부산대학교 PUSAN NATIONAL UNIVERSITY BAE Lab Bigdata Analytics & Engineering

02.

Process Mining Overview

소 속 : 부산대학교 산업공학과
이 름 : 배 혜 림 교수
이메일 : hrbae@pusan.ac.kr

Contents

2.1 Process Model

2.2 Process Discovery

2.3 Conformance Techniques

2.4 Extension

2.5 Refined Process Mining Framework

2.1 Process Model

What is Process Model

Petri-Net

프로세스를 모델링하는 그래픽 표기법으로, Place, Transition, Arc 구성 요소를 활용하여 동시성과 동기화 등의 복잡한 개념을 설명

Transition System

State와 State 간의 Transition을 그래프로 표현하는 모델링 방법으로, 시스템의 모델링, 검증, 분석 등에 활용

Process Tree

트리를 사용하여 프로세스를 모델링하는 방식으로, 계층적 구조와 직관적인 표현으로 프로세스의 병렬, 순차, 대안 등의 실행 관계를 쉽게 이해 가능

2.1 Process Model

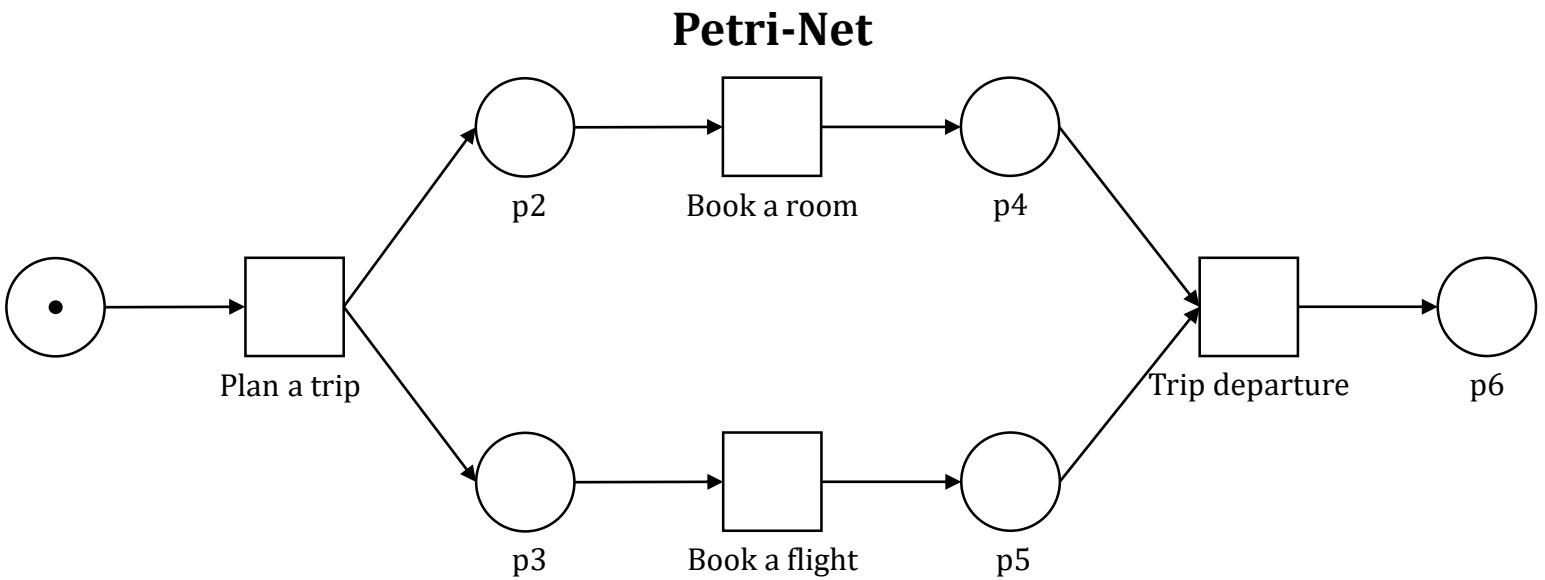
1. Petri-Net

Token (●) : 동작을 나타내는 추상적인 개념 : Arc를 따라 움직이는 개체

Place (◯) : 시스템에서 발생하는 이벤트나 데이터의 위치로 Token이 저장되는 곳

Transition (□) : 이벤트 로그에서 Activity로 처리해야 할 작업

Arc (⟶) : Token의 이동경로로 Place와 Transition을 연결



단순하고 직관적이며 가장 많이 사용되는 Process Model

2.1 Process Model

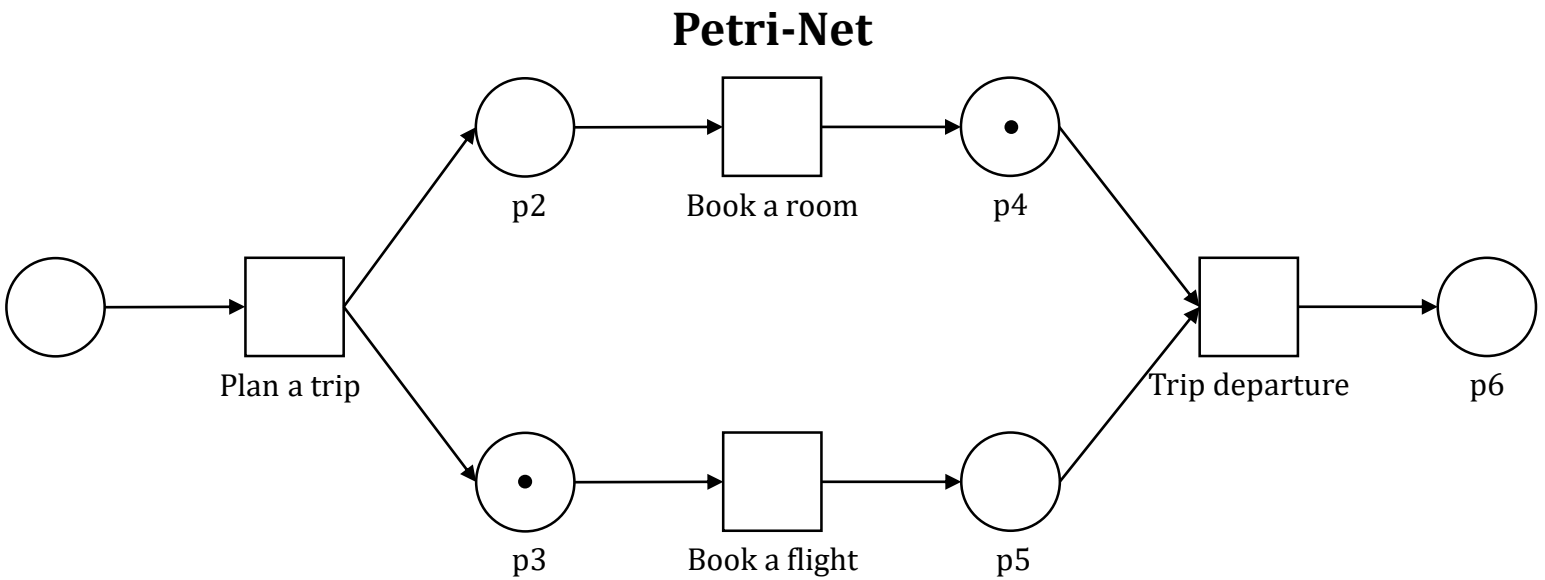
1. Petri-Net

Marking : 페트리 넷 안에 token이 있는 상태 그 자체를 의미

Initial Marking : 처음 상태의 Marking

Reachable Marking : Initial Marking으로부터 도달할 수 있는 Marking을 의미

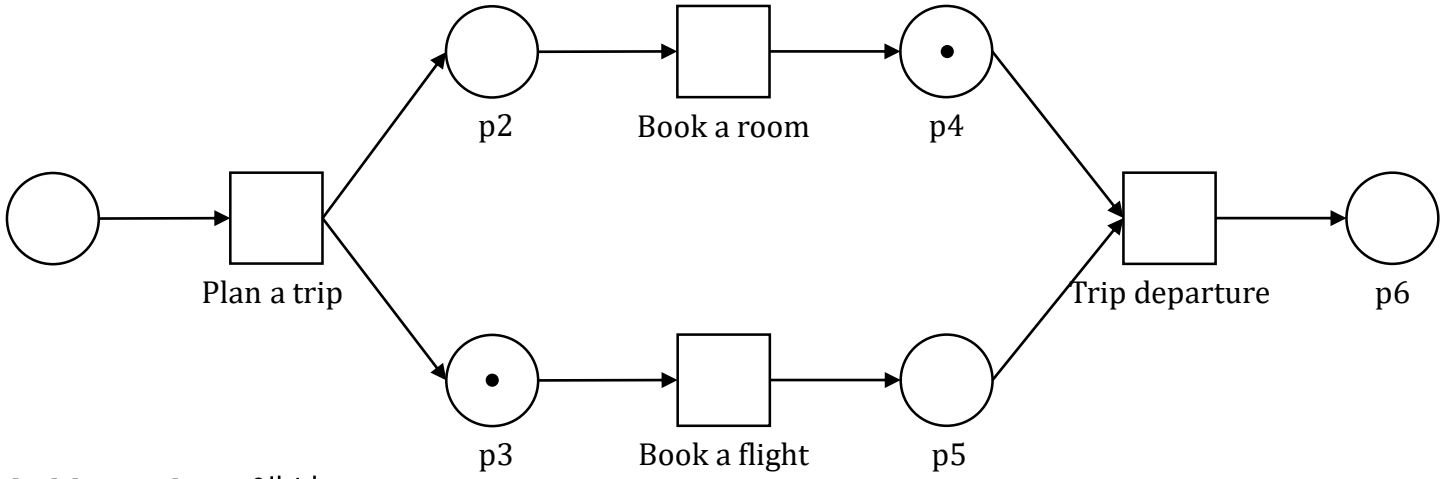
Unreachable Marking : Initial Marking으로부터 도달할 수 없는 Marking을 의미



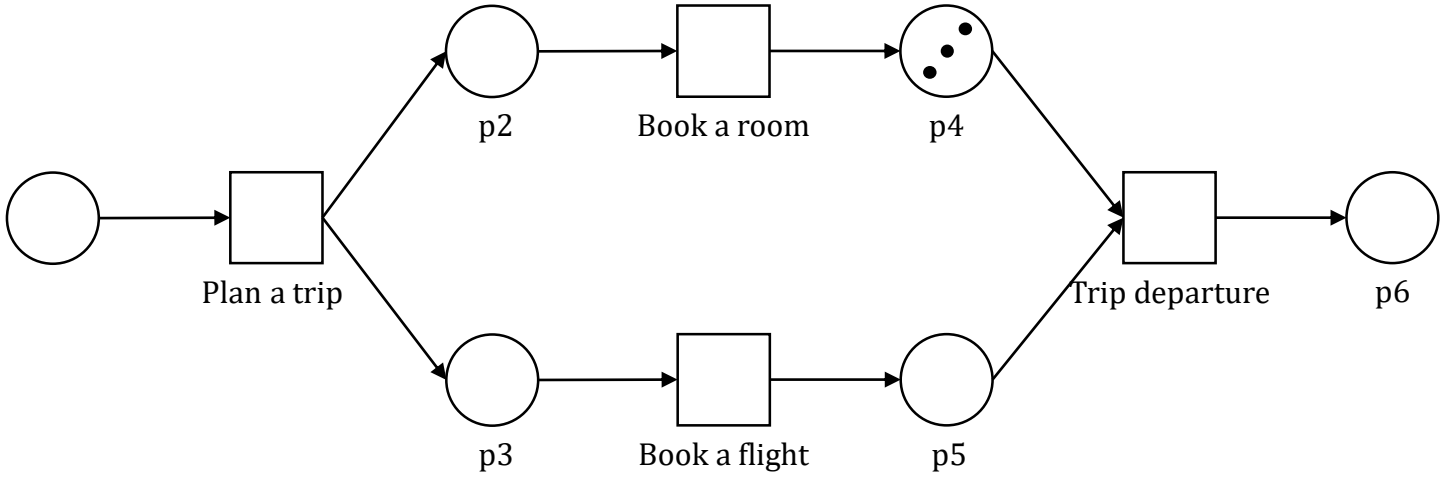
2.1 Process Model

1. Petri-Net

Reachable Marking 예시



Unreachable Marking 예시

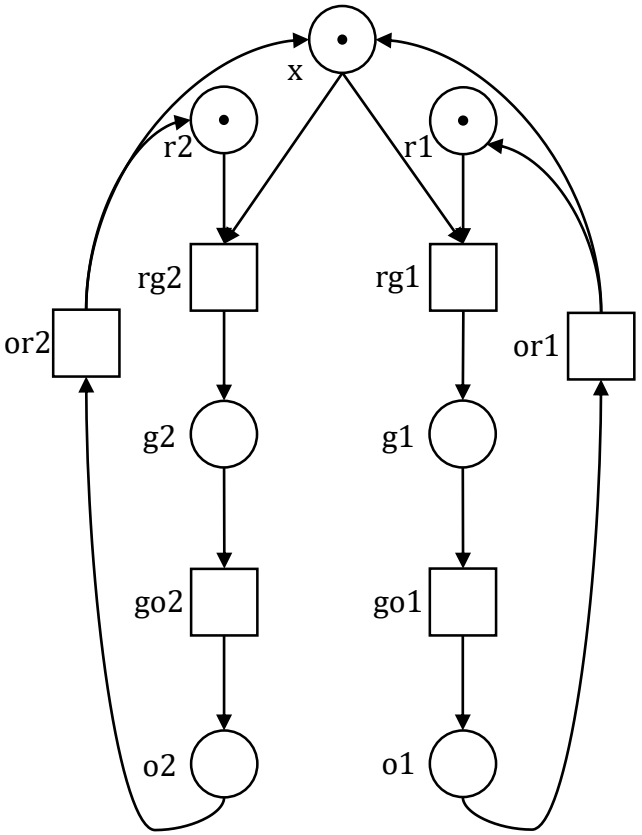


2.1 Process Model

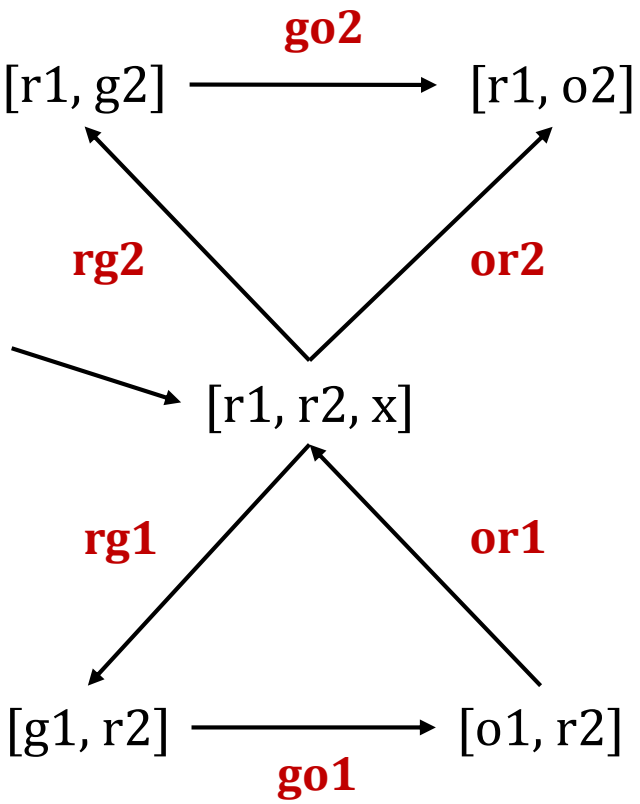
2. Transition System

- Petri-Net으로 부터 도출한 Transition System
- Transition System을 도출하는 방법은?

Petri-Net



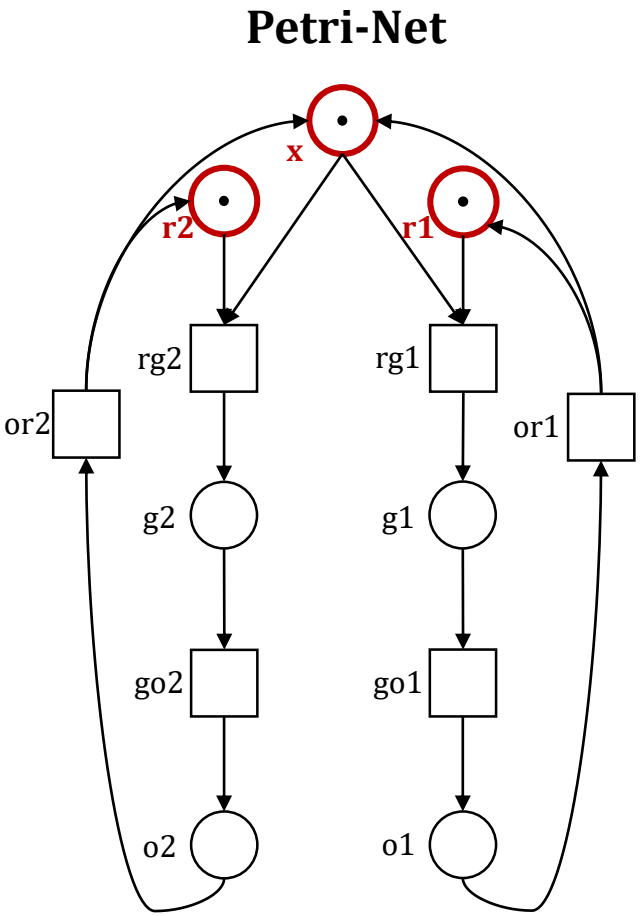
Transition System



2.1 Process Model

2. Transition System

Step 1) 첫 시작점을 Marking. [r1, r2, x]



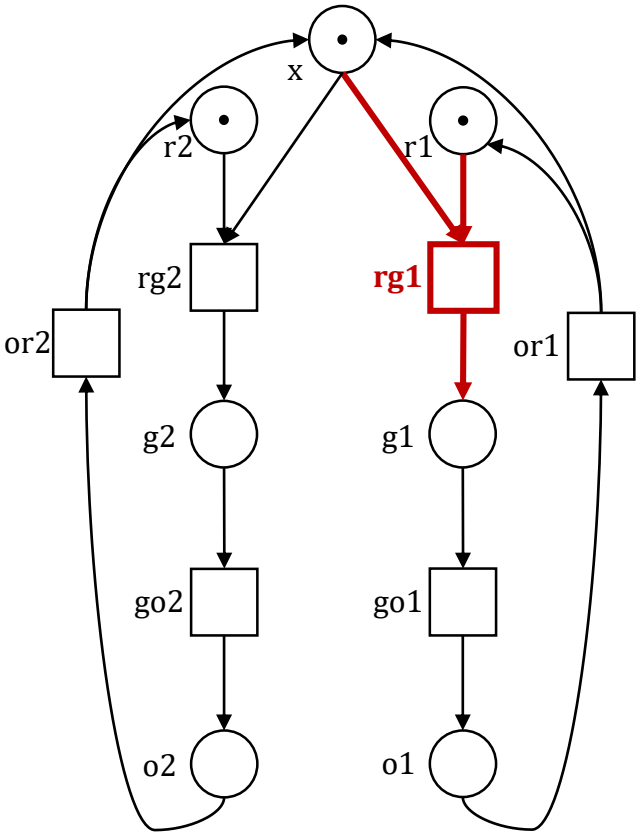
[r1, r2, x]

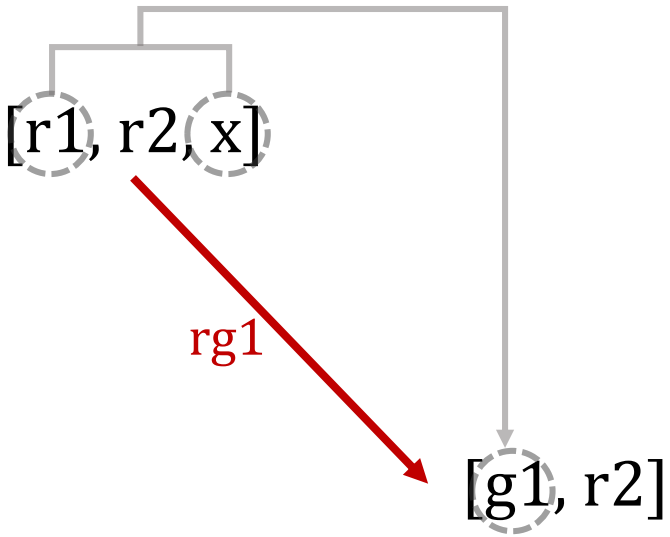
2.1 Process Model

2. Transition System

Step 2) 진행될 수 있는 Transition을 선택, 진행 가능한 Token들을 새로운 place로 Marking.
즉, rg2, rg1 Transition으로 진행 가능하고 rg1를 선택 시 r1과 x는 다음 place인 g1으로 Marking.

Petri-Net



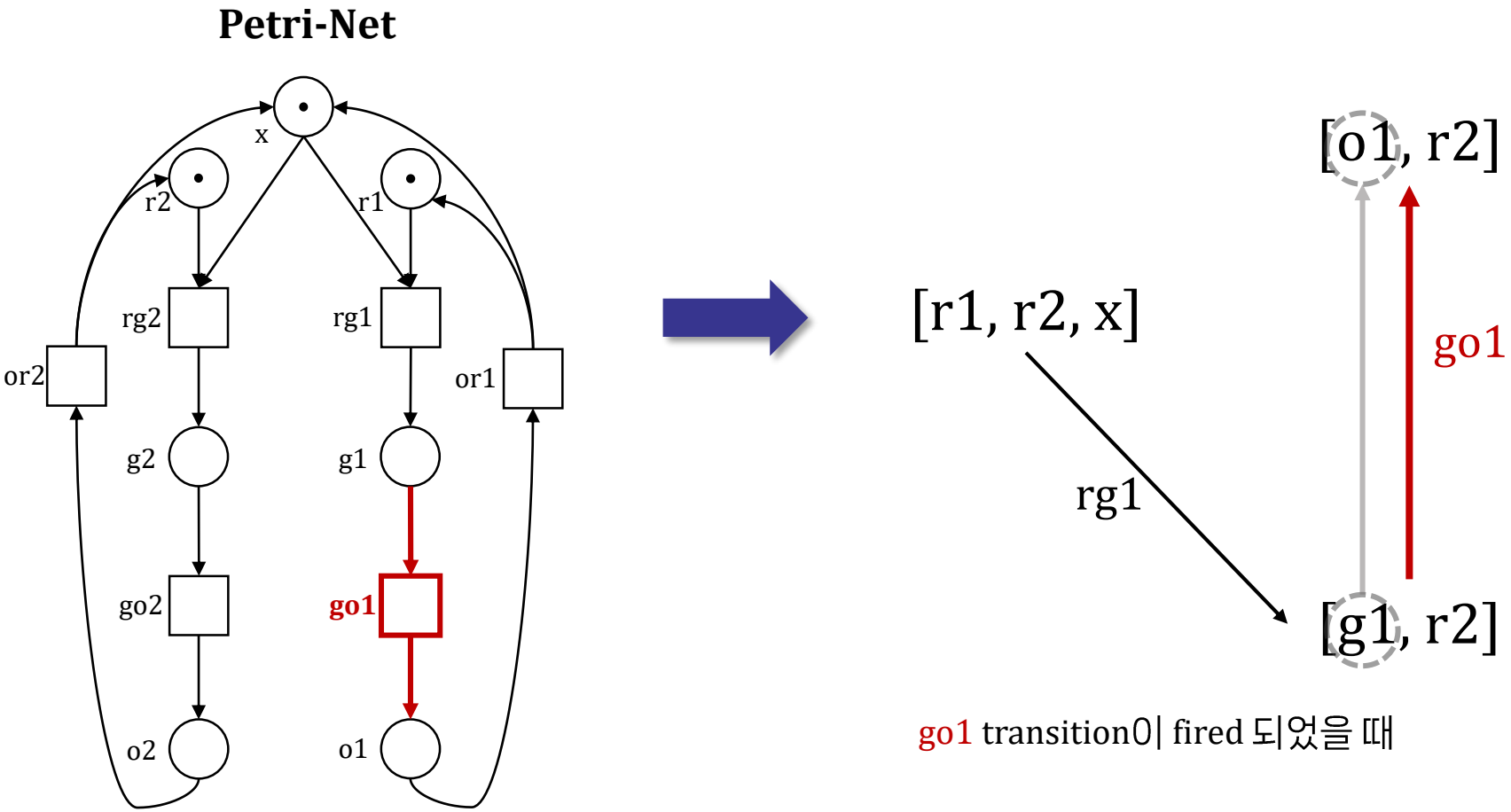


rg1 transition이 fired 되었을 때

2.1 Process Model

2. Transition System

Step 3) 모든 Transition에 대해 Step 2를 반복.

