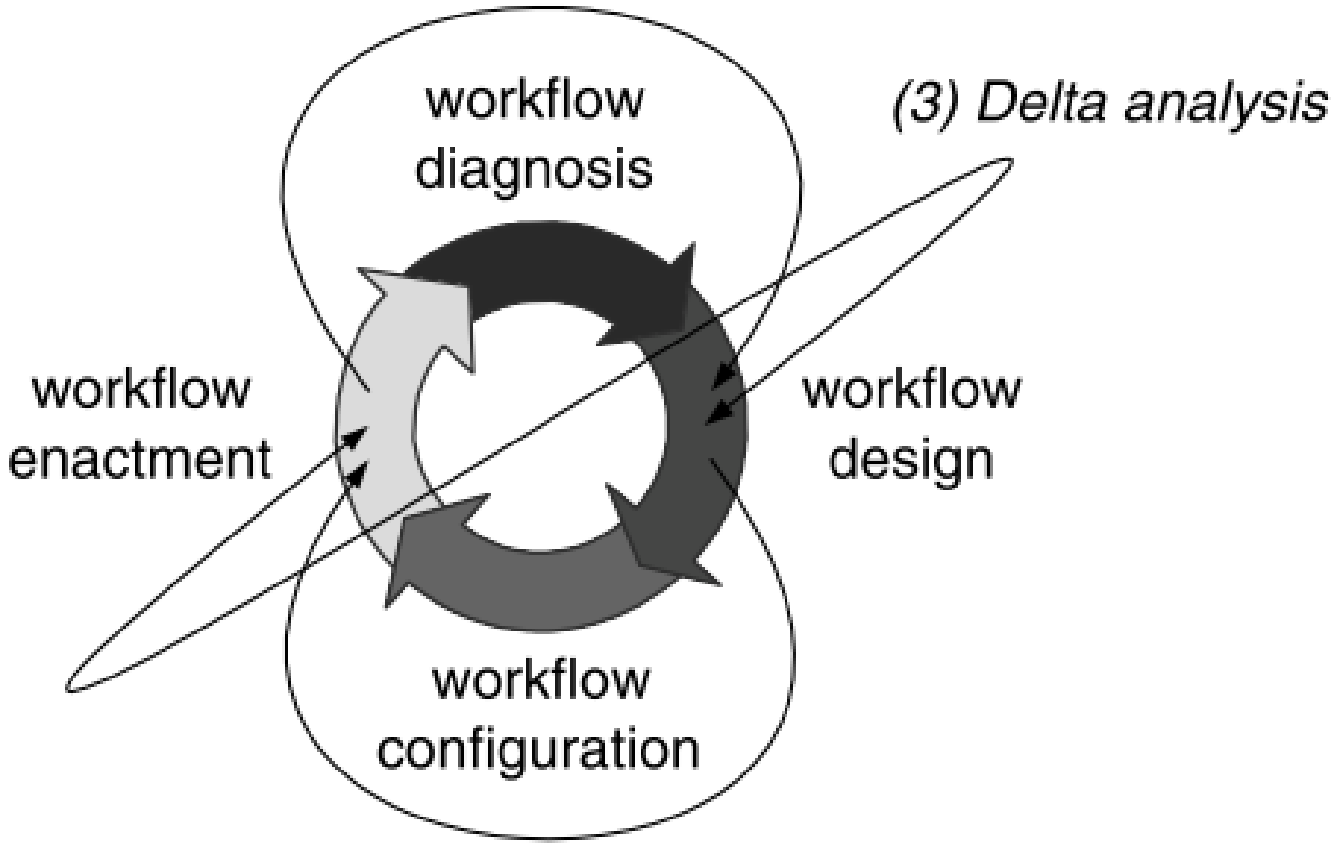
Process Mining

PROCESS MINING



PROCESS MINING

- Workflow diagnosis aims at exploiting runtime information to assist in business process re-engineering (BP design)
- Workflow/Process mining goal: collect data at runtime to support workflow design & analysis
  - Runtime Data can lead to a model explaining the events that took place
- Workflow modelling tends to be subjective
- Workflow mining can lead to the correction of subjective models according to the actual situation
  - Delta analysis: find discrepancies between constructed design & execution registered
    - Example: deviations due to human activities/work

PROCESS MINING

- Process mining: refers to methods for distilling a structured process description from a set of real executions
- An a posteriori model is constructed to be compared with an a priori model
- Can assist in adapting workflow to changing circumstances & fix design imperfections

WORKFLOW PROCESS/MINING

- Information extracted from transaction logs recording events
- Transaction logs properties:
  - Each event maps to a workflow task
  - Each event maps to a particular case
  - Events are totally ordered
- Interesting Aspects:
  - Time information can reveal task causality relations
  - Case & event type information can reveal relation between attributes of case & actual route/path taken by case
- Presence of Workflow Management System is not assumed
  - Just the availability of transaction logs
- Logs are exploited to construct a workflow specification mapping to the behaviour registered

PROCESS MINING EXAMPLE

- One case involves normal processing
- Other cases can involve further processing, a timeout and repeated execution of events
- Challenge is to discover a good model with as little information as possible
- Suppose only the task ordering is retained from log
  - 4 cases involve tasks A, B, C, D
  - 1 case involves tasks A, E and D
  - Each case starts with A and ends with D
  - If B is executed, C is also executed (C is before B in some cases)
- By assuming that the log is complete, containing all representative cases, then a petri net model can be obtained

