Master Thesis

Process Enhancement by Incorporating Negative Instances in Model Repair

Author: **Kefang Ding**

Email: **kefang.ding@rwth-aachen.de**

Supervisor: Dr. Sebastiaan J. van Zelst

Examiners: Prof. Wil M.P. van der Aalst
Prof. Thomas Rose

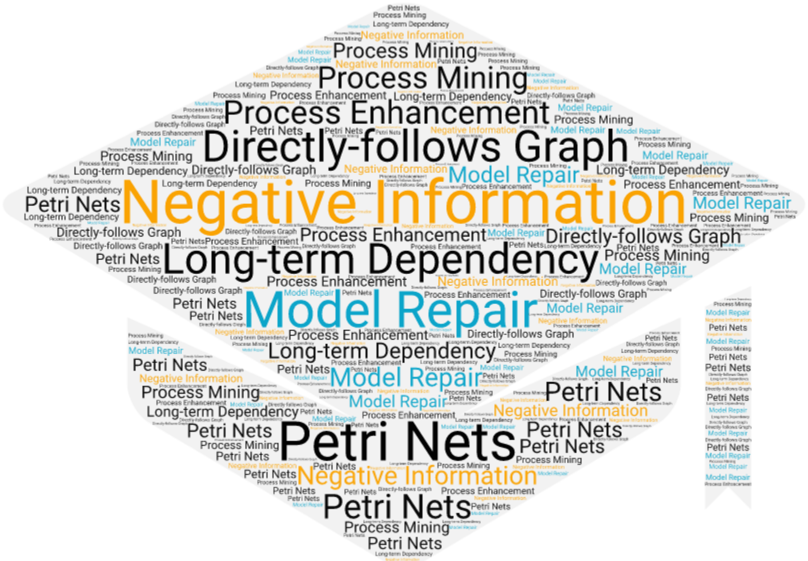
Institute: Lehrstuhl für Process and Data Science

Chair of Process and Data Science

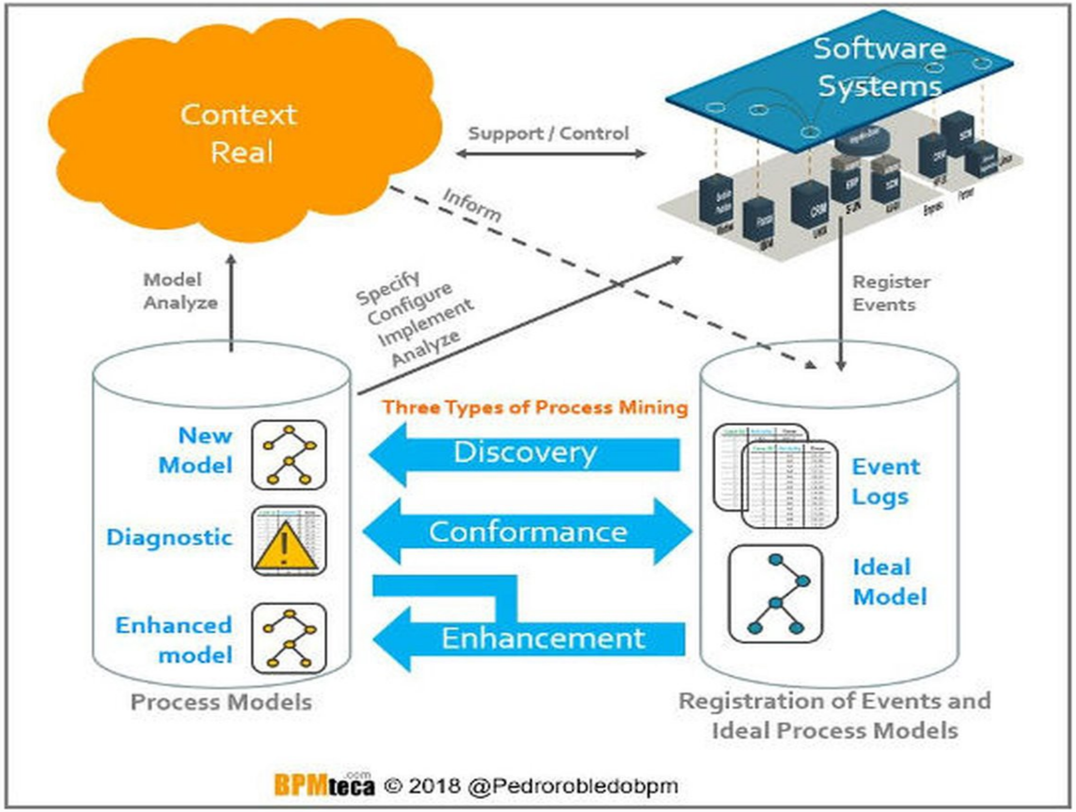
RWTH AACHEN UNIVERSITY

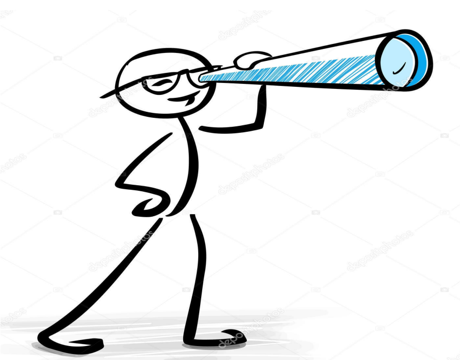
Outline

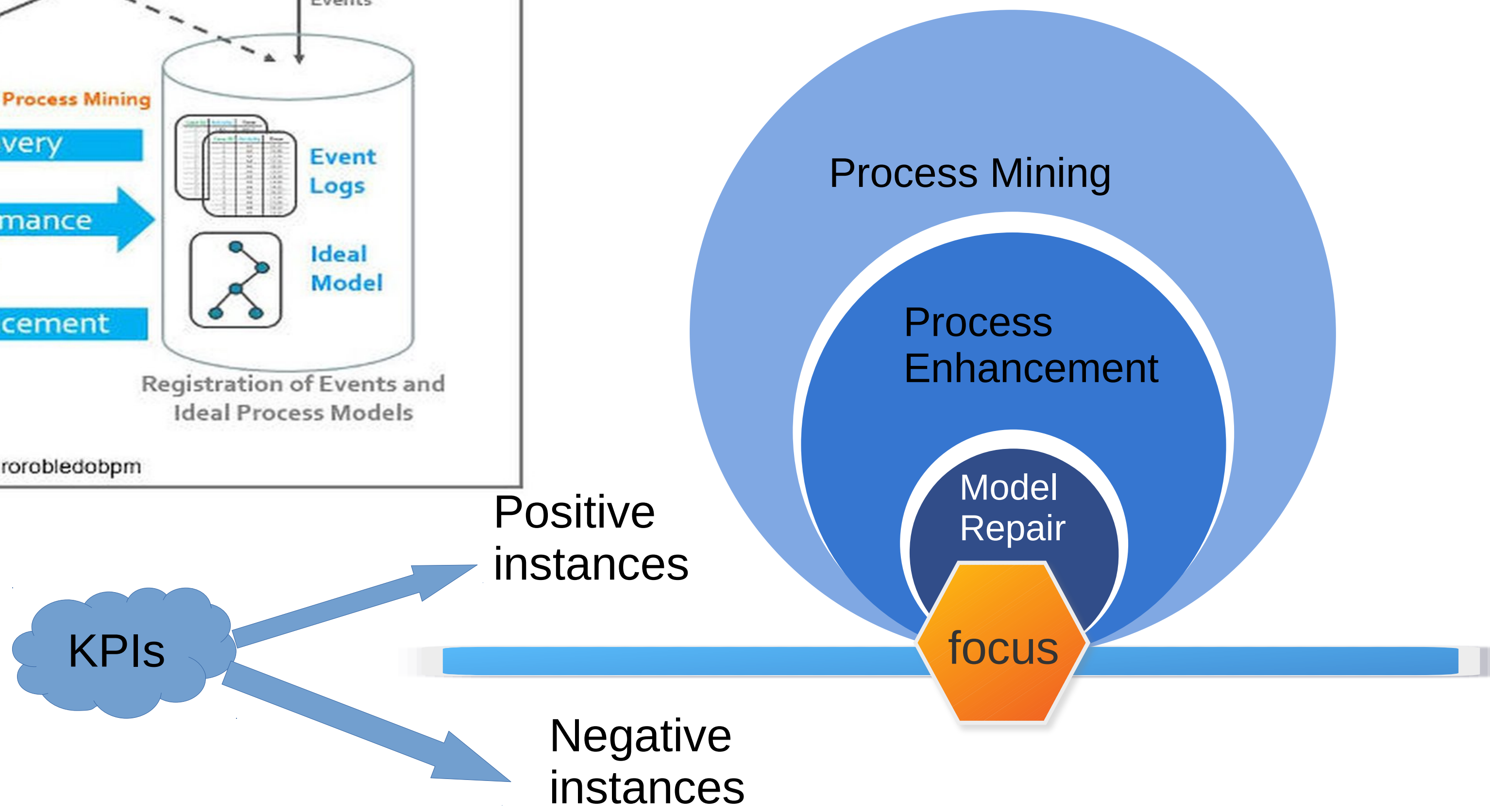
- **Motivation for Research**
  - Scope
  - Related work
  - Motivating examples
- **Problem Definition**
- **Approach**
- **Demo**
- **Evaluation**
- **Conclusion**



Scope







https://medium.com/@pedrorobledobpm/process-mining-plays-an-essential-role-in-digital-transformation-384839236bbe

Literature Review

- **Rediscovery**
  - Inductive Miner
- **Model Repair by Fahland**
  - Deviations
  - Subprocesses
- **Model Repair by Dees**
  - Event log with performance labels
  - Classify deviations
  - Fahland’s repair only on positive deviations

Chair of Process and Data Science

RWTH AACHEN UNIVERSITY

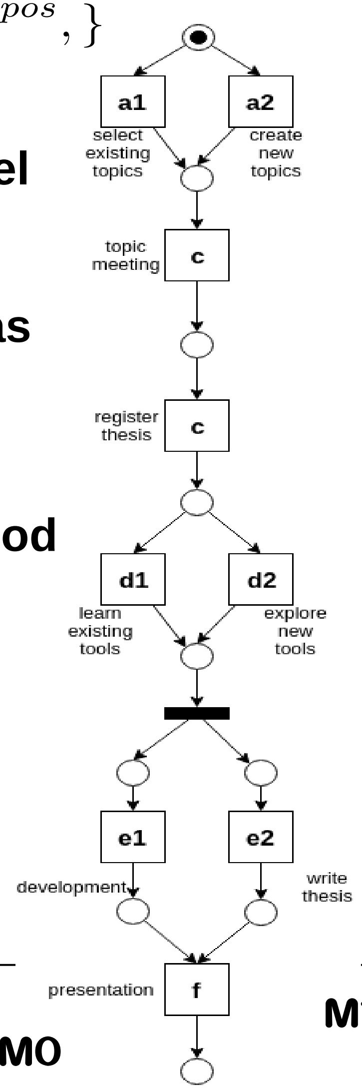
Research Problem – shortcominngs

Positive: x1: write propose
x2: check course requirement

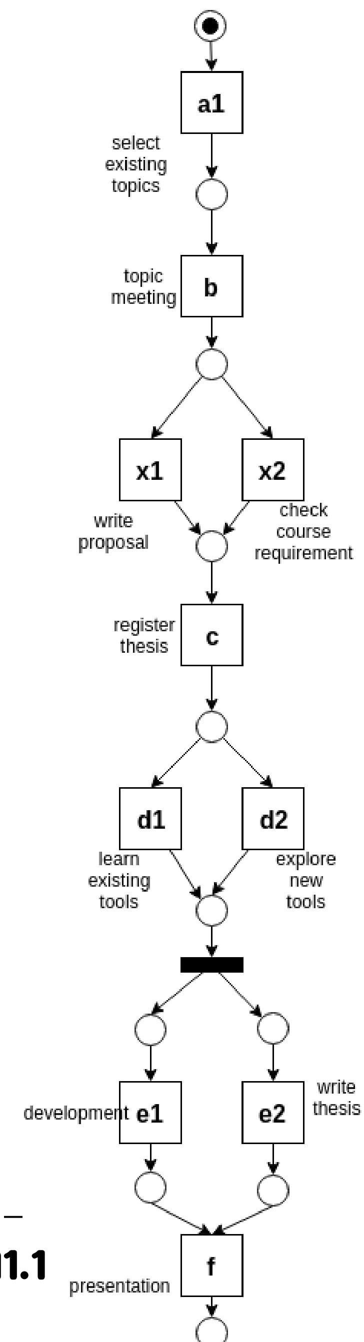
$$L_1 := \{< a1, b, \mathbf{x1}, c, d1, e1, e2, f >^{50,pos}, < a1, b, \mathbf{x2}, c, d2, e1, e2, f >^{50,pos}, \}$$

- **IM not consider reference model**
- **Fahland's: add subprocesses as loops**
- **Dee's: same as Fahland's method**

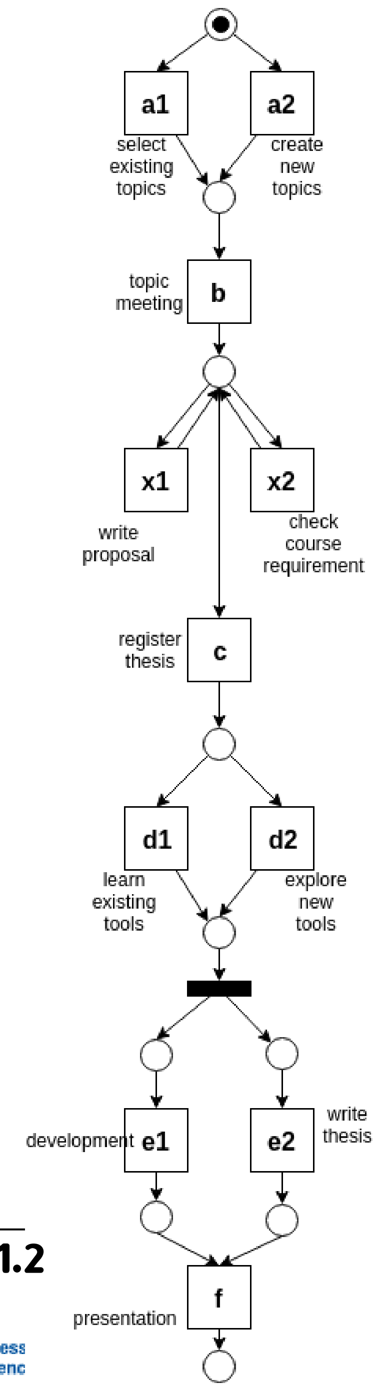
Similarity, or Precision decrease by adding subprocess in loops



M0



M1.1



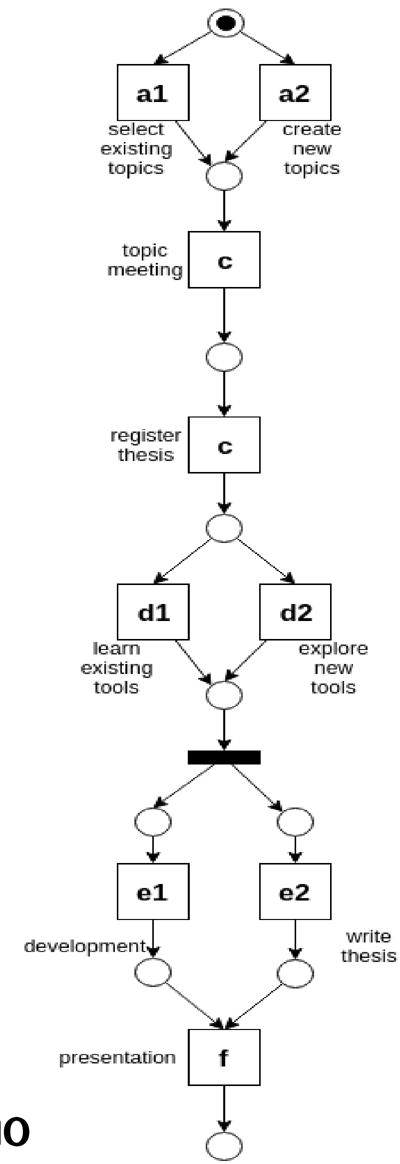
M1.2

Research Problem – shortcomings

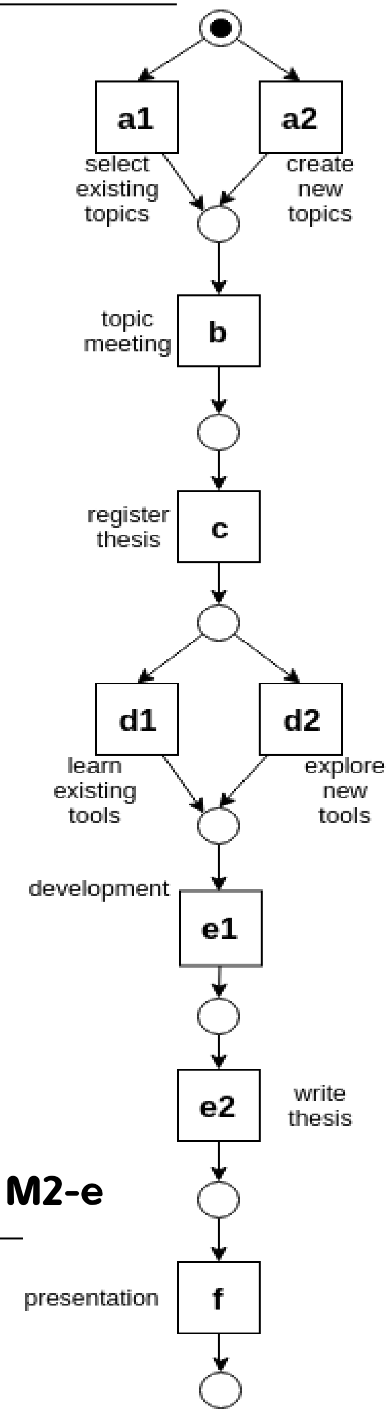
$$
\begin{aligned}
L_2 := \{ &< a1, b, c, d2, \mathbf{e1}, \mathbf{e2}, f >^{30,pos}, \\
&< a2, b, c, d1, \mathbf{e1}, \mathbf{e2}, f >^{20,pos}; \\
&< a2, b, c, d2, \mathbf{e2}, \mathbf{e1}, f >^{10,pos}; \\
&< a1, b, c, d2, \mathbf{e2}, \mathbf{e1}, f >^{20,neg}, \\
&< a1, b, c, d1, \mathbf{e2}, \mathbf{e1}, f >^{20,neg}; \\
&< a2, b, c, d1, \mathbf{e1}, \mathbf{e2}, f >^{5,neg} \}
\end{aligned}
$$

