ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Business Process Mining

Federico Chesani
Department of
Computer Science and Engineering
University of Bologna

Credits and sources

These slides have been partly inspired by:

- Process Mining Manifesto
  van der Aalst W. et al. (2012) Process Mining Manifesto. In: Daniel F., Barkaoui K., Dustdar S. (eds) Business Process Management Workshops. BPM 2011. LNBIP, vol 99. Springer, Berlin, Heidelberg https://doi.org/10.1007/978-3-642-28108-2_19
  Available also as a IEEE Task Force on BPM document:
  https://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_mining_manifesto

- Wil M. P. van der Aalst: Process Mining - Data Science in Action, Second Edition. Springer 2016, ISBN 978-3-662-49850-7, pp. 3-452
- http://www.processmining.org/book/start

- https://www.coursera.org/learn/process-mining

- http://www.promtools.org/

Outline

- From the Manifesto: definitions and tasks
- The Manifesto's Guiding Principles
    - Event logs: why, requirements, and standards

- Process Discovery

- Conformance Checking, and Fitness

- Challenges…

Business Process Mining

Defined as:

"… to the discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs …"

It includes:

- (automated) process discovery
- conformance checking
- social network/organizational mining
- automated construction of simulated models
- model extension
- model repair
- case prediction
- history-based recommendations

W.r.t. other (Business intelligence) related disciplines, the focus here is always the **process**.

Business Process Mining

Defined as:

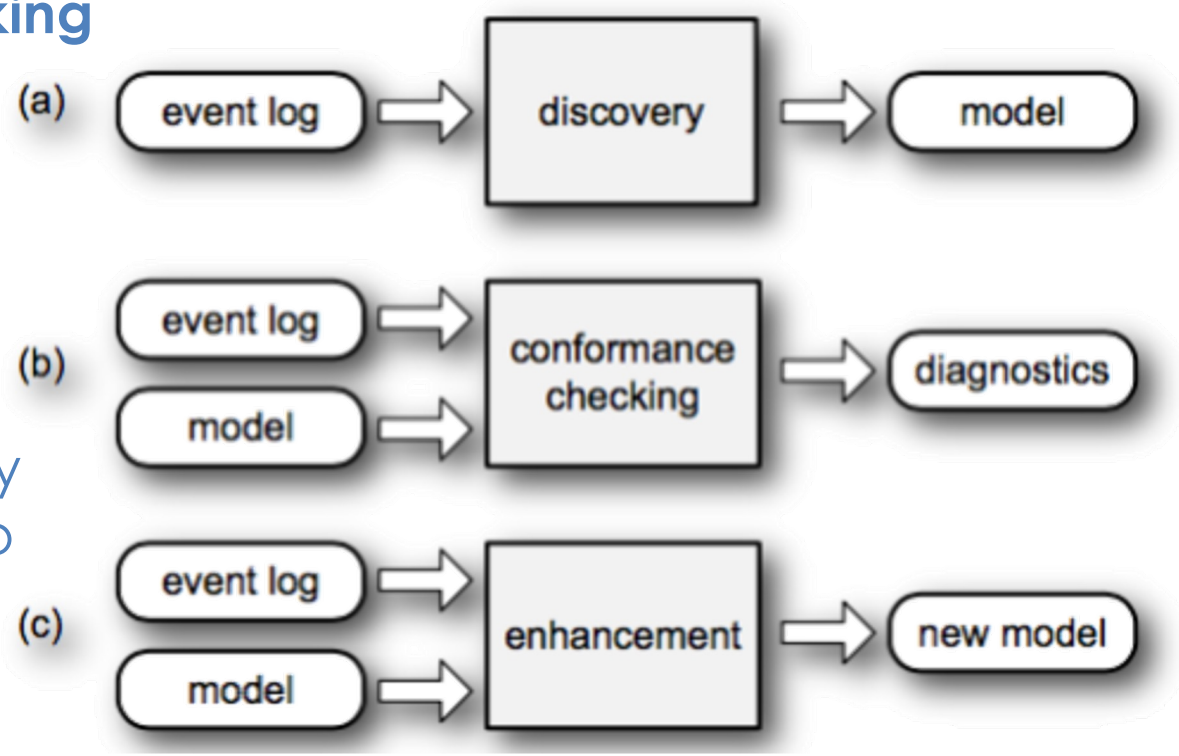
"… to the discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs …"

- discover, monitor and improve…
- … real processes … (vs. assumed processes)
- … from event logs …

… discover, monitor and improve …

Three main types of process mining, from the input/output perspective:

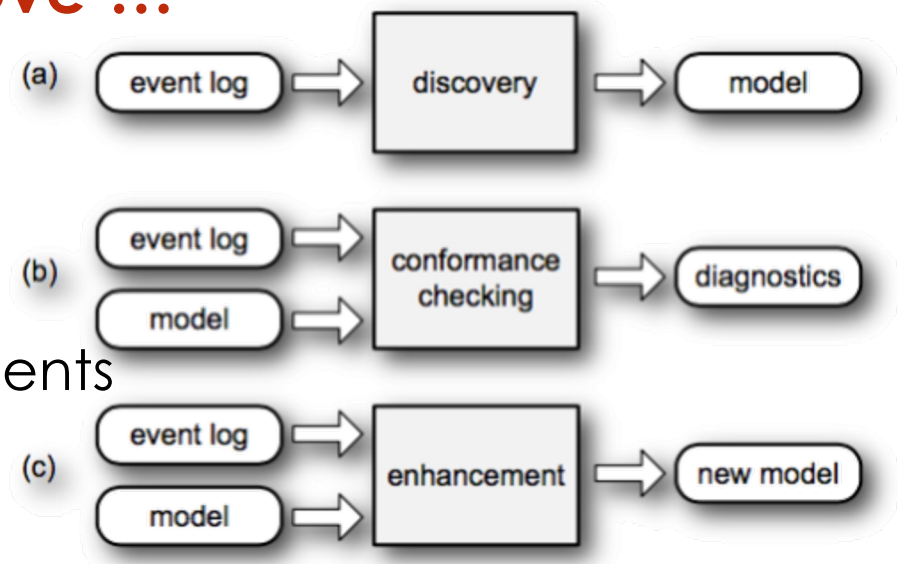
1. **Discovery**
2. **Conformance Checking**
3. **Enhancement**



Assumption:

It is possible to sequentially record events that refer to activities

… discover, monitor and improve …



Assumption:

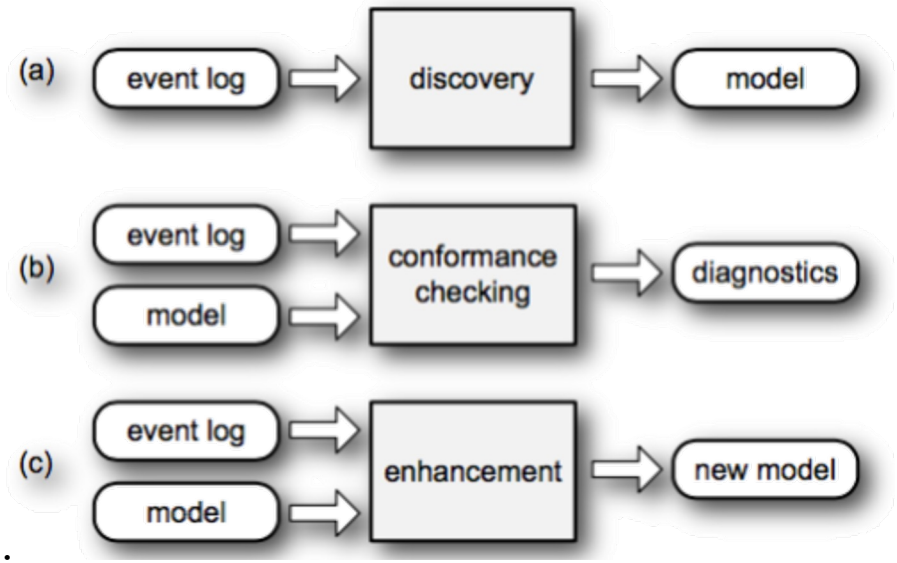
It is possible to sequentially record events such that

- each event refers to an activity
- each event is related to a particular case
- extra information such as timestamp, resources, data elements are recorder as well

In other words, an event log should exist!

… easy to say, harder to achieve…

Common misconceptions



- Discovery is not *only* control-flow discovery
  (however, huge research effort has been put in particular on this aspect)

- Process mining is not just data mining…
  (e.g., concurrency behaviours are hard to be captured by data mining techniques)

- Process mining is not limited to offline (post mortem) analysis

6 Guiding Principles

1. Event Data should be treated as first-class citizens
2. Log Extraction should be driven by questions
3. Concurrency, Choice and other basic control-flow constructs should be supported
4. Events should be related to model elements
5. Models should be treated as purposeful abstractions of reality
6. Process mining should be a continuous process

GP1: Event Data should be treated as first-class citizens

The starting point is always the event log, intended as a collection of events related by some case identifier

Desiderata:

- events should be trustworthy (what is logged, indeed it happened)
- events log should be complete (???)
  (e.g., given some scope, no relevant events are missing)
- events should have a well-defined semantics
  (and a formal mapping to model elements)
- privacy and security concerns should be addressed as well

GP1: Event Data should be treated as first-class citizens

Syntactically, a standard for event logs has been proposed:

IEEE 1849-2016 XES Standard: IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams

https://xes-standard.org/

Minimum requirements for a log to be compliant:

- event description
- relation to activity start/activity end
- time stamp
- case id

Event logs: 5 quality levels

Level	Characterization	Examples
★★★★★	Highest level: the event log is of excellent quality (i.e., trustworthy and complete) and events are well-defined. Events are recorded in an automatic, systematic, reliable, and safe manner. Privacy and security considerations are addressed adequately. Moreover, the events recorded (and all of their attributes) have clear semantics. This implies the existence of one or more ontologies. Events and their attributes point to this ontology.	Semantically annotated logs of BPM systems.
★★★★	Events are recorded automatically and in a systematic and reliable manner, i.e., logs are trustworthy and complete. Unlike the systems operating at level ★★★, notions such as process instance (case) and activity are supported in an explicit manner.	Events logs of traditional BPM/ workflow systems.
★★★	Events are recorded automatically, but no systematic approach is followed to record events. However, unlike logs at level ★★, there is some level of guarantee that the events recorded match reality (i.e., the event log is trustworthy but not necessarily complete). Consider, for example, the events recorded by an ERP system. Although events need to be extracted from a variety of tables, the information can be assumed to be correct (e.g., it is safe to assume that a payment recorded by the ERP actually exists and vice versa).	Tables in ERP systems, event logs of CRM systems, transaction logs of messaging systems, event logs of high-tech systems, etc.
★★	Events are recorded automatically, i.e., as a by-product of some information system. Coverage varies, i.e., no systematic approach is followed to decide which events are recorded. Moreover, it is possible to bypass the information system. Hence, events may be missing or not recorded properly.	Event logs of document and product management systems, error logs of embedded systems, worksheets of service engineers, etc.
★	Lowest level: event logs are of poor quality. Recorded events may not correspond to reality and events may be missing. Event logs for which events are recorded by hand typically have such characteristics.	Trails left in paper documents routed through the organization ("yellow notes"), paper-based medical records, etc.

Table 1: Maturity levels for event logs.

EVENT LOGS

≠

Log Files

GP2: Log Extraction should be driven by questions

It is rarely the case that logs are already available in the right format

- e.g., SAP stores process data in hundreds of tables… which query to run?
- drill on customer, on order, on order line, on delivery?
- which level of detail?
- in distributed environments (clouds), event logs might be consequence of proper join of different system logs… it is not just a simple append of log files

GP3: Concurrency, Choice and other basic control-flow constructs should be supported

Let us suppose to observe the following event log:

L = {

ABCDE,

ABDCE,

ACBDE,

ACDBE,

ADBCE,

ADCBE

}

Which is the control flow?

- Activities A and E are always the first, and the last, respectively
- Activities B, C and D are executed in all the possible orders

GP3: Concurrency, Choice and other basic control-flow constructs should be supported

