Process Discovery: An Introduction

Based on an event log a process model is constructed capturing the behavior in the log

Barbara Re

Process Mining

Problem Statement

Focusing on discovery the control-flow perspective

Definition (General process discovery problem)

Let $\mathcal{L}$ be an event log. A process discovery algorithm is a function that maps $\mathcal{L}$ onto a process model such that the model is representative for the behavior seen in the event log. The challenge is to find such an algorithm.

This definition does not specify what kind of process model should be generated, e.g., a BPMN, EPC, YAWL, or Petri net model
To make things more concrete:

- we define the target to be a Petri net model
- we use a simple event log as input

A simple event log $\mathcal{L}$ is a multi-set of traces over $\mathcal{A}$, i.e., $\mathcal{L} \in \mathbb{B}(\mathcal{A}^*)$
$\mathcal{L}1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$

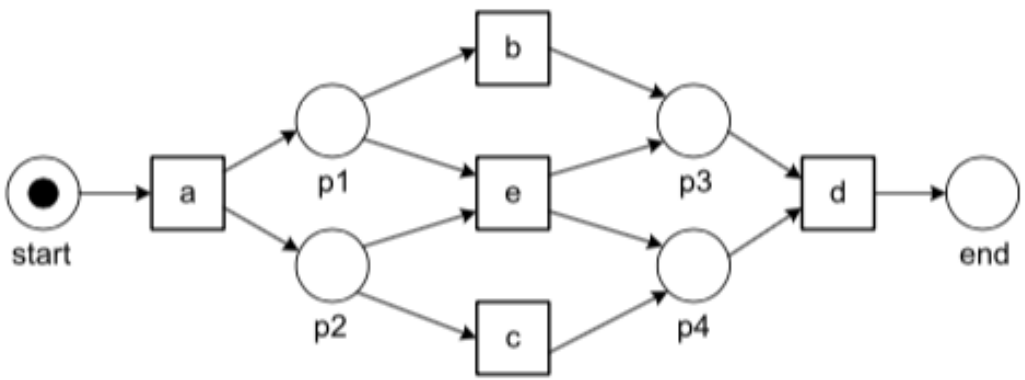
The goal is now to discover a Petri net that can replay event log L1
—> Ideally, the Petri net is a sound WF-net

Discovery

Definition (Specific process discovery problem)

A process discovery algorithm is a function γ that maps a log $\mathcal{L} \in \mathbb{B}(\mathcal{A}^*)$ onto a marked Petri net $\gamma(\mathcal{L}) = (\mathcal{N}, \mathcal{M})$. Ideally, $\mathcal{N}$ is a sound WF-net and all traces in $\mathcal{L}$ correspond to possible firing sequences of $(\mathcal{N}, \mathcal{M})$.

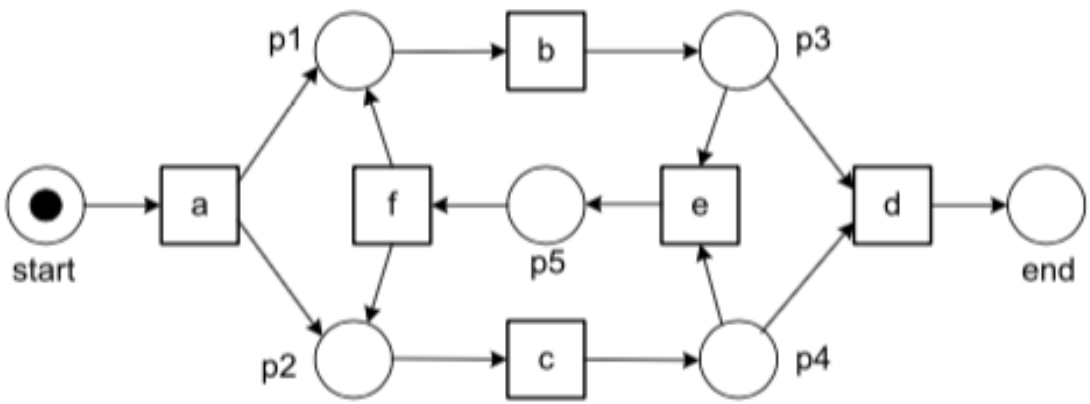
Function γ defines a so-called Play-In technique
Based on $\mathcal{L}1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$, a process discovery algorithm γ could discover the following WF-net



It is easy to see that the WF-net can indeed replay all traces in the event log

Discovery into practice

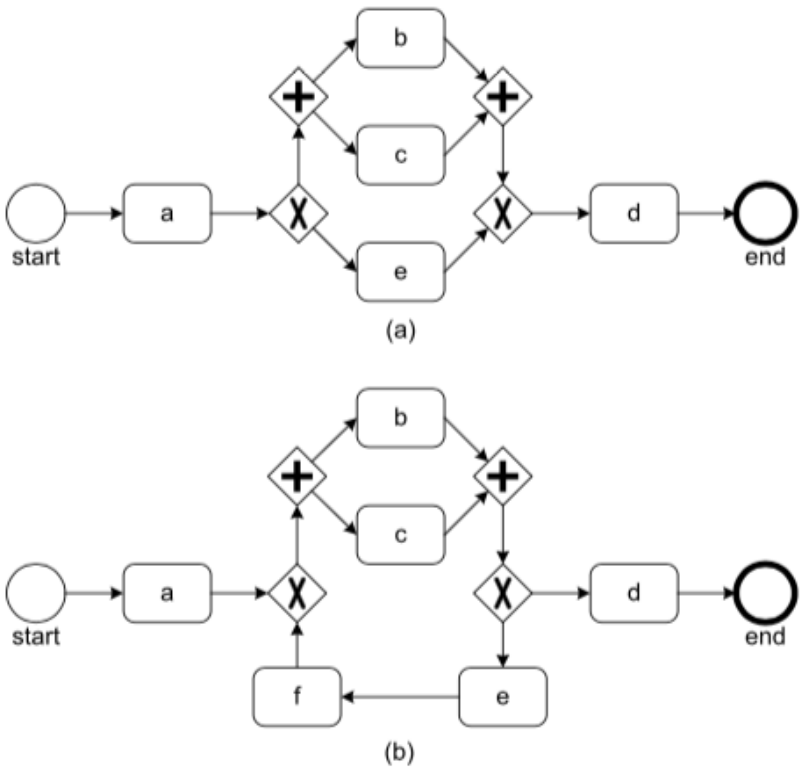
$\mathcal{L}2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^4, \langle a, b, c, e, f, b, c, d\rangle^2,$
$\langle a, b, c, e, f, c, b, d\rangle^2, \langle a, c, b, e, f, b, c, d\rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d\rangle$
$\mathcal{L}2$ is a simple event log consisting of 13 cases represented by 6 different traces.
Based on event log L2, let's discover the WF-net!!!



This WF-net can indeed replay all traces in the log. However, not all firing sequences of $\mathcal{N}2$ correspond to traces in $\mathcal{L}2$.

Discovery is language indipendent

Compact formalisms with formal semantics like Petri nets are most suitable to develop and explain process mining algorithms. The representation used to show results to end users is less relevant for the actual process discovery task.



The discovered model should be ?representative? for the behavior seen in the event log

Discovery is language indipendent

The discovered model should be ?representative? for the behavior seen in the event log

- (Fitness) The discovered model should allow for the behavior seen in the event log
- (Precision) The discovered model should not allow for behavior completely un-related to what was seen in the event log
- (Generalization) The discovered model should generalize the example behavior seen in the event log
- (Simplicity) The discovered model should be as simple as possible

It turns out to be challenging to balance the four quality criteria.

- an oversimplified model is likely to have a low fitness or lack of precision
- there is an obvious trade-off between underfitting and overfitting

A Simple Algorithm for Process Discovery

alpha-algorithm

The alpha-algorithm should not be seen as a very practical mining technique as:

- it has problems with noise
- infrequent/incomplete behavior
- complex routing constructs

INPUT: is a simple event log $\mathcal{L}$ over $\mathcal{A}$
OUTPUT : is a marked Petri net $\alpha(\mathcal{L}) = (\mathcal{N}, \mathcal{M})$

The α-algorithm scans the event log for particular patterns –> For example, if activity a is followed by b but b is never followed by a, then it is assumed that there is a causal dependency between a and b.

To reflect this dependency, the corresponding Petri net should have a place connecting a to b.

We distinguish four log-based ordering relations that aim to capture relevant patterns in the log.

Log-based ordering relations

Definition 6.3 (Log-based ordering relations) Let L be an event log over $\mathscr{A}$, i.e., $L \in \mathbb{B}(\mathscr{A}^*)$. Let $a, b \in \mathscr{A}$.

- $a >_L b$ if and only if there is a trace $\sigma = \langle t_1, t_2, t_3, \ldots, t_n \rangle$ and $i \in \{1, \ldots, n-1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$;
- $a \rightarrow_L b$ if and only if $a >_L b$ and $b \not>_L a$;
- $a \#_L b$ if and only if $a \not>_L b$ and $b \not>_L a$; and
- $a \parallel_L b$ if and only if $a >_L b$ and $b >_L a$.

Consider, for instance, $L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$ again. For this event log, the following log-based ordering relations can be found:

$$>_{L_1} = \{(a,b), (a,c), (a,e), (b,c), (c,b), (b,d), (c,d), (e,d)\}$$

$$\rightarrow_{L_1} = \{(a,b), (a,c), (a,e), (b,d), (c,d), (e,d)\}$$

$$\#_{L_1} = \{(a,a), (a,d), (b,b), (b,e), (c,c), (c,e), (d,a), (d,d), (e,b), (e,c), (e,e)\}$$

$$\parallel_{L_1} = \{(b,c), (c,b)\}$$

Rediscovering Process Models

Challenges