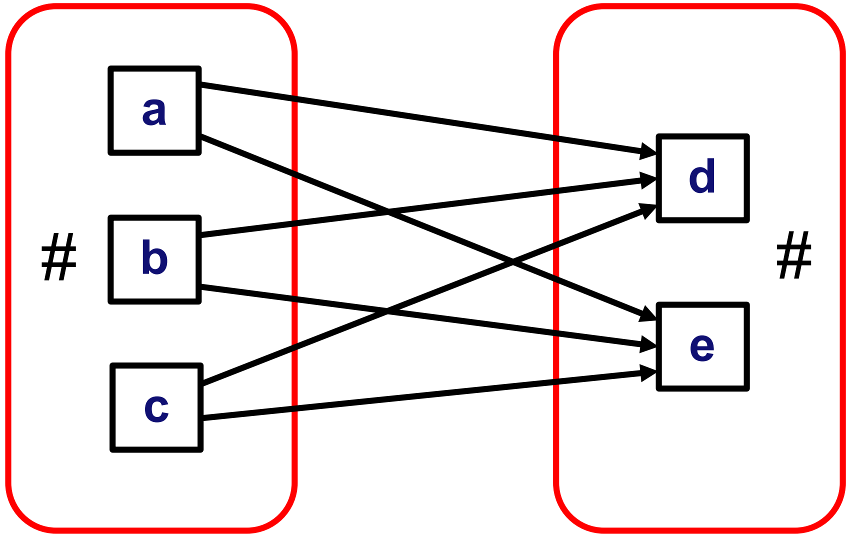

**Represents a place!**



# Many overlapping places



# Many overlapping places




Defines 20 smaller candidates!

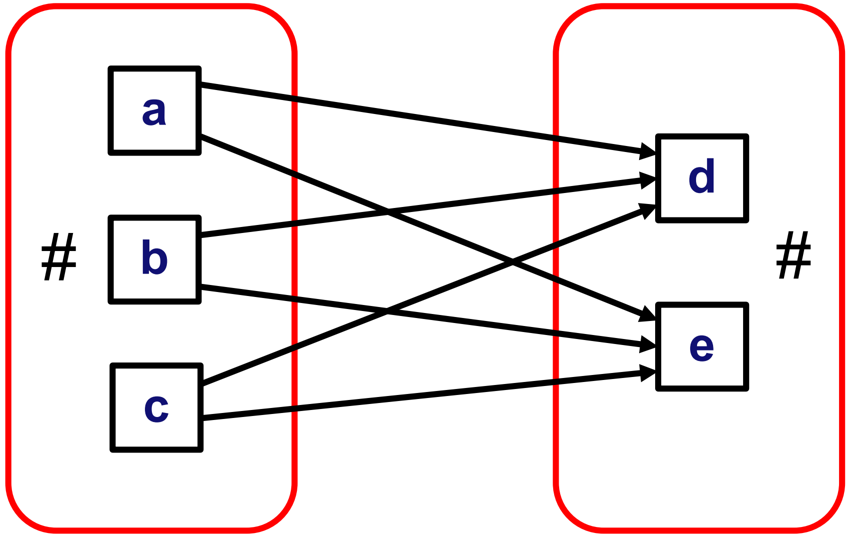
If $(\{a,b,c\},\{d,e\})$ is a candidate then also $(\{a,b\},\{d,e\})$, $(\{a,c\},\{d,e\})$, $(\{b,c\},\{d,e\})$, $(\{a,b,c\},\{d\})$, $(\{a,b,c\},\{e\})$, $(\{a\},\{d,e\})$, $(\{b\},\{d,e\})$, $(\{c\},\{d,e\})$, $(\{a,b\},\{d\})$, $(\{a,c\},\{d\})$, $(\{b,c\},\{d\})$, $(\{a,b\},\{e\})$, $(\{a,c\},\{e\})$, $(\{b,c\},\{e\})$, $(\{a\},\{d\})$, $(\{a\},\{e\})$, $(\{b\},\{d\})$, $(\{b\},\{e\})$, $(\{c\},\{d\})$, and $(\{c\},\{e\})$ !



Chair of Process and Data Science

# Step 2: Only use the maximal candidates

2. $Sel = \{(A_1, A_2) \in Cnd \mid \forall_{(A'_1, A'_2) \in Cnd}\ A_1 \subseteq A'_1 \ \wedge A_2 \subseteq A'_2 \implies (A_1, A_2) = (A'_1, A'_2)\}$ *are the selected maximal places,*




It should be impossible to add an activity to $A_1$ or $A_2$

# The rest is just bookkeeping

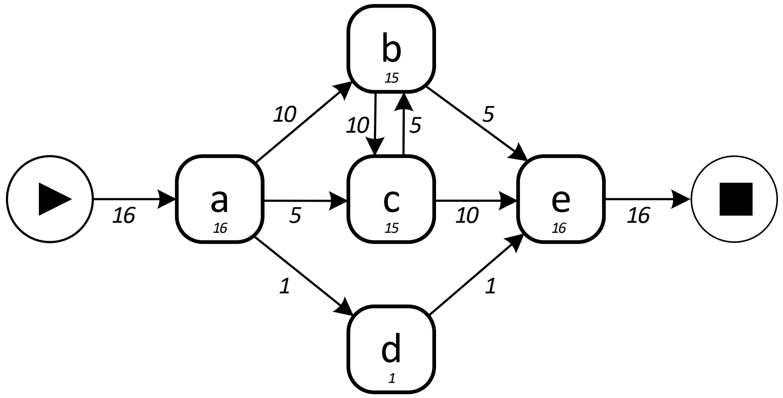
3. $P = \{p_{(A_1,A_2)} \mid (A_1, A_2) \in Sel\} \cup \{p_{\blacktriangleright}, p_{\blacksquare}\}$ *is the set of all places,*
4. $T = \{t_a \mid a \in A'\}$ *is the set of transitions,*
5. $F = \{(t_a, p_{(A_1,A_2)}) \mid (A_1, A_2) \in Sel \wedge a \in A_1\} \cup \{(p_{(A_1,A_2)}, t_a) \mid (A_1, A_2) \in Sel \wedge a \in A_2\} \cup \{(p_{\blacktriangleright}, t_{\blacktriangleright}), (t_{\blacksquare}, p_{\blacksquare})\}$ *is the set of arcs,*
6. $l = \{(t_a, a) \mid a \in A\}$ *is the labeling function,*
7. $M_{init} = [p_{\blacktriangleright}]$ *is the initial marking,* $M_{final} = [p_{\blacksquare}]$ *is the final marking, and*
8. $disc_{alpha}(L) = ((P, T, F, l), M_{init}, M_{final})$ *is the discovered accepting Petri net.*

Add places, transitions, arcs, and initial and final marking.

Chair of Process and Data Science

# Example

$$L_1 = [\langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{5}, \langle a,d,e\rangle] \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$$




$Cnd$ = {({▶},{a}),({a},{b}),({a},{c}),({a},{d}),({a},{b,d}),({a},{c,d}), ({b},{e}),({c},{e}),({d},{e}),({b,d},{e}),({c,d},{e}),({e},{■})}

$Sel$ = {({▶},{a}),({a},{b,d}),({a},{c,d}),({b,d},{e}),({c,d},{e}),({e},{■})}

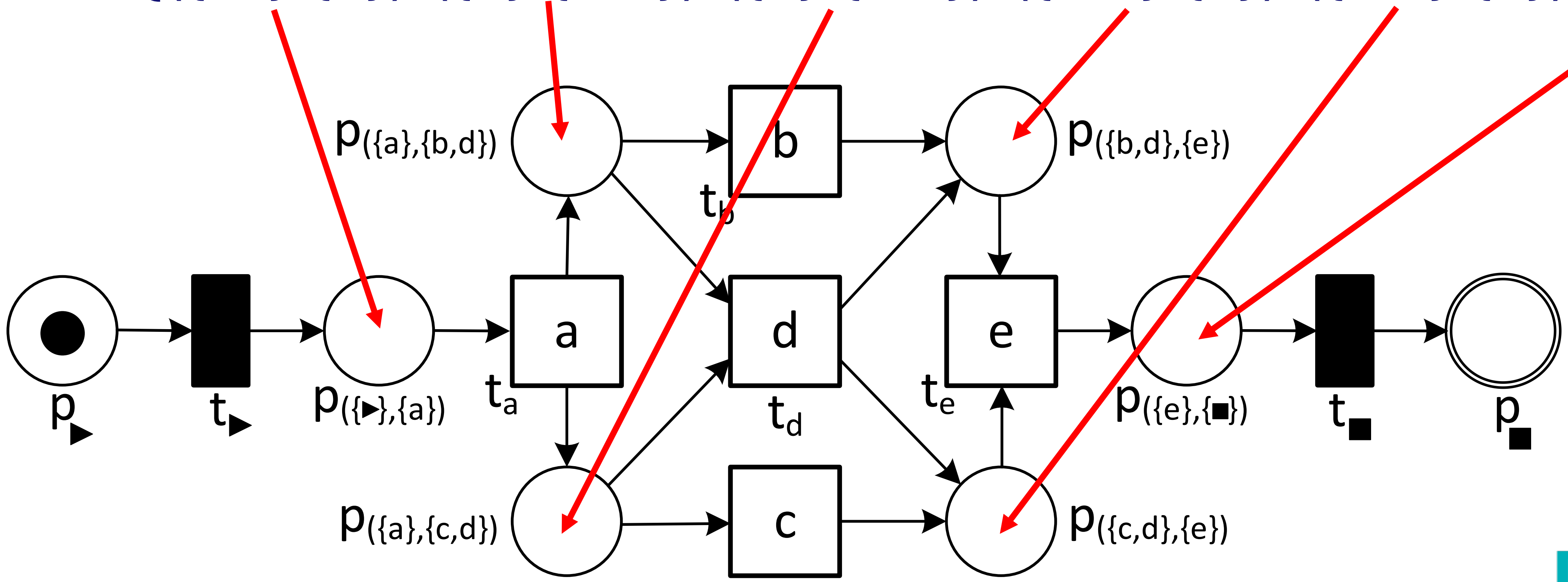


Chair of Process and Data Science

# Example

$$L_1 = [\langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{5}, \langle a,d,e\rangle] \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$$

$Sel = \{(\{\blacktriangleright\},\{a\}),(\{a\},\{b,d\}),(\{a\},\{c,d\}),(\{b,d\},\{e\}),(\{c,d\},\{e\}),(\{e\},\{\blacksquare\})\}$

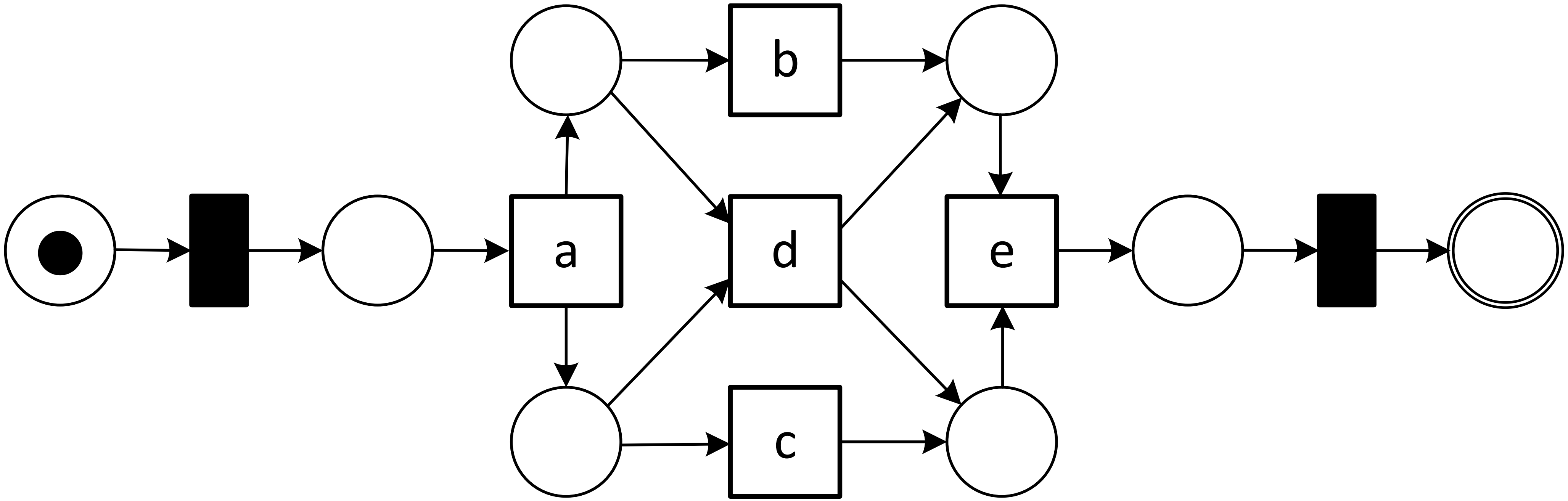





Chair of Process and Data Science

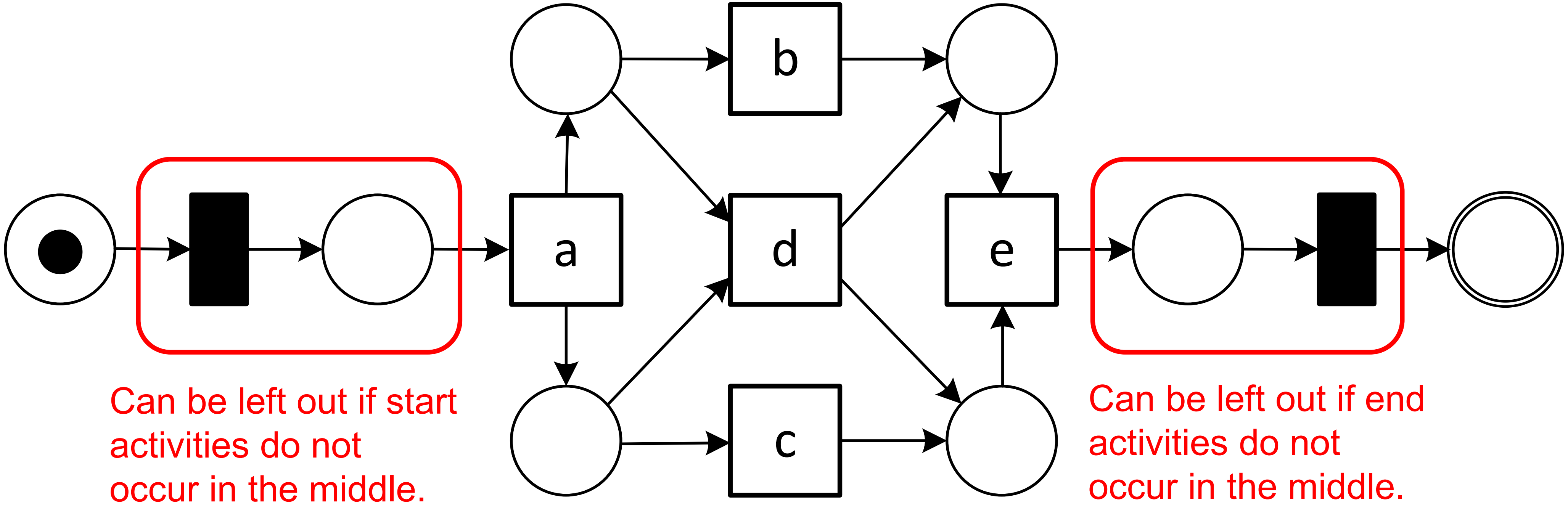
# Remove place and transition names to improve readability

$$L_1 = [\langle a, b, c, e\rangle^{10}, \langle a, c, b, e\rangle^{5}, \langle a, d, e\rangle] \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$$



# Example

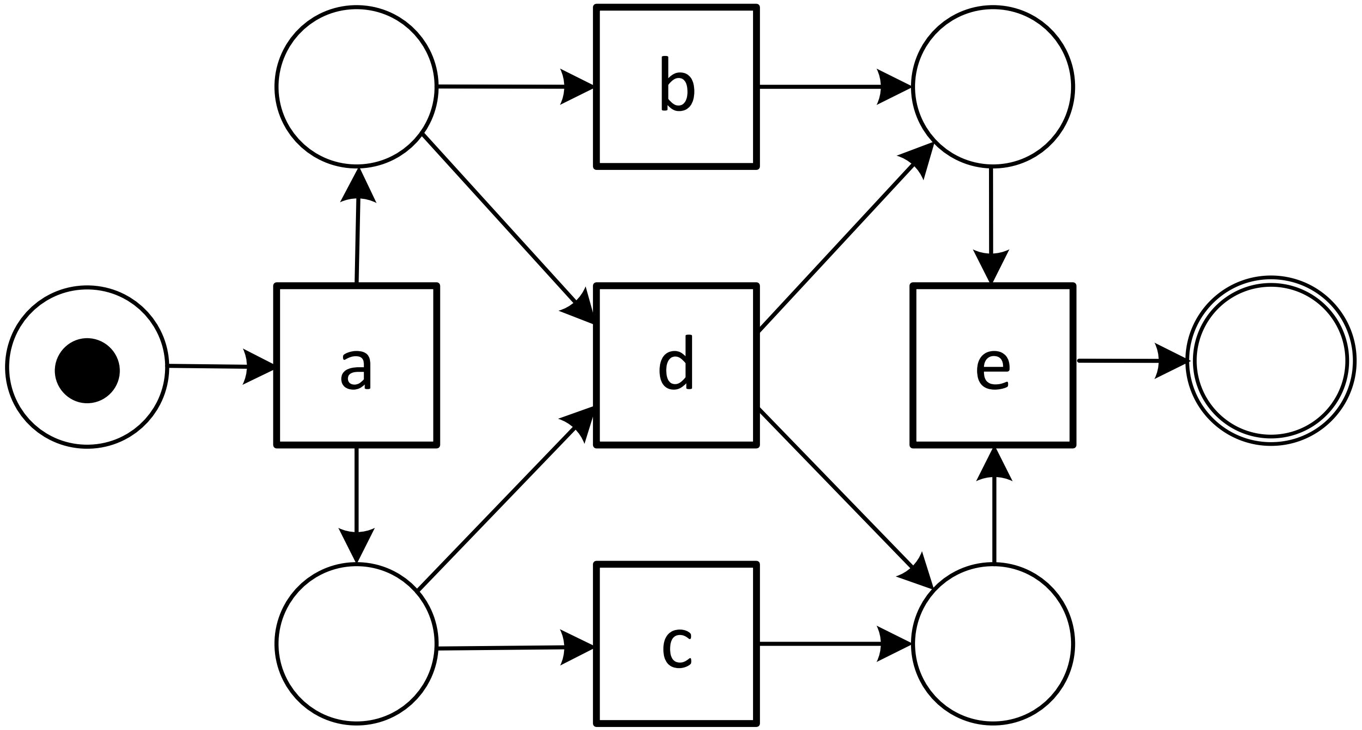
$$L_1 = [\langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{5}, \langle a,d,e\rangle] \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$$




Different from original paper to allow for a larger class of models to be discovered correctly.

