Process Mining - 02269

Lecture 2

Control flow discovery with the Alpha Algorithm

Andrea Burattin

The Context

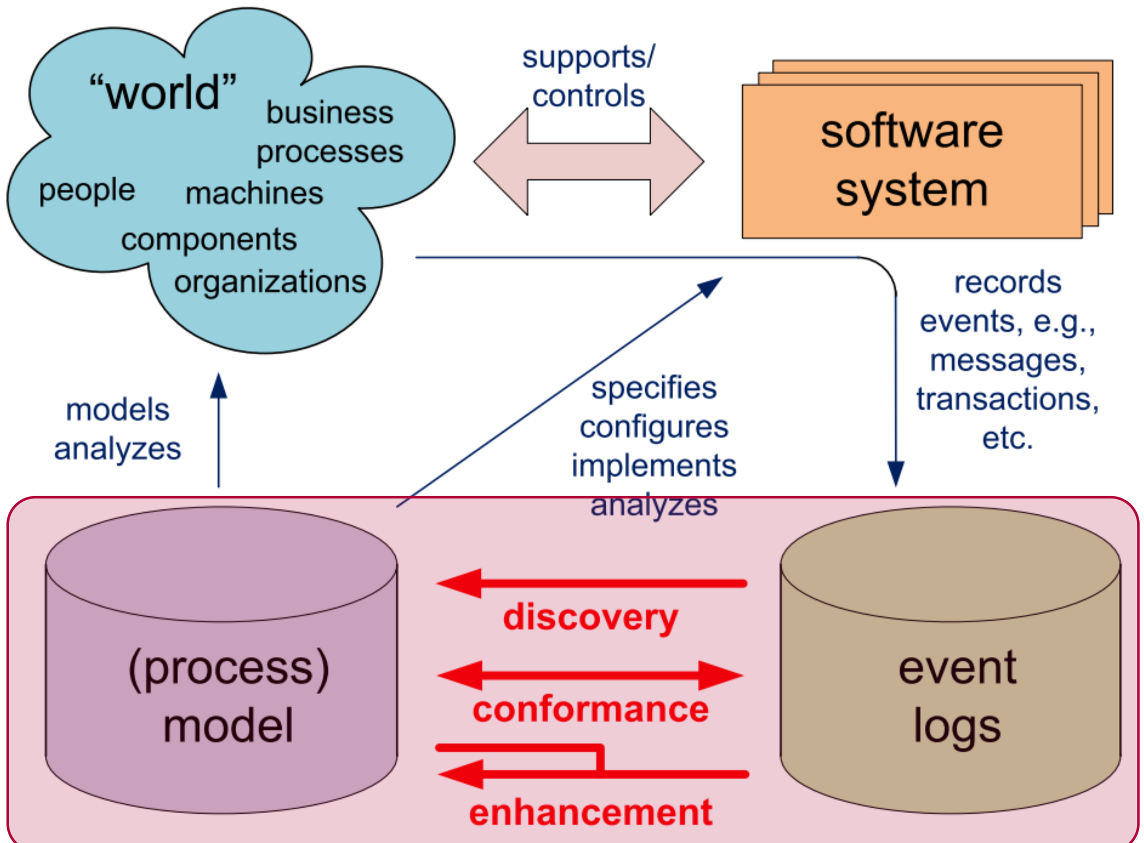


Image source: W. van der Aalst, "Process Mining", 2nd ed, Springer 2016.

Control Flow Discovery

- Discovery
  - Create process model for the observed behaviour
  - But: typically not every possible behaviour (i.e., trace) may have been executed and thus recorded

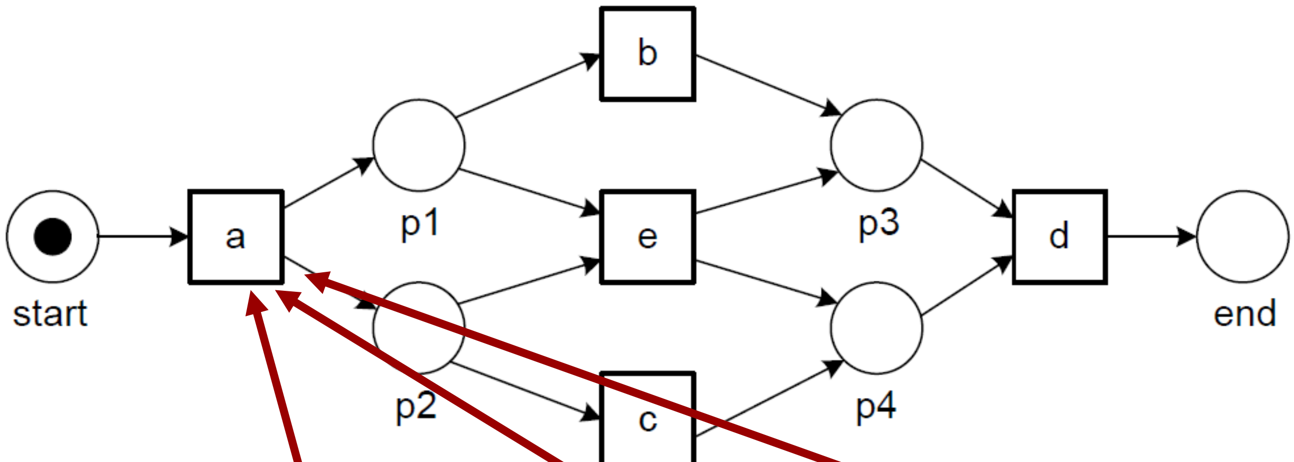
- Quality dimensions
  - Fitness: model should allow for behaviour seen in log
  - Precision – Generalisation trade-off: model does not allow for behaviour completely unrelated to log, but generalizes to some extent what is seen in the log
  - Simplicity: discovered model should be as simple as possible