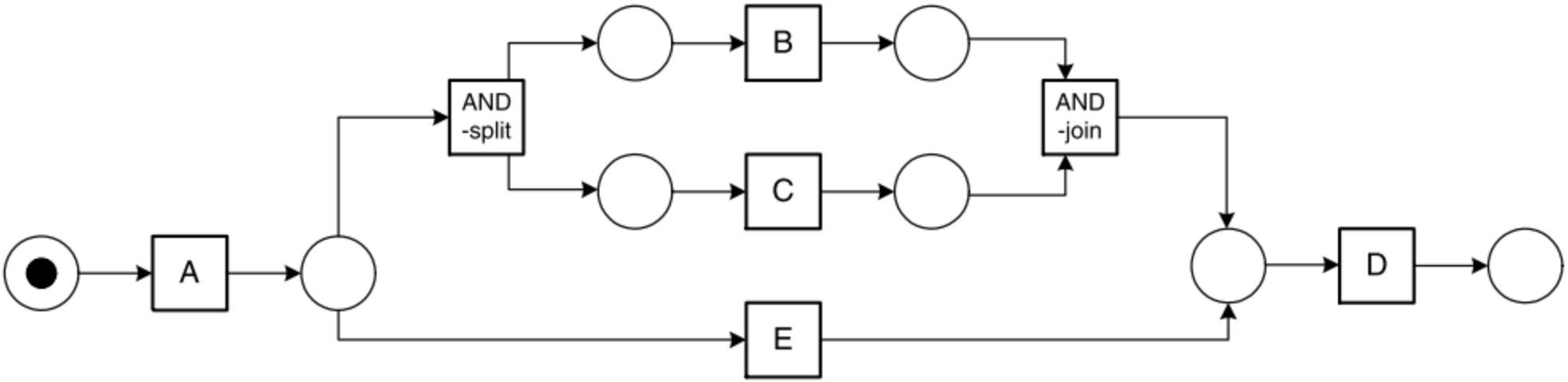
EXAMPLE –TRANSACTION LOG (STAFFWARE)

Case/Task	Event	User	Date
Case 2			
	Start	bvdongen@staffw_e	2015/06/19 12:58
Register	Process to	bvdongen@staffw_e	2015/06/19 12:58
Register	Released by	bvdongen@staffw_e	2015/06/19 12:58
Send Questionnaire	Process to	bvdongen@staffw_e	2015/06/19 12:58
Evaluate	Process to	bvdongen@staffw_e	2015/06/19 12:58
Send Questionnaire	Released by	bvdongen@staffw_e	2015/06/19 13:00
Receive Questionnaire	Process to	bvdongen@staffw_e	2015/06/19 13:00
Receive Questionnaire	Released by	bvdongen@staffw_e	2015/06/19 13:00
Evaluate	Released by	bvdongen@staffw_e	2015/06/19 13:00
Archive	Process to	bvdongen@staffw_e	2015/06/19 13:00
Archive	Released by	bvdongen@staffw_e	2015/06/19 13:00
	Terminated		2015/06/19 13:00

EXAMPLE – FILTERED LOG

Case	Task	Case	Task
Case 1	A	Case 3	C
Case 2	A	Case 3	D
Case 3	A	Case 4	B
Case 3	B	Case 5	E
Case 1	B	Case 5	D
Case 1	C	Case 4	D
Case 2	C		
Case 4	A		
Case 2	B		
Case 2	D		
Case 5	A		
Case 4	C		
Case 1	D		

EXAMPLE – PETRI NET MODEL



CHALLENGES

- Mining more difficult for large models
  - Not all possible combinations can be captured/detected
    - E.g., interleaving of 10 tasks or paths with very low probability
- Logs can contain noise:
  - Incorrect parts – human or technical errors
  - Incomplete parts – manual tasks or tasks handled by other organisational unit/system
  - Exception referral – rare or undesired events
    - E.g., Ordering of events reversed due to time pressure
    - E.g., Causally unrelated events happen next to each other
- Exploitation of all possible information present in log (event type, timestamp & attributes)
- Legal issues (e.g., legislation on privacy/protection of private data -> anonymization)

LOG DESCRIPTION

- XML-based log description format has been proposed
  - Used for recording workflow events in WFMS
  - Independent from the tool used to perform the monitoring
  - Can be used for log exchange – interoperability
  - Can lead to reducing implementation effort
  - Promotes use of process mining in multiple contexts

LOG DESCRIPTION

<!ELEMENT WorkFlow_log (source?, process+)>

<!ELEMENT source EMPTY>

<!ATTLIST source program (staffware | inconcert | pnet | IBM_MQ | other) #REQUIRED>

<!ELEMENT process (case*)>

<!ATTLIST process id ID #REQUIRED description CDATA “none” >

<!ELEMENT case (log_line*)>

<!ATTLIST case id ID #REQUIRED description CDATA “none” >

<!ELEMENT log_line (task_name, task_instance?, event?, date?, time?)>

<!ELEMENT task_name (#PCDATA)>

<!ELEMENT task_instance (#PCDATA)>

<!ELEMENT event EMPTY>

<!ATTLIST event kind (normal | schedule | start | withdraw | suspend | resume | abort | complete) #REQUIRED>