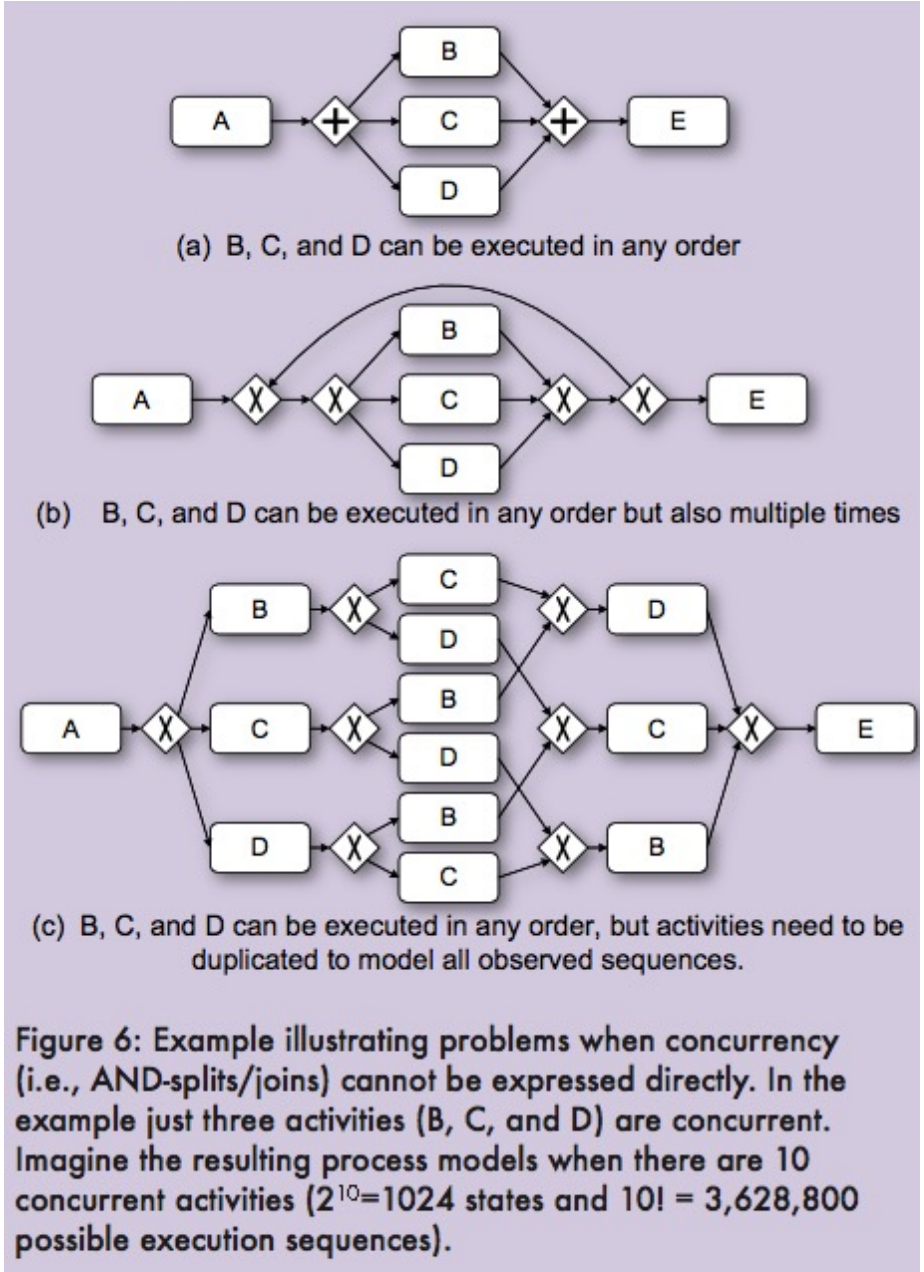


Figure 6: Example illustrating problems when concurrency (i.e., AND-splits/joins) cannot be expressed directly. In the example just three activities (B, C, and D) are concurrent. Imagine the resulting process models when there are 10 concurrent activities (2^{10}=1024 states and 10! = 3,628,800 possible execution sequences).

- (a) the tool support concurrency -> the learned model is simple and concise
- (b) the tool does not support concurrency -> the learned model underfits the log
- (c) the tool perfectly fits the model -> not scalable to many activities; same activity compares in two different points in the model

GP4: Events should be related to model elements
GP5: Models should be treated as purposeful abstractions of reality

Which relation between the events in the logs, and the elements in the model?

- need proper mapping
- need proper semantics -> model ontologies

The “geographic map” metaphor:

- which level of detail
- aggregation of information… how?
- use of colors, of ”geographic position”, etc. ?

GP6: Process Mining should be a continuous process

Applications such as:

- model drift detection
- prediction
- recommendation
- enhancement

all requires that continuous process discovery and monitor is carried through while systems execute.

Process Discovery

General process discovery problem

Let L be an event log as defined … . A process discovery algorithm is a function that maps L onto a process model such that the model is "representative" for the behavior seen in the event log.

WHICH PROCESS MODELLING LANGUAGE?

Specific process discovery problem

A process discovery algorithm is a function γ that maps a log L onto a marked Petri net $\gamma(L) = (N,M)$.

Ideally, N is a sound WF-net and all traces in L correspond to possible firing sequences of (N,M).

Process Discovery - example

Let L = {
<abcd>
<abcd>
<abcd>
<acbd>
<acbd>
<aed>
}

From this…

… to this.

HOW???.

The α-algorithm

- It was one of the first algorithms to deal with concurrency
- Presented in
  W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, **16**(9):1128–1142, 2004.
- It outputs (sound) WorkflowNets

The α-algorithm

Intuition:

- Let us decide which are the relation we are interested in…
- Look in the log for all such relations
- Focus on the (most interesting) relations, and identify (biggest) sets of activities involved
- Remove redundancies
- Represent them as a PetriNet transition
- Add the start, the end, and output the PetriNet

The α-algorithm – Log-based ordering relations

Let us decide which are the relation we are interested in…

We are surely interested in those relation where an event appears immediately after another event… -> ordering relations

- $a >_L b$ 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
- $a \|_L b$ if and only if $a >_L b$ and $b >_L a$.

