Process Mining
(ProMi)

Winter 2015/16

Matthias Weidlich
II. Process Discovery

Mining Algorithms

Inductive Miner
Further Issues in Discovery

The Heuristic Miner improves the robustness against noise compared to the $\alpha(+/++)$ – algorithms

Yet, there are still fundamental issues:

• Discovery may yield a model that is unsound
• Discovered model may not be able to replay (generate) all the traces observed in the log

These requirements are important in practice:

• Usefulness of discovered models
• Traceability of discovered models
Deadlocks

Traces:
ABC
BAC
Lack of Fitness

Traces:
ABCD
ACBD
AD
The Representational Bias

Representational Bias: Discovery results depend on target formalism

- **Only what can be specified** in target formalism can be discovered – matter of expressiveness
- **Everything that can be specified** in target formalism can be discovered – matter of correctness

As such, target formalism defines bounds for discovery
Example: Concurrency & Silent Transitions

Traces:
ABCD
ACBD
AD

Dependency graph

WF-net system

WF-net system incl. silent steps
Example: N-out-of-M Join

Traces:
ABCD

ACDBE

ACBDE

ACBDE

ACBED

ACBED

ACBED

ADCEB

ACDEB
The Target Formalism

To cope with the representational bias
  • Use formalism that is expressive
  • Use formalism that defines only correct models

Expressivity is particularly related to labelling
  • Silent steps (silent transitions in Petri net terminology)
  • Uniqueness of transition labels
Impact of Labelled Transitions

Traces:
ABCA
ACBA
Block-Structured Workflow Nets

Silent transition
(aka $\tau$-transition)
Process Trees

Notion of a process tree describes a language, i.e., a set of complete traces

Syntax defined recursively based on set of operators

- Alphabet of symbols $\Sigma$ and silent label $\tau$
- Set of operators $\Omega$
- Recursive definition:
  - $x$ with $x \in \Sigma \cup \{\tau\}$ is a process tree
  - Let $M_1, \ldots, M_k$, $k > 0$, be process trees and $\omega \in \Omega$, then $\Omega(M_1, \ldots, M_k)$ is a process tree
Process Trees – Semantics

Language $L$ of process tree $M$ is also defined recursively

- Based on language operator $\omega_l$ for each operator $\omega$
- Semantics derived by language join
  - $L(a) = \{a\}$ for $a \in \Sigma$ and $L(\tau) = \{}$
  - $L(\omega(M_1, \ldots, M_n)) = \omega_1(L(M_1), \ldots, L(M_n))$

Common operators:
- Exclusive choice:
  
  $$x_1(L_1, \ldots, L_n) = \bigcup_{1 \leq i \leq n} L_i$$

- Sequence:
  
  $$\rightarrow_1 (L_1, \ldots, L_n) = \{t_1 t_2 \ldots t_n | \forall i \in 1 \ldots n: t_i \in L_i\}$$
Further operators:

- **Structured Loop (one body, multiple loop back paths):**

  \[ \bigcup_1 \left( L_1, ..., L_n \right) \]

  \[ = \{ t_1 t'_1 t_2 t'_2 ... t_m \mid \forall i \in 1 ... m: t_i \in L_1 \land t'_i \in \bigcup_{2 \leq j \leq n} L_j \} \]

- **Interleaved (“parallel”) execution:**

  \[ \land_1 \left( L_1, ..., L_n \right) = \{ t \mid t \in \{ t_1, ..., t_n \} \simeq \land \forall i \in 1 ... n: t_i \in L_i \} \]

  with \( \{ t_1, ..., t_n \} \simeq \) as the set of all interleavings of traces \( t_1, ..., t_n \)
Example

\[\rightarrow(a, \Omega(\rightarrow(\land(x(b, c), d), e), f), x(g, h))\]
Inductive Miner

Inductive Miner: Discovery approach to construct a Process Tree for a given log

- All discovered models correspond to sound, block-structured WF-net systems
- The model always fits, i.e., the model can generate the traces in the log

In line with the syntax/semantics definition of Process Trees, the approach works recursively

- Divide and conquer: split log, construct part of process tree
- Proceed with handling split parts of log separately

Details:
Inductive Miner – Intuition

Select cut
(operator & activity partition)

Split log
Recurse

→ \{a\}\{b, c\}

∧\{b\}\{c\}

(slides partly due to Sander Leemans)
Cut Selection – Intuition

- Partition direct successor graph
- $\rightarrow$: edges crossing one-way only
- $\wedge$: all edges crossing in both ways
Cuts in more detail...

Begin with graph spanned by direct successor relation

A cut is a partition of the graph nodes into disjoint sets $\Sigma_1, \ldots, \Sigma_n$

- **Sequence cut:** For any two nodes $a \in \Sigma_i, b \in \Sigma_j, i \leq j$, there is a path from $a$ to $b$, but not vice versa
- **Exclusive cut:** No two nodes $a \in \Sigma_i, b \in \Sigma_j, i \neq j$ are connected
- **Parallel cut:** Each $\Sigma_i$ has a start and an end node, and any two nodes $a \in \Sigma_i, b \in \Sigma_j, i \neq j$ are connected by edges $(a, b)$ and $(b, a)$
- **Loop cut:** Each $\Sigma_i$ has a start and an end node, and no two nodes $a \in \Sigma_i, b \in \Sigma_j, i, j > 1$ are connected, and any edge between $\Sigma_1$ and $\Sigma_i, i > 1$ either leaves an end node of $\Sigma_1$ or reaches a start node of $\Sigma_1$

Consider only maximal cuts
Cut Types

exclusive choice:

sequence:

parallel:

loop:
Algorithm

Inductive Miner Algorithm:
Input: Direct successor graph

1) Try to find one of the four cut types
2) If cut is trivial (empty or single transition), return empty trace or trace containing single transition
3) Else, if non-trivial cut is found
   a) Apply log split according to cut type
   b) Insert node into process tree according to cut type
   c) Recurse by applying the algorithm to all log parts
4) Else, return flower model for the log
Flower Model

Basic structure to represent all possible traces induced by a set of tasks/transitions

• The WF-net system constructed by adding and connecting initial and final places is always sound

• The model fits any log containing traces built of the given set of transitions
Take Away

Inductive Miner creates sound and fitting models

Tries to create models “as structured” as possible

Rediscoverability results available

Plain version of the algorithm is not robust against noise