# Heuristics Miner for Time Intervals

Andrea Burattin and Alessandro Sperduti

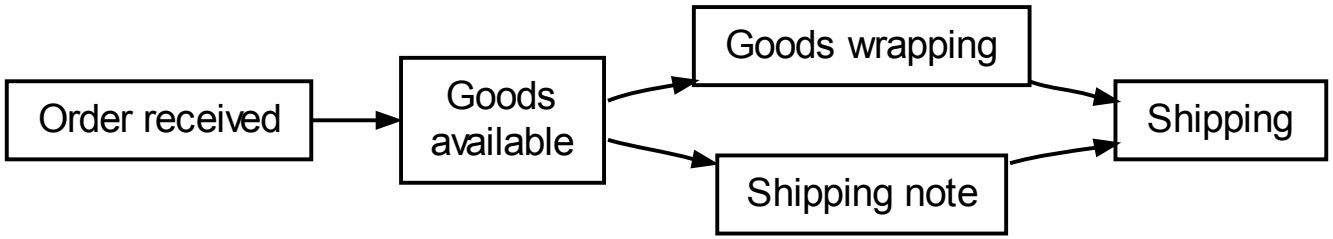
Department of Pure and Applied Mathematics
University of Padua, Italy

April 28th, 2010

# What is Process Mining I

## Business Process

From the IEEE Glossary: "*a sequence of steps performed for a given purpose; for example, the software development process*", that changes *inputs* into *outputs*.




Each performed action is registered into a *log*

# What is Process Mining II

## Main process mining areas

When the model of the process is not available:

Control-flow discovery aims to build a model describing the behaviour of the process;

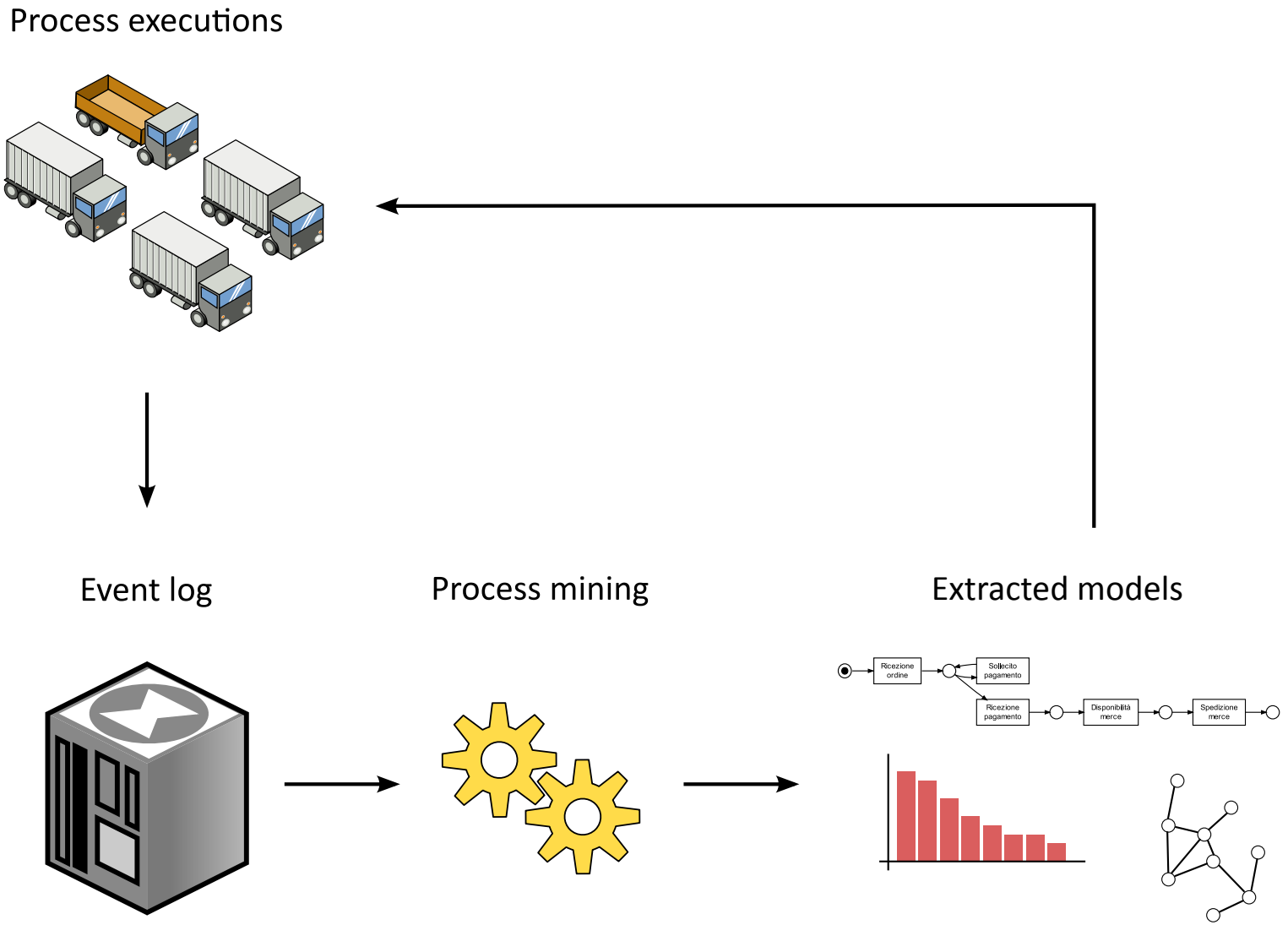
When the model of the process is available:

Conformance analysis tries to fit a log to the given process model.