Discovery Example

$$\underbrace{L_1 = [\langle a, b, c, d \rangle}_{}^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

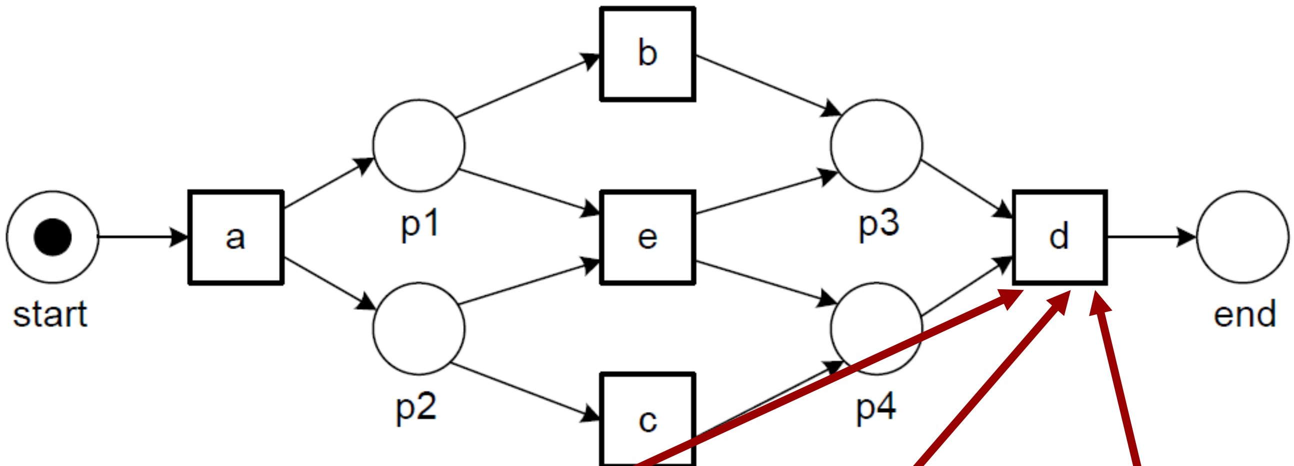
L_1 contains the sequence $\langle a, b, c, d \rangle$ three times

Discovery Example



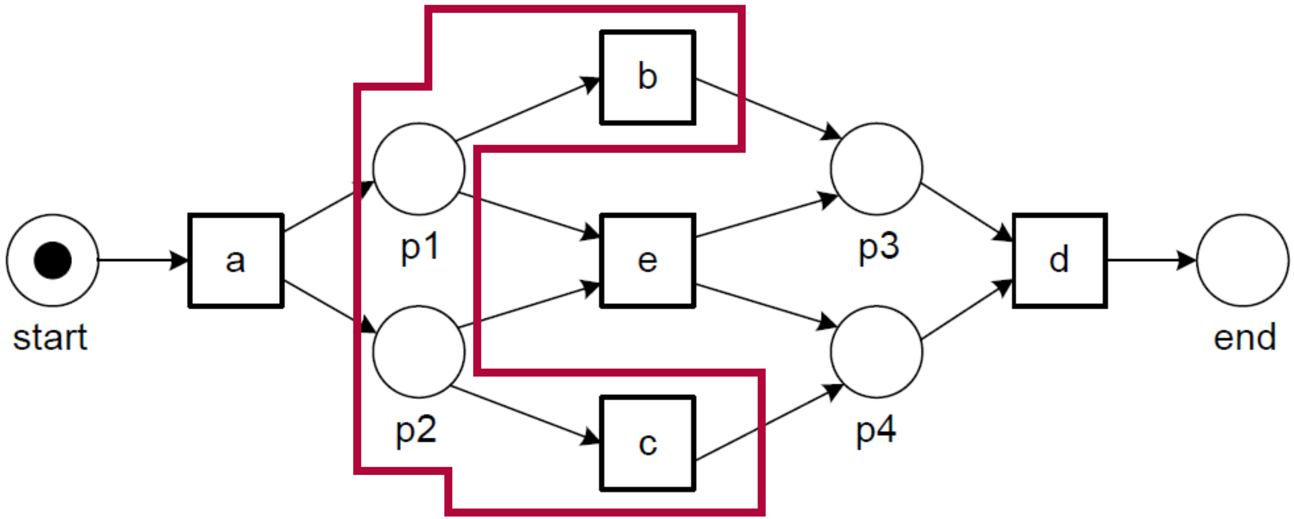
$$L_1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$$