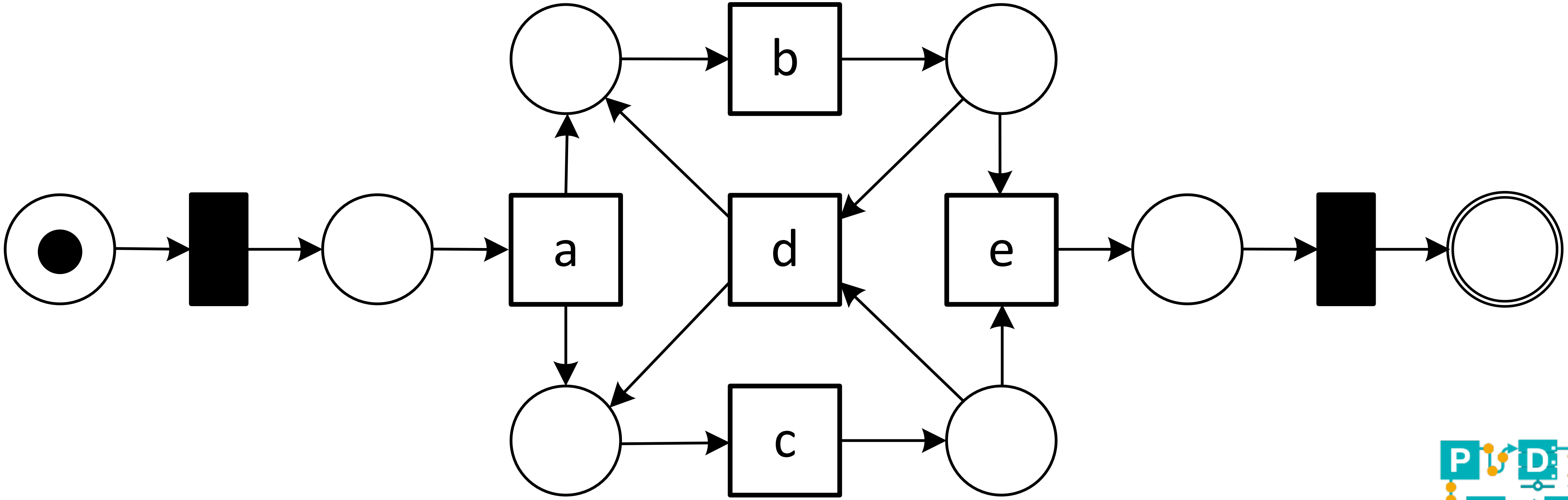
# Example

$$L_1 = [\langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{5}, \langle a,d,e\rangle] \in \mathcal{B}(\mathcal{U}_{act}{}^{*})$$



# Another example

$$L_2 = [\langle a,b,c,e\rangle^{50}, \langle a,c,b,e\rangle^{40}, \langle a,b,c,d,b,c,e\rangle^{30}, \langle a,c,b,d,b,c,e\rangle^{20}, \langle a,b,c,d,c,b,e\rangle^{10}, \langle a,c,b,d,c,b,d,b,c,e\rangle^{10}]$$

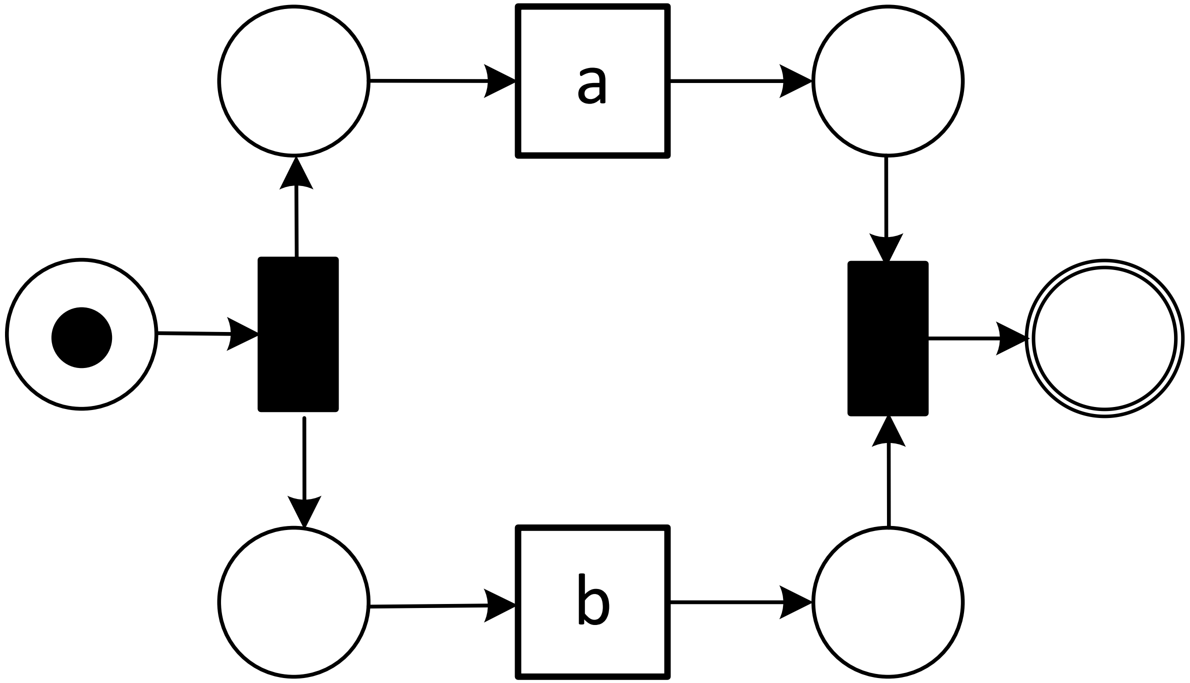





Chair of Process and Data Science

# Another example

$$L_4 = [\langle a, b\rangle^{35}, \langle b, a\rangle^{15}]$$



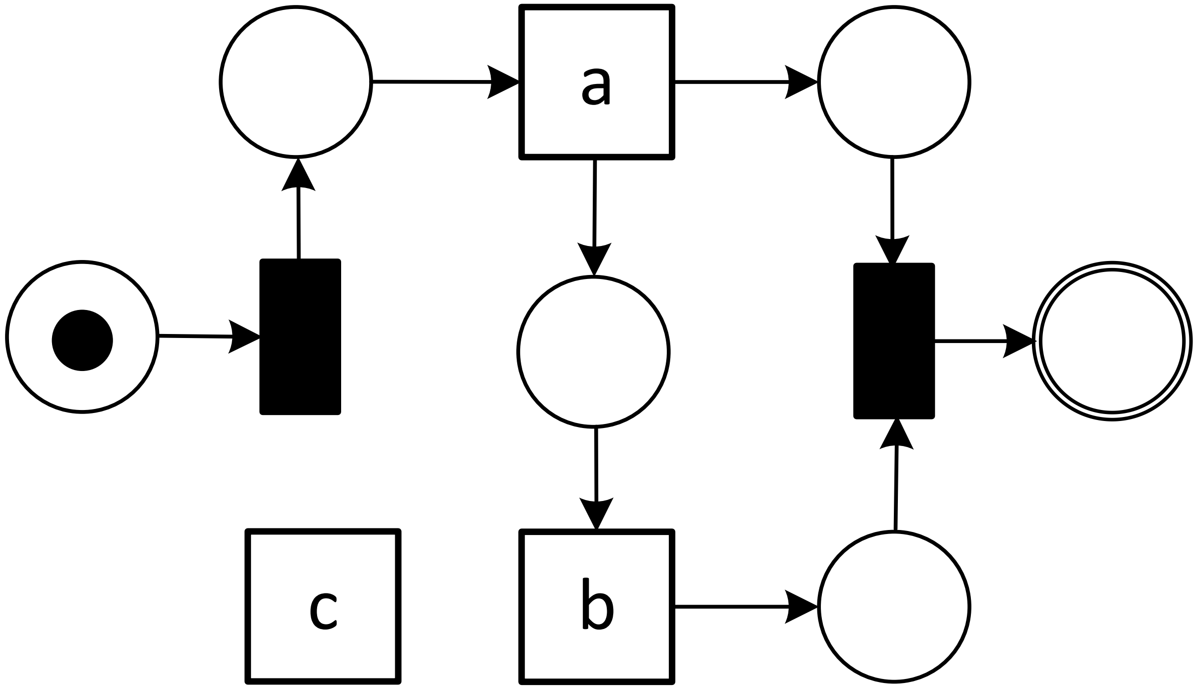

Illustrates why it makes sense to add an artificial start and end.

# Properties of the Alpha algorithm

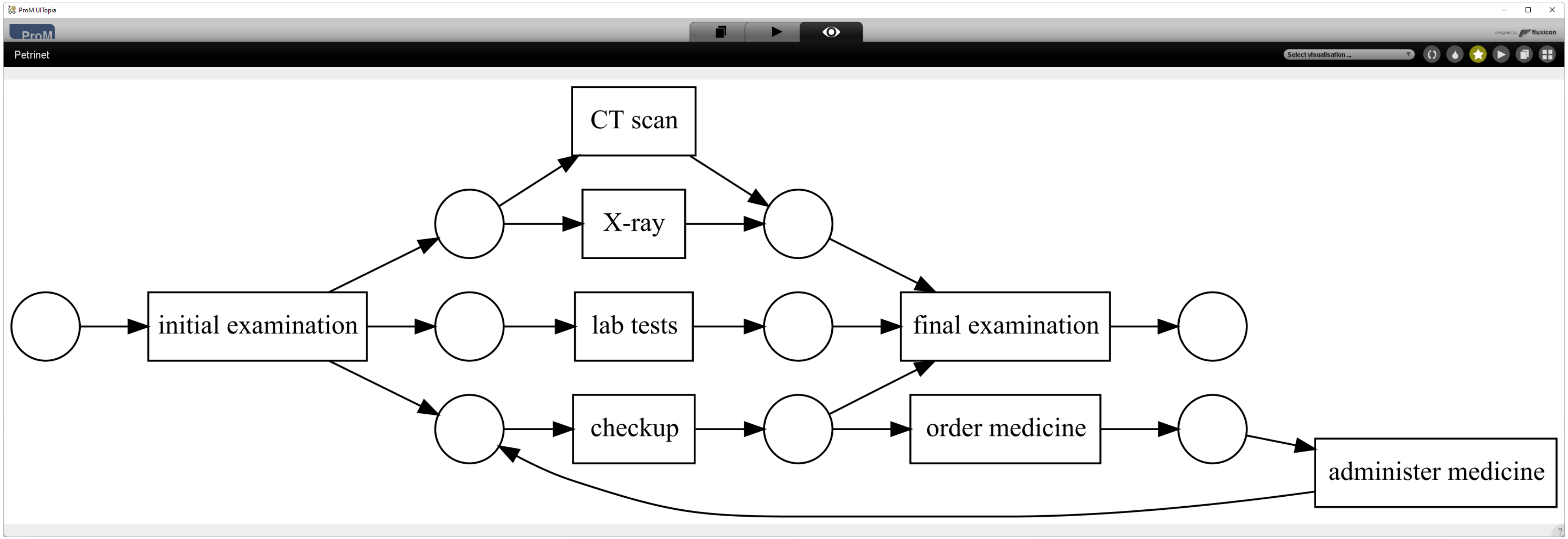
- Scalable (only needs the DFG)
- Guarantees for a subclass of free-choice nets.
- Cannot handle:
  - Short loops (loops of length 1 or 2)
  - Skipping (i.e., silent transitions).
- Although not practical in real-life scenarios, it nicely illustrates the essence of process discovery.
- See “Workflow Mining: Discovering Process Models from Event Logs. IEEE Trans. Knowl. Data Eng. 16(9): 1128-1142 (2004)” for guarantees and limitations.

# Example showing limitations

$$L_5 = [\langle a \rangle^{10}, \langle a, b \rangle^8, \langle a, c, b \rangle^6, \langle a, c, c, b \rangle^3, \langle a, c, c, c, b \rangle]$$



# Example in ProM
## 1856 cases, 11761 events, 197 variants






Chair of Process and Data Science

# Top-down discovery



Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

# Top-down discovery

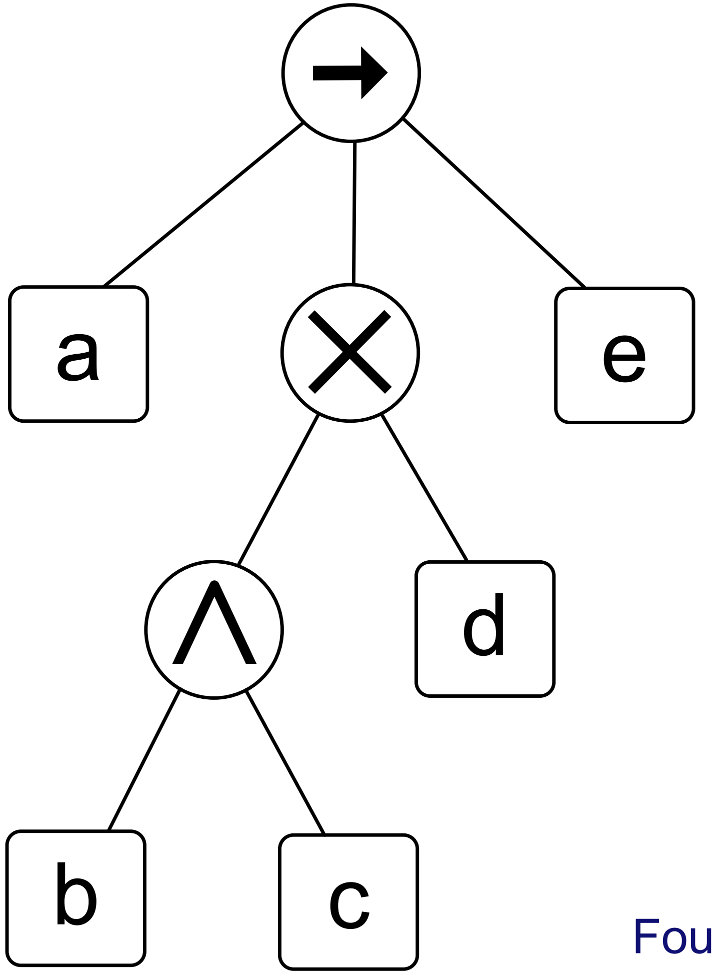
- Divide and conquer.
- Split the problem recursively into smaller problems such that things get trivial.
- An example is the Inductive Mining (IM) technique:
  - Uses process trees.
  - The leading approach
  - Implemented in ProM, Celonis, and many other tools.

# Process Trees

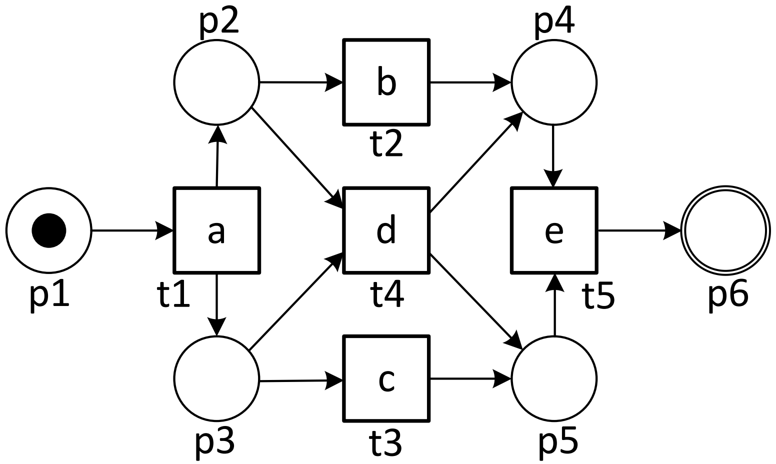


PADS Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

# A process tree




Semantics

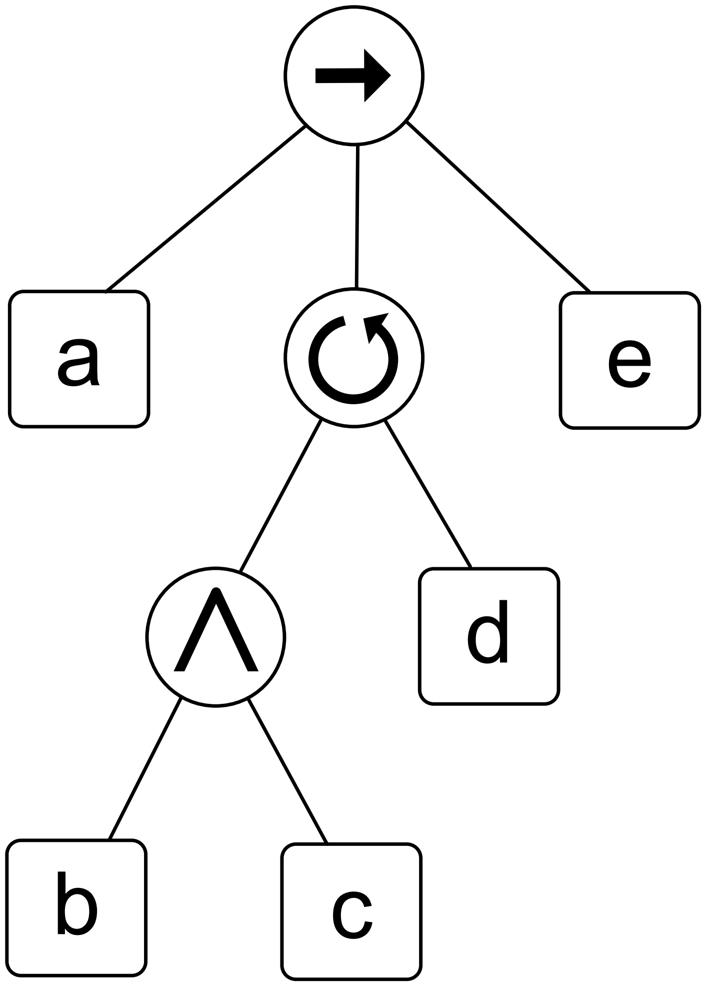



Four types of operators: → (sequential composition), × (exclusive choice), ∧ (parallel composition), and ⟲ (redo loop).

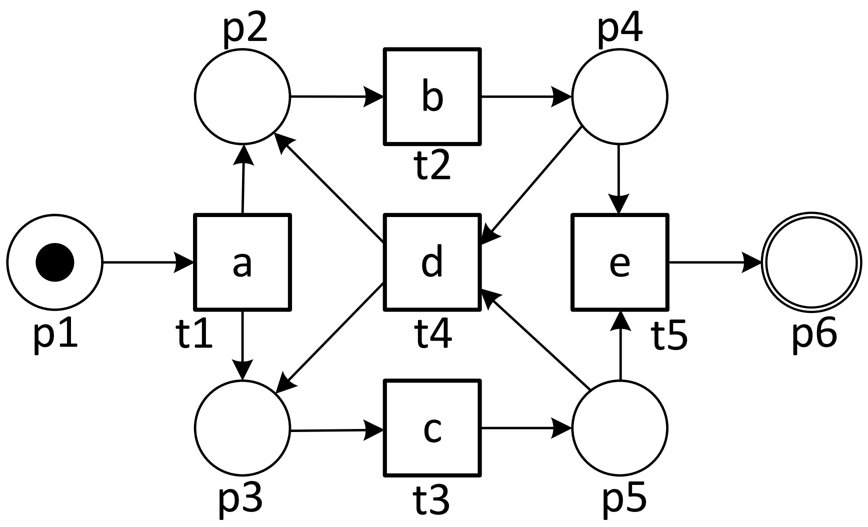


Chair of Process and Data Science

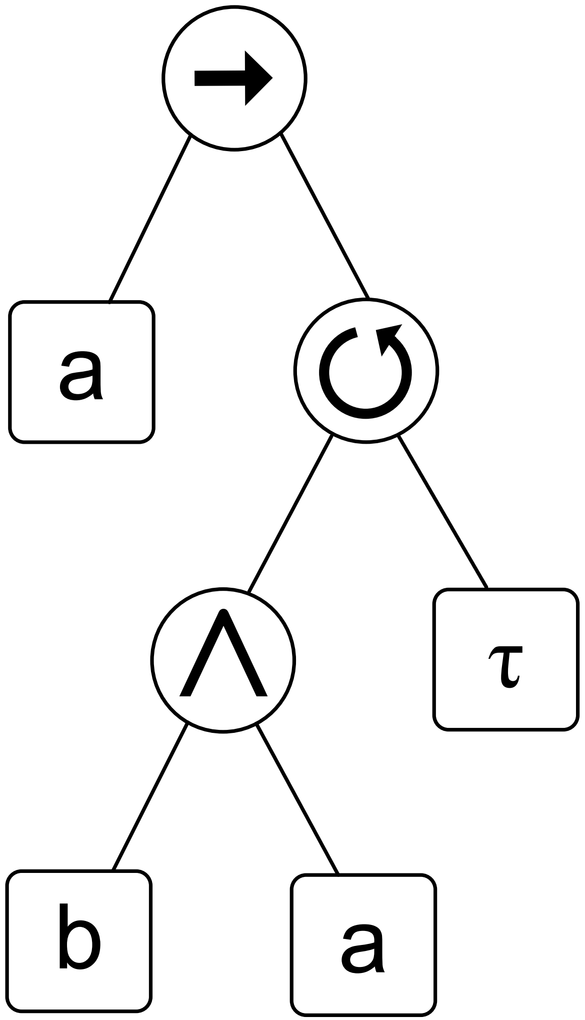
# Another process tree



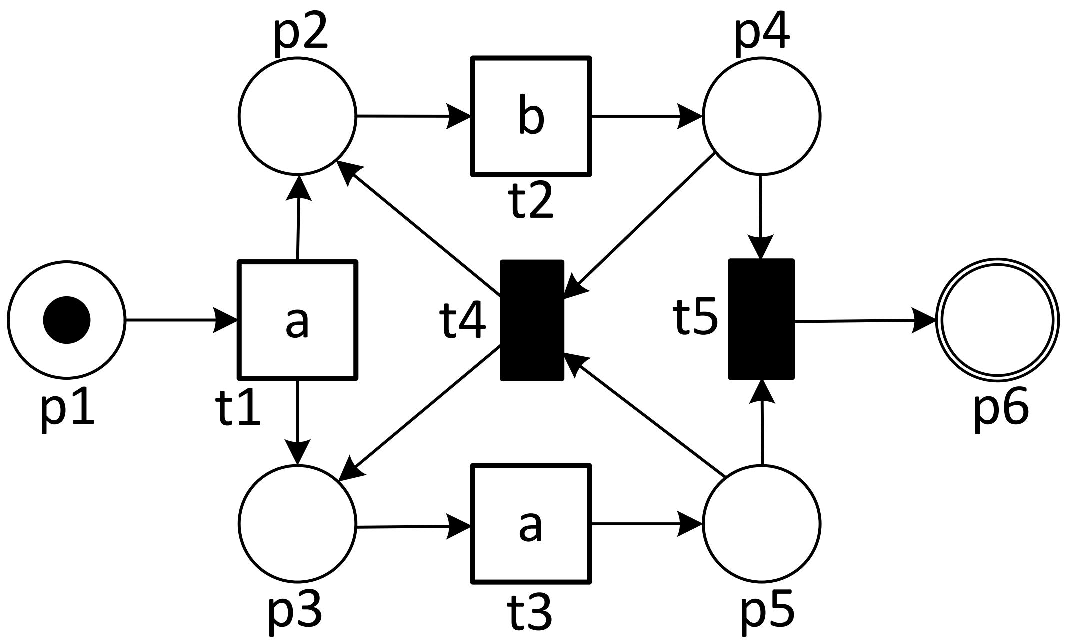

Semantics



# Another process tree




Semantics



# Inductive Mining

Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

# Inductive Mining (IM)

- Decompose the event log into smaller events logs until the problem get trivial.
- Four types of cuts corresponding to the operators: → (sequential composition), × (exclusive choice), ∧ (parallel composition), and ⟲ (redo loop).
- In each step the activities are partitioned into subsets until they are singletons.
- Developed by Sander Leemans in the context of his PhD thesis (NWO project “Don't Search for the Undesirable! Avoiding “Blind Alleys” in Process Mining” 2012-2017)

# Event log

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| bi | cb | ac | at | as | bo | ep | ck | |
| bi | cb | ac | as | at | bo | ep | ck | |
| bi | cb | at | ac | as | bo | ep | ck | |
| bi | cb | at | as | ac | bo | ck | | |
| bi | cb | as | ac | at | bo | ep | ck | |
| bi | cb | as | at | ac | bo | ep | ck | |
| bi | cb | ac | at | am | bo | ep | ck | |
| bi | cb | ac | am | at | bo | ep | ck | |
| bi | cb | at | ac | am | bo | ck | | |
| bi | cb | at | am | ac | bo | ep | ck | |
| bi | cb | am | ac | at | bo | ep | ck | |
| bi | cb | am | at | ac | bo | ep | ck | |
| bi | cb | at | as | ac | ac | bo | ep | ck |
| bi | cb | as | ac | ac | at | ac | bo | ck |
| bi | cb | as | at | ac | ac | bo | ep | ck |

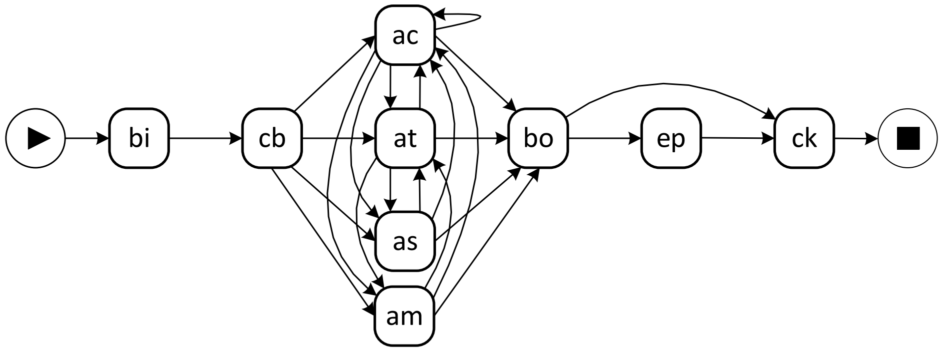
Activities: buy ingredients (bi), create base (cb), add cheese (ac), add tomato (at), add salami (as), add mushrooms (am), bake in oven (bo), eat pizza (ep), and clean kitchen (ck).