Independent from the process model availability:

Organizational mining tries to extract a "social network" that establishes relations between actions' authors;

# What is Process Mining III



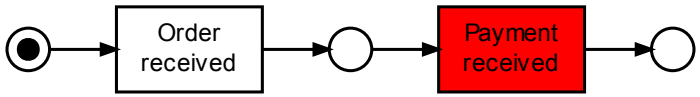
# Control–flow discovery example run

| # | Activities | Completion Time |
|---|---|---|
| | **Instance 1** | |
| 1 | Order received | apr 21, 2010 12:00 |
| 2 | Payment received | apr 22, 2010 09:00 |
| 3 | Goods available | apr 26, 2010 08:30 |
| 4 | Shipping | apr 26, 2010 10:15 |
| | **Instance 2** | |
| 1 | Order received | apr 23, 2010 15:45 |
| 2 | Payment reminder | apr 25, 2010 15:45 |
| 3 | Payment received | apr 25, 2010 17:31 |
| 4 | Goods available | apr 26, 2010 10:00 |
| 5 | Shipping | apr 26, 2010 12:30 |

# Control–flow discovery example run

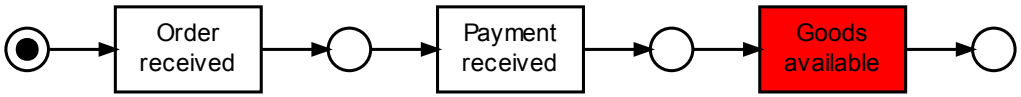

| # | Activities | Completion Time |
|---|---|---|
| | **Instance 1** | |
| 1 | Order received | apr 21, 2010 12:00 |
| 2 | Payment received | apr 22, 2010 09:00 |
| 3 | Goods available | apr 26, 2010 08:30 |
| 4 | Shipping | apr 26, 2010 10:15 |
| | **Instance 2** | |
| 1 | Order received | apr 23, 2010 15:45 |
| 2 | Payment reminder | apr 25, 2010 15:45 |
| 3 | Payment received | apr 25, 2010 17:31 |
| 4 | Goods available | apr 26, 2010 10:00 |
| 5 | Shipping | apr 26, 2010 12:30 |

# Control–flow discovery example run



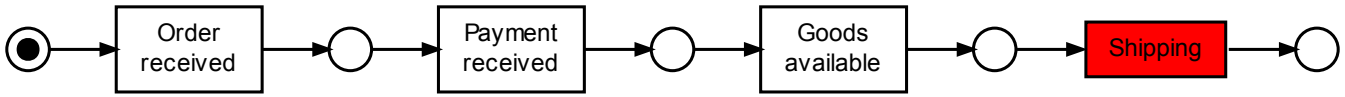

| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | #1 > #2 |
| 3 | Goods available | apr 26, 2010 08:30 | |
| 4 | Shipping | apr 26, 2010 10:15 | |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | |
| 3 | Payment received | apr 25, 2010 17:31 | |
| 4 | Goods available | apr 26, 2010 10:00 | |
| 5 | Shipping | apr 26, 2010 12:30 | |

# Control–flow discovery example run



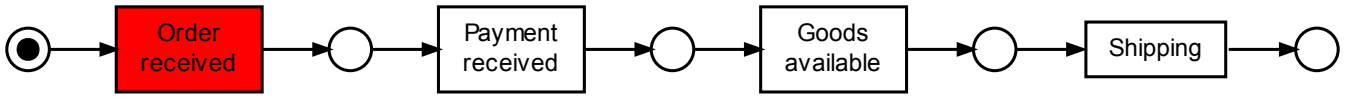

| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | |
| 3 | Goods available | apr 26, 2010 08:30 | #2 > #3 |
| 4 | Shipping | apr 26, 2010 10:15 | |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | |
| 3 | Payment received | apr 25, 2010 17:31 | |
| 4 | Goods available | apr 26, 2010 10:00 | |
| 5 | Shipping | apr 26, 2010 12:30 | |

# Control–flow discovery example run



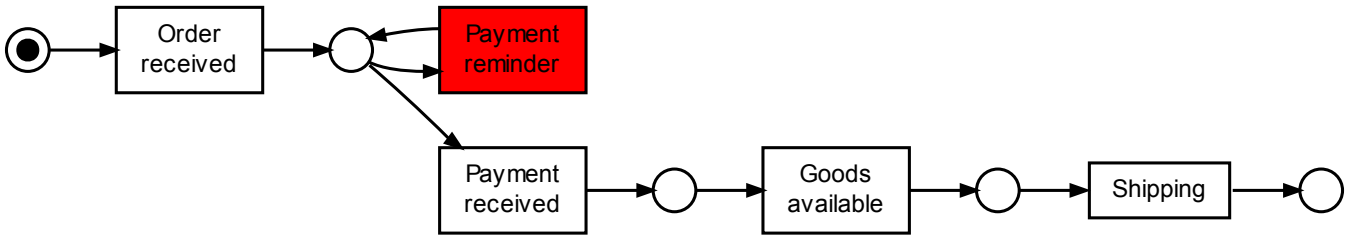

| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | |
| 3 | Goods available | apr 26, 2010 08:30 | |
| 4 | Shipping | apr 26, 2010 10:15 | $\#3 > \#4$ |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | |
| 3 | Payment received | apr 25, 2010 17:31 | |
| 4 | Goods available | apr 26, 2010 10:00 | |
| 5 | Shipping | apr 26, 2010 12:30 | |