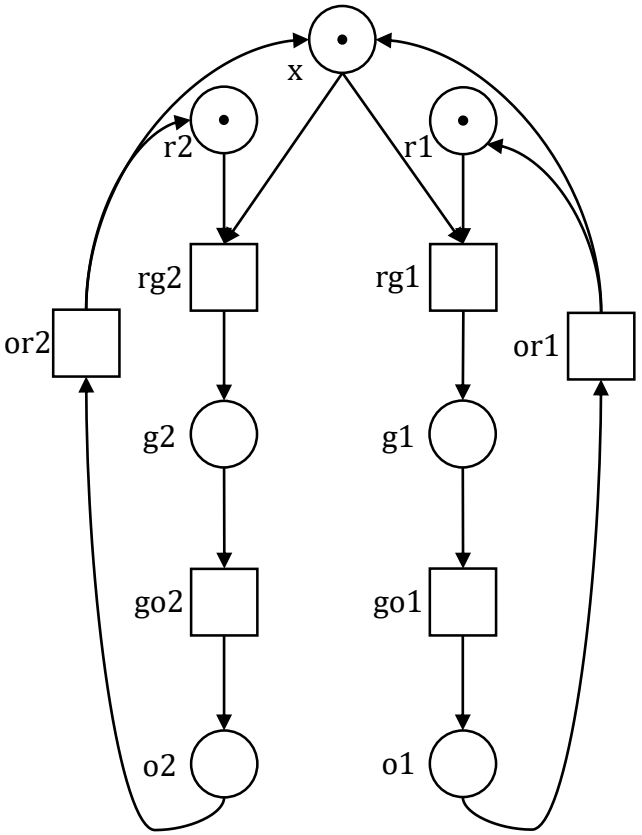


2.1 Process Model

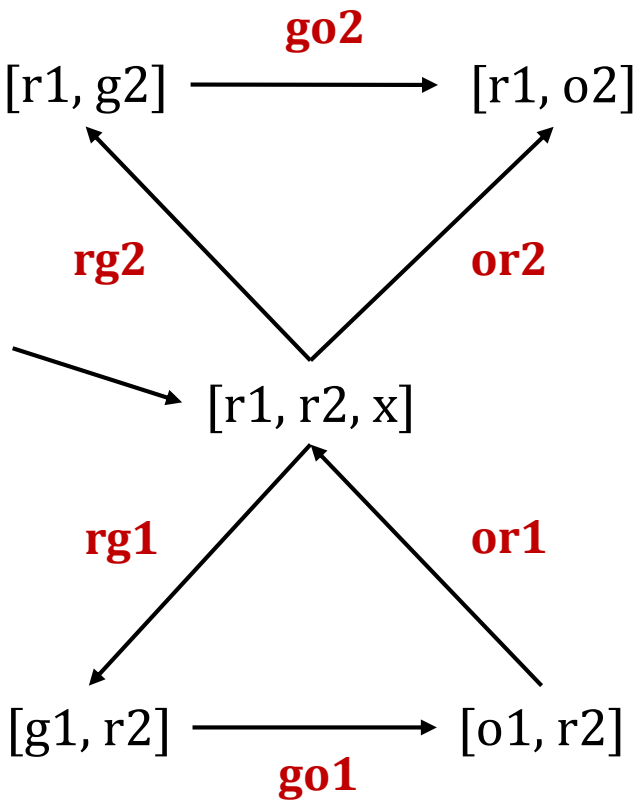
2. Transition System

- 완성된 Transition System

Petri-Net



Transition System



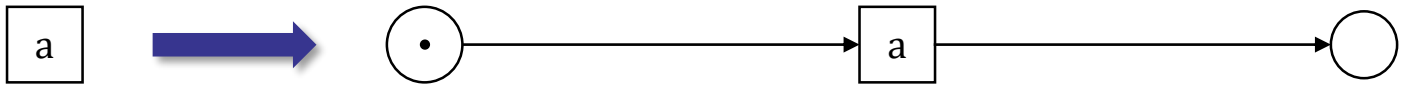
2.1 Process Model

3. Process Tree Components

- Activity

  – Normal activity :

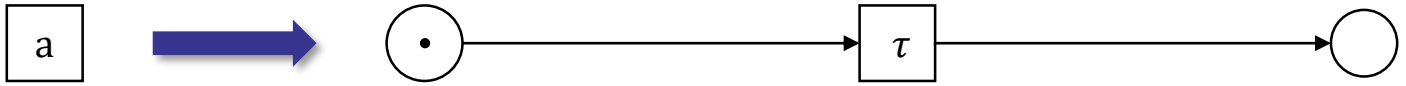
    Activity들이 순차적으로 진행



  – Silent activity :

    아무 Activity도 일어나지 않음을 표시한 Activity

    차후 그래프를 그릴 때 사용하기 위하여 Process Tree에서 사용하는 요소



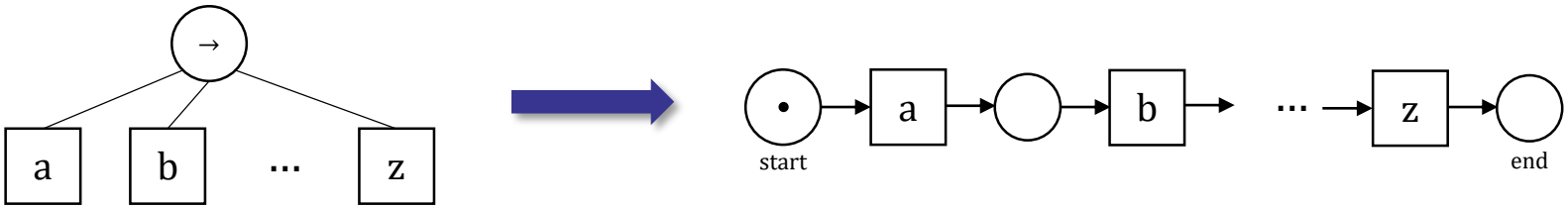
2.1 Process Model

3. Process Tree Components

- Operator

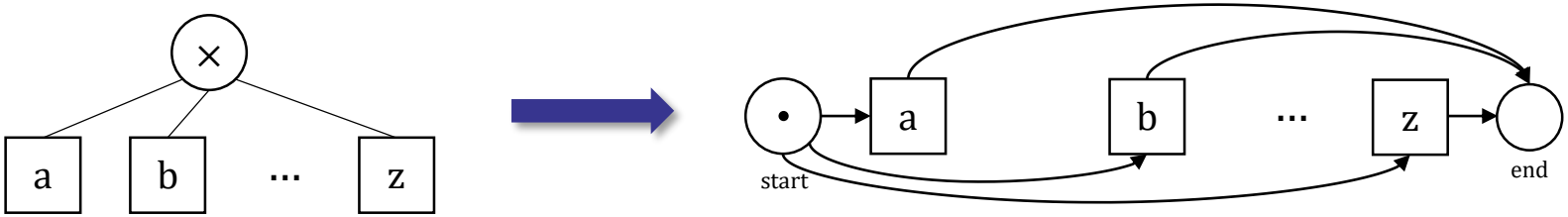
  – Sequential composition :

  Activity들이 순차적으로 실행



  – Exclusive choice:

  Activity들 중 하나만 실행



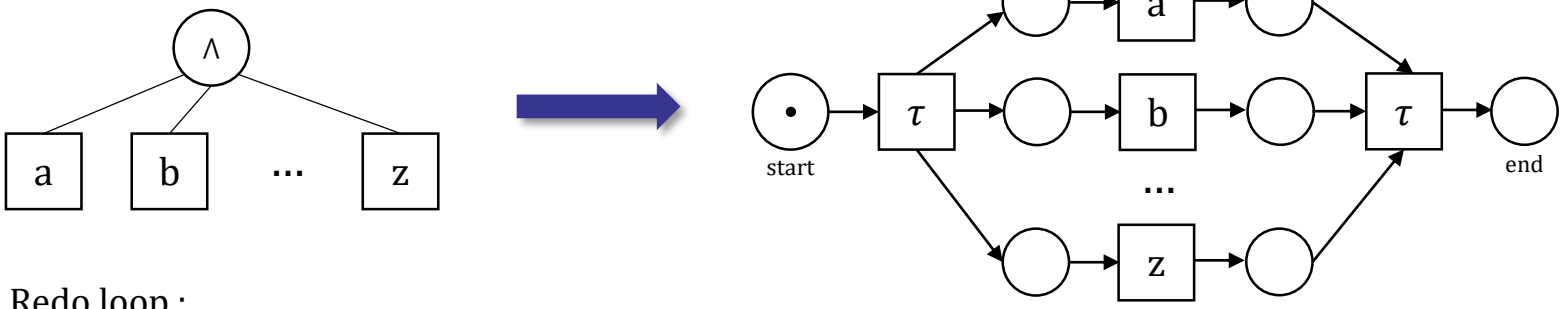
2.1 Process Model

3. Process Tree Components

- Operator

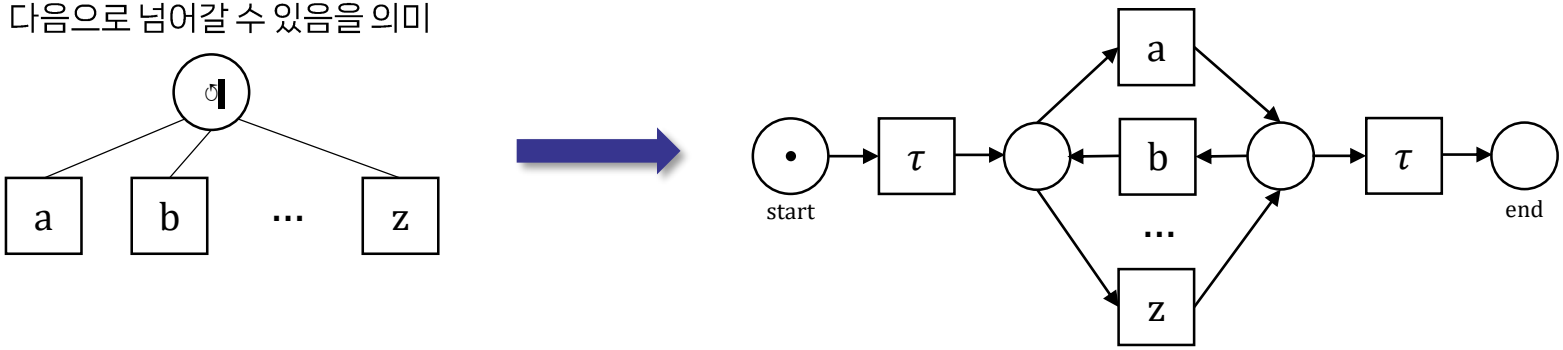
  – Parallel composition :

  Activity들이 순서 없이 병렬적으로 실행



  – Redo loop :

  Activity가 하나 실행된 후 다시 돌아가 하위 Activity들을 실행하거나
  다음으로 넘어갈 수 있음을 의미

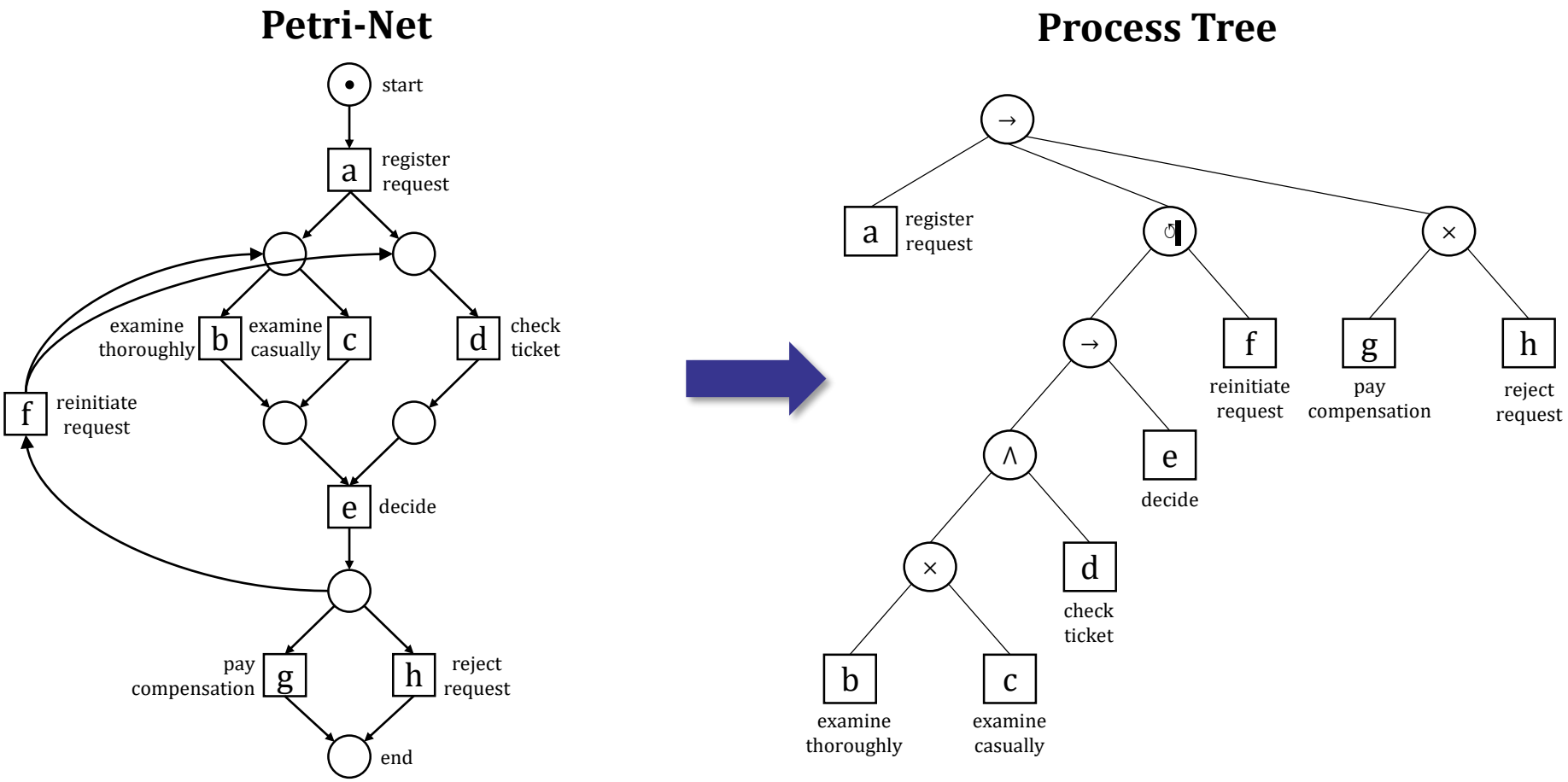


Place간의 연결되는 것을 막기 위하여 Silent activity를 사용

2.1 Process Model

3. Process Tree

- Petri-Net으로부터 도출한 Process Tree.
- Process Tree를 도출하는 방법은?

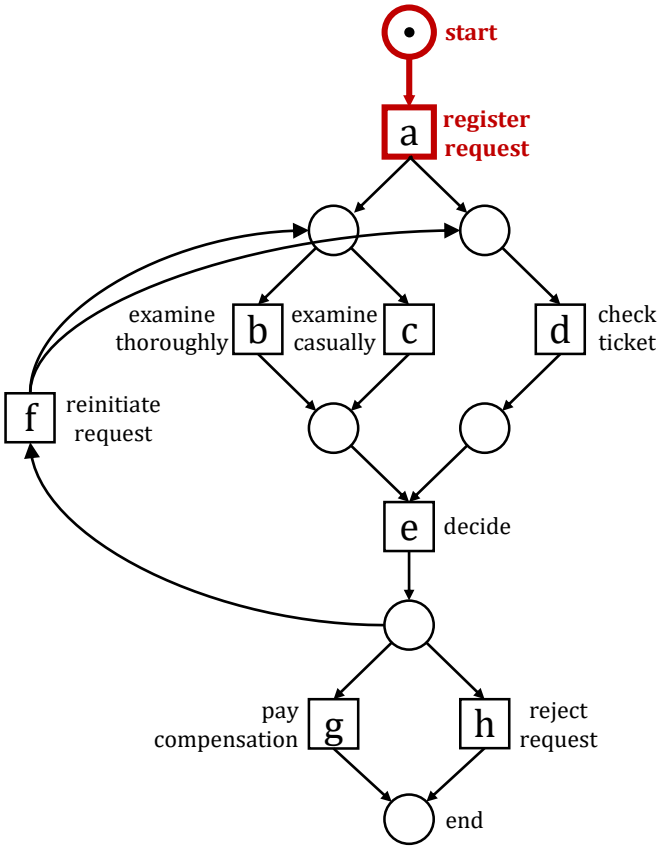


2.1 Process Model

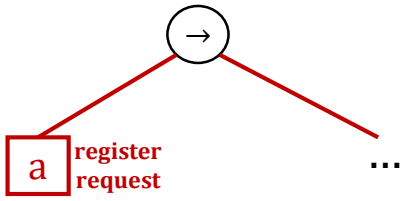
3. Process Tree

시작점부터 Operator와 Activity의 관계를 보며 단계별로 그림

Petri-Net



start에서 Activity a로만 진행하므로 →

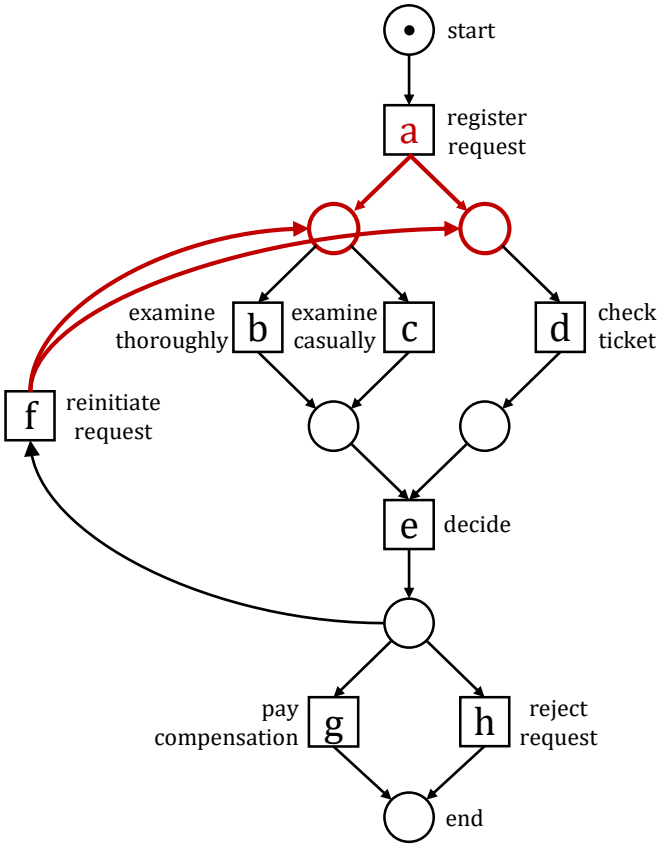


2.1 Process Model

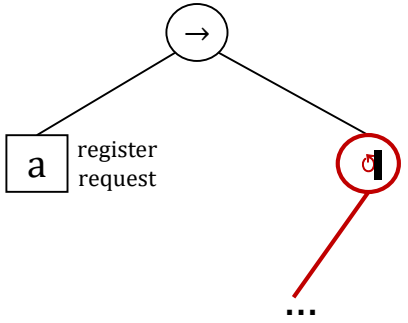
3. Process Tree

시작점부터 Operator와 Activity의 관계를 보며 단계별로 그림

Petri-Net



a 다음 Operator가 redo loop 형태를 띄므로 ↺

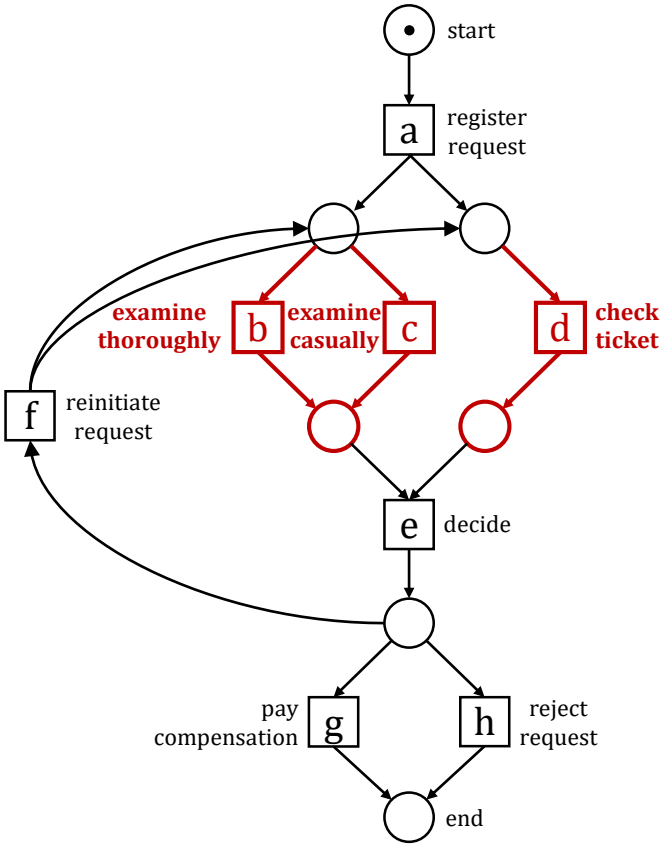


2.1 Process Model

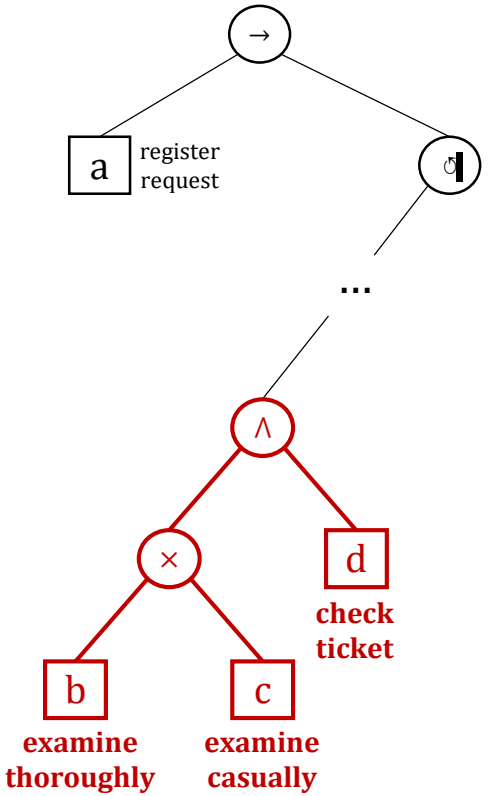
3. Process Tree

시작점부터 Operator와 Activity의 관계를 보며 단계별로 그림

Petri-Net



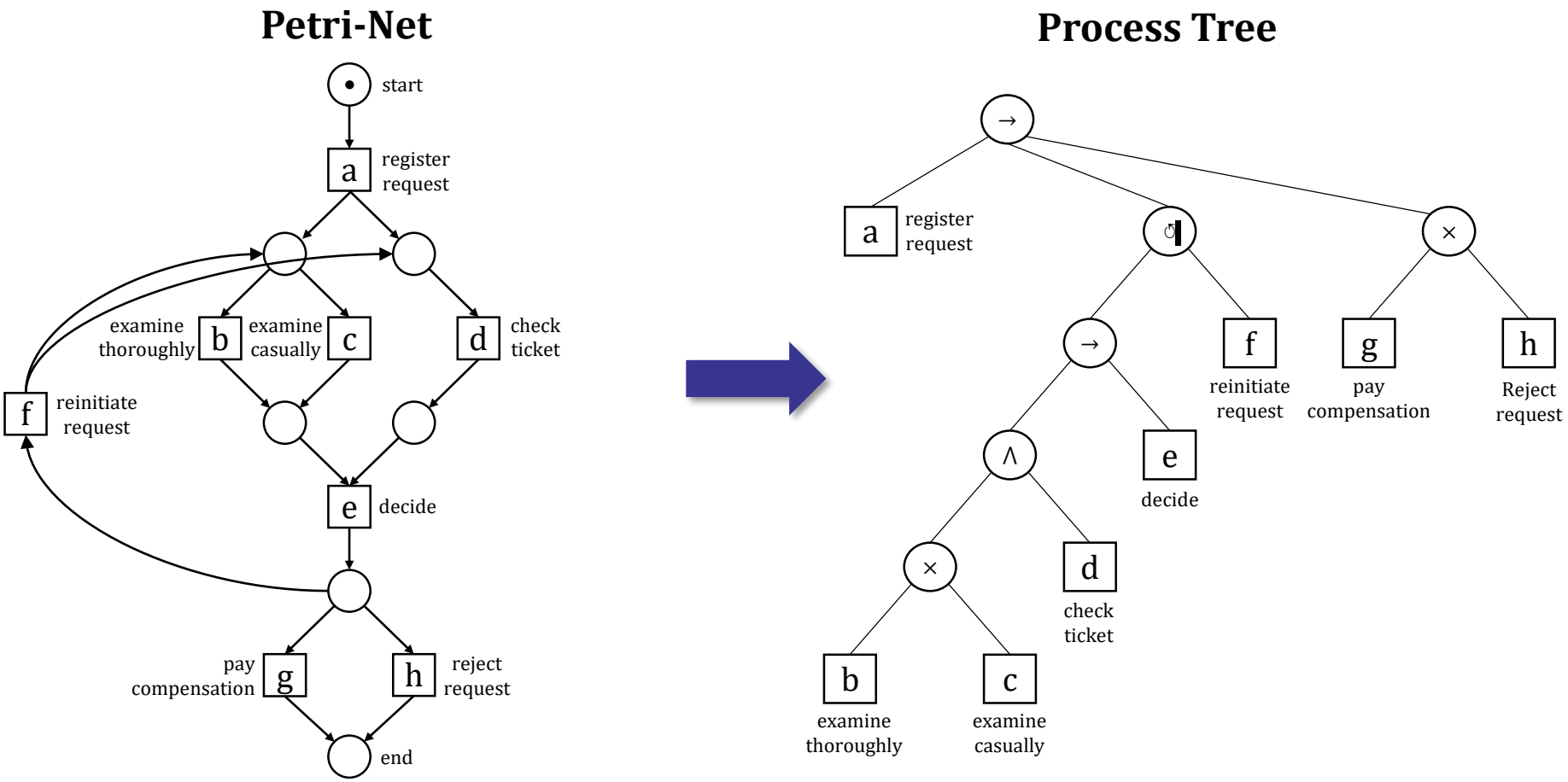
b, c가 exclusive하게 연결되고
(b, c)는 d와 parallel하게 연결된다.



2.1 Process Model

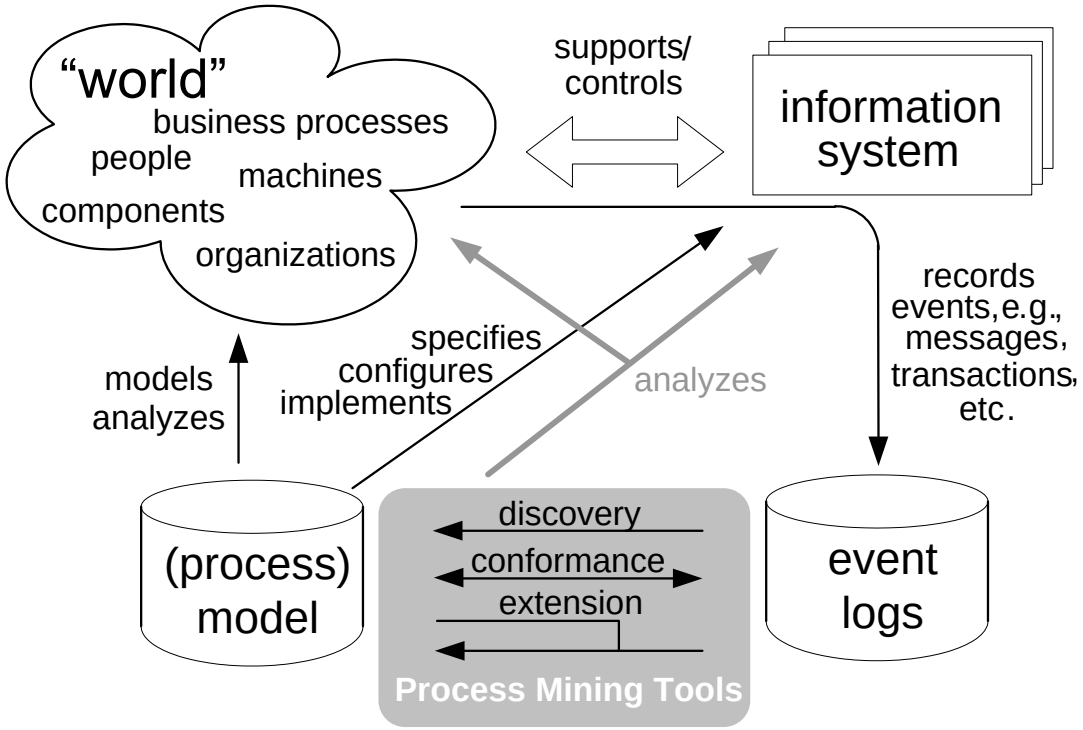
3. Process Tree

단계별로 진행하면 Process Tree 도출 가능



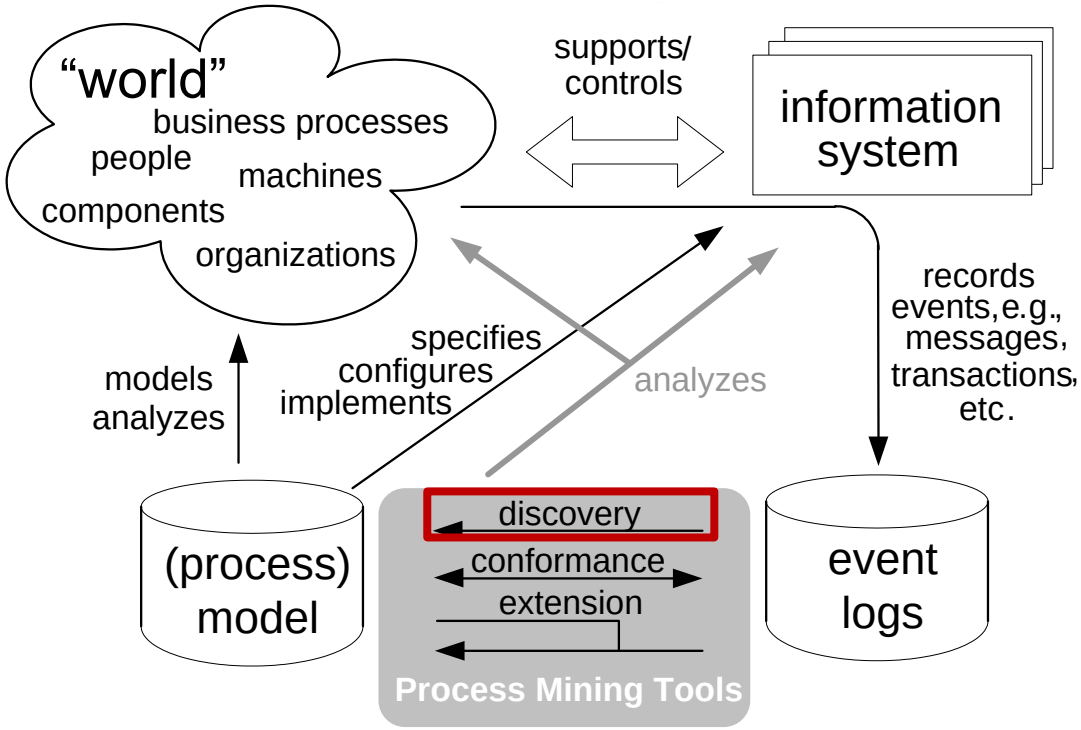
2.2 Process Discovery

1. Types of PM Algorithm

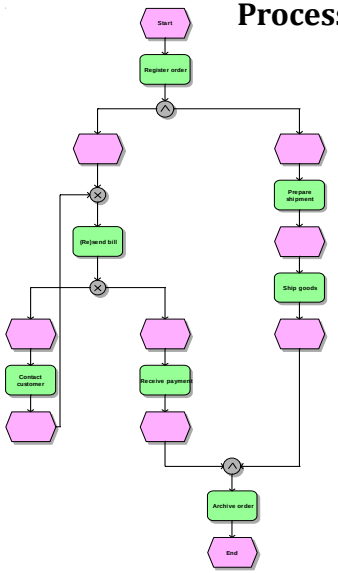


2.2 Process Discovery

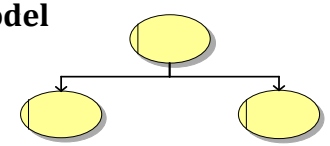
1. Types of PM Algorithm - ① discovery



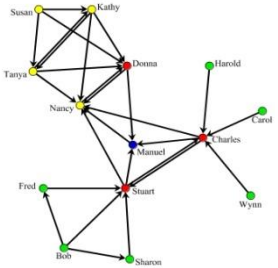
Process Model



Organizational Model

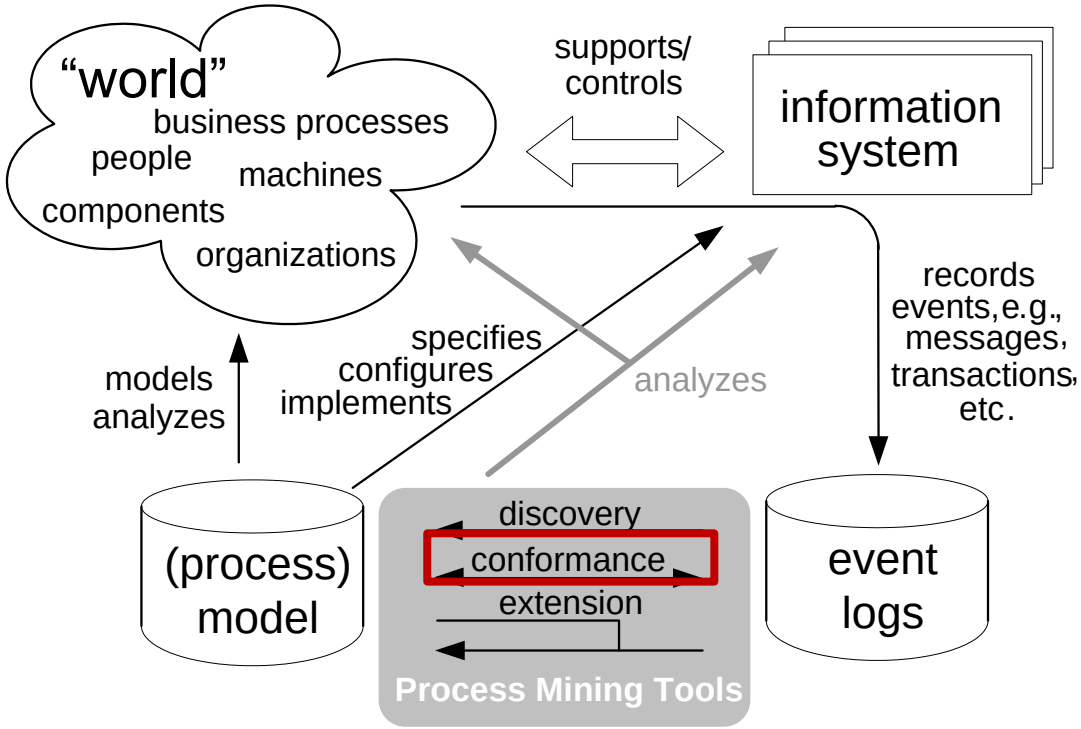


Social Network

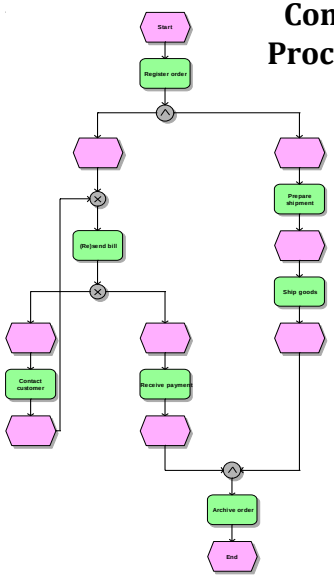


2.2 Process Discovery

1. Types of PM Algorithm - ② conformance



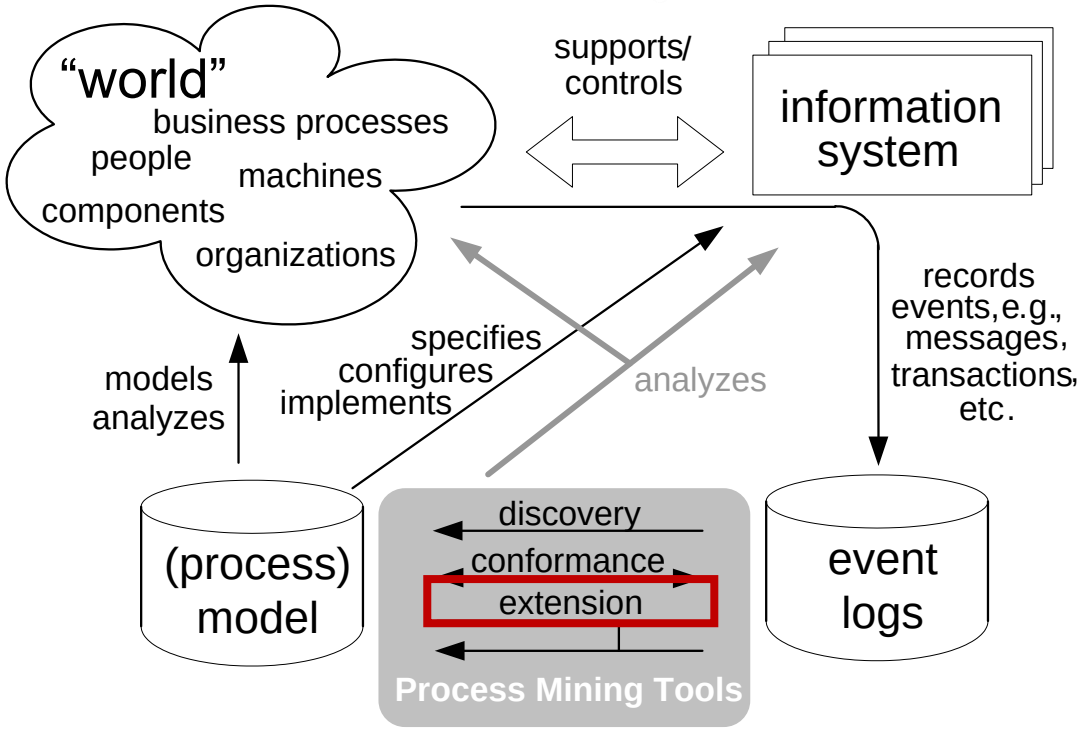
Compliance Process Model



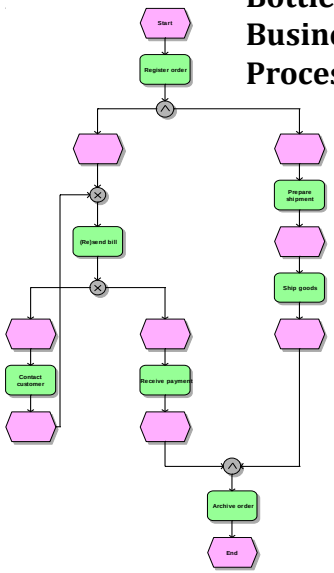
Auditing/Security

2.2 Process Discovery

1. Types of PM Algorithm – ③ extension



Bottlenecks/
Business Rules
Process Model

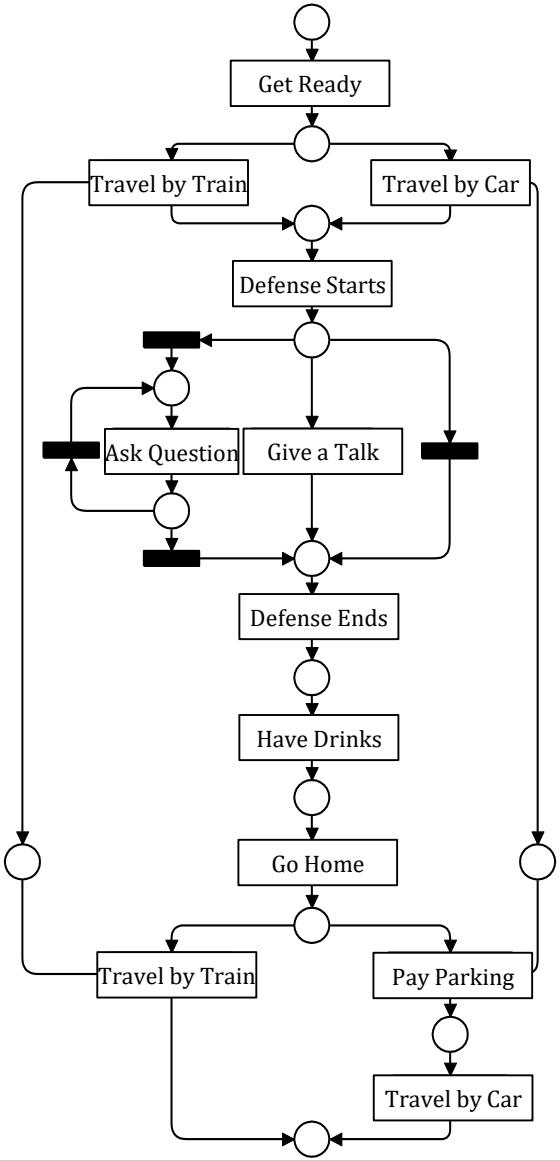


Performance Analysis

2.2 Process Discovery

2. Common Construct

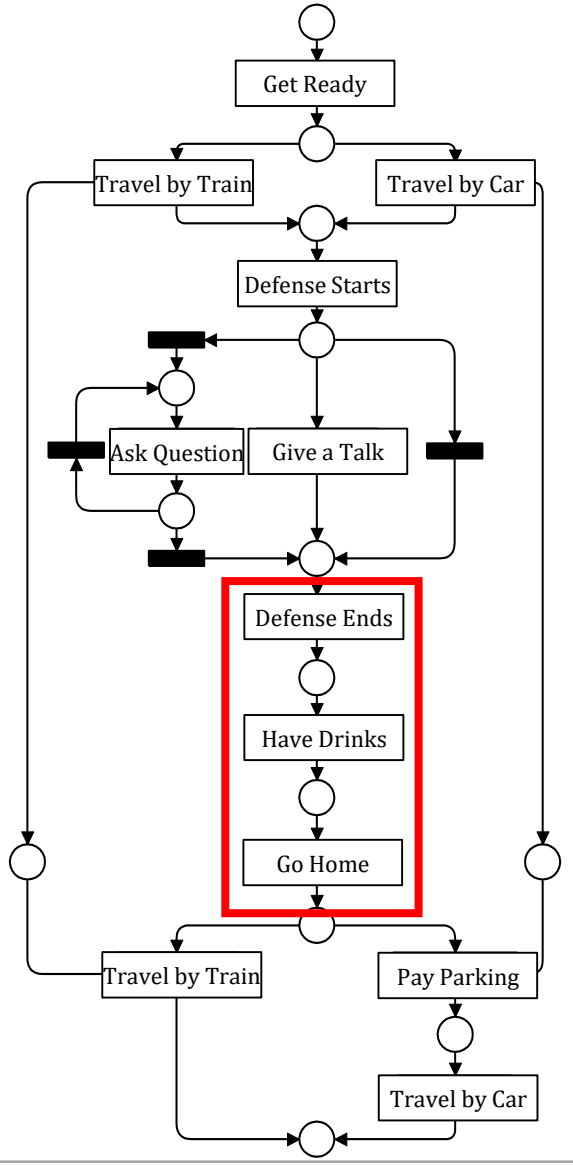
- Sequence
- Splits
- Joins
- Loops
- Non-Free Choice
- Invisible Tasks
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

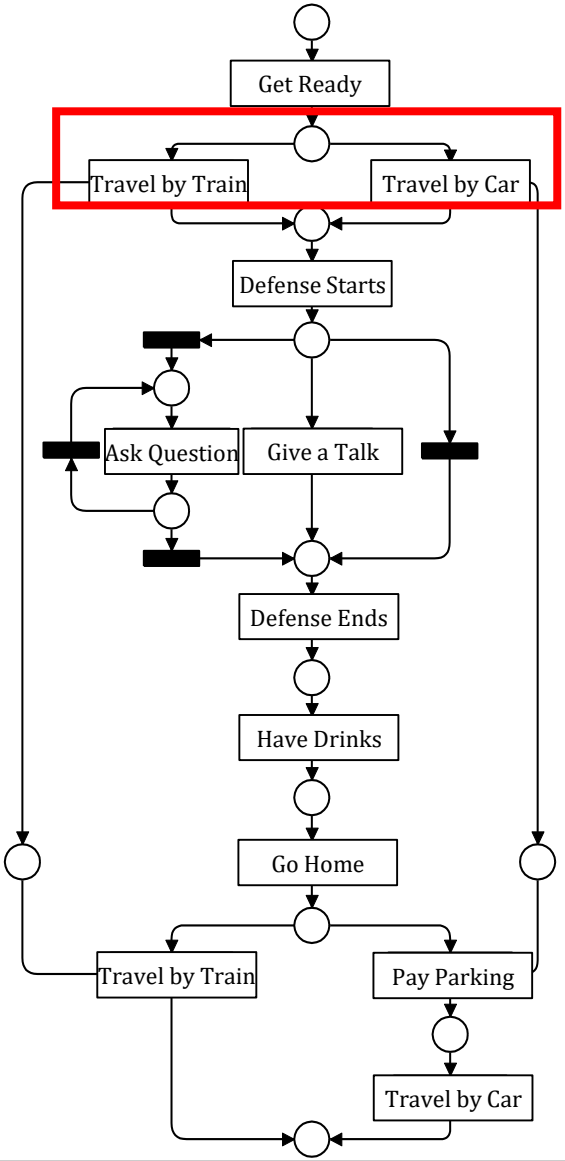
- **Sequence**
- Splits
- Joins
- Loops
- Non-Free Choice
- Invisible Tasks
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

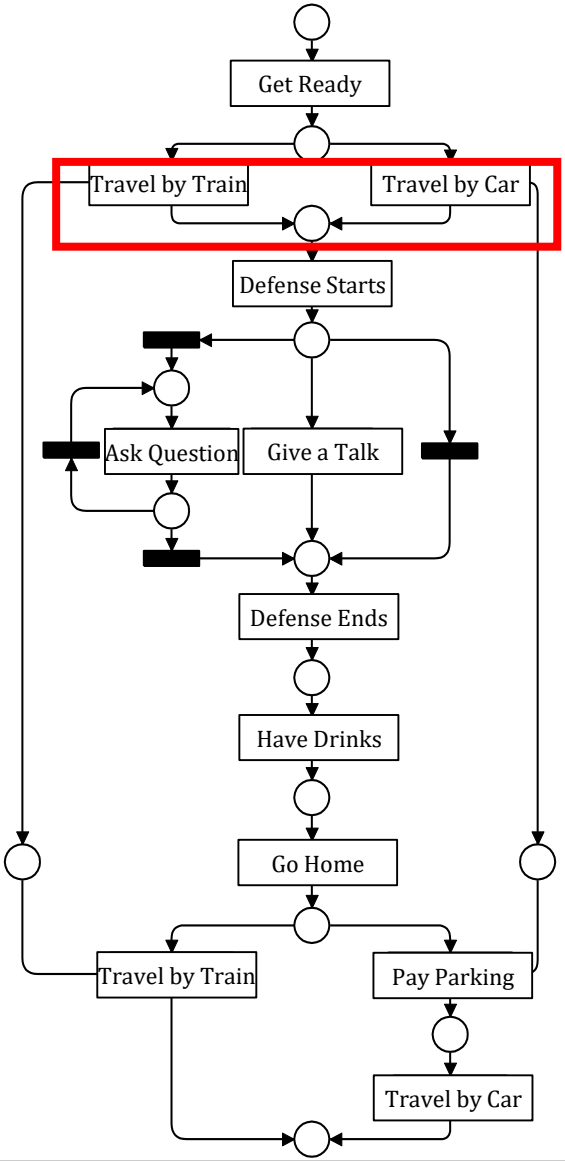
- Sequence
- **Splits**
- Joins
- Loops
- Non-Free Choice
- Invisible Tasks
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

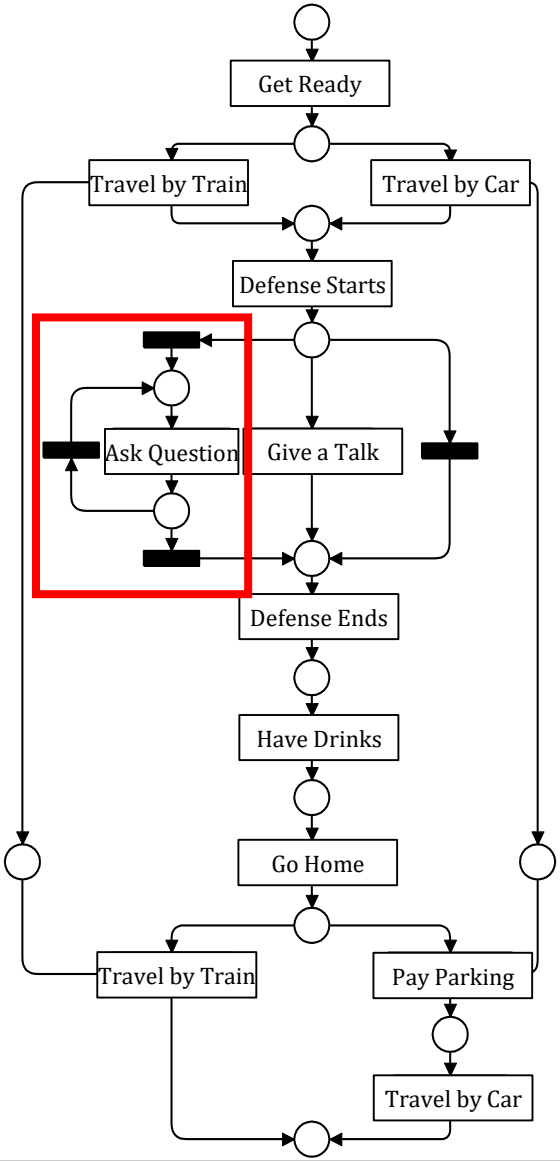
- Sequence
- Splits
- **Joins**
- Loops
- Non-Free Choice
- Invisible Tasks
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

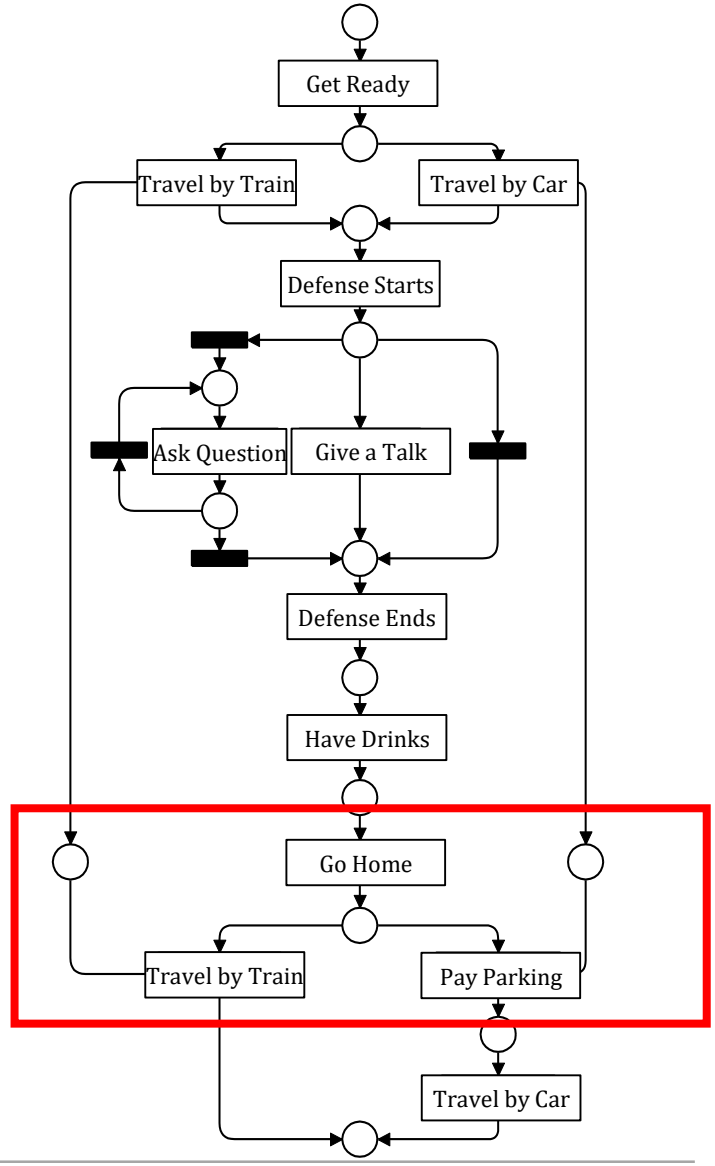
- Sequence
- Splits
- Joins
- **Loops**
- Non-Free Choice
- Invisible Tasks
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

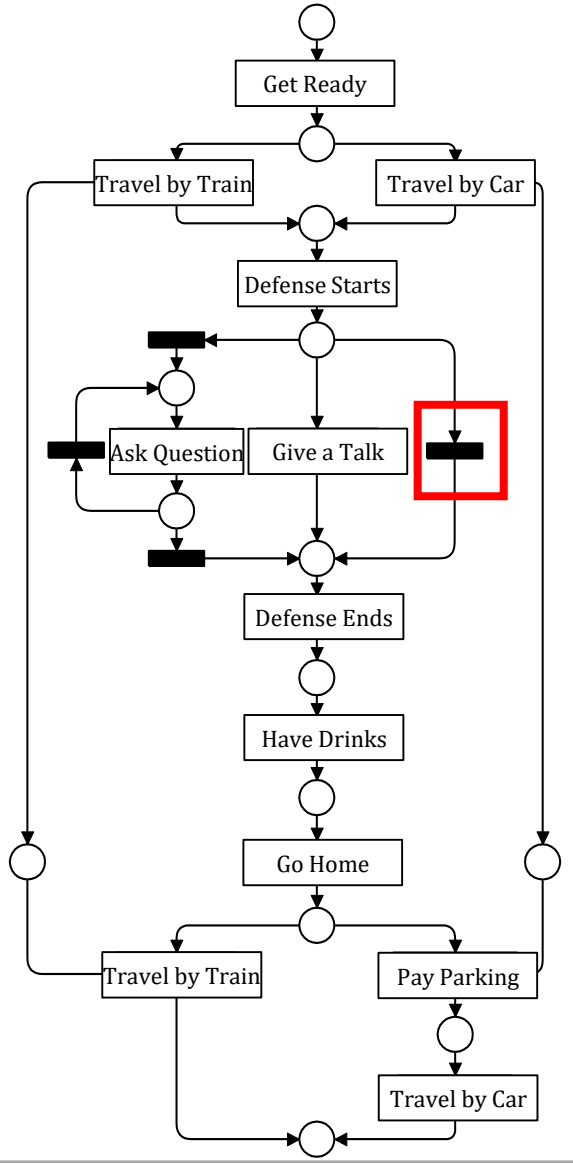
- Sequence
- Splits
- Joins
- Loops
- **Non-Free Choice**
- Invisible Tasks
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

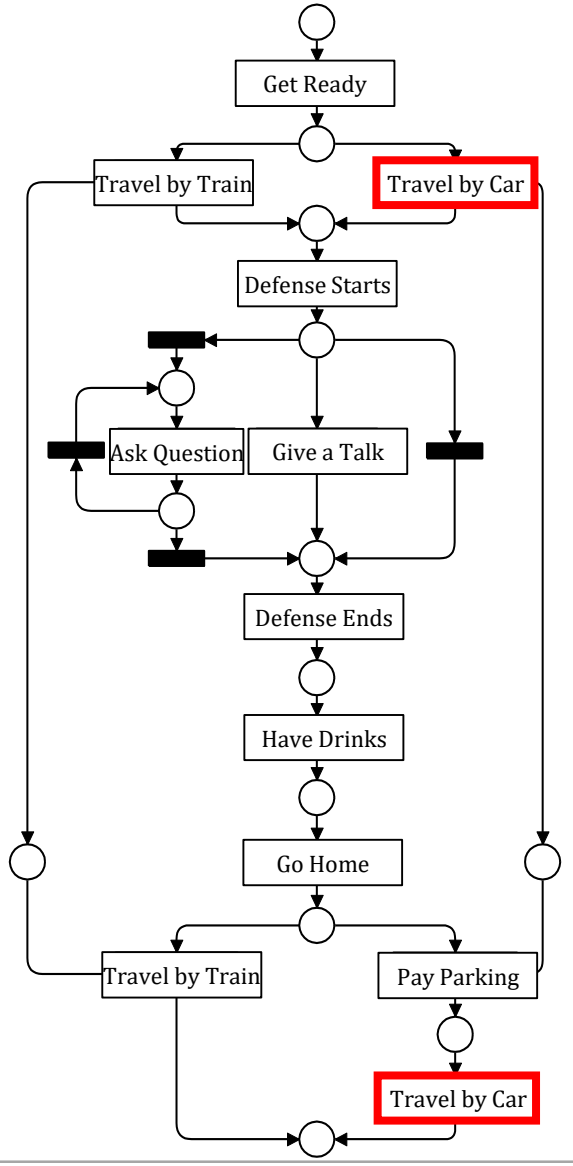
- Sequence
- Splits
- Joins
- Loops
- Non-Free Choice
- **Invisible Tasks**
- Duplicate Tasks



2.2 Process Discovery

2. Common Construct

- Sequence
- Splits
- Joins
- Loops
- Non-Free Choice
- Invisible Tasks
- **Duplicate Tasks**

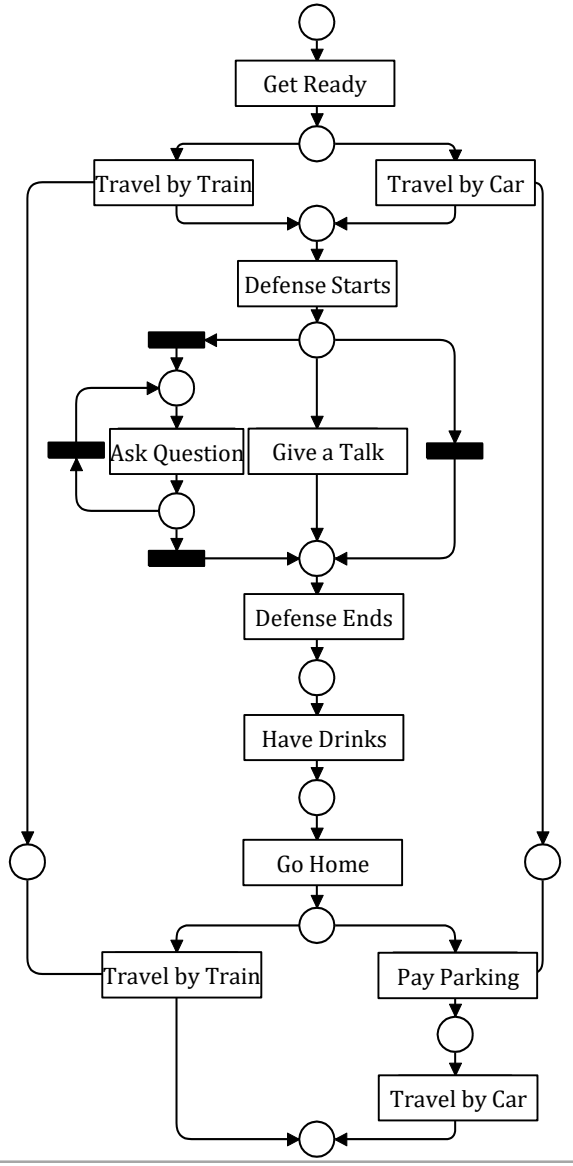


2.2 Process Discovery

2. Common Construct

- Sequence
- Splits
- Joins
- Loops
- Non-Free Choice
- Invisible Tasks
- Duplicate Tasks

+ Noise in Logs!



2.2 Process Discovery

3. Input Format

- Example of Event Log - Case, Event, Case ID, Activity, Timestamp

Patient	Activity	Timestamp	Doctor	Age	Cost
5781	Make X-ray	23-1-2014@10.30	Dr. Jones	45	70
5833	Blood test	23-1-2014@10.27	Dr. Scott	24	40
5781	Blood test	23-1-2014@10.49	Dr. Scott	45	40
5781	CT scan	23-1-2014@11.10	Dr. Fox	45	1200
5833	Surgery	23-1-2014@12.34	Dr. Scott	24	2300
5781	Handle payment	23-1-2014@12.41	Carol Hope	45	0

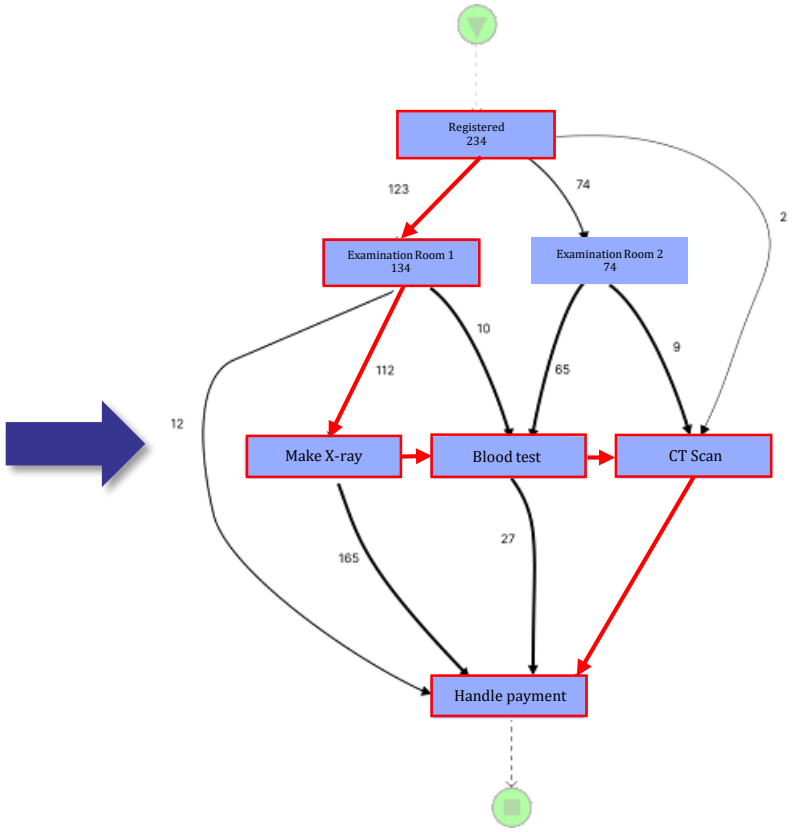
Case : Each Patient	**Single task**
Event : Each Row	**Unit of Case**
Case ID : Patient Num	**Unique value that distinguishes the Case**
Activity	**Activity at an event**
Timestamp	**The Time when the Activity starts**

2.2 Process Discovery

3. Input Format

Patient	Activity	Timestamp	Doctor	Age	Cost
5781	Registered	23-1-2014@10.00	Dr. Han	45	10
5781	Examination Room 1	23-1-2014@10.15	Dr. Han	45	10
5781	Make X-ray	23-1-2014@10.30	Dr. Jones	45	70
5833	Blood test	23-1-2014@10.27	Dr. Scott	24	40
5781	Blood test	23-1-2014@10.49	Dr. Scott	45	40
5781	CT scan	23-1-2014@11.10	Dr. Fox	45	1200
5833	Surgery	23-1-2014@12.34	Dr. Scott	24	2300
5781	Handle payment	23-1-2014@12.41	Carol Hope	45	0

Event logs



Model

2.2 Process Discovery

4. Relationship

Event-Log

Trace Number	Trace
Trace 1	A B C D
Trace 2	A C B D
Trace 3	E F

Direct succession: >

x>y iff for some case
x is directly followed by y

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

Causality: →

x→y iff **x>y** and **not y>x**

Parallel: ||

x||y iff **x>y** and **y>x**

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

→

A→B
A→C
B→D
C→D
E→F

B||C
C||B

Unrelated: #

x#y iff **not x>y**
and **not y>x**

A#D	D#A	F#A
A#E	D#E	F#B
A#F	D#F	F#C
B#E	E#A	F#D
B#F	E#B	
C#E	E#C	
C#F	E#D	

2.2 Process Discovery

5. Alpha Algorithm

- Event-Log → Process Model

Step 1

Patient	Activity	Timestamp	Doctor	Age	Cost
5781	Make X-ray	23-1-2014@10.30	Dr. Jones	45	70
5833	Blood test	23-1-2014@10.27	Dr. Scott	24	40
5781	Blood test	23-1-2014@10.49	Dr. Scott	45	40
5781	CT scan	23-1-2014@11.10	Dr. Fox	45	1200
5833	Surgery	23-1-2014@12.34	Dr. Scott	24	2300
5781	Handle payment	23-1-2014@12.41	Carol Hope	45	0

Read Event-Log

Step 2

```
case 5781 : task A
case 5833 : task A
case 5781 : task B
      etc
```

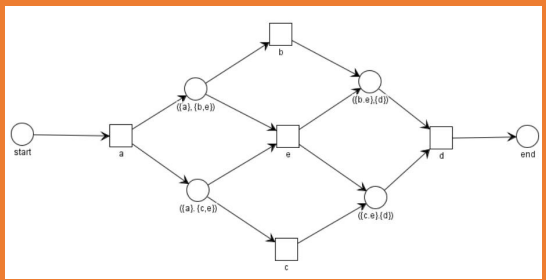
Get the set of Tasks

Step 3

A>B A>C	B||C C||B	A→B A→C

Infer the Ordering Relation

Step 4



Build the net based on inferred relations

Algorithm Step

2.2 Process Discovery

5. Alpha Algorithm – Infer the Ordering Relation

$>_{L_1} = \{(a,b), (a,c), (a,e), (b,c), (c,b), (b,d), (c,d), (e,d)\}$

$\rightarrow_{L_1} = \{(a,b), (a,c), (a,e), (b,d), (c,d), (e,d)\}$

$\#_{L_1} = \{(a,a), (a,d), (b,b), (b,e), (c,c), (c,e), (d,a), (d,d), (e,b), (e,c), (e,e)\}$

$\|_{L_1} = \{(b,c), (c,b)\}$

해당 Ordering Relations를 사용해보자

Activity들 간의 Relation 정리

Direct succession(>)

Causality(→)

Parallel(||)

Unrelated(#)

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Footprint Matrix

2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

Formalization

Let L be an event log over $T \subseteq \mathcal{A}$. $\alpha(L)$ is defined as follows.

1. $T_L = \{ t \in T \mid \exists_{\sigma \in L}\ t \epsilon \sigma \}$, 1. 모든 Activity를 찾음.
2. $T_I = \{ t \in T \mid \exists_{\sigma \in L}\ t = first(\sigma) \}$, 2. Start Activity를 찾음.
3. $T_O = \{ t \in T \mid \exists_{\sigma \in L}\ t = last(\sigma) \}$, 3. End Activity를 찾음.
4. $X_L = \{ (A,B) \mid A \subseteq T_W \wedge A \neq 0 \wedge B \subseteq T_L \wedge B \neq 0 \wedge \forall_{a \in A} \forall_{b \in B}\ a \rightarrow_L b \wedge \forall_{a_1, a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_L b_2 \}$, 4. Casuality(→) relation set을 찾음.
5. $Y_L = \{ (A,B) \in X \mid \forall_{(A',B') \in X_L}\ A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$, 5. Place set을 찾음.
6. $P_L = \{ p_{(A,B)} \mid (A,B) \subseteq Y_L \} \cup \{ i_L, o_L \}$, 6. 모델의 구조를 확인.
7. $F_L = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_L \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_L \wedge b \in B \} \cup \{ (i_L, t) \mid t \in T_I \} \cup \{ (t, o_L) \mid t \in T_O \}$, and 7. Link를 확인.
8. $a(L) = (P_L, T_L, F_L)$.

2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

$$L_1 = [< a, b, c, d >^3, < a, c, b, d >^2, < a, e, d >]$$

event log

Step 1) $T_I = \{ t \in T \mid \exists_{\sigma \in L}\ t \in \sigma \}$

1. 모든 Activity를 찾음.
{a, b, c, d, e}

Step 2) $T_I = \{ t \in T \mid \exists_{\sigma \in L}\ t = first(\sigma) \}$

2. Start Activity를 찾음.
{a}

Step 3) $T_O = \{ t \in T \mid \exists_{\sigma \in L}\ t = last(\sigma) \}$

3. End Activity를 찾음.
{d}

2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

Step 4) $X_L = \{ (A,B) \mid A \subseteq T_W \wedge A \neq 0 \wedge B \subseteq T_L \wedge B \neq 0 \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a_1, a_2 \in A} a_1 \#_L \wedge \forall_{b_1, b_2 \in B} b_1 \#_L b_2 \}$

4. Casuality(→) relation set을 찾음.

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

$$X_{L_1} = \{(\{a\}, \{b\}), (\{a\}, \{c\}), (\{a\}, \{e\}), (\{a\}, \{b, e\}), (\{a\}, \{c, e\}),$$
$$(\{b\}, \{d\}), (\{c\}, \{d\}), (\{e\}, \{d\}), (\{b, e\}, \{d\}), (\{c, e\}, \{d\})\}$$

2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

Step 5) $Y_L = \{ (A,B) \in X \mid \forall_{(A',B') \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$

5. Place set을 찾음.

$$X_{L_1} = \{(\{a\},\{b\}),(\{a\},\{c\}),(\{a\},\{e\}),(\{a\},\{b,e\}),(\{a\},\{c,e\}),$$
$$(\{b\},\{d\}),(\{c\},\{d\}),(\{e\},\{d\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

↓ Remove Not Maximal

$$X_{L_1} = \{(\cancel{\{a\},\{b\}}),(\cancel{\{a\},\{c\}}),(\cancel{\{a\},\{e\}}),(\{a\},\{b,e\}),(\{a\},\{c,e\}),$$
$$(\cancel{\{b\},\{d\}}),(\cancel{\{c\},\{d\}}),(\cancel{\{e\},\{d\}}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

↓

$$Y_{L_1} = \{(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

Step 6) $P_L = \{ p_{(A,B)} \mid (A,B) \subseteq Y_L \} \cup \{ i_L, o_L \}$ 6. 모델의 구조를 확인.

Step 7) $F_L = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_L \wedge a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_L \wedge b \in B \} \cup \{ (i_L, t) \mid t \in T_I \}$

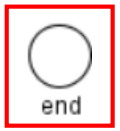
$\cup \{ (t,o_L) \mid t \in T_O \}$, and

Step 8) $a(L) = (P_L,T_L,F_L)$ 7. Link를 확인.

2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

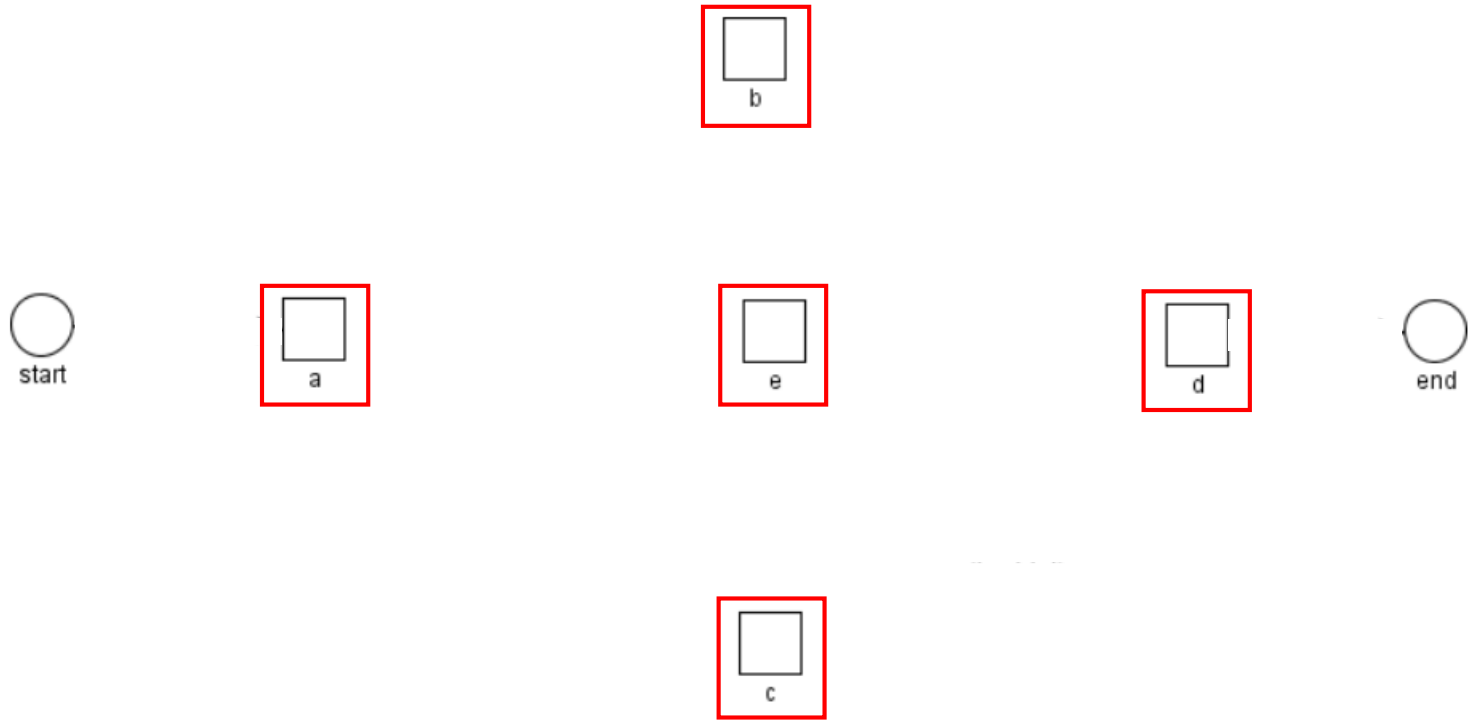
Step 1) Drawing “Start place” and “End place”



2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

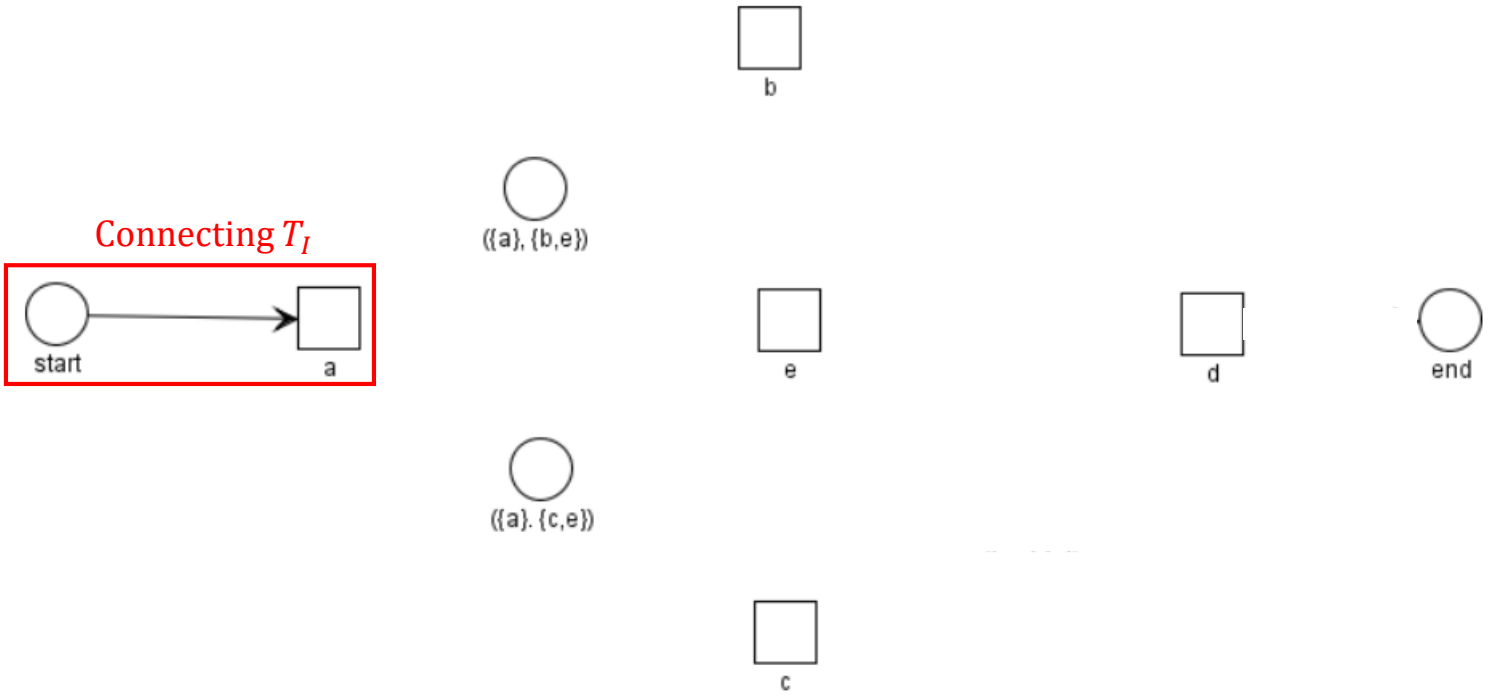
Step 2) Drawing all Activity in "T_L" → T_L = {a, b, c, d, e}



2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

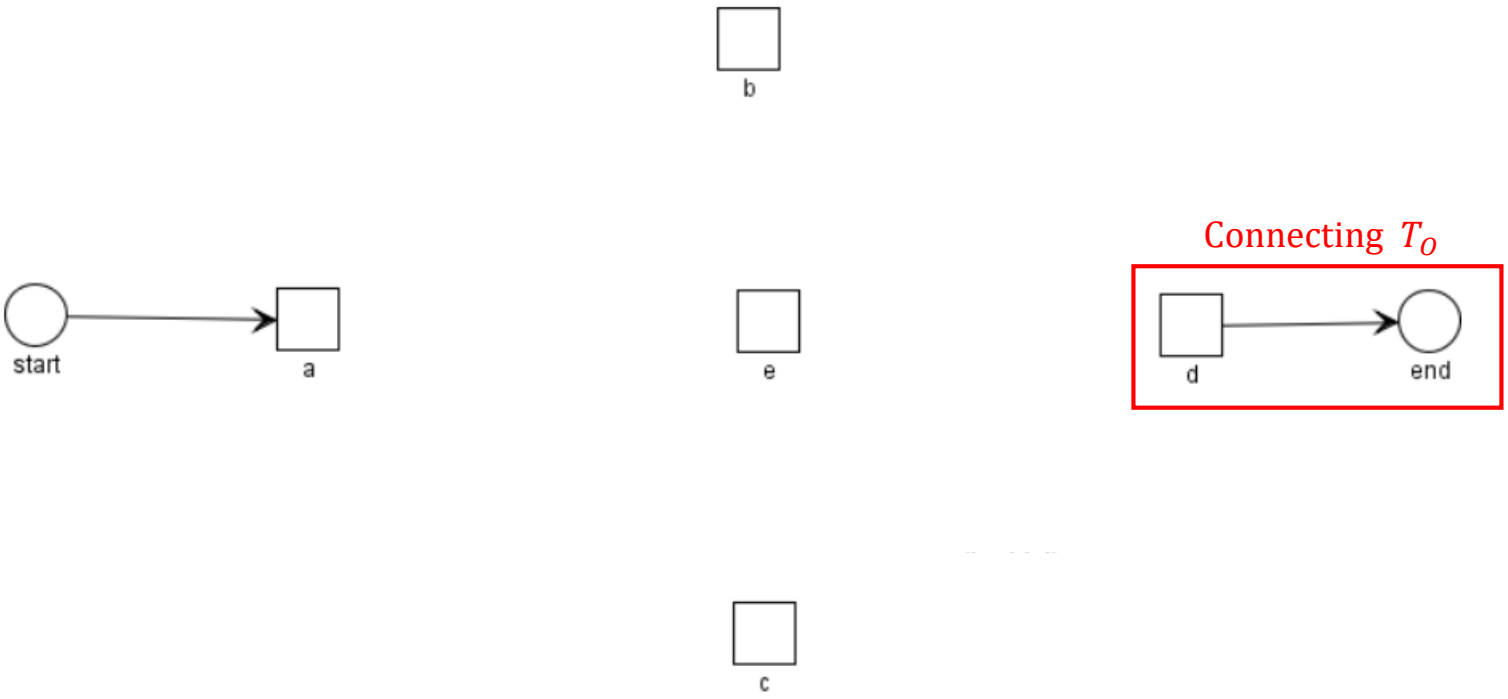
Step 3) Connecting "T_I" to "Start place" → T_I = {a}



2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

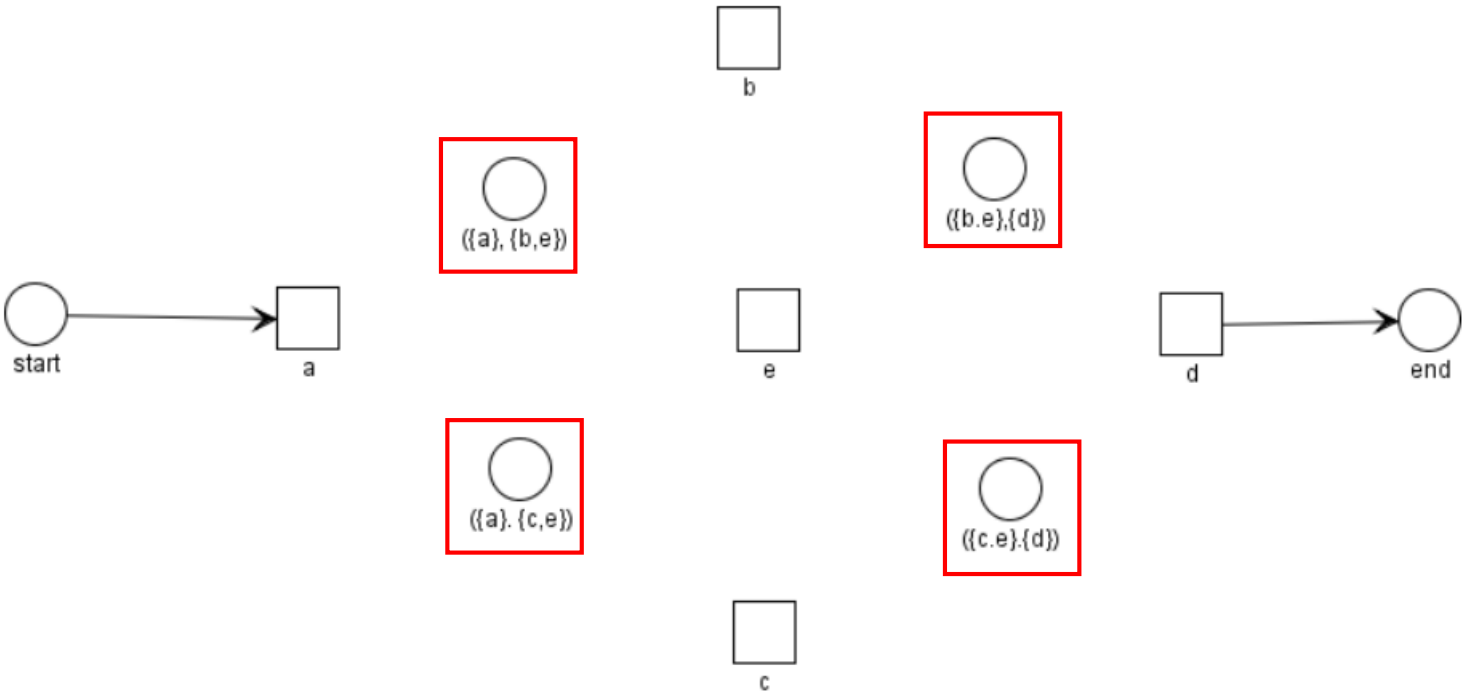
Step 4) Connecting "T_O" to "End place" → T_O = {d}



2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

Step 5) Drawing place at "Y_L" set → Y_L = { ({a},{b,e}) , ({a},{c,e}) , ({b,e},{d}) , ({c,e},{d}) }

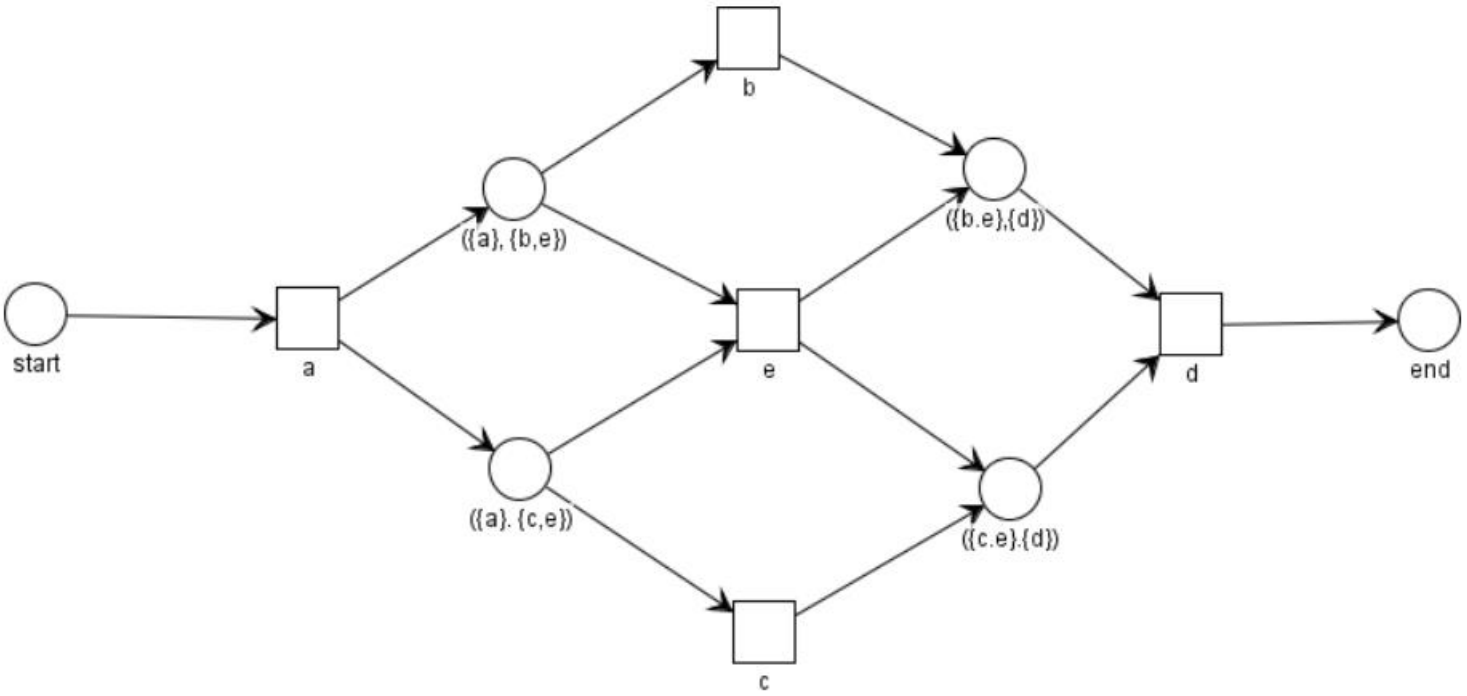


2.2 Process Discovery

5. Alpha Algorithm – Build the net based on inferred relations

Step 6) Drawing Arc

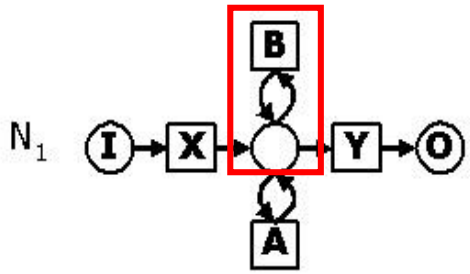
Step 7) Complete

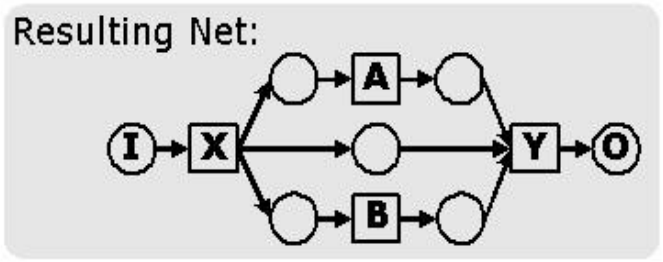


2.2 Process Discovery

5. Alpha Algorithm – Cons

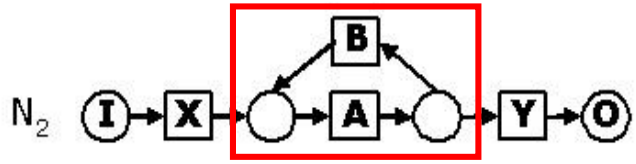
B>B and not B>B implies B→B → impossible

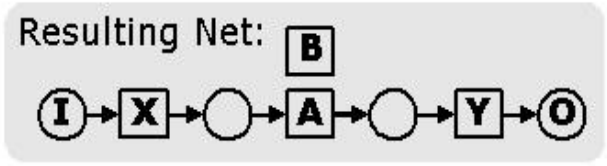




One-length

Why no short loops?





Two-length

A>B and B>A implies A||B and B||A instead of A→B and B→A

2.2 Process Discovery

6. Heuristic Algorithm - Insight

If there are 2 cases

	Workflow	Probability
Case 1	A→B	92%
Case 2	A→C	8%

We can say "A casually follow B"

In process mining, We can do it by
Heuristic Algorithm with threshold

Let W be an event log over T, and $a, b \in T$:

- $|a >_W b|$ is the number of times $a >_W b$ occurs in W,
- $a \Rightarrow_W b = \left(\frac{|a >_W b| - |b >_W a|}{|a >_W b| + |b >_W a| + 1} \right)$

2.2 Process Discovery

6. Heuristic Algorithm(HM)

- Event-Log → Process Model

Step 1

Patient	Activity	Timestamp	Doctor	Age	Cost
5781	Make X-ray	23-1-2014@10.30	Dr. Jones	45	70
5833	Blood test	23-1-2014@10.27	Dr. Scott	24	40
5781	Blood test	23-1-2014@10.49	Dr. Scott	45	40
5781	CT scan	23-1-2014@11.10	Dr. Fox	45	1200
5833	Surgery	23-1-2014@12.34	Dr. Scott	24	2300
5781	Handle payment	23-1-2014@12.41	Carol Hope	45	0

Read Event-Log

Step 2

```
case 5781 : task A
case 5833 : task A
case 5781 : task B
      etc
```

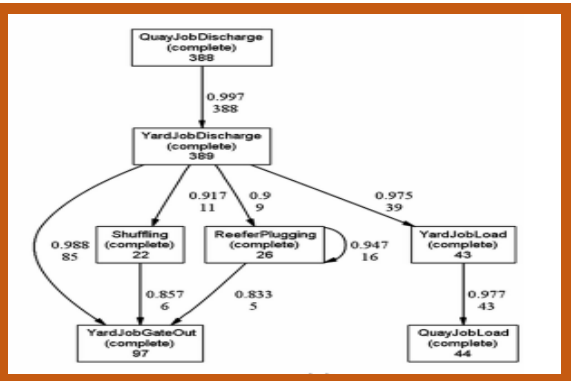
Get the set of Tasks

Step 3

A>B ~~A>C~~ | B||C C||B | A→B ~~A→C~~

Infer the Ordering Relation
Based on Inferred Relation

Step 4



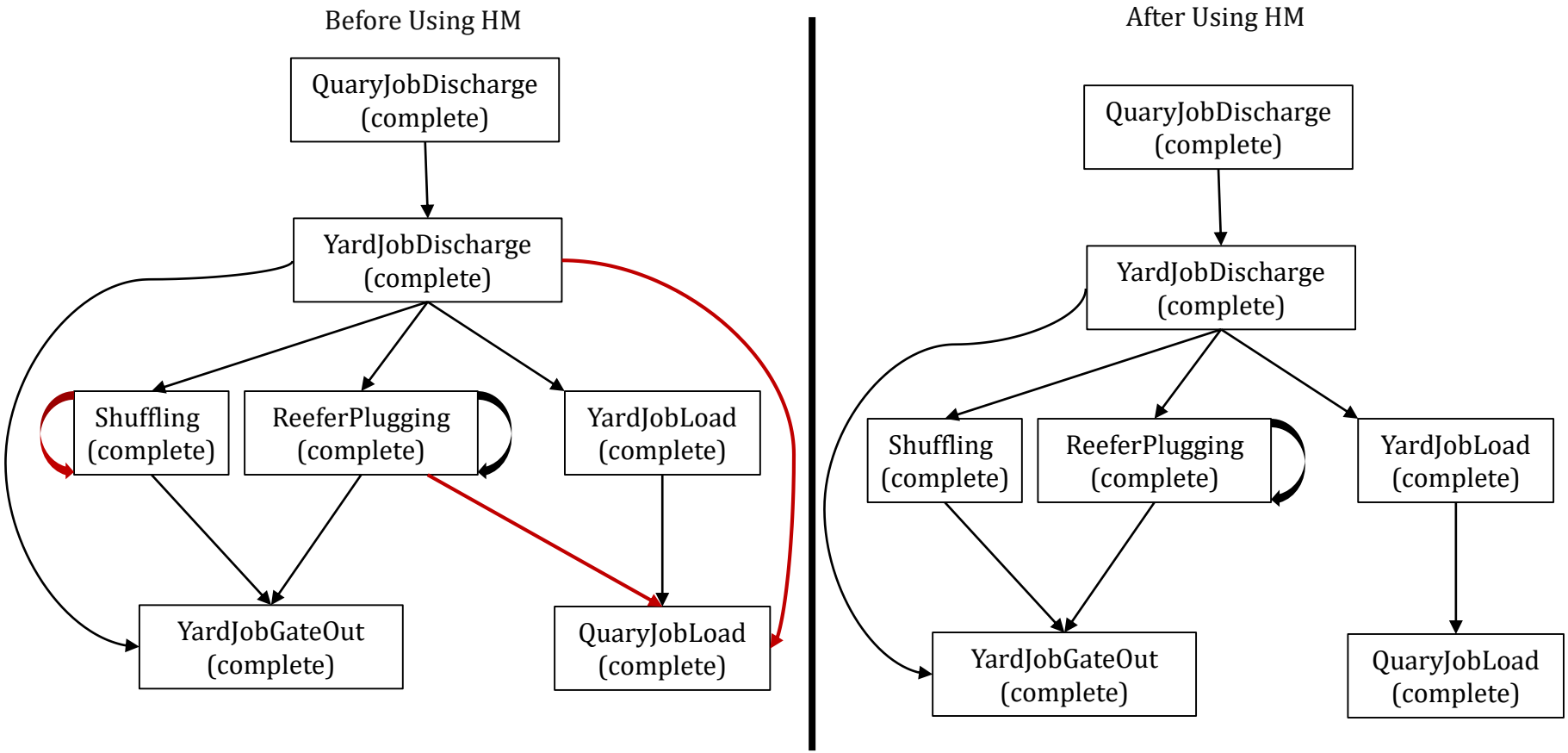
Build the net based on inferred relations

Algorithm Step

2.2 Process Discovery

6. Heuristic Algorithm(HM)

- Example of Heuristic Miner

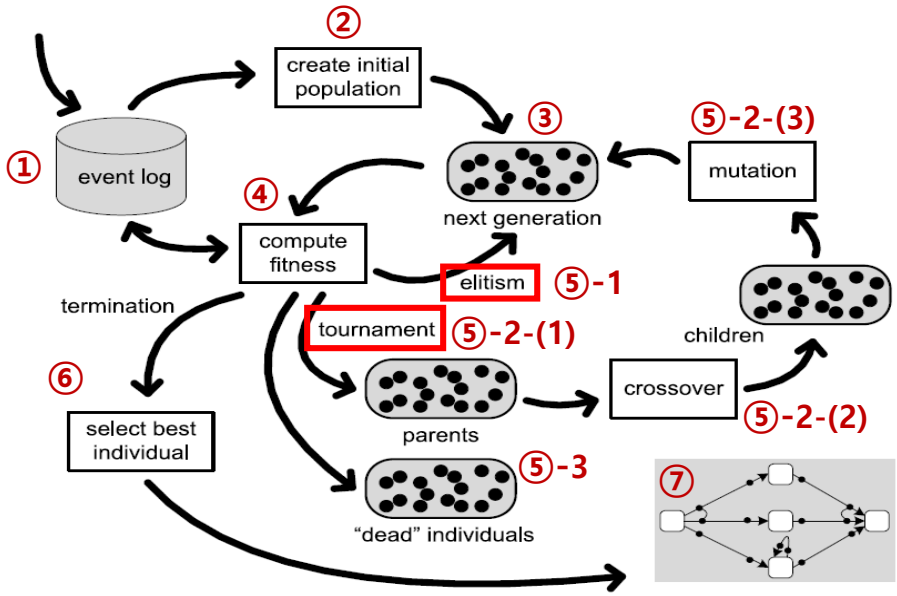


2.2 Process Discovery

7. Genetic Miner - Genetic Process Mining(GPM)

- Search technique that mimics **the process of evolution** in biological systems.
- Step ③,④,⑤ are repeated.

Compute fitness



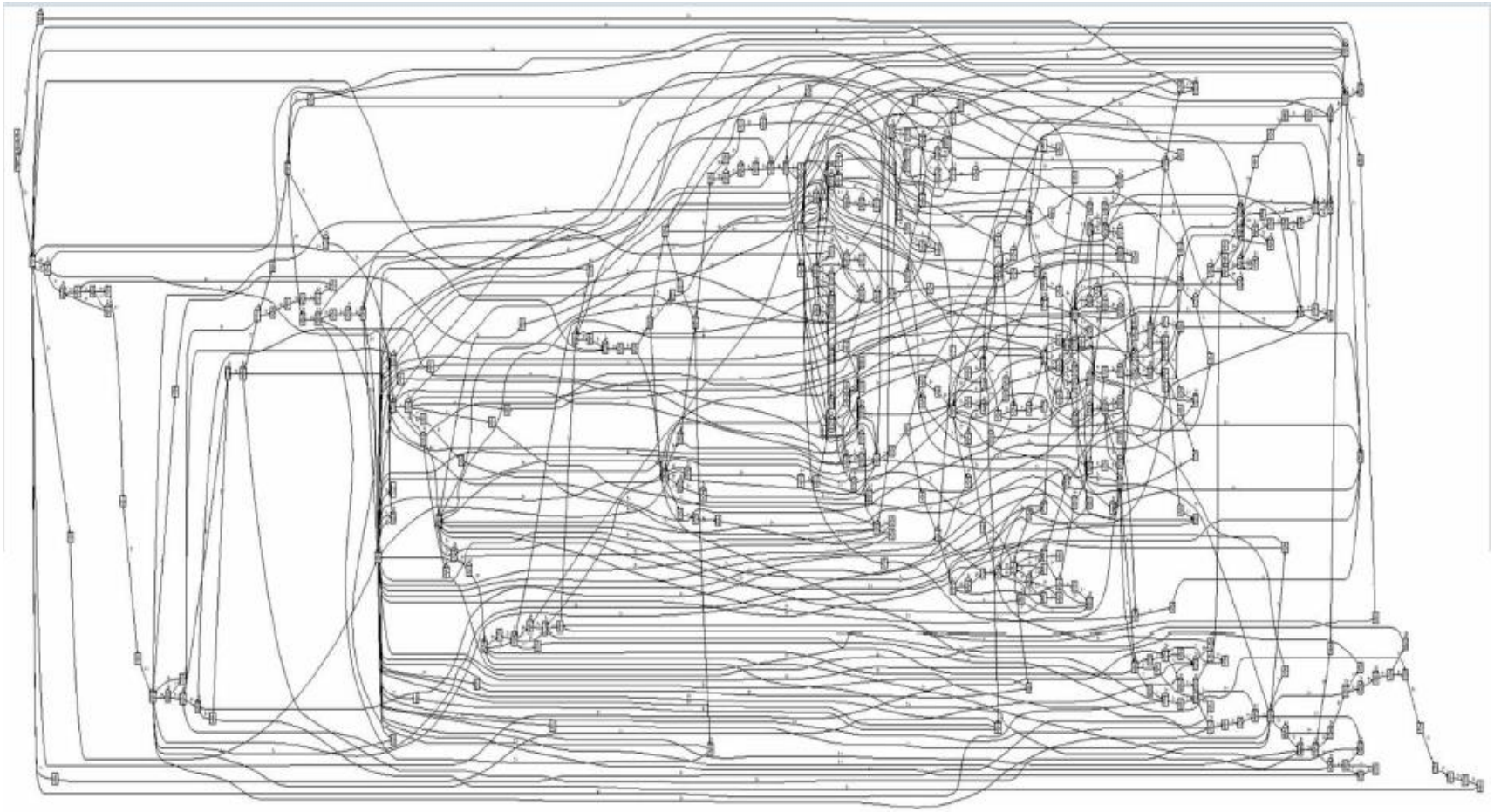
Advantages
- Tackle all common structural constructs - Robust to noise

Disadvantages
- Computational Time

2.2 Process Discovery

8. Fuzzy Miner

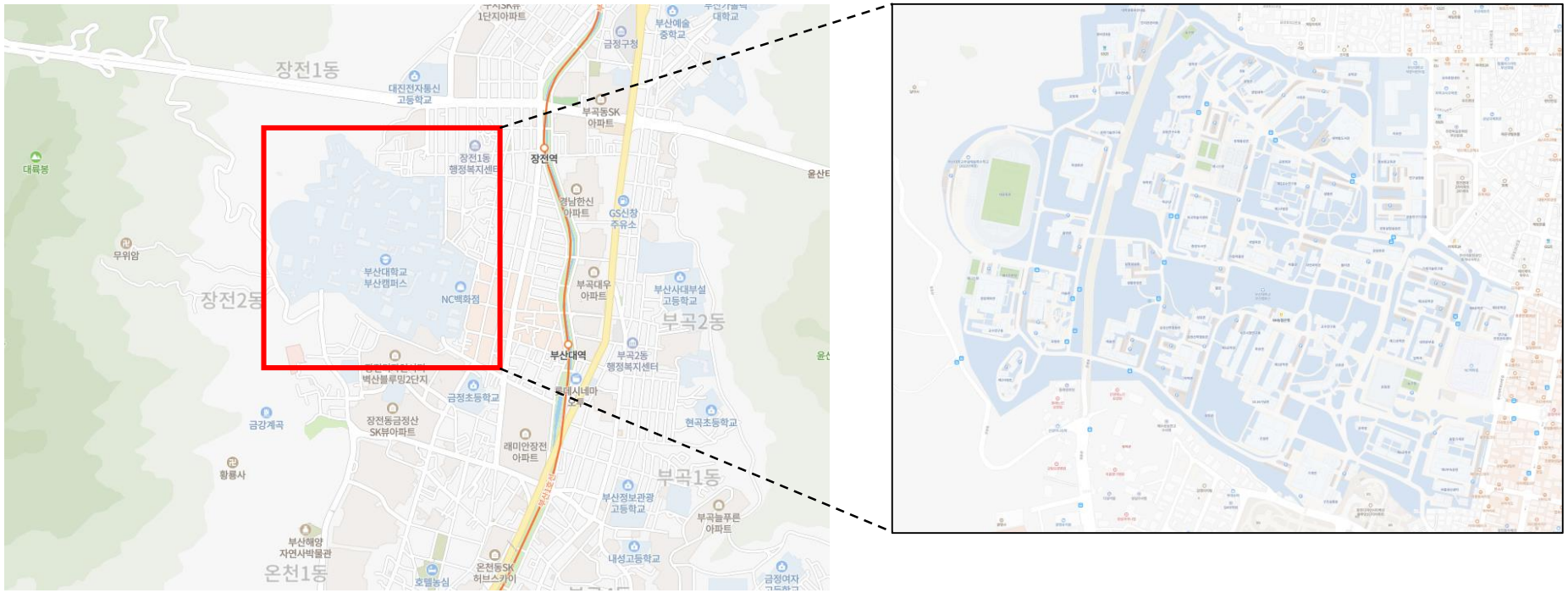
Mine less structured processes!



2.2 Process Discovery

8. Fuzzy Miner

Motivation : A Map can skips some Information



We will do it in Fuzzy Miner

2.2 Process Discovery

8. Fuzzy Miner

- **Significance : (event or link) relative importance**

  $Sig(e_i)$ $Sig(e_i, e_j)$

- **Correlation : Correlation with 2 events**

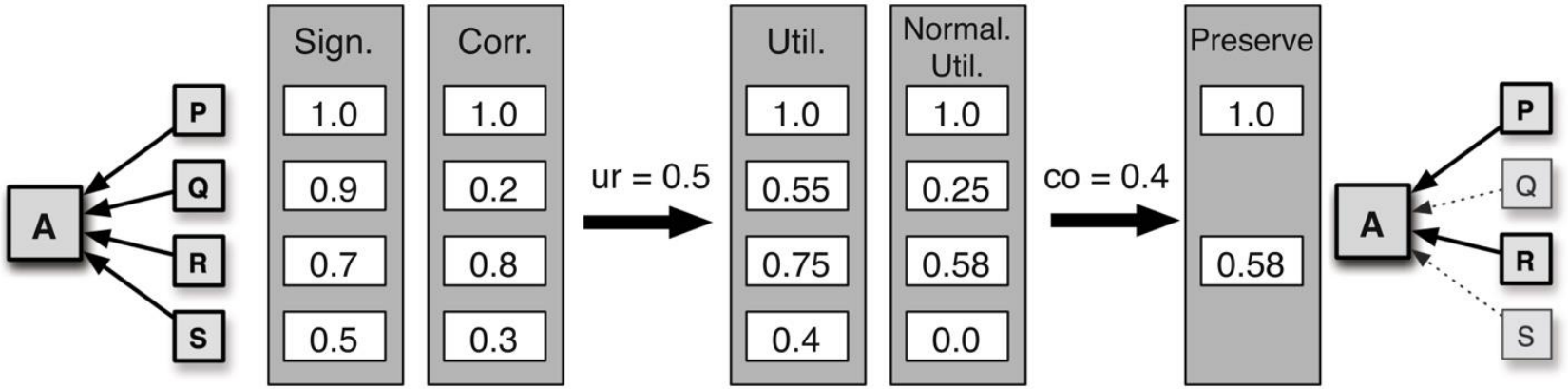
$$Rel(e_i, e_j) = \frac{1}{2} \times \frac{Sig(e_i, e_j)}{\sum_{e_x \in \sigma} Sig(e_i, e_x)} + \frac{1}{2} \times \frac{Sig(e_i, e_j)}{\sum_{e_x \in \sigma} Sig(e_x, e_j)}$$

High Significance : should exist in the model

Less Significance:

But highly correlated : in the cluster

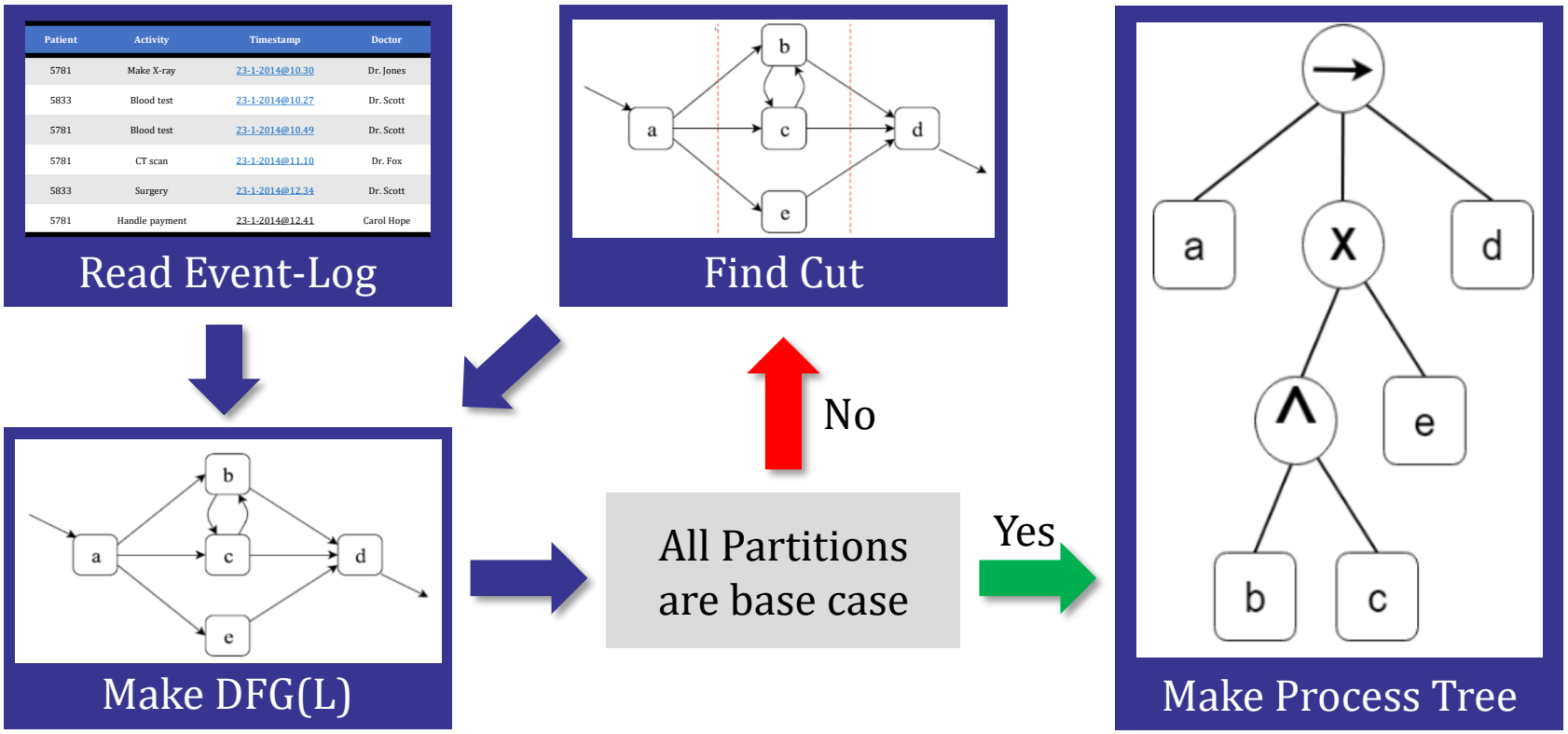
And lowly correlated : should be omitted



2.2 Process Discovery

9. Inductive Miner

- **Inductive Miner : Event Log --> Process Tree**
- Guarantee Soundness



2.2 Process Discovery

9. Inductive Miner - Make DFG(L)

- **What is DFG**
  - Directly Followed Graph
  - Shows Direct succession

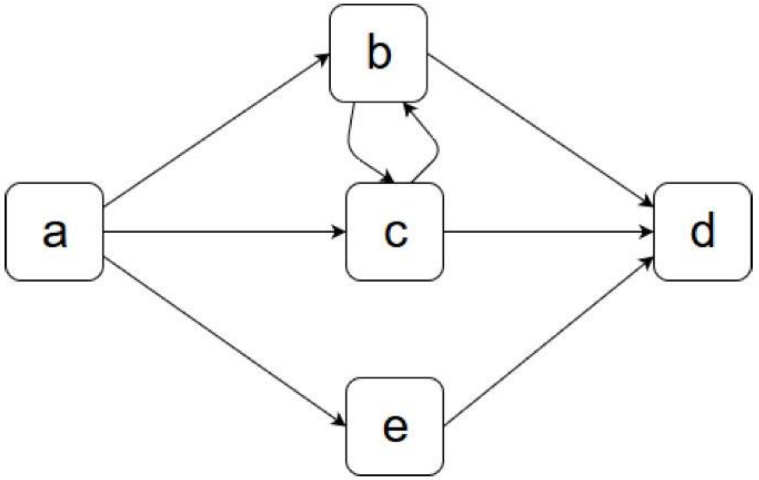
Direct succession: >

x>y iff for some case x is directly followed by y

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

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

DFG



2.2 Process Discovery

9. Inductive Miner - Make DFG(L)

$$We\ define\ G(L) = (A_L, \mapsto_L, A_L^s, A_L^e)\ with:$$

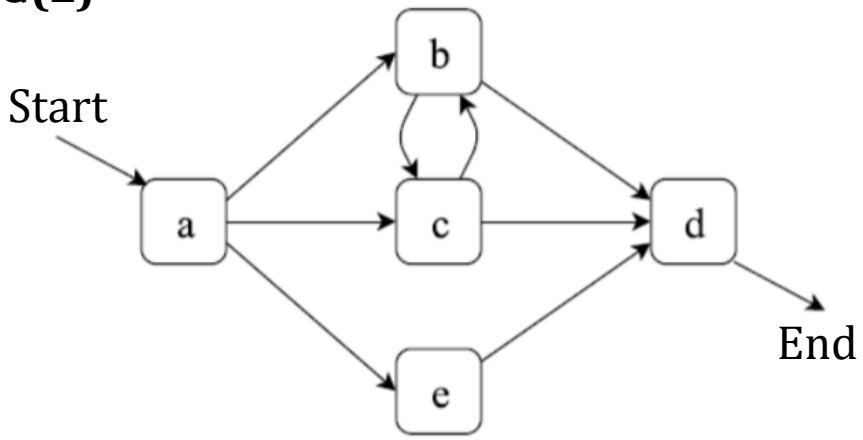
- $A_L = \{a \in \sigma \mid \sigma \in L\}(\sigma \in L \equiv L(\sigma) > 0)$ — All Activity
- $\mapsto_L = \{(a,b) \in A \times A \mid a >_L b\}$ — All Direct Succession(x>y)
- $A_L^s = \{a \in A \mid \exists_{\sigma \in L}(\sigma(1) = a)\}$ — Start Activity in every traces
- $A_L^e = \{a \in A \mid \exists_{\sigma \in L}(\sigma(|\sigma|) = a)\}$ — End Activity in every traces

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

A, B, C, D, E | A | D

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

DFG(L) is DFG + G(L)



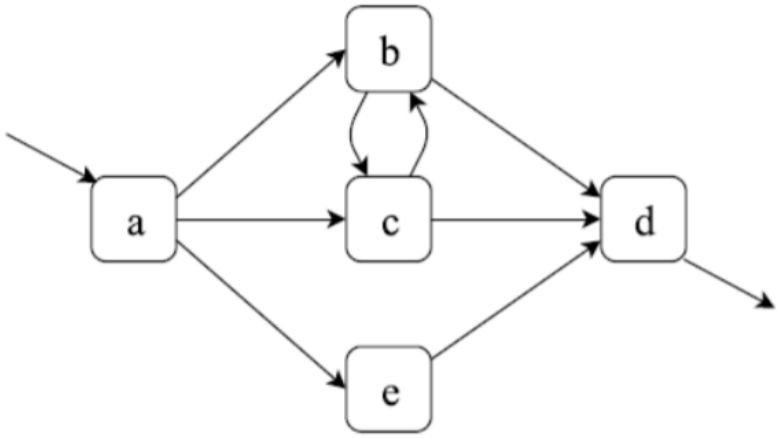
2.2 Process Discovery

9. Inductive Miner – Check Base Case

- **What is Base Case**
  - Consist of Only 1 Activity
  - Consist of Only Silent Activity

Silent Activity

Among the activities, activities that are not actually performed but are included in the process model



Here is 5 activities.

So, It's not Base Case.

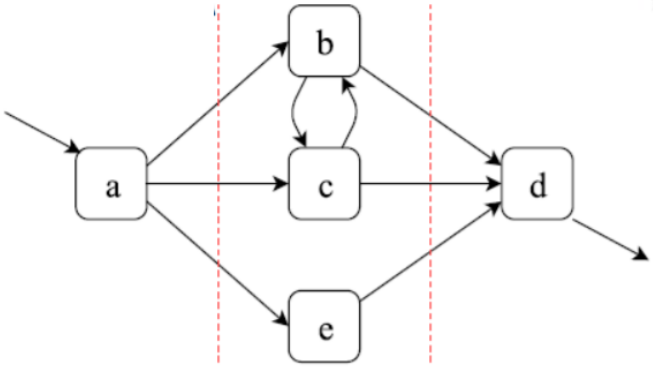
We have to find Cut.

2.2 Process Discovery

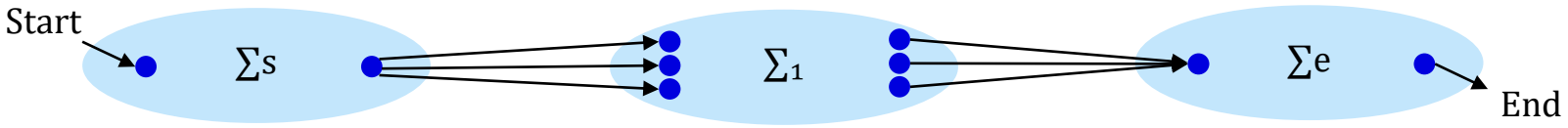
9. Inductive Miner – Find Cut

Partition is a part of DFG(L)

Partitioning below DFG(L) based on Red Line



Σ_i is a partition, Blue dot is Activity



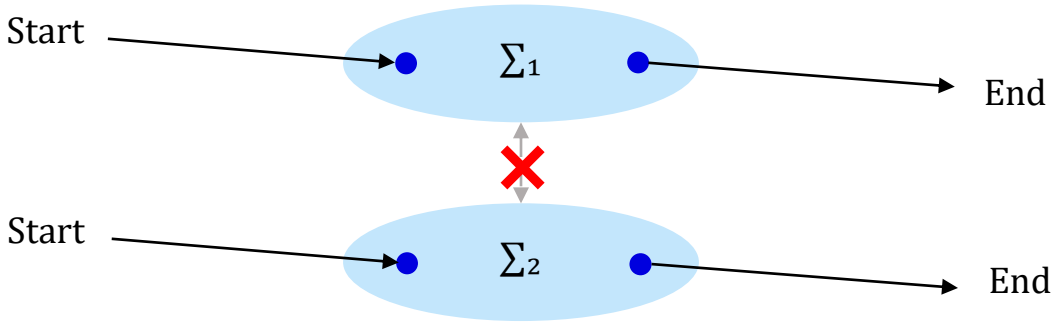
2.2 Process Discovery

9. Inductive Miner – Find Cut

Σ_i is a partition, Blue dot is Activity

1. Exclusive-Choice Cut

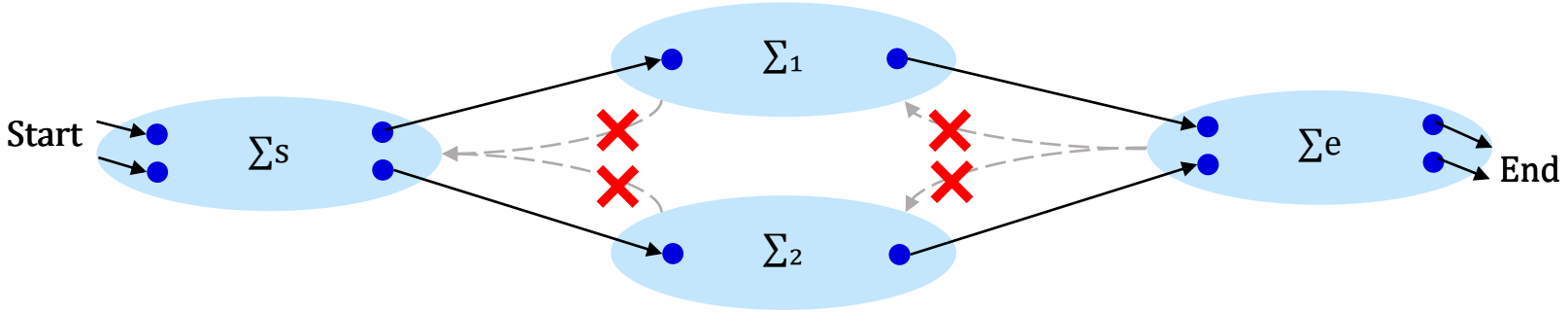
Do **not connect** Between Partitions (Σ_1, Σ_2)



2. Sequence Cut

Only **1 way connect** Between Partitions

(Σ_1 have Only Outgoing arc, Do *Not* have Incoming arc from Σ_e)



2.2 Process Discovery

9. Inductive Miner – Find Cut

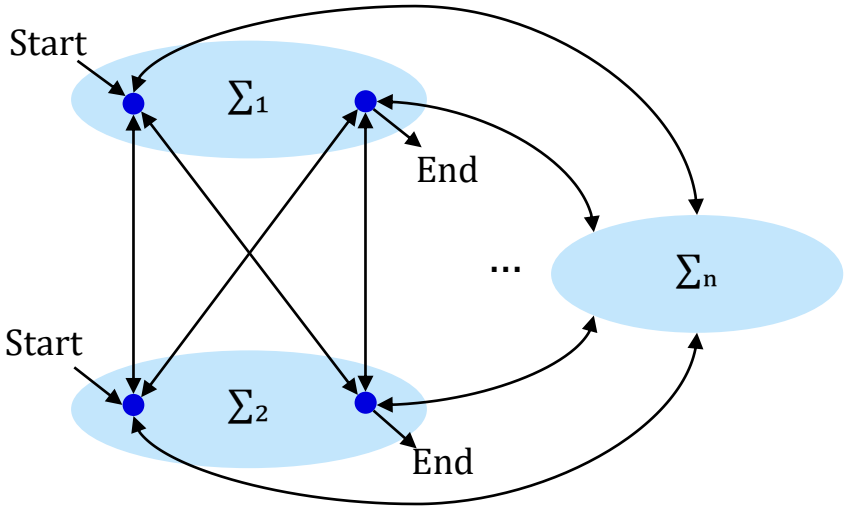
Σ_i is a partition, Blue dot is Activity

3. Parallel Cut

(1) Each partition has **Start&End Activity**

(2) Each partition's Activities have **Parallel**

Parallel: ||
x||y iff x>y and y>x



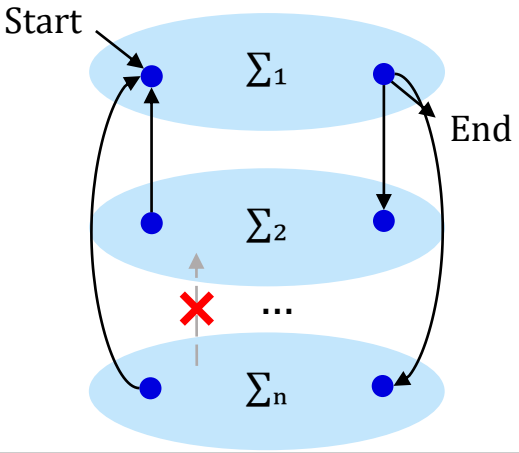
4. Redo-Loop Cut

(1) **Only 1 partition(Σ_1)** has Start & End Activity

(2) If there is Incoming(Outgoing) at Σ_1, that activity is start(end) activity

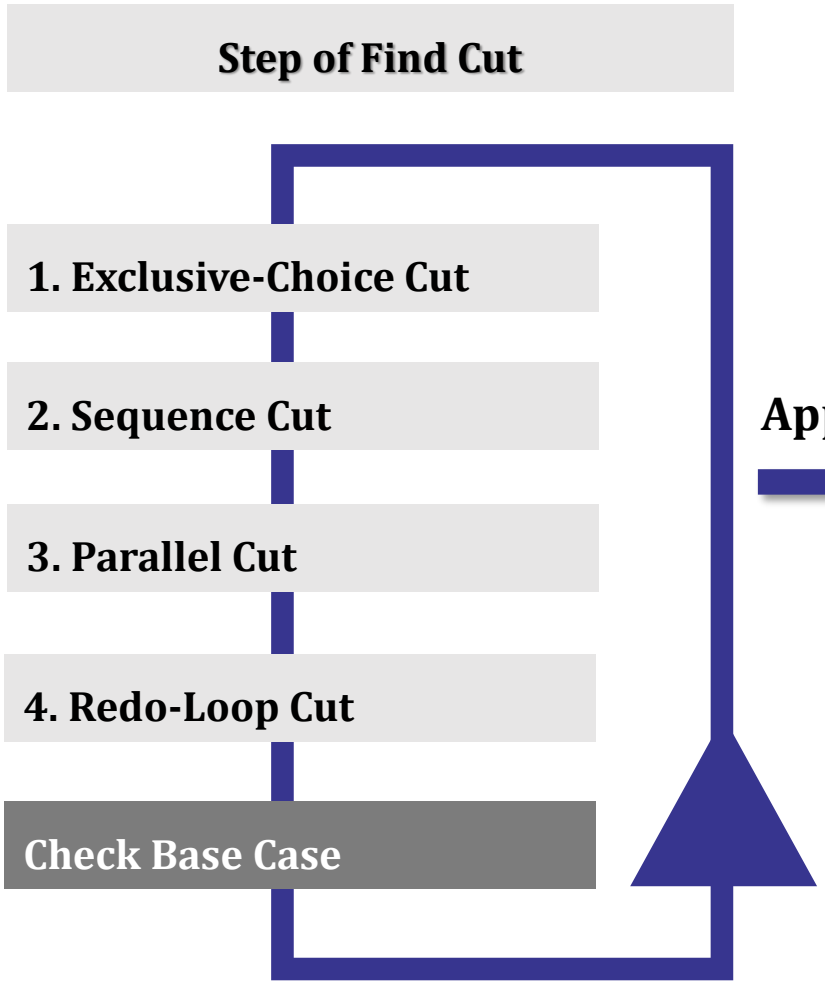
(3) There is no connections without Σ_1

(4) If there is Incoming(Outgoing) arc from Σ_1, Every Σ_1 's start(end) activity has another Incoming(Outgoing) arc to different partition

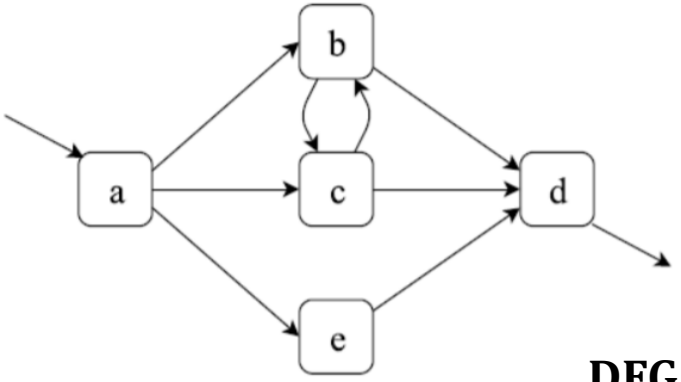


2.2 Process Discovery

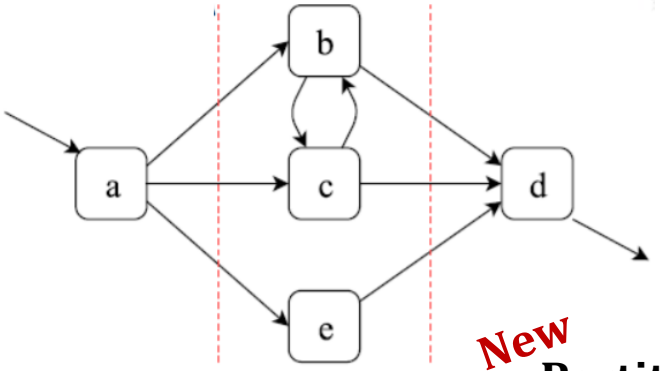
9. Inductive Miner – Find Cut



Apply Find Cut



DFG(L)

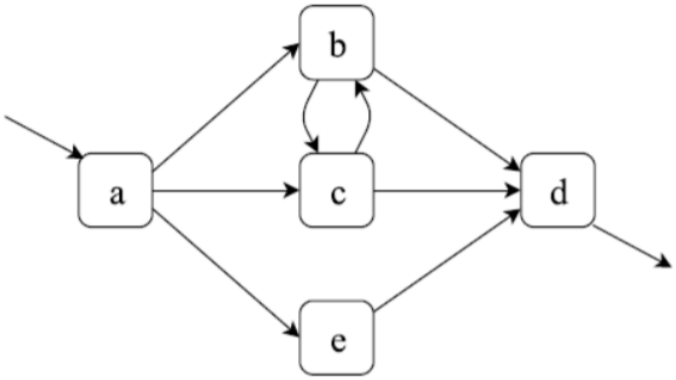


New Partition

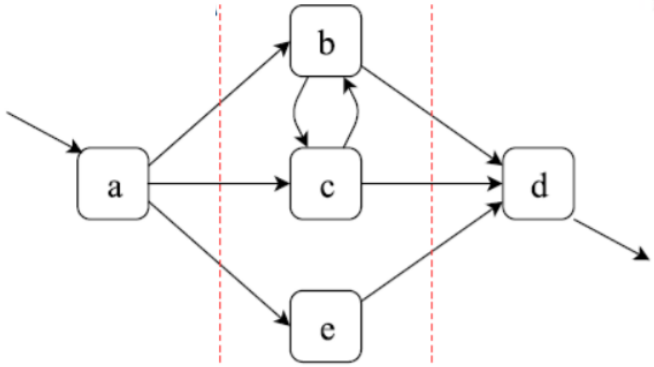
2.2 Process Discovery

9. Inductive Miner – Find Cut

Example



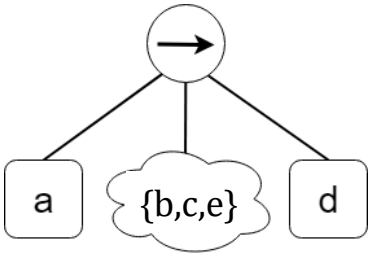
1. Exclusive-Choice Cut	Not Applicable
2. Sequence Cut	**Stop! Applicable**
3. Parallel Cut	
4. Redo-Loop Cut	



2. Sequence Cut

Only **1 way connect** Between Partitions

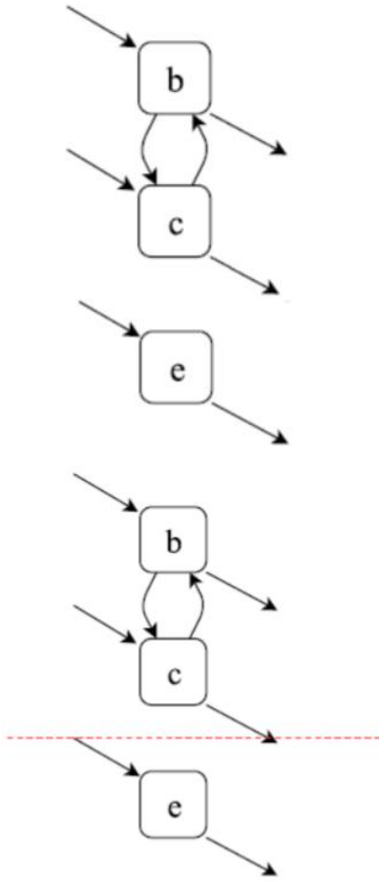
And we get process tree



2.2 Process Discovery

9. Inductive Miner – Find Cut

Check Base Case and **Drawing DFG** which is not Base Case

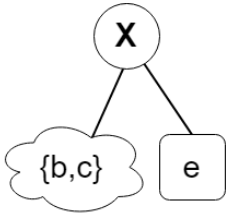


1. Exclusive-Choice Cut	**Stop! Applicable**
2. Sequence Cut	
3. Parallel Cut	
4. Redo-Loop Cut	

1. Exclusive-Choice Cut

Do **not connect** Between Partitions

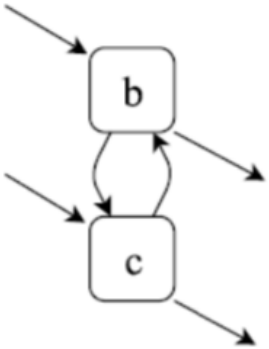
And we get process tree



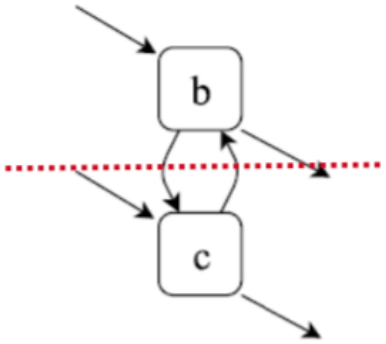
2.2 Process Discovery

9. Inductive Miner – Find Cut

Check Base Case and **Drawing DFG** which is not Base Case



1. Exclusive-Choice Cut	Not Applicable
2. Sequence Cut	Not Applicable
3. Parallel Cut	**Stop! Applicable**
4. Redo-Loop Cut	

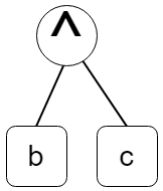


3. Parallel Cut

(1) Each partition has **Start&End Activity**

(2) Each partition's Activities have **Parallel**

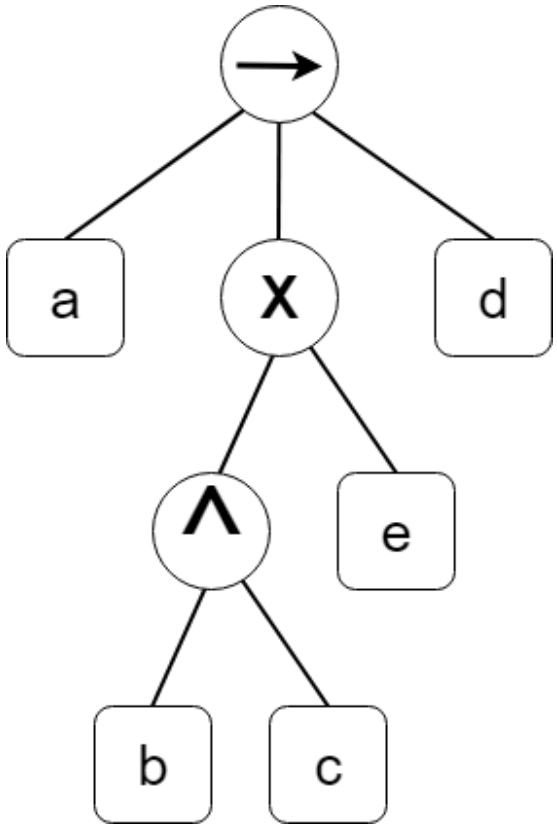
And we get process tree



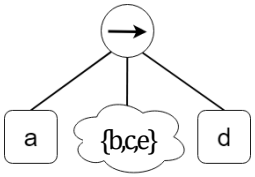
2.2 Process Discovery

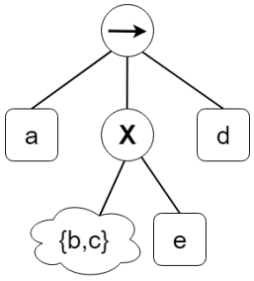
9. Inductive Miner – Find Cut

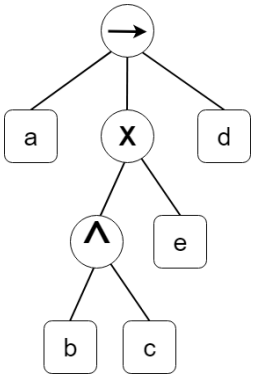
Check Base Case and Complete Process Tree



1. Exclusive-Choice Cut	Not Applicable
2. Sequence Cut	Not Applicable
3. Parallel Cut	Not Applicable
4. Redo-Loop Cut	Not Applicable





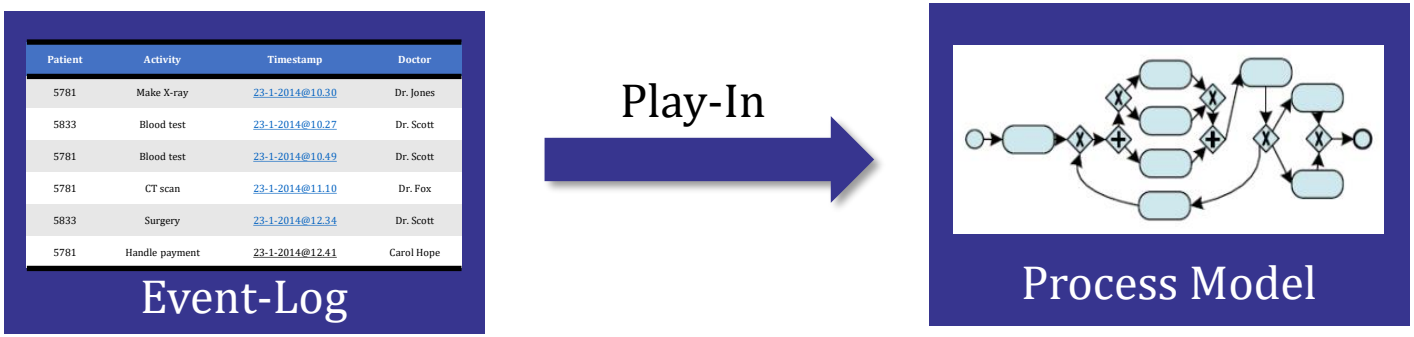


2.3 Conformance Techniques

1. Play-In & Play-Out & Replay

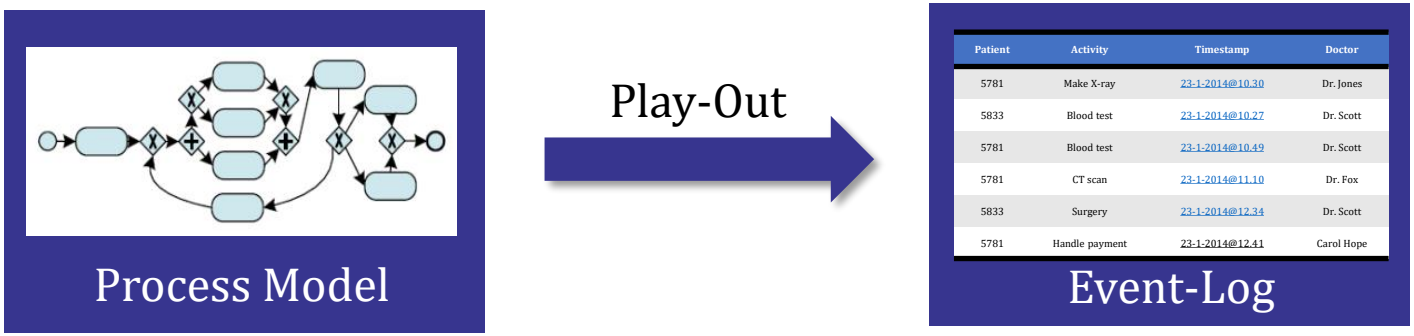
- Play-In

  : Derive Process Model from Event Log



- Play-Out

  : Make Event Log from Process Model

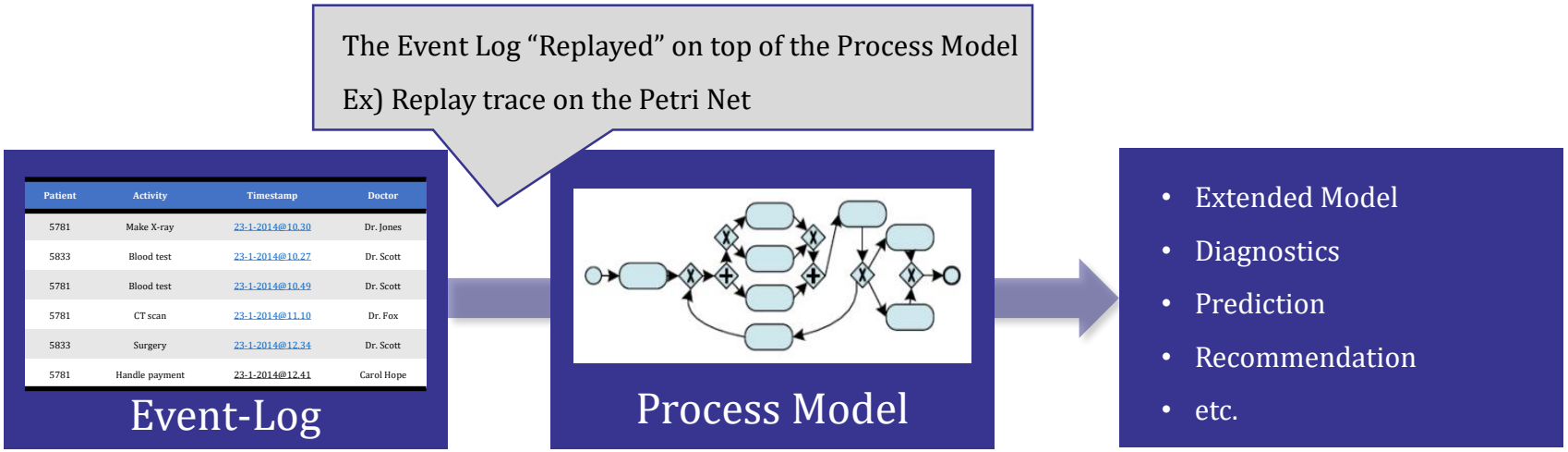


2.3 Conformance Techniques

1. Play-In & Play-Out & Replay

- Replay

  : Both Process Model & Event Log are input of Replay



- Various Purposes of "Event Log Replay"
  - Conformance Checking
  - Extending the model with frequencies and temporal information
  - Constructing Predictive Models
  - Operational Support

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

- Conformance Checking

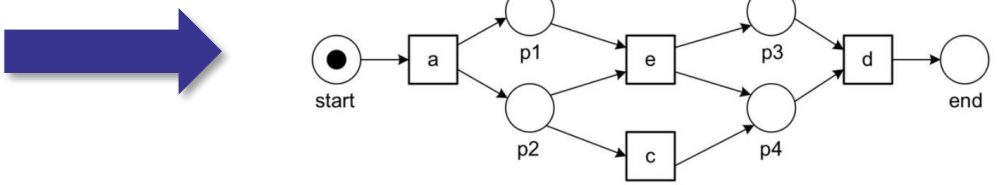
  : Discrepancies between the log and the model can be detected and quantified by replaying the log

- Footprint Matrix

  The most basic and simplest Conformance Checking Technique that starts with creating a Footprint Matrix

EX) With Alpha Algorithm

$$L_1 = [< a, b, c, d >^3, < a, c, b, d >^2, < a, e, d >]$$



도출한 프로세스 모델이 실제 데이터와 비교해봤을 때 **얼마나 잘 부합하는가?**

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

Step 1) Log-based Ordering Relations 파악 (from Event Log)

- Event Log로 부터 Activity간의 Relation을 파악함
- Direct Succesion(>) : 하나의 Trace에서 바로 뒤에 따라오는 directly followed Relation
- Casuality(->) : Activity x 와 y 에 대해 x > y는 성립하지만 그 역은 성립하지 않는 경우의 관계
- Parallel(||) : Activity x 와 y 에 대해 x > y 와 y > x 가 성립하는 관계
- Choice(#) : 두 Activity 사이에 아무런 관계가 없음

발생횟수

$$L_1 = [< a, b, c, d >^3, < a, c, b, d >^2, < a, e, d >]$$

- Trace
: 하나의 케이스 안에서 Activity가 어떤 순서로 일어났는 지만 나타내는 정보

$$>_{L_1} = \{(a,b),(a,c),(a,e),(b,c),(c,b),(b,d),(c,d),(e,d)\}$$

$$\rightarrow_{L_1} = \{(a,b),(a,c),(a,e),(b,d),(c,d),(e,d)\}$$

$$\#_{L_1} = \{(a,a),(a,d),(b,b),(b,e),(c,c),(c,e),(d,a),(d,d),(e,b),(e,c),(e,e)\}$$

$$\|_{L_1} = \{(b,c),(c,b)\}$$

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

Step 2) Making Footprint Matrix (from Event Log)

- Activity Relation

$$>_{L_1} = \{(a,b),(a,c),(a,e),(b,c),(c,b),(b,d),(c,d),(e,d)\}$$

$$\rightarrow_{L_1} = \{(a,b),(a,c),(a,e),(b,d),(c,d),(e,d)\}$$

$$\#_{L_1} = \{(a,a),(a,d),(b,b),(b,e),(c,c),(c,e),(d,a),(d,d),(e,b),(e,c),(e,e)\}$$

$$\|_{L_1} = \{(b,c),(c,b)\}$$

Activity Relation을 아래 Matrix의 해당 위치에 채운다

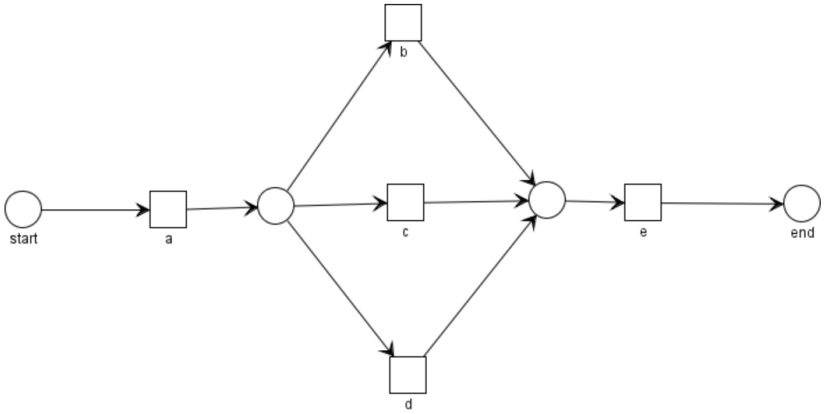
	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

Step 3) Log-based Ordering Relations 파악 (from Model)

- Event Log로 부터 도출한 것과 반대로 Process Model에서 부터 Footprint Matrix 도출



- Event Log from Process Model above

$$L = < a, b, e >, < a, c, e >, < a, d, e >$$

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

Step 4) Making Footprint Matrix (from Model)

- Event Log from Process Model

$$L = < a, b, e >, < a, c, e >, < a, d, e >$$

- Activity Relation

$> = \{(a, b), (b, e), (a, c), (c, e), (a, d), (d, e)\}$

$\rightarrow = \{(a, b), (b, e), (a, c), (c, e), (a, d), (d, e)\}$

$|| = \{\}$

$\# = \{(a, a), (a, e), (b, b), (b, c), (b, d), (c, b), (c, c), (c, d), (d, b), (d, c), (d, d), (e, e)\}$

- Footprint Matrix

	a	b	c	d	e
a	#	->	->	->	#
b	<-	#	#	#	->
c	<-	#	#	#	->
d	<-	#	#	#	->
e	#	<-	<-	<-	#

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

Step 5) Compare two Footprint Matrix and Calculate

- Footprint Matrix from Event Log

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

- Footprint Matrix from Process Model

	a	b	c	d	e
a	#	→	→	→	#
b	←	#	#	#	→
c	←	#	#	#	→
d	←	#	#	#	→
e	#	←	←	←	#

Activity Relations in Red Box are Different

$$Footprint\ Conformance = 1 - \frac{Mismatch\ Cell}{Match\ Cell}$$

$$Example's\ Footprint\ Conformance = 1 - \frac{16}{25} = 0.36$$

2.3 Conformance Techniques

2. Footprint Comparison Conformance Checking

Limitation of Footprint Matrix Conformance Checking

1. Event Log의 빈도에 대한 정보를 전혀 고려하지 않음
   - Activity 간의 Relation에만 초점을 맞추기 때문에 실제 Event Log에서 100번 일어난 Relation이 틀리든 1번 일어난 Relation이 틀리든 Conformance 값 자체에는 똑같은 영향을 미침
2. 적합도, 정확도, 일반화 수준을 고려하지 못함
   - 오로지 하나의 값에만 의존해 Conformance Checking을 시행함
3. Process Pattern 파악의 어려움이 존재함
   - Directly Follows Relation에만 의존하기 때문에, 두개의 Activity에 대한 정보만을 사용하며, 이에 따라 하나의 케이스 내에서 어떤 Pattern이 발생했는지에 대한 정보는 전혀 확인 불가능함

2.3 Conformance Techniques

3. Token Based Replay

<Token's Counters>

Produced Token : p

- 앞 Transition에서 만들어져 Place로 들어오는 Token

Consumed Token : c

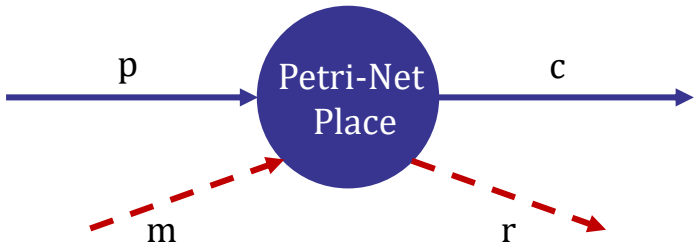
- 해당 Place에 있다가 다음 Transition에서 사용되는 Token

Missing Token : m

- 해당 Place에 없었는데 다음 Transition에 의해 사용되어야 하는 Token

Remaining Token : r

- 해당 Place에 Produced 되었지만 다음 Transition에서 사용되지 않아 남아있는 Token

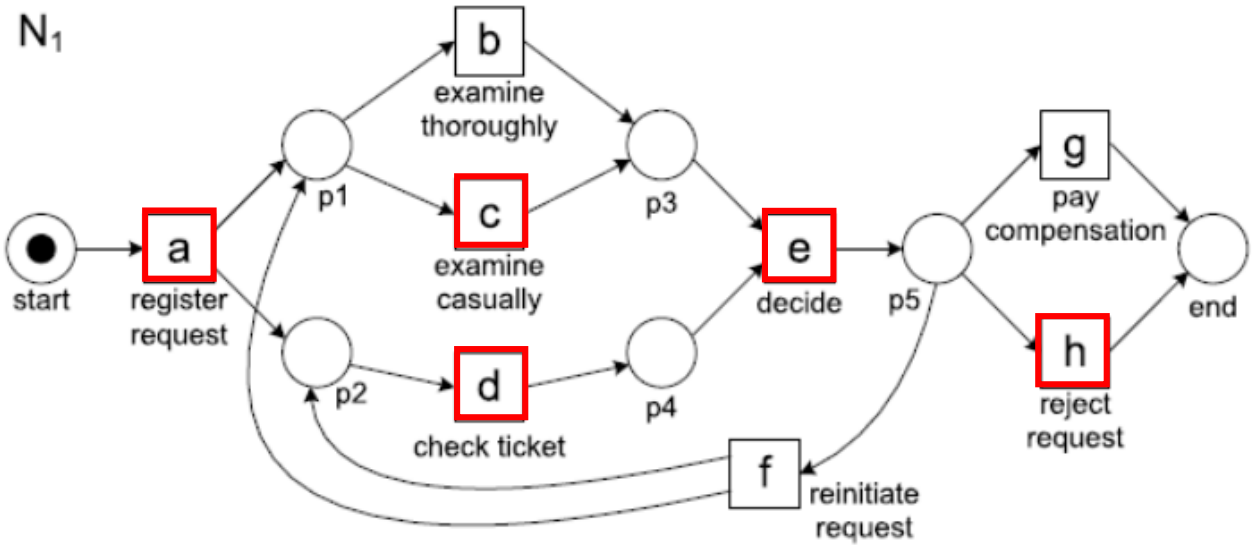


2.3 Conformance Techniques

3. Token Based Replay

Step 1) Petri-Net & Trace for Example

예시 Petri-Net : N_1



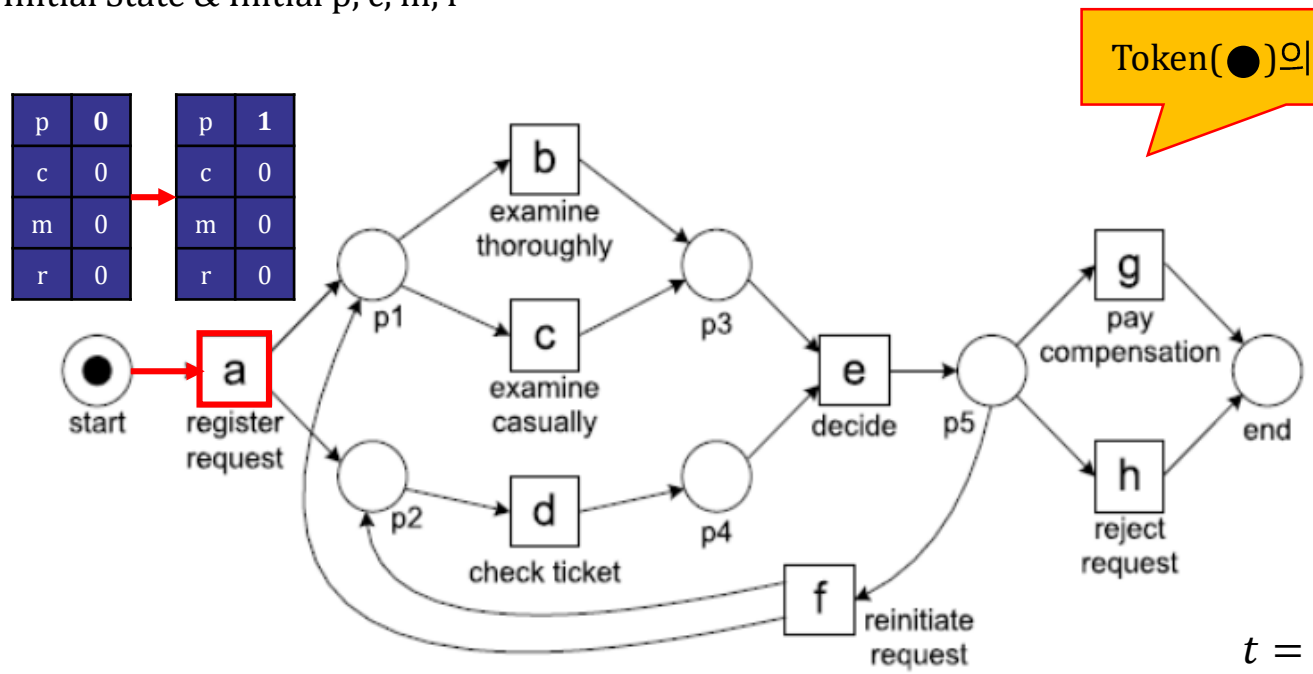
예시 Trace

$$t = < a, c, d, e, h >$$

2.3 Conformance Techniques

3. Token Based Replay

Step 2) Initial State & Initial p, c, m, r



예시 Trace

$t = < a, c, d, e, h >$

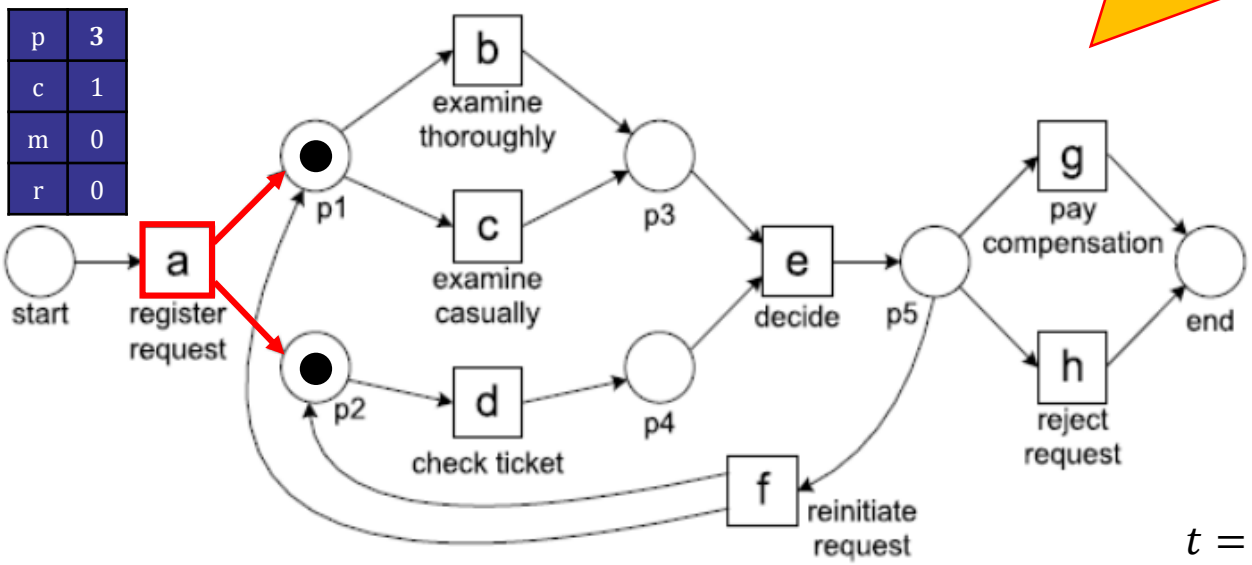
- Token Replay를 시작하기 전 최초 상태의 Petri-Net에서 produced, Consumed, Missing, Remaining은 모두 0이 됨.
- Token Replay시작과 함께 Start Place에 토큰이 하나 만들어지기 때문에 p = 1이 됨.

2.3 Conformance Techniques

3. Token Based Replay

Step 3) Activity a Occurrence

Token(●)의 움직임에 집중



예시 Trace

$t = < a, c, d, e, h >$

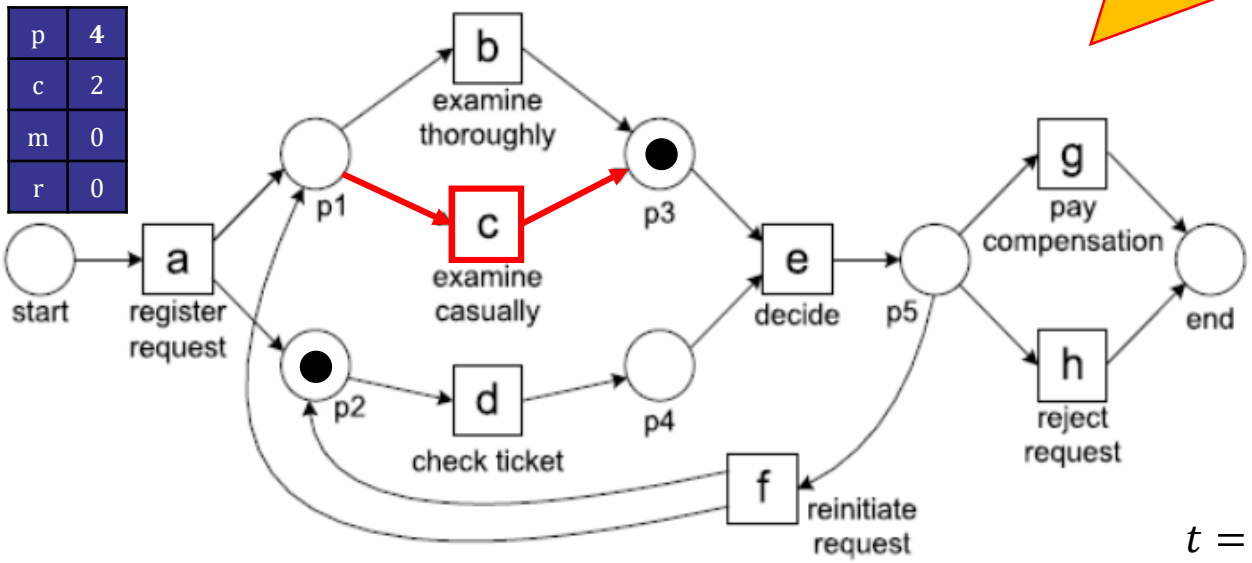
- Activity a가 발생하면 Start Place에 있던 Token은 Consumed → c + 1
- Activity a가 발생하면 p1과 p2 Place에 Token이 Produced → p + 2

2.3 Conformance Techniques

3. Token Based Replay

Step 4) Activity c Occurrence

Token(●)의 움직임에 집중



예시 Trace

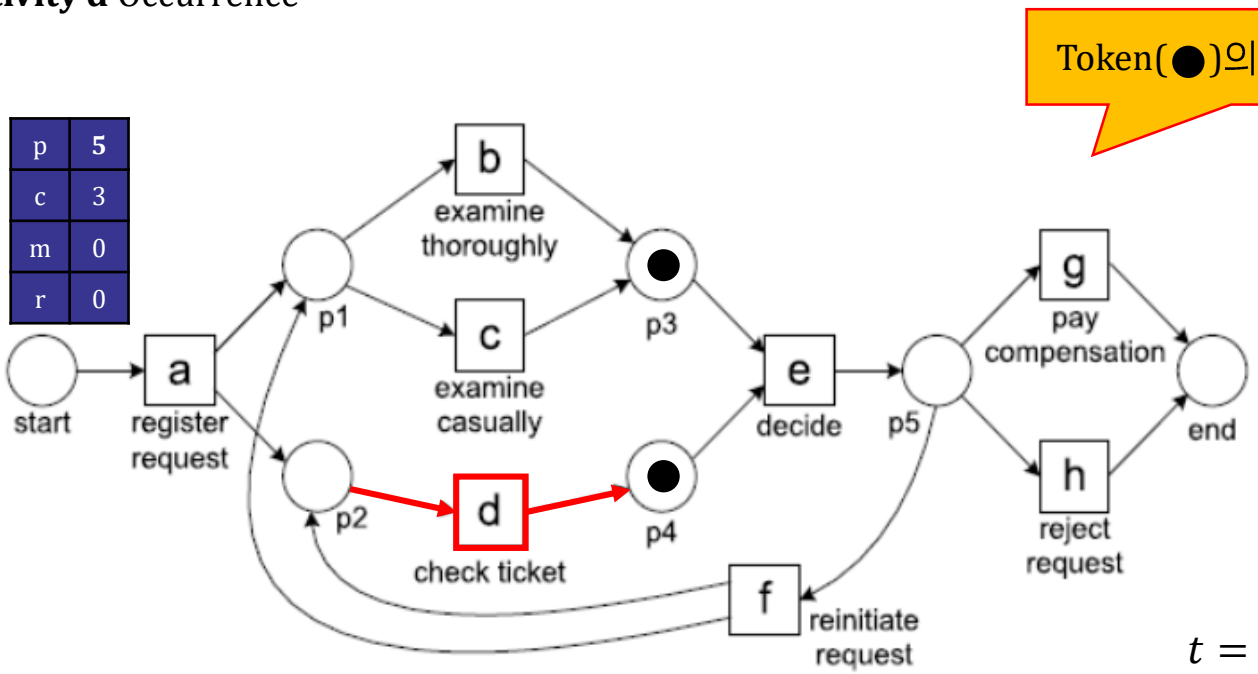
$t = < a, c, d, e, h >$

- Activity c가 발생하면 p1 Place에 있던 Token은 Consumed → c + 1
- Activity c가 발생하면 p3 Place에 Token이 Produced → p + 1

2.3 Conformance Techniques

3. Token Based Replay

Step 5) Activity d Occurrence



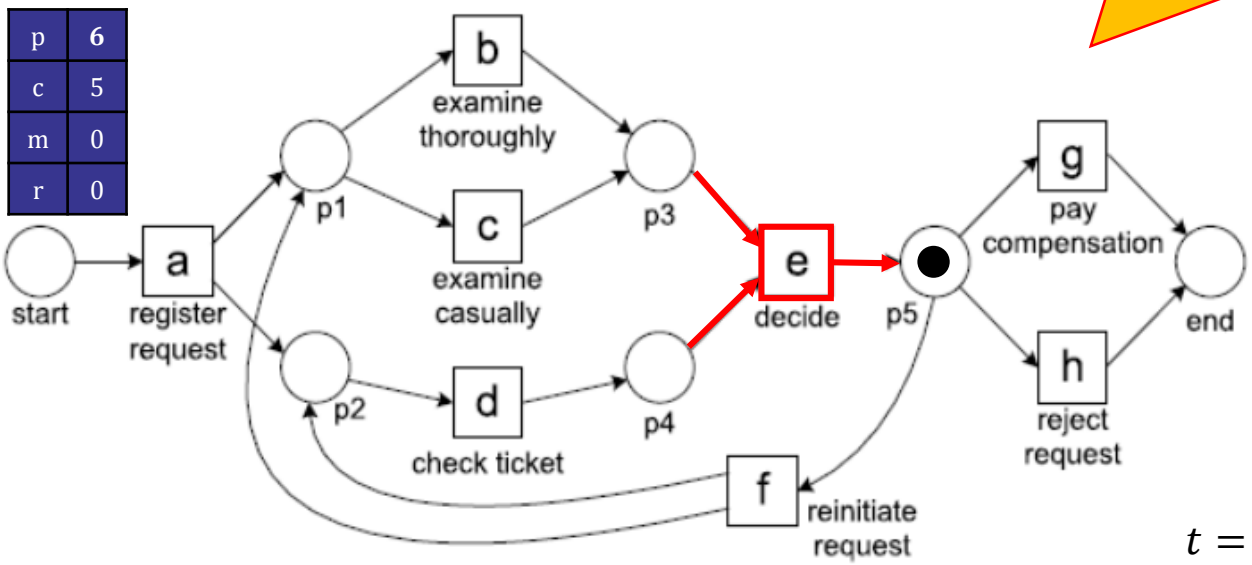
- Activity d가 발생하면 p2 Place에 있던 Token은 Consumed → c + 1
- Activity d가 발생하면 p4 Place에 Token이 Produced → p + 1

2.3 Conformance Techniques

3. Token Based Replay

Step 6) Activity e Occurrence

Token(●)의 움직임에 집중



예시 Trace

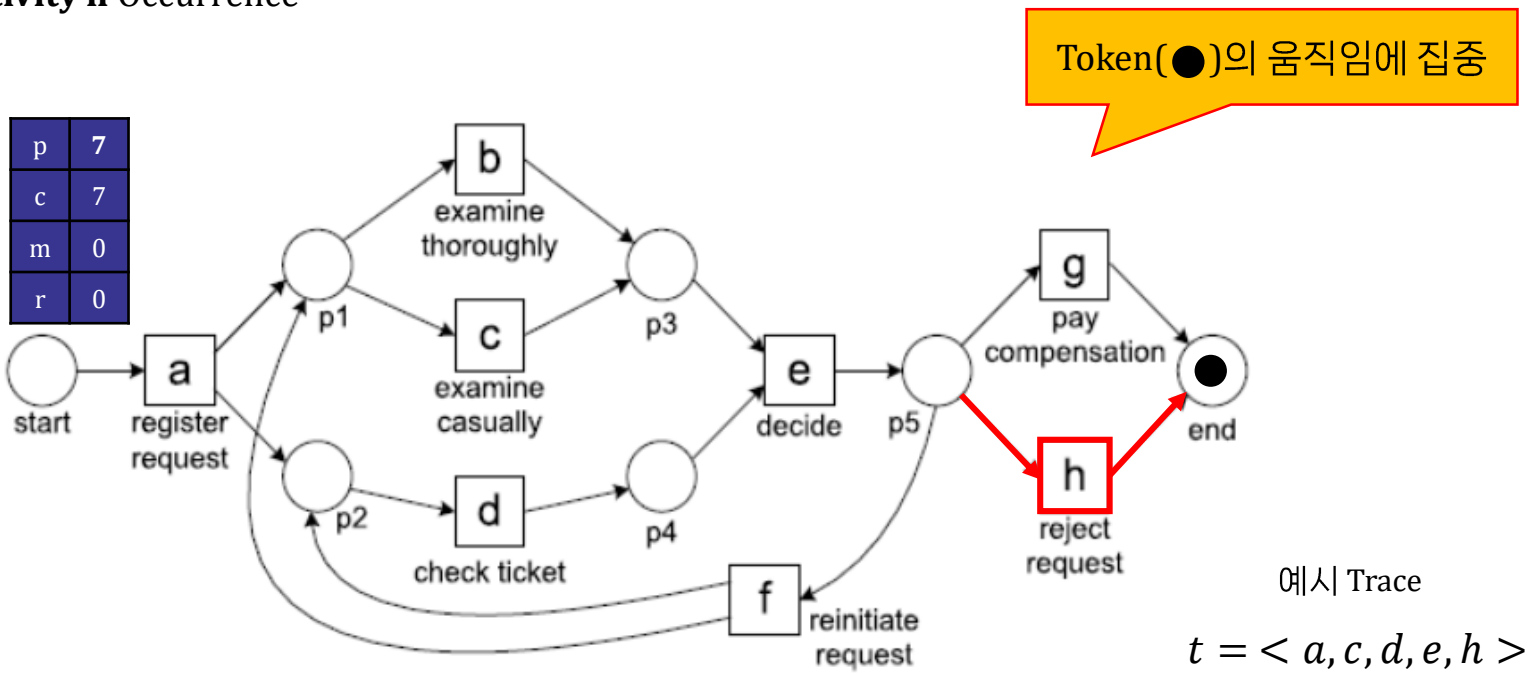
$t = < a, c, d, e, h >$

- Activity e가 발생하면 p3 와 p4 Place에 있던 Token은 Consumed → c + 2
- Activity e가 발생하면 p5 Place에 Token이 Produced → p + 1

2.3 Conformance Techniques

3. Token Based Replay

Step 6) Activity h Occurrence



예시 Trace

$t = < a, c, d, e, h >$

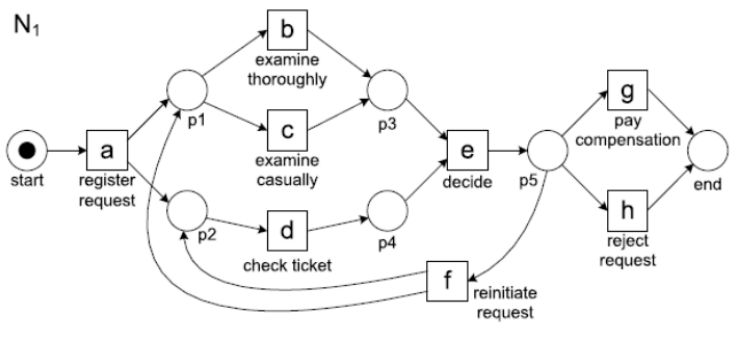
- Activity h가 발생하면 p5 Place에 있던 Token은 Consumed → c + 1
- Activity h가 발생하면 End Place에 Token이 Produced → p + 1
- 여기서 end는 마지막 Place 이기 때문에 token이 Consumed → c + 1

2.3 Conformance Techniques

3. Token Based Replay

Step 7) Conformance Checking

- 예시 Petri-Net



- 예시 Trace

$$t = < a, c, d, e, h >$$

- 최종 Token Counters

Produced	7
Consumed	7
Missing	0
Remaining	0

$$fitness = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

- Consumed된 것 중에 Missing이 아닌 정상적인 Token
- Produced된 것 중에 Remained되지 않고 정상적인 Token

→ Missing과 Remaining의 숫자가 적을 수록 Fitness는 높아짐

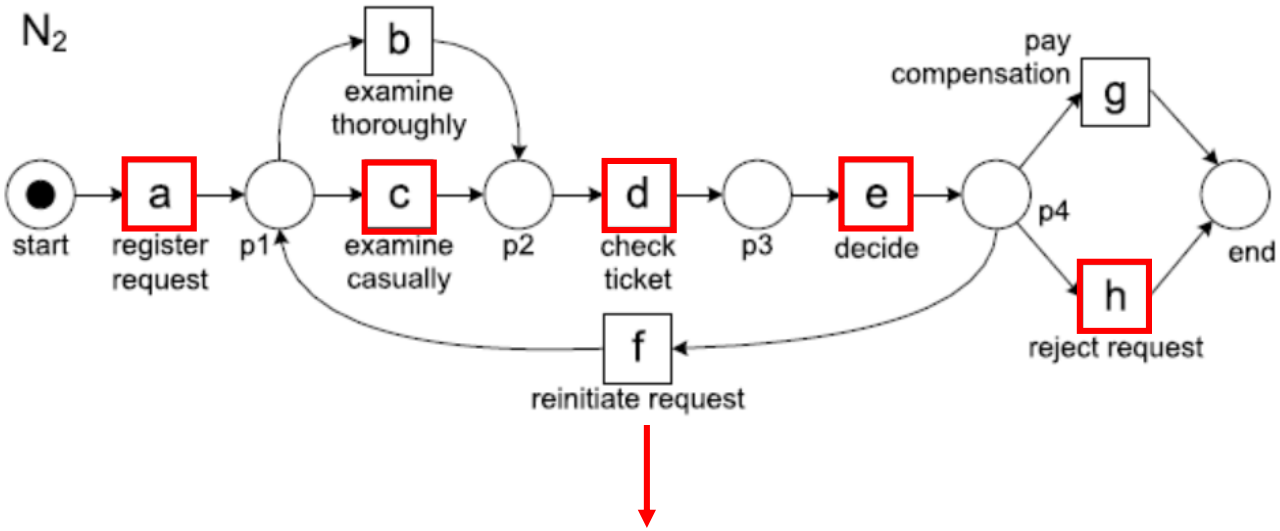
$$fitness = \frac{1}{2}\left(1 - \frac{0}{7}\right) + \frac{1}{2}\left(1 - \frac{0}{7}\right) = 1$$

2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 1) Petri-Net & Trace for Example

예시 Petri-Net : N_2



예시 Trace

$$t = < a, d, c, e, h >$$

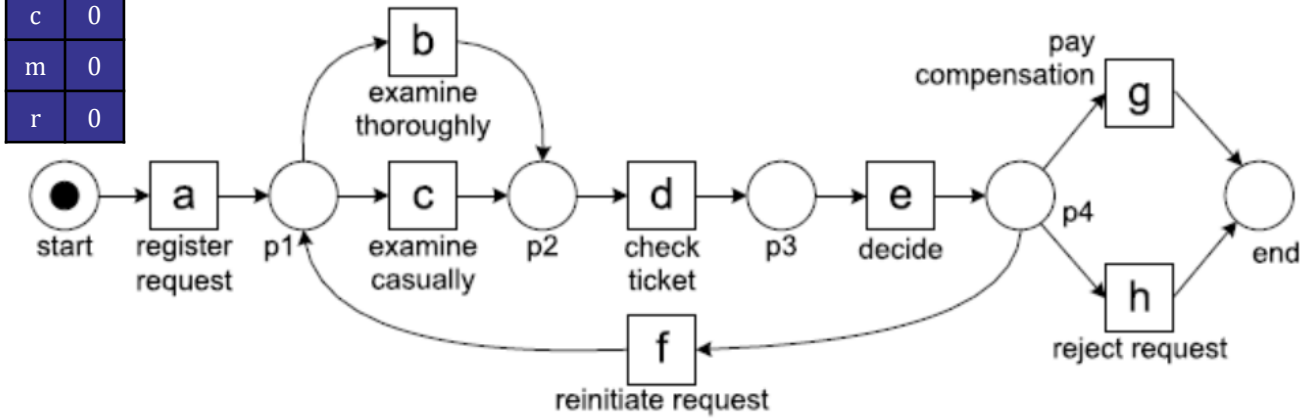
2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 2) Initial State & Initial p,c,m,r

$t = < a, d, c, e, h >$

p	1
c	0
m	0
r	0



Token(●)의 움직임에 집중

- 최초 시작에는 동일하게 p, c, m, r이 0으로 시작
- Start Place에 Token이 Produced 되며 Token Replay를 시작 → p + 1

예시 Trace

$t = < a, d, c, e, h >$

2.3 Conformance Techniques

