1. Introduction

2. Basics

3. Unsupervised Methods

4. Supervised Methods

5. Advanced Topics
   5.1 Process Mining
   5.2 Outlook
Motivation

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Location</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>14.01.2018 10:32</td>
<td>MUNICH</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>14.01.2018 11:40</td>
<td>MUNICH</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>14.01.2018 15:17</td>
<td>MUNICH</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>14.01.2018 22:12</td>
<td>LEIPZIG</td>
<td>1</td>
</tr>
</tbody>
</table>

Process Model

- **supports/controls**
- **configures**
- **log events**
- **discovery**
- **conformance**
- **enhancement**
Notions

- Process: System of actions, movements (e.g. sign document, customer call, financial transaction, delivery of goods)
- Different instances/cases should follow a common process description
- Each case contains actions as events (their sequence is called *trace*)
- An event is represented by at least
  - A case identifier
  - An activity label
  - A timestamp
- but may also comprise additional (meta-)information (e.g. involved (work) resources)
Petri Nets as Process Model

Start → a → b → d → End

Places

Transitions

c
Main Tasks

1. *Process Discovery:*
   Mine multiple sequences of actions to derive a workflow pattern

2. *Conformance Checking:*
   Use previously mined model to judge the validity of a new case

3. *Process Enhancement:*
   Evolve models with new data, find deviations
## Process Discovery

### Input

<table>
<thead>
<tr>
<th>#</th>
<th>trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048</td>
<td>ace</td>
</tr>
<tr>
<td>1234</td>
<td>acdce</td>
</tr>
<tr>
<td>404</td>
<td>acdcdce</td>
</tr>
<tr>
<td>120</td>
<td>acdcdcdce</td>
</tr>
<tr>
<td>42</td>
<td>ab</td>
</tr>
<tr>
<td>5</td>
<td>acdb</td>
</tr>
</tbody>
</table>

### Quality Measures

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness</td>
<td>ability to replay the log</td>
</tr>
<tr>
<td>Simplicity</td>
<td>simplified as much as possible</td>
</tr>
<tr>
<td>Generalization</td>
<td>no underfitting of log</td>
</tr>
<tr>
<td>Precision</td>
<td>no overfitting of log</td>
</tr>
</tbody>
</table>

### Output

![Process Flow Diagram]

- **a**
- **b**
- **c**
- **d**
- **e**

Advanced Topics  Process Mining  February 6, 2019  491
Example Discovery Algorithm: $\alpha$-Miner\textsuperscript{22}

1. Scan the log for all activities
2. For each pair of activities and $a$, we define the relations
   - $a > b$ if for some case $a$ is immediately followed by $b$ (direct succession)
   - $a \parallel b$ if $a > b$ and $b > a$ (parallelism)
   - $a \rightarrow b$ if $a > b$ and not $b > a$ (causality)
   - $a \# b$ if not $a > b$ and not $b > a$
3. All activities, having only $\#$ or $\rightarrow$ in their row are starting activities. They are collected in $T_{in}$.
4. Analogously, $\#$ or $\leftarrow$ determine $T_{out}$.

Example: $\{abcd, acbd, acd\}$

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

$T_{in} = \{a\}$, $T_{out} = \{d\}$

\textsuperscript{22}van der Aalst, Weijters, Maruster (2003). "Workflow Mining: Discovering process models from event logs", IEEE Transactions on Knowledge and Data Engineering, vol 16
Example Discovery Algorithm: $\alpha$-Miner

1. Prepare a Petri net. The set of transitions is equal to activities

2. A starting place is created and connected to each node in $T_{in}$

3. Also, a final place is created and each node in $T_{out}$ is connected to it

4. Determine all pairs of sets $A$ and $B$, such that
   - $\forall a_1, a_2 \in A : a_1 \# a_2$
   - $\forall b_1, b_2 \in B : b_1 \# b_2$
   - $\forall a \in A, b \in B : a \rightarrow b$

5. A place is added in between $A$ and $B$ and connected accordingly

Diagram:

2. ![Diagram 2]

3. ![Diagram 3]

4. $A = \{a\}, B = \{b, c\}$

5. ![Diagram 5]
Conformance Checking

Use previously mined model to judge the validity of a new case (similar to binary classification: valid vs. invalid)

**Input**
- Model
- Trace

**Aims**
- Model reasoning
- Auditing
- Security (fraud detection)
Example Conformance Checking Algorithm: Token-Replay

Replay the event in the model. Count:

- the number of produced tokens \( p \)
- the number of consumed tokens \( c \)
- the number of missing tokens \( m \)
- the number of remaining tokens \( r \)

Output a *fitness* value

\[
f = \frac{1}{2} \left( 1 - \frac{m}{c} \right) + \frac{1}{2} \left( 1 - \frac{r}{p} \right)
\]

The fitness value ranges between 0 and 1, where 1 is a perfect match.