# Control–flow discovery example run



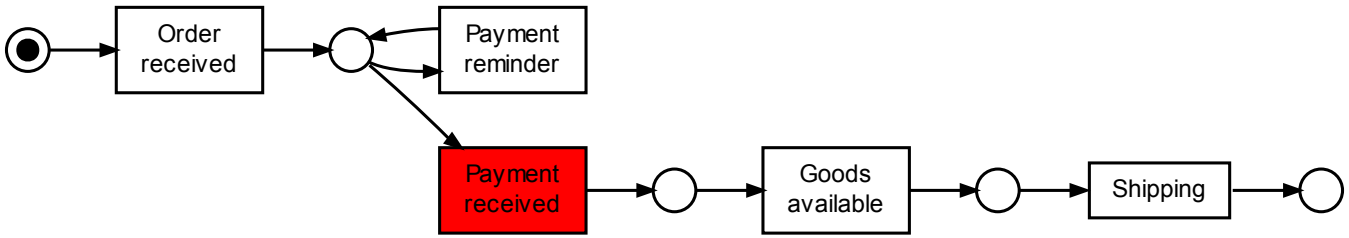

| # | Activities | Completion Time |
|---|---|---|
| | **Instance 1** | |
| 1 | Order received | apr 21, 2010 12:00 |
| 2 | Payment received | apr 22, 2010 09:00 |
| 3 | Goods available | apr 26, 2010 08:30 |
| 4 | Shipping | apr 26, 2010 10:15 |
| | **Instance 2** | |
| 1 | Order received | apr 23, 2010 15:45 |
| 2 | Payment reminder | apr 25, 2010 15:45 |
| 3 | Payment received | apr 25, 2010 17:31 |
| 4 | Goods available | apr 26, 2010 10:00 |
| 5 | Shipping | apr 26, 2010 12:30 |

# Control–flow discovery example run



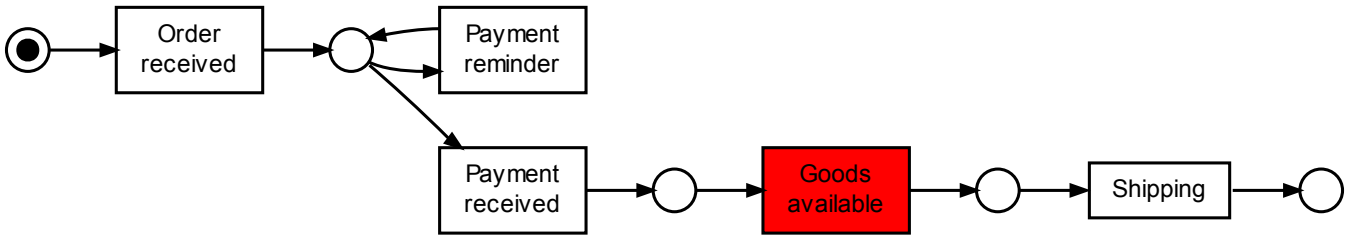

| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | |
| 3 | Goods available | apr 26, 2010 08:30 | |
| 4 | Shipping | apr 26, 2010 10:15 | |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | $\#1 > \#2$ |
| 3 | Payment received | apr 25, 2010 17:31 | |
| 4 | Goods available | apr 26, 2010 10:00 | |
| 5 | Shipping | apr 26, 2010 12:30 | |

# Control–flow discovery example run



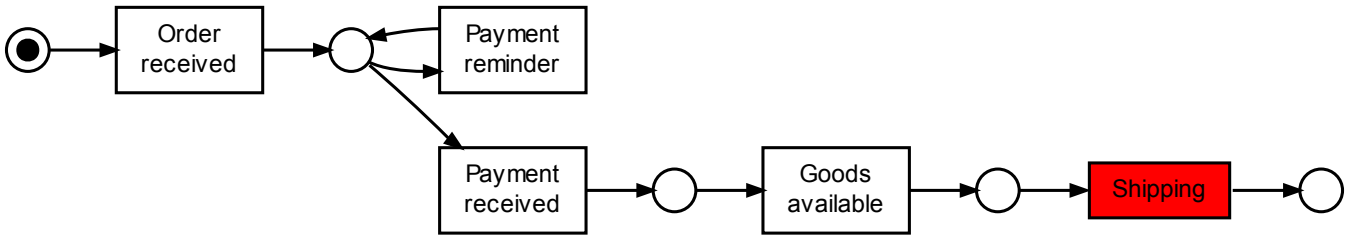

| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | |
| 3 | Goods available | apr 26, 2010 08:30 | |
| 4 | Shipping | apr 26, 2010 10:15 | |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | |
| 3 | Payment received | apr 25, 2010 17:31 | #2 > #3 |
| 4 | Goods available | apr 26, 2010 10:00 | |
| 5 | Shipping | apr 26, 2010 12:30 | |

# Control–flow discovery example run




| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | |
| 3 | Goods available | apr 26, 2010 08:30 | |
| 4 | Shipping | apr 26, 2010 10:15 | |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | |
| 3 | Payment received | apr 25, 2010 17:31 | |
| 4 | Goods available | apr 26, 2010 10:00 | $\#3 > \#4$ |
| 5 | Shipping | apr 26, 2010 12:30 | |

# Control–flow discovery example run




| # | Activities | Completion Time | |
|---|---|---|---|
| | **Instance 1** | | |
| 1 | Order received | apr 21, 2010 12:00 | |
| 2 | Payment received | apr 22, 2010 09:00 | |
| 3 | Goods available | apr 26, 2010 08:30 | |
| 4 | Shipping | apr 26, 2010 10:15 | |
| | **Instance 2** | | |
| 1 | Order received | apr 23, 2010 15:45 | |
| 2 | Payment reminder | apr 25, 2010 15:45 | |
| 3 | Payment received | apr 25, 2010 17:31 | |
| 4 | Goods available | apr 26, 2010 10:00 | |
| 5 | Shipping | apr 26, 2010 12:30 | $\#4 > \#5$ |

# Control–flow discovery example run

| # | Activities | Completion Time |
|---|---|---|
| | **Instance 1** | |
| 1 | Order received | apr 21, 2010 12:00 |
| 2 | Payment received | apr 22, 2010 09:00 |
| 3 | Goods available | apr 26, 2010 08:30 |
| 4 | Shipping | apr 26, 2010 10:15 |
| | **Instance 2** | |
| 1 | Order received | apr 23, 2010 15:45 |
| 2 | Payment reminder | apr 25, 2010 15:45 |
| 3 | Payment received | apr 25, 2010 17:31 |
| 4 | Goods available | apr 26, 2010 12:30 |
| 5 | Shipping | apr 26, 2010 12:30 |

# Control–flow discovery example run

| # | Activities | Completion Time |
|---|---|---|
| | **Instance 1** | |
| 1 | Order received | apr 21, 2010 12:00 |
| 2 | Payment received | apr 22, 2010 09:00 |
| 3 | Goods available | apr 26, 2010 08:30 |
| 4 | Shipping | apr 26, 2010 10:15 |
| | **Instance 2** | |
| 1 | Order received | apr 23, 2010 15:45 |
| 2 | Payment reminder | apr 25, 2010 15:45 |
| 3 | Payment received | apr 25, 2010 17:31 |
| 4 | Goods available | apr 26, 2010 12:30 |
| 5 | Shipping | apr 26, 2010 12:30 |

# Control–flow discovery example run

| # | Activities | Completion Time |
|---|---|---|
| | **Instance 1** | |
| 1 | Order received | apr 21, 2010 12:00 |
| 2 | Payment received | apr 22, 2010 09:00 |
| 3 | Goods available | apr 26, 2010 08:30 |
| 4 | Shipping | apr 26, 2010 10:15 |
| | **Instance 2** | |
| 1 | Order received | apr 23, 2010 15:45 |
| 2 | Goods available | apr 25, 2010 15:45 |
| 3 | Payment received | apr 25, 2010 17:31 |
| 4 | Payment reminder | apr 26, 2010 12:30 |
| 5 | Shipping | apr 26, 2010 12:30 |

# Heuristics Miner, core behaviour I

Heuristics Miner evaluates a "*dependency function*" between two activities (e.g. $X$, $Y$), in order to decide if the relationship holds:

$$X \Rightarrow Y = \frac{|X > Y| - |Y > X|}{|X > Y| + |Y > X| + 1}$$

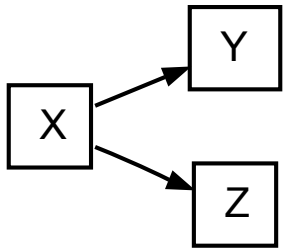
Where:

- $X > Y$ holds if $X$ executed at time $t$ and $Y$ at $t + 1$
- $|X > Y|$ is the number of times that $X > Y$ holds in the log

With all the relations above a threshold, the algorithm builds a directed graph with all the dependencies.

