CSCI427/927 Systems Development

Alpha Algorithm in Process Mining

α-algorithm or **α-miner**

Acknowledgement: Materials on these slides are adapted from "Process-oriented System Analysis & Process mining"

Definitions

- Let T be a set of activities (Tasks) and T^* the set of all sequences of arbitrary length over T, then we have:
  - $\sigma \in T^*$ *is called execution sequence, if all activities* in *σ belong to the same process instance*
  - $W \subseteq T^*$ *is called execution log* (w*orkflow log)*
- Assumptions
  - In each process model, each activity appears at least once
  - Each direct neighbor relation between activities is represented at least once

Execution Logs

```
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 Logs

Execution sequences:

Case 1: ABCD

Case 2: ACBD

Case 3: ABCD

Case 4: ACBD

Case 5: EF

Resulting workflow 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
```

Order relations

Log based order relations for pairs of activities a, b $\in$ T in a workflow log W:

- Direct successor

  $a >_{\mathbf{w}} b$ i.e. in an execution sequence *b* directly follows a

- Causality

  $a \rightarrow_{\mathbf{w}} b$ i.e. $a >_w b$ and not $b >_w a$

- Concurrency

  $a \parallel_{\mathbf{w}} b$ i.e. $a >_w b$ and $b >_w a$

- Exclusiveness

  $a \#_{\mathbf{w}} b$ i.e. not $a >_w b$ and not $b >_w a$

  - *Activity pairs which never succeed each other*

Execution log analysis

- *W* = {ABCD, ACBD, EF}
  - Direct successor
  - Causality
  - Concurrency

```
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 analysis

- *W* = {ABCD, ACBD, EF}
  - Direct successor
  - Causality
  - Concurrency

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

- The idea is to utilize order relations for deriving a **workflow net** that is compliant with these relations

α-Algorithm

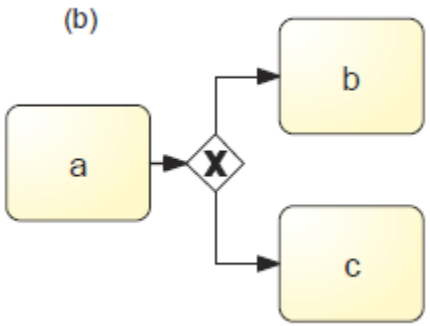
- Idea (a)

$$\Leftrightarrow$$

$$a \rightarrow b$$

α-Algorithm

- Idea (b)

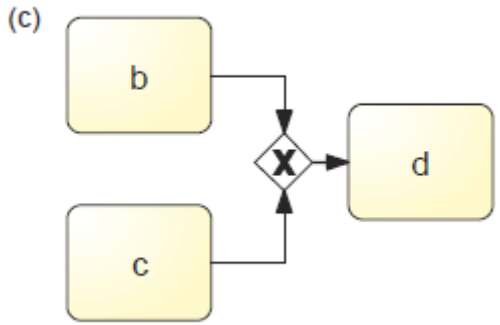


$\Leftrightarrow$

$a \rightarrow b,\ a \rightarrow c$ and $b \ \# \ c$

α-Algorithm

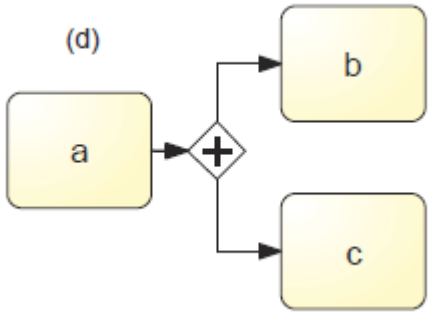
- Idea (c)



$$b \rightarrow d,\ c \rightarrow d \text{ and } b \ \#\ c$$

α-Algorithm

- Idea (d)

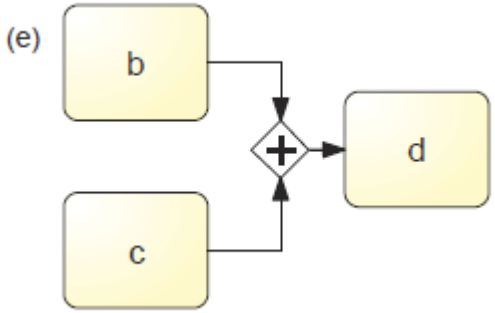


$\Leftrightarrow$

$a \rightarrow b$, $a \rightarrow c$ and $b \,||\, c$

α-Algorithm

- Idea (e)



$\Leftrightarrow$

$b \rightarrow d,\ c \rightarrow d$ and $b \,||\, c$

The Alpha-Algorithm (simplified)

1. Identify the set of **all tasks** in the log as T_W.
2. Identify the set of all tasks that have been observed as the **first task** in some case as T_I.
3. Identify the set of all tasks that have been observed as the **last task** in some case as T_O.
4. Identify the set of all connections to be potentially represented in the process model as a set X_W. Add the following elements to X_W:
    a. Pattern (a): all pairs for which hold $a \rightarrow b$.
    b. Pattern (b): all triples for which hold $a \rightarrow (b \# c)$.
    c. Pattern (c): all triples for which hold $(b \# c) \rightarrow d$.

- Note that triples for which Pattern (d) $a \rightarrow (b \| c)$ or Pattern (e) $(b \| c) \rightarrow d$ hold are not included in X_W.

The Alpha-Algorithm (cont.)

5. Construct the set Y_W as a subset of X_W by:
   a. Eliminating $a \rightarrow b$ and $a \rightarrow c$ if there exists some $a \rightarrow (b \# c)$.
   b. Eliminating $b \rightarrow c$ and $b \rightarrow d$ if there exists some $(b \# c) \rightarrow d$.

6. Connect start and end events in the following way:
   a. If there are multiple tasks in the set T_I of first tasks, then draw a start event leading to an XOR-split, which connects to every task in T_I. Otherwise, directly connect the start event with the only first task.

   b. For each task in the set T_O of last tasks, add an end event and draw an arc from the task to the end event.

The Alpha-Algorithm (cont.)

7. Construct the flow arcs in the following way:
   a. Pattern (a): For each $a \rightarrow b$ in Y_W, draw an arc a to b.
   b. Pattern (b): For each $a \rightarrow (b \# c)$ in Y_W, draw an arc from a to an XOR-split, and from there to b and c.
   c. Pattern (c): For each $(b \# c) \rightarrow d$ in Y_W, draw an arc from b and c to an XOR-join, and from there to d.
   d. Pattern (d) and (e): If a task in the so constructed process model has multiple incoming or multiple outgoing arcs, bundle these arcs with an AND-split or AND-join, respectively.

8. Return the newly constructed process model.

α-Algorithm Example

```
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 Example

```
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
```