Discovery Example



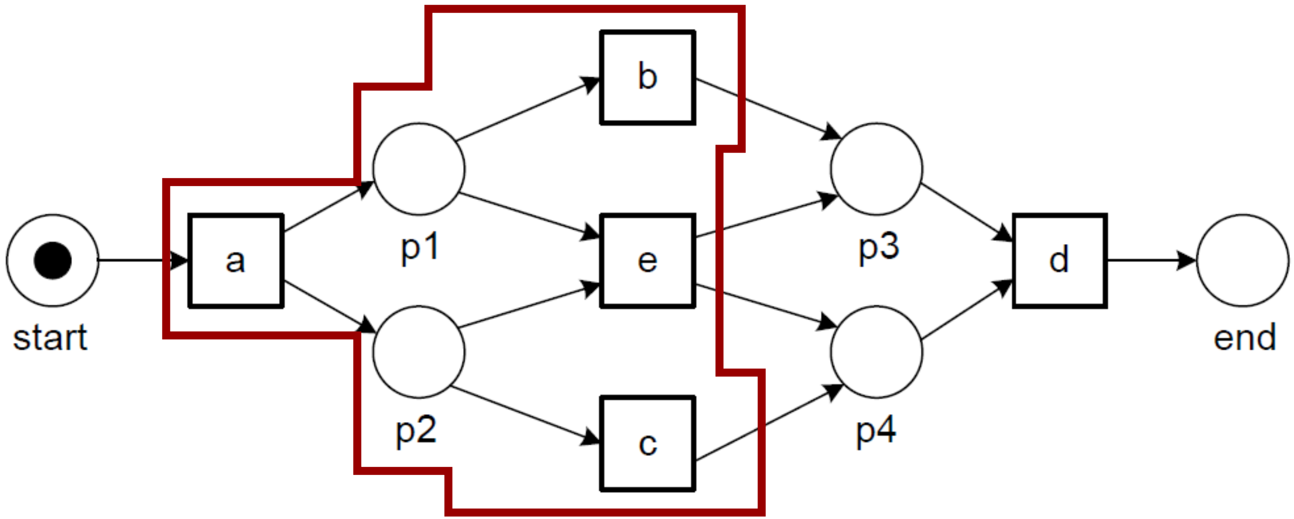
$$L_1 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$$

Discovery Example



$$L_1 = [\langle a, \boxed{b, c}, d\rangle^3, \langle a, \boxed{c, b}, d\rangle^2, \langle a, e, d\rangle]$$

Discovery Example



$$L_1 = [\langle \boxed{a, b, c}, d\rangle^3, \langle \boxed{a, c, b}, d\rangle^2, \langle \boxed{a, e}, d\rangle]$$

Discovery Example

Every trace starts with “a”

$$L_2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^4, \langle a, b, c, e, f, b, c, d\rangle^2, \langle a, b, c, e, f, c, b, d\rangle,$$
$$\langle a, c, b, e, f, b, c, d\rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d\rangle]$$

Discovery Example

Every trace ends with "d"

$$L_2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^4, \langle a, b, c, e, f, b, c, d\rangle^2, \langle a, b, c, e, f, c, b, d\rangle,$$
$$\langle a, c, b, e, f, b, c, d\rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d\rangle]$$

Discovery Example

"b" and "c" always occur together without a particular order

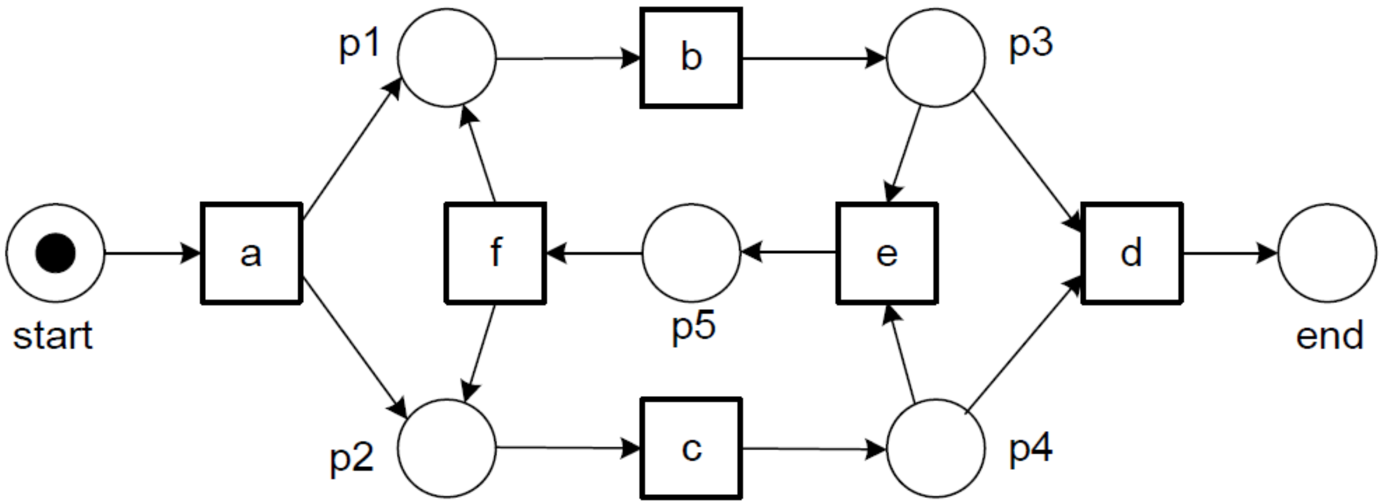
$$L_2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^4, \langle a, b, c, e, f, b, c, d\rangle^2, \langle a, b, c, e, f, c, b, d\rangle,$$
$$\langle a, c, b, e, f, b, c, d\rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d\rangle]$$