Chair of Process and Data Science

# Create a DFG for the whole event log

bi cb ac at as bo ep ck
bi cb ac as at bo ep ck
bi cb at ac as bo ep ck
bi cb at as ac bo ck
bi cb as ac at bo ep ck
bi cb as at ac bo ep ck
bi cb ac at am bo ep ck
bi cb ac am at bo ep ck
bi cb at ac am bo ck
bi cb at am ac bo ep ck
bi cb am ac at bo ep ck
bi cb am at ac bo ep ck
bi cb at as ac ac bo ep ck
bi cb as ac ac at ac bo ck
bi cb as at ac ac bo ep ck

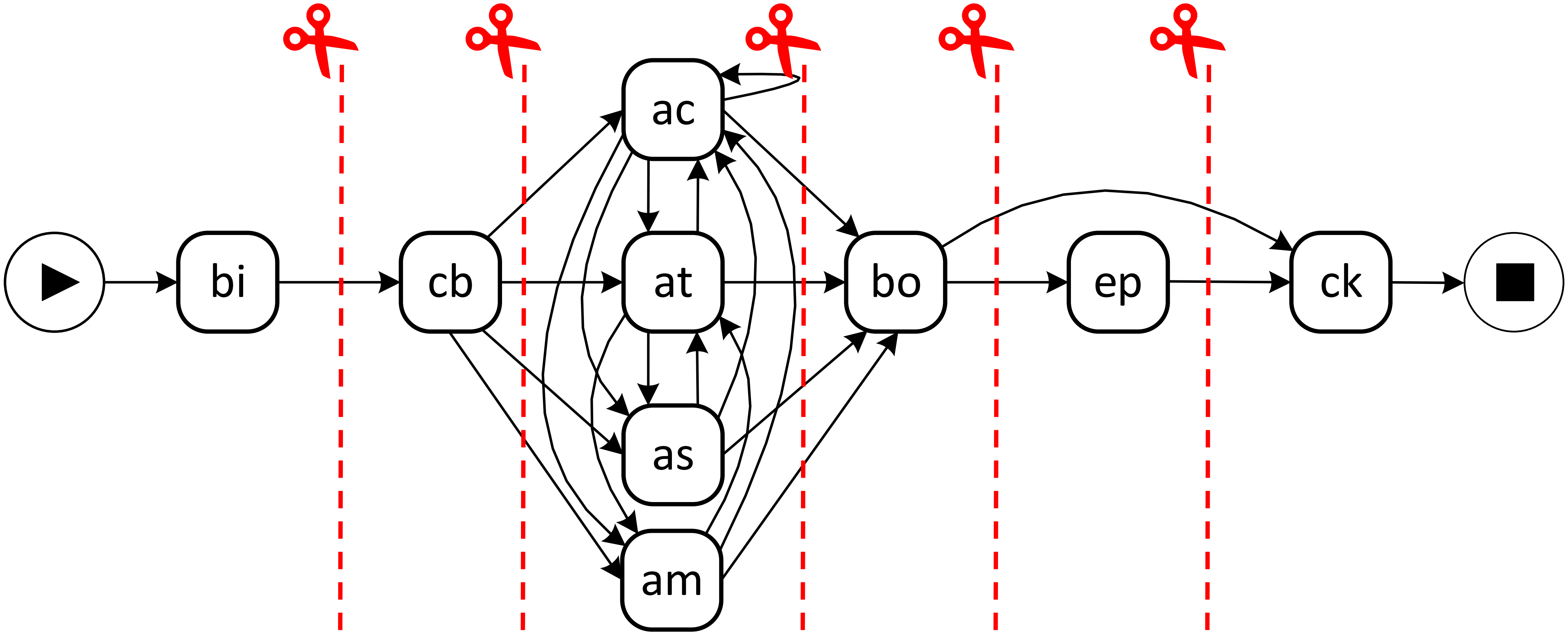



Frequencies omitted for readability



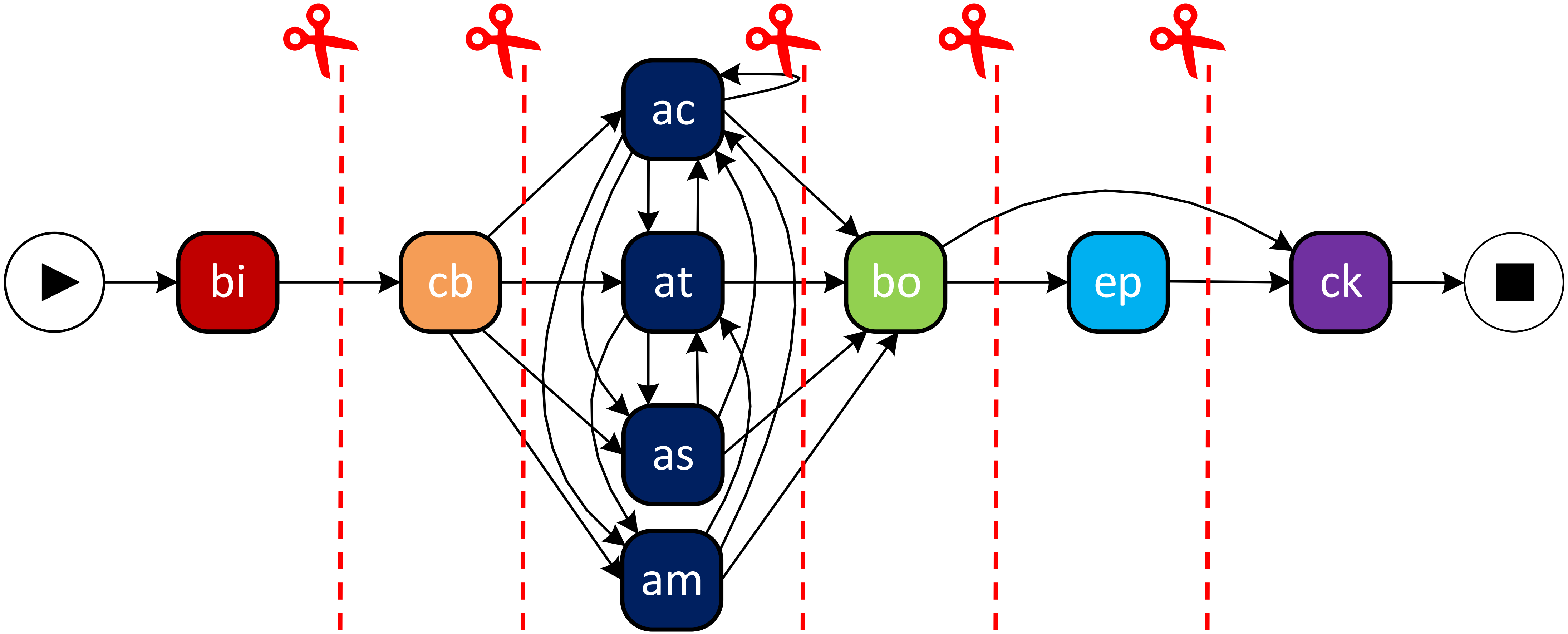
Chair of Process and Data Science

# Apply a sequence cut



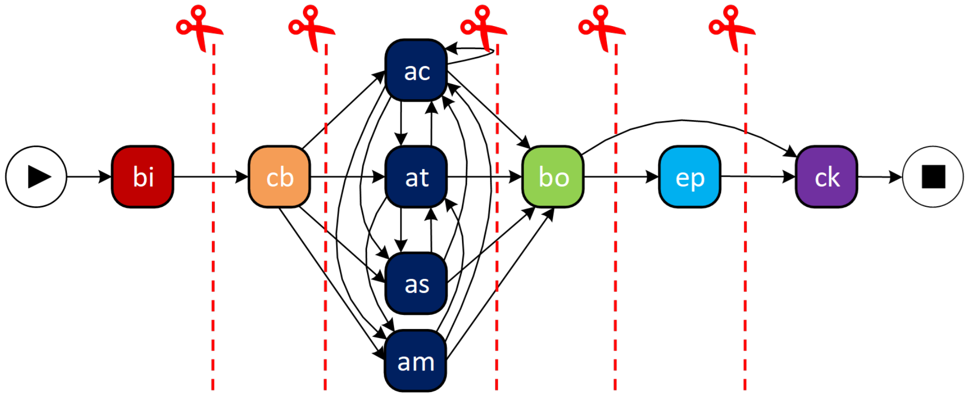

There is a sequence cut when the DFG can be split into sequential parts where only “forward connections” are possible. Note that we need to use the non-reflexive transitive closure of F.

# Sequence cut partitions activities in six subsets

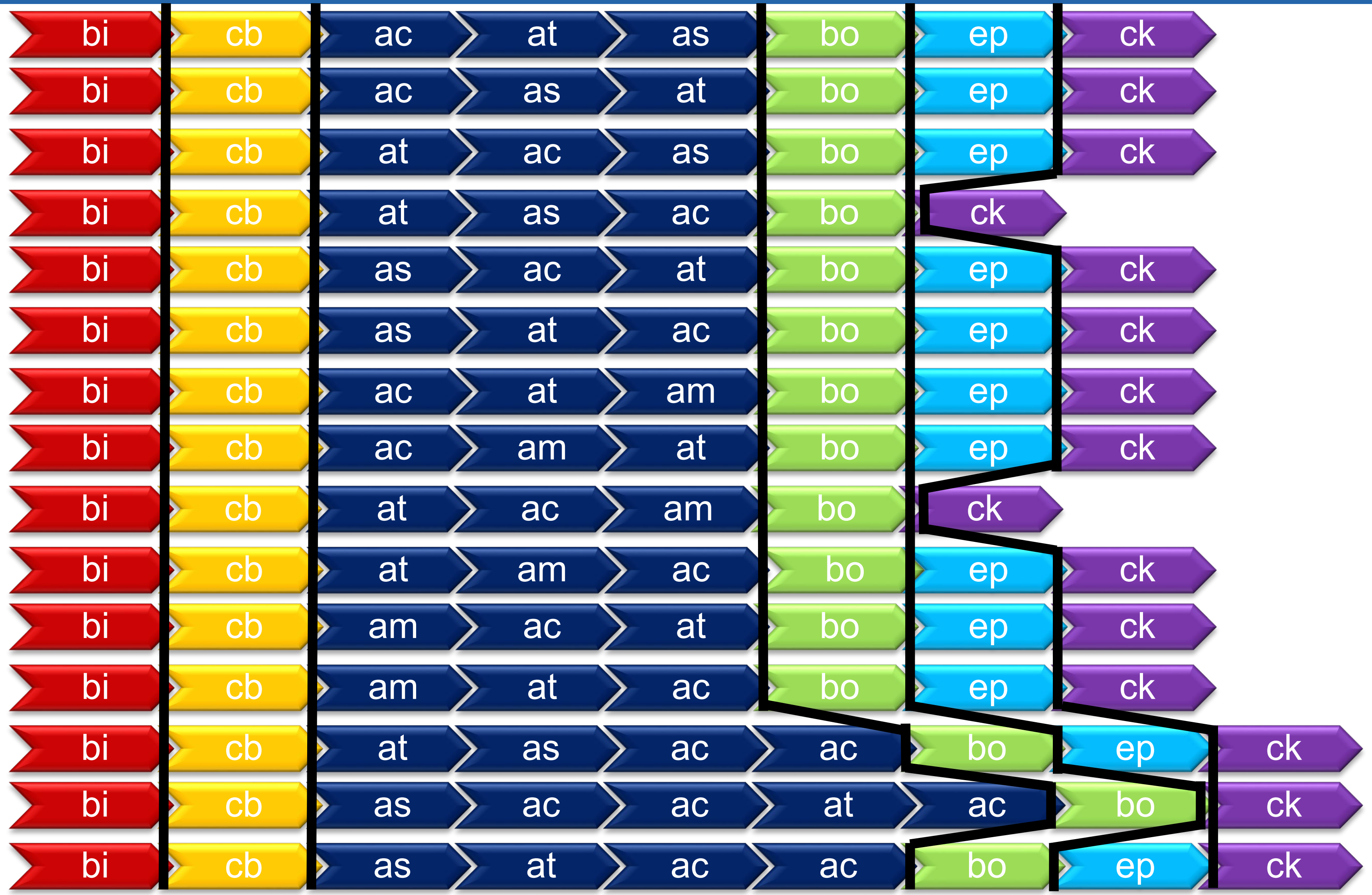


# Color the events based on the partitioning

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| bi | cb | ac | at | as | bo | ep | ck | |
| bi | cb | ac | as | at | bo | ep | ck | |
| bi | cb | at | ac | as | bo | ep | ck | |
| bi | cb | at | as | ac | bo | ck | | |
| bi | cb | as | ac | at | bo | ep | ck | |
| bi | cb | as | at | ac | bo | ep | ck | |
| bi | cb | ac | at | am | bo | ep | ck | |
| bi | cb | ac | am | at | bo | ep | ck | |
| bi | cb | at | ac | am | bo | ck | | |
| bi | cb | at | am | ac | bo | ep | ck | |
| bi | cb | am | ac | at | bo | ep | ck | |
| bi | cb | am | at | ac | bo | ep | ck | |
| bi | cb | at | as | ac | ac | bo | ep | ck |
| bi | cb | as | ac | ac | at | ac | bo | ck |
| bi | cb | as | at | ac | ac | bo | ep | ck |



# Split the event log based on the partitioning





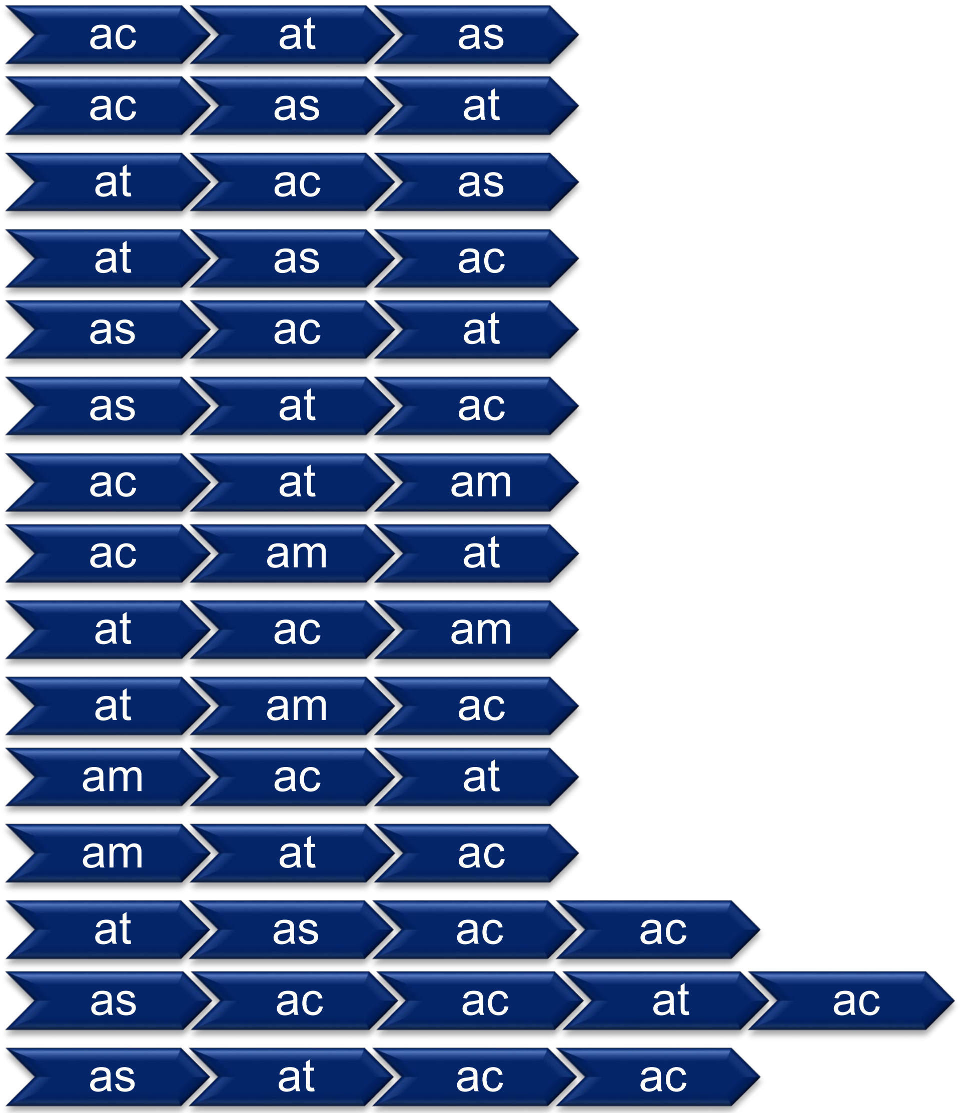

Chair of Process and Data Science

# Five of the projected event logs refer to a single activity (base case)




Chair of Process and Data Science

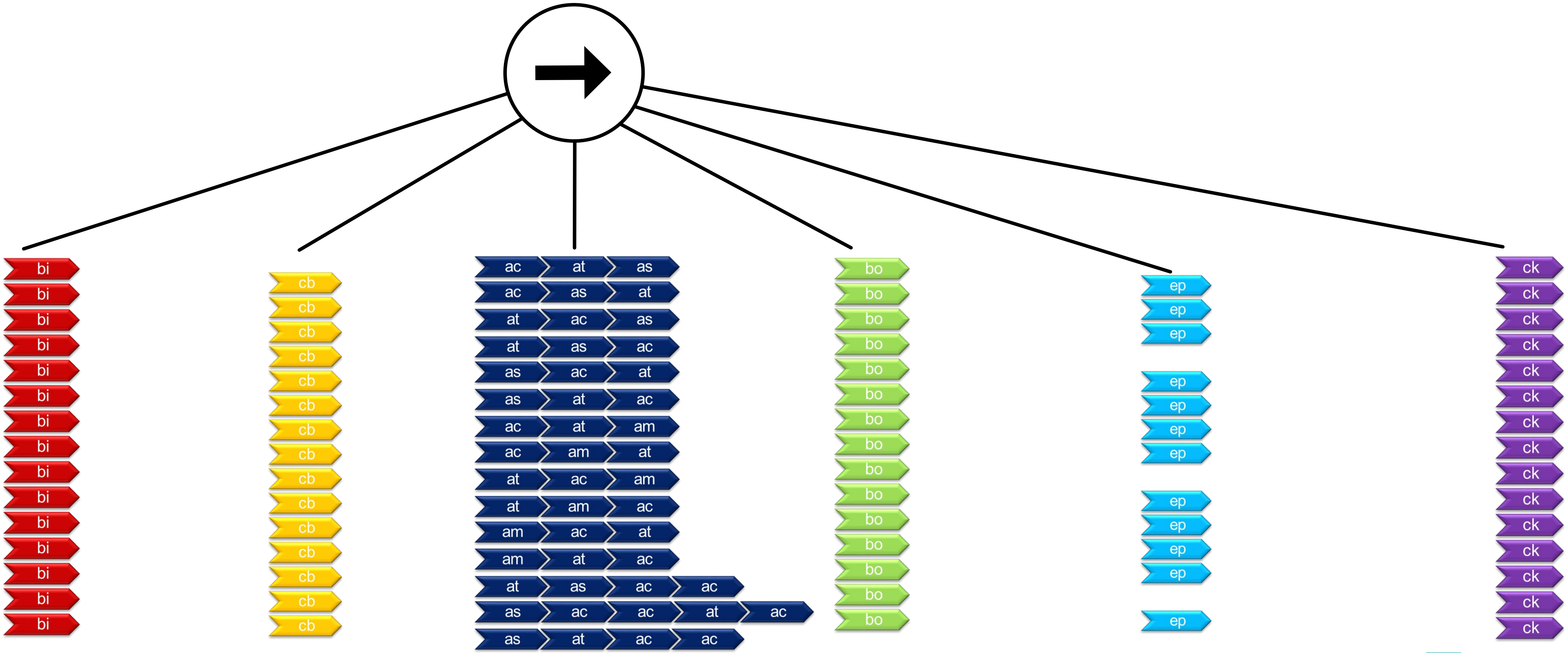
# The blue group has four activities






Chair of Process and Data Science

# Recursion: Apply algorithm to all sublogs

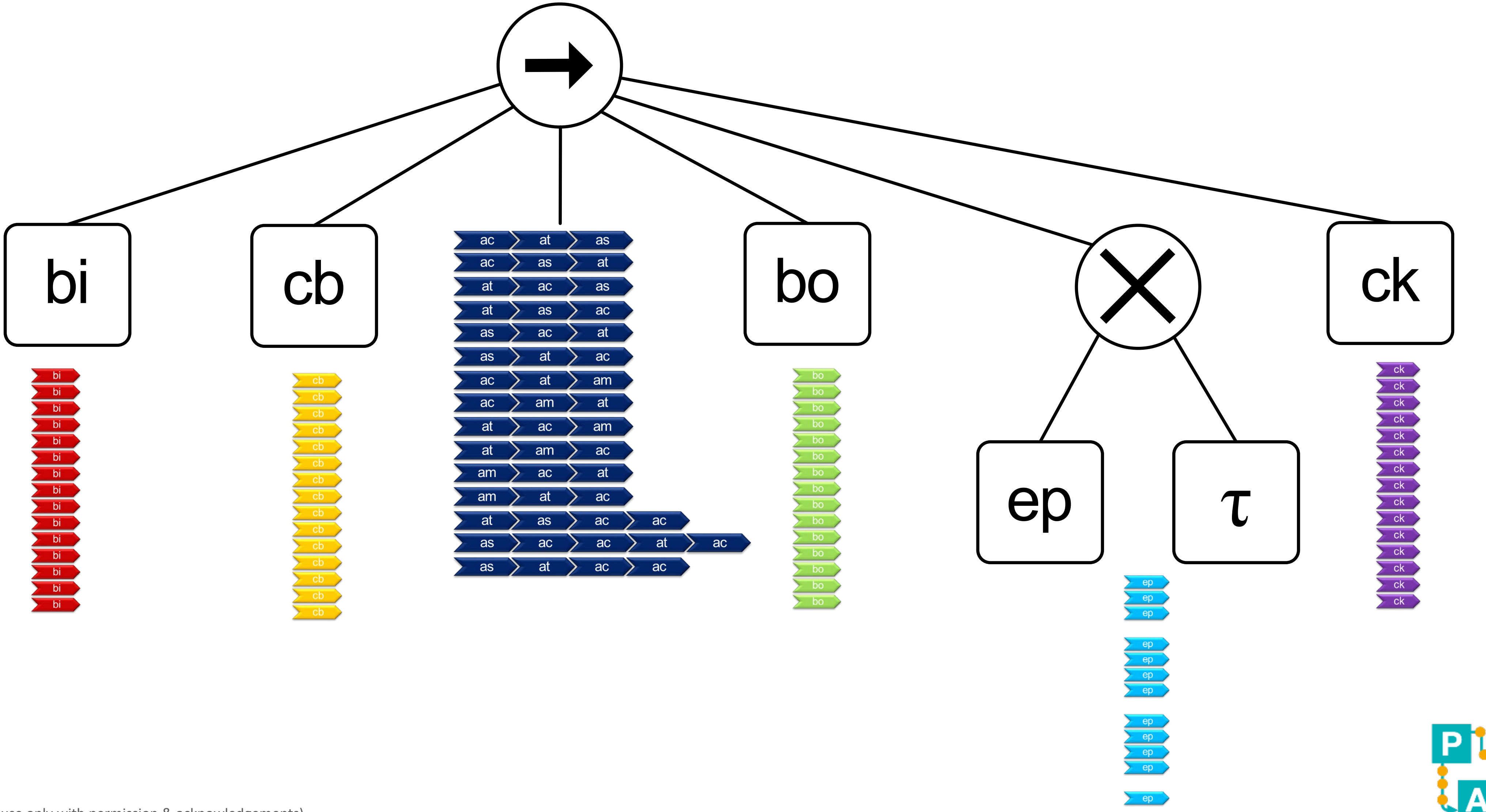



Five of the projected event logs refer to a single activity (base case).