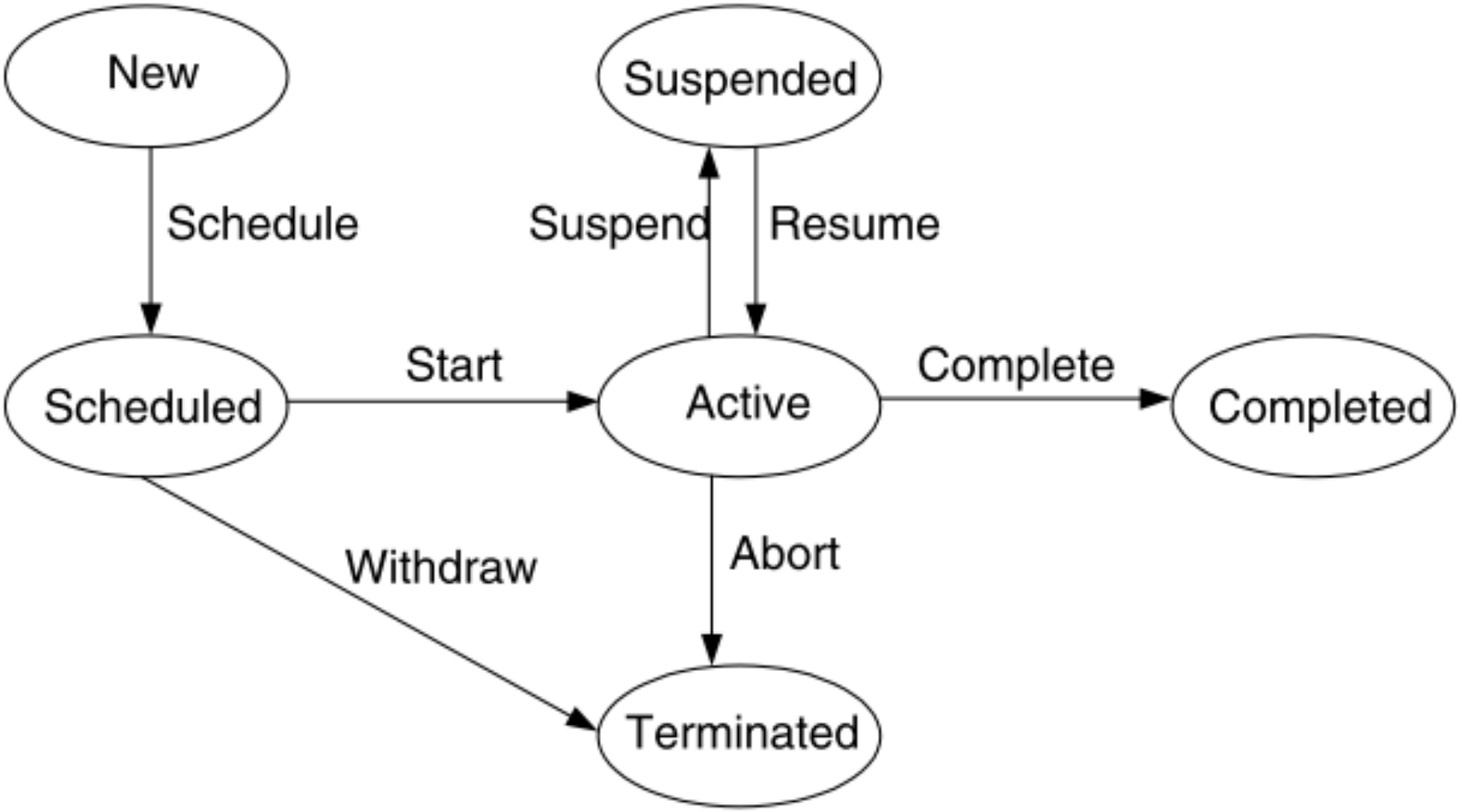
<!ELEMENT date (#PCDATA)>

<!ELEMENT time (#PCDATA)>

LOG DESCRIPTION

- Peculiarities about the schema:
  - Task instance uniquely identifies a particular task execution in a workflow
    - When many appearances occur in the workflow model, e.g., in a loop
  - 5 event types can be captured:
    - Schedule: event ready for execution/enabled
    - Withdraw: task deleted from working list and its state becomes terminated
    - Active: task is started and removed from working list
    - Abort: task cannot finish execution/complete and its state becomes terminated
    - Suspend/Resume: task is suspended / each execution resumes
  - In some cases, a normal event is captured, mapping to the atomic execution of task (i.e., events schedule, start, complete occur)

EVENT TYPES


New
Suspended
Schedule
Suspend
Resume
Scheduled
Start
Active
Complete
Completed
Withdraw
Abort
Terminated