Discovery Example

Every "e" is followed by an "f", every "f" is preceded by an "e"

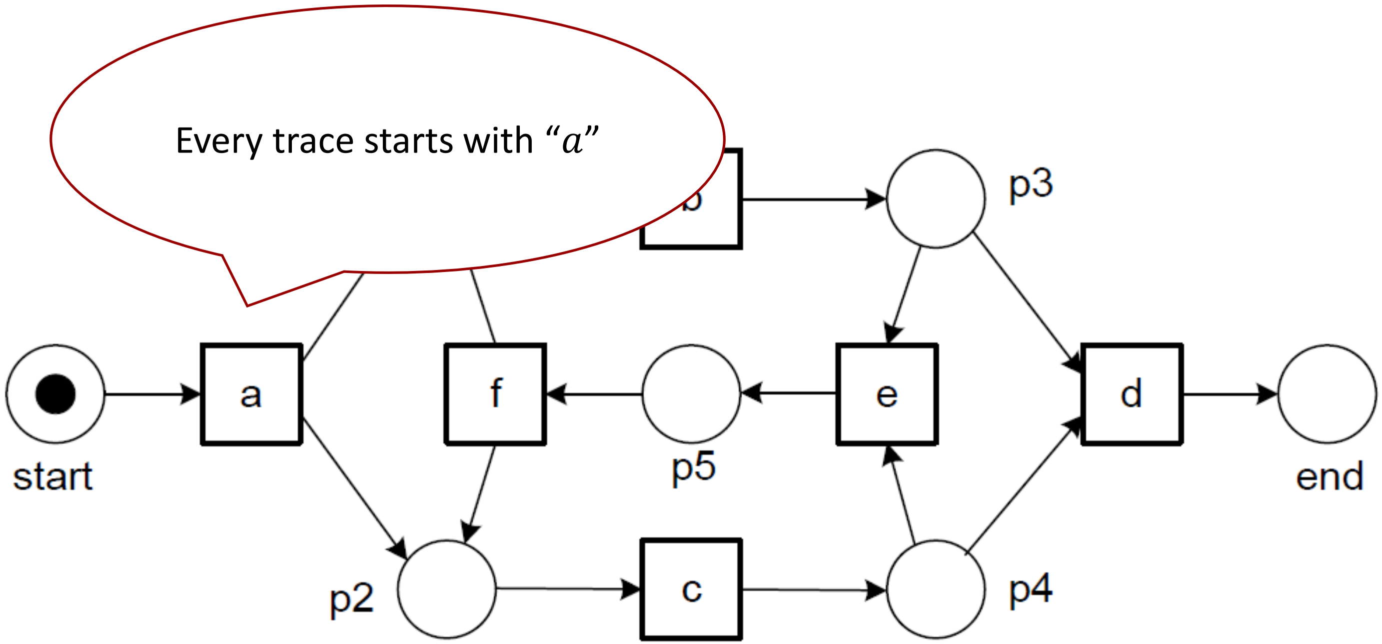
$$L_2 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^4, \langle a, b, c, e, f, b, c, d \rangle^2, \langle a, b, c, e, f, c, b, d \rangle,$$
$$\langle a, c, b, e, f, b, c, d \rangle^2, \langle a, c, b, e, f, b, c, e, f, c, b, d \rangle]$$

Discovery Example



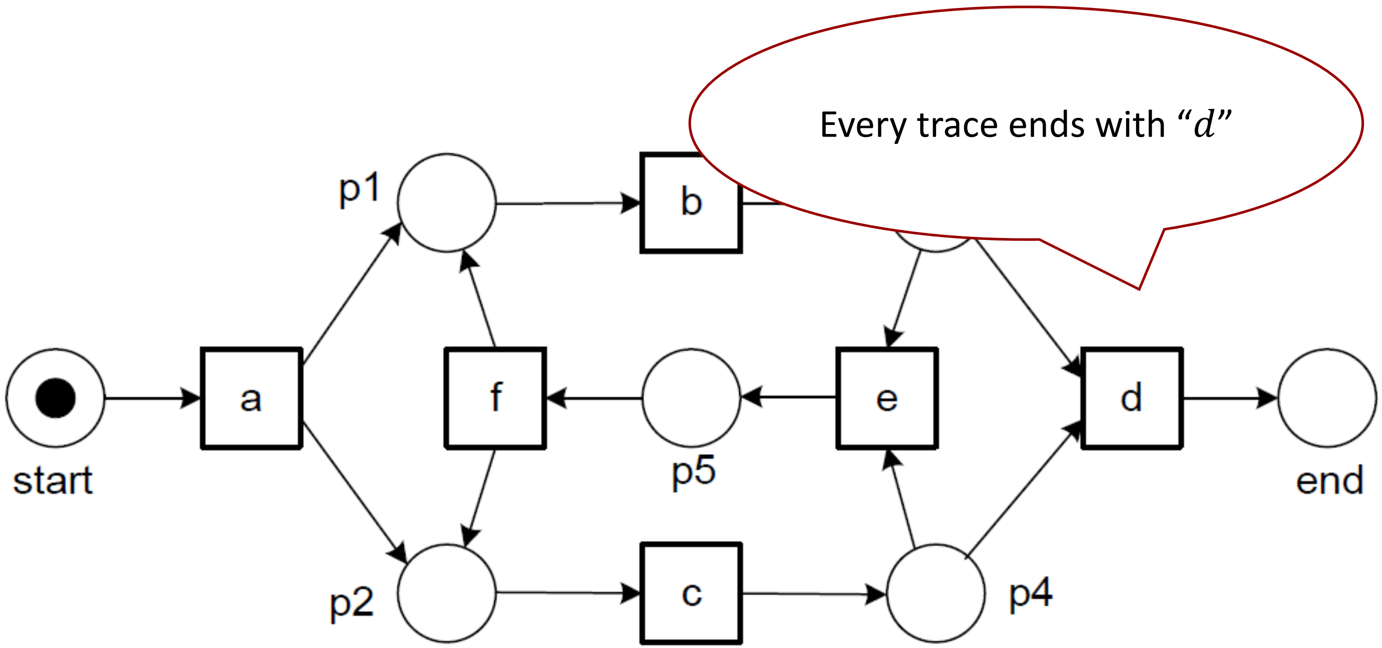
$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle, \langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Discovery Example