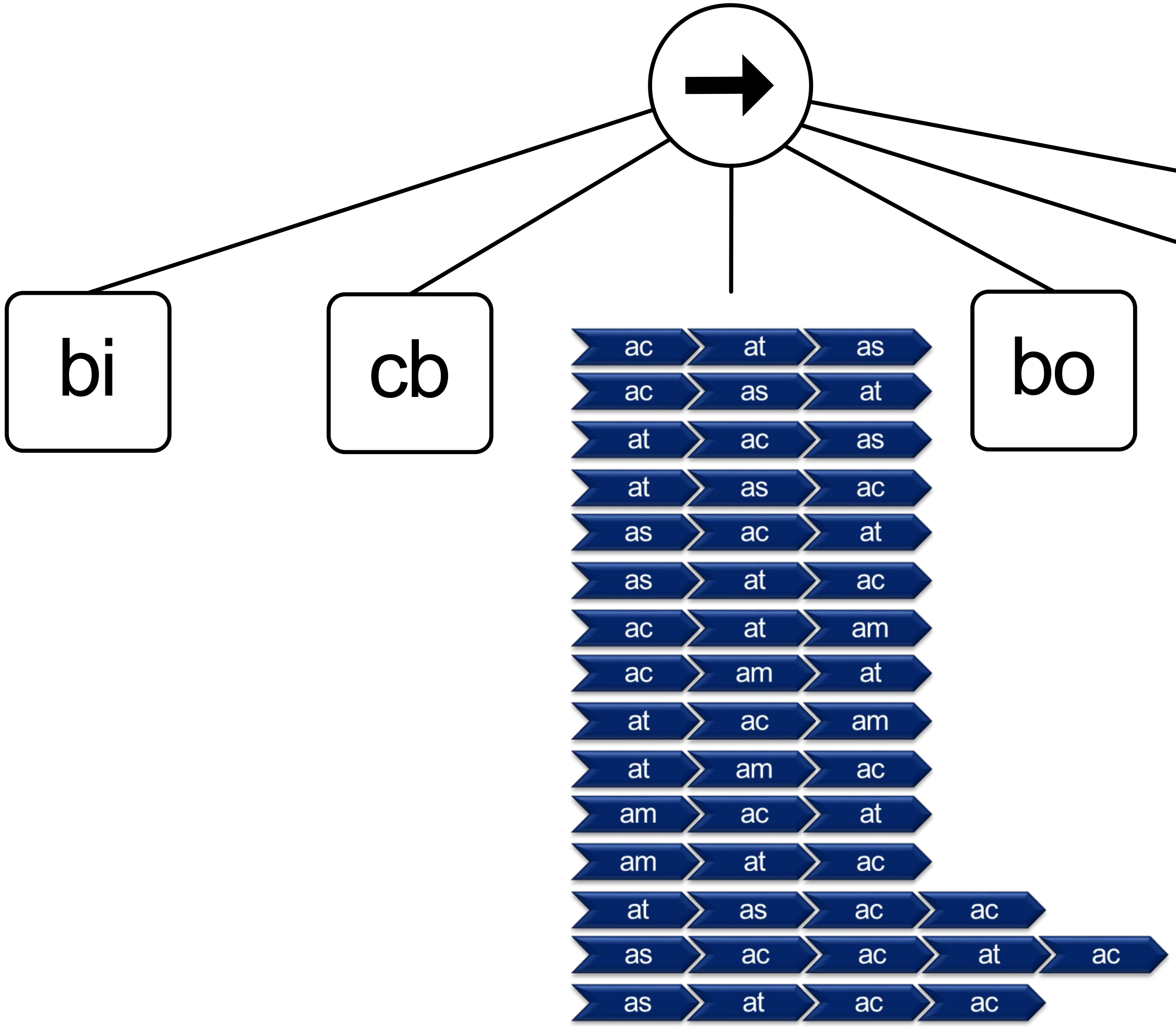
Chair of Process and Data Science

# Handling the base cases (ep can be skipped)




Chair of Process and Data Science

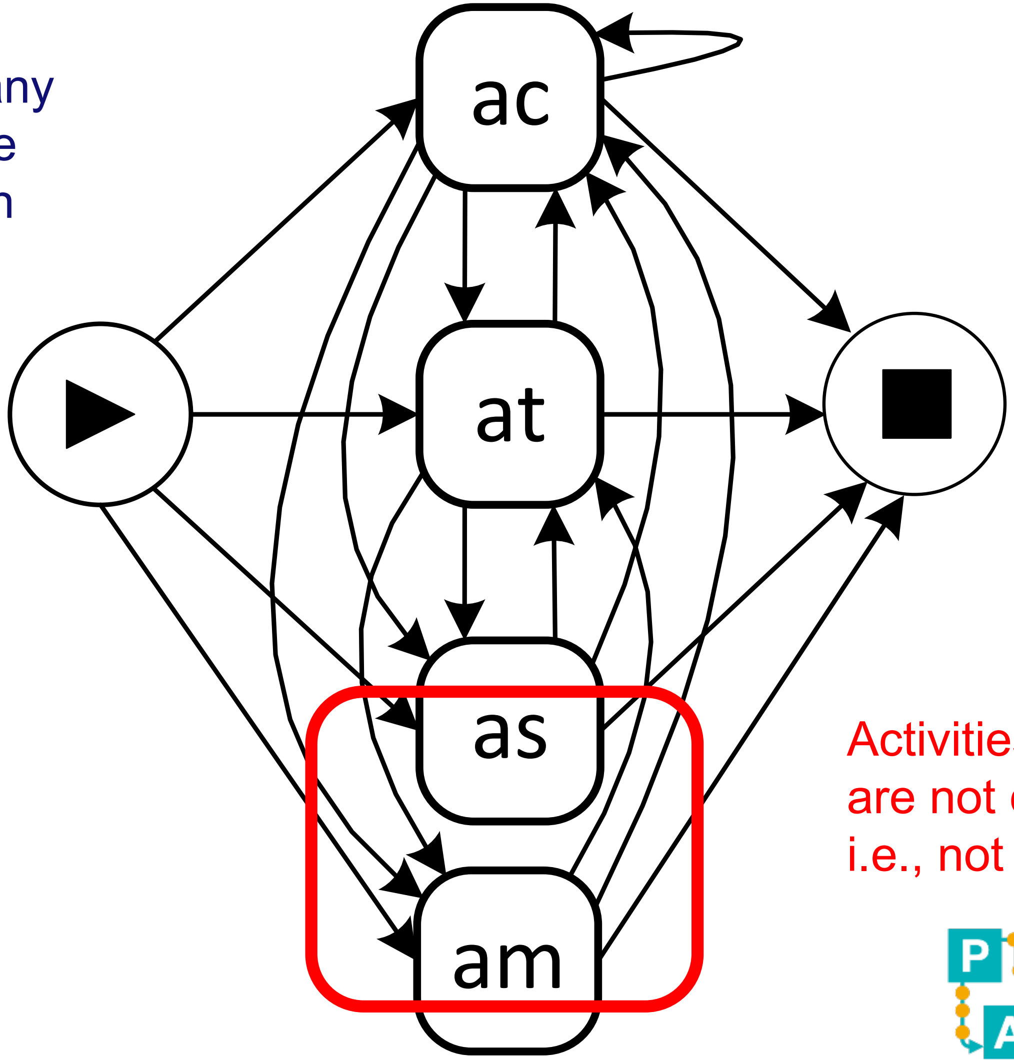
# Only the blue event log remains



# Continue with the blue event log

ac → at → as
ac → as → at
at → ac → as
at → as → ac
as → ac → at
as → at → ac
ac → at → am
ac → am → at
at → ac → am
at → am → ac
am → ac → at
am → at → ac
at → as → ac → ac
as → ac → ac → at → ac
as → at → ac → ac

There is a parallel cut when the DFG can be split into concurrent parts where any activity in one part can be followed by any activity in another part.




Activities as and am are not connected, i.e., not concurrent

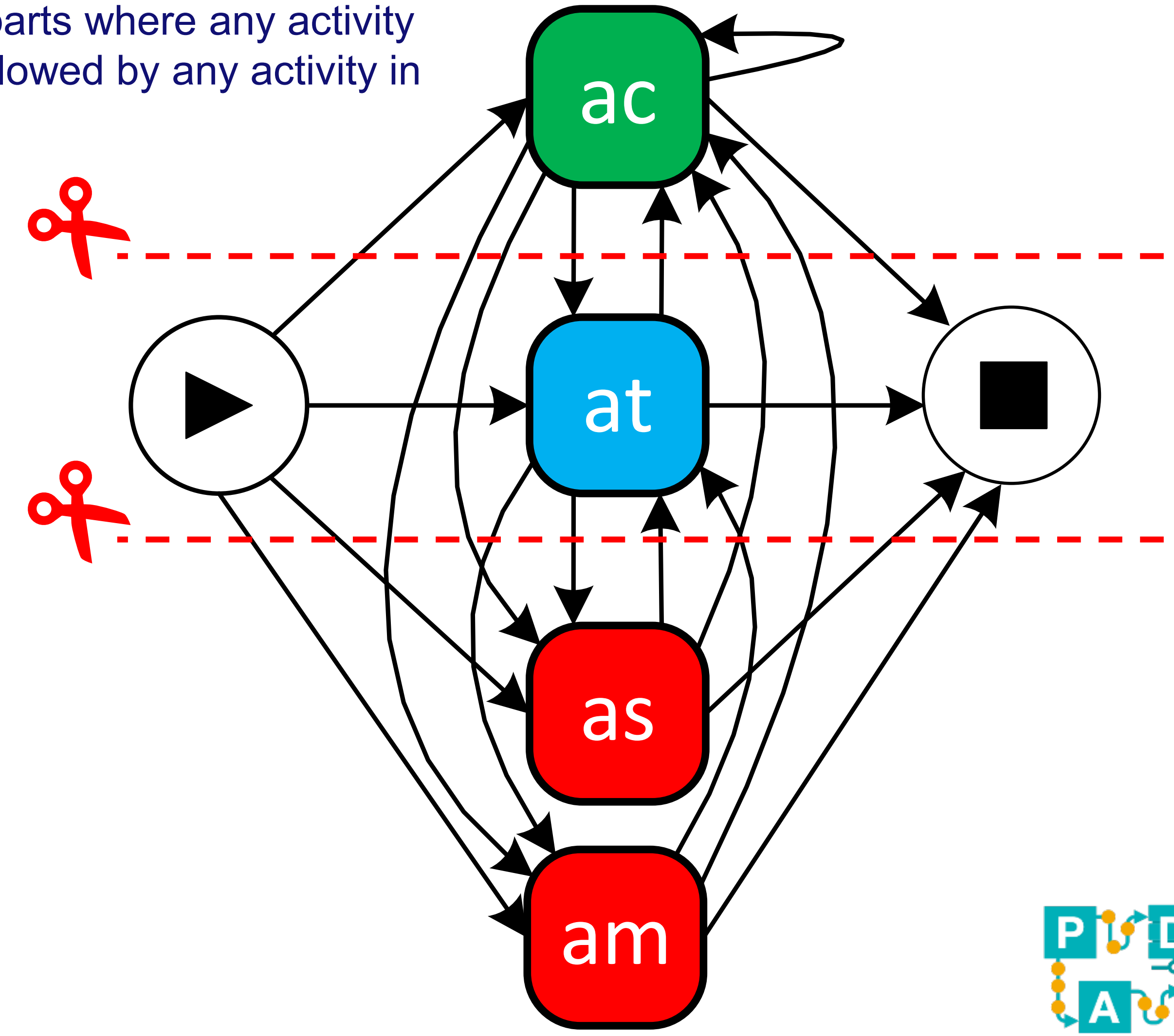


Chair of Process and Data Science

# Apply a parallel cut resulting in three activity groups

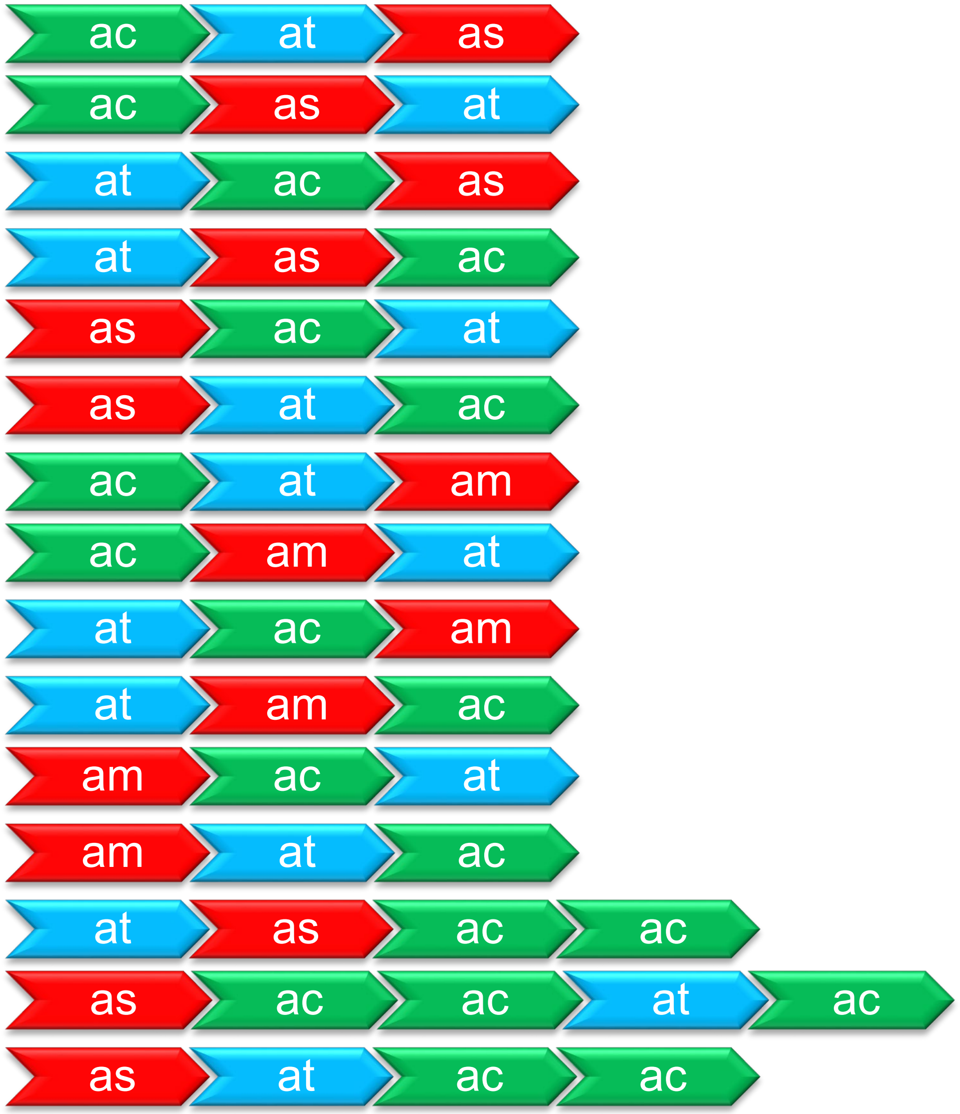
ac, at, as
ac, as, at
at, ac, as
at, as, ac
as, ac, at
as, at, ac
ac, at, am
ac, am, at
at, ac, am
at, am, ac
am, ac, at
am, at, ac
at, as, ac, ac
as, ac, ac, at, ac
as, at, ac, ac

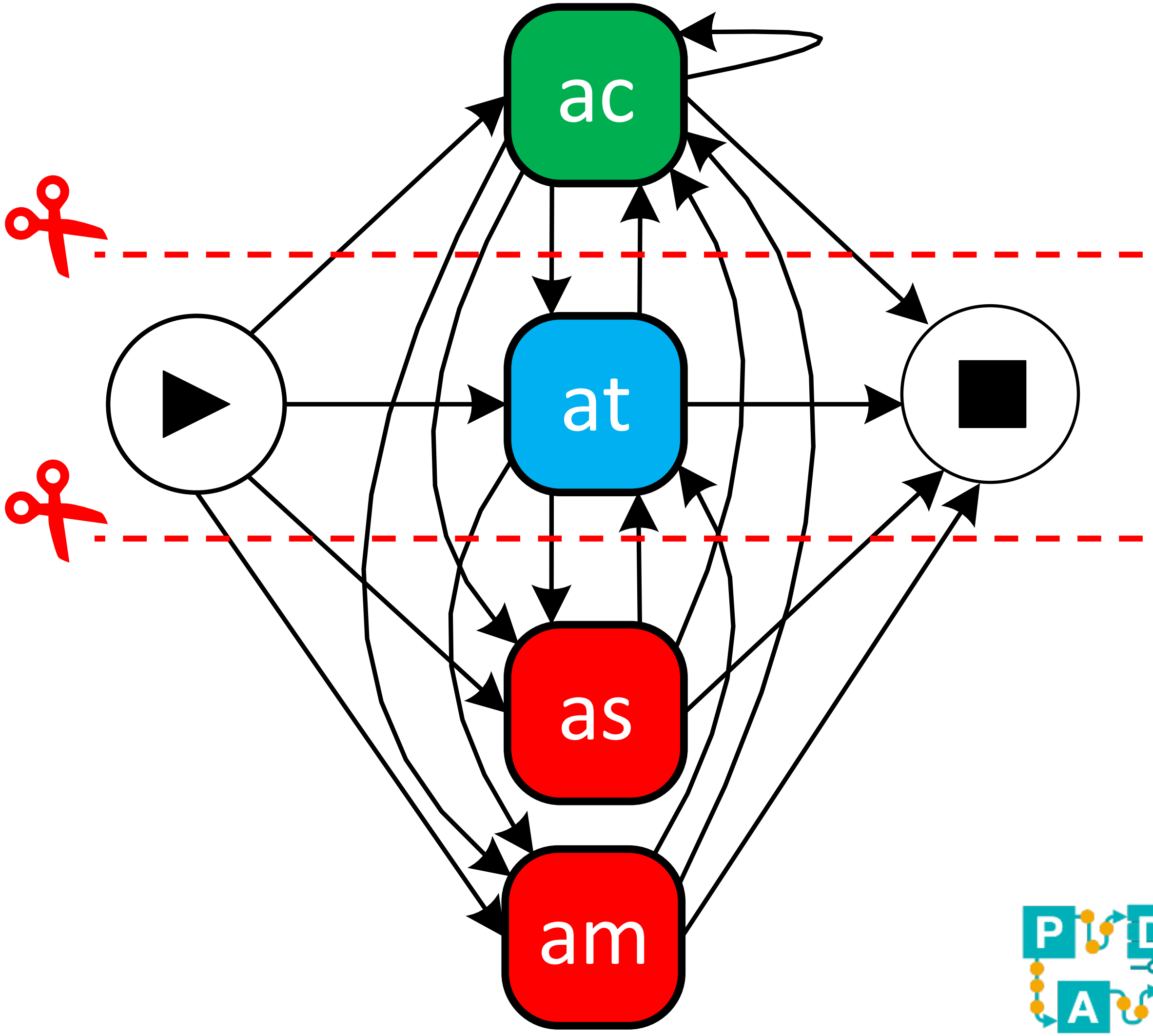
There is a parallel cut when the DFG can be split into concurrent parts where any activity in one part can be followed by any activity in another part.