Which is the most interesting one?

The α-algorithm – Look in the log for such relations…

Let L = {

<abcd>

<abcd>

<abcd>

<acbd>

<acbd>

<aed>

}

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

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

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

$$\|_{L_1} = \{(b,c),(c,b)\}$$

The α-algorithm – the footprint matrix…

Let L = {
<abcd>
<abcd>
<abcd>
<acbd>
<acbd>
<aed>
}

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

by construction, second, third and fourth type of ordering relations are mutually exclusive

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

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

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

$$\|_{L_1} = \{(b,c),(c,b)\}$$

The α-algorithm – looking for → and # (only)

Let L = {
<abcd>
<abcd>
<abcd>
<acbd>
<acbd>
<aed>
}

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

Look for SETs (A,B) of activities such that for the elements of each it holds: A→B, and A#A, and B#B

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

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

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

$$\|_{L_1} = \{(b, c), (c, b)\}$$

The α-algorithm – looking for → and # (only)

Let L = {

<abcd>

<abcd>

<abcd>

<acbd>

<acbd>

<aed>

}

	a_1	a_2	…	a_m	b_1	b_2	…	b_n
a_1	#	#	…	#	→	→	…	→
a_2	#	#	…	#	→	→	…	→
…	…	…	…	…	…	…	…	…
a_m	#	#	…	#	→	→	…	→
b_1	←	←	…	←	#	#	…	#
b_2	←	←	…	←	#	#	…	#
…	…	…	…	…	…	…	…	…
b_n	←	←	…	←	#	#	…	#

Look for SETs (A,B) of activities such that for the elements of each it holds: A→B, and A#A, and B#B…

… BY SWAPPING ROWS AND COLUMNS, looking for a pattern

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

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

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

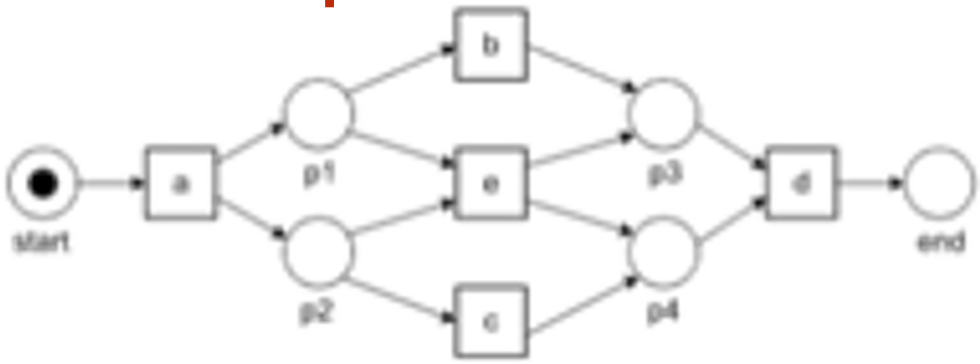
$$\|_{L_1} = \{(b,c),(c,b)\}$$

The α-algorithm

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

The α-algorithm – the example…

Let L = {
<abcd>
<abcd>
<abcd>
<acbd>
<acbd>
<aed>
}



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

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

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

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

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

$$\|_{L_1} = \{(b, c), (c, b)\}$$

The α-algorithm – some known limits

- It can learn complex nets, that are not necessarily, e.g. with implicit transition places

- It cannot learn loops of dimension 1 and 2

  α^+--algorithm presented in:

  A.K. Alves de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Workflow Mining: Current Status and Future Directions. In R. Meersman, Z. Tari, and D.C. Schmidt, editors, On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, volume 2888 of Lecture Notes in Computer Science, pages 389–406. Springer, Berlin, 2003.

- It does not deal with *non-local-dependencies* – partially solved in:

  L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun. Mining Process Models with Non-Free-Choice Constructs. *Data Mining and Knowledge Discovery*, **15**(2):145–180, 2007.

- Frequencies are not taken into account

- No duplicate activities, no silent transitions

Other learning algorithms…

- Heuristic Mining
- Genetic Process Mining
- Region-based mining
- Inductive Mining

But… which should be the right model to learn?

The concept of right model is wrong!
Such a model does not exist!!!!!!!