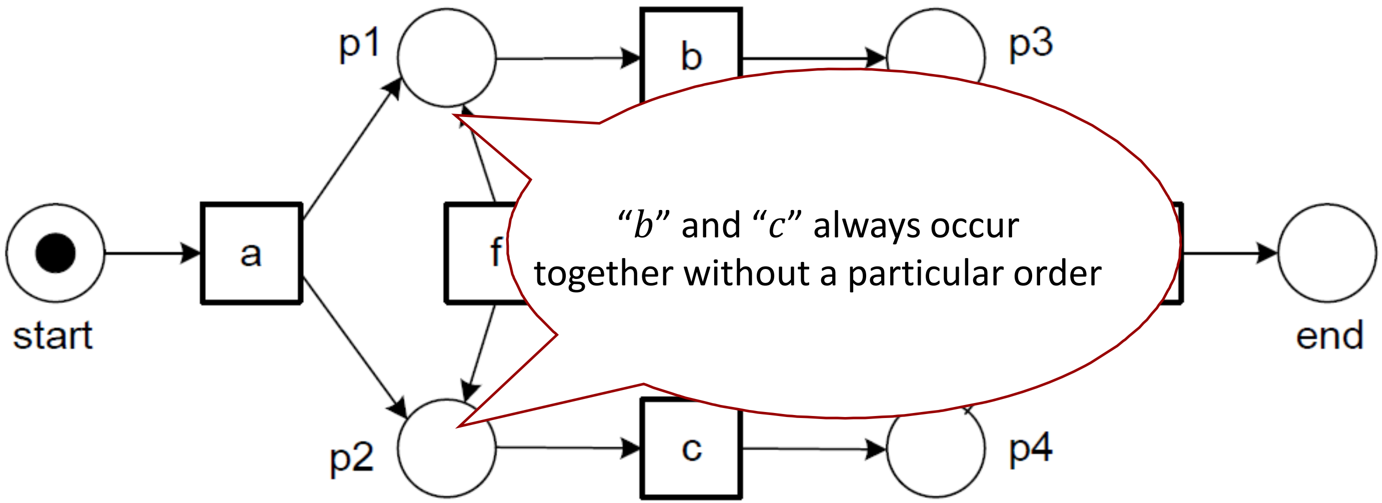
$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle,$$
$$\langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Discovery Example



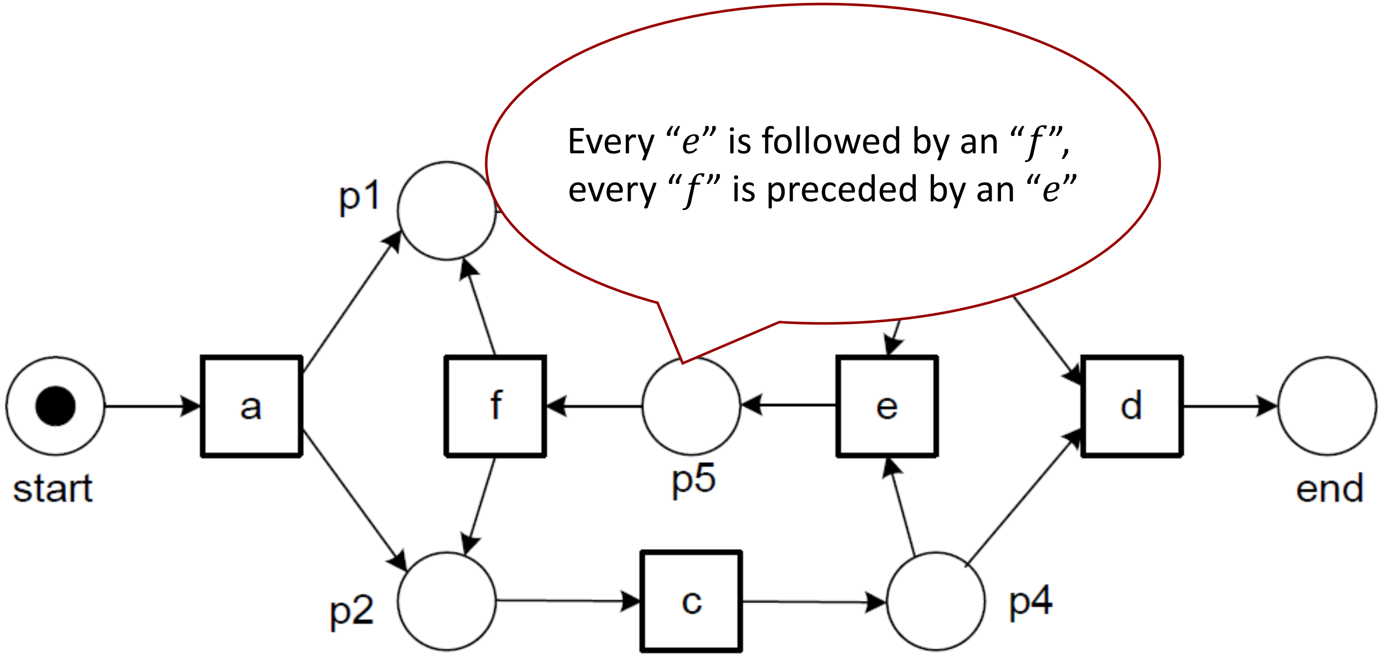
$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle,$$
$$\langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Discovery Example