Chair of Process and Data Science

# Apply a parallel cut resulting in three activity groups

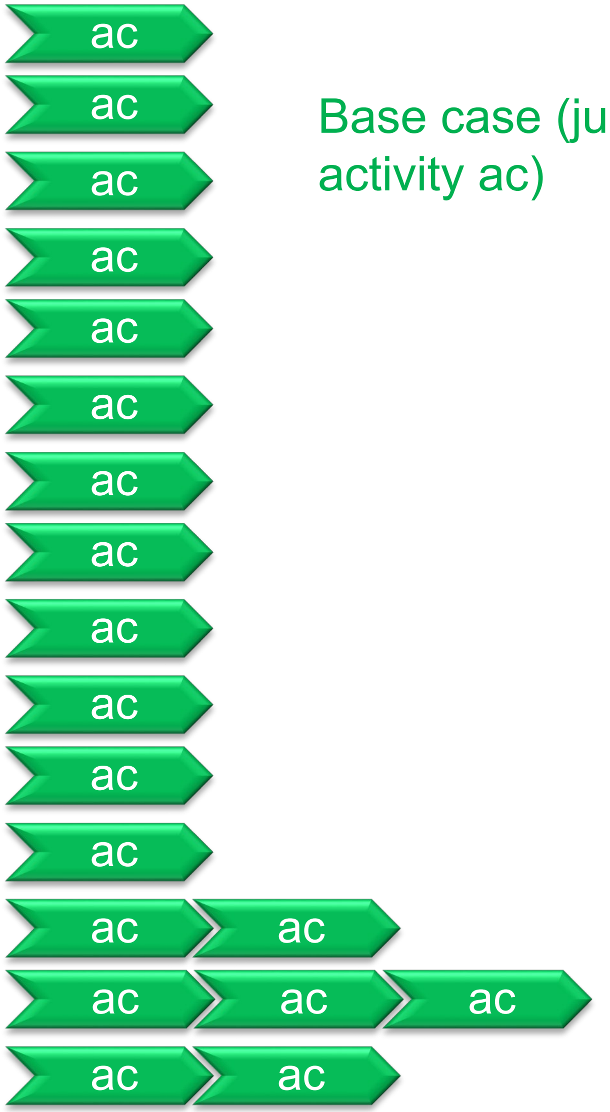


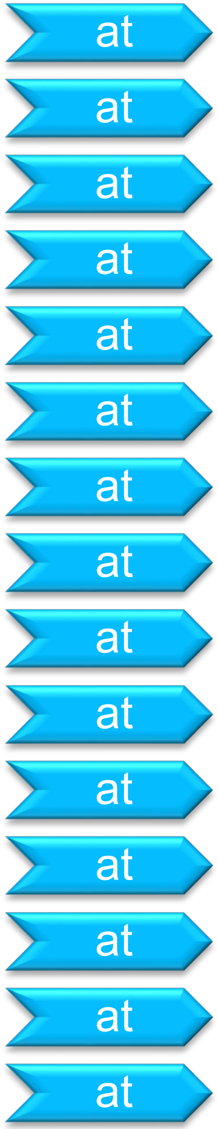



Chair of Process and Data Science

# Three new event logs are created



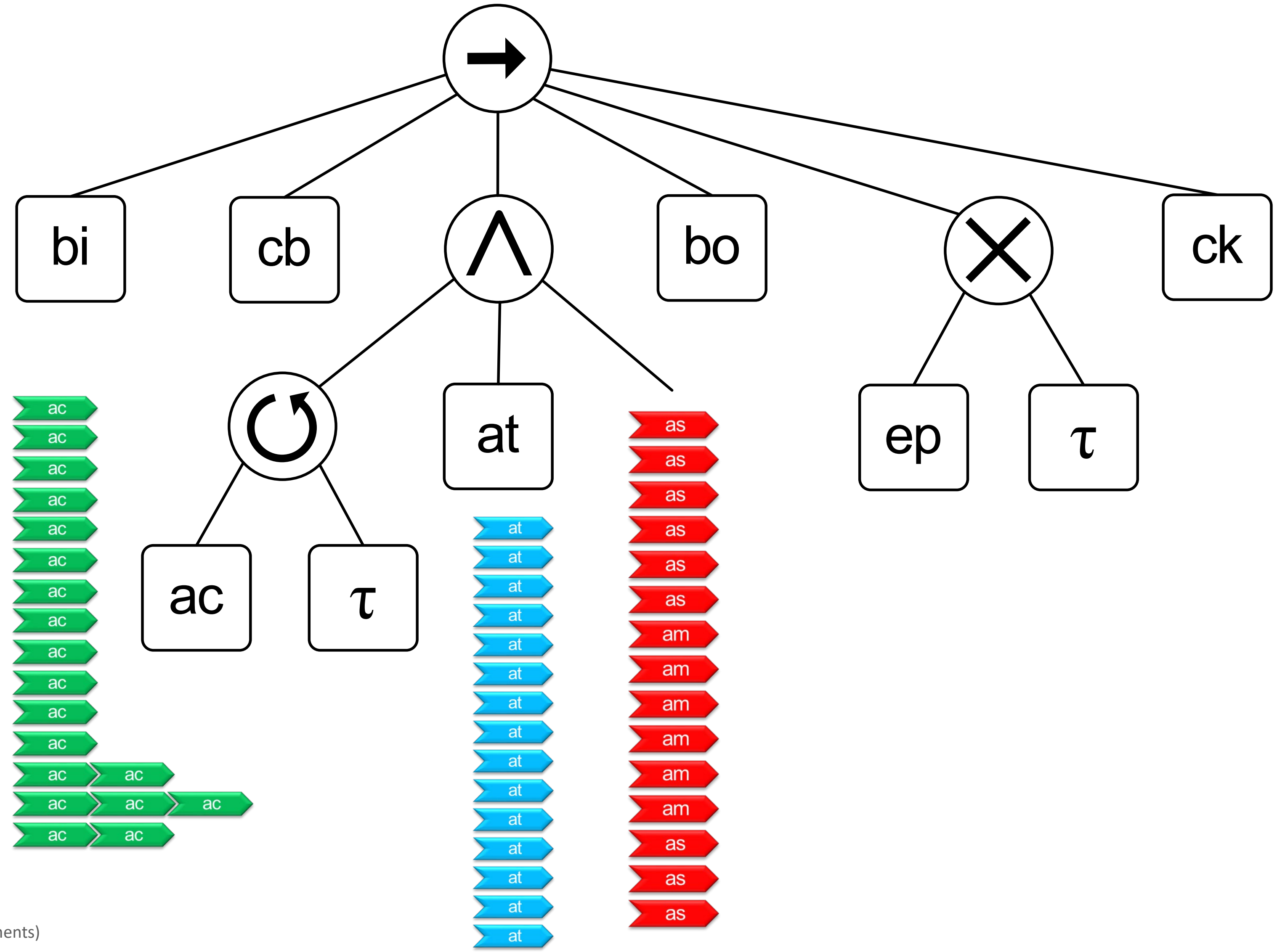

Base case (just activity ac)




Base case (just activity at)


Not a base case, still two activities as and am.

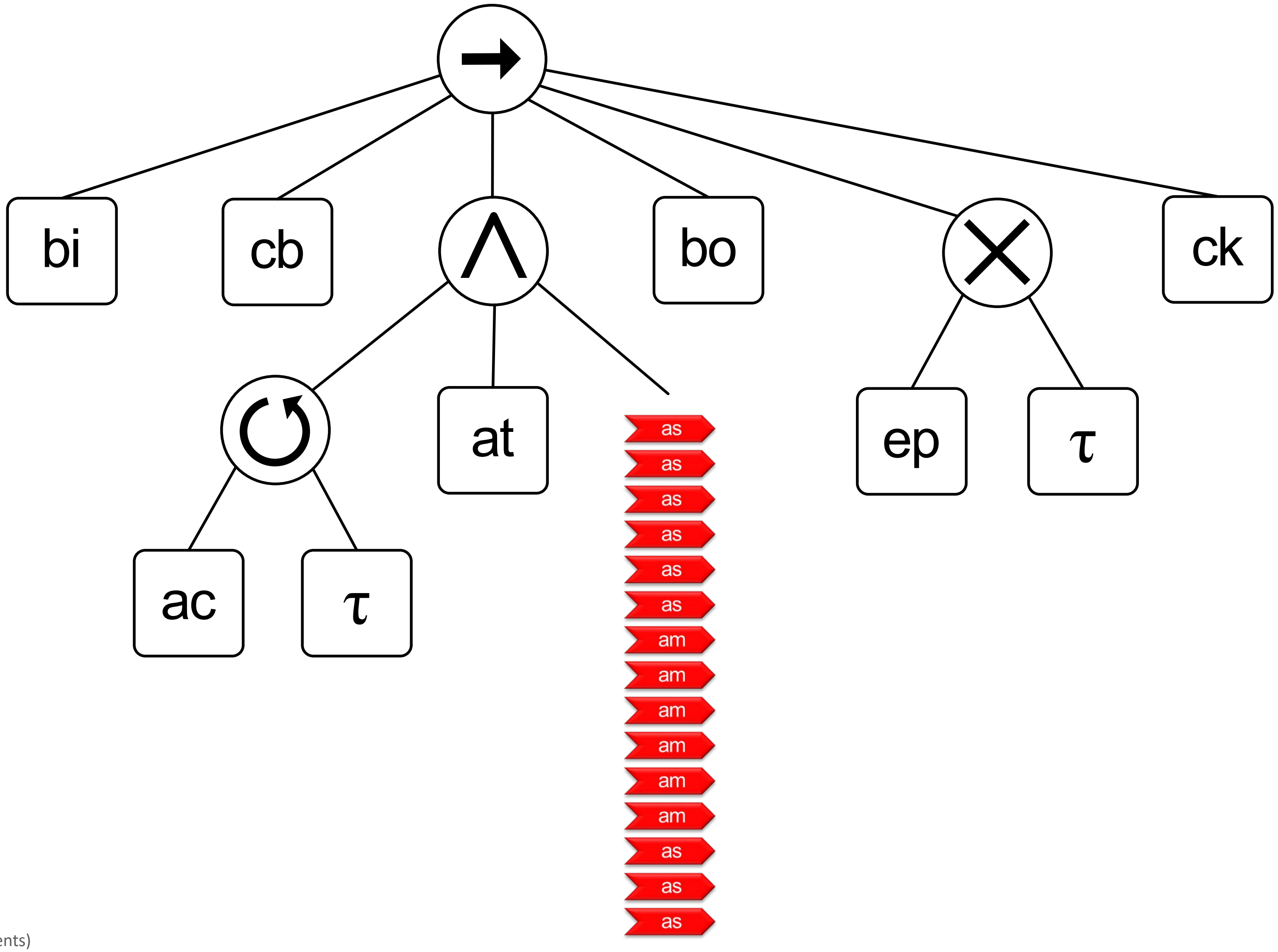
# Handling the base cases (ac can be repeated)






Chair of Process and Data Science

# Only the red event log remains

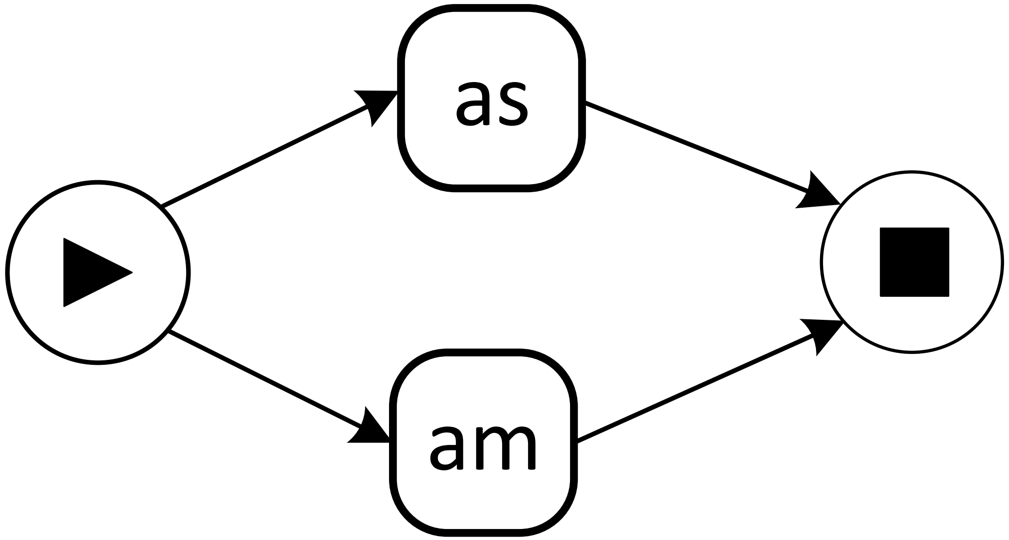





Chair of Process and Data Science

# Continue with the red event log

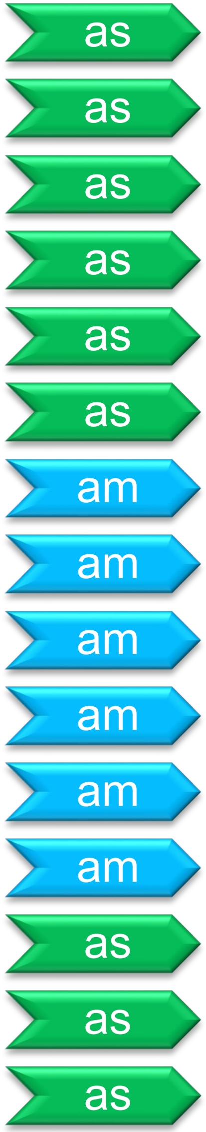
as
as
as
as
as
as
am
am
am
am
am
am
as
as
as



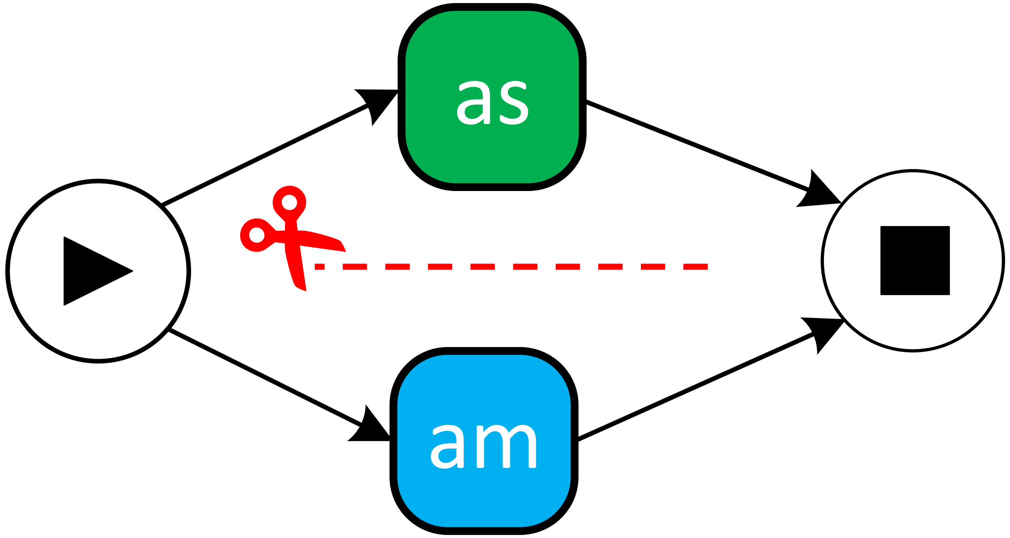



Chair of Process and Data Science

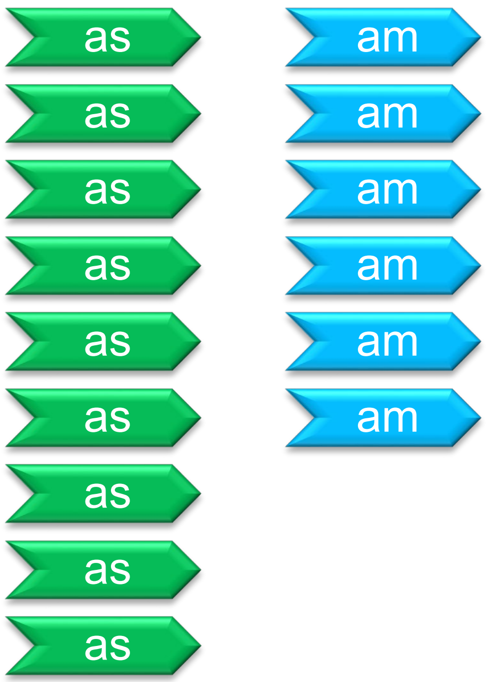
# We find an exclusive-choice cut



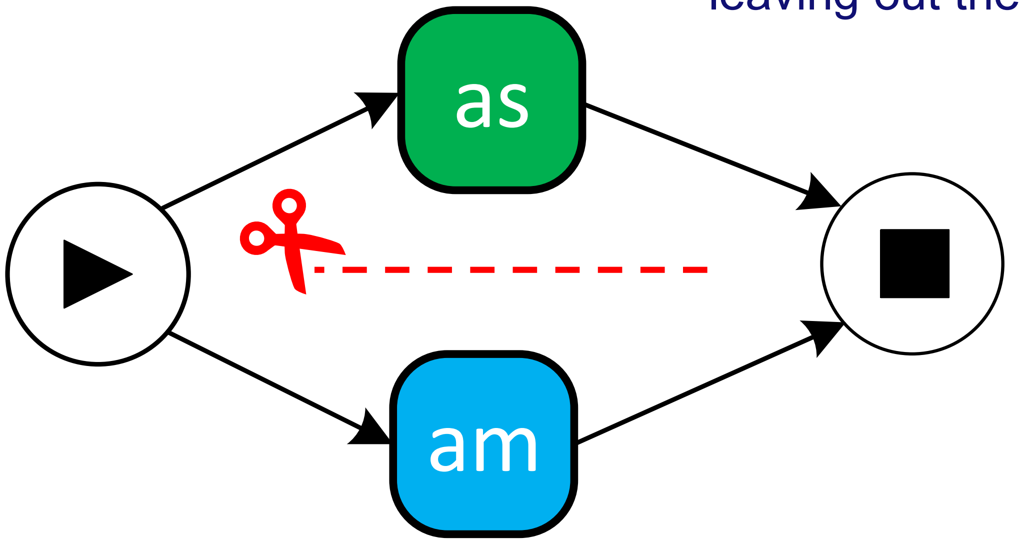

There is an exclusive-choice cut when the DFG can be split into disconnected parts after leaving out the artificial start and end.



# We find an exclusive-choice cut



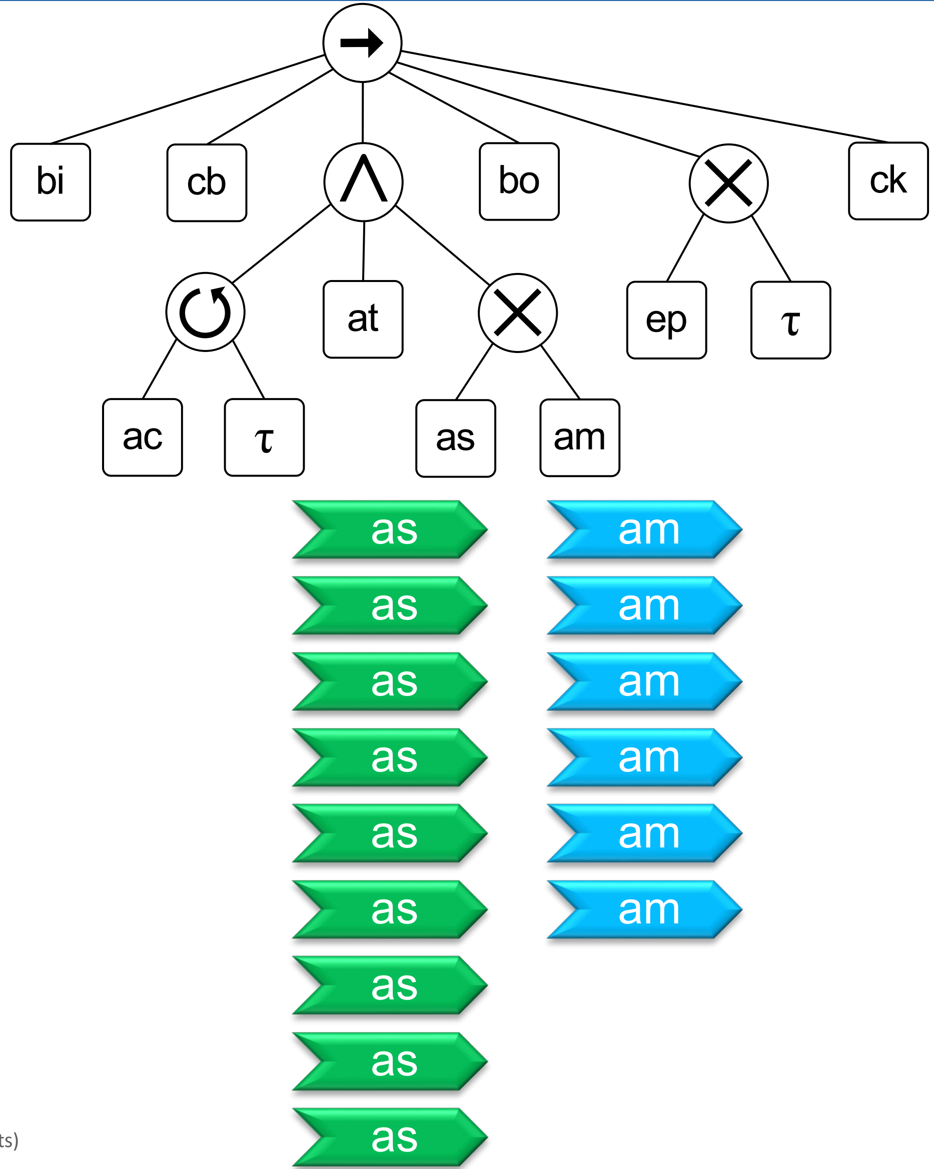

There is an exclusive-choice cut when the DFG can be split into disconnected parts after leaving out the artificial start and end.



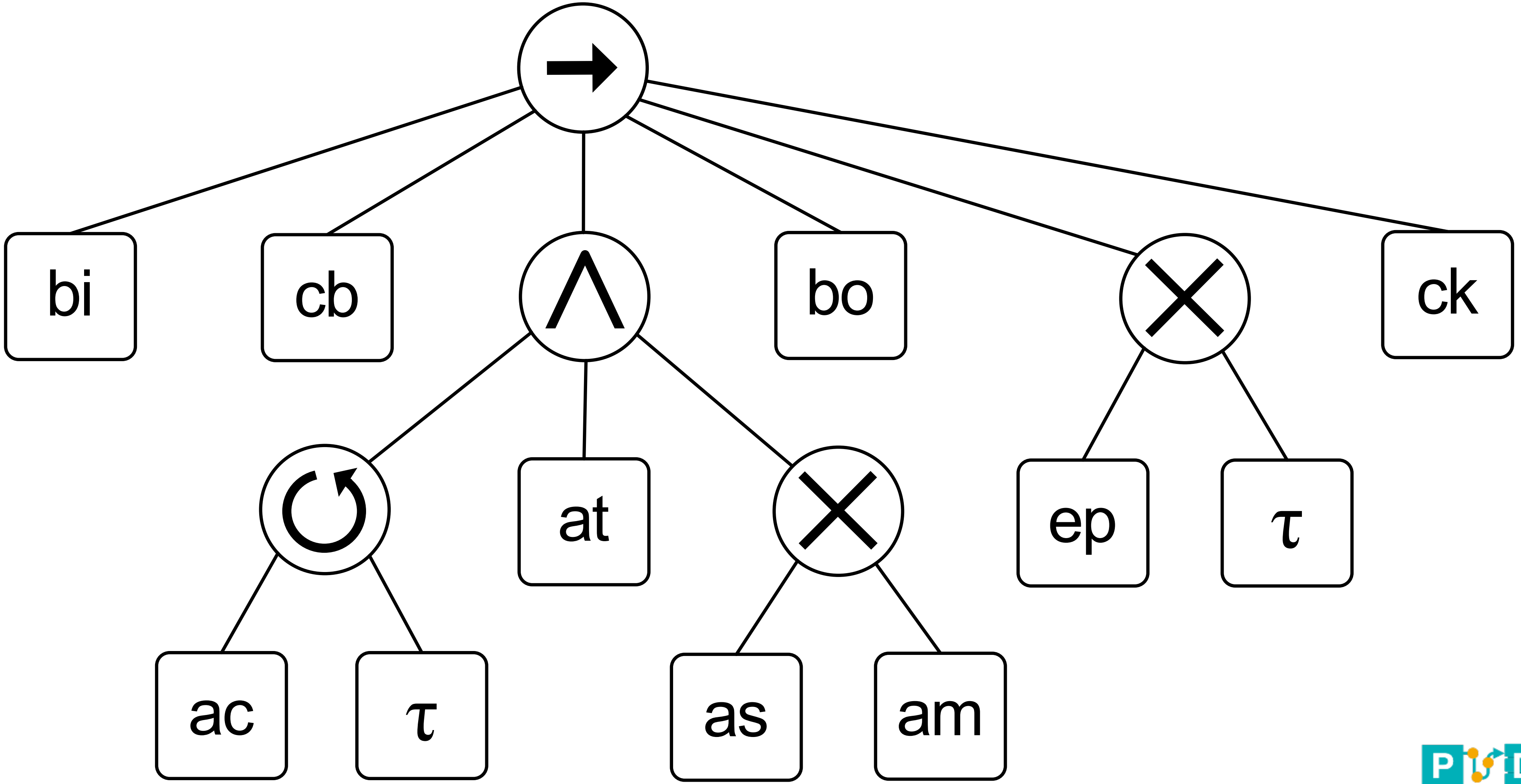

Note that projection is now different than for the sequence and parallel cuts.

Chair of Process and Data Science

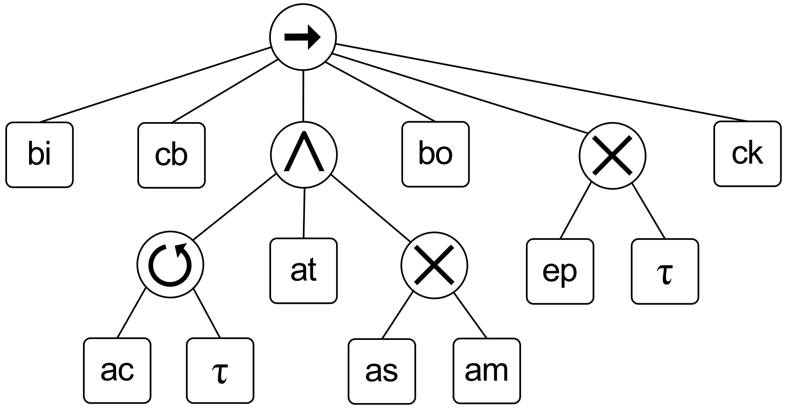
# We end up with two base cases

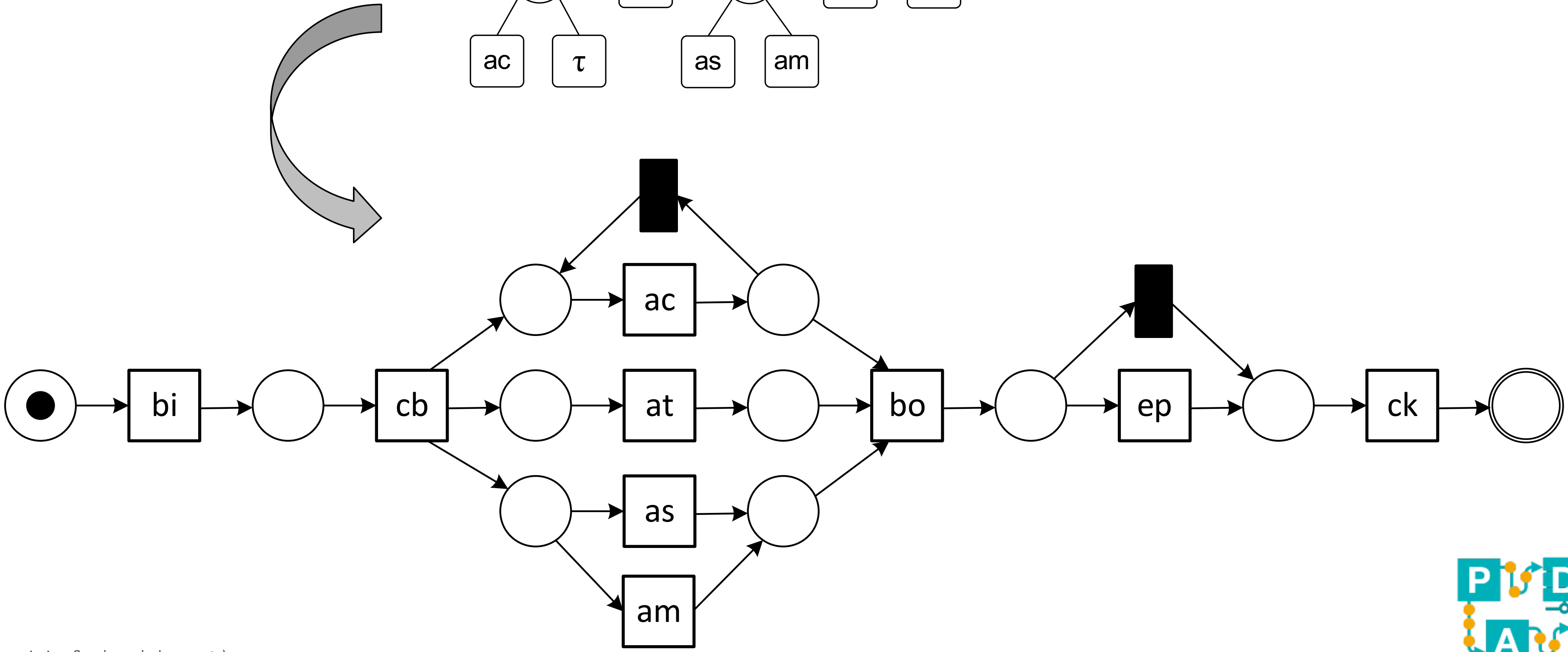


# The process tree returned by the Inductive Mining algorithm



# Can be visualized using Petri nets or BPMN

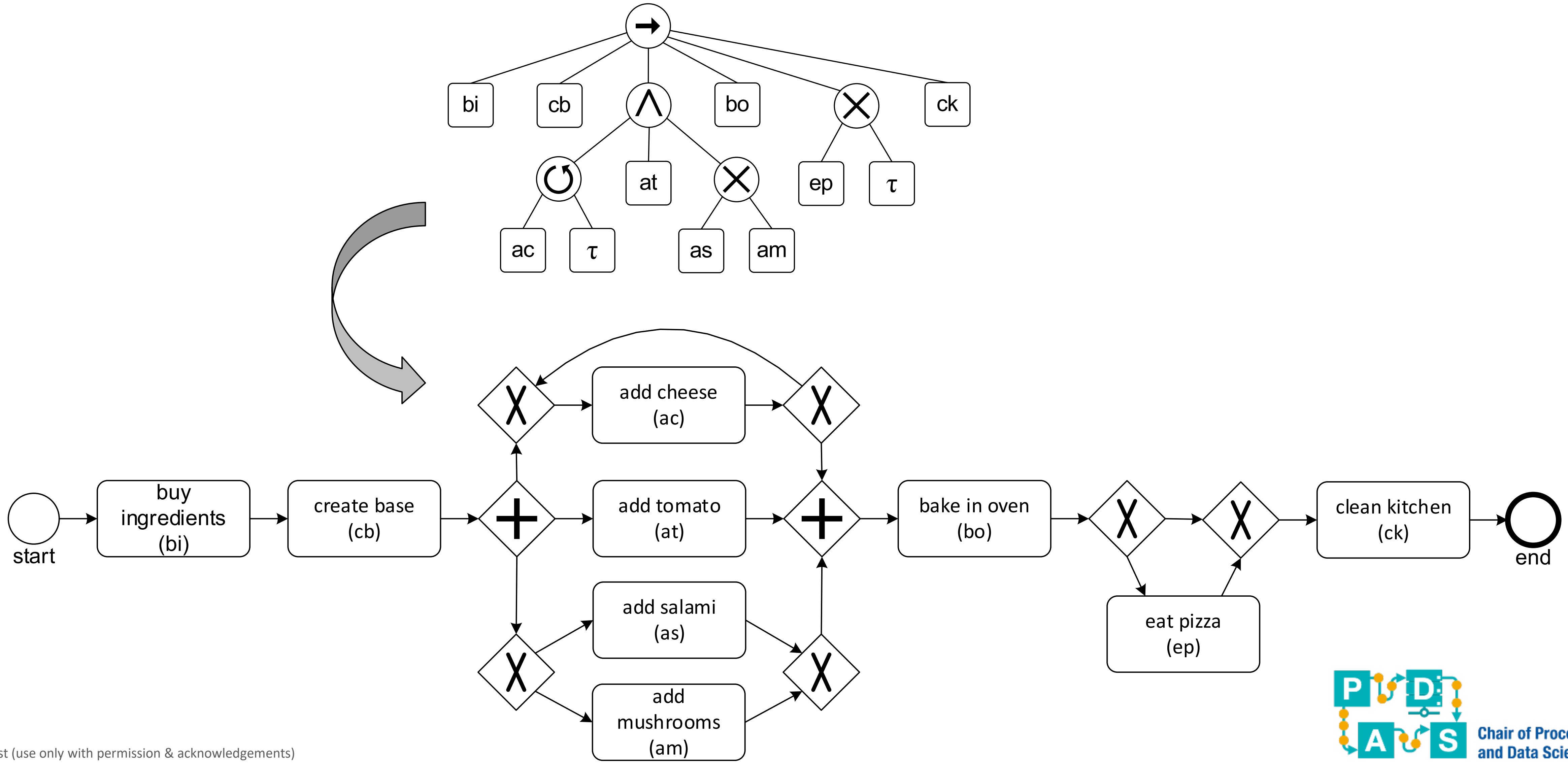





Chair of Process and Data Science

# Can be visualized using Petri nets or BPMN

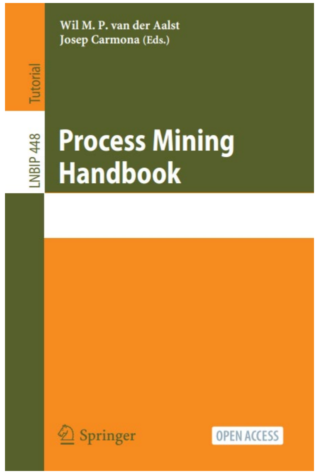


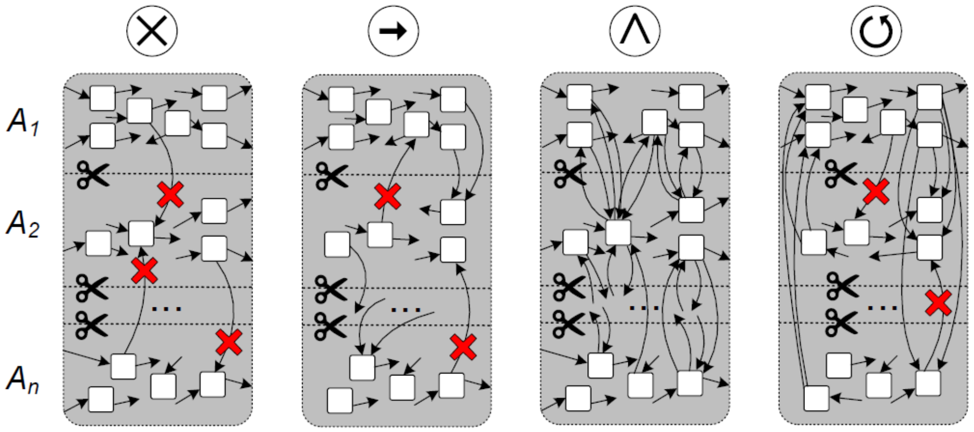

Chair of Process and Data Science

# The details

**Definition 23 (Sequence, Exclusive-Choice, Parallel, and Redo-Loop Cuts).** *Let* $L \in \mathcal{B}(\mathcal{U}_{act}{}^*)$ *be an event log having a DFG* $disc_{DFG}(L) = (A, F)$ *based on* $L$ *(note that* $A = act(L)$*) with start activities* $A^{start} = \{a \in A \mid (\blacktriangleright, a) \in F\}$ *and end activities* $A^{end} = \{a \in A \mid (a, \blacksquare) \in F\}$*. An* $n$*-ary* $\oplus$*-cut of* $L$ *is a partition of* $A$ *into* $n \geq 2$ *pairwise disjoint subsets* $A_1, A_2, \ldots, A_n$ *(i.e.,* $A = \bigcup_{i \in \{1,\ldots,n\}} A_i$ *and* $A_i \cap A_j = \emptyset$ *for* $i \neq j$*) with* $\oplus \in \{\rightarrow, \times, \wedge, \circlearrowleft\}$*. Such a* $\oplus$*-cut is denoted* $(\oplus, A_1, A_2, \ldots A_n)$*. For each type of operator* $\oplus \in \{\rightarrow, \times, \wedge, \circlearrowleft\}$ *specific conditions apply:*

- *An* exclusive-choice cut *of* $L$ *is a cut* $(\times, A_1, A_2, \ldots A_n)$ *such that*
  - $\forall_{i,j \in \{1,\ldots,n\}} \forall_{a \in A_i} \forall_{b \in A_j}\ i \neq j \Rightarrow (a, b) \notin F$.
- *A* sequence cut *of* $L$ *is a cut* $(\rightarrow, A_1, A_2, \ldots A_n)$ *such that*
  - $\forall_{i,j \in \{1,\ldots,n\}} \forall_{a \in A_i} \forall_{b \in A_j}\ i < j \Rightarrow ((a, b) \in F^+ \wedge (b, a) \notin F^+)$. *(Note that* $F^+$ *is the non-reflexive transitive closure of* $F$*, i.e.,* $(a, b) \in F^+$ *means that there is a path from* $a$ *to* $b$ *in the DFG.)*
- *A* parallel cut *of* $L$ *is a cut* $(\wedge, A_1, A_2, \ldots A_n)$ *such that*
  - $\forall_{i \in \{1,\ldots,n\}}\ A_i \cap A^{start} \neq \emptyset \wedge A_i \cap A^{end} \neq \emptyset$ *and*
  - $\forall_{i,j \in \{1,\ldots,n\}} \forall_{a \in A_i} \forall_{b \in A_j}\ i \neq j \Rightarrow (a, b) \in F$.
- *A* redo-loop cut *of* $L$ *is a cut* $(\circlearrowleft, A_1, A_2, \ldots A_n)$ *such that*
  - $A^{start} \cup A^{end} \subseteq A_1$,
  - $\forall_{i,j \in \{2,\ldots,n\}} \forall_{a \in A_i} \forall_{b \in A_j}\ i \neq j \Rightarrow (a, b) \notin F$,
  - $\{a \in A_1 \mid (a, b) \in F \wedge b \notin A_1\} = A^{end}$,
  - $\{a \in A_1 \mid (b, a) \in F \wedge b \notin A_1\} = A^{start}$,
  - $\forall_{(a,b) \in F}\ a \in A_1 \wedge b \notin A_1 \Rightarrow \forall_{a' \in A^{end}}\ (a', b) \in F$, *and*
  - $\forall_{(b,a) \in F}\ a \in A_1 \wedge b \notin A_1 \Rightarrow \forall_{a' \in A^{start}}\ (b, a') \in F$.