- There is a log…
- There can be different models capturing the same process
  - Trace equivalence
  - Bisimulation equivalence
  - Branching bisimulation equivalence

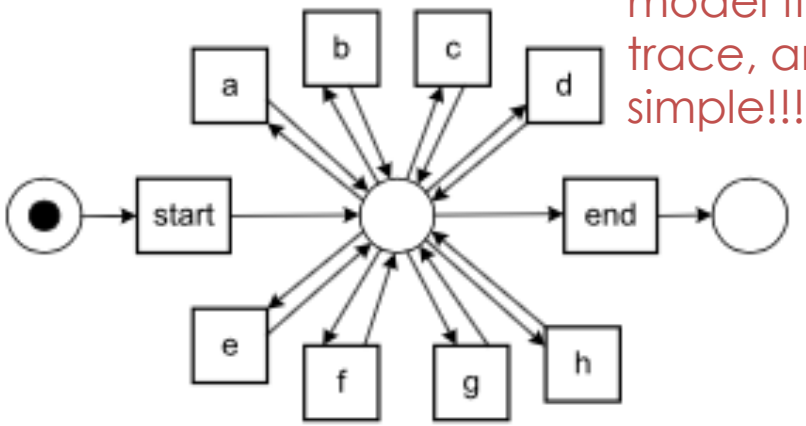
But… what model have we learned?

Models are view on the reality!

- A model views reality from a particular angle
- A model frames only a piece of reality
- A model views reality with a certain resolution

General criteria for judging a model:

- Fitness
- Simplicity
- Precision
- Generalization



The Flower model fit ANY trace, and it's simple!!!

Conformance checking in Process Mining

The terms conformance checking has a specific meaning in Process Mining: it focuses on events on the log, and relates them to the process executions forecasted by the model

Discrepancies can have different meanings:

- if the model is intended to be descriptive, then the model need to be improved/adjusted
- if the model is intended to be prescriptive (normative), then discrepancies are interpreted as deviations
    - undesirable vs. desirable deviations

In Logic, and MAS, the conformance checking has a completely different meaning, and it is about a prescriptive model, and the respect of agents

Conformance checking via Token replay

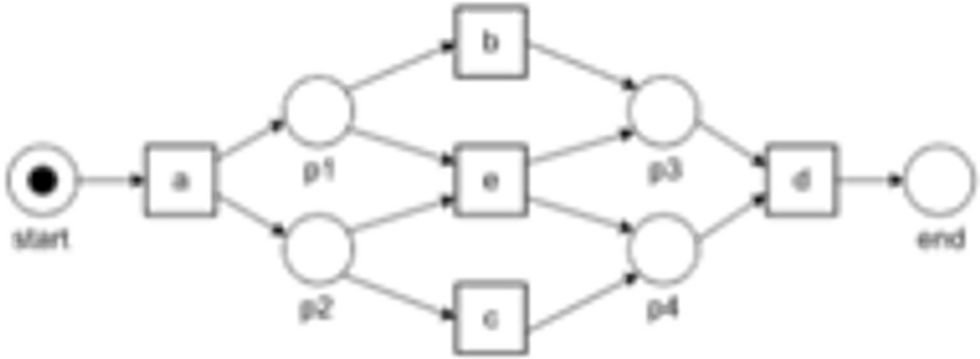
The starting point is a Petri Net.

Given a trace, it can be interpreted as the ordered list of transitions that must be activated. It is possible to replay a trace, following tokens that *moves* around in the PetriNet.

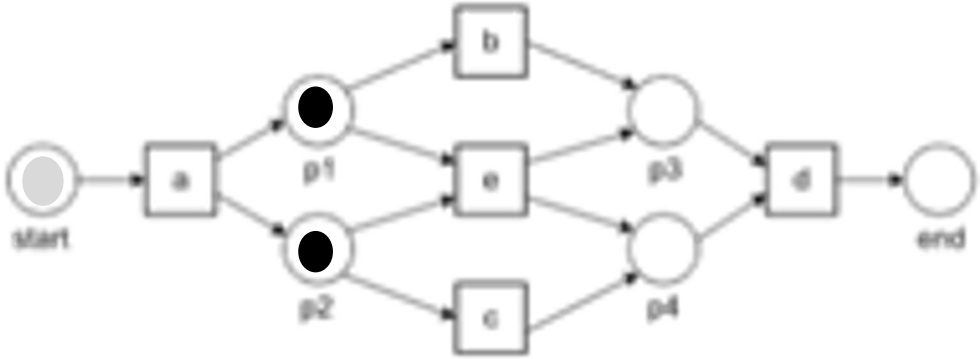
σ = <abcd>

Conformance checking via Token replay

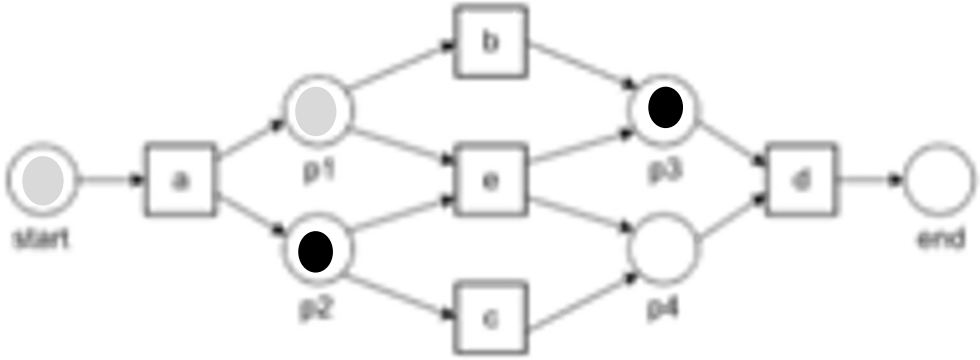
σ = <abcd>



σ = <abcd>

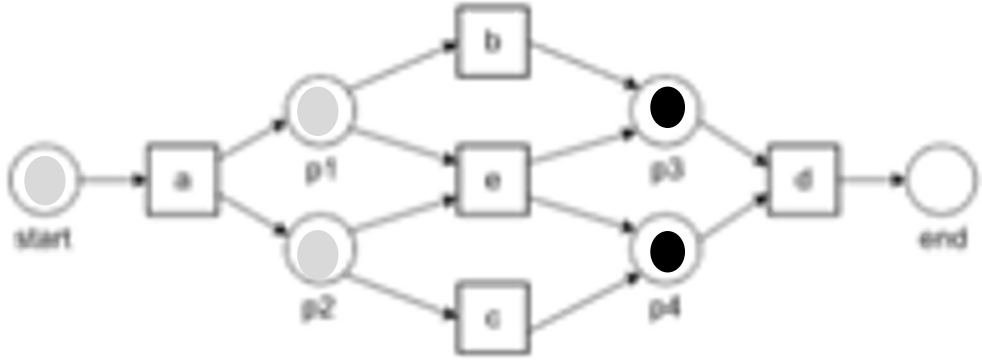


σ = <abcd>

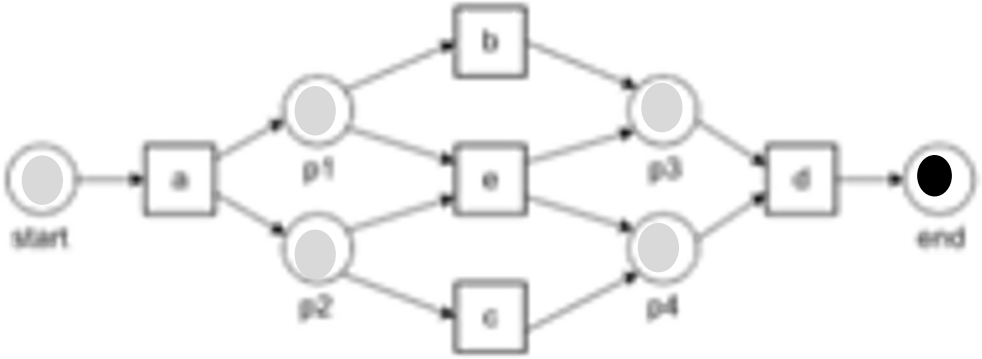


Conformance checking via Token replay

σ = <abcd>



σ = <abcd>



Conformance checking via Token replay

σ = <abcd>

If a model can replay a trace, then the model fits that trace…

Naïve idea: let's count how many traces can be replayed by the model, and compute the fitness as the ratio between replayed traces and total number of traces in the log…

Very limited approach: in this method, either a trace can be replayed or not… independently of how much part of the trace can be effectively replayed…

Conformance checking via Token replay

σ = <abcd>

Better idea: let us replay the trace, and when needed, let us add /remove tokens…

… then let us keep track of how many tokens we added/remove, and compute a more precise fitness function…

Conformance checking via Token replay

σ = <abcd>

Definitions:

- p: # of produced tokens (as for the normal running of the net)
- c: # of consumed tokens (as for the normal running of the net)
- m: # of missing tokens (that have been added)
- r: # of remaining tokens (that we add to remove ad the end)

Conformance checking via Token replay - Fitness

$$fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

$$fitness(L, N) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}}\right)$$

Note: this fitness function measures the tokens in place, but is commonly interpreted as a measures on events. E.g., fitness(L,N)=0.9 is interprets as "90% of events in log L can be replayed by the model N"

Conformance checking via Token replay - Limits

Fitness defined through the token replay has few drawbacks:

- fitness values tend to be too high for extremely problematic logs
- if there are many deviations, the net get flooded of tokens, thus allowing for any behaviour
- the approach is PetriNet-specific
- if a case does not fit at all, an ad-hoc path is not considered/created
- more difficult to achieve if there are duplicated activities and/or silent transitions

Conformance checking via Alignment

Alignment is based on aligning the case with the possible instance generated by the model

$$\gamma_1 = \begin{array}{|c|c|c|c|c|} \hline a & d & b & e & h \\ \hline a & d & b & e & h \\ \hline \end{array}$$

Log…

Trace allowed by the model…

Optimal Alignment

$$\gamma_{2a} = \begin{array}{|c|c|c|c|c|c|} \hline a & \gg & d & b & e & h \\ \hline a & b & d & \gg & e & h \\ \hline \end{array} \qquad \gamma_{2b} = \begin{array}{|c|c|c|c|c|c|} \hline a & \gg & d & b & e & h \\ \hline a & c & d & \gg & e & h \\ \hline \end{array} \qquad \gamma_{2c} = \begin{array}{|c|c|c|c|c|c|} \hline a & d & b & \gg & e & h \\ \hline a & \gg & b & d & e & h \\ \hline \end{array}$$

The symbol ≫ indicates a misalignment

Conformance checking via Alignment

To compute fitness, two elements have to be determined:

- the *optimal alignment* defined as the alignment with lowest possible cost
- the *worst alignment* possible

Note that for each move of misalignment, different costs can be associated, on the base of the *type of misalignment,* and on the specific activity involved

Conformance checking via Alignment

$$fitness(\sigma, N) = 1 - \frac{\delta(\lambda^N_{opt}(\sigma))}{\delta(\lambda^N_{worst}(\sigma))}$$

$$fitness(L, N) = 1 - \frac{\sum_{\sigma \in L} L(\sigma) \times \delta(\lambda^N_{opt}(\sigma))}{\sum_{\sigma \in L} L(\sigma) \times \delta(\lambda^N_{worst}(\sigma))}$$

Challenges for Process Mining…

C1: Finding, merging and cleaning event data

From machine learning and data mining experience, we know that at least 70% of time of any discovery process is spent on preparing the data.

Most frequent issues:

- Data is distributed
- data is object centric, rather than process centric (ORM exasperate such aspect)
- data may be incomplete
- outliers: hot to identify them, how to deal with them
- events may contain different granularity data
- events happen within contexts (weather conditions, week days, etc.)

Challenges …

C2: Dealing with complex Event Logs having diverse characteristics

Which are the differences between:

- a log with 1000 cases, 10 events per case (average), little flow variation
- a log with 100 cases, 100 events per case (average), each a unique path

How to deal with such diversity of processes, using a single discovery algorithm?

Challenges …

C3: Creating representative benchmarks

Which benchmarks?

- real benchmarks
- ad-hoc synthetic benchmarks with controlled properties on data distribution, temporal distribution, missing events distribution, resource allocation issues, etc.

Some initiative for providing log benchmarks:

- www.processmining.org
- Business Process Intelligence Challenge, since 2011, provides every year a challenge for discovery.
- At ICPM 2019 the first conformance cheking challenge has been proposed, on real logs of health processes https://icpmconference.org/2019/icpm-2019/contests-challenges/1st-conformance-checking-challenge-2019-ccc19/

Challenges …

C4: Dealing with Concept drift

Drift happens when the process changes while being analyzed…
i.e., always in real life!!!

- Successful business are highly dynamics -> models are obsolete even before we complete to learn them
- Costs: continuous learning ask for cheap and fast tool

C5: Improving the repreentational bias used for process discovery

Distinguish between:

- language used to learn the process
- language used to represent the process

Example: BPM offers more than 50 symbols, PetriNet provides just three symbols…

Challenges …

C6: Balancing between Quality Criteria such as Fitness, Simplicity, Precision, and Generalization

- **Fitness**: how much the discovered model describes the behaviour seen in the log?
- **Simplicity**: is there a simpler model capturing the same behaviour?
- **Precision**: does the model allow too much behaviour?
- **Generalization**: is the model general enough?

Other Challenges …

C7: Cross-organizational mining

C8: Providing operational support

C9: Combining process mining with other types of analysis

C10: Improving usability for non-experts

C11: Improving understandability for non -experts

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Federico Chesani

DISI – Department of Computer Science and Engineering

federico.chesani@unibo.it

www.unibo.it