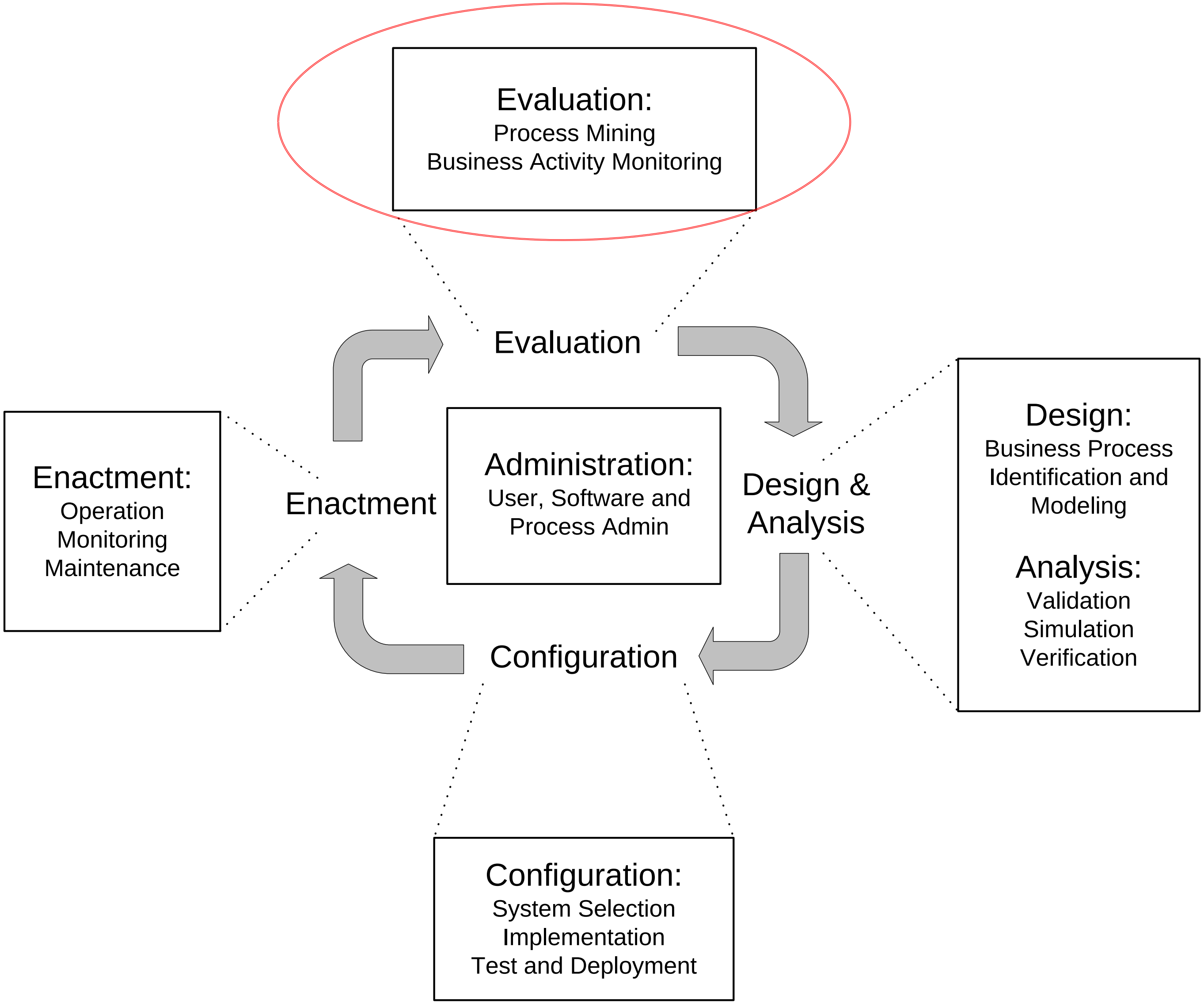
Topic 4: Process Mining



from M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2007

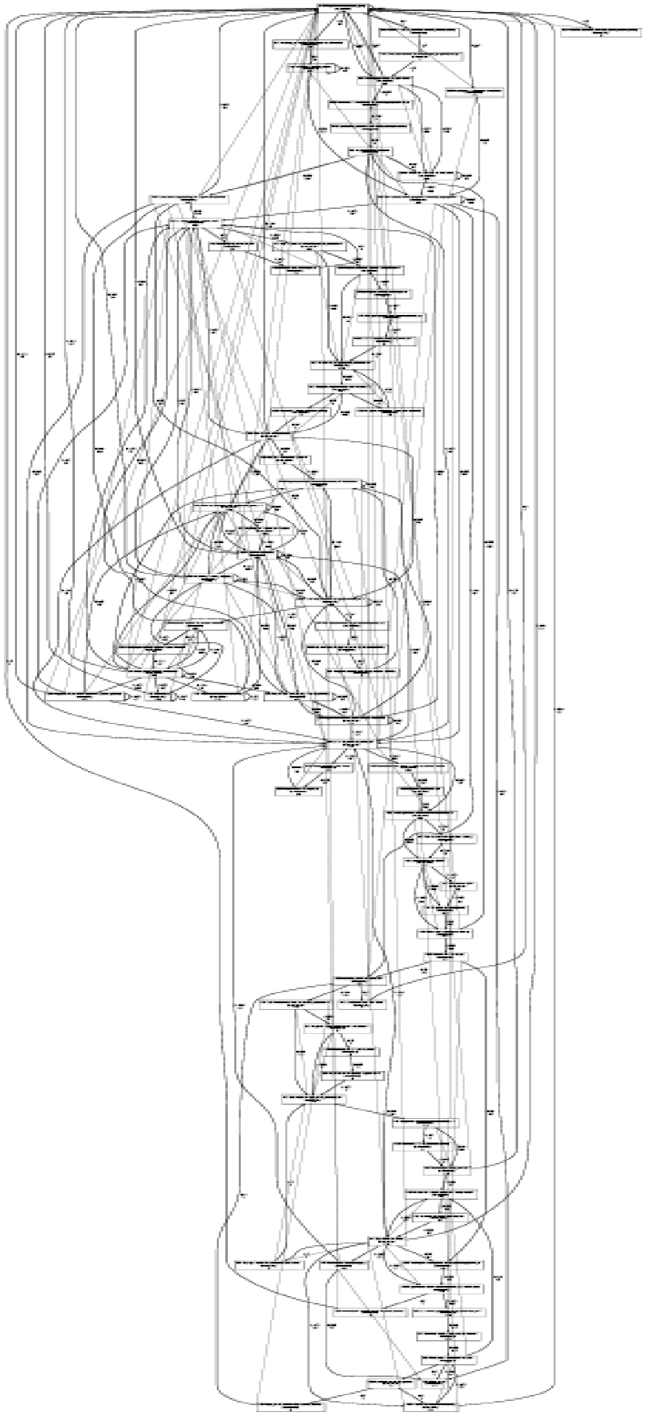
Structure

- Motivation
- Mining algorithms and their properties
  - α – Algorithm
  - α+(+) – Algorithm
  - Noise in execution logs
  - Outlook on other approaches
- Evaluation of created process models
  - Over-fitting / Under-fitting of Process models
  - Dimensions for the evaluation
  - Fitness Metric
  - Precision Metric

[Slides partially taken from Wil van der Aalst]

Motivation

- So far
  - Consideration of process models
  - Creation of process models as a creative act
- In this topic/chapter
  - Consideration of information that arose during operations of companies
  - So called: logs
- Log
  - Sequential series of log entries, record the events in the company



Log Entries

- Exemplery log entries
  - *Invoice check for invoice number 4567 completed on 12.11.2010 at 9:19:57*
  - *Function StoreCustomerData(„Müller“, c1987, „Bad Bentheim“) executed on 12.11.2010 at 9:22:24*
  - *Sending invoice for invoice number 4567 completed on 12.11.2010 at 9:23:18*
  - *Function ContactCustomer(c1987, PromoMailing) executed on 12.11.2010 at 9:24:10*
  - *Function StoreCustomerData(„Miller“, c1988, „Osnabrück“) executed on 12.11.2010 at 9:26:08*
  - *Invoice check for invoice number 4568 completed on 12.11.2010 at 9:26:38*
  - *Function ContactCustomer(c1988, PromoMailing) executed on 12.11.2010 at 9:27:32*

Logs Conceal Information

- Logs hold valuable information that they can answer questions
  - How many process instances were enacted?
  - Are there any recurring patterns in the execution of activities?
  - Can process models be derived from the data?
  - Which paths in these models are taken as frequently?
  - Are there any paths that were never executed?
- Process Mining
  - Area of research that focuses on these issues
  - Important parts: Process Discovery and Process Conformance