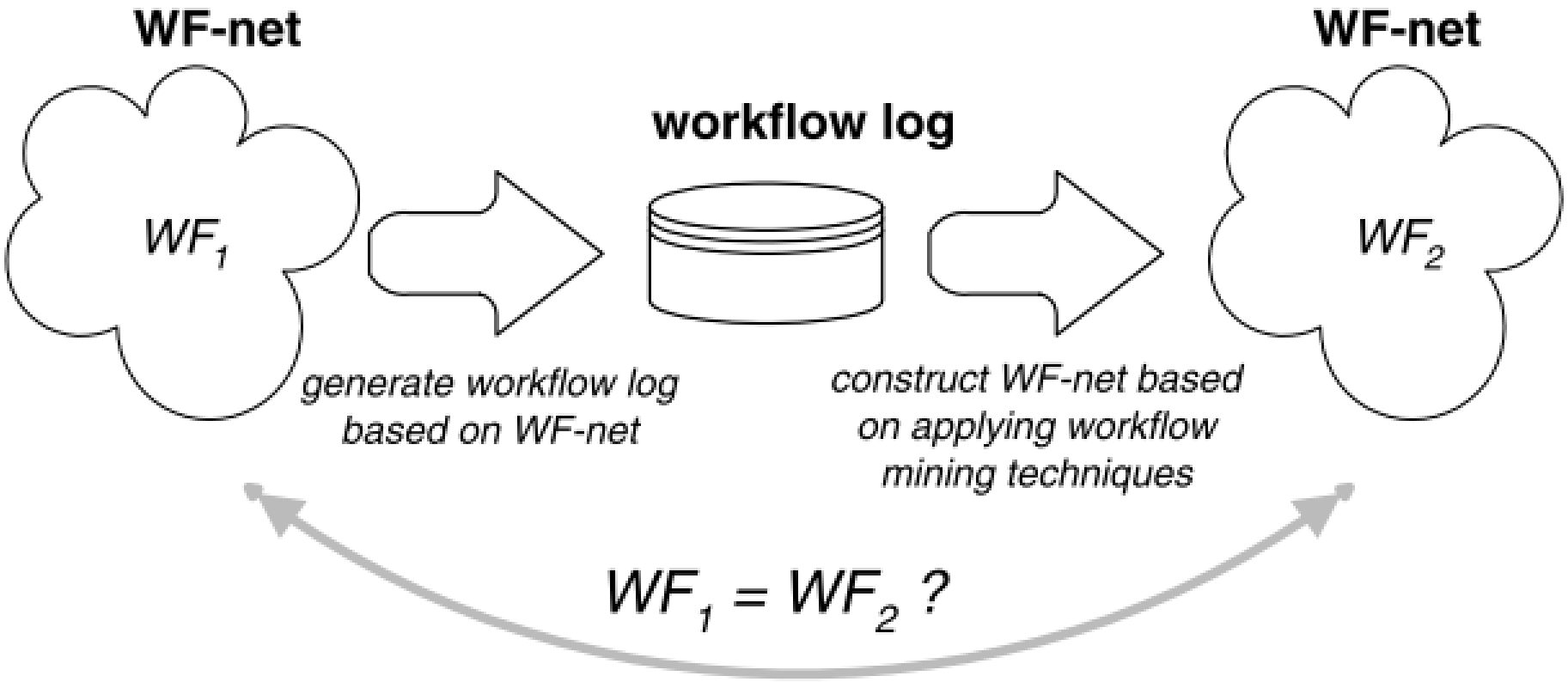
EVENT CAPTURING DIFFERENCES

- Some systems employ a more fine-grained level
  - Example – MQSeries FSM:
    - Task in error, expired, terminated by user, force-finished -> mapped to withdraw/abort
- Others consider a subset of events
  - Example – Staffware:
    - Schedule, Withdraw & Complete events are only captured
    - It also captures other events, such as Expired & Start (starting of a case)

WORKFLOW MODEL (RE-)DISCOVERY

- Special class of Petri-nets, called workflow-nets (WF-nets) can be employed
- Assumed that no noise exists and log is representative enough
- Goal: check which workflow model classes can be discovered through analysis/mining of log file
- Formally: Suppose initial model is captured through WF-net W1 and that a WF-net W2 is constructed through the log file. For which classes of WF-net, we have the case that $W1 \equiv W2$?