# Heuristics Miner, core behaviour II

We can have both $X \Rightarrow Y$ and $X \Rightarrow Z$:



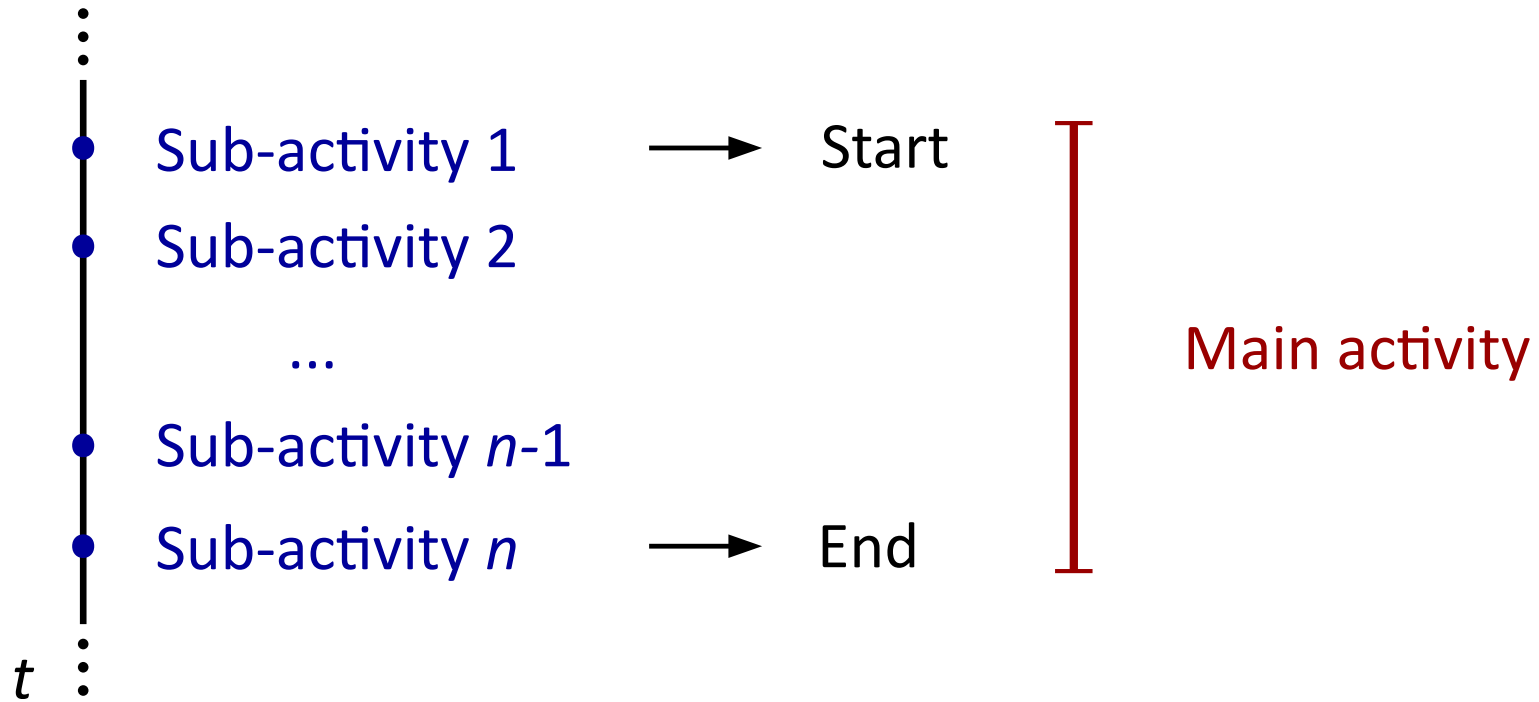

$Y$ and $Z$ can be executed in mutual exclusion (XOR) or in parallel (in no specific order; AND)