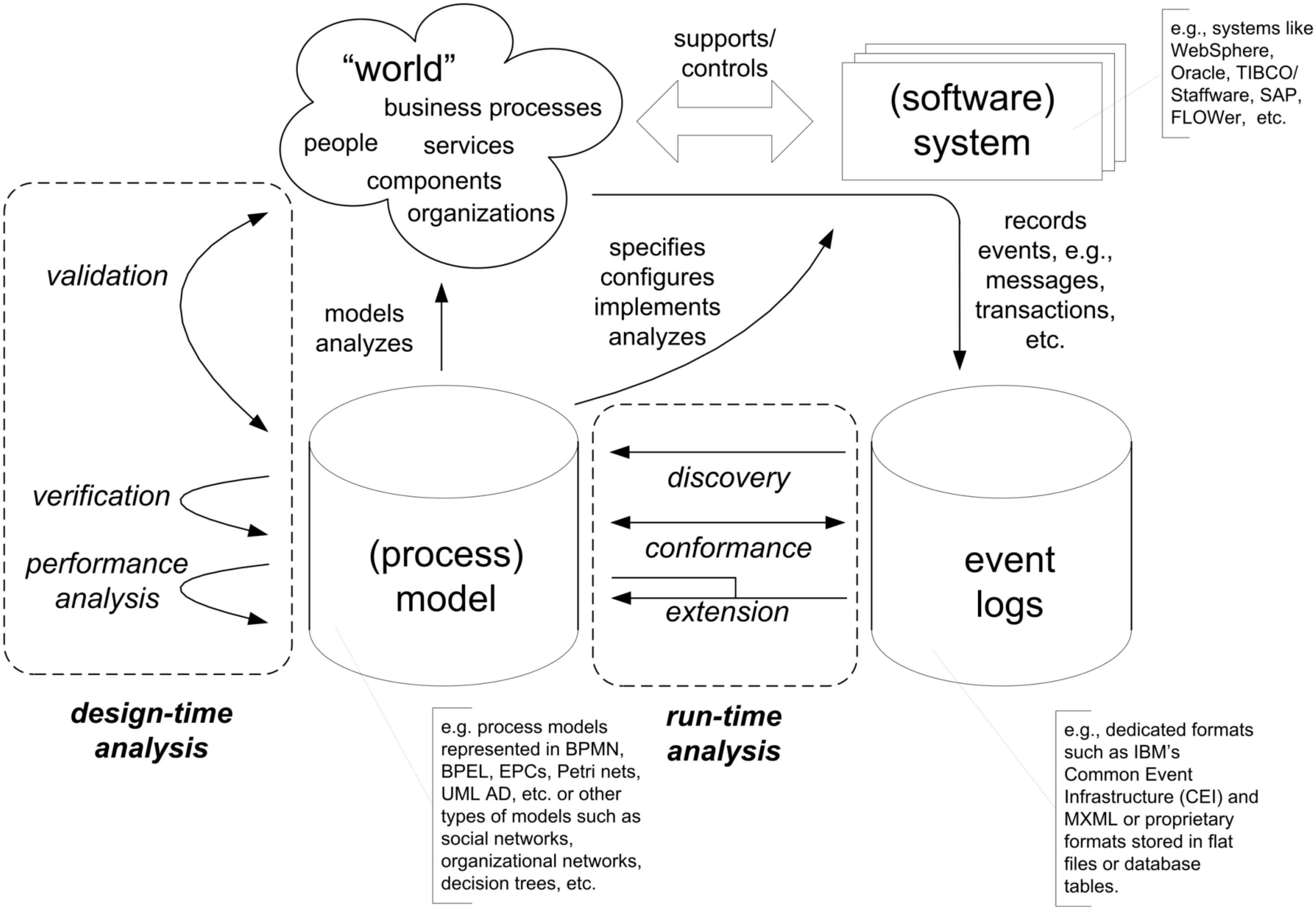
Motivation

- Process Discovery
  - A method to detect process models based on execution logs
  - Input: execution logs, ordered lists of activities, with a time stamp and caseID
  - Output: a process model that could have generated the execution log
- Hint
  - CaseID is frequently not directly present in the data and must be determined by pre-processing!

Motivation

- Process Conformance
  - Method for analyzing the relationships of logs and process models
    - Input: Execution logs and process model(s)
    - Output: Knowledge about their relationships (*fitting*)
- Hint
  - Process mining is versatile because many systems generate execution logs, not only Process-oriented Information Systems!
  - In principle, not only IT systems are used, also similar information collected!
  - Examples: Laboratory notebook or logbook

Overall Picture



Execution Logs

- Assumptions
  - Execution logs define a total order of events. An event can be assigned to an activity and a process instance.
  - All events in an execution log represent process instances of the same process model.
- Hint
  - It might be the case that real logs contain events of process instances belonging to different process models.
  - Also preprocessing is required to isolate logs and group them by process model.
- Abstraction
  - Mining algorithms are usually based on abstraction of logs
  - We focus on the CaseID as well as the executed process activity

Execution logs

- Log format
  - (caseID, Activity)
- Example
  - *Check invoice for invoice number 4567 ended on 12/11/2010 at 9:19:57*
  - *Function StoreCustomerData(„Müller“, c1987, „Bad Bentheim“) executed on 12.11.2010 at 9:22:24*
  - *invoice sending for invoice number 4567 ended on 12/11/2010 at 9:23:18*
  - *Function ContactCustomer(c1987, PromoMailing) executed on 12.11.2010 at 9:24:10*
- *Resulting Log*
  - *(4567, Check Invoice), (c1987, StoreCustomerData), (4567, Send invoice), (c1987, ContactCustomer)*

Execution log

- Further abstraction
  - A‘s and B‘s
  - (case id, task id)
- Further Information
  - Event type, time, resource, data
  - Not used in the approaches we discuss
- Note
  - The execution of an activity is represented by an event in the log
  - Not: Start / End- events for the activities

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

Process Discovery Algorithms

- Simplest algorithm: α – Algorithm
  - Relatively simple, certain properties can be proved
  - Sensitive to noise, therefore not suitable for real data
    - Noise refers to incorrectly written logs
- Thereupon: α+(+) – Algorithm
  - α+ und α++ as extensions for des α – algorithm to discover further process structures
  - Also sensitive to *Noise*
- Finally: Prospects for further methods, which can deal with noise

Definitions

- Let T be a set of tasks and T^* be the set of all sequnces or arbitrary lengths over T, it shall:
  - *$\sigma \in T^*$ is called an execution sequence, if all activities in σ belong to the same process instance*
  - *$W \subseteq T^*$ is called workflow log*
- Assumptions
  - In a process model each activity occurs at most once
  - Any direct neighborhood relation between activities is observed at least once

Execution log

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

Execution log

Execution sequences:

Case 1: ABCD

Case 2: ACBD

Case 3: ABCD

Case 4: ACBD

Case 5: EF

Corresponding execution log:

W = {ABCD, ACBD, EF}

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

Ordering relations

Log-based Ordering relations for a pair of activities $a, b \in T$ in a workflow log:

- Direct follower

$a >_w b$ are in execution seqeunce iff *b* directly follows a.