$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle, \langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Discovery Example



$$L_2 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^4, \langle a,b,c,e,f,b,c,d\rangle^2, \langle a,b,c,e,f,c,b,d\rangle,$$
$$\langle a,c,b,e,f,b,c,d\rangle^2, \langle a,c,b,e,f,b,c,e,f,c,b,d\rangle]$$

Discovery Algorithms

- Simple algorithm: α – algorithm
  - Detect concurrent execution of activities
  - Relatively easy, certain properties can be proven
  - Not robust against noise, incomplete or erroneous logs
  - Therefore, nice for illustration, but limited use in practice

- Extensions: α+(+) – Algorithms
  - α+ and α++ extend α – Algorithm to broaden the spectrum of constructs that may be discovered
  - Still not robust against noise

- Robust algorithms will be discussed later

See also Process Mining: Discovery, Conformance and Enhancement of Business Processes by W.M.P. van der Aalst, Springer Verlag, 2011 (ISBN 978-3-642-19344-6). Slides partly due to Wil van der Aalst

Workflow Log

- General notion of event log
  - Format of log entries: (timestamp, case ID, activity, additional attributes …)
  - Various abstractions can be applied

- One abstraction: the notion of a workflow log
  - Set of all distinct sequences of activity executions
  - Let T be a set of activities (aka tasks)
  - Let T^* the set of all finite sequences over T
  - $\sigma \in T^*$ is a trace, all tasks in σ belong to the same case
  - $W \subseteq T^*$ is a workflow log

No cardinalities

No case IDs, …

- Assumption: each task occurs at most once in a process model

Workflow Log

- Traces:
  - Case 1: $\langle ABCD \rangle$
  - Case 2: $\langle ACBD \rangle$
  - Case 3: $\langle ABCD \rangle$
  - Case 4: $\langle ACBD \rangle$
  - Case 5: $\langle EF \rangle$

- Workflow log
  - $W = \{\langle ABCD \rangle, \langle ACBD \rangle, \langle EF \rangle\}$

No cardinalities,
no case IDs

```
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 tasks $x, y\ b \in T$ in a workflow log W:
  - Direct succession

    $x > y$ iff in ≥ 1 case x is *directly* followed by y
  - Causality

    $x \rightarrow y$ 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 $y > x$

Example

Workflow log:

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

```
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
```

1.

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

2.

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

3.

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

4.

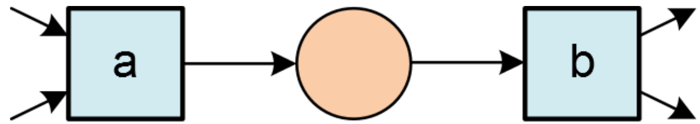
$A \# A$
$A \# D$
$A \# E$
$A \# F$
$B \# B$
$B \# E$
$B \# F$
$C \# C$
...

α-Algorithm Idea

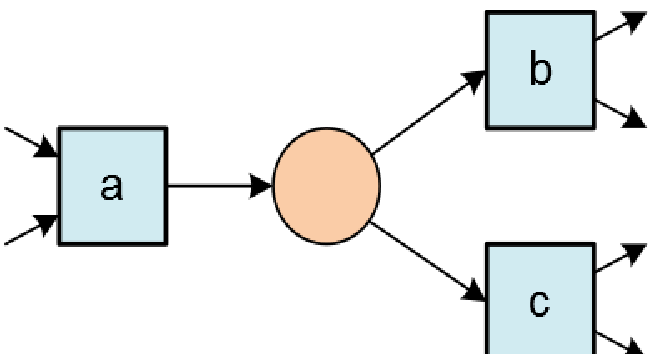
- Idea: create workflow net based on the ordering relations, such that the ordering is obeyed by the net

- Realisation: derive a Petri net fragment from the each entry of the ordering relations

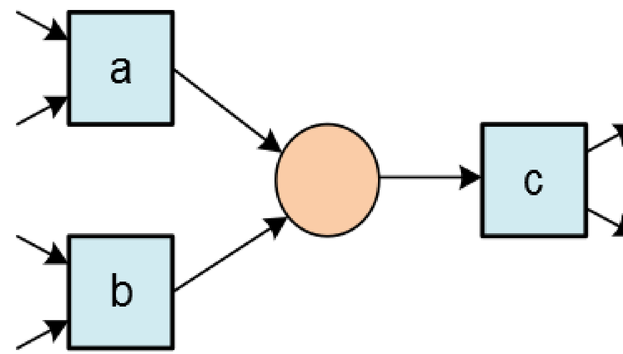
Basic ideas



(a) sequence pattern: a→b

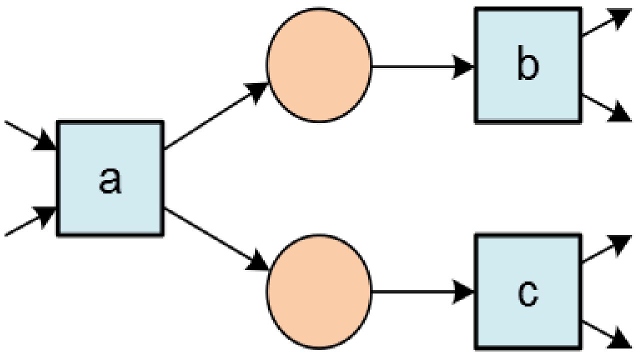


(b) XOR-split pattern:
a→b, a→c, and b#c

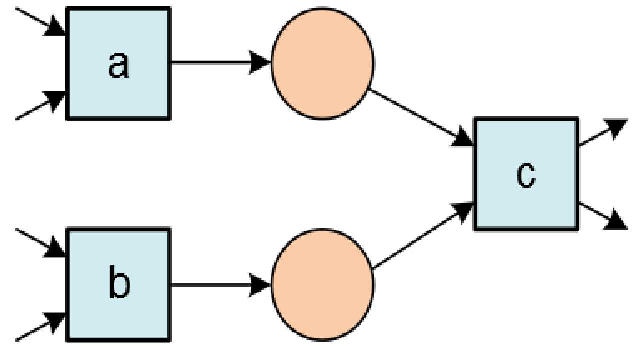


(c) XOR-join pattern:
a→c, b→c, and a#b

Basic ideas



(d) AND-split pattern:
a→b, a→c, and b||c



(e) AND-join pattern:
a→c, b→c, and a||b

α-Algorithm

- Let W be an event log over T
- $\alpha(W)$ is defined as follows
  - $T_W = \{\, t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \,\}$
  - $T_I = \{\, t \in T \mid \exists_{\sigma \in W}\, t = \text{first}(\sigma) \,\}$
  - $T_O = \{\, t \in T \mid \exists_{\sigma \in W}\, t = \text{last}(\sigma) \,\}$
  - $X_W = \{(A,B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land$
    $\forall_{a \in A} \forall_{b \in B}\, a \rightarrow b \land \forall_{a_1,a_2 \in A} a_1 \# a_2 \land \forall_{b_1,b_2 \in B} b_1 \# b_2\}$
  - $Y_W = \left\{ (A,B) \in X_W \,\middle|\, \forall_{(A',B') \in X_W} A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \right\}$
  - $P_W = \{\, p(A,B) \mid (A,B) \in Y_W \,\} \cup \{i_W, o_W\}$
  - $F_W = \left\{ \big(a, p(A,B)\big) \mid (A,B) \in Y_W \land a \in A \right\}$
    $\cup \{\, (p(A,B), b) \mid (A,B) \in Y_W \land b \in B \,\} \cup \{\, (i_W, t) \mid t \in T_I \,\} \cup \{\, (t, o_W) \mid t \in T_O \,\}$
  - $\alpha(W) = (P_W, T_W, F_W)$

α-Algorithm

- Let W be an event log over T
- $\alpha(W)$ is defined as follows
  - $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$
  - $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = \text{first}(\sigma) \}$
  - $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = \text{last}(\sigma) \}$
  - $X_W = \{(A,B) \mid A \subseteq T_W \wedge A \neq \emptyset \wedge B \subseteq T_W \wedge B \neq \emptyset \wedge$
    $\forall_{a \in A} \forall_{b \in B}\, a \rightarrow b \wedge \forall_{a_1,a_2 \in A} a_1 \# a_2 \wedge \forall_{b_1,b_2 \in B} b_1 \# b_2\}$
  - $Y_W = \left\{ (A,B) \in X_W \,\middle|\, \forall_{(A',B') \in X_W} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \right\}$
  - $P_W = \{ p(A,B) \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$
  - $F_W = \{ \left(a, p(A,B)\right) \mid (A,B) \in Y_W \wedge a \in A \}$
    $\cup \{ (p(A,B), b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$
  - $\alpha(W) = (P_W, T_W, F_W)$

Result is a WF-net:
- P_W is set of places
- T_W is set of transitions
- F_W is flow relation

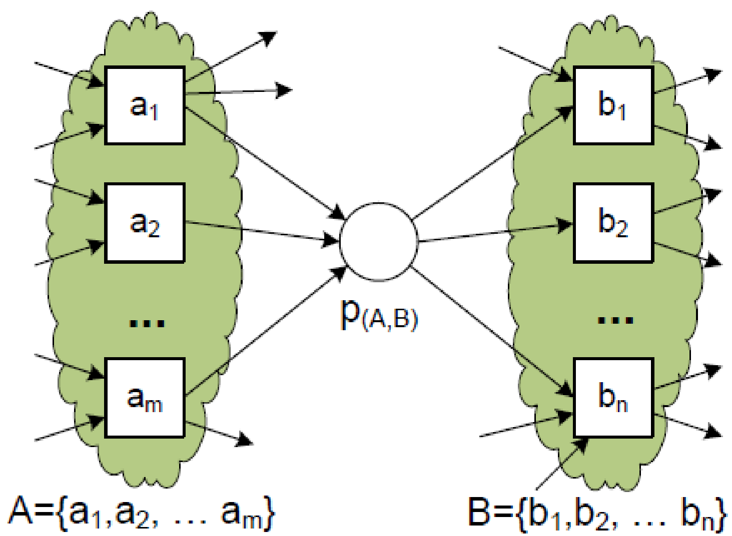
α-Algorithm

- Let W be an event log over T
- $\alpha(W)$ is defined as follows
  - $T_W = \{ t \in T \mid \exists_{\sigma \in W}\ t \in \sigma \}$
  - $T_I = \{ t \in T \mid \exists_{\sigma \in W}\ t = \mathrm{first}(\sigma) \}$
  - $T_O = \{ t \in T \mid \exists_{\sigma \in W}\ t = \mathrm{last}(\sigma) \}$
  - $X_W = \{(A,B) \mid A \subseteq T_W \wedge A \neq \emptyset \wedge B \subseteq T_W \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B}\ a \rightarrow b \wedge \forall_{a_1,a_2 \in A} a_1 \# a_2 \wedge \forall_{b_1,b_2 \in B} b_1 \# b_2\}$
  - $Y_W = \left\{ (A,B) \in X_W \,\middle|\, \forall_{(A',B') \in X_W} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \right\}$
  - $P_W = \{ p(A,B) \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$
  - $F_W = \{ \big(a, p(A,B)\big) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p(A,B), b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$
  - $\alpha(W) = (P_W, T_W, F_W)$

- Derive set of transitions from all traces (one per activity)
- Identify initial/final transitions

Key idea: find places

- $X_W = \{(A,B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land \forall_{a \in A} \forall_{b \in B}\ a \rightarrow b \land \forall_{a_1,a_2 \in A}\ a_1 \# a_2 \land \forall_{b_1,b_2 \in B}\ b_1 \# b_2\}$
- $Y_W = \left\{ (A,B) \in X_W \,\middle|\, \forall_{(A',B') \in X_W}\ A \subseteq A' \land B \subseteq B' \Rightarrow (A,B) = (A',B') \right\}$



α-Algorithm

- Let W be an event log over T
- $\alpha(W)$ is defined as follows
  - $T_W = \{\, t \in T \mid \exists_{\sigma \in W}\ t \in \sigma \,\}$
  - $T_I = \{\, t \in T \mid \exists_{\sigma \in W}\ t = \mathrm{first}(\sigma) \,\}$
  - $T_O = \{\, t \in T \mid \exists_{\sigma \in W}\ t = \mathrm{last}(\sigma) \,\}$
  - $X_W = \{(A,B) \mid A \subseteq T_W \wedge A \neq \emptyset \wedge B \subseteq T_W \wedge B \neq \emptyset \wedge$
    $\forall_{a \in A} \forall_{b \in B}\ a \rightarrow b \wedge \forall_{a_1,a_2 \in A} a_1 \# a_2 \wedge \forall_{b_1,b_2 \in B} b_1 \# b_2\}$
  - $Y_W = \left\{ (A,B) \in X_W \;\middle|\; \forall_{(A',B') \in X_W} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \right\}$
  - $P_W = \{\, p(A,B) \mid (A,B) \in Y_W \,\} \cup \{i_W, o_W\}$
  - $F_W = \left\{ \left(a, p(A,B)\right) \mid (A,B) \in Y_W \wedge a \in A \right\}$
    $\cup \{\, (p(A,B), b) \mid (A,B) \in Y_W \wedge b \in B \,\} \cup \{\, (i_W, t) \mid t \in T_I \,\} \cup \{\, (t, o_W) \mid t \in T_O \,\}$
  - $\alpha(W) = (P_W, T_W, F_W)$

- Place created for each element in Y_W
- Two special places for input/output

α-Algorithm

- Let W be an event log over T
- $\alpha(W)$ is defined as follows
  - $T_W = \{ t \in T \mid \exists_{\sigma \in W}\ t \ \in \sigma \}$
  - $T_I = \{ t \in T \mid \exists_{\sigma \in W}\ t = \text{first}(\sigma) \}$
  - $T_O = \{ t \in T \mid \exists_{\sigma \in W}\ t = \text{last}(\sigma) \}$
  - $X_W = \{(A, B) \mid A \subseteq T_W \land A \neq \emptyset \land B \subseteq T_W \land B \neq \emptyset \land$ $\forall_{a \in A} \forall_{b \in B}\ a \rightarrow b \land \forall_{a_1, a_2 \in A} a_1 \# a_2 \land \forall_{b_1, b_2 \in B} b_1 \# b_2\}$
  - $Y_W = \left\{ (A, B) \in X_W \;\middle|\; \forall_{(A', B') \in X_W} A \subseteq A' \land B \subseteq B' \Rightarrow (A, B) = \ (A', B') \right\}$
  - $P_W = \{ p(A, B) \mid (A, B) \in Y_W \} \cup \{i_W, o_W\}$
  - $F_W = \left\{ \big(a, p(A, B)\big) \mid (A, B) \in Y_W \land a \in A \right\}$ $\cup \{ (p(A, B), b) \mid (A, B) \in Y_W \land b \ \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$
  - $\alpha(W) = (P_W, T_W, F_W)$

- Everything is connected populating the flow relations

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

$\alpha(W)$: