Business Process Intelligence Course
〈 Lecture 2 〉

Process Discovery:
The $\alpha$ Algorithm

prof.dr.ir. Wil van der Aalst
www.processmining.org

©Wil van der Aalst TU/e (use only with permission & acknowledgements)
Overview

Part I: Preliminaries

- Chapter 2: Process Modeling and Analysis
- Chapter 3: Data Mining

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 III: Beyond Process Discovery

- Chapter 7: Conformance Checking
- Chapter 8: Mining Additional Perspectives
- Chapter 9: Operational Support

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
Process Discovery

event log

process model
process models
"Business Process Management (BPM) is the discipline that combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes"
Productivity improvements (selection)

- **Adam Smith** (1723-1790) showed the advantages of the division of labor. (rational self-interest and competition can lead to economic prosperity)

- **Frederick Taylor** (1856-1915) introduced the initial principles of scientific management. (replace rule-of-thumb work methods with methods based on a scientific study of the tasks)

- **Henry Ford** (1863-1947) introduced the production line for the mass production of “black T-Fords”.

- Since 1950 computers and digital communication infrastructures are the most dominant factor influencing business processes and their management.

- Data science: the next level!
Role of (process) models

• **Operations management** (in particular operations research) is a branch of management science heavily relying on modeling and analysis.

• **Models are used**
  - to reason *about processes* (redesign) and
  - to make decisions *inside processes* (planning and control).

• **Process models** may be used to:
  - discuss responsibilities,
  - analyze compliance,
  - predict performance using simulation, and
  - configure a WFM/BPM system.
BPM lifecycle

- (re)design
- run & adjust
- implement/configure

model-based analysis

data-based analysis
Different notations

Business Process Modeling Notation (BPMN)

Petri nets (often the subclass of workflow nets)

Event-Driven Process Chains (EPCs)

Event log (example/observed behavior)
Petri nets

AND-split

XOR-split

XOR-join

AND-join

XOR-split

start

register request

examine thoroughly

examine casually

check ticket

decide

reinitiate request

pay compensation

reject request

date

transition

place

token
Marking is a multiset of places

[start] [c1, c3, c4^2] [c5^3, end^2]
Semantics

[start]  [c1,c2]  [c2,c3]  [c3,c4]  [c5]  [end]

Only one of infinitely many possible firing sequences!
Reachability graph
How many states?

(a)  

(b)  

(c)
Good model?

- Start
- Register request
- Examine casually
- Examine thoroughly
- Check ticket
- Decide
- Pay compensation
- Reject request
- Reinitiate request
- End

The diagram illustrates a process flow with decision points and actions such as registering a request, examining it casually or thoroughly, checking the ticket, deciding on compensation, and handling rejection or reinitiation of the request.
Good model?

start register request

c1

examine thoroughly

c2

examine casually

check ticket

c3
decide

c5

reinitiate request

pay compensation

reject request

end
Good model?
Good model?

- start
- register request
- examine casually
- examine thoroughly
- check ticket
- decide
- c5
- pay compensation
- reject request
- reinitiate request
- end
A WorkFlow net (WF-net) has one source place (typically called `start` or `i`) and one sink place (typically called `end` or `o`) and all other nodes are on a path from source to sink.

A WF-net is **sound** if and only if the following properties hold:

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
Sound?

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
Sound?

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
Sound?

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
Sound?

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
Sound?

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
Sound?

Examples of reachable markings:
- \([c5^2]\]
- \([c1,c2,c3,c4]\]
- \([c1^2,c2^2]\]
- etc.

- **safeness:** places cannot hold multiple tokens at the same time,
- **proper completion:** if the sink place is marked, all other places are empty,
- **option to complete:** it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts:** for any transition there is a firing sequence enabling it.
Sound?

- **safeness**: places cannot hold multiple tokens at the same time,
- **proper completion**: if the sink place is marked, all other places are empty,
- **option to complete**: it is always possible to reach the marking that marks just the sink place, and
- **absence of dead parts**: for any transition there is a firing sequence enabling it.
No need to check proper completion: It is implied by other properties

**option to complete** (it is always possible to reach the marking that marks just the sink place) implies **proper completion** (if the sink place is marked all other places are empty)
Model-based analysis

verification (like soundness checking)

performance analysis (simulation)
Limitations of model-based analysis

- Verification and performance analysis heavily rely on the availability of high quality models.
- When the models and reality have little in common, model-based analysis does not make much sense.
- There is often a poor alignment between hand-made models and reality.
- Process mining aims to address these problems by establishing a direct connection between the models and actual low-level event data about the process.
- Process discovery techniques allow for viewing the same reality from different angles and at different levels of abstraction.
More general: Problems of models

• The model describes an idealized version of reality.
• Inability to adequately capture human behavior.
• The model is at the wrong abstraction level.
• Therefore, we advocate the use of event data:
  - Process mining allows for the extraction of models based on facts.
  - Moreover, process mining does not aim at creating a single model of the process.
  - Instead, it provides various views on the same reality at different abstraction levels.
  - For example, users can decide to look at the most frequent behavior to get a simple model ("80% model").
  - However, they can also inspect the full behavior by deriving the "100% model" covering all cases observed.
process discovery
Process discovery

“world”

- people
- machines
- components
- organizations

business processes

models analyzes

supports/controls

software system

records events, e.g., messages, transactions, etc.

specifies configures implements analyzes

(event)

model

(discovery)

conformance

enhancement

©Wil van der Aalst TU/e (use only with permission & acknowledgements)
Process discovery = Play-In

Play-In
- event log
- process model

Play-Out
- process model
- event log

Replay
- event log
- process model
- extended model showing times, frequencies, etc.
- diagnostics
- predictions
- recommendations
Event log contains all possible traces of model and vice versa.

\[ L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle] \]
Generalization: event log contains only subset of all possible traces of model.
Notation is less relevant (e.g. BPMN)

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

\[ L_2 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle, \langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle] \]
In general, there is a trade-off between the following four quality criteria:

1. **Fitness**: the discovered model should allow for the behavior seen in the event log.

2. **Precision (avoid underfitting)**: the discovered model should not allow for behavior completely unrelated to what was seen in the event log.

3. **Generalization (avoid overfitting)**: the discovered model should generalize the example behavior seen in the event log.

4. **Simplicity**: the discovered model should be as simple as possible.
α
Process Discovery: example of algorithm
>, →, ||, # relations

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

- Direct succession: \( x \succ y \) iff for some case \( x \) is directly followed by \( y \).
- Causality: \( x \rightarrow y \) iff \( x \succ y \) and not \( y \succ x \).
- Parallel: \( x \parallel y \) iff \( x \succ y \) and \( y \succ x \).
- Choice: \( x \# y \) iff not \( x \succ y \) and not \( y \succ x \).
Basic Idea Used by $\alpha$ Algorithm (1)

(a) sequence pattern: $a \rightarrow b$
Basic Idea Used by $\alpha$ Algorithm (2)

(b) XOR-split pattern: $a \rightarrow b$, $a \rightarrow c$, and $b \# c$

(c) XOR-join pattern: $a \rightarrow c$, $b \rightarrow c$, and $a \# b$
Basic Idea Used by $\alpha$ Algorithm (3)

(d) AND-split pattern:
\[ a \rightarrow b, \ a \rightarrow c, \text{ and } b \parallel c \]

(e) AND-join pattern:
\[ a \rightarrow c, \ b \rightarrow c, \text{ and } a \parallel b \]
Example Revisited

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

Result produced by \( \alpha \) algorithm

\begin{align*}
a &> b \\
a &> c \\
a &> e \\
b &> c \\
b &> d \\
c &> b \\
c &> d \\
e &> d
\end{align*}
Footprint of $L_1$

$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$
Footprint of $L_2$

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

![Diagram of a Petri net with places labeled p1 to p5 and transitions labeled a to f.]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>#</td>
<td>→</td>
<td>→</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>b</td>
<td>←</td>
<td>#</td>
<td>→</td>
<td>→</td>
<td>←</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>←</td>
<td>→</td>
<td>#</td>
<td>→</td>
<td>←</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>#</td>
<td>←</td>
<td>←</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>e</td>
<td>#</td>
<td>←</td>
<td>←</td>
<td>#</td>
<td>#</td>
<td>→</td>
</tr>
<tr>
<td>f</td>
<td>#</td>
<td>→</td>
<td>→</td>
<td>#</td>
<td>←</td>
<td>#</td>
</tr>
</tbody>
</table>
Simple patterns

(a) sequence pattern: $a \rightarrow b$

(b) XOR-split pattern:

$\begin{align*}
&a \rightarrow b, \\
&b \rightarrow c, \\
&c \rightarrow b
\end{align*}$

(c) XOR-join pattern:

$\begin{align*}
&a \rightarrow c, \\
&b \rightarrow c, \\
&c \rightarrow b
\end{align*}$

(d) AND-split pattern:

$\begin{align*}
&a \rightarrow b, \\
&b \rightarrow c, \\
&c \rightarrow b
\end{align*}$

(e) AND-join pattern:

$\begin{align*}
&a \rightarrow c, \\
&b \rightarrow c, \\
&c \rightarrow b
\end{align*}$
Algorithm

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

1. $T_L = \{ t \in T \mid \exists \sigma \in L \ t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists \sigma \in L \ t = \text{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists \sigma \in L \ t = \text{last}(\sigma) \}$,
4. $X_L = \{ (A,B) \mid A \subseteq T_L \land A \neq \emptyset \land B \subseteq T_L \land B \neq \emptyset \land \forall a \in A \forall b \in B \ a \rightarrow_L b \land \forall a_1,a_2 \in A \ a_1 \#_L a_2 \land \forall b_1,b_2 \in B \ b_1 \#_L b_2 \}$,
5. $Y_L = \{ (A,B) \in X_L \mid \forall (A',B') \in X_L \ A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_L = \{ p_{(A,B)} \mid (A,B) \in Y_L \} \cup \{ i_L,o_L \}$,
7. $F_L = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_L \land a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_L \land b \in B \} \cup \{ (i_L,t) \mid t \in T_I \} \cup \{ (t,o_L) \mid t \in T_O \}$, and
8. $\alpha(L) = (P_L,T_L,F_L)$. 
The $\alpha$–algorithm

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

1. $T_L = \{ t \in T \mid \exists \sigma \in L \ t \in \sigma \}$,
   Each activity in $L$ corresponds to a transition in $\alpha(L)$.

2. $T_I = \{ t \in T \mid \exists \sigma \in L \ t = \text{first}(\sigma) \}$
   Fix the set of start activities – that is, the first elements of each trace: $\langle t_1, \ldots, t_n \rangle$, $\ldots$, $\langle t'_1, \ldots, t'_m \rangle$

3. $T_O = \{ t \in T \mid \exists \sigma \in L \ t = \text{last}(\sigma) \}$
   Fix the set of end activities – that is, elements that appear last in a trace: $\langle t_1, \ldots, t_n \rangle$, $\ldots$, $\langle t'_1, \ldots, t'_m \rangle$
Step 4: Calculate pairs $(A, B)$
Step 5: Delete nonmaximal pairs $(A, B)$
Step 6: Determine places $p_{(A,B)}$ from pairs $(A, B)$
Find pairs (A, B) of sets of activities such that every element $a \in A$ and every element $b \in B$ are causally related (i.e., $a \rightarrow_L b$), all elements in A are independent ($a_1 \#_L a_2$), and all elements in B are independent ($b_1 \#_L b_2$).
Places as footprints

\[ A = \{ a_1, a_2, \ldots, a_m \} \quad B = \{ b_1, b_2, \ldots, b_n \} \]

\[ p_{(A,B)} \]

\begin{tabular}{|c|c|c|c|c|}
\hline
\( a_1 \) & \( a_2 \) & \ldots & \( a_m \) & \hline
\hline
\hline
\hline
\end{tabular}
The $\alpha$–algorithm (cont.)

5. $Y_L = \{ (A,B) \in X_L \mid \forall (A',B') \in X_L A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \}$

Delete from set $X_L$ all pairs $(A, B)$ that are not maximal!
The $\alpha$–algorithm (cont.)

6. $P_L = \{ p_{(A,B)} \mid (A,B) \in Y_L \} \cup \{i_L,o_L\}$,

Determine the place set: Each element $(A, B)$ of $Y_L$ is a place. To ensure the workflow structure, add a source place $i_L$ and a target place $o_L$. 

\[ A=\{a_1,a_2, \ldots, a_m\} \quad B=\{b_1,b_2, \ldots, b_n\} \]
The $\alpha$–algorithm (cont.)

7. $F_L = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_L \land a \in A \} 
\cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_L \land b \in B \} 
\cup \{ (i_L,t) \mid t \in T_I \} \cup \{ (t,o_L) \mid t \in T_O \}$

Determine the flow relation: Connect each place $p_{(A,B)}$ with each element $a$ of its set $A$ of source transitions and with each element of its set $B$ of target transitions. In addition, draw an arc from the source place $i_L$ to each start transition $t \in T_I$ and an arc from each end transition $t \in T_O$ to the sink place $o_L$.

8. $\alpha(L) = (P_L, T_L, F_L)$
\[ L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle] \]

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(#_{L_1})</td>
<td>(\rightarrow_{L_1})</td>
<td>(\rightarrow_{L_1})</td>
<td>(#_{L_1})</td>
<td>(\rightarrow_{L_1})</td>
</tr>
<tr>
<td>(b)</td>
<td>(\leftarrow_{L_1})</td>
<td>(#_{L_1})</td>
<td>(\parallel_{L_1})</td>
<td>(\rightarrow_{L_1})</td>
<td>(#_{L_1})</td>
</tr>
<tr>
<td>(c)</td>
<td>(\leftarrow_{L_1})</td>
<td>(\parallel_{L_1})</td>
<td>(#_{L_1})</td>
<td>(\rightarrow_{L_1})</td>
<td>(#_{L_1})</td>
</tr>
<tr>
<td>(d)</td>
<td>(#_{L_1})</td>
<td>(\leftarrow_{L_1})</td>
<td>(\leftarrow_{L_1})</td>
<td>(#_{L_1})</td>
<td>(\leftarrow_{L_1})</td>
</tr>
<tr>
<td>(e)</td>
<td>(\leftarrow_{L_1})</td>
<td>(#_{L_1})</td>
<td>(#_{L_1})</td>
<td>(\rightarrow_{L_1})</td>
<td>(#_{L_1})</td>
</tr>
</tbody>
</table>

\[
X_{L_1} = \{(\{a\}, \{b\}), (\{a\}, \{c\}), (\{a\}, \{e\}), (\{a\}, \{b, e\}), (\{a\}, \{c, e\}),
(\{b\}, \{d\}), (\{c\}, \{d\}), (\{e\}, \{d\}), (\{b, e\}, \{d\}), (\{c, e\}, \{d\})\}
\]

\[
Y_{L_1} = \{(\{a\}, \{b, e\}), (\{a\}, \{c, e\}), (\{b, e\}, \{d\}), (\{c, e\}, \{d\})\}
\]
Another event log $L_3$

$L_3 = [\langle a, b, c, d, e, f, b, d, c, e, g \rangle,$
\[\langle a, b, d, c, e, g \rangle^2,
\langle a, b, c, d, e, f, b, c, d, e, f, b, d, c, e, g \rangle]\]

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>#</td>
<td>$\rightarrow$</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>$b$</td>
<td>$\leftarrow$</td>
<td>#</td>
<td>$\rightarrow$</td>
<td>$\rightarrow$</td>
<td>#</td>
<td>$\leftarrow$</td>
<td>#</td>
</tr>
<tr>
<td>$c$</td>
<td>#</td>
<td>$\leftarrow$</td>
<td>#</td>
<td>#</td>
<td>$\parallel$</td>
<td>$\rightarrow$</td>
<td>#</td>
</tr>
<tr>
<td>$d$</td>
<td>#</td>
<td>$\leftarrow$</td>
<td>$\parallel$</td>
<td>#</td>
<td>$\rightarrow$</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>$e$</td>
<td>#</td>
<td>#</td>
<td>$\leftarrow$</td>
<td>$\leftarrow$</td>
<td>#</td>
<td>$\rightarrow$</td>
<td>$\rightarrow$</td>
</tr>
<tr>
<td>$f$</td>
<td>#</td>
<td>$\rightarrow$</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>$\leftarrow$</td>
<td>#</td>
</tr>
<tr>
<td>$g$</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>$\leftarrow$</td>
<td>#</td>
</tr>
</tbody>
</table>
Model for $L_3$

$L_3 = \langle a, b, c, d, e, f, b, d, c, e, g \rangle,$
\[ \langle a, b, d, c, e, g \rangle^2, \]
\[ \langle a, b, c, d, e, f, b, c, d, e, f, b, d, c, e, g \rangle \]
\[ L_3 = [\langle a,b,c,d,e,f,b,d,c,e,g \rangle, \langle a,b,d,c,e,g \rangle^2, \langle a,b,c,d,e,f,b,c,d,e,f,b,d,c,e,g \rangle] \]
Mine for a Petri Net using Alpha-algorithm
B.F. van Dongen (b.f.v.dongen@tue.nl)
AlphaMiner
Another event log $L_4$

$L_4 = [\langle a, c, d\rangle^{45}, \langle b, c, d\rangle^{42}, \langle a, c, e\rangle^{38}, \langle b, c, e\rangle^{22}]$
Event log $L_5$

\[ L_5 = [\langle a, b, e, f \rangle^2, \langle a, b, e, c, d, b, f \rangle^3, \langle a, b, c, e, d, b, f \rangle^2, \langle a, b, c, d, e, b, f \rangle^4, \langle a, e, b, c, d, b, f \rangle^3] \]

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td>#</td>
<td></td>
<td>#</td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>$b$</td>
<td>←</td>
<td></td>
<td>#</td>
<td>←</td>
<td></td>
<td>→</td>
</tr>
<tr>
<td>$c$</td>
<td>#</td>
<td>←</td>
<td></td>
<td>#</td>
<td>←</td>
<td></td>
</tr>
<tr>
<td>$d$</td>
<td>#</td>
<td>→</td>
<td>←</td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>$e$</td>
<td>←</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>$f$</td>
<td>#</td>
<td>←</td>
<td></td>
<td>#</td>
<td></td>
<td>#</td>
</tr>
</tbody>
</table>
\[ T_L = \{a, b, c, d, e, f\} \]
\[ T_I = \{a\} \]
\[ T_I = \{f\} \]
\[ X_L = \{ (\{a\}, \{b\}) , (\{a\}, \{e\}) , (\{b\}, \{c\}) , (\{b\}, \{f\}) , (\{c\}, \{d\}) , (\{d\}, \{b\}) , (\{e\}, \{f\}) , (\{a,d\}, \{b\}) , (\{b\}, \{c,f\}) \} \]
\[ Y_L = \{ (\{a\}, \{e\}) , (\{c\}, \{d\}) , (\{e\}, \{f\}) , (\{a,d\}, \{b\}) , (\{b\}, \{c,f\}) \} \]
\[ P_L = \{ p(\{a\}, \{e\}) , p(\{c\}, \{d\}) , p(\{e\}, \{f\}) , p(\{a,d\}, \{b\}) , p(\{b\}, \{c,f\}) , iL , oL \} \]
\[ F_L = \{ (a, p(\{a\}, \{e\})) , (p(\{a\}, \{e\}), e) , (c, p(\{c\}, \{d\})) , (p(\{c\}, \{d\}), d) , (e, p(\{e\}, \{f\})) , (p(\{e\}, \{f\}), f) , (a, p(\{a,d\}, \{b\})) , (d, p(\{a,d\}, \{b\})) , (p(\{a,d\}, \{b\}), b) , (b, p(\{b\}, \{c,f\})) , (p(\{b\}, \{c,f\}), c) , (p(\{b\}, \{c,f\}), f) , (iL, a), (f, oL) \} \]
\[ \alpha(L) = (P_L, T_L, F_L) \]
Discovered model

\[ X_L = \{ (\{a\}, \{b\}), (\{a\}, \{e\}), (\{b\}, \{c\}), (\{b\}, \{f\}), (\{c\}, \{d\}), (\{d\}, \{b\}), (\{e\}, \{f\}), (\{a,d\}, \{b\}), (\{b\}, \{c,f\}) \} \]

\[ Y_L = \{ (\{a\}, \{e\}), (\{c\}, \{d\}), (\{e\}, \{f\}), (\{a,d\}, \{b\}), (\{b\}, \{c,f\}) \} \]
\( p(\{a\}) \quad p(\{e\}) \quad p(\{b\}, \{c, f\}) \quad p(\{a, d\}, \{b\}) \quad p(\{c\}, \{d\}) \)
ProM covers the whole process mining spectrum (alpha miner is one of 600+ plugins)

http://www.promtools.org/
http://www.processmining.org/

ProM 6.3

Processes are an integral part of today’s world, driving services and internal functionalities in businesses, governmental bodies, and many other domains.
limitations of $\alpha$ algorithm
Limitation of \( \alpha \) algorithm: Implicit places

\[
L_6 = [\langle a, c, e, g \rangle^2, \langle a, e, c, g \rangle^3, \langle b, d, f, g \rangle^2, \langle b, f, d, g \rangle^4]\]
Limitation of $\alpha$ algorithm: Loops of length 1

$$L_7 = [\langle a, c \rangle^2, \langle a, b, c \rangle^3, \langle a, b, b, c \rangle^2, \langle a, b, b, b, b, b, c \rangle^1]$$

Desired model: A Petri net with a single token initially on the place $a$, transitions $a \rightarrow b$ and $a \rightarrow c$, and output places $b$, $b$, and $c$. The net also includes a loop $b \parallel b$. Additional transitions $a > b$, $a > c$, $b > b$, and $b > c$ are shown, with additional tokens on places $a$, $c$, and $b$.
Limitation of α algorithm: Loops of length 2

\[ L_8 = [\langle a, b, d \rangle^3, \langle a, b, c, b, d \rangle^2, \langle a, b, c, b, c, b, d \rangle] \]
Limitation of $\alpha$ algorithm: Non-local dependencies

$L_9 = [\langle a, c, d \rangle^{45}, \langle b, c, e \rangle^{42}]$

Not discovered!

$L_4 = [\langle a, c, d \rangle^{45}, \langle b, c, d \rangle^{42}, \langle a, c, e \rangle^{38}, \langle b, c, e \rangle^{22}]$
Difficult constructs for $\alpha$ algorithm

[Diagram showing complex constructs labeled a, b, c with crossing arrows indicating difficulty]
evaluation and bias
Rediscovering process models

The rediscovery problem: Is the discovered model $N'$ equivalent to the original model $N$?
Equivalence: trace equivalence, bisimilarity, and branching bisimilarity

Three trace equivalent transition systems: $TS_1$ and $TS_2$ are not bisimilar, but $TS_2$ and $TS_3$ are bisimilar.
Branching bisimilarity defined for YAWL

TS₁ and TS₂ are not branching bisimilar (although trace equivalent).
Challenge: finding the right representational bias

$L_{10} = [\langle a, a \rangle^{55}]$

There is no WF-net with unique visible labels that exhibits this behavior.
Another example

\[ L_{11} = [\langle a, b, c \rangle^{20}, \langle a, c \rangle^{30}] \]

There is no WF-net with unique visible labels that exhibits this behavior.
OR-split/join model

- Create an event log containing all possible full firing sequences.
- Take this event log and apply the alpha algorithm?
- Comment on differences and explain reason.
Log loaded
Model discovered
Region-based miner (with label splitting)
Challenge: noise and incompleteness

- To discover a suitable process model it is assumed that the event log contains a representative sample of behavior.
- Two related phenomena:
  - **Noise**: the event log contains rare and infrequent behavior not representative for the typical behavior of the process.
  - **Incompleteness**: the event log contains too few events to be able to discover some of the underlying control-flow structures.
More on incompleteness

To illustrate the relevance of completeness, consider a process consisting of 10 activities that can be executed in parallel and a corresponding log that contains information about 10,000 cases. The total number of possible interleavings in the model with 10 concurrent activities is $10! = 3,628,800$. Hence, it is impossible that each interleaving is present in the log as there are fewer cases (10,000) than potential traces (3,628,800). Even if there are 3,628,800 cases in the log, it is extremely unlikely that all possible variations are present. To motivate this consider the following analogy. In a group of 365 people it is very unlikely that everyone has a different birthdate. The probability is $365!/365^{365} \approx 1.454955 \times 10^{-157} \approx 0$, i.e., incredibly small. The number of atoms in the universe is often estimated to be approximately $10^{79}$ [129].

See also chapter 3 (cross-validation, precision, recall, etc.)
Challenge: Balancing Between Underfitting and Overfitting
What is the best model?

©Wil van der Aalst TU/e (use only with permission & acknowledgements)
What is the best model?

ACD 99
ACE 88
BCE 85
BCD 78
What is the best model?

ACD 99
ACE 2
BCE 85
BCD 3
Balance four forces

- **Fitness**: ability to explain observed behavior
- **Precision**: avoiding overfitting
- **Generalization**: avoiding underfitting
- **Simplicity**: Occam’s Razor

Diagram with arrows labeled:
- Lift
- Thrust
- Drag
- Gravity
Example: one log four models

- fitness
- simplicity
- generalization
- precision

“able to replay event log”
“Occam’s razor”
“not overfitting the log”
“not underfitting the log”

N1: fitness = +, precision = +, generalization = +, simplicity = +
N2: fitness = -, precision = +, generalization = -, simplicity = +
N3: fitness = +, precision = +, generalization = +, simplicity = +
N4: fitness = +, precision = +, generalization = -, simplicity = -

<table>
<thead>
<tr>
<th>#</th>
<th>trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>455</td>
<td>acdeh</td>
</tr>
<tr>
<td>191</td>
<td>abdeg</td>
</tr>
<tr>
<td>177</td>
<td>acdeh</td>
</tr>
<tr>
<td>144</td>
<td>abdeh</td>
</tr>
<tr>
<td>111</td>
<td>acdeg</td>
</tr>
<tr>
<td>82</td>
<td>acdeh</td>
</tr>
<tr>
<td>56</td>
<td>abbeh</td>
</tr>
<tr>
<td>47</td>
<td>acdefdbeh</td>
</tr>
<tr>
<td>38</td>
<td>abbeg</td>
</tr>
<tr>
<td>33</td>
<td>acdefbdeh</td>
</tr>
<tr>
<td>14</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>11</td>
<td>acdefdbeg</td>
</tr>
<tr>
<td>9</td>
<td>acdefcdeh</td>
</tr>
<tr>
<td>8</td>
<td>acdefdbheh</td>
</tr>
<tr>
<td>5</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>3</td>
<td>acdefbdefbdeg</td>
</tr>
<tr>
<td>2</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>2</td>
<td>acdefbdefbdeg</td>
</tr>
<tr>
<td>1</td>
<td>acdefbdefbdeh</td>
</tr>
<tr>
<td>1</td>
<td>acdefbdefcdefdbeg</td>
</tr>
<tr>
<td>1391</td>
<td></td>
</tr>
</tbody>
</table>
Model $N_1$

$N_1 : \text{fitness} = +, \text{precision} = +, \text{generalization} = +, \text{simplicity} = +$

<table>
<thead>
<tr>
<th>#</th>
<th>trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>455</td>
<td>acdeh</td>
</tr>
<tr>
<td>191</td>
<td>abdeg</td>
</tr>
<tr>
<td>177</td>
<td>adceh</td>
</tr>
<tr>
<td>144</td>
<td>abdeh</td>
</tr>
<tr>
<td>111</td>
<td>acdeg</td>
</tr>
<tr>
<td>82</td>
<td>adceg</td>
</tr>
<tr>
<td>56</td>
<td>adbeh</td>
</tr>
<tr>
<td>47</td>
<td>acdefdbeh</td>
</tr>
<tr>
<td>38</td>
<td>adbeg</td>
</tr>
<tr>
<td>33</td>
<td>acdefbdeh</td>
</tr>
<tr>
<td>14</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>11</td>
<td>acdefdbeg</td>
</tr>
<tr>
<td>9</td>
<td>adcefcdeh</td>
</tr>
<tr>
<td>8</td>
<td>acdefdbeh</td>
</tr>
<tr>
<td>5</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>3</td>
<td>acdefbdefdbeg</td>
</tr>
<tr>
<td>2</td>
<td>adcefdbeg</td>
</tr>
<tr>
<td>2</td>
<td>adcefbdefbdeg</td>
</tr>
<tr>
<td>1</td>
<td>adcefdbebdhe</td>
</tr>
<tr>
<td>1</td>
<td>adbfbdedefbg</td>
</tr>
<tr>
<td>1</td>
<td>adcefdbebcdebd</td>
</tr>
<tr>
<td>1391</td>
<td></td>
</tr>
</tbody>
</table>
Model $N_2$

$N_2 : \text{fitness} = -, \text{precision} = +, \text{generalization} = -, \text{simplicity} = +$

<table>
<thead>
<tr>
<th>#</th>
<th>trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>455</td>
<td>acdeh</td>
</tr>
<tr>
<td>191</td>
<td>abdeg</td>
</tr>
<tr>
<td>177</td>
<td>adceh</td>
</tr>
<tr>
<td>144</td>
<td>abdeh</td>
</tr>
<tr>
<td>111</td>
<td>acdeg</td>
</tr>
<tr>
<td>82</td>
<td>adceg</td>
</tr>
<tr>
<td>56</td>
<td>adbeh</td>
</tr>
<tr>
<td>47</td>
<td>acdefdbeh</td>
</tr>
<tr>
<td>38</td>
<td>adbeg</td>
</tr>
<tr>
<td>33</td>
<td>acdefbdeh</td>
</tr>
<tr>
<td>14</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>11</td>
<td>acdefdbeg</td>
</tr>
<tr>
<td>9</td>
<td>acdefcdeh</td>
</tr>
<tr>
<td>8</td>
<td>acdefdbeh</td>
</tr>
<tr>
<td>5</td>
<td>acdefbdeg</td>
</tr>
<tr>
<td>3</td>
<td>acdefbdefdbeg</td>
</tr>
<tr>
<td>2</td>
<td>acdefdbeg</td>
</tr>
<tr>
<td>2</td>
<td>acdefbdefbdeg</td>
</tr>
<tr>
<td>1</td>
<td>acdefdbefdbdeh</td>
</tr>
<tr>
<td>1</td>
<td>adbefbdefdbeg</td>
</tr>
<tr>
<td>1</td>
<td>acdefdbefcdefdbeg</td>
</tr>
</tbody>
</table>

non-fitting
Model $N_3$

$N_3: fitness = +, precision = -, generalization = +, simplicity = +$

underfitting
Model $N_4$

$N_4 : fitness = +, precision = +, generalization = -, simplicity = -$
Why is process mining such a difficult problem?

- There are **no negative examples** (i.e., a log shows what has happened but does not show what could not happen).
- Due to concurrency, loops, and choices the search space has a complex structure and the log typically contains only a **fraction** of all possible behaviors.
- There is **no clear relation** between the size of a model and its behavior (i.e., a smaller model may generate more or less behavior although classical analysis and evaluation methods typically assume some monotonicity property).
Concurrency: How many traces?

- Consider a process model with a start activity and end activity and in the middle $k$ parallel activities.
  - How many traces are possible?
  - Does the alpha algorithm need to see all of these to rediscover the original model?
- Suppose now that the $k$ parallel activities are optional.
  - How many traces are possible?
  - Is the alpha algorithm able to discover such constructs?
Concurrency: How many traces?

• Consider a process model with a start activity and end activity and in the middle $k$ parallel activities.
  - How many traces are possible?
  - Does the alpha algorithm need to see all of these to rediscover the original model?

• $k! = k(k-1)(k-2)\ldots1$ traces are possible:
  - $5! = 120$
  - $10! = 3628800$
  - $20! = 2432902008176640000$

• It is easy to construct a log consisting of $k(k-1)$ traces that is complete in terms of directly follows relation:
  - $5 \times 4 = 20$
  - $10 \times 9 = 90$
  - $20 \times 19 = 380$
Concurrency: How many traces?

- Suppose now that the $k$ parallel activities are optional.
  - How many traces are possible?
  - Is the alpha algorithm able to discover such constructs?

The alpha algorithm is unable to discover such constructs: it will discover $k$ parallel non-optional activities.
conclusion and outlook
• Alpha algorithm is rather simplistic: but good to understand the main principles.
• We will see more advanced algorithms.
• Process mining is more than discovery!