$$X \Rightarrow (Y \wedge Z) = \frac{|Y > Z| + |Z > Y|}{|X > Y| + |X > Z| + 1}$$

If the value is above a threshold than AND relation else XOR

# Time intervals in the logs

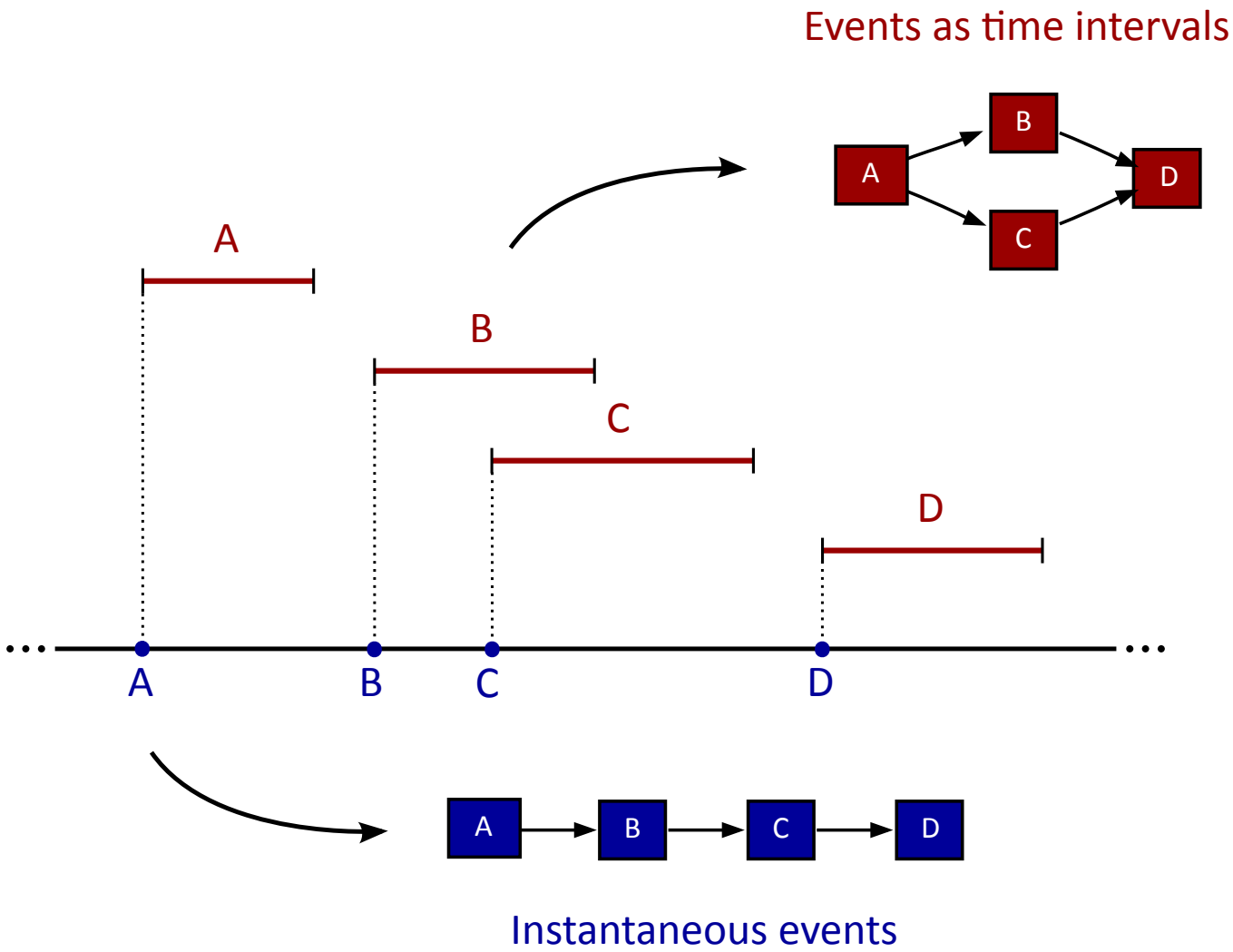
Many times, activities are stored inside the log in terms of many "sub-activities", composing the main one:




Considering the *first* and the *last* sub-activity, we can build a time interval for the main activity.

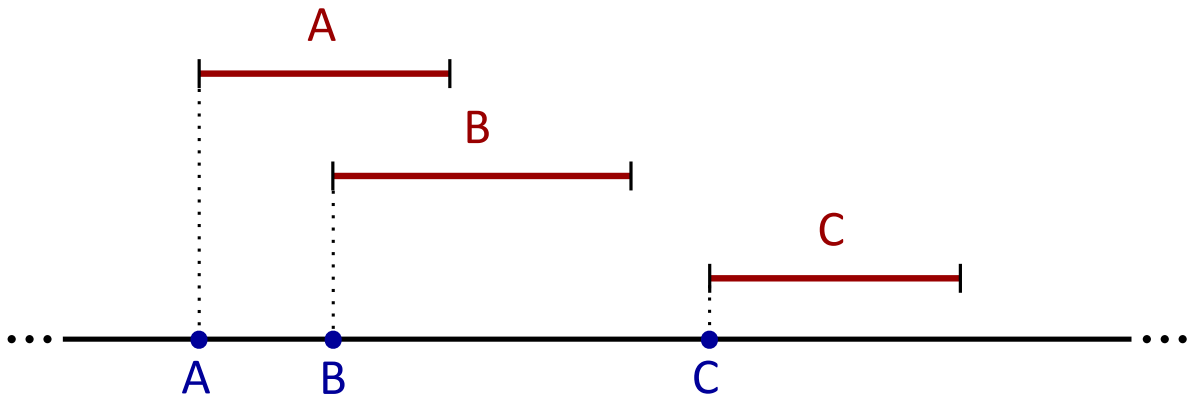
# Information on the intervals vs time spot

Allen's Interval Algebra: the "overlap relation"



# New definitions, with time intervals support I

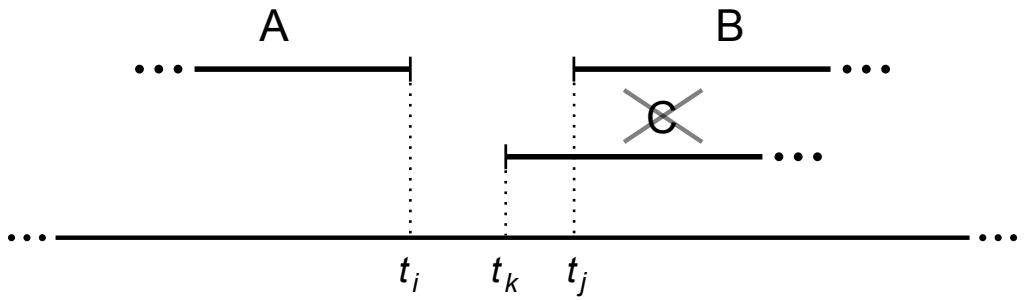
| | | |
|---|---|---|
| **Heuristics Miner** | $a > b$ | direct succession of points |
| **Heuristics Miner++** | $a \overline{>} b$ | direct succession of intervals |



