Positioning Process Mining

Process mining
(data mining, machine learning, business intelligence)

Process model analysis
(simulation, verification, etc.)

Data-oriented analysis
(data mining, machine learning, business intelligence)

Performance-oriented questions, problems and solutions

Compliance-oriented questions, problems and solutions
Overview

"world"  business processes  people  machines  components  organizations

models  analyzes

software system

supports/controls

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

specifies  configures  implements  analyzes

(event logs)

(conformance)

(discovery)

enhancement

(models)

(analyzes)

(business processes)

(discovery)
Hundreds of plug-ins available covering the whole process mining spectrum

ProM
process mining workbench

open-source (L-GPL)

Download from: www.processmining.org
Commercial Alternatives

- **Disco (Fluxicon)**
- **Perceptive Process Mining** (before Futura Reflect and BPM|one)
- **ARIS Process Performance Manager**
- **QPR ProcessAnalyzer**
- **Interstage Process Discovery (Fujitsu)**
- **Discovery Analyst (StereoLOGIC)**
- **XMAssay (XMPro)**
- ...
Process Discovery
Process Discovery (small selection)

- automata-based learning
- heuristic mining
- genetic mining
- stochastic task graphs
- fuzzy mining
- mining block structures
  - $\alpha$ algorithm
  - $\alpha#$ algorithm
  - $\alpha++$ algorithm
- distributed genetic mining
- language-based regions
- state-based regions
- LTL mining
- neural networks
- hidden Markov models
- conformal process graph
- partial-order based mining
- ILP mining
Typical Representational Bias

- (Labeled) Petri Nets, WF-nets, etc.
- Subsets of
  - BPMN diagrams,
  - UML Activity Diagrams,
  - Event-Driven Process Chains (EPCs),
  - YAWL,
  - Statecharts?
  - etc.
- Transition Systems
- (Hidden) Markov Models
- …
Alternative Representational Bias

1. **C-nets** (XOR/AND/OR-split/join graphs; more likely to be sound due to declarative semantics).

2. **Declare models** (constraint based, grounded in LTL; anything is possible unless forbidden)

3. **Process Trees** (similar to subsets of various process algebras; sound by structure)
Petri net/Region-based View
Petri net view:
Just discover the places …

Adding a place limits behavior:
• overfitting ≈ adding too many places
• underfitting ≈ adding too few places
The Petri net below can replay any trace over \{a,b,c,d,e\}

\[
L_1 = \left[ \langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle \right]
\]
Place limits behavior

1. abcd
2. aed
3. acbd
4. abcd
5. abcd
6. acbd

1. abcd
2. bcd
3. aed
4. abed
5. cbd
6. acbd
Example: Process Discovery Using State-Based Regions
Example of State-Based Region

- enter: b, e
- leave: d
- do-not-cross: a, c
A place is **feasible** if it can be added without disabling any of the traces in the event log.

for any $\sigma \in L, k \in \{1, \ldots, |\sigma|\}, \sigma_1 = h d^{k-1}(\sigma), a = \sigma(k), \sigma_2 = h d^k(\sigma) = \sigma_1 \oplus a$:

$$c + \sum_{t \in X} \partial_{\text{multiset}}(\sigma_1)(t) - \sum_{t \in Y} \partial_{\text{multiset}}(\sigma_2)(t) \geq 0.$$
Example of Language-Based Regions

1. accd
2. bd
3. bce
4. ace
5. acd
6. bcce
7. ade

\[
\begin{align*}
&\text{accd} : 0 + 0 - 0 \geq 0 \\
&a \downarrow \text{ccd} : 0 + 1 - 1 \geq 0 \\
&ac \downarrow \text{cd} : 0 + 2 - 2 \geq 0 \\
&\text{acc} \downarrow d : 0 + 3 - 3 \geq 0 \\
&\downarrow \text{ade} : 0 + 0 - 0 \geq 0 \\
&a \downarrow \text{de} : 0 + 1 - 1 \geq 0 \\
&ad \downarrow e : 0 + 1 - 2 < 0
\end{align*}
\]
Creating a Transition System
Learning a Transition System

- **past**, **future**, **past+future**
- sequence, multiset, set abstraction
- limited horizon to abstract further
- filtering e.g. based on transaction type, names, etc.
- labels based on activity name or other features
Sometimes called the "prefix automaton"

\[ L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle] \]
Future Without Abstraction

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

\[ L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle] \]
Only Last Event Matters For State

\[ L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle] \]
Using ProM
$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$
Run Plugin
Select (scroll or by name)
Start Plugin "Mine Transition System"
Start Window

past

future

attributes
Abstraction

list, multiset, or set

all, or only last k events
Which events to filter?
Which labels need to be visible?
Any repair actions?

- Remove self loops
- Improve diamond structure
- Merge states with identical inflow
Check configuration
Resulting transition system
Convert transition system to Petri net
Resulting Petri net
$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$
State-Based Regions
What is a (state-based) region?

- a = enter
- b = enter
- c = exit
- d = exit
- e = do not cross
- f = do not cross
Starting point: A Transition System

• We assume that there is only one initial state (otherwise preprocessing needed).
• It is convenient to also have just one final state that can always be reached (not strictly necessary)
• All states need to be reachable!
Definition

- A region $r$ is a set of states, such that for all transitions $(s_0, e, s_0')$, $(s_1, e, s_1')$ in the transition system holds that:
  1) $s_0 \in r$ and $s_0' \notin r$ implies that $s_1 \in r$ and $s_1' \notin r$
  2) $s_0 \notin r$ and $s_0' \in r$ implies that $s_1 \notin r$ and $s_1' \in r$

- In words: A region is a set of states, such that, if a transition exits the region, then all equally labeled transitions exit the region, and if a transition enters the region, then all equally labeled transitions enter the region. All events not entering or exiting the region do not cross the region.
Example of a region

- a enters
- b exits
- c does not cross
- d does not cross
- e does not cross
Example of a region

- a does not cross
- b enters
- c does not cross
- d does not cross
- e exits
Example of a region

• a exits
• b does not cross
• c does not cross
• d does not cross
• e does not cross

Places corresponding to regions containing the initial state are initially marked.
Example of a region

- a enters
- b does not cross
- c does not cross
- d exits
- e does not cross
• a enters
• b does not cross and exits
• c does not cross and exits
• d does not cross and exits
• e does not cross
• a does not cross
• b does not cross and exits
• c does not cross and exits
• d does not cross and exits
• e does not cross
Multiple regions

Etc.
Selectively chosen regions ...
Regions – Region Properties

- Let $S$ be the set of all states of a transition system.
- **Trivial Regions:** Both $S$ and $\emptyset$ are called the trivial regions,
- **Complements:** If $r$ is a region, then $S \setminus r$ is a region,
- **Pre-/Post-regions:** If event $e$ exits (enters) a region $r$, then $r$ is a pre- (post-)region of $e$,
- **Minimal regions:** If $r_0$ and $r_1$ are regions, and $r_0$ is a subset of $r_1$, then $r_1 \setminus r_0$ is a region.

The latter implies the existence of (non-trivial) *minimal regions*. 
Trivial regions

a, b, c, d, e do not cross
Complement:
If $r$ is a region, then $S \setminus r$ is a region

"exits" and "enters" are swapped
If $r_0$ and $r_1$ are regions, and $r_0$ is a subset of $r_1$, then $r_1 \setminus r_0$ is a region.
Not minimal yet …
Pre and post regions

- If event $e$ enters a region $r$, then $r$ is a post-region of $e$.
  - $r$ is post-region of $a$
  - $r$ is post-region of $b$

- If event $e$ exits a region $r$, then $r$ is a pre-region of $e$.
  - $r$ is pre-region of $c$
  - $r$ is pre-region of $d$

• $\text{pre}(e)$ is the set of all (minimal) pre-regions or $e$. 
• $\text{pre}(e)$ is the set of all (minimal) pre-regions or $e$. 
• Both are sets of sets!
Basic algorithm to construct a Petri net

- For each event in the transition system, a *transition* is generated in the Petri net.
- Compute the *minimal non-trivial regions*.
- For each minimal non-trivial in the transition system, a *place* is generated in the Petri net.
- Add corresponding *arcs* (post-regions are output places and pre-regions are input places).
- A *token* is added to each place that corresponds to a region containing the initial state.

The resulting Petri net is called the *minimal saturated net*. 
Load Petri net with 10 parallel activities
Construct reachability graph
Reachability graph (\(1+2^{10}+1 = 1026\) states)
Apply state-based regions to fold state space
Discovered Petri net

- Petri net is rediscovered!
- Odd example, normally the transition system is constructed from an event log.
26 transitions, 28 places, 1 token
It is not that simple…
(but all problems can be repaired)
Consider an event log containing just \(<a,a>\) traces

prefix automaton

Only trivial regions: \(\emptyset\) and \(\{s1,s2,s3\}\)

Petri net

Also allows for:
- a
- aaaa
- aaaaaaaaaa
Consider an event log containing traces <a,c>, <a,b,c>, <a,b,b,c>, <a,b,b,b,c>, ...

transition system able to generate log

Regions:
- \{s1\} (a exits, b and c do not cross)
- \{s2\} (a enters, b does not cross, c exits)
- \{s3\} (a and b does not cross, c enters)

Petri net

Also allows for:
- bbac
- acbbbb
- babcb
Consider an event log containing traces \(<a,b>, <b>\)

Regions:
- \(\{s1,s2\}\) (a does not cross, b exits)
- \(\{s3,s4\}\) (a does not cross, b enters)
- \(\{s1,s3\}\) (a exits and b does not cross)
- \(\{s2,s4\}\) (a enters and b does not cross)
Petri net

\{s1,s3\} \rightarrow a \rightarrow \{s1,s2\}

{\{s2,s4\} \rightarrow b \rightarrow \{s3,s4\}}

Also allows for trace <b,a>!
All underfitting, but feasible

\begin{align*}
\text{s1} & \xrightarrow{a} \text{s2} & \text{s2} & \xrightarrow{a} \text{s3} \\
\text{s1} & \xrightarrow{a} \text{s2} & \xrightarrow{c} \text{s3} \\
\text{s2} & \xrightarrow{a} \text{s3} & \xrightarrow{b} \text{s4} \\
\end{align*}
Using ProM
(uses label splitting to solve problem)
Using ProM
(addresses self-loop problem)
Using ProM
(uses label splitting to solve problem)

two "b" transitions
At the other end of the spectrum ...
A completely different example of a process discovery technique: Genetic Mining

- requires a lot of computing power, but can be distributed easily,
- can deal with noise, infrequent behavior, duplicate tasks, invisible tasks,
- allows for incremental improvement and combinations with other approaches (heuristics post-optimization, etc.).
Genetic process mining: Overview

- Event log
- Create initial population
- Compute fitness
- Elitism
- Tournament
- Select best individual
- Parents
- "Dead" individuals
- Next generation
- Mutation
- Children
- Crossover
- Termination
Example: crossover
Example: mutation
Link between process discovery and conformance checking

- Event log
- Compute fitness
- Create initial population
- Next generation
- Elitism
- Parents
- Tournament
- Crossover
- Children
- Mutation
- "Dead" individuals
- Select best individual
- Termination

Event log is used to compute fitness, which is used to create an initial population. The population evolves through elitism, parents, crossover, and mutation, resulting in a new generation. The process continues until termination, selecting the best individual.
How good is my model?
Four Competing Quality Criteria

- **fitness**: “able to replay event log”
- **simplicity**: “Occam’s razor”
- **generalization**: “not overfitting the log”
- **precision**: “not underfitting the log”

These criteria are essential for effective process discovery.
Example: one log four models

“able to replay event log”
“Occam’s razor”
fitness

simplicity
generalization
“not overfitting the log”
“not underfitting the log”
precision

process discovery

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

“able to replay event log”
“Occam’s razor”
fitness

simplicity
generalization
“not overfitting the log”
“not underfitting the log”
precision

process discovery

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

“able to replay event log”
“Occam’s razor”
fitness

simplicity
generalization
“not overfitting the log”
“not underfitting the log”
precision

process discovery

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

“able to replay event log”
“Occam’s razor”
fitness

simplicity
generalization
“not overfitting the log”
“not underfitting the log”
precision

process discovery

N1: fitness = +, precision = +, generalization = +, simplicity = +
N2: fitness = -, precision = +, generalization = -, simplicity = +
N3: fitness = +, precision = -, generalization = +, simplicity = +
N4: fitness = +, precision = +, generalization = -, simplicity = -
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>adbgh</td>
</tr>
<tr>
<td>33</td>
<td>acdefbdheh</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>acdefdbdefdbeg</td>
</tr>
<tr>
<td>2</td>
<td>acdefdbeg</td>
</tr>
<tr>
<td>1</td>
<td>adcefdbebfbddeh</td>
</tr>
<tr>
<td>1</td>
<td>adbebdedefdbeg</td>
</tr>
<tr>
<td>1</td>
<td>adbebdedefdbeg</td>
</tr>
<tr>
<td>1</td>
<td>adcefdbebcdebddeh</td>
</tr>
</tbody>
</table>
Model $N_2$

$N_2: \text{fitness} = -$ \text{, simplicity} = +$
Model $N_3$

<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>adcefcdhe</td>
</tr>
<tr>
<td>8</td>
<td>adcefdbeh</td>
</tr>
<tr>
<td>5</td>
<td>adcefbdeg</td>
</tr>
<tr>
<td>3</td>
<td>acdefbdedefbeg</td>
</tr>
<tr>
<td>2</td>
<td>adcefdbeg</td>
</tr>
<tr>
<td>2</td>
<td>acdefbdedefbeg</td>
</tr>
<tr>
<td>1</td>
<td>adcefbdefbdeh</td>
</tr>
<tr>
<td>1</td>
<td>adcefbdefdbeg</td>
</tr>
<tr>
<td>1</td>
<td>adcefdbebefcdefdbeg</td>
</tr>
</tbody>
</table>

- B: examine thoroughly
- D: check ticket
- G: pay

### Notes
- N3: fitness = +, precision = -, generalization = +, simplicity = +
Model $N_4$

$N_4: \text{fitness} = +, \text{precision} = +, \text{generalization} = -, \text{simplicity} = -$
Conclusion

Still many challenging and highly relevant open problems in process discovery!