- Causality

$a \rightarrow_w b$ iff $a >_w b$ but not $b >_w a$

- Parallelism

$a \parallel_w b$ iff $a >_w b$ and $b >_w a$

- Exclusiveness

$a \#_w b$ iff not $a >_w b$ and not $b >_w a$

  - *Activity pairs that never follow each other*

Analysis of Worflow log

- *W* = {ABCD, ACBD, EF}
  - Direct sequence
  - Causaility
  - Parallelsim

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

Analysis of Worflow log

W = {ABCD, ACBD, EF}

- Direct sequence
- Causaility
- Parallelsim

1)

A>B
A>C
B>C
B>D
C>B
C>D
E>F

2)

A→B
A→C
B→D
C→D
E→F

3)

B||C
C||B

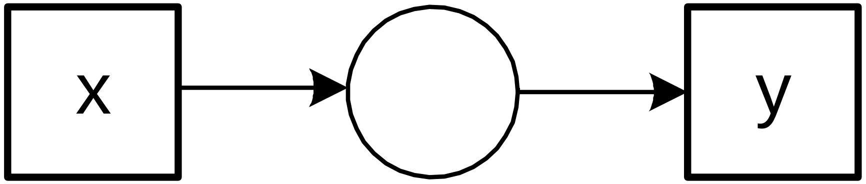
```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

α-Algorithm

- Idea
  - Use ordering relationships to generate a workflow-net, so that all generated execution sequences are respected.
- Concrete
  - From any order relation a Petri net fragment is derived, which can produce the corresponding order between activities.

α-Algorithm

- Idea 1

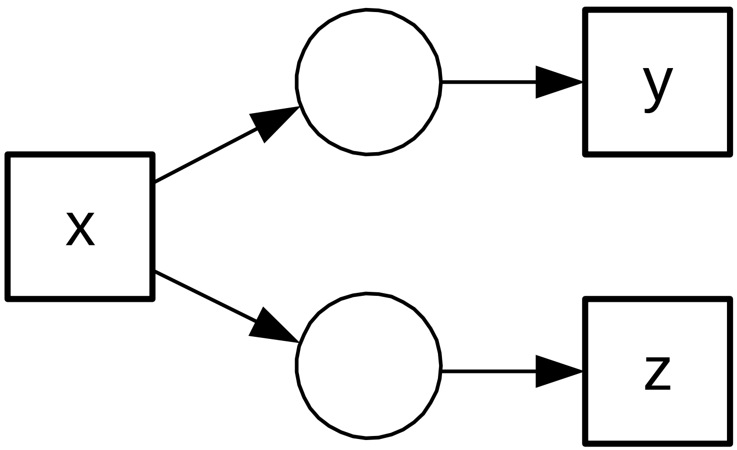


$$\Leftrightarrow$$

$$x \rightarrow y$$

α-Algorithm

- Idea 2

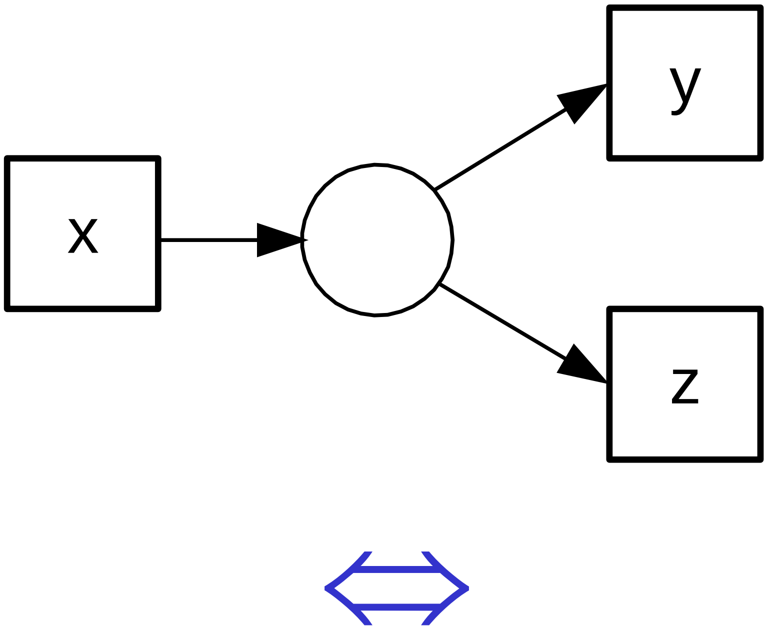


$$\Leftrightarrow$$

$$x \rightarrow y,\ x \rightarrow z \text{ and } y \parallel z$$

α-Algorithm

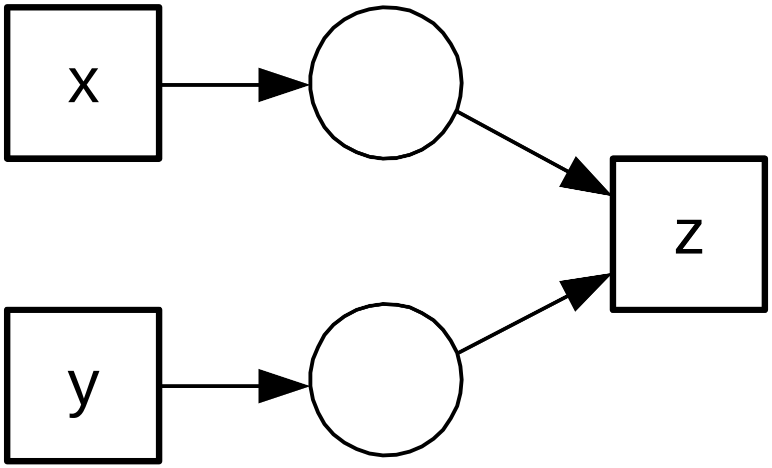
- Idea 3



$$x \rightarrow y,\ x \rightarrow z \text{ and } y \# z$$

α-Algorithm

- Idea 4

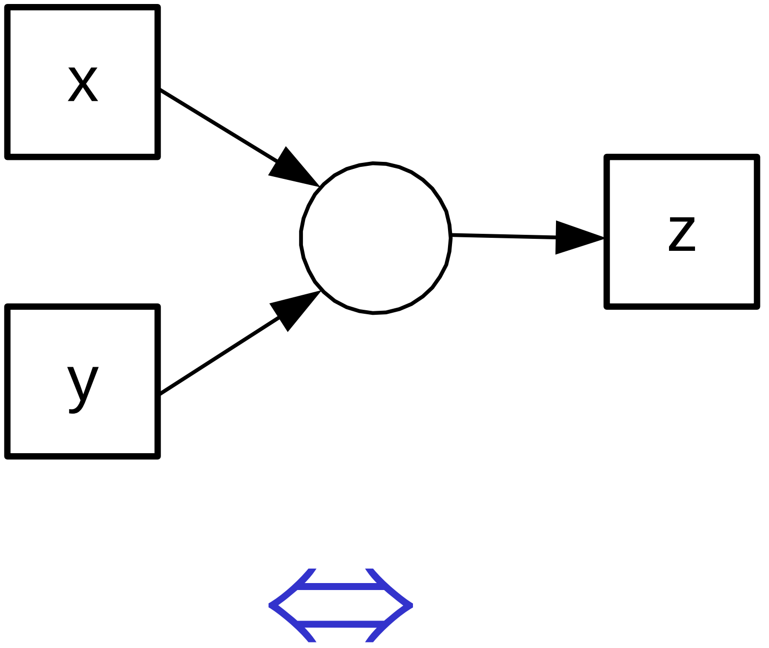


$$\Leftrightarrow$$

$$x \rightarrow z,\ y \rightarrow z \text{ and } x \parallel y$$

α-Algorithm

- Idea 5



$$x \rightarrow z,\ y \rightarrow z \text{ and } x \# y$$

α-Algorithm

Let W be a workflow log over T. $\alpha(W)$ defined as follows:

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{ i_W, o_W \}$,
7. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$,
8. $\alpha(W) = (P_W, T_W, F_W)$.

α-Algorithm

Let W be a workflow log over T. $\alpha(W)$ defined as follows:

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y$
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B)$ …
   $\{ (p_{(A,B)},b) \mid (A,B) \in$ …
   $\{ (i_W,t) \mid t \in T_I\} \cup \{$ …
8. $\alpha(W) = (P_W,T_W,F_W)$.

Result is a WF-Net:

P_W : Set of places