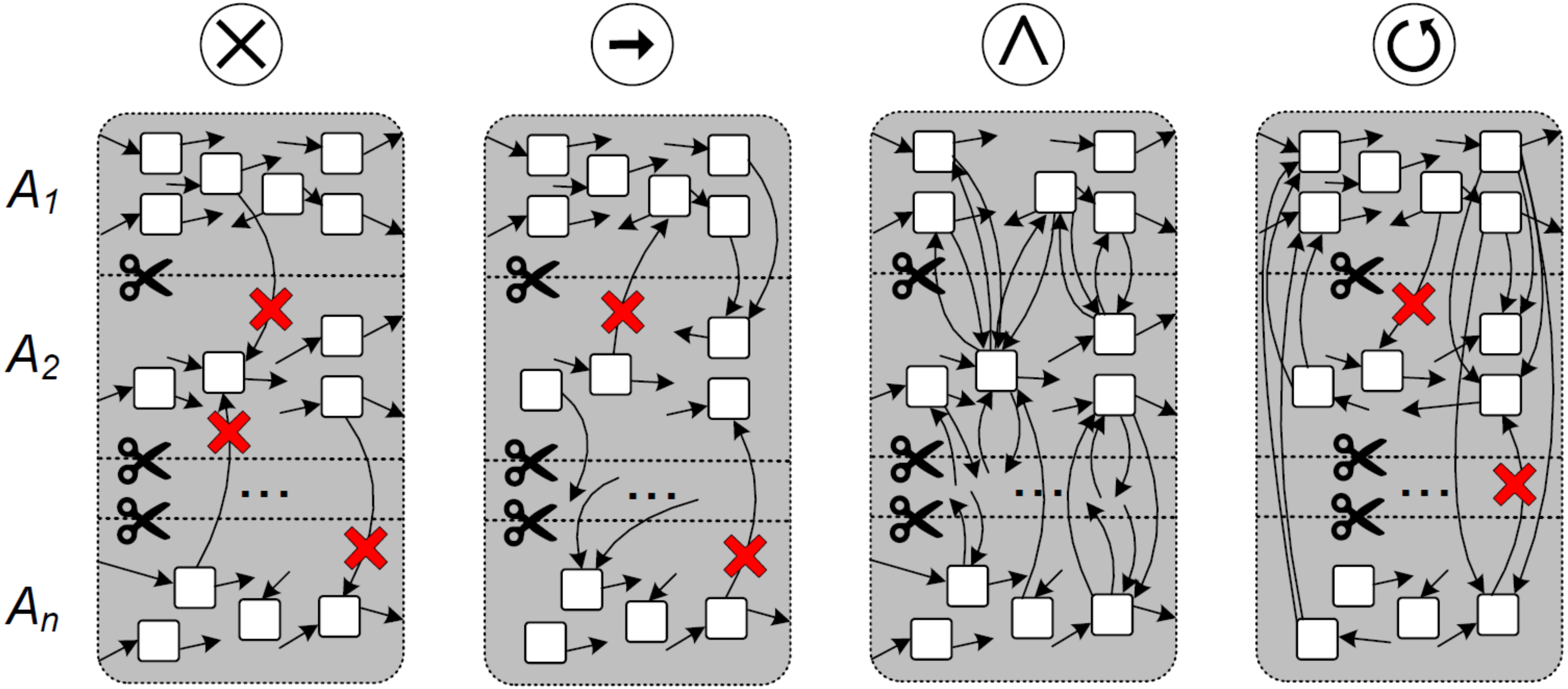



(a) exclusive-choice cut (b) sequence cut (c) parallel cut (d) redo-loop cut

Chair of Process and Data Science

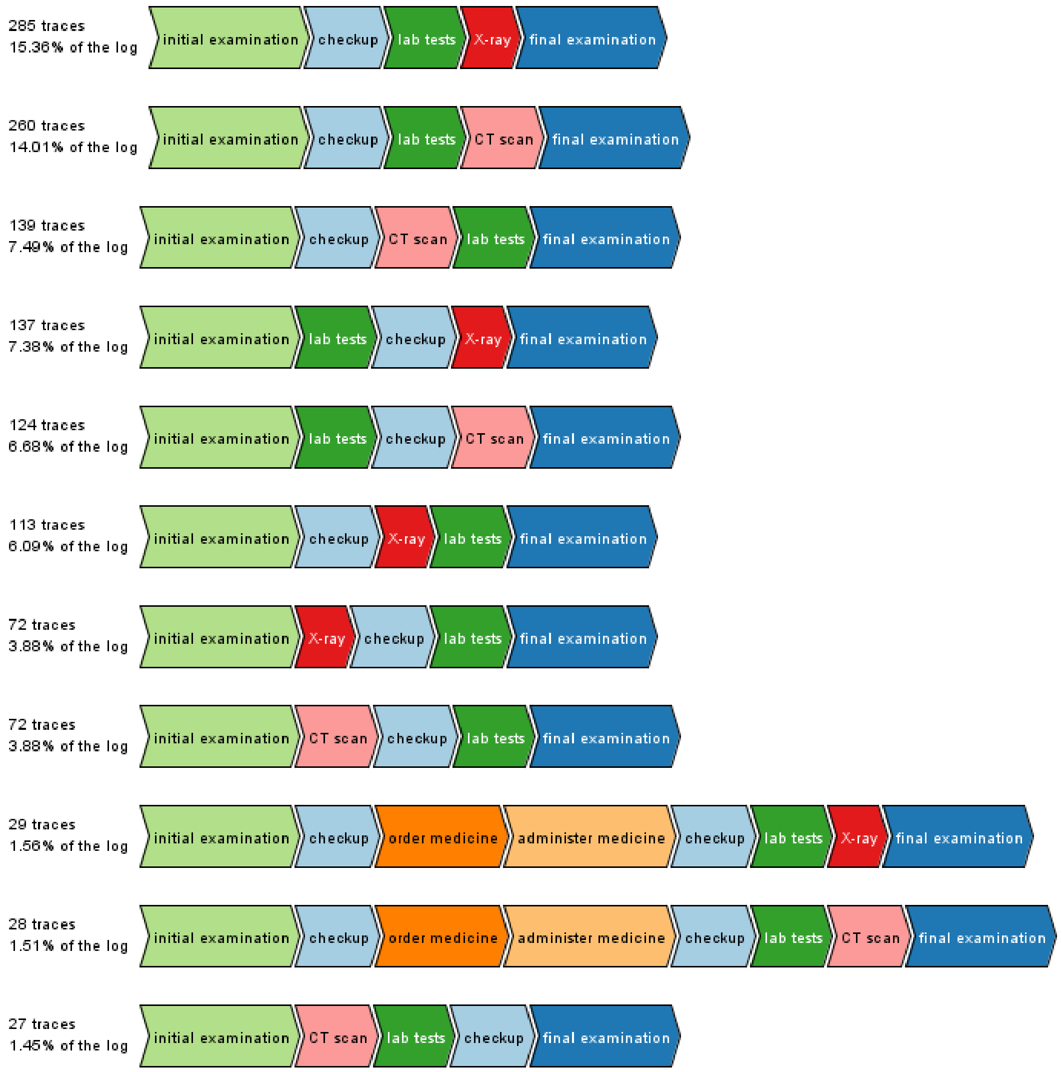
# Four types of cuts



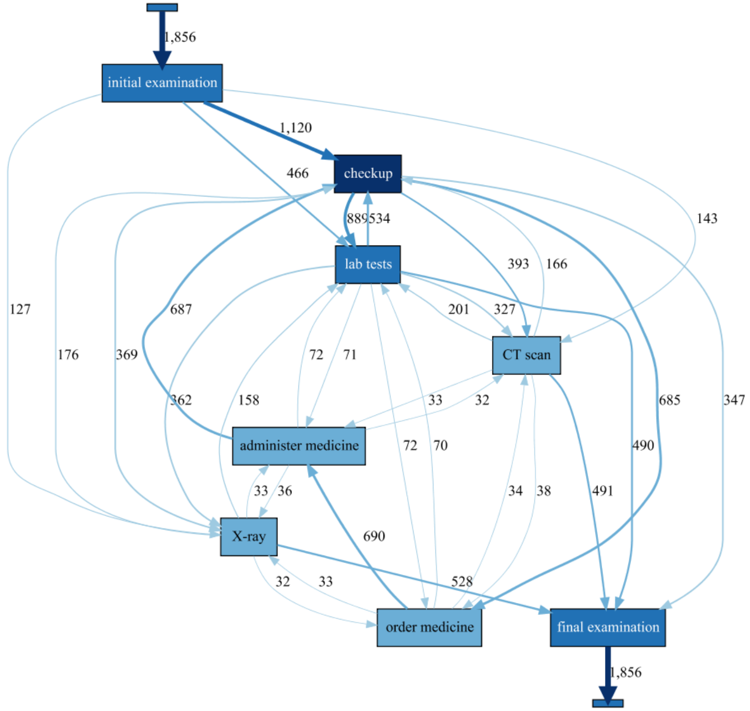

(a) exclusive-choice cut (b) sequence cut (c) parallel cut (d) redo-loop cut

Chair of Process and Data Science

# Another example




- **1856 cases, 197 variants**
- **11761 events**
- **8 unique activities**

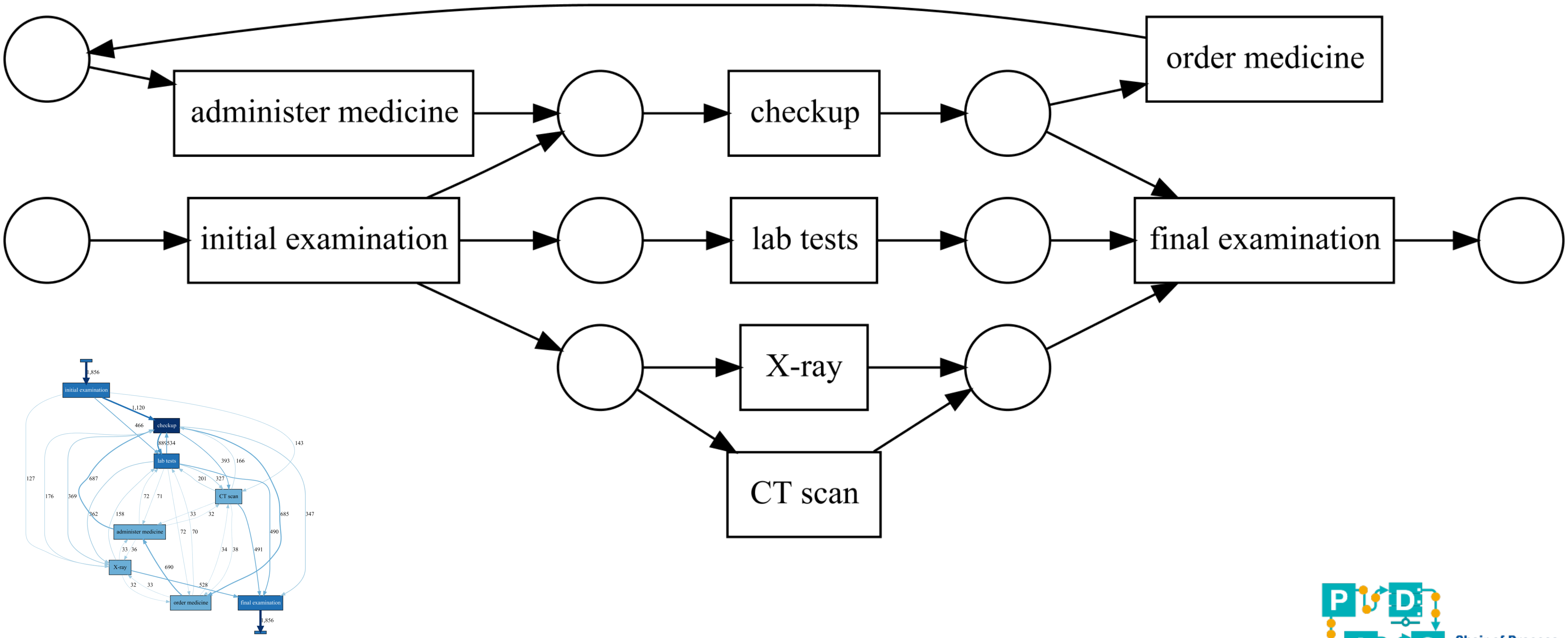



Just 11 of 197 variants



Chair of Process and Data Science

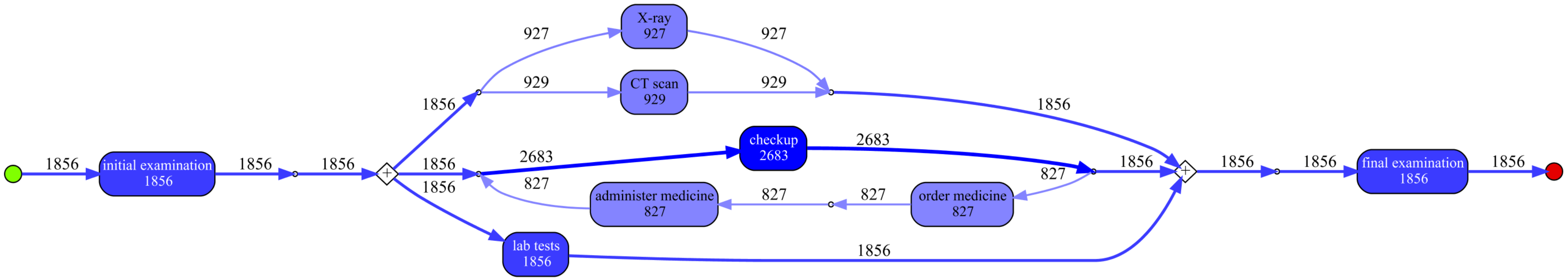
# Alpha algorithm (ProM)

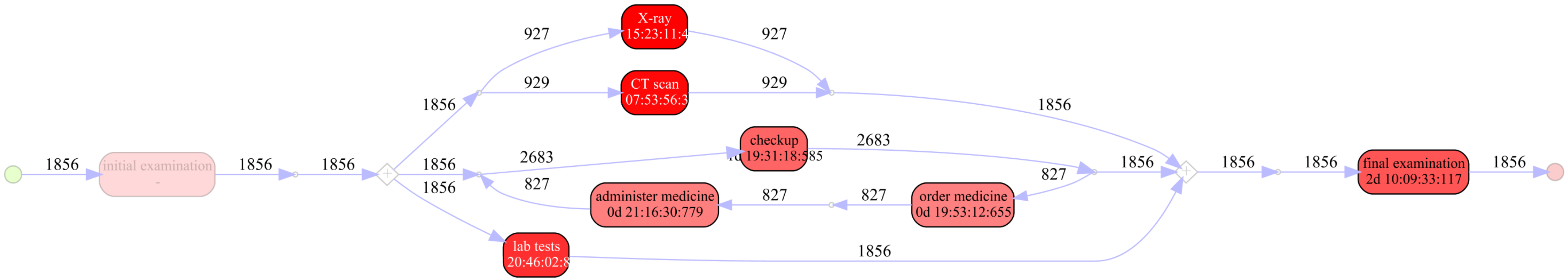





Chair of Process and Data Science

# Inductive visual miner (ProM)

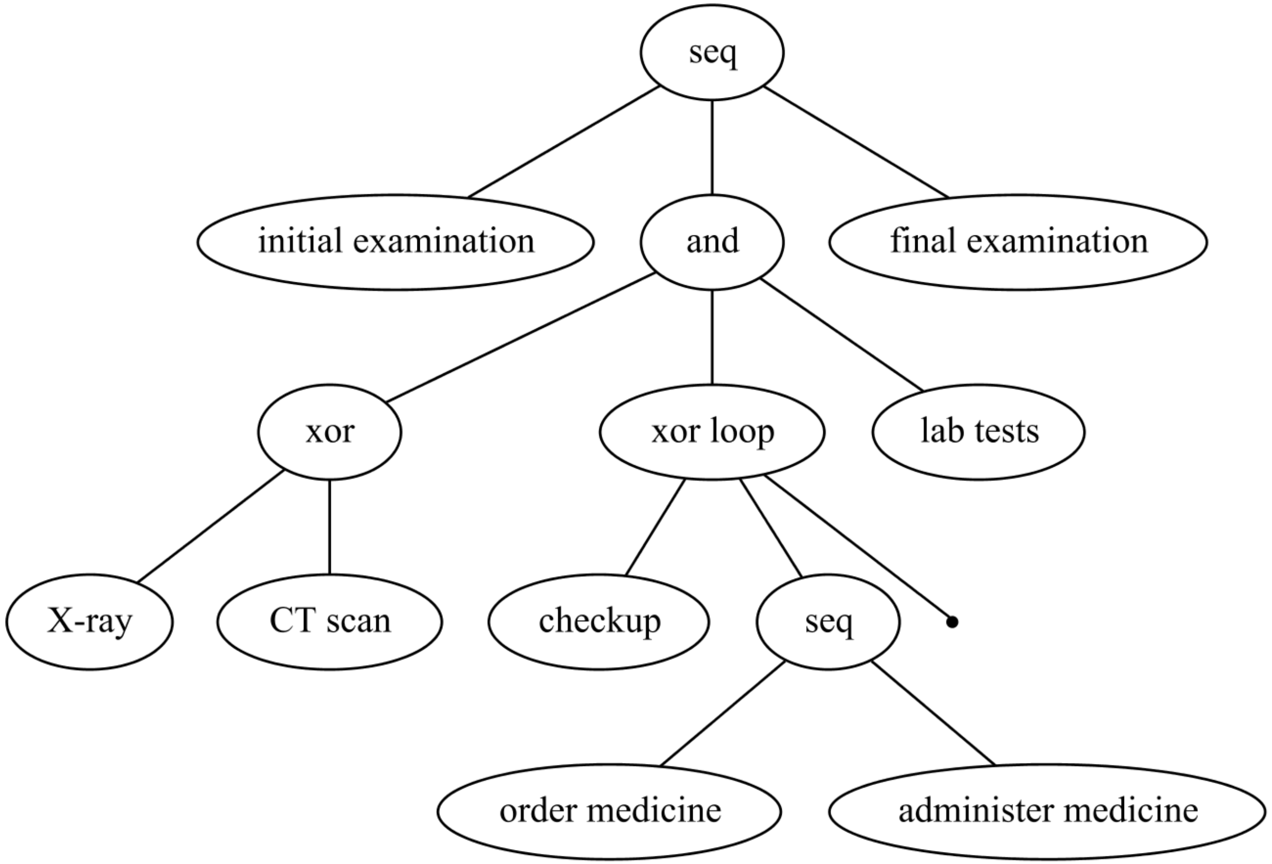


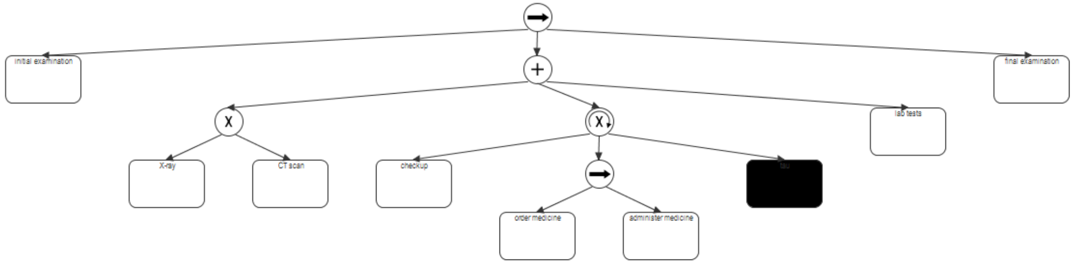



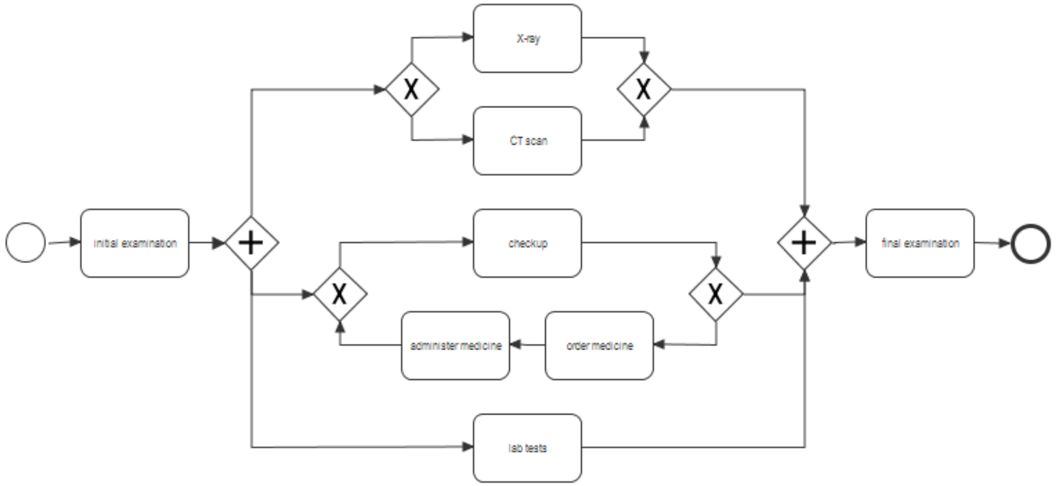



Chair of Process and Data Science

# Different visualizations in ProM



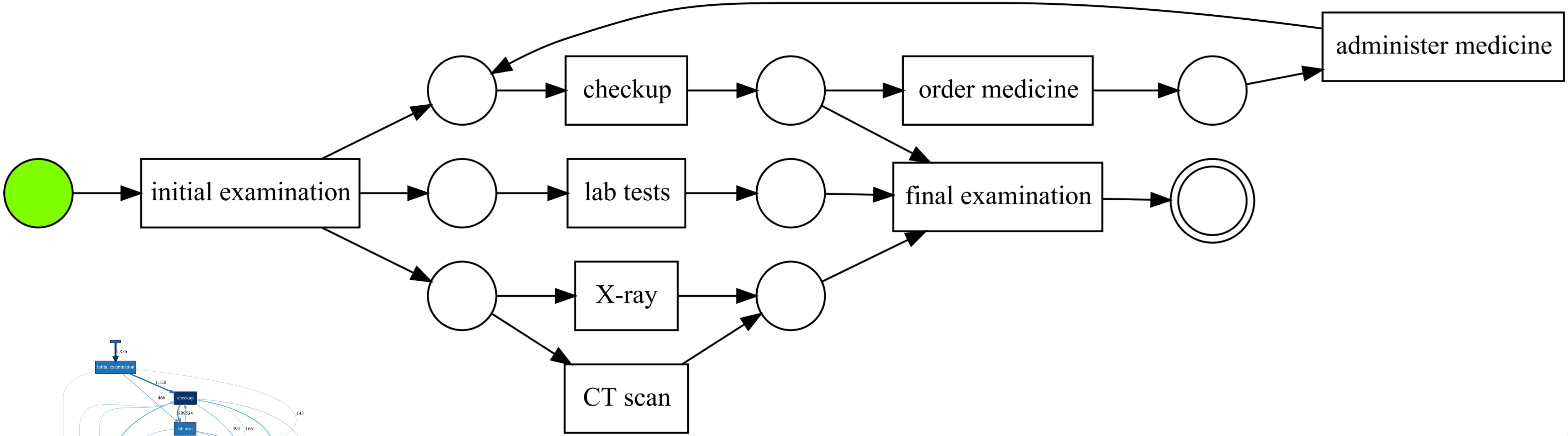



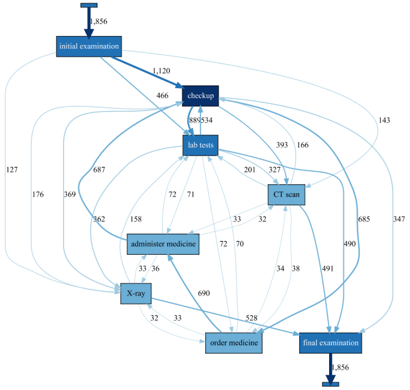





Chair of Process and Data Science

# Mapped onto an accepting Petri net

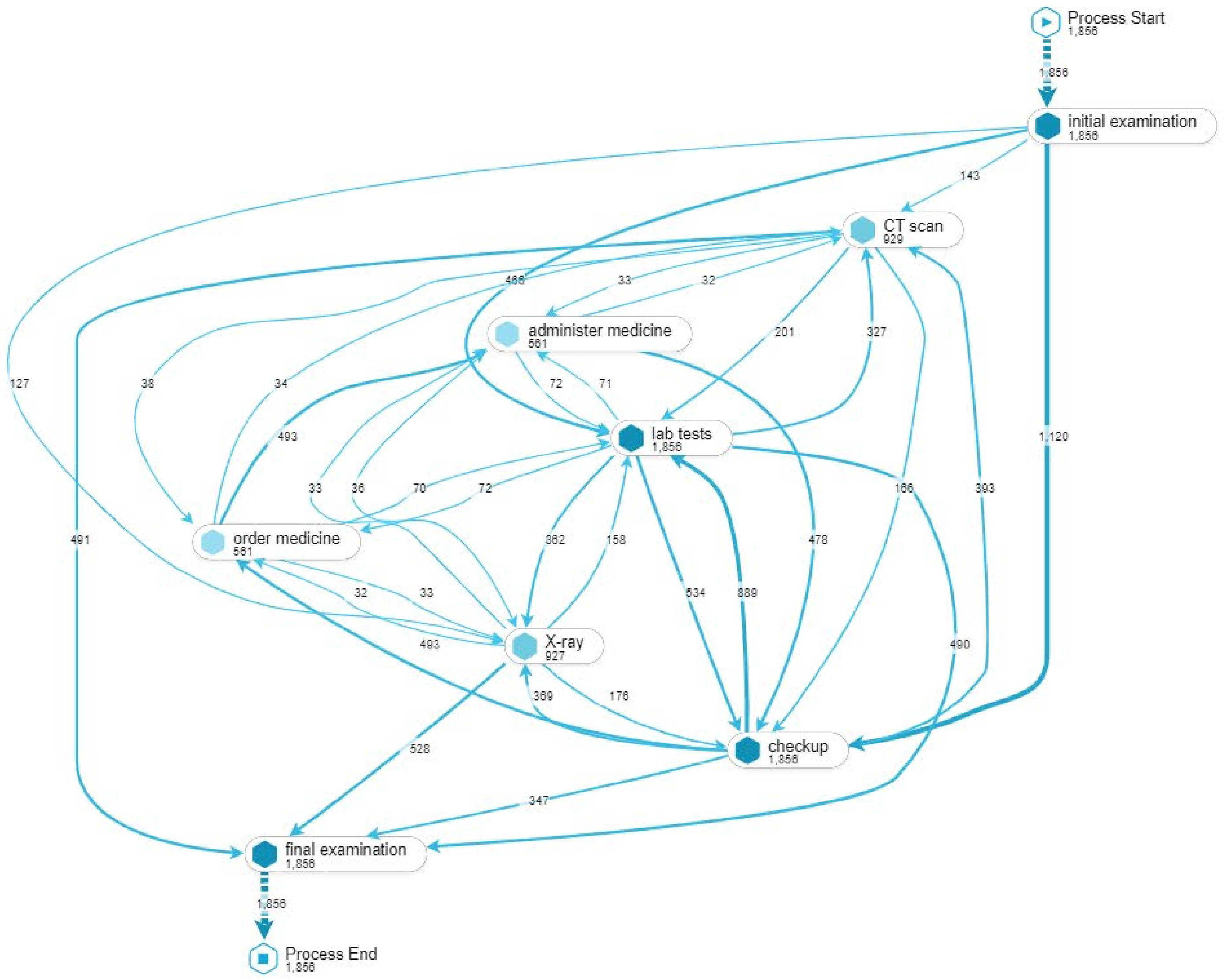


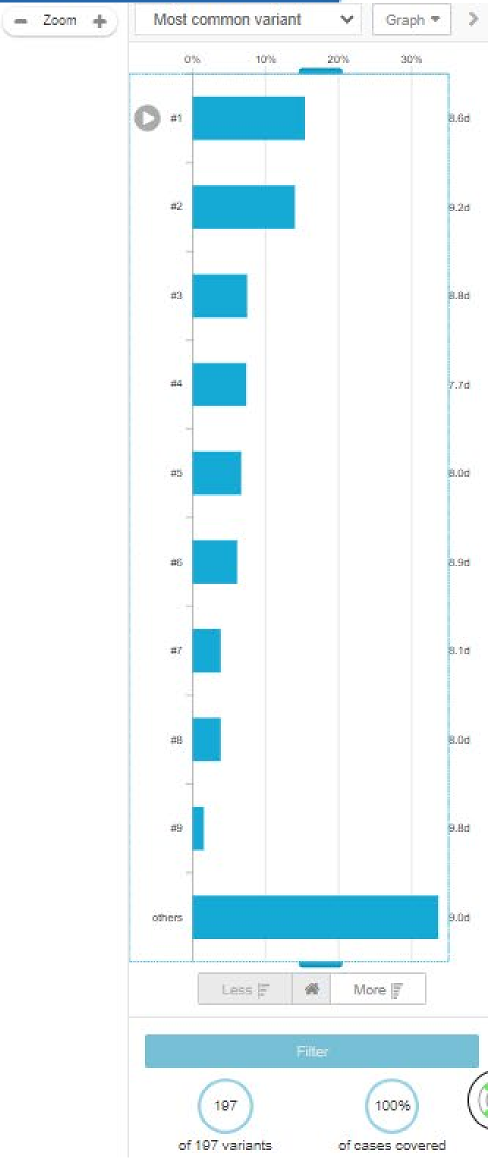





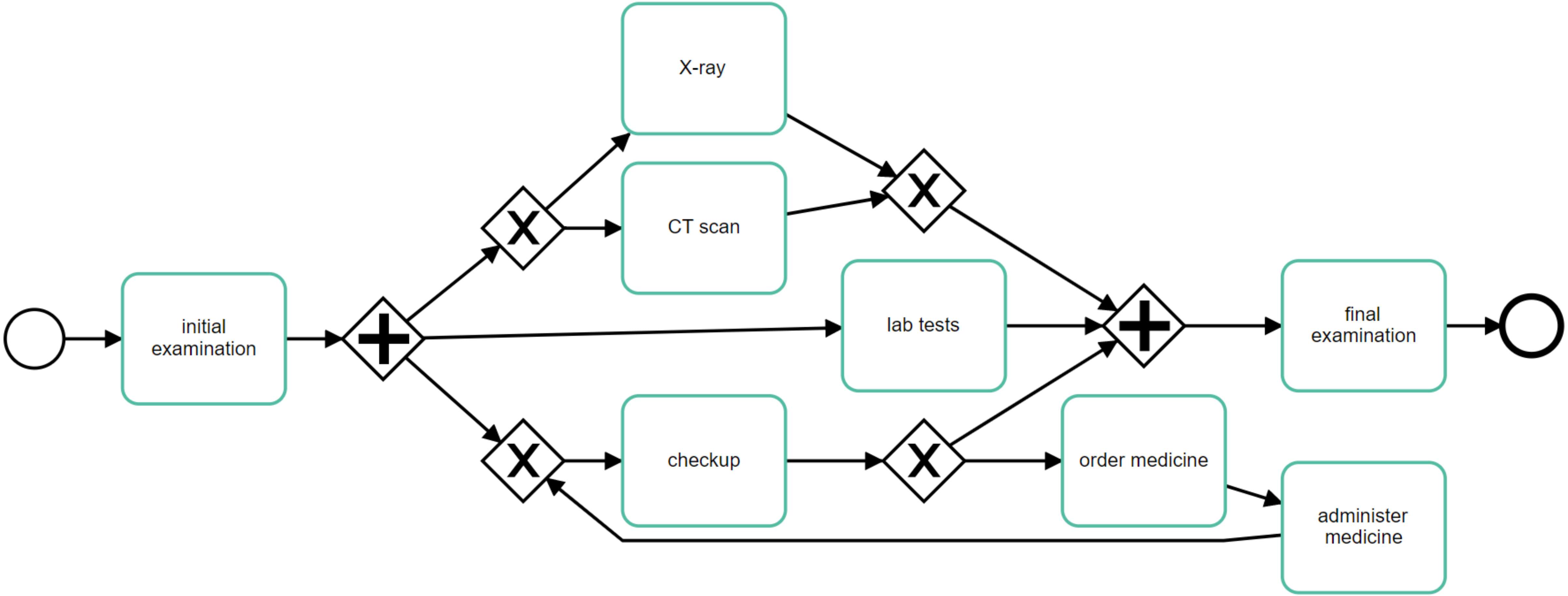

Chair of Process and Data Science

# Celonis also reports 1856 cases, 197 variants, and 11761 events



# Celonis finds the same process tree using the Inductive Mining algorithm

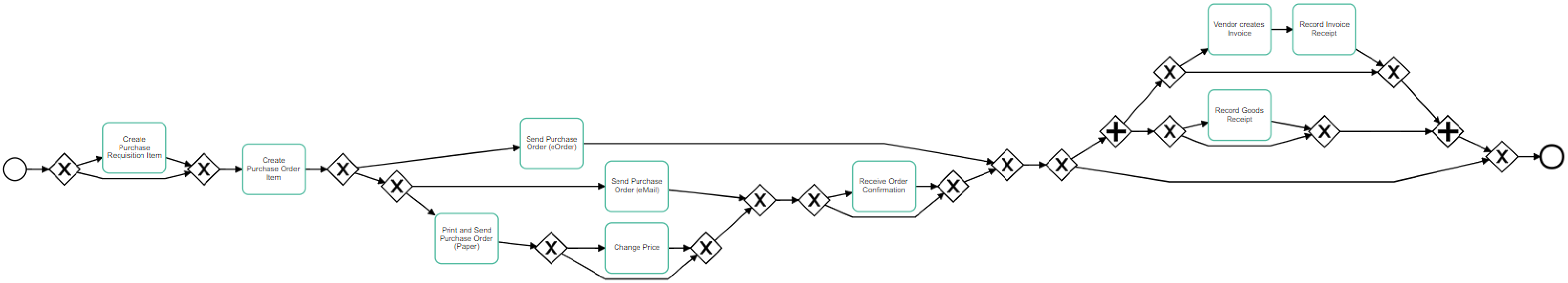


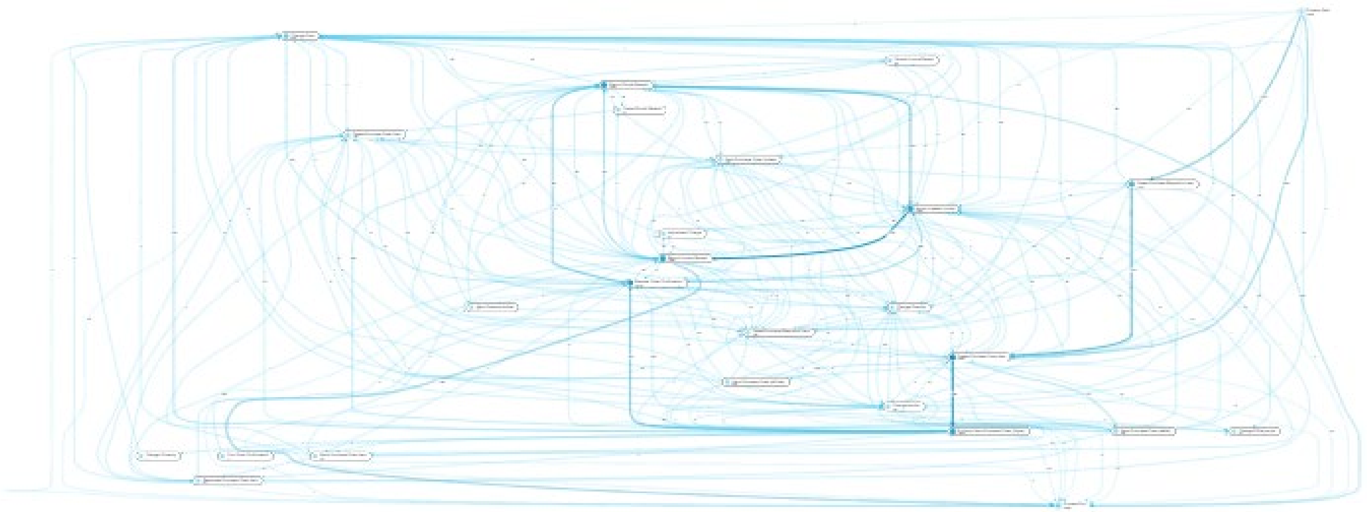



Chair of Process and Data Science

# Also works well on large real-life event logs

## (but you need to put in the work)



Chair of Process and Data Science

# Summary: Inductive Mining

- The models are guaranteed to be sound, i.e., no deadlocks, no livelocks, and no other anomalies.
- The basic algorithm guarantees that the event log can be reproduce completely (of course one can filter if desired).
- The algorithm has good performance (and there are also more scalable variants) and implemented in several tools.
- There are various additional theoretical guarantees, i.e., rediscover the process tree used to create the event log.

# Conclusion

# Foundations of Process Discovery

## Baseline: Discovering DFG + filtering

2 lines of mathematics

## Bottom-up discovery

### Alpha algorithm

8 lines of mathematics

## Top-down discovery

### Inductive mining

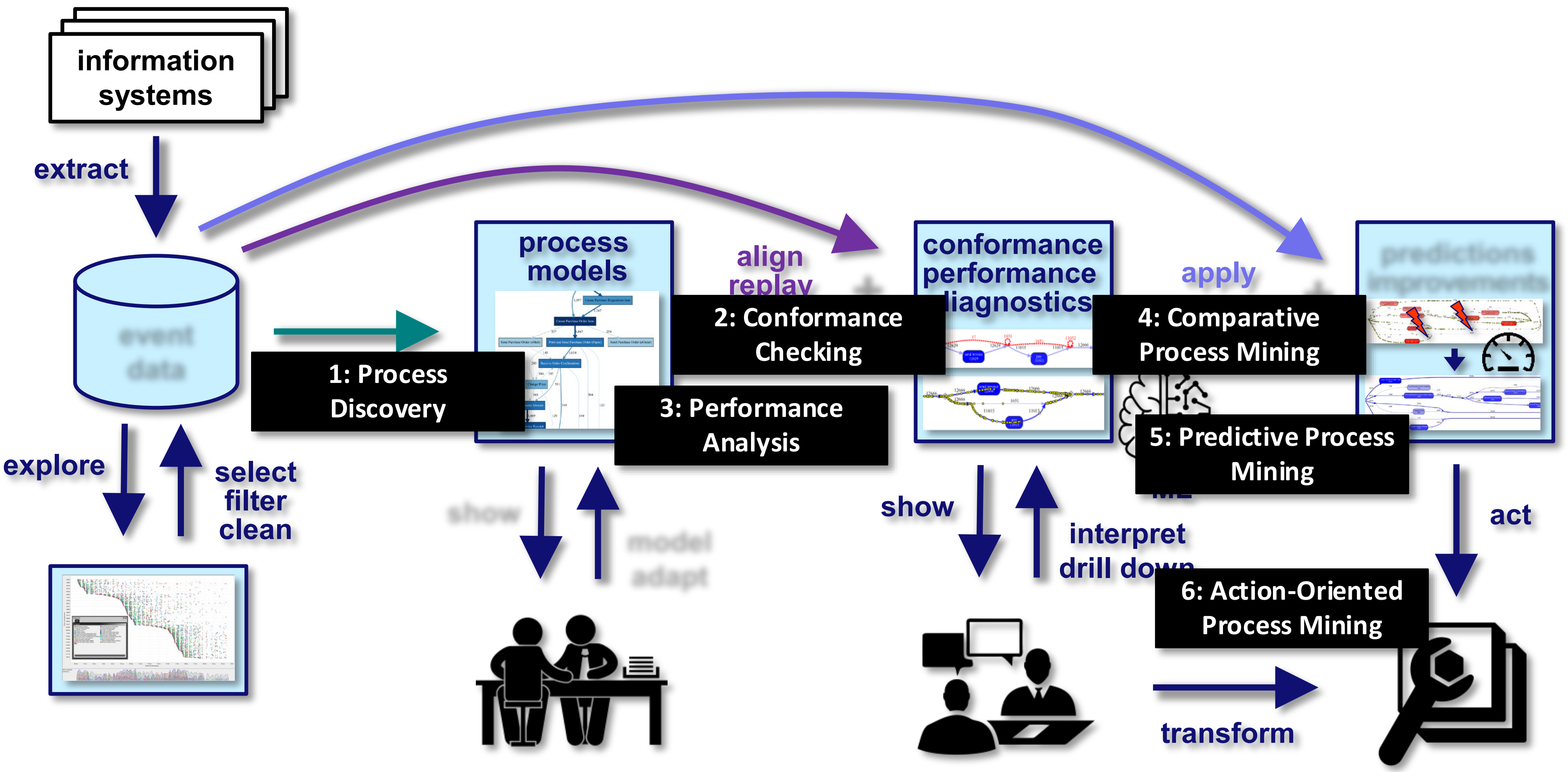
approximately 20 lines of mathematics

**Not a solved problem!**



Chair of Process and Data Science

# Discovery is just one of many techniques






Chair of Process and Data Science

# Websites

- www.processmining.org
- www.process-mining-summer-school.org
- www.tf-pm.org
- www.promtools.org
- www.celonis.com/academic-signup
- xes-standard.org
- ocel-standard.org
- www.pads.rwth-aachen.de
- www.vdaalst.com


Chair of Process and Data Science

# Online courses

- **Coursera course**
  **“Process Mining: Data science in Action”**
  **Register via coursera.org/learn/process-mining (152.345 participants since 2015).**

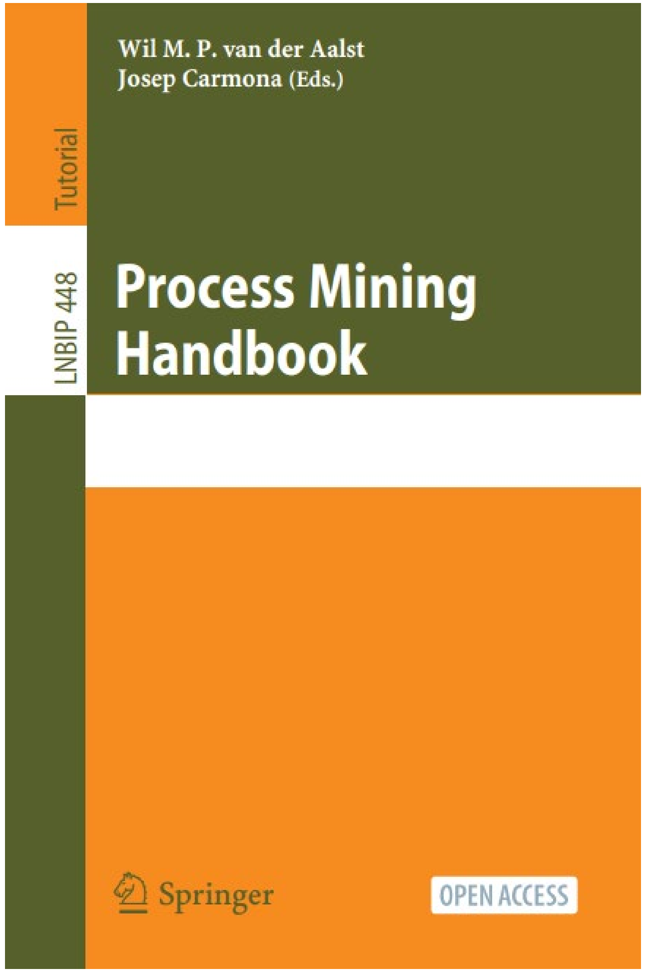
- **Celonis/RWTH course**
  **“Process Mining: From Theory to Execution”**
  **Register via www.celonis.com/wils-process-mining-class.**

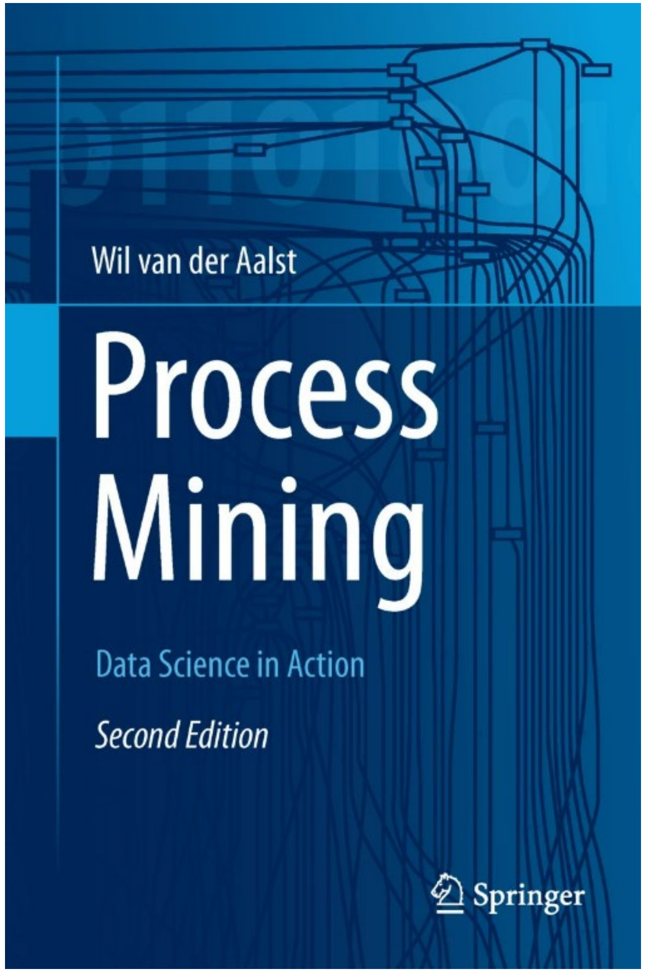

coursera TU/e
celonis RWTH RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

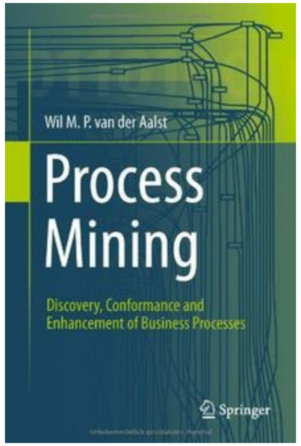
(edX is coming)

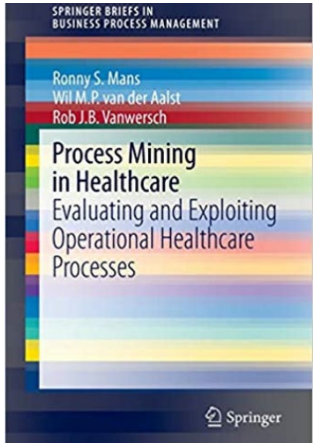
PADS Chair of Process and Data Science

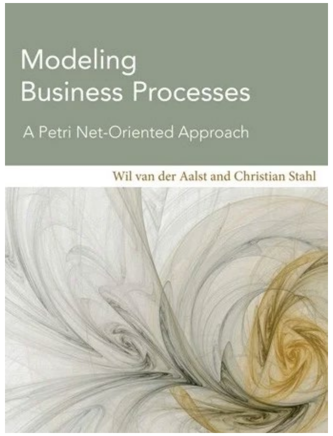
# Books (not intended to be complete)

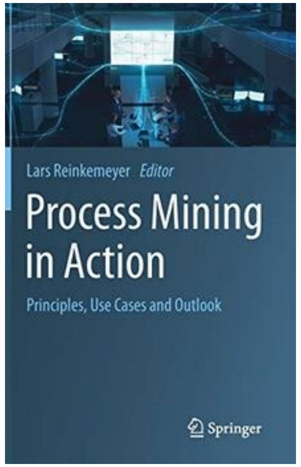


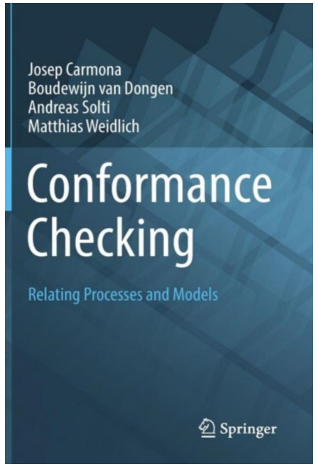





Chair of Process and Data Science