- **IM keeps the model same**
- **Fahland’s keep model same**
- **Dee’s keep model same**

Unable to adapt model with negative information



M0



M2-e

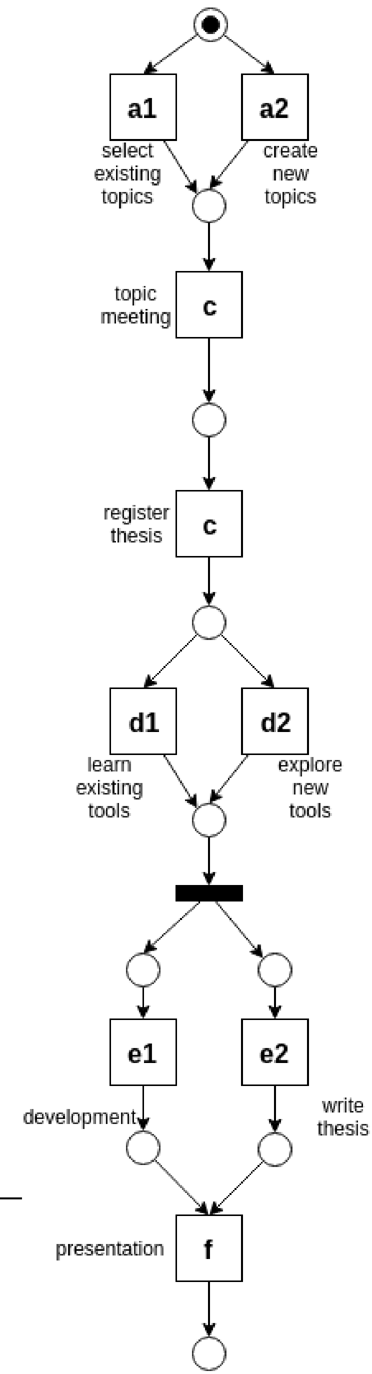
Research Problem – shortcomings

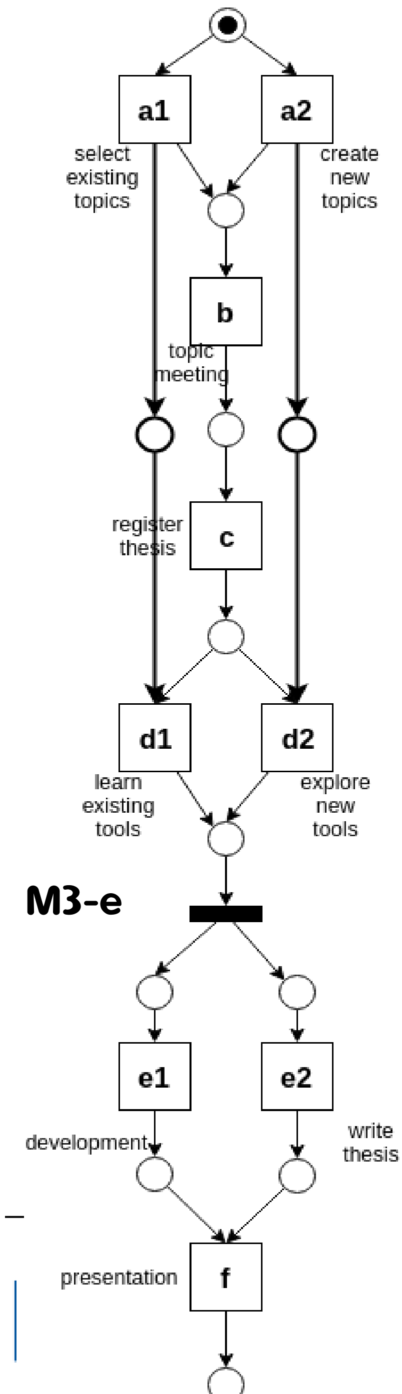
$$
\begin{aligned}
L_3 := \{ &< \mathbf{a1}, b, c, \mathbf{d1}, e1, e2, f >^{50,pos}, \\
&< \mathbf{a2}, b, c, \mathbf{d2}, e1, e2, f >^{50,pos}; \\
&< \mathbf{a1}, b, c, \mathbf{d2}, e1, e2, f >^{50,neg}, \\
&< \mathbf{a2}, b, c, \mathbf{d1}, e1, e2, f >^{50,neg} \}
\end{aligned}
$$

- **Long-term dependency**
  - Choices decides choices
  - Additional places to limit behavior

Unable to detect long-term dependency

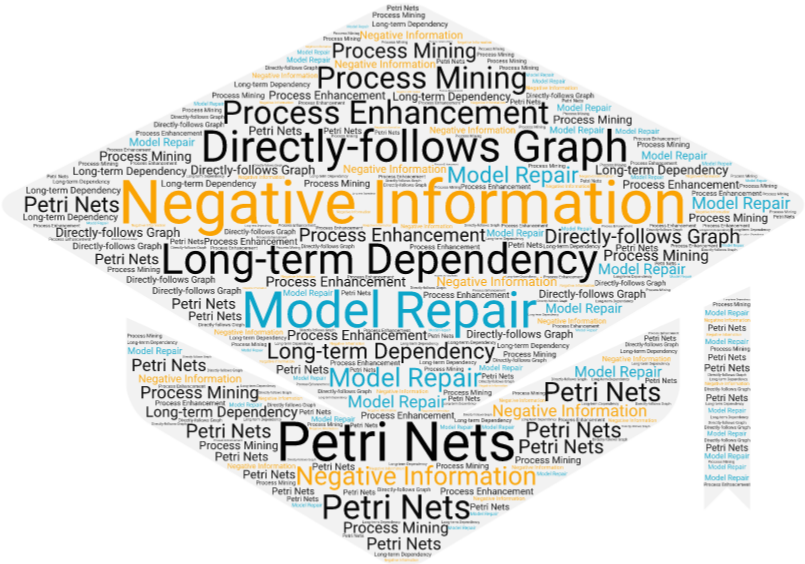
M0





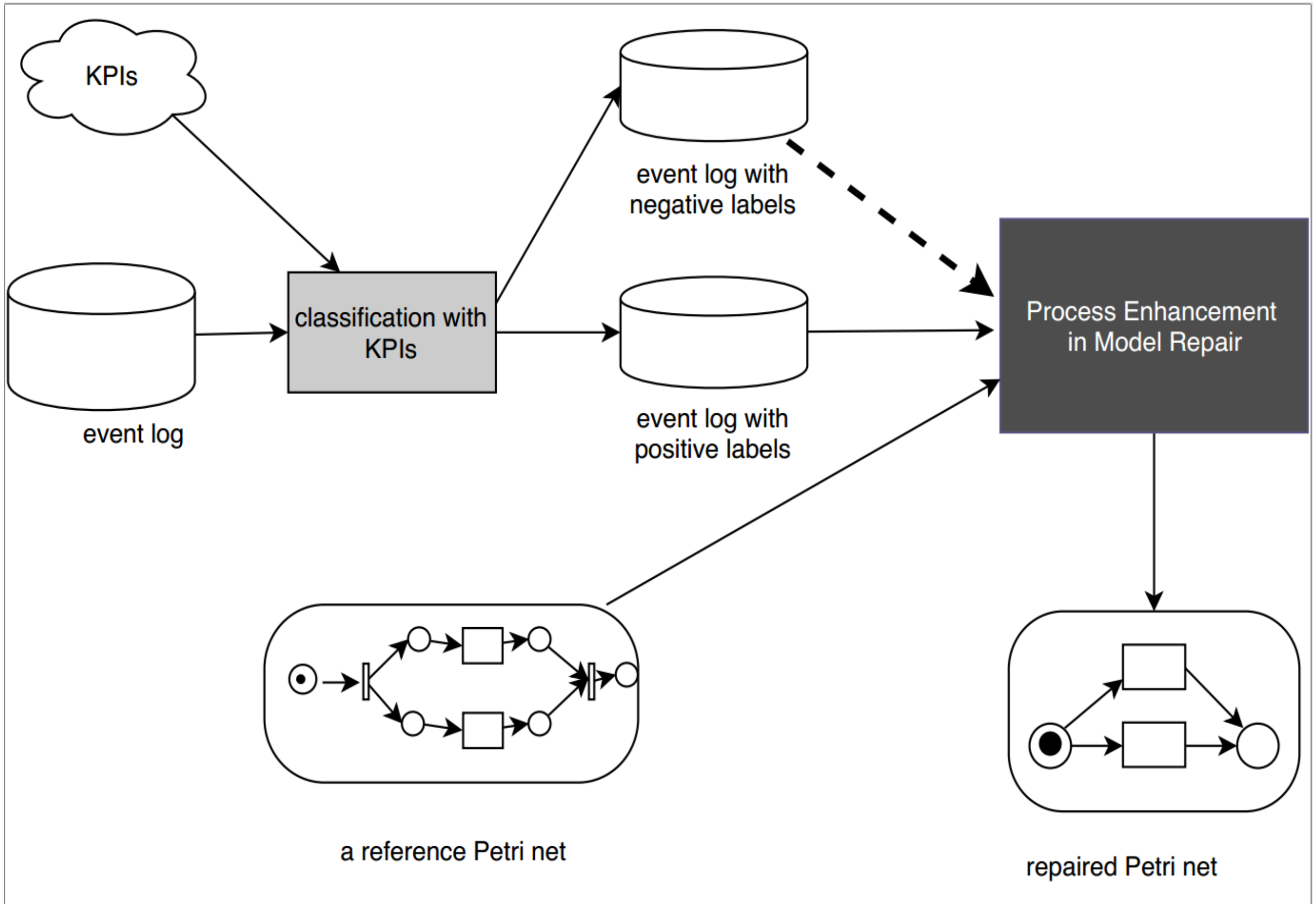
Outline

- Motivation for Research
- Problem Definition
- Approach
- Demo
- Evaluation
- Conclusion



Research Problem

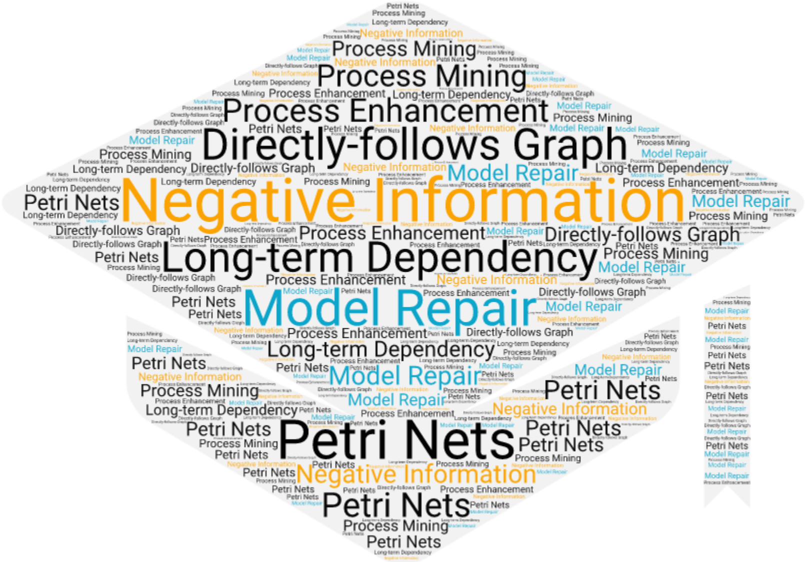
Given an **event log with labels**, a **reference Petri net**, how to incorporate **negative instances** to generate the **repaired Petri net** which supports better performance?



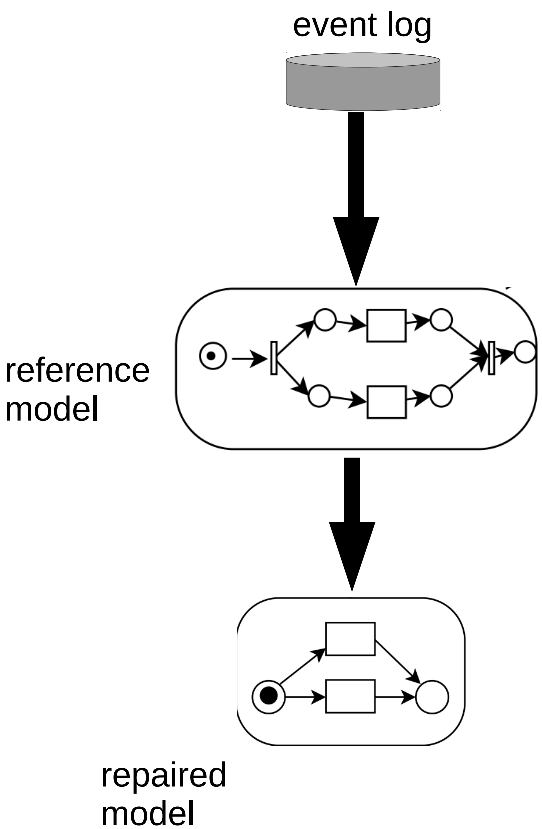
– enforce positive instances
– block negative instances
– similar
– simple
……

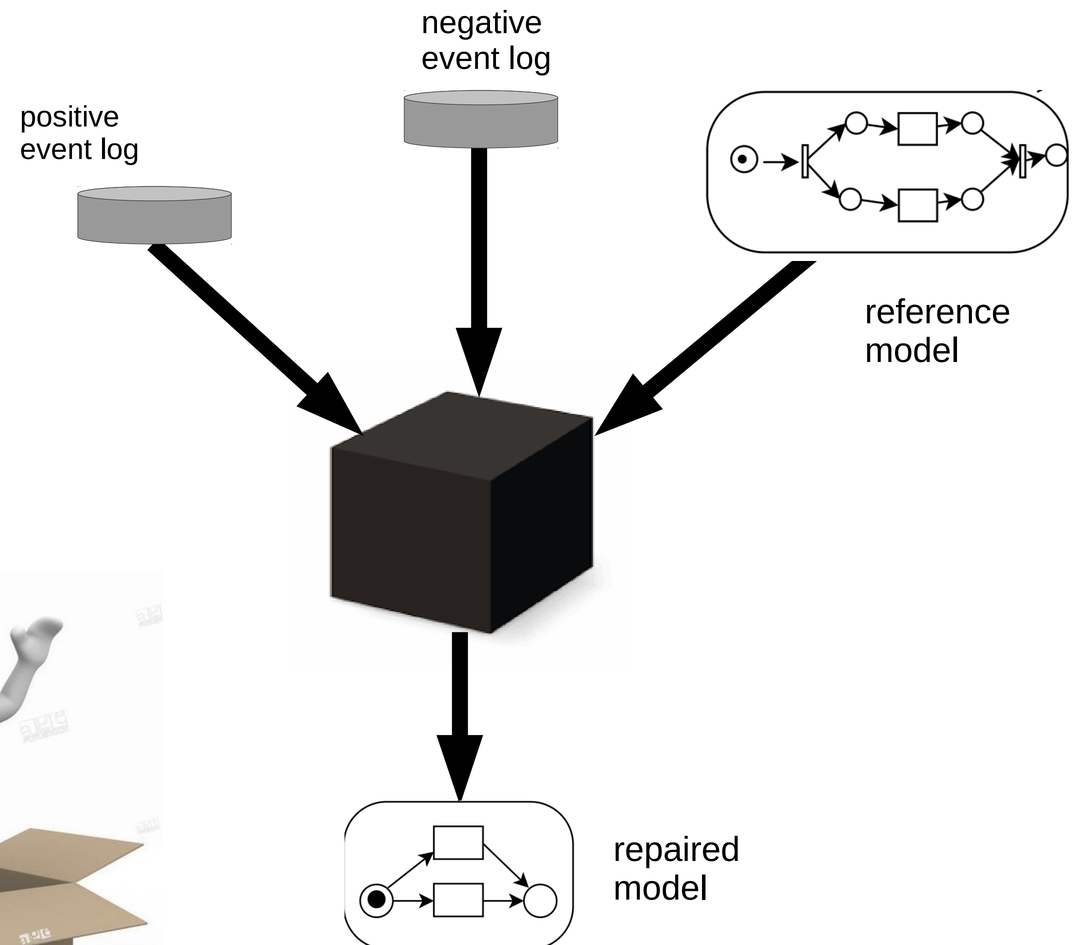
Outline

- **Motivation for Research**
- **Problem Definition**
- **Approach**
  - Framework
  - Data models
  - Modules
- **Demo**
- **Evaluation**
- **Conclusion**

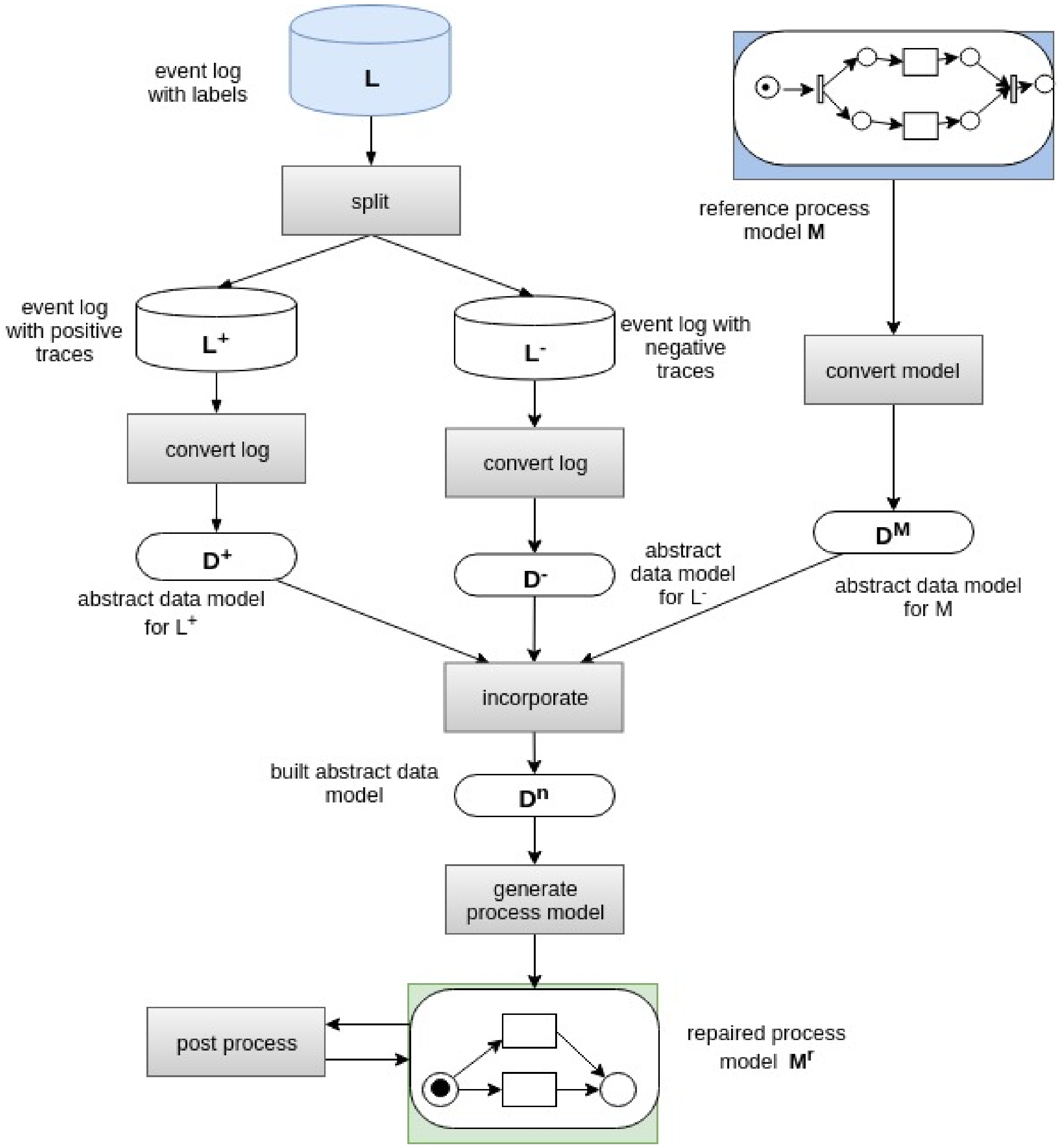


Algorithm – framework





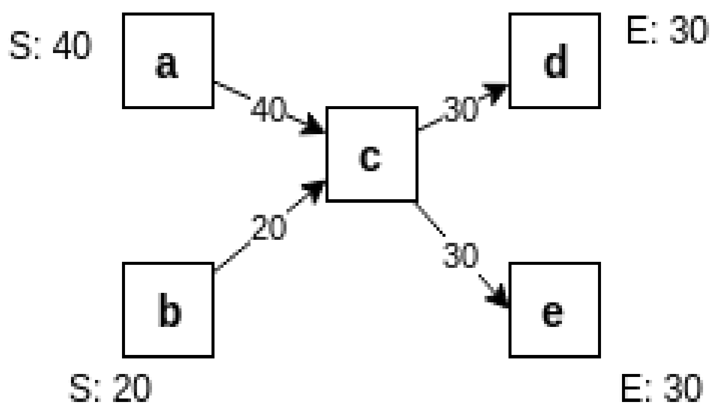
Algorithm – framework



Data Model – directly-follows graph

A directly-follows graph of an event log L is $G(L) = (A, F, A_{start}, A_{end})$ where A is the set of activities in L, $F = \{(a,b) \in A \times A | a >_L b\}$ is the directly-follows relation set, A_{start}, A_{end} are the set of start and end activities respectively,

$$A_{start} = \{a | \exists \sigma \in L, a = \sigma(1)\}, A_{end} = \{a | \exists \sigma \in L, a = \sigma(|\sigma|)\}$$



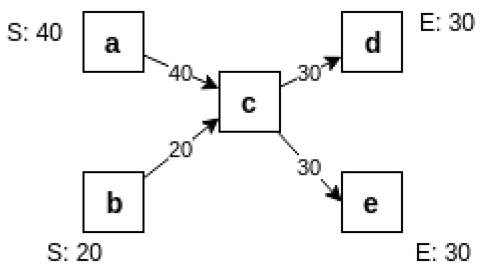
$$L = \{<a,c,d>^{20}, <b,c,e>^{10}, <a,c,e>^{20}, <b,c,d>^{10}\}$$

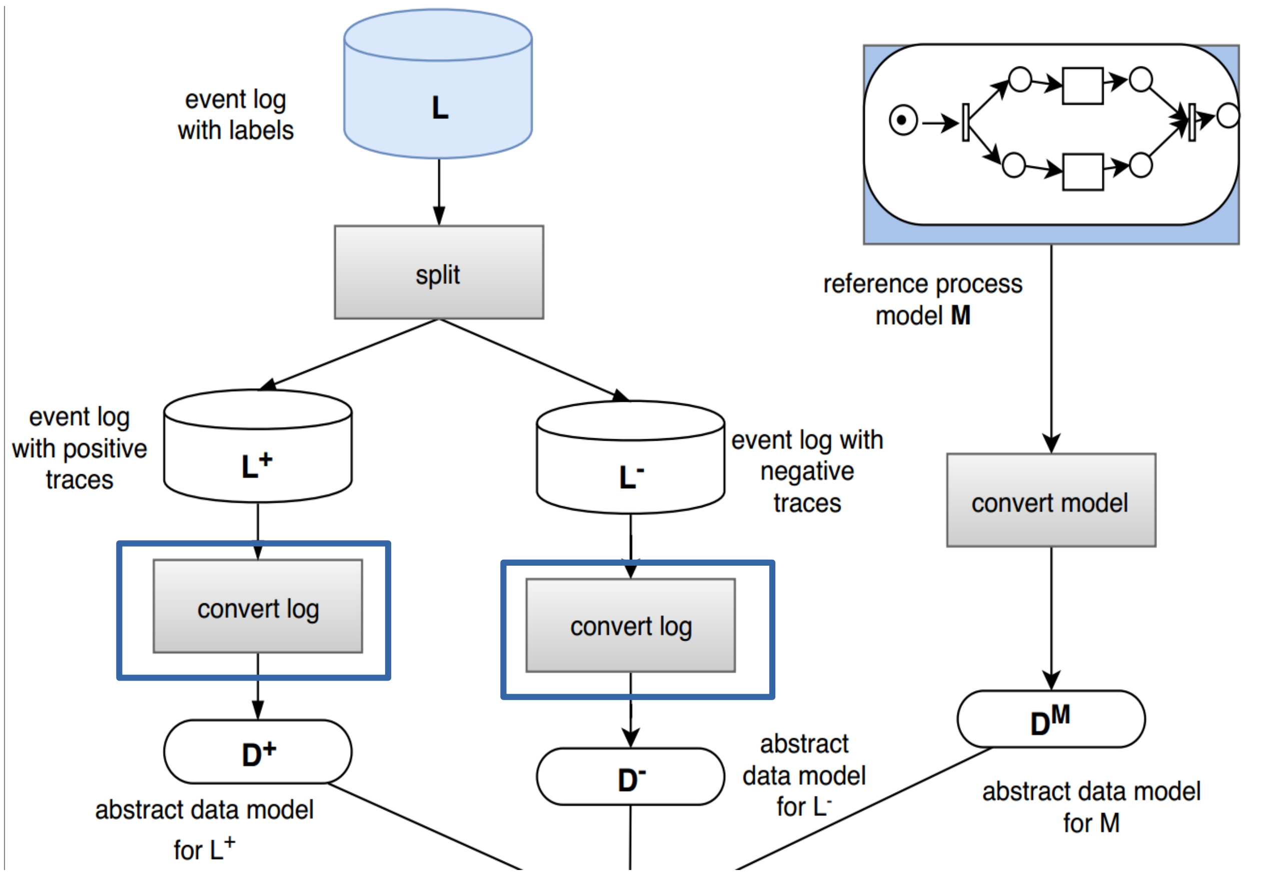
Convert to directly-follows graph

Event log

- Directly-follows relation
- Existing plugin

$$L=\{\langle a,c,d\rangle^{20},\langle b,c,e\rangle^{10},\langle a,c,e\rangle^{20},\langle b,c,d\rangle^{10}\}$$

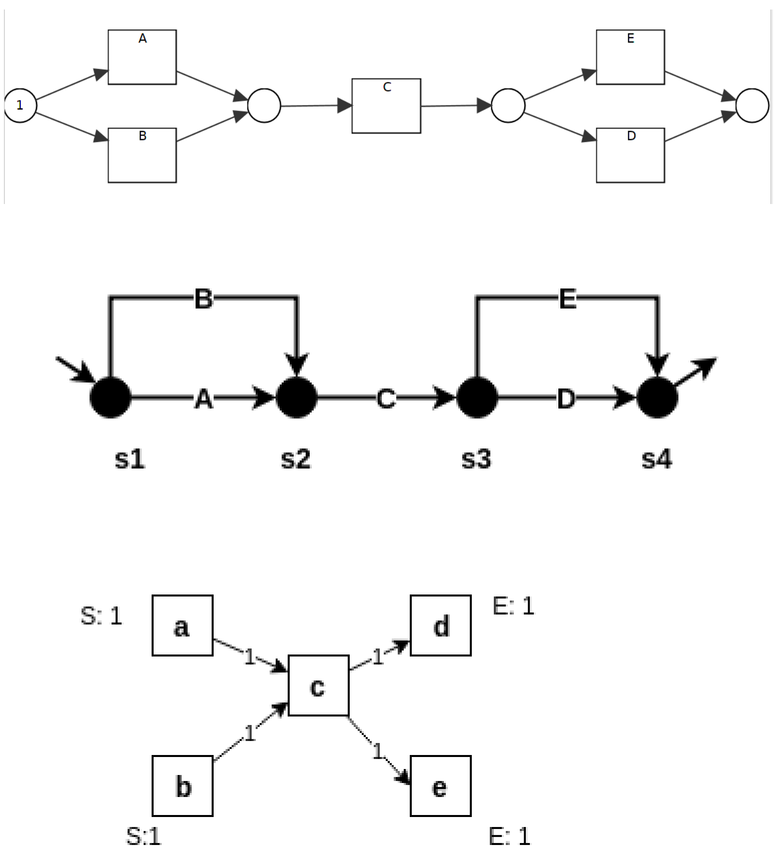


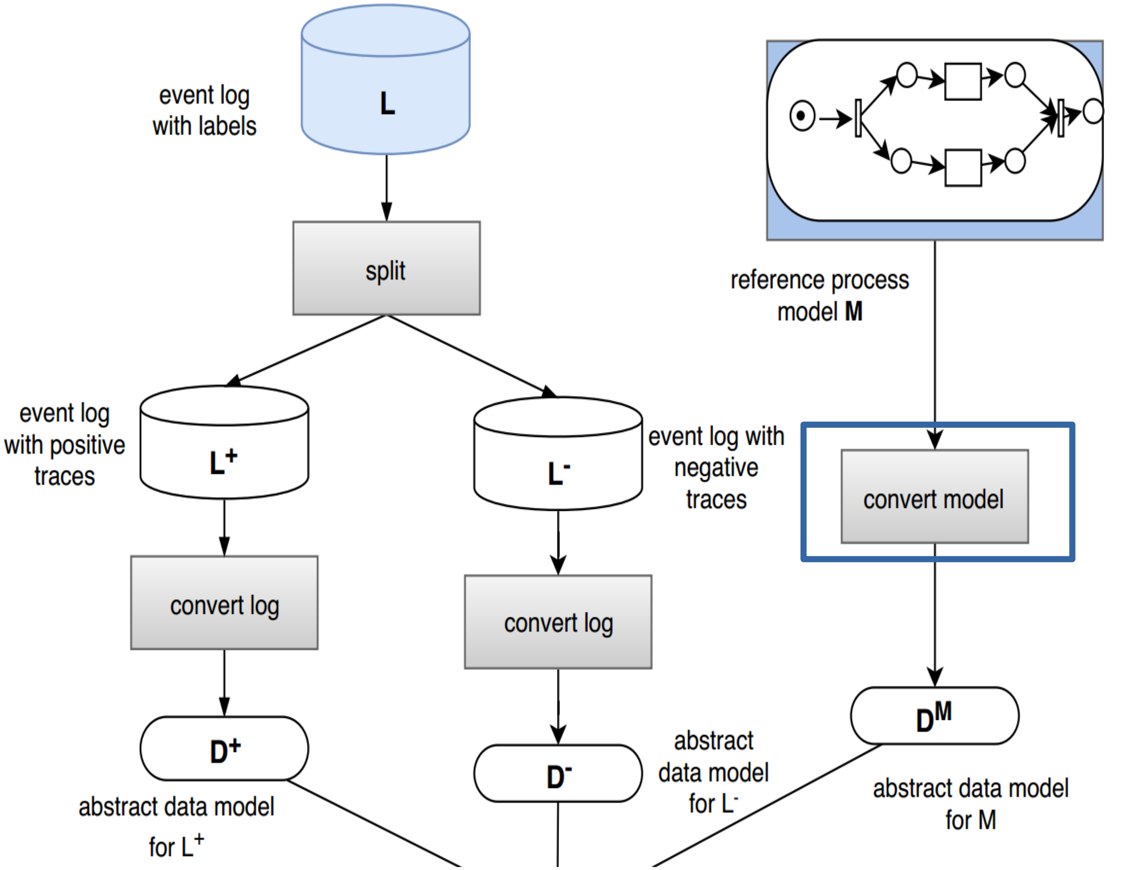


Convert to directly-follows graph

Petri net

- Transition System
- Transitions before and after states





Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

Data Model

- **Unification of cardinality**
  - Models from existing model, positive and negative event log
  - For any directly-follows relation

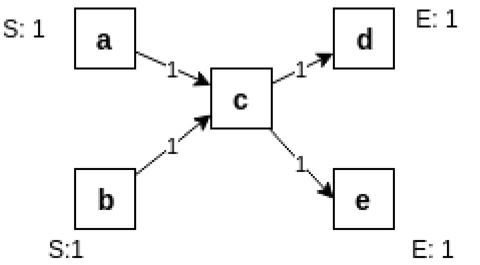
$$u(a,b) = \frac{c(a,b)}{\sum_{(a,b')\in F} c(a,b')}$$

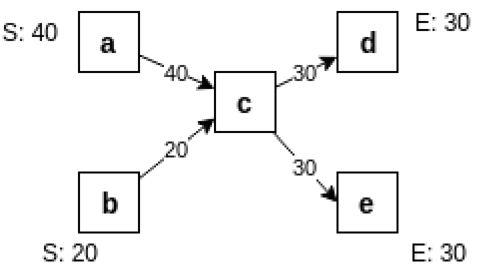
  - For any start activity
  -

$$u(a) = \frac{c(a)}{\sum_{a'\in A_{start}} c(a')}$$

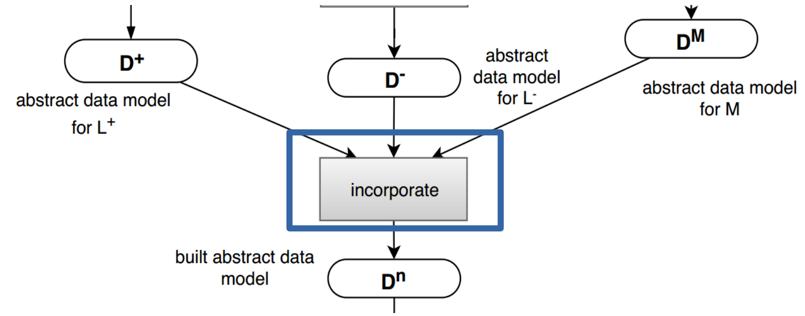
  - For any end activity

$$u(a) = \frac{c(a)}{\sum_{a'\in A_{end}} c(a')}$$





Incorporate Data Models



- **Incorporate method**
  - For any directly-follows relation

    $$u^n(a,b) = u^M(a,b) + u^+(a,b) - u^-(a,b)$$

  - For any start activity,

    $$a \in A^M_{start} \cup A^+_{start} \cup A^-_{start}, u^n(a) = u^M(a) + u^+(a) - u^-(a)$$

  - For any end activity,

    $$a \in A^M_{end} \cup A^+_{end} \cup A^-_{end}, u^n(a) = u^M(a) + u^+(a) - u^-(a)$$

- **Weighted value**
  - Three control weights w^+, w^-, w^M

    $$u^n_w(a,b) = w^M \cdot u^M(a,b) + w^+ \cdot u^+(a,b) - w^- \cdot u^-(a,b)$$

$w^M = 1, w^+ = w^- = 0$ ➡ Keep the reference model

$w^+ = 1, w^M = w^- = 0$ ➡ Mine a new model from positive instances

Generate Petri net

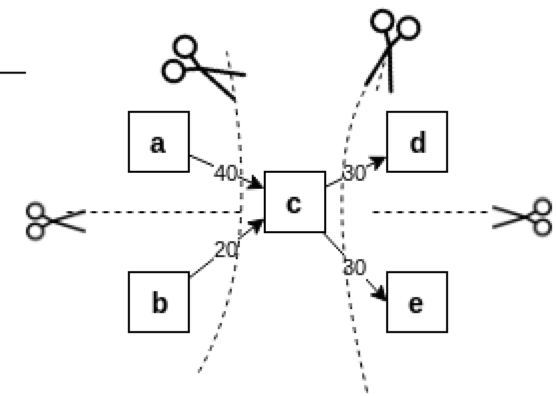
- **Choose directly-follows relation**

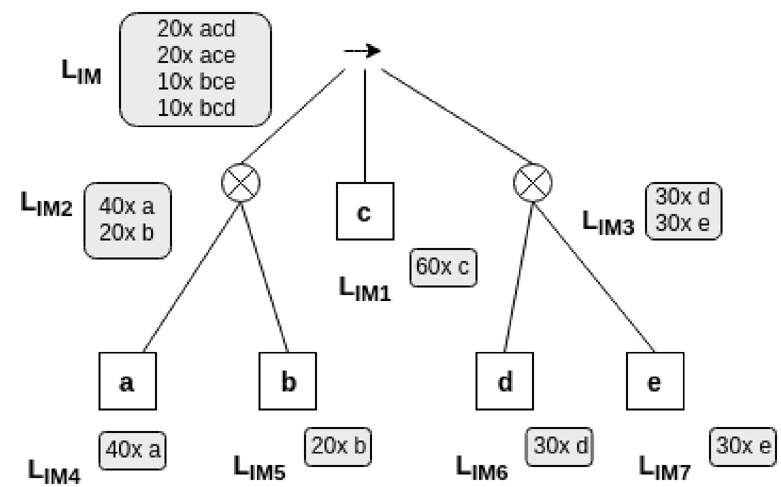
$$u_w^n(a,b) > t$$

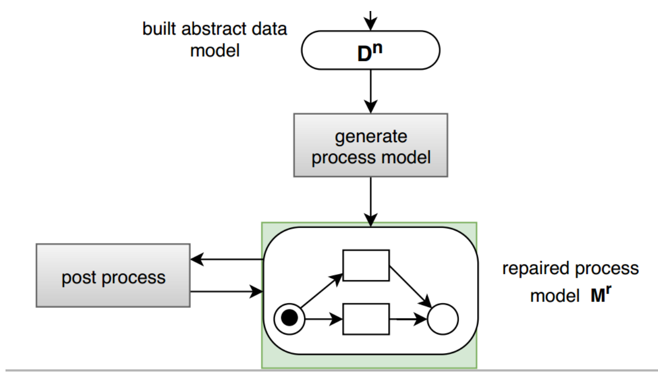
  - t=0

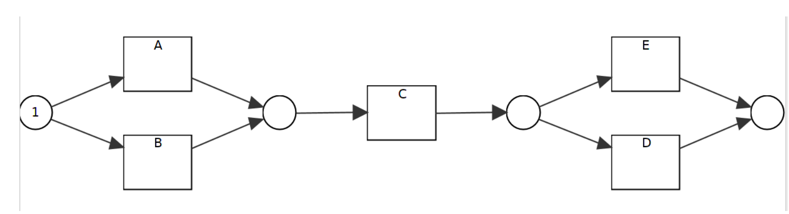
- **Assign normal cardinality back**
  - $c^n(a,b) = u_w^n(a,b) \cdot (|L^+| + |L^-|)$

- **IM discovery algorithm**
  - Directly-follows graph
  - Process tree
  - Petri net





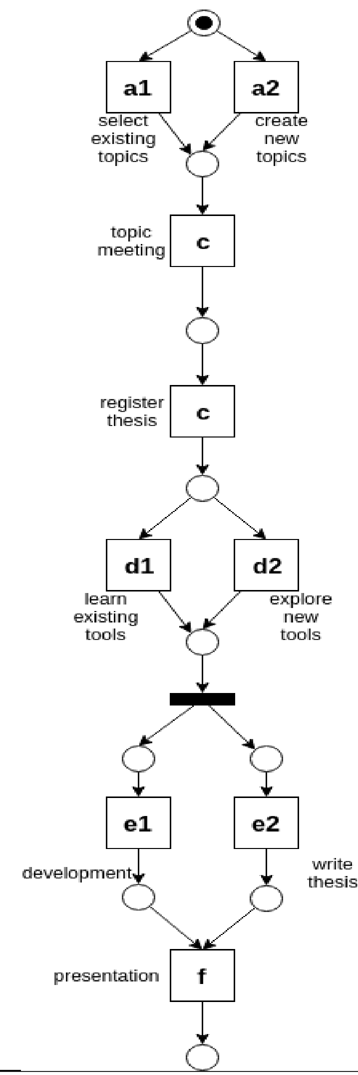


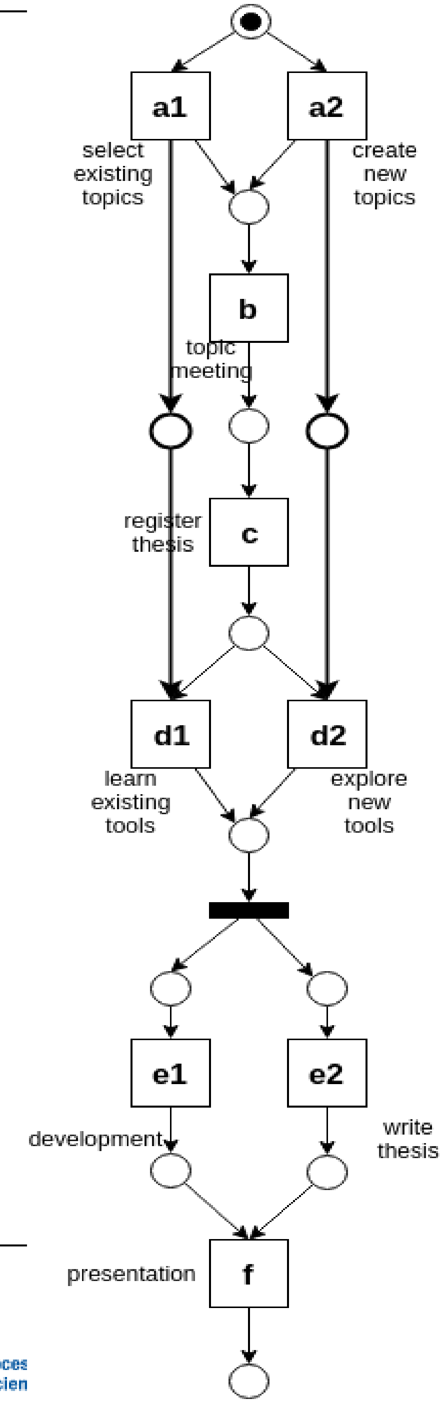


Post Process Petri net

- **Long-term dependency**
  - Choices dependency
  - Exclusive choices: xor blocks/braches
  - Strong correlation

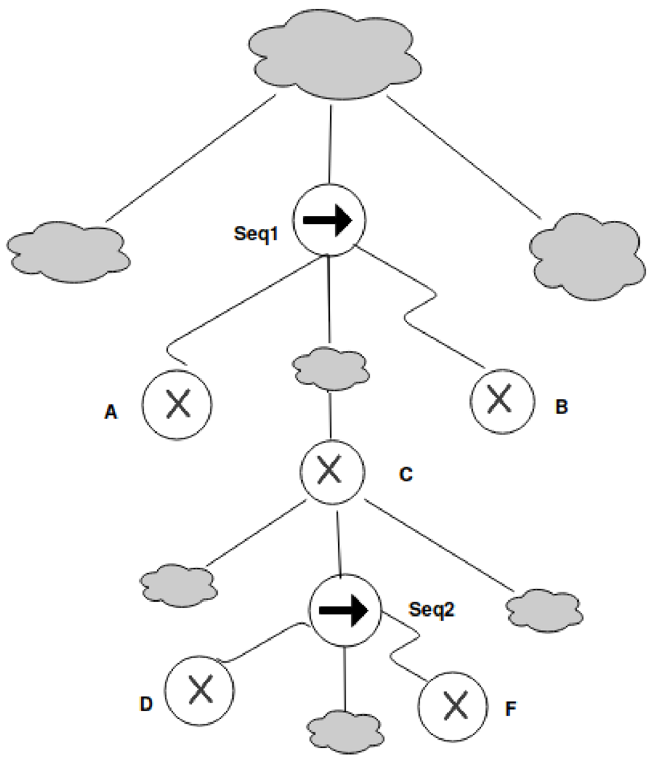
$$
\begin{aligned}
L_3 := \{ & <\mathbf{a1}, b, c, \mathbf{d1}, e1, e2, f>^{50,pos}, \\
& <\mathbf{a2}, b, c, \mathbf{d2}, e1, e2, f>^{50,pos}; \\
& <\mathbf{a1}, b, c, \mathbf{d2}, e1, e2, f>^{50,neg}, \\
& <\mathbf{a2}, b, c, \mathbf{d1}, e1, e2, f>^{50,neg} \}
\end{aligned}
$$





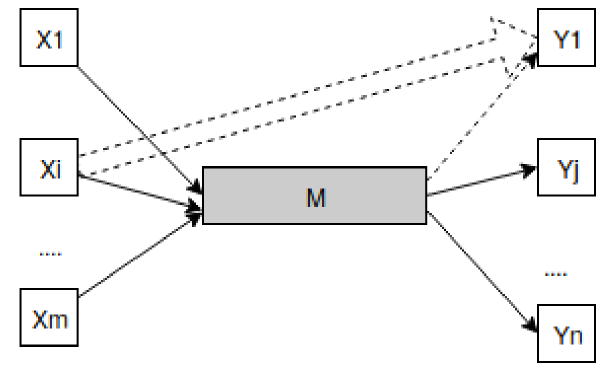
Post Process Petri net

- **Long-term dependency**
  - Process tree as intermediate result
  - Exclusive choices
  - In order
    - Least common ancestor is Seq
      - A < C < B, D<F
    - In same level
      - A,B,C pair
      - D,F pair
  - Strong correlation



Post Process Petri net

- **Long-term dependency**
  - Strong correlation
    - Frequency



Frequency of an xor branch X_i in an event log L is the count of traces which replay this xor branch, $f : X \rightarrow N, f_L(X) = \sum_{\sigma \in L} |\{\sigma | \sigma \models X\}|$

Frequency of multiple xor branches is $\mathrm{f}_L(X_1, X_2, ..., X_n) = \sum_{\sigma \in L} |\{\sigma | \forall X_i, \sigma \models X_i\}|$

    - Correlation over t

$$d(X_i, Y_j) = w^+ \cdot d^+(X_i, Y_j) - w^- \cdot d^-(X_i, Y_j) > t$$

$$d^+(X_i, Y_j) = \frac{f_{L^+}(X_i, Y_j)}{\sum_{Y^k \in T, k \neq j} f_{L^+}(X_i, Y^k)} \quad d^-(X_i, Y_j) = \frac{f_{L^-}(X_i, Y_j)}{\sum_{Y^k \in T, Y^k \neq X_i} f_{L^-}(X_i, Y^j)}$$

Algorithm – add long-term dependency

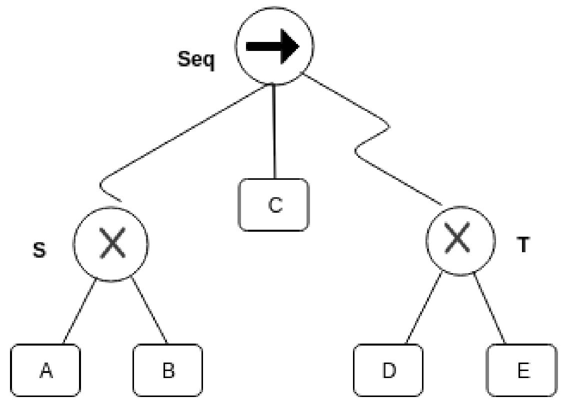
- **Long-term dependency Situations**

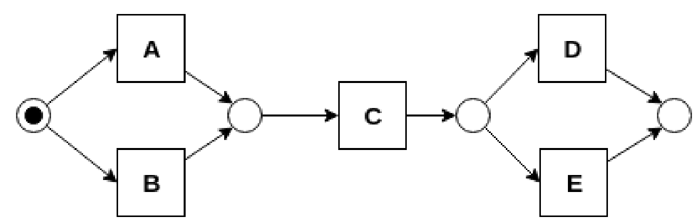
$$LT_S := \{X_i | \exists Y_j, X_i \rightsquigarrow Y_j \in LT\}$$

$$LT_T := \{Y_j | \exists X_i, X_i \rightsquigarrow Y_j \in LT\}$$

1. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow D, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}, |LT| = |S| \cdot |T|$.
2. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}$ $LT_S = S$ and $LT_T = T, |LT| < |S| \cdot |T|$.
3. $LT = \{A \rightsquigarrow D, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}$ $LT_S = S$ and $LT_T = T, |LT| < |S| \cdot |T|$.
4. $LT = \{A \rightsquigarrow D, B \rightsquigarrow D\}$.
   $LT_S = S, LT_T \subsetneq T$.
5. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T = T$.
6. $LT = \{A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T \subsetneq T$.
7. $LT = \emptyset$

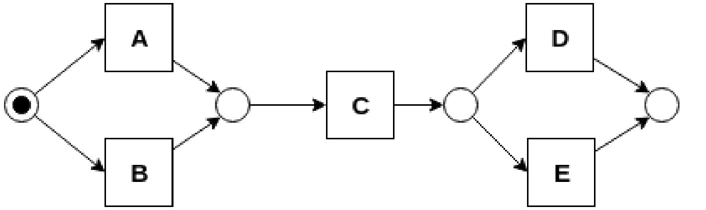
   - Situation 1 is full dependency ==> no consideration
   - Situation 7 is empty. ==> no consideration



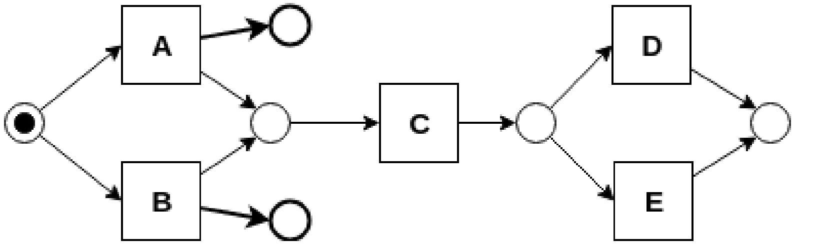


Algorithm – add long-term dependency

- How to express on Petri net
  - ✓ Add silent transition
    - Add control place as post-place post after S

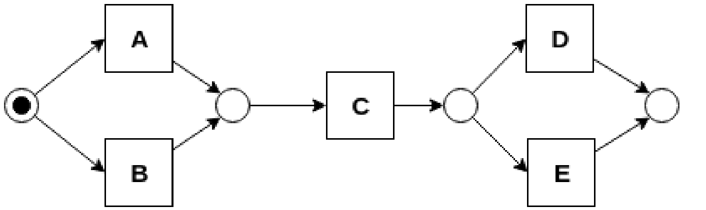


$$LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}.$$

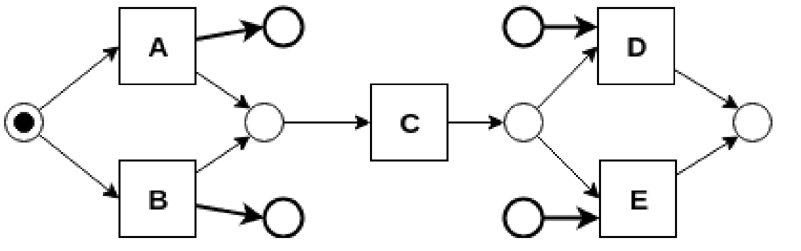


Algorithm – add long-term dependency

- How to express on Petri net
  - ✔ Add silent transition
    - Add control place as post-place post after S
    - Add control place as pre-place before T

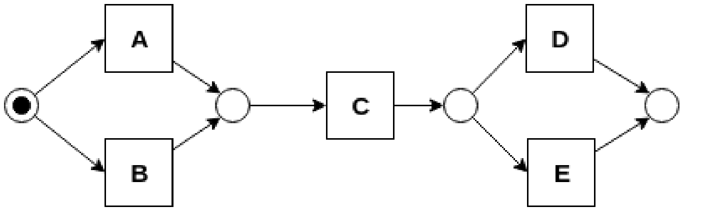


$$LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}.$$

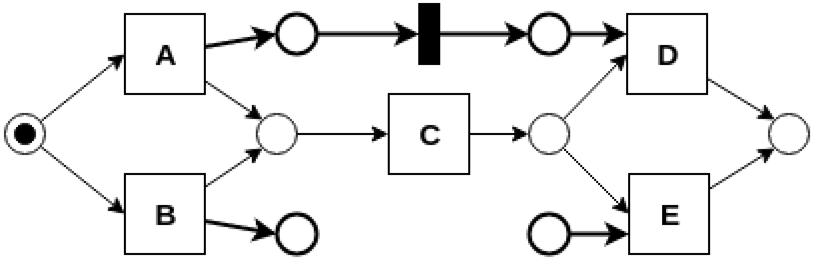


Algorithm – add long-term dependency

- How to express on Petri net
  - Add silent transition
    - Add control place as post-place post after S
    - Add control place as pre-place before T
    - Add silent transitions for each long-term dependency

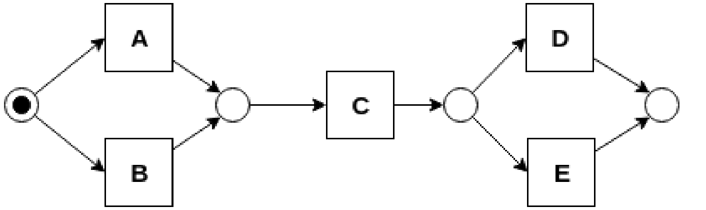


$$LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}.$$

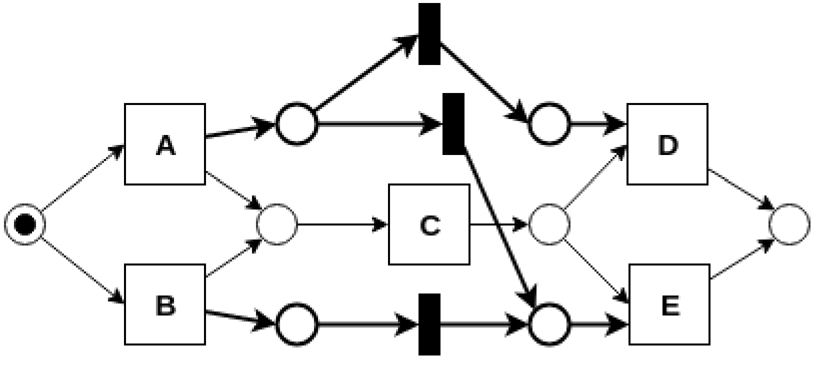


Algorithm – add long-term dependency

- How to express on Petri net
  - ✓ Add silent transition
    - Add control place as post-place post after S
    - Add control place as pre-place before T
    - Add silent transitions for each long-term dependency



$$LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}.$$



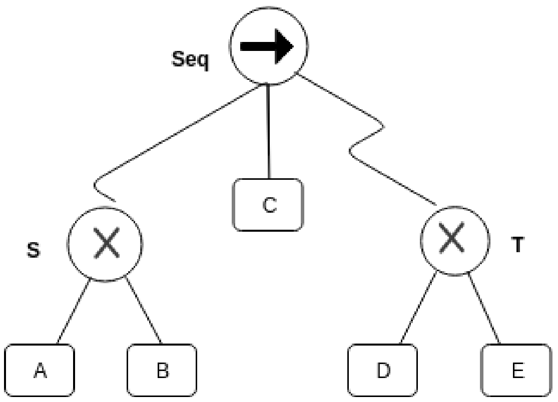
Algorithm – add long-term dependency

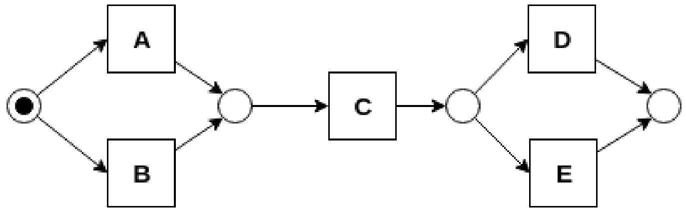
- **Long-term dependency Situations**

$$LT_S := \{X_i | \exists Y_j, X_i \rightsquigarrow Y_j \in LT\}$$

$$LT_T := \{Y_j | \exists X_i, X_i \rightsquigarrow Y_j \in LT\}$$

1. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow D, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}, |LT| = |S| \cdot |T|$.
2. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}$ $LT_S = S$ and $LT_T = T, |LT| < |S| \cdot |T|$.
3. $LT = \{A \rightsquigarrow D, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}$ $LT_S = S$ and $LT_T = T, |LT| < |S| \cdot |T|$.
4. $LT = \{A \rightsquigarrow D, B \rightsquigarrow D\}$.
   $LT_S = S, LT_T \subsetneq T$.
5. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T = T$.
6. $LT = \{A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T \subsetneq T$.
7. $LT = \emptyset$





   - Situation 1 is full dependency ==> no consideration
   - Situation 7 is empty. ==> no consideration

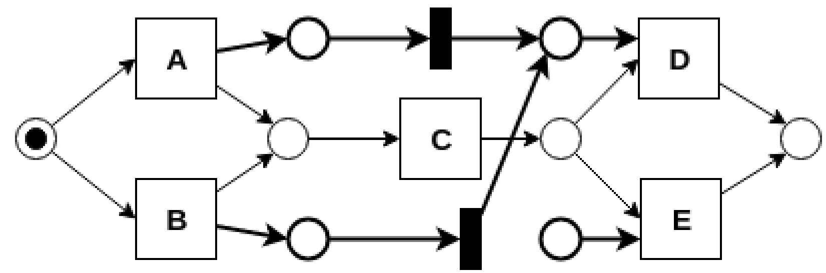
Algorithm – add long-term dependency

- **Express on Petri net**

  4. $LT = \{A \rightsquigarrow D, B \rightsquigarrow D\}$.
     $LT_S = S, LT_T \subsetneq T$.

  5. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E\}$.
     $LT_S \subsetneq S, LT_T = T$.

  6. $LT = \{A \rightsquigarrow E\}$.
     $LT_S \subsetneq S, LT_T \subsetneq T$.



Sound:
- ✓ Safeness.
- ✓ Proper completion.
- ✓ Option to complete.
- ✓ No dead parts.

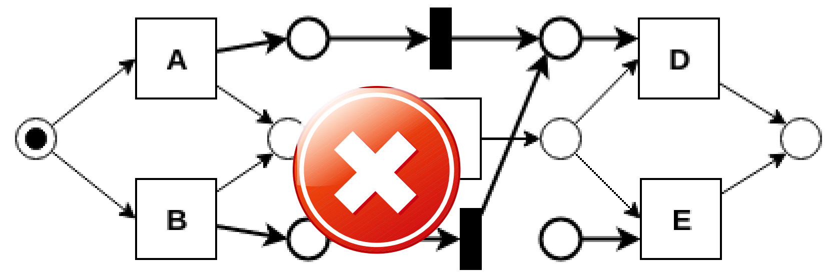
Algorithm – add long-term dependency

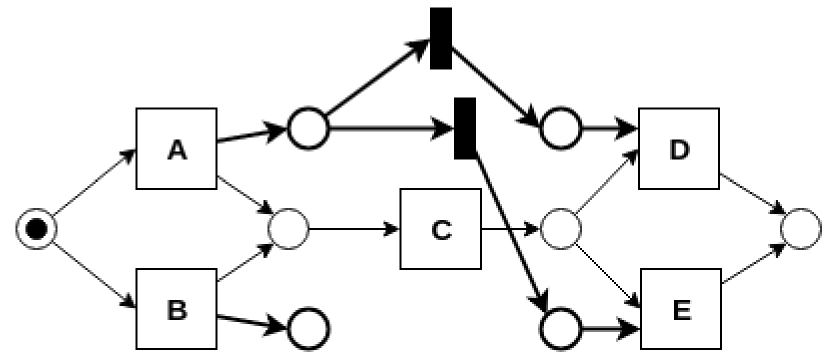
- **Express on Petri net**

4. $LT = \{A \rightsquigarrow D, B \rightsquigarrow D\}$.
   $LT_S = S, LT_T \subsetneq T$.

5. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T = T$.

6. $LT = \{A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T \subsetneq T$.





Sound:
- ✓ Safeness.
- ✓ Proper completion.
- ✓ Option to complete.
- ✓ No dead parts.

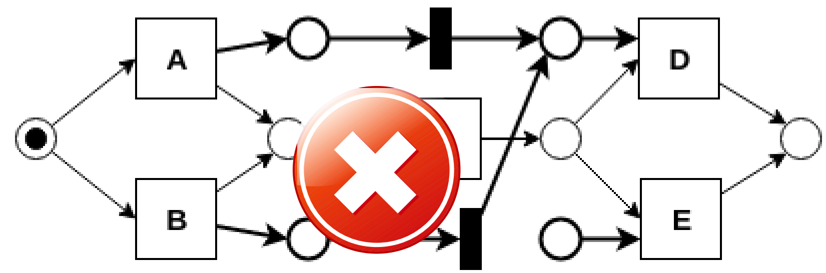
Algorithm – add long-term dependency

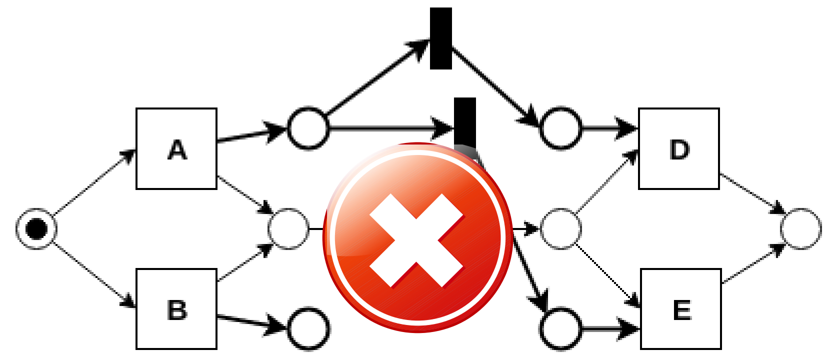
- **Express on Petri net**

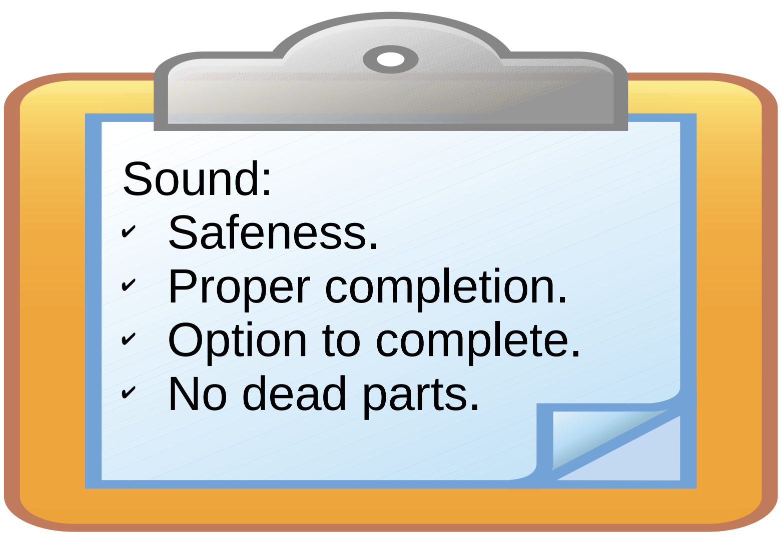
4. $LT = \{A \rightsquigarrow D, B \rightsquigarrow D\}$.
   $LT_S = S, LT_T \subsetneq T$.

5. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T = T$.

6. $LT = \{A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T \subsetneq T$.







Petri net – soundnes

Sound:

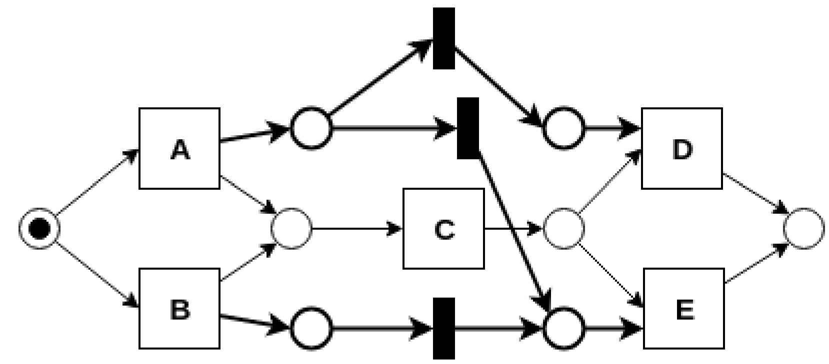
- Safeness.
  Places cannot hold multiple tokens at the same time.
- Proper completion.
  If the sink place is marked, all other places are empty.
- Option to complete.
  It is always possible to reach the final marking from any reachable marking.
- No dead parts.
  For any transition, there exists a path from source to sink place through it.

Soundness proof

Given $S = \{X_1, X_2, ...X_m\}$ and $T = \{Y_1, Y_2, ...Y_n\}$ with $LT = \{X_i \rightsquigarrow Y_j | 1 \le i \le m, 1 \le j \le n\}$. W.l.o.g., X_i is fired.

The marking distribution is

$$M(p_{X_i}) = 1; \quad \forall p_{X_{i'}} \in P_S, i' \neq i, M(p_{X_{i'}}) = 0$$



Type 1 $LT_S = S, LT_T = T$

soundness

- Safeness.

$$\sum M(p_{X_i}) \le 1, \sum M(p_{Y_j}) \le 1$$

- Proper completion.

After firing Y_j, $\sum M(p_{X_i}) = 0, \sum M(p_{Y_j}) = 0$

- Option to complete & No dead parts.

$\forall X_i \in S$ is enabled at beginning

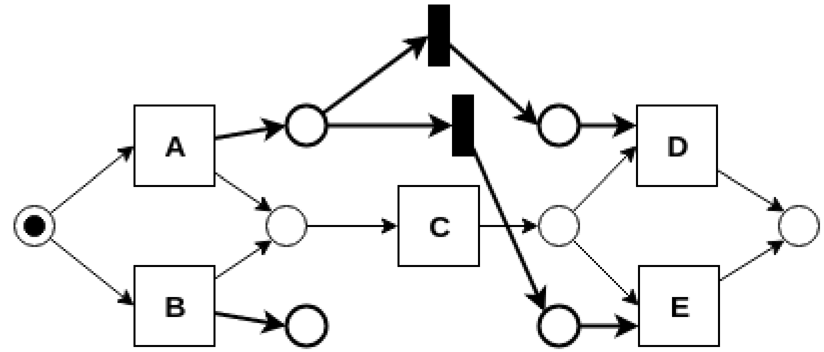
$\forall X_i \in S$, since $LT_S = S$, $\Rightarrow \exists Y_j \in T, X_i \rightsquigarrow Y_j$,

ϵ is enabled with $p_{X_i} \rightarrow \epsilon \rightarrow p_{Y_j}$

$\forall Y_j \in T$, since $LT_T = T$, $\Rightarrow \exists X_i \in S, X_i \rightsquigarrow Y_j$

Soundness proof

Given $S = \{X_1, X_2, \ldots X_m\}$ and $T = \{Y_1, Y_2, \ldots Y_n\}$ with $LT = \{X_i \rightsquigarrow Y_j | 1 \leq i \leq m, 1 \leq j \leq n\}$. W.l.o.g., X_i is fired.



The marking distribution is

$$M(p_{X_i}) = 1; \quad \forall p_{X_{i'}} \in P_S, i' \neq i, M(p_{X_{i'}}) = 0$$

Type 2 $\quad LT_S \subsetneq S, LT_T \subsetneq T$

soundness

- Safeness.

$$\sum M(p_{X_i}) \leq 1, \sum M(p_{Y_j}) \leq 1$$

- Proper completion.

  After firing Y_j, $\sum M(p_{X_i}) = 0, \sum M(p_{Y_j}) = 0$

- Option to complete & No dead parts.

$\forall X_i \in S$ is enabled at beginning

if $LT_S \subsetneq S, \Rightarrow \exists X_i \in S, X_i \notin LT_S$, token remains

if $LT_T \subsetneq T, \Rightarrow \exists Y_j \in T, Y_j \notin LT_T$, token misses

Algorithm – add long-term dependency

- **Long-term dependency Situations**

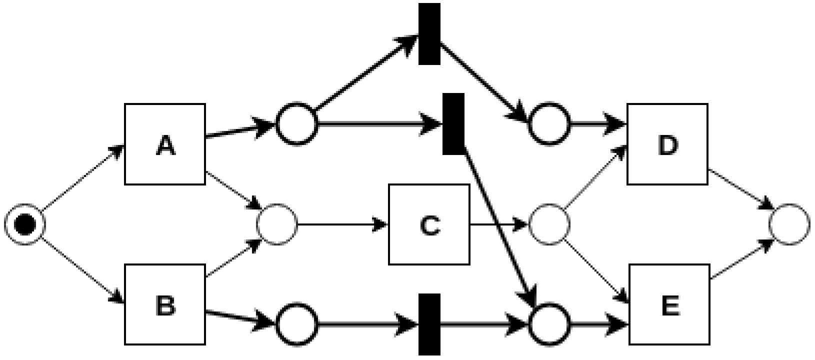
1. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow D, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}, |LT| = |S| \cdot |T|$.
2. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}$ $LT_S = S$ and $LT_T = T, |LT| < |S| \cdot |T|$.
3. $LT = \{A \rightsquigarrow D, B \rightsquigarrow E\}$.
   $LT_S = \{A, B\}, LT_T = \{D, E\}$ $LT_S = S$ and $LT_T = T, |LT| < |S| \cdot |T|$.
4. $LT = \{A \rightsquigarrow D, B \rightsquigarrow D\}$.
   $LT_S = S, LT_T \subsetneq T$.
5. $LT = \{A \rightsquigarrow D, A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T = T$.
6. $LT = \{A \rightsquigarrow E\}$.
   $LT_S \subsetneq S, LT_T \subsetneq T$.
7. $LT = \emptyset$

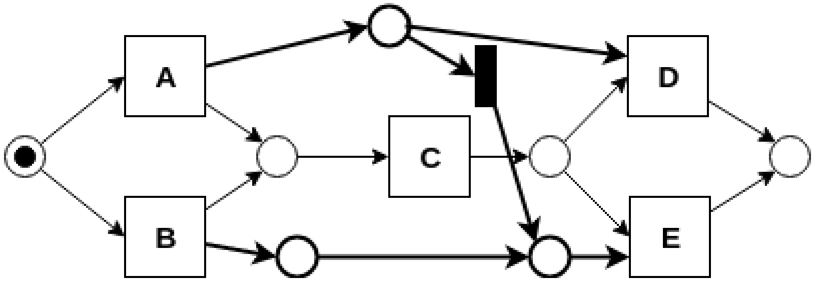
Sound:
- ✓ Safeness.
- ✓ Proper completion.
- ✓ Option to complete.
- ✓ No dead parts.

  - Consider only $LT_S = S, LT_T = T, |LT| < |S| \cdot |T$

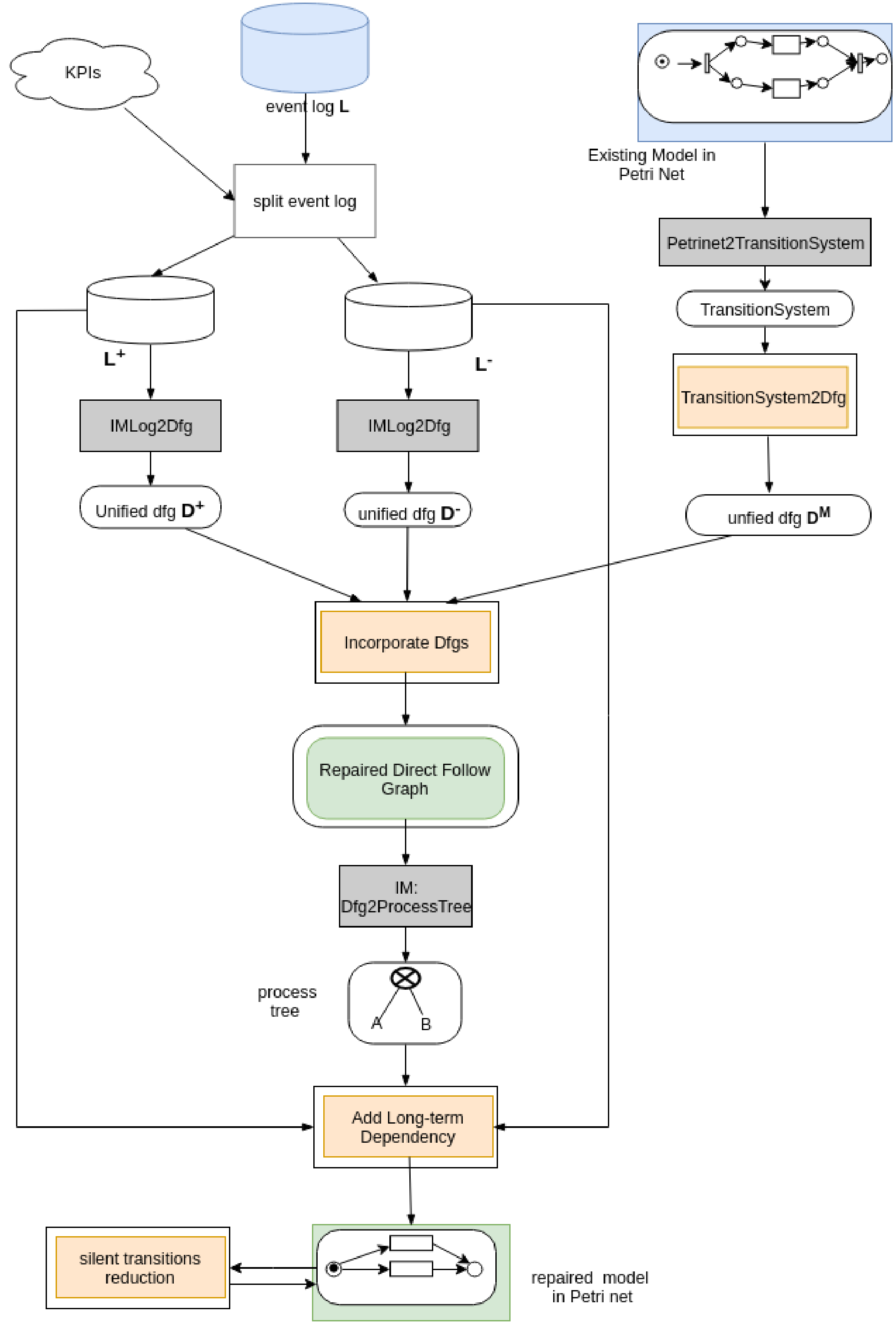
Post process

- **Delete redundant silent transitions**



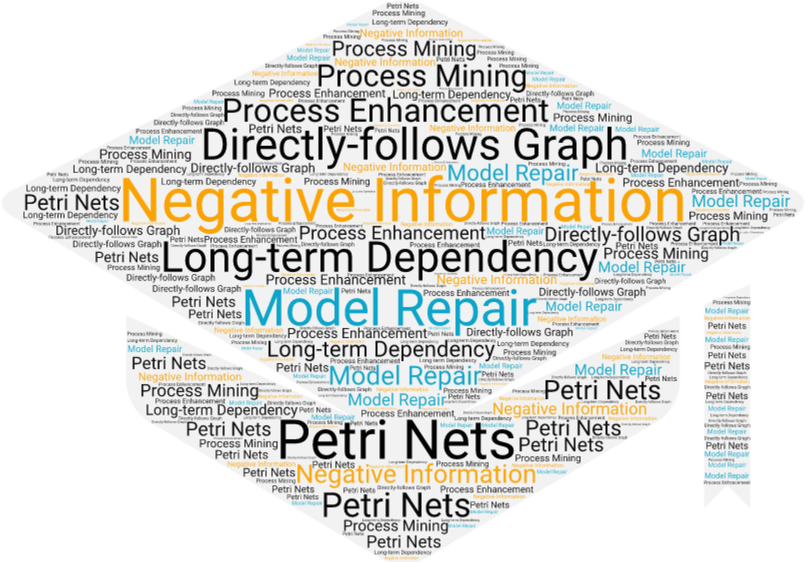


Algorithm – architecture



Outline

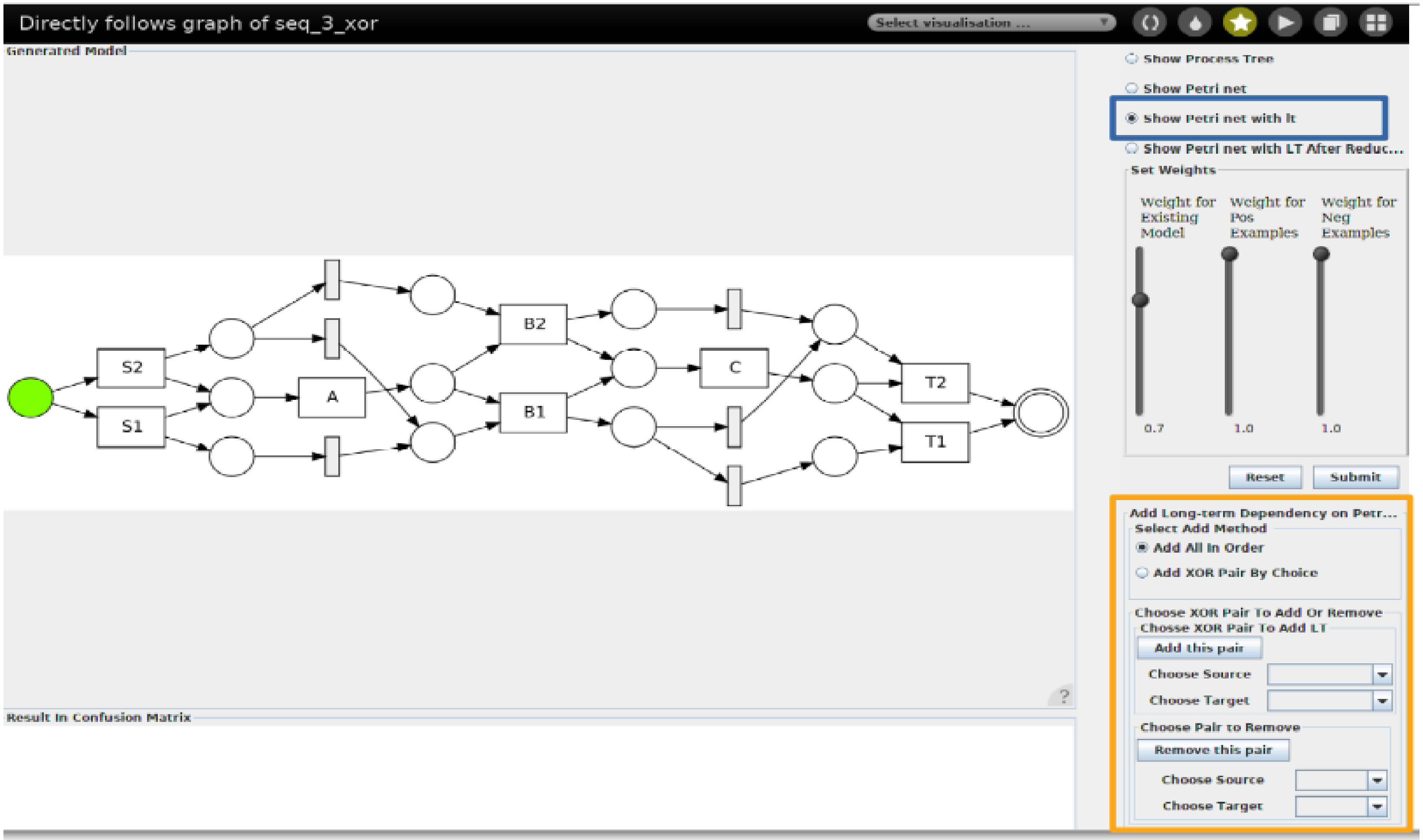
- **Motivation for Research**
- **Problem Definition**
- **Approach**
- **Demo Show**
  - ProM
  - KNIME
- **Evaluation**
- **Conclusion**



Chair of Process and Data Science

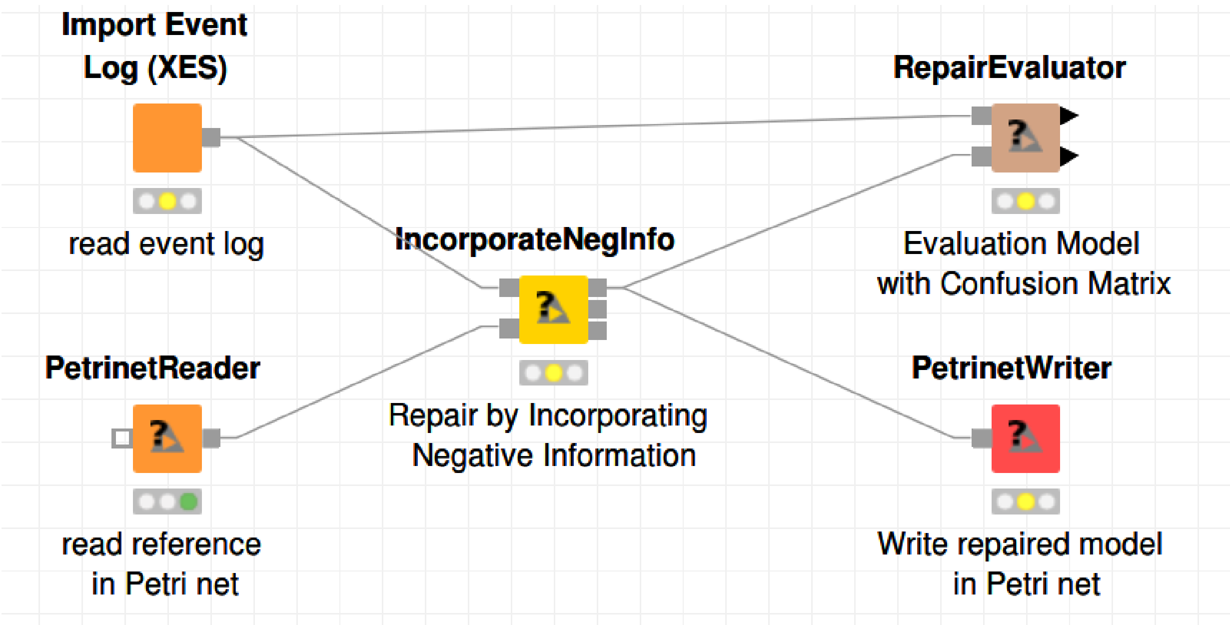
RWTH AACHEN UNIVERSITY

Demo


Directly follows graph of seq_3_xor
Select visualisation ...
Generated Model
S2
S1
A
B2
B1
C
T2
T1
Show Process Tree
Show Petri net
Show Petri net with lt
Show Petri net with LT After Reduc...
Set Weights
Weight for Existing Model
Weight for Pos Examples
Weight for Neg Examples
0.7
1.0
1.0
Reset
Submit
Add Long-term Dependency on Petr...
Select Add Method
Add All In Order
Add XOR Pair By Choice
Choose XOR Pair To Add Or Remove
Choose XOR Pair To Add LT
Add this pair
Choose Source
Choose Target
Choose Pair to Remove
Remove this pair
Choose Source
Choose Target
Result In Confusion Matrix

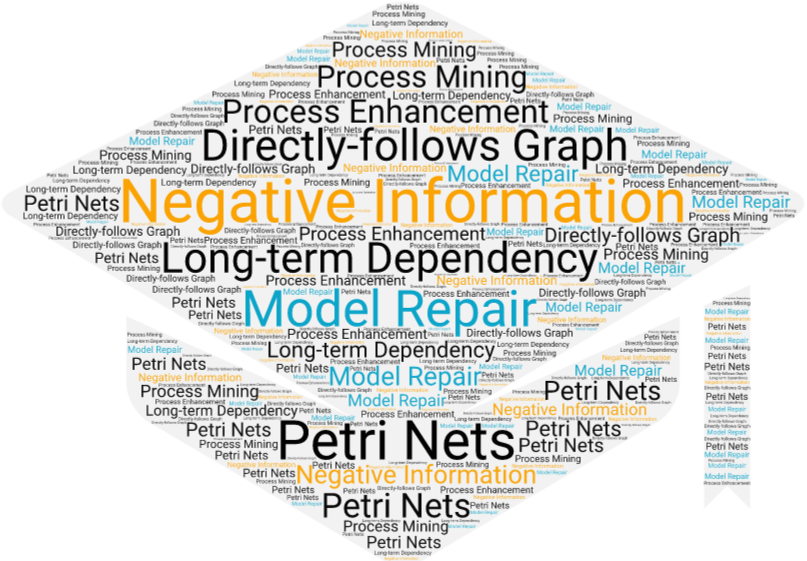
Demo

One more slide to show the workflow in KNIME?? But not here, right??



Outline

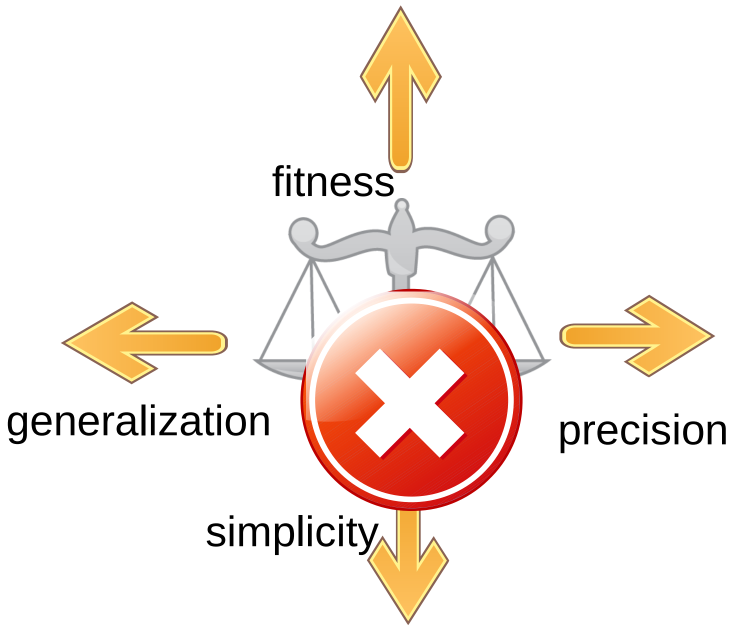
- **Motivation for Research**
- **Problem Definition**
- **Approach**
- **Demo**
- **Evaluation**
  - Synthetic data
  - Real life data
- **Conclusion**



Chair of Process and Data Science

RWTH AACHEN UNIVERSITY

Evaluation



- **Confusion matrix**
  - Recall

$$Recall = \frac{TP}{TP + FN}$$

  - Precision

$$Precision = \frac{TP}{TP + FP}$$

  - Accuracy

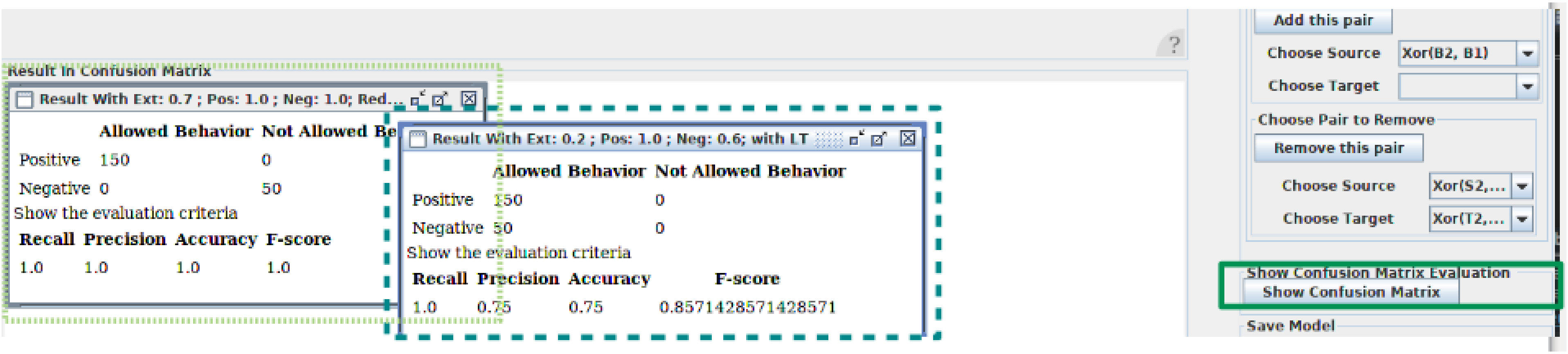
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  - F1

$$F_1 = \frac{2 * Recall * Precision}{Precision + Recall}$$

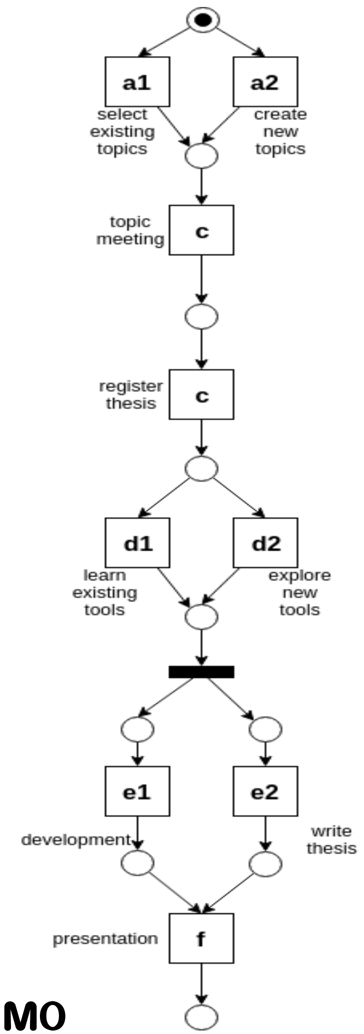
	Allowed behavior	Not allowed behavior
positive	TP	FN
negative	FP	TN

Demo --result



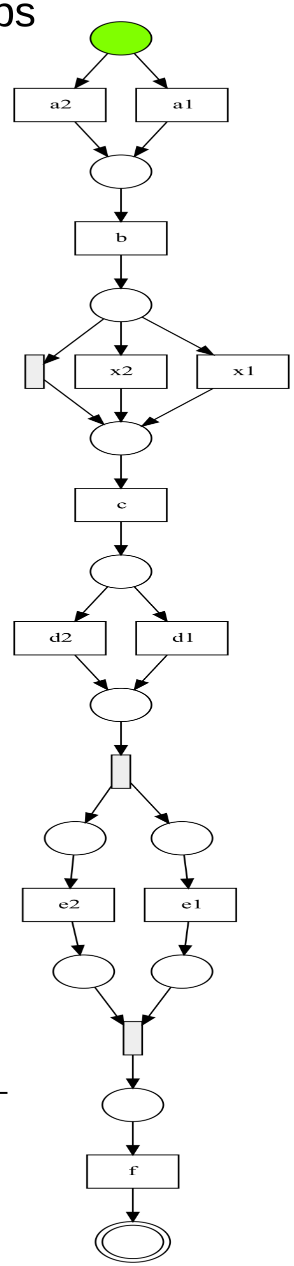
Result In Confusion Matrix
Result With Ext: 0.7 ; Pos: 1.0 ; Neg: 1.0; Red...
Allowed Behavior Not Allowed Be
Positive 150 0
Negative 0 50
Show the evaluation criteria
Recall Precision Accuracy F-score
1.0 1.0 1.0 1.0
Result With Ext: 0.2 ; Pos: 1.0 ; Neg: 0.6; with LT
Allowed Behavior Not Allowed Behavior
Positive 150 0
Negative 50 0
Show the evaluation criteria
Recall Precision Accuracy F-score
1.0 0.75 0.75 0.8571428571428571
Add this pair
Choose Source Xor(B2, B1)
Choose Target
Choose Pair to Remove
Remove this pair
Choose Source Xor(S2,...
Choose Target Xor(T2,...
Show Confusion Matrix Evaluation
Show Confusion Matrix
Save Model

Demo --result



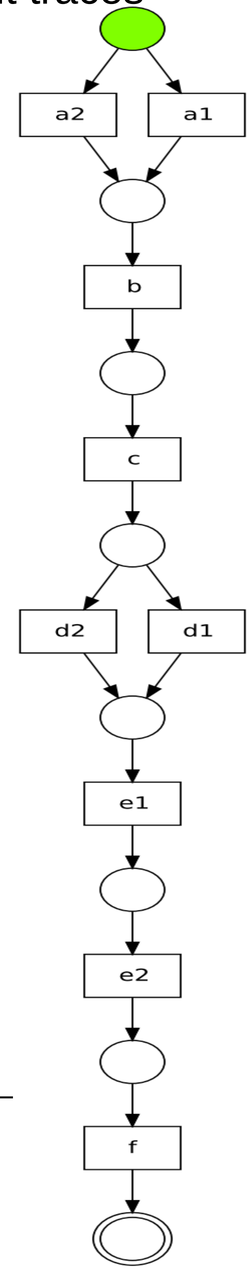
M0

Overcome s1:
add subprocesses
as loops



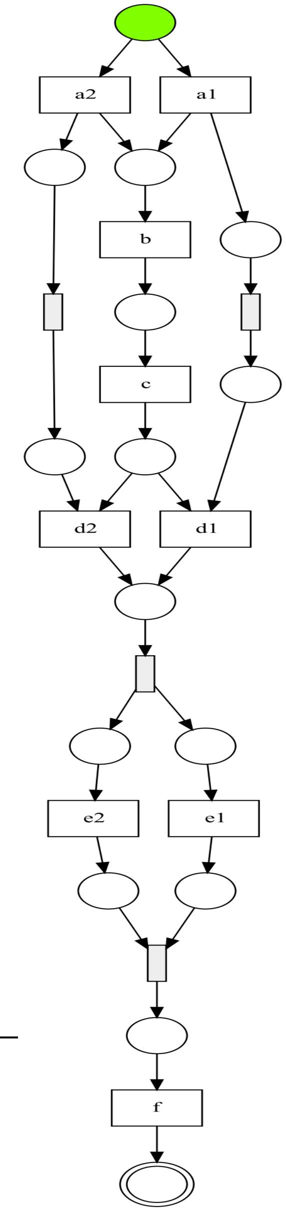
M1.3

Overcome s2:
Unable to adapt model
with fit traces



M2.3

Overcome s3:
Unable to detect
long-term dependency



M3.3

Demo --result

Situation	Method	Generated Model	Confusion matrix measurements							
			TP	FP	TN	FN	recall	precision	accuracy	F1
S1	IM-Infrequent Noise threshold: 20%	M1.1	50	50	0	0	1	0.5	0.5	0.667
	Fahland's Repair Model	M1.2	50	50	0	0	1	0.5	0.5	0.667
	Dfg-repair	M1.3	50	50	0	0	**1**	**0.5**	**0.5**	**0.667**
S2	IM/Fahland's	M0	60	45	0	0	**1**	0.571	0.571	0.727
	Dfg-repair	M2.3	50	5	40	10	0.833	**0.909**	**0.857**	**0.870**
S3	IM/Fahland repair	M0	100	100	0	0	1	0.5	0.5	0.667
	Dfg-repair	M3.3	100	0	100	0	**1**	**1**	**1**	**1**

Conclusion:

- ✓ Conquer shortcomings of current techniques in listed situations,
- ✓ Better precision, accuracy, F1 score

Experiments -- Real life data

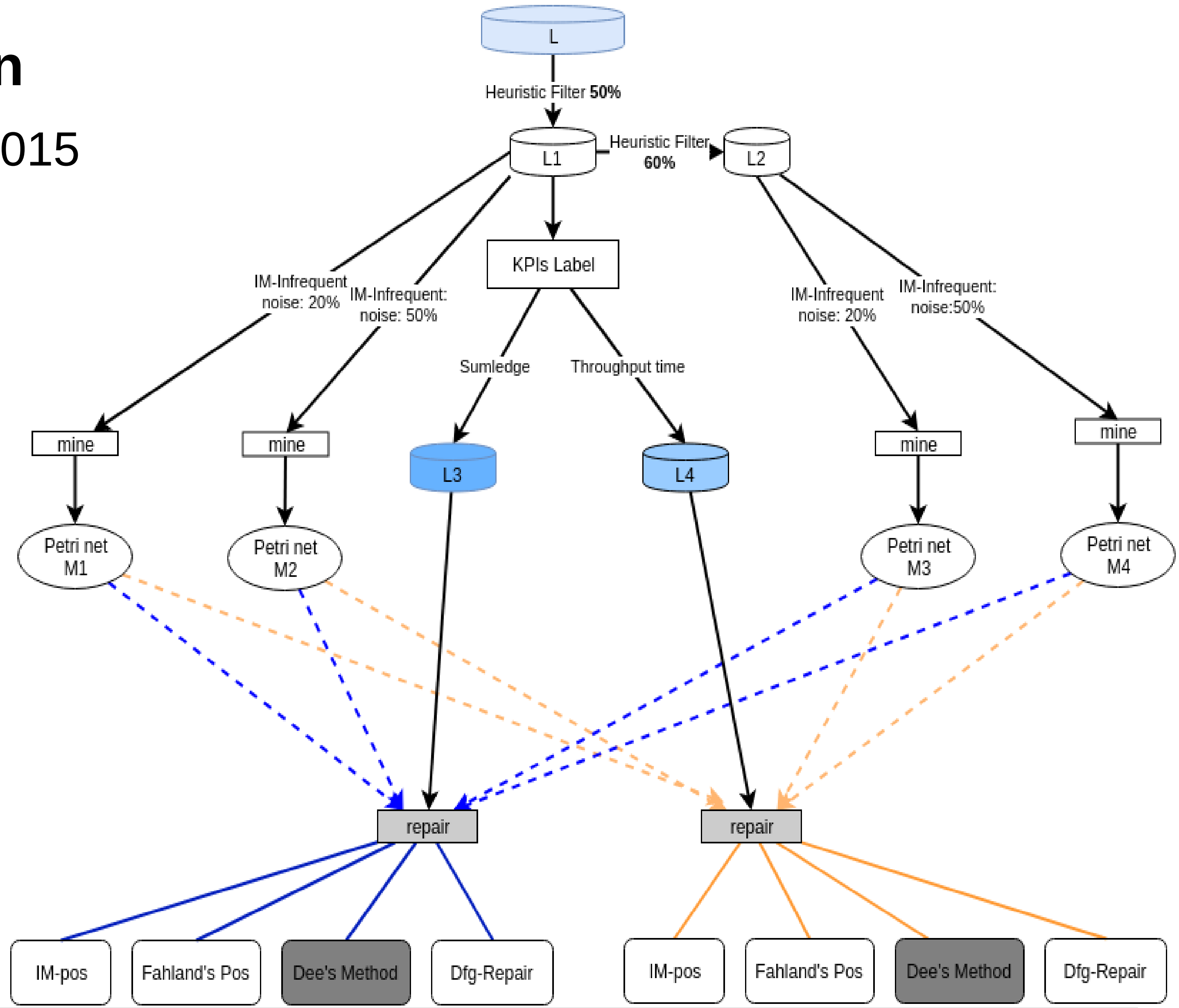
- **Data description**
  - BPI Challenge 2015

    BPIC15_1.xes.xml:

    1199 cases,

    52217 events

    398 event classes



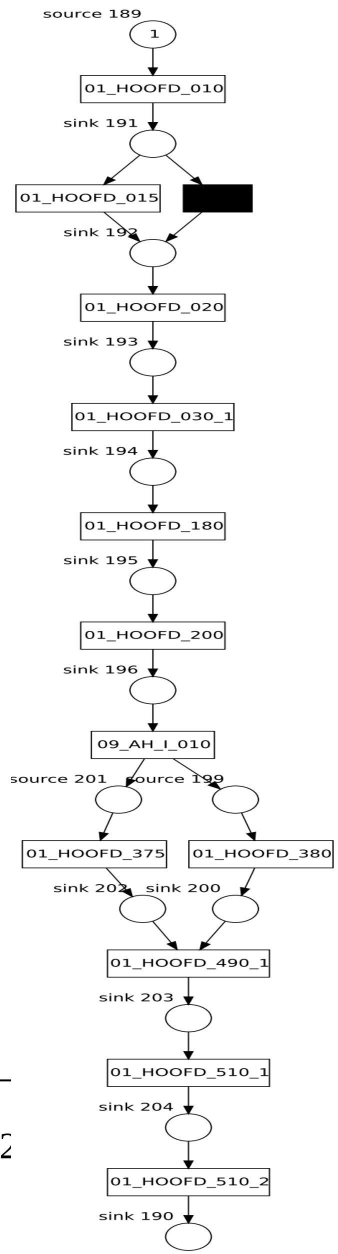
Experiments

- **Event logs**

ID	Description	Traces Num	Events Num	Event Class
D1	Heuristic filter 40%	495	9565	20
D2	Heuristic filter 60% on D1	378	4566	12
D3.1	Classify on Sumledge; Below 70% as positive	349	6744	20
D3.2	Classify on Sumledge; over 70% as negative	146	2811	20
D3.3	Union of D3.1 and D3.2	495	9565	20
D4.1	Classify on throughput time; Below 70% as positive	346	6719	20
D4.2	Classify on Sumledge; over 70% as negative	146	2846	20
D4.3	Union of D4.1 and D4.2	495	9565	20

Experiments

- **Petri net Models**

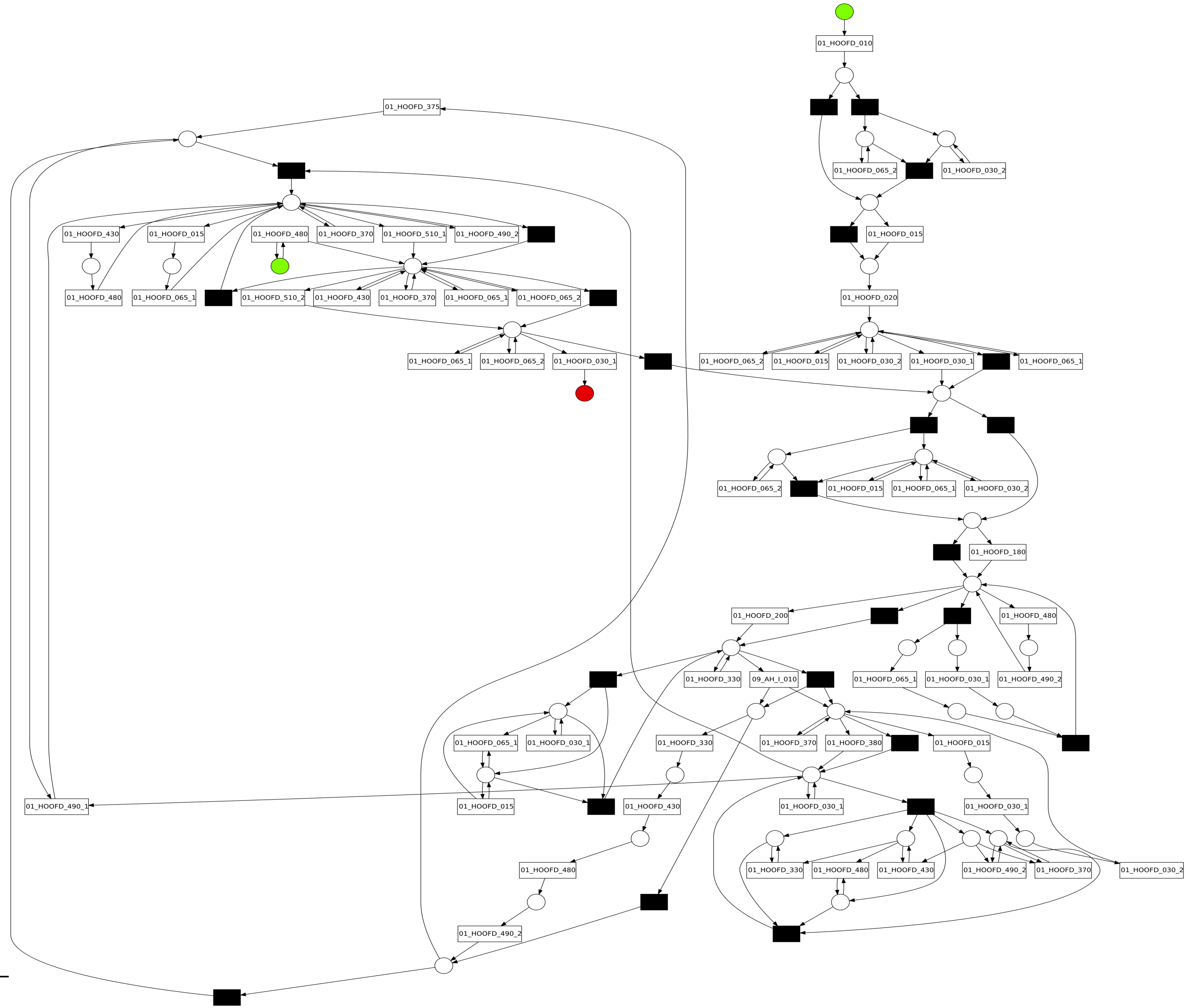


M3

Model ID	Data ID	Confusion matrix							
		TP	FP	TN	FN	recall	Precision	Accuracy	F1
M1	D3.3	112	40	106	237	0.321	0.737	0.440	0.447
	D4.3	131	21	128	215	0.379	0.862	0.523	0.526
M2	D3.3	106	39	107	243	0.304	0.731	0.430	0.429
	D4.3	125	20	129	221	0.361	0.862	0.513	0.509
M3	D3.3	0	0	146	349	0	NaN	0.295	0
	D4.3	0	0	149	346	0	NaN	0.301	0
M4	D3.3	0	0	146	349	0	NaN	0.295	0
	D4.3	0	0	149	346	0	NaN	0.301	0

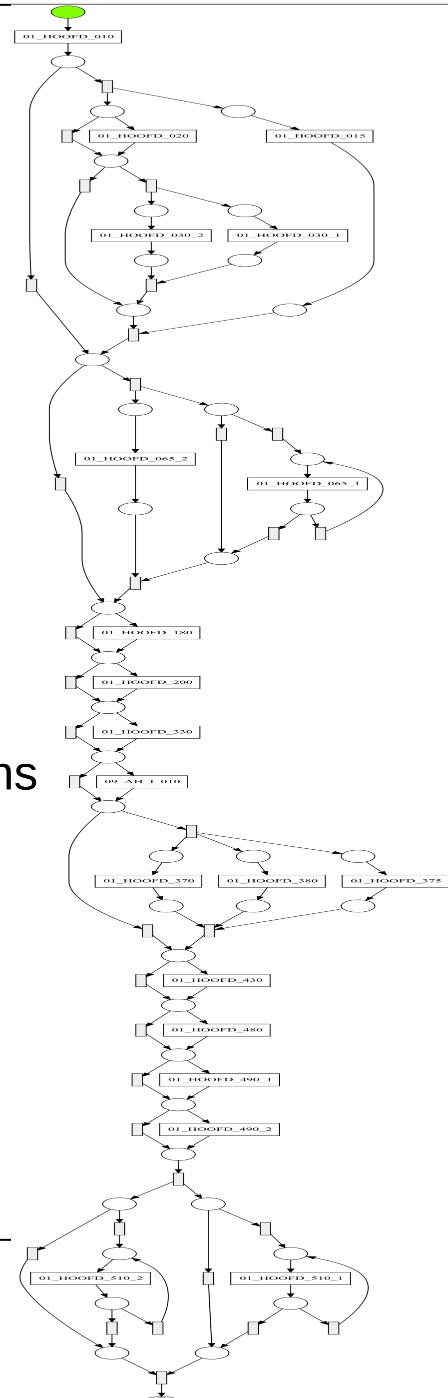
Experiments- result

Fanhland's method to repair M3 with default setting

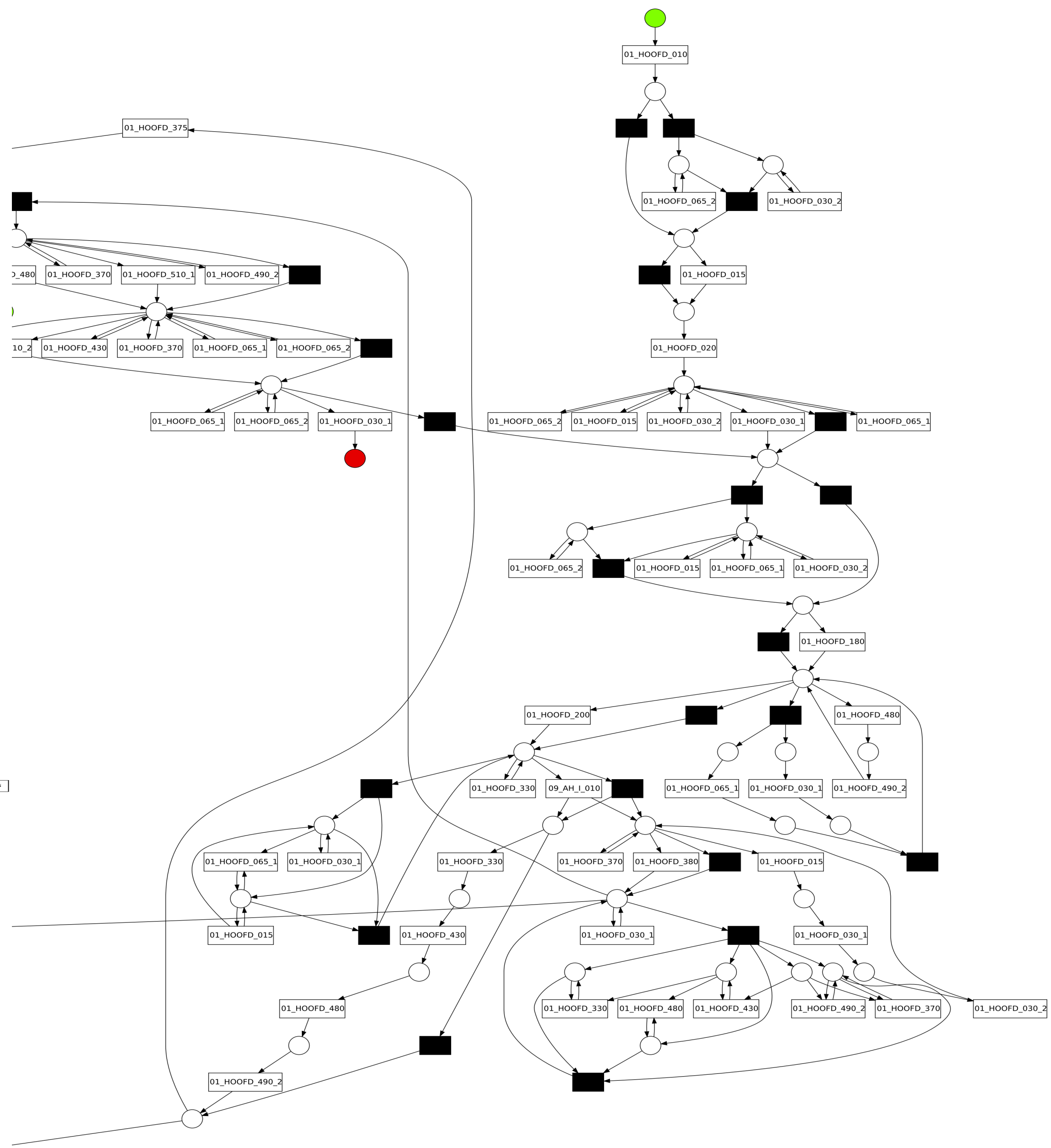


Experiments results

Dfg-repair with default setting



Much simpler with **less** silent transitions and **no** duplicate transitions

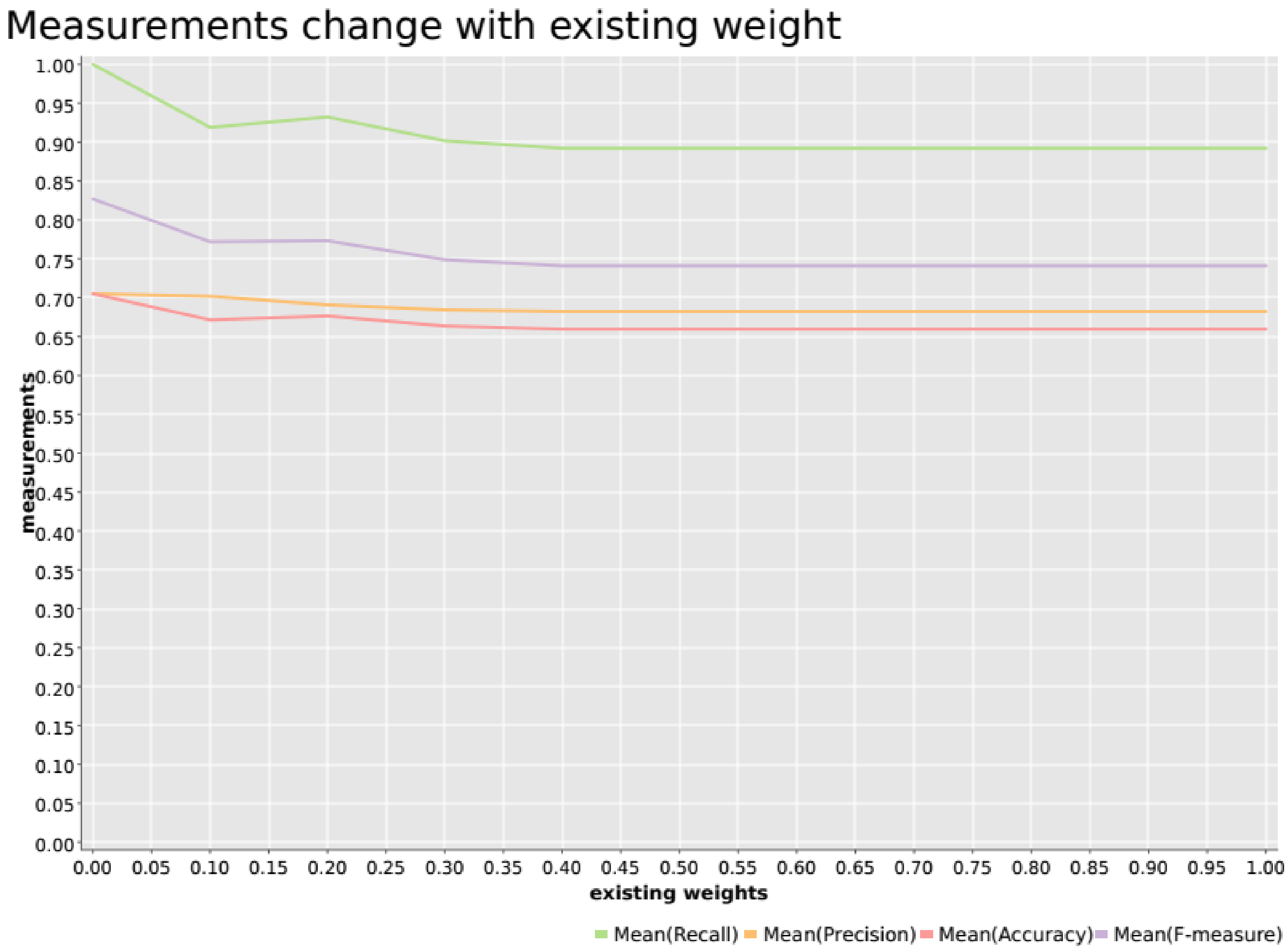


Experiment result

- **Put a slider**

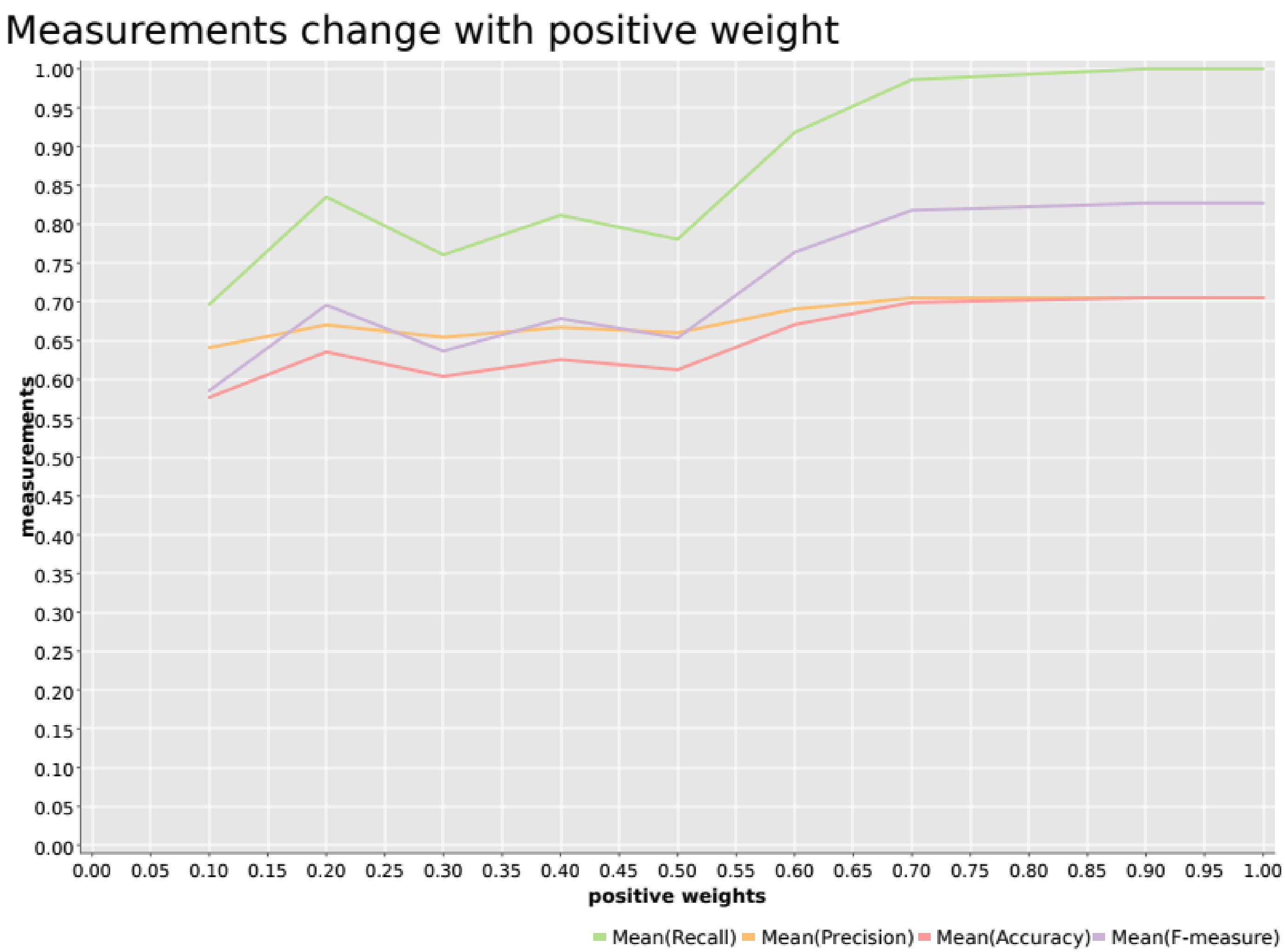
Experiment result

- **Weight for the reference model**



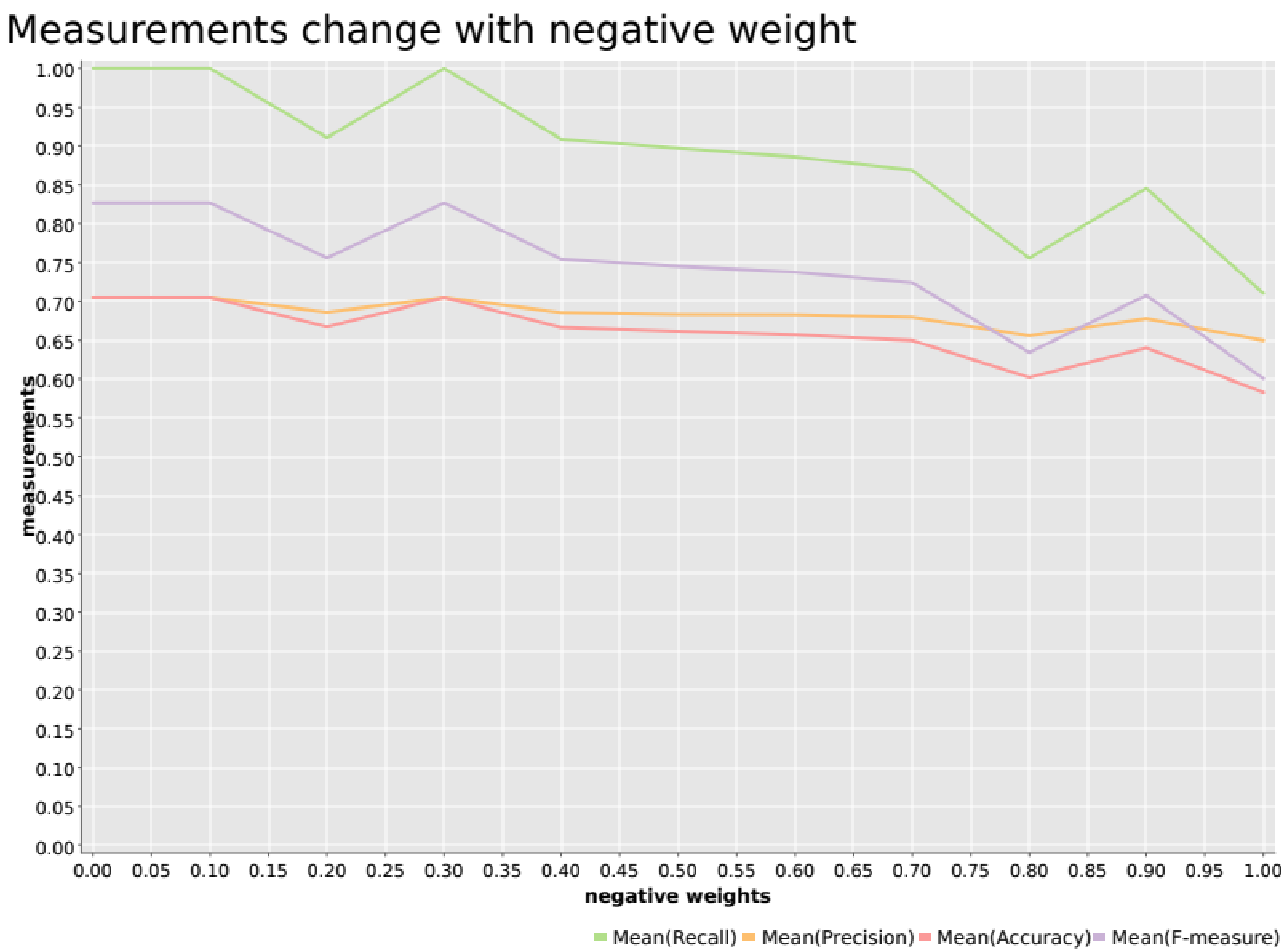
Experiment result

- **Weight for positive instance**



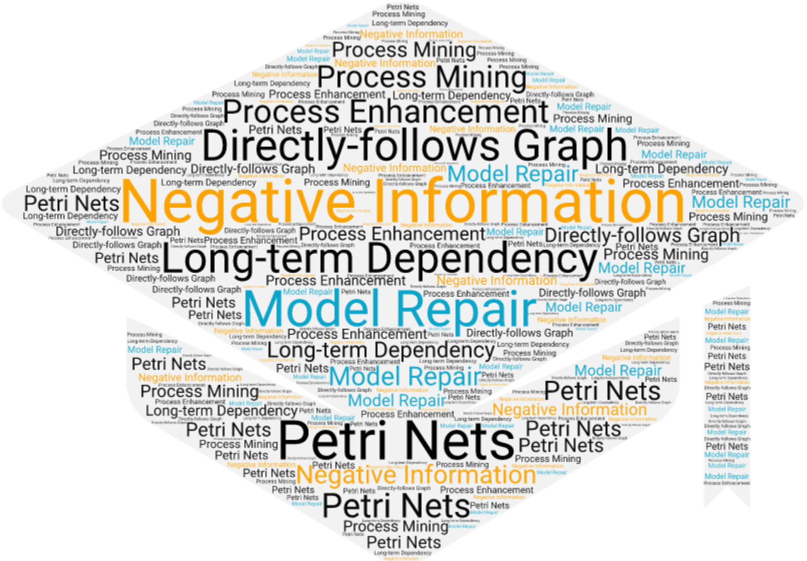
Experiment result

- **Weight for negative instance**



Outline

- **Motivation for Research**
- **Problem Definition**
- **Approach**
- **Demo**
- **Evaluation**
- **Conclusion**



Conclusion

- Conquer the shortcomings
- Repair model with better precision, accuracy, F1

- Feasible to use in practice

- In observation, repaired model simpler, run faster

Further Work

- Improve the balance rules
- Improve the rules for long-term dependency
- Drop process tree as intermediate model
- Extend to another choice relation

Questions & Answers