T_W : Set of transitions

F_W : Set of flow relation

α-Algorithm

All execution sequences are examined to determine the set Tw.

Let W be a workflow log over T … t

$\in T \mid \exists_{\sigma \in W}\, t \in \sigma\}$,

1. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma) \}$,
2. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
3. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
4. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
5. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
6. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup$
   $\{ (p_{(A,B)},b) \mid (A,B) \in Y_W \wedge b \in B \} \cup$
   $\{ (i_W,t) \mid t \in T_I\} \cup \{ (t,o_W) \mid t \in T_O\}$,
7. $\alpha(W) = (P_W, T_W, F_W)$.

$W = \{ABCD, ACBD, EF\}$

$T_W = \{A, B, C, D, E, F\}$

α-Algorithm

first(σ) [*last*(σ)] denote the first[last] transition in the execution sequence σ.

T_I [T_O] is the set of all initial[final] transitions.

Let *W* be a workflow log over *T*

$T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,

1. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma)$
2. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
3. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
4. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
5. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{ i_W, o_W \}$,
6. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge$
   $\{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge$ I
   $\{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W)$
7. $\alpha(W) = (P_W, T_W, F_W)$.

$W = \{ABCD, ACBD, EF\}$

$T_W = \{A, B, C, D, E, F\}$

$T_I = \{A, E\}$

$T_O = \{D, F\}$

α-Algorithm

$T_I = \{A, E\}$

$T_O = \{D, F\}$

$\{ (i_W,t) \mid t \in T_I\} = \{(i_W,A), (i_W,E)\}$

$\{ (t,o_W) \mid t \in T_O\} = \{(D,o_W), (F,o_W)\}$

i_W is the initial place, o_W is the final place.

In step 7 they will be connected to the transitions in sets in T_I and T_O.

4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\ a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\ a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\ b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup$
   $\{ (p_{(A,B)},b) \mid (A,B) \in Y_W \wedge b \in B \} \cup$
   $\{ (i_W,t) \mid t \in T_I\} \cup \{ (t,o_W) \mid t \in T_O\}$,
8. $\alpha(W) = (P_W,T_W,F_W)$.

α-Algorithm

Let W be a workflow log over T. $\alpha($

All other places have the form Form $p_{(A,B)}$, in which A and B are the predecessor or the successor transitions.

A place is inserted between *a* and *b*, iff. $a \rightarrow_W b$.

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W} \, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W} \, t = first(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W} \, t = last(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B} \, a \rightarrow_W b \wedge \forall_{a1,a2 \in A} \, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B} \, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup$
   $\{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge b \in B \} \cup$
   $\{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$,
8. $\alpha(W) = (P_W, T_W, F_W)$.

α-Algorithm

Some places are combined in the case of XOR-splits / joins instead of AND-splits / joins. For this purpose, the relations X_W and Y_W are constructed.

Let *W* be a workflow log over *T*. α(

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = first(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = last(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$,
8. $\alpha(W) = (P_W, T_W, F_W)$.

α-Algorithm

Let W be a workflow log over T. α(

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma\}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,

$\in X \mid \forall_{(A',B') \in}$ },

$\mid (A,B) \in Y_W \}$

$\{ (a,p_{(A,B)}) \mid (A,B) \in Y$

$(p_{(A,B)},b) \mid (A,B) \in Y_W$

$(i_W,t) \mid t \in T_I\} \cup \{ (t,o$

$) = (P_W,T_W,F_W)$.

(A,B) ∈ X_W if causality of each element in A for each element in B exists and the elements in A and B were never observed in the direct sequence.

$A \rightarrow B$
$A \rightarrow C$
$B \rightarrow D$
$C \rightarrow D$
$E \rightarrow F$

$B \| C$
$C \| B$

$T_W = \{A, B, C, D, E, F\}$

$X_W = \{ (\{A\},\{B\}), (\{A\},\{C\}), (\{B\},\{D\}), (\{C\},\{D\}), (\{E\},\{F\})\}$

Hint: Because of B||C the tuples ({A}, {B,C}) and ({B,C},{D}) are not in X_W

α-Algorithm

Y_W = { ({A},{B}), ({A},{C}), ({B},{D}), ({C},{D}), ({E},{F})}

Hint: $Y_W = X_W$ because
$\forall (A,B) \in X_W : |A| = 1 \wedge |B| = 1$

$\alpha(W)$ defined as follows:

Y_W is derived from X_W by taking the largest elements with respect to containment

2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X_W \mid \forall_{(A',B') \in X_W} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$,
8. $\alpha(W) = (P_W, T_W, F_W)$.

Y_W = { ({A},{B}), ({A},{C}), ({B},{D}), ({C},{D}), ({E},{F})}

P_W = { $p_{(\{A\},\{B\})}$, $p_{(\{A\},\{C\})}$, $p_{(\{B\},\{D\})}$, $p_{(\{C\},\{D\})}$, $p_{(\{E\},\{F\})}$, i_W , o_W}

F_W = {(A,$p_{(\{A\},\{B\})}$), (A,$p_{(\{A\},\{C\})}$), (B,$p_{(\{B\},\{D\})}$), (C,$p_{(\{C\},\{D\})}$), (E,$p_{(\{E\},\{F\})}$), ($p_{(\{A\},\{B\})}$,B), ($p_{(\{A\},\{C\})}$,C), ($p_{(\{B\},\{D\})}$,D), ($p_{(\{C\},\{D\})}$,D), ($p_{(\{E\},\{F\})}$,F), (i_W,A), (i_W,E), (D, o_W), (F, o_W)}

(W) defined as follows:

Y_W is used to produce places and edges accordingly.

,

$\wedge \ \forall_{a \in A} \forall_{b \in B}\ a \rightarrow_W b \ \wedge$

$\#_W b_2$ },

5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,

6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,

7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup$
$\{ (p_{(A,B)},b) \mid (A,B) \in Y_W \wedge b \in B \} \cup$
$\{ (i_W,t) \mid t \in T_I\} \cup \{ (t,o_W) \mid t \in T_O\}$,

8. $\alpha(W) = (P_W,T_W,F_W)$.

α-Algorithm Example

Execution log:

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```

$= \{ABCD, A \quad BD, EF\}$

$T_W = \{A, B, \quad , D, E, F\}$

$T_I = \{A, E\}$

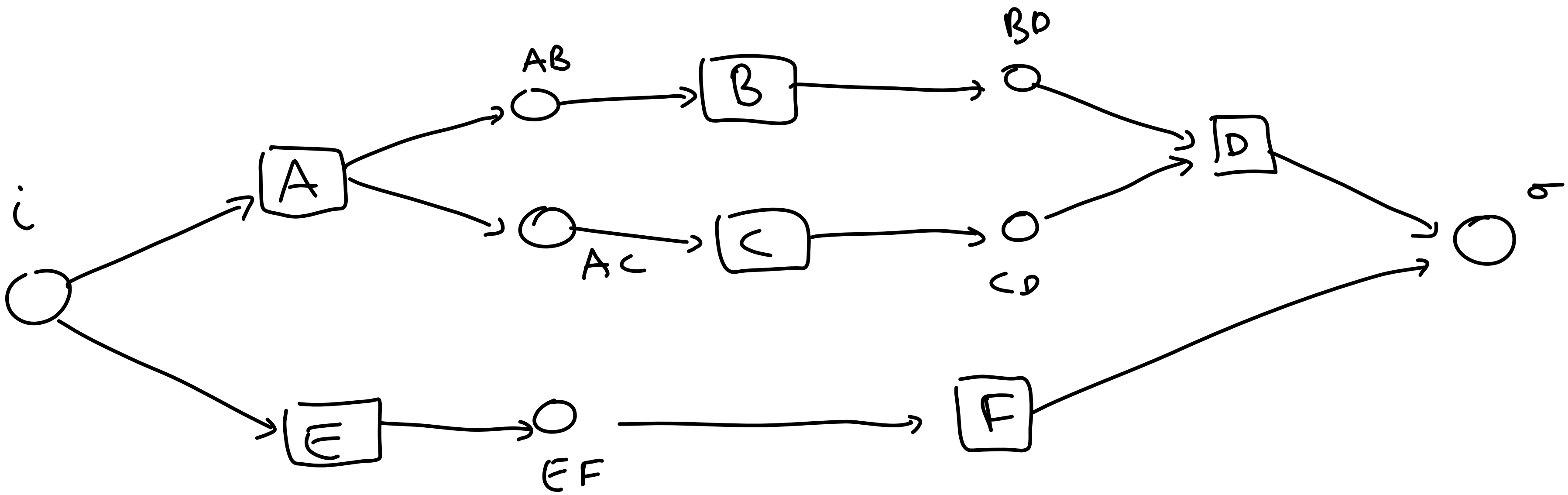
$T_O = \{D, F\}$

$X_W = \{ \quad \{A\}, \quad B\}), (\{A\}, \{C\}), \quad \{B\}, \quad D\}),$
$\quad C\}, \quad D\}), \quad (\{E\}, \{F\}) \}$

$Y_W = X_W$, wg $|A| = |B| = 1, \forall (A,B) \in X_W$

$P_W = \{ p_{AB}, \; p_{AC}, \quad p_{CD} \; p_{EF} \} \cup \{ i_W, o_W \}$

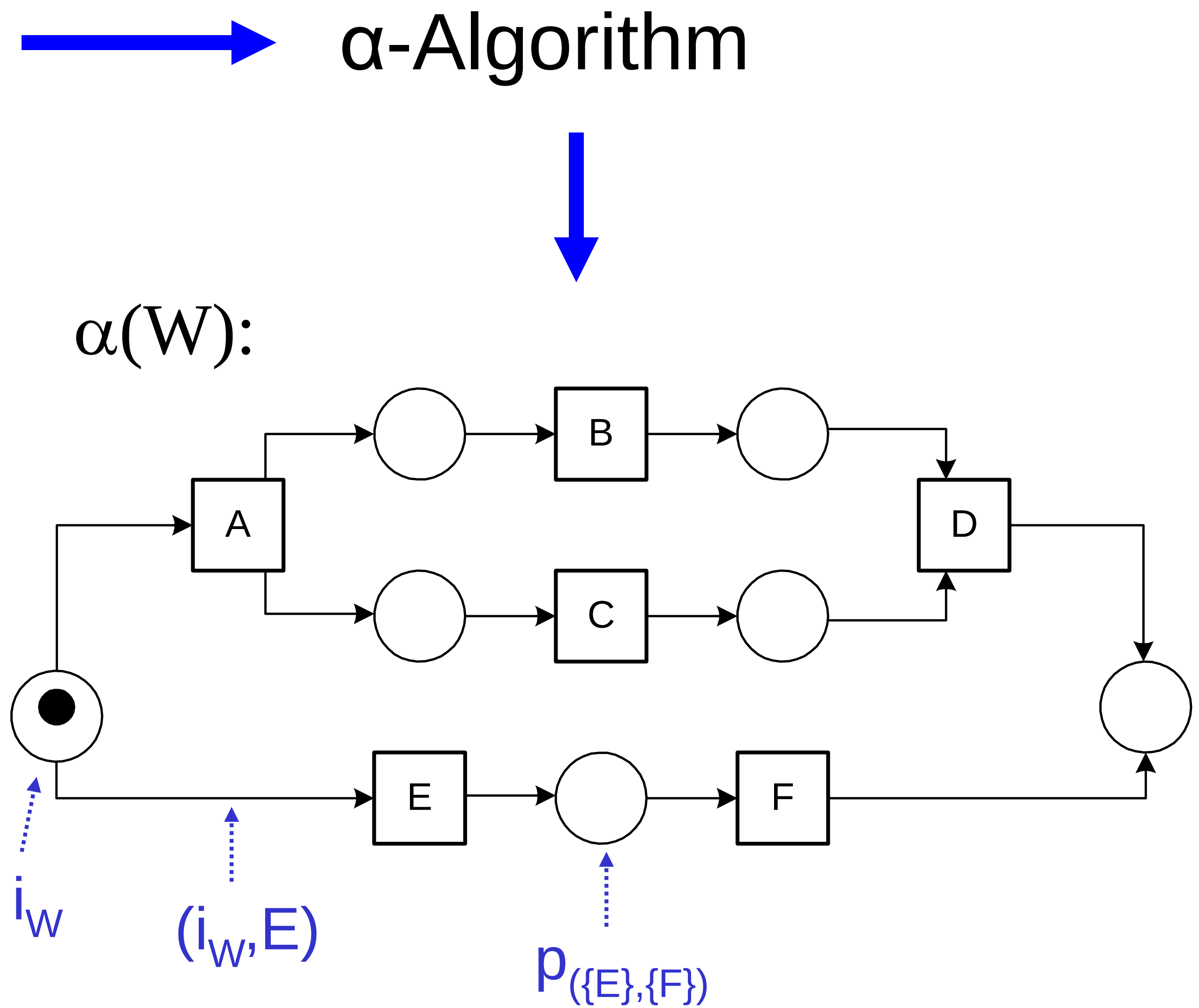
$T_I = \{ A \;\; E \}, \quad T_O = \{ \;\; , F \}$



α-Algorithm Example

Execution log:

```
case 1 : task A
case 2 : task A
case 3 : task A
case 3 : task B
case 1 : task B
case 1 : task C
case 2 : task C
case 4 : task A
case 2 : task B
case 2 : task D
case 5 : task E
case 4 : task C
case 1 : task D
case 3 : task C
case 3 : task D
case 4 : task B
case 5 : task F
case 4 : task D
```



Another Example

- Case 1: abdeh
- Case 2: adceg
- Case 3: acdefbdeg
- Case 4: adbeh
- Case 5: acdefdcefcdeh
- Case 6: acdeg

<table>
<tr><th></th><th>a</th><th>b</th><th>c</th><th>d</th><th>e</th><th>f</th><th>g</th><th>h</th></tr>
<tr><td>a</td><td>#</td><td>>, -></td><td>>, -></td><td>>, -></td><td>#</td><td>#</td><td>#</td><td>#</td></tr>
<tr><td>b</td><td></td><td>#</td><td>#</td><td>>, ||</td><td>>, -></td><td></td><td>#</td><td>#</td></tr>
<tr><td>c</td><td></td><td></td><td>#</td><td>>, ||</td><td>>, -></td><td></td><td></td><td></td></tr>
<tr><td>d</td><td></td><td>>, ||</td><td>>, ||</td><td>#</td><td>>, -></td><td></td><td>#</td><td>#</td></tr>
<tr><td>e</td><td>#</td><td></td><td></td><td></td><td>#</td><td>>, -></td><td>>, -></td><td>>, -></td></tr>
<tr><td>f</td><td></td><td>>, -></td><td>>, -></td><td>>, -></td><td></td><td></td><td></td><td></td></tr>
<tr><td>g</td><td>#</td><td>#</td><td>#</td><td>#</td><td></td><td>#</td><td>#</td><td>#</td></tr>
<tr><td>h</td><td>#</td><td>#</td><td>#</td><td>#</td><td></td><td>#</td><td></td><td></td></tr>
</table>

α-Algorithm

Let W be a workflow log over T. $\alpha(W)$ defined as follows:

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{first}(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \mathit{last}(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$,
8. $\alpha(W) = (P_W, T_W, F_W)$.

- Tw = { a, b, c, d, e, f, g, h}
- Ti = {a}
- To ={g, h}
- Xw = {({a}, {b}), ({a}, {c}), ({a}, {d}), ({b}, {e}), ({c},{e}), ({d},{e}), ({e}, {f}), ({e},{g}), ({e},{h}), ({f}, {b}), ({f},{c}), ({f}, {d}), ({a}, {b, c}), ({f}, { b, c}) ({ a, f}, {b}), ({ a, f},{c}), ({a, f}, { b, c}), ({a,f}, {d}), ({e}, { f, g, h}), ({b, c}, {e})}
- Yw = {({a,f}, { b, c}), ({a, f}, {d}), ({b,c}, {e}), ({d}, {e}), ({e}, {f, g, h})}