WORKFLOW MODEL (RE-)DISCOVERY

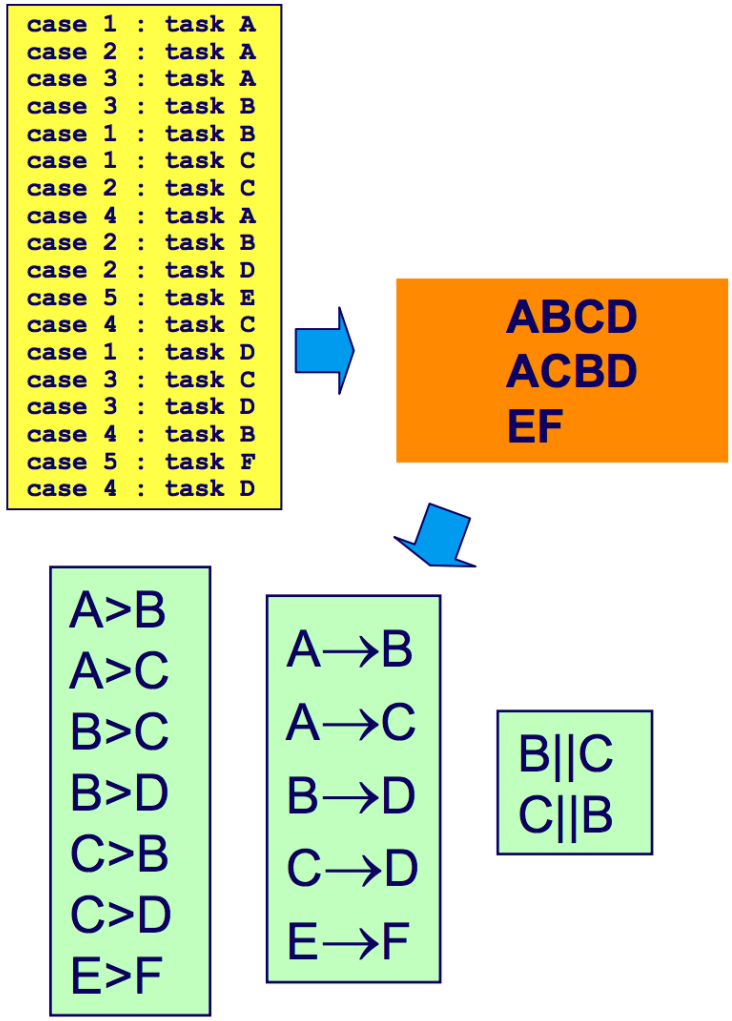


WORKFLOW MODEL (RE-)DISCOVERY

- The α algorithm has been proposed to deal with this
- Based on 4 ordering relations that can be discovered from the log (where *W* denotes workflow log, *T* set of tasks, a, b ε T):
  - $a >_w b$ ⇔ exists trace $\sigma=t_1,t_2,\ldots,t_{n-1}$ and $i \in \{1,2,\ldots,n-2\}$ st. $\sigma \in W$ and $t_i=a$ and $t_{i+1}=b$
  - $a \rightarrow_w b$ ⇔ $a >_w b$ and $b \not> _w a$
  - $a \#_w b$ ⇔ $a \not> _w b$ and $b \not> _w a$
  - $a \|_w b$ ⇔ $a >_w b$ and $b >_w a$
- Classical limits in PN theory apply in this case
  - Addressing non-free-choice constructs
- Many problems undecidable in PNs are decidable for free-choice nets

WORKFLOW MODEL (RE-)DISCOVERY

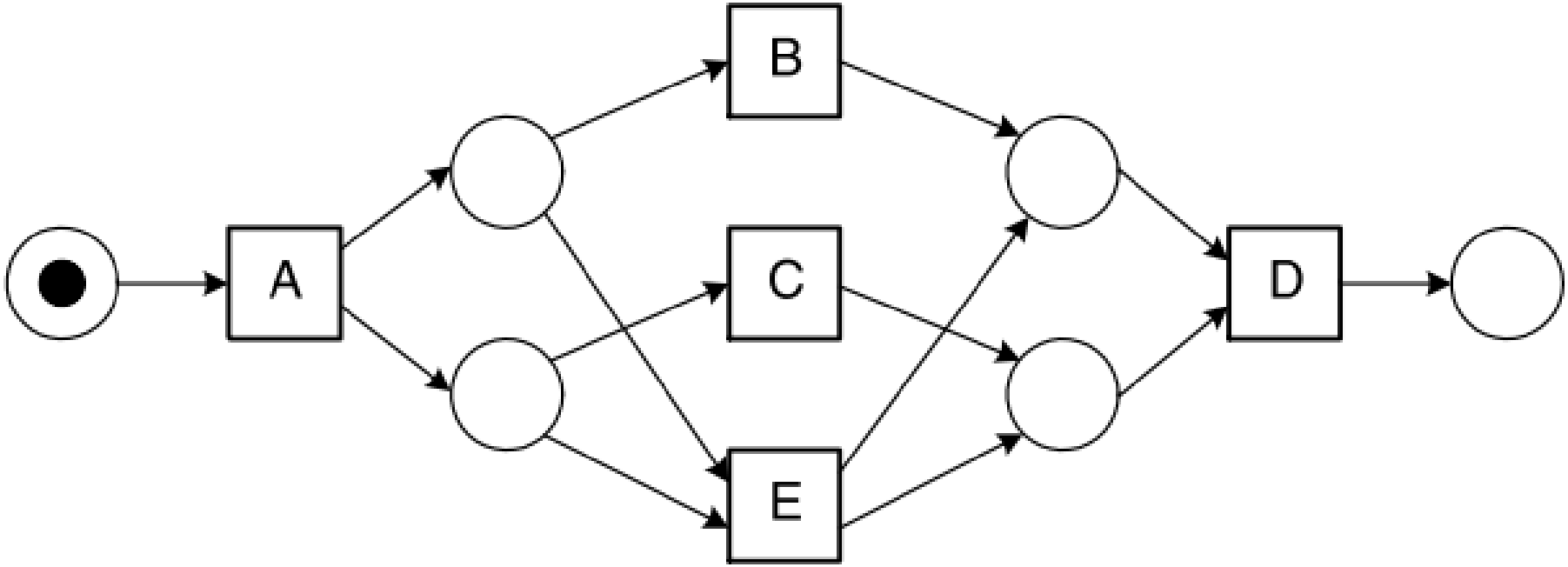
- **Direct succession**: x>y iff for some case x is directly followed by y.
- **Causality**: x → y iff x>y and not y>x.
- **Parallel**: x||y iff x>y and y>x
- **Choice**: x#y iff not x>y and not y>x.



WORKFLOW MODEL (RE-)DISCOVERY

- α algorithm cannot detect some kinds of loops & multiple tasks with same name
- Multiple WF-nets can be discovered from one log
  - Two syntactically different WF-nets can have the same behaviour
- α algorithm creates the simplest/smallest possible WF-net
- It can mine timed workflow logs & calculate all performance metric types
- Tool support: EMiT (Enhanced Mining Tool)

STAFFWARE EXAMPLE – WF-NETS



NOISE & INCOMPLETE LOGS

- In practice, logs are incomplete & contain noise
- Leads to difficulty in assessing the kind of relation between tasks
  - E.g., error in log file wrt order between a and b (wrong case where b precedes a) leads to wrong conclusion that a $\|_w$ b and not a $>_w$ b
- Problem addressed by heuristic approaches
  - Also attempt to solve issues with α algorithm (loops & non-free-choice)

HEURISTIC APPROACHES

- 3 main steps are involved:
  - Dependency/frequency table (D/F table) construction
  - Mining basic relations (R-table) from D/F table
  - Reconstruction of WF-net model from R-table
- D/F table construction:
  - For each task A, generate the following information
    - Overall task frequency (#A)
    - Frequency of A preceded directly by B (#B > A)
    - Frequency of A followed directly by B (#A > B)
    - Frequency of A directly or indirectly preceded by B but before next appearance of B (#B >>> A)
    - Frequency of A directly or indirectly followed by B but before next appearance of A (#A >>> B)
    - Metric measuring strength of causality between A and B (#A → B)

Heuristic = involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods

Source: https://www.merriam-webster.com/dictionary/heuristic

D/F TABLE CONSTRUCTION

- Last metric rationale – Discover whether:
  - If always A precedes (directly or indirectly) B or B precedes (directly or indirectly) A
- Calculation:
  - In a specific trace, if n intermediate tasks are between A and B, then increase #A → B counter with δ^n where $\delta \in [0.0, 1.0]$. Do symmetrically similar for the #B → A counter. Decrease by δ^n if opposite case occurs. Usual value for δ is 0.8.
  - We stop searching when next appearance of A or B takes place
  - Finally, divide each counter with minimum frequency of A or B, i.e., min(#A, #B)

BASIC RELATION TABLE CONSTRUCTION

$$\text{if } \left(\#A \rightarrow B \geq N \wedge \#A > B \geq \theta \wedge \#B > A \leq \theta\right) \Rightarrow A \rightarrow_W B$$

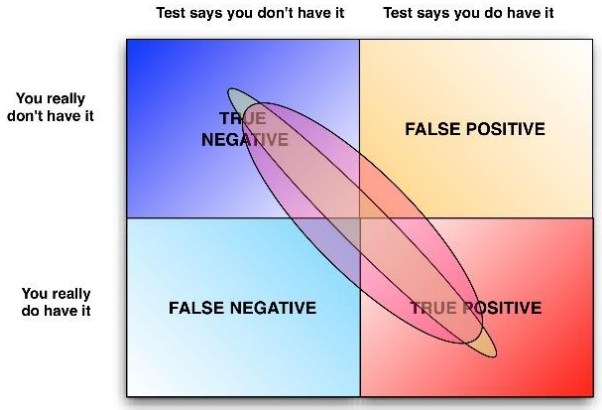
- Use heuristic rules to assess the type of relation between A and B
  - Example:
    - N is noise factor (typically 0.05 but can increase if expected noise is higher)
    - θ is set to 1 + Round(N ● #L) / #T where #L is the trace number in log and #T is the number of (unique) tasks

WF-NET RECONSTRUCTION FROM R-TABLE

- Use α algorithm to perform the reconstruction
- Can employ an extra step to check based on task frequency whether number of task occurrences is consistent for resulting PN
- Disadvantage:
  - Use of threshold value -> how do you know exactly the noise in a log?
  - Addressed in two alternative ways:
    - Use machine learning techniques to derive the threshold
    - Formulate other rules and measurements without thresholds
- Tool support: Little Thumb

QUALITY OF MINED WORKFLOW MODEL

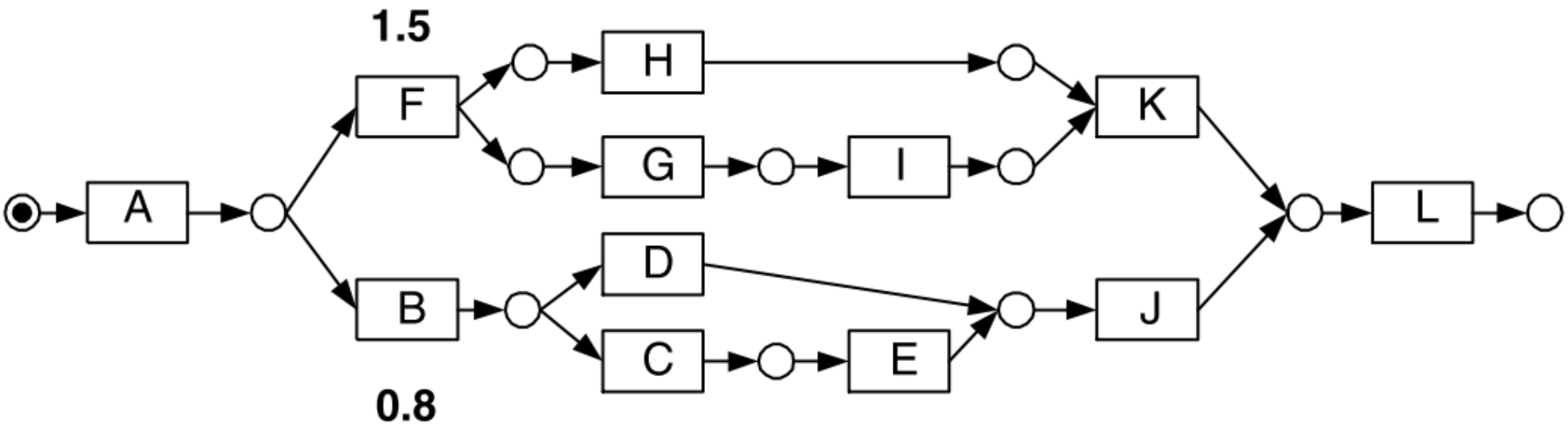
- Criterion for quality
  - Consistency of mined model with the traces in the log
- Common & simple check:
  - Execute all traces against the model
    - If a trace cannot be executed, then a discrepancy exists
    - Problem: noisy traces lead to false negatives
- Other way:
  - Directly compare original model with reconstructed one
    - If original model is known
    - Quality related to amount of correct relations mined
    - Tool Support: ExperDiTo for performing an experimental evaluation of quality based on specific input



EVALUATION OF 2ND METHOD

- Create testing input through varying typical workflow parameters that usually differ across different workflows:
  - Total number of event types
  - Amount of available information in log
  - Amount of noise
  - Imbalance in OR-Splits and AND-splits (e.g., different priorities in tasks of an OR-Split)
- Evaluation Results:
  - More noise, less balance & less cases negatively affect quality
  - Number of events types does not influence quality wrt the causal relation discovery. For complex PNs, causal relations are more difficult to detect
  - Difficult to reach conclusion on performance of exclusive/parallel relation discovery -> depends on causal relation discovery.

EXAMPLE – UNBALANCED PETRI NET



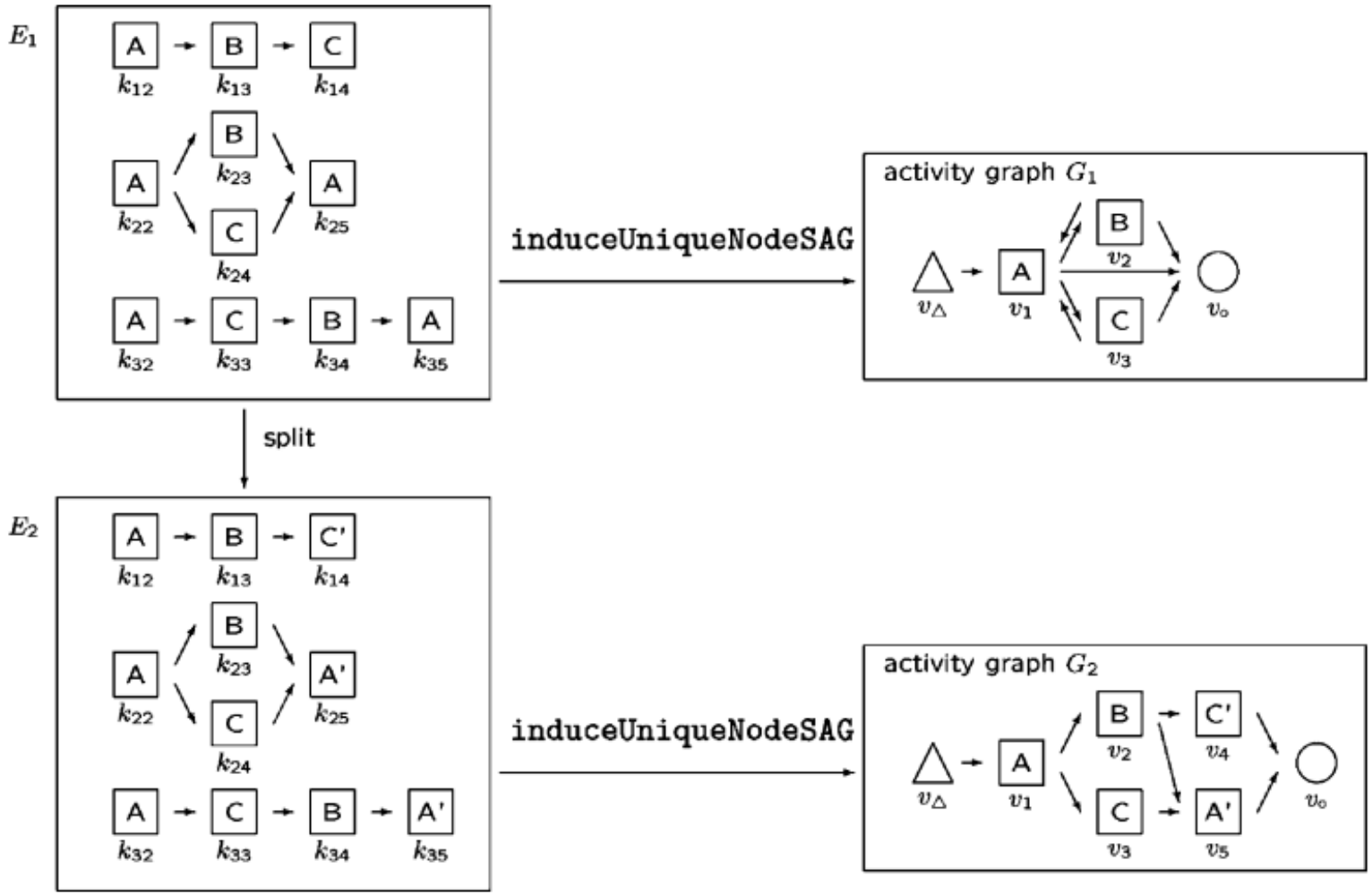
DUPLICATE TASK HANDLING

- For some processes (e.g., part release for passenger car development), tasks can have the same name
  - Could remedy this through renaming but the model might not be known in advance (that's why process mining is employed)
- Solution comprises two steps:
  - Induction
  - Transformation
- Tool Support: InWoLvE (Inductive Workflow Learning via Examples)
  - Implements also two other induction steps restricted to sequential workflows
  - Interfaces with ADONIS tool for workflow log and process model interchange

INDUCTION STEP

- Create a Stochastic Task Graph (STG) from log
- Search procedure employed:
  - Based on machine learning & grammatical inference
  - Search for mapping between tasks in logs and tasks in model
  - Search space is a lattice of such mappings
    - Partially ordered mappings
    - Lattice limited by top (all log tasks with equal name to one task in model) & bottom elements (opposite case where mapping is bijection between log tasks and model tasks)
  - Search performed from top to bottom. Split operator used to create more specific/lower elements (two model tasks used to group respective log tasks in two clusters)
  - Search guided by log likelihood of the STG per stochastic sample. n workflow instances sharing the same ordering of tasks are considered as n different cases.

SPLIT EXAMPLE



TRANSFORMATION STEP

- STG is transformed into a block-structure workflow model in ADONIS format
  - STG does not explicitly distinguish between alternative & parallel routing
- Comprises 3 main sub-steps:
  - Analyze synchronisation structures of workflow instances in log
  - Generate synchronisation structure of workflow model
  - Generate final model

BLOCK-STRUCTURED WORKFLOW MINING

- Differences with previous approaches:
  - Only block structure workflow patterns are considered
  - Rewriting rather than graph-based techniques are employed
  - Mine complete and minimal models
    - Only but all logged cases are covered
    - Stronger notion of completeness is embraced
- Block structure models:
  - Constructed from nested blocks
  - Can be distinguished into operators & constants:
    - Operators build process flow
    - Constants are tasks or sub-workflows
  - A model is a tree with operands as leaves

BLOCK-STRUCTURED WORKFLOW MINING

- Block-structured models can be represented as terms sets
  - e.g., S(a,P(b,c)) where S is sequence, P is parallel and a,b are tasks
  - Each term is well-formed
  - An algebra can be specified comprising axioms for commutativity, distributivity, associativity and so on
    - Axioms used as basis for term re-writing systems
- Mining procedure comprises 5 main steps:
  1. Create a trace for each instance of a process from log. Trace comprises start and complete events in timely correct order. Traces are condensed into groups based on the sequence of start & complete events. Groups map to the process paths.
  2. Trace groups are used by a time-forward algorithm to generate initial model expressed in DNF form (one alternative operator involving all possible paths as blocks)

BLOCK-STRUCTURED WORKFLOW MINING

- Mining procedure comprises 5 main steps:

3. Address relations between tasks resulting from the random task order in log without a real precedence between them. These pseudo-precedence relations are removed. Such relations are identified through term-rewriting by transforming the model into a form enumerating all sequences of tasks inside parallel operators enclosed in the overall alternative. Search algorithm then performs the identification by discovering smallest subset of sequences completely explaining the blocks in the model. The resulting model is reversed to the initial DNF form.
4. The overall alternative is split and its parts are moved to that time point where decision cannot be postponed. Relies on transformation step based on distributivity axioms. It merges building blocks while shifting alternatives. Input model becomes condensed.

BLOCK-STRUCTURED WORKFLOW MINING

- Mining procedure comprises 5 main steps:
  5. Optional decision making step based on decision tree deduction (deduction on each model decision point). Input data are the workflow context per trace. Intermediate output is decision trees which are transformed into rules attached to alternative operators

- Tool support: Process Miner
  - Allows editing & exporting the model
  - Also contains simulation component

TOOL COMPARISON

Dimension	EMiT	Little Thumb	InWoLvE	Process Miner
Structure	Graph	Graph	Graph	Block
Time	yes	no	no	no
Basic parallelism	yes	yes	yes	yes
Non-free choice	no	no	no	no
Basic loops	yes	yes	yes	yes
Arbitrary loops	yes	yes	no	no
Hidden tasks	no	no	no	no
Duplicate tasks	no	no	yes	no
Noise	no	yes	yes	no

OPEN PROBLEMS

- Timing information not used to improve mining result
- Other information parts are also not exploited (e.g., data objects)
- Advanced routing constructs, like non-free choice, as well as hidden or duplicate tasks are not (totally) handled
- Noise handling needs advanced techniques than simple heuristics

RECOMMENDED READING

- W.M.P van der Aalst, B.F van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M Weijters. Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering 47 (2003), 237-267.
- https://www.youtube.com/watch?v=AQcgkhPaePM : Process Mining
- https://www.youtube.com/watch?v=hsdHZ5MJBNM : Process Mining
- https://www.youtube.com/watch?v=gR4IJsP9obI : Event Log