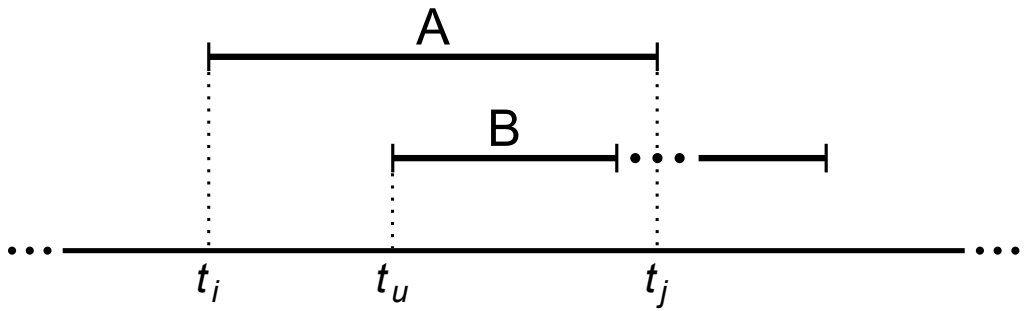

$A > B$ holds $A \overline{>} B$ does not hold $A \overline{>} C$ holds

# New definitions, with time intervals support II

**Direct succession**, $A \overline{>} B$




**Parallelism** (overlap relation), $A \parallel B$



# New definitions, with time intervals support III

**Dependency function**, for time intervals

$$X \Rightarrow Y = \frac{|X \overline{>} Y| - |Y \overline{>} X|}{|X \overline{>} Y| + |Y \overline{>} X| + 2 \cdot |X \| Y| + 1}$$

# New definitions, with time intervals support III

**Dependency function**, for time intervals

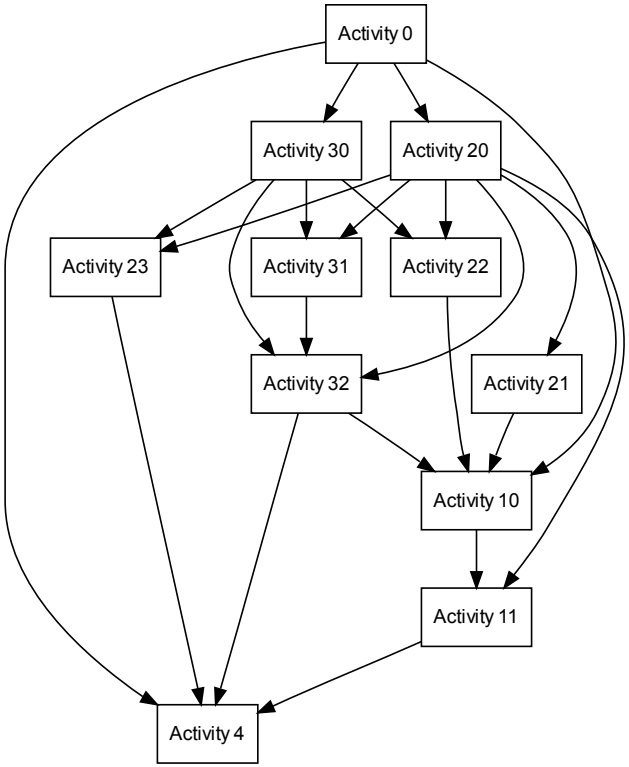
$$X \Rightarrow Y = \frac{|X \overline{>} Y| - |Y \overline{>} X|}{|X \overline{>} Y| + |Y \overline{>} X| + 2 \cdot |X \| Y| + 1}$$

**AND function**, for time intervals

$$X \Rightarrow (Y \wedge Z) = \frac{|Y \overline{>} Z| + |Z \overline{>} Y| + 2 \cdot |Y \| Z|}{|X \overline{>} Y| + |X \overline{>} Z| + 1}$$

# Results for a real case

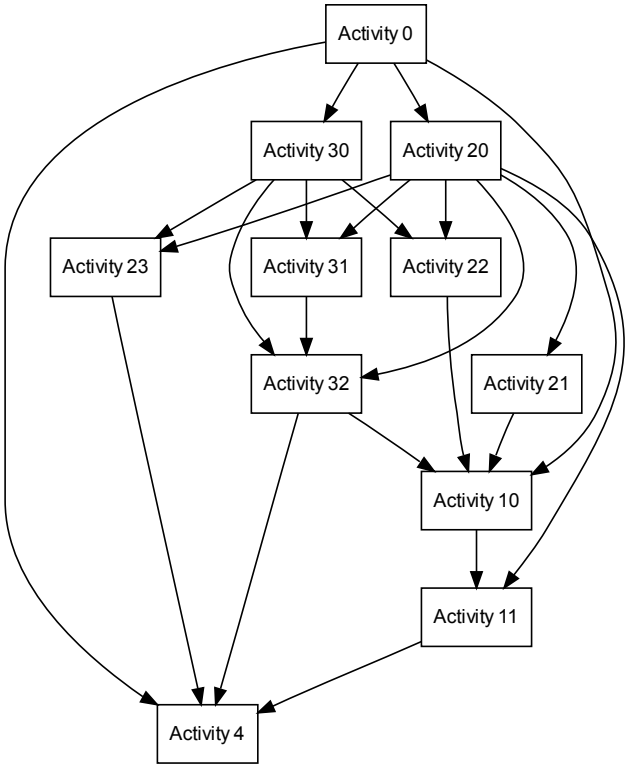
Result for
**Heuristics Miner**



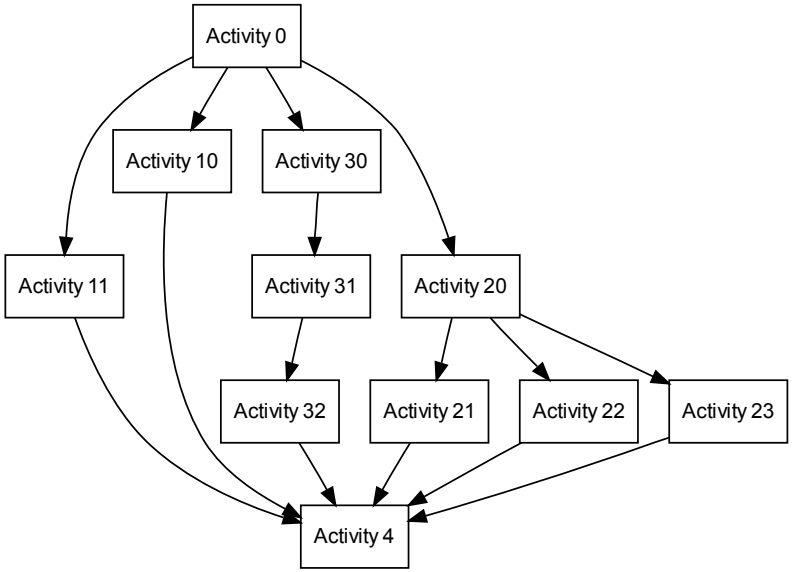

Log composed of 1465 log traces (in H.M., considering only the starting event)

# Results for a real case

Result for
**Heuristics Miner**

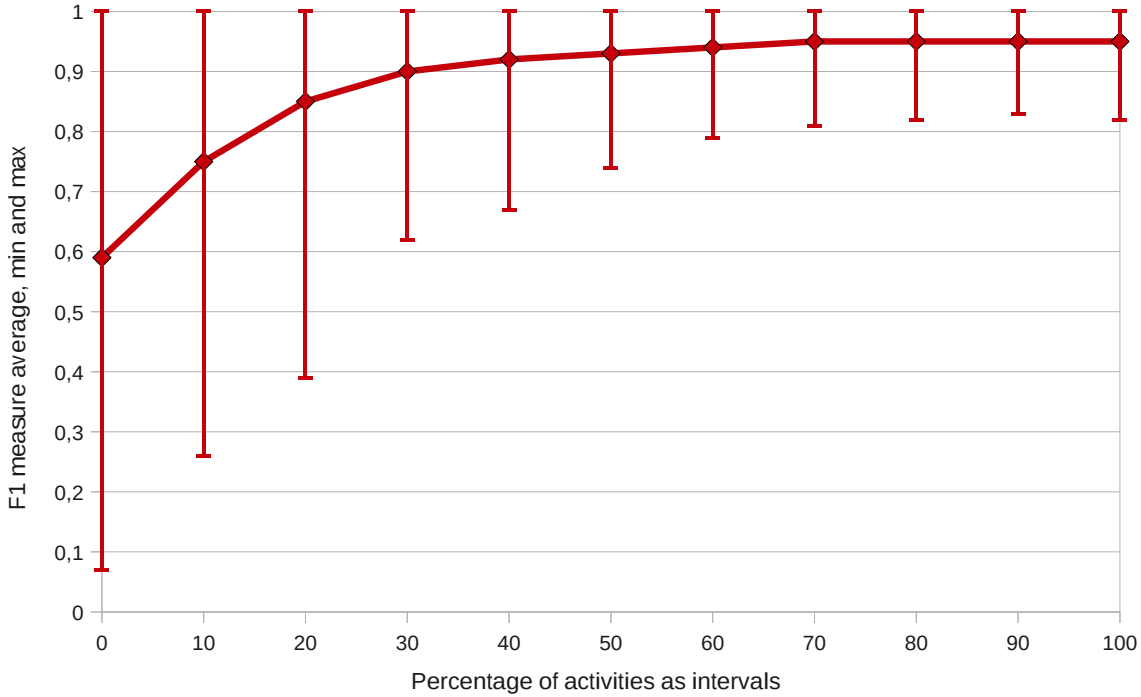



Result for
**Heuristics Miner++**




Log composed of 1465 log traces (in H.M., considering only the starting event)

# Results for an artificial dataset




$$F_1 = 2 \cdot \frac{p \cdot r}{p + r}$$

$$p = \frac{tp}{tp + fp}$$

$$r = \frac{tp}{tp + fn}$$

Test data: 100 random processes logs with 6 activities

- $tp$: correctly mined dependencies
- $fp$: dependencies present in the original model but not in the mined one
- $fn$: dependencies present in mined model but not in the original one

# Conclusions and future works

What we achieved:

- We considered each activity as a time interval
- Added the "notion" of time intervals into the Heuristics Miner algorithm
- The new version of the algorithm is "backward compatible"

Possible future works:

- Test of the algorithm against more (and bigger) processes
- Autonomous identification of best parameters' values
- Support for "noise" into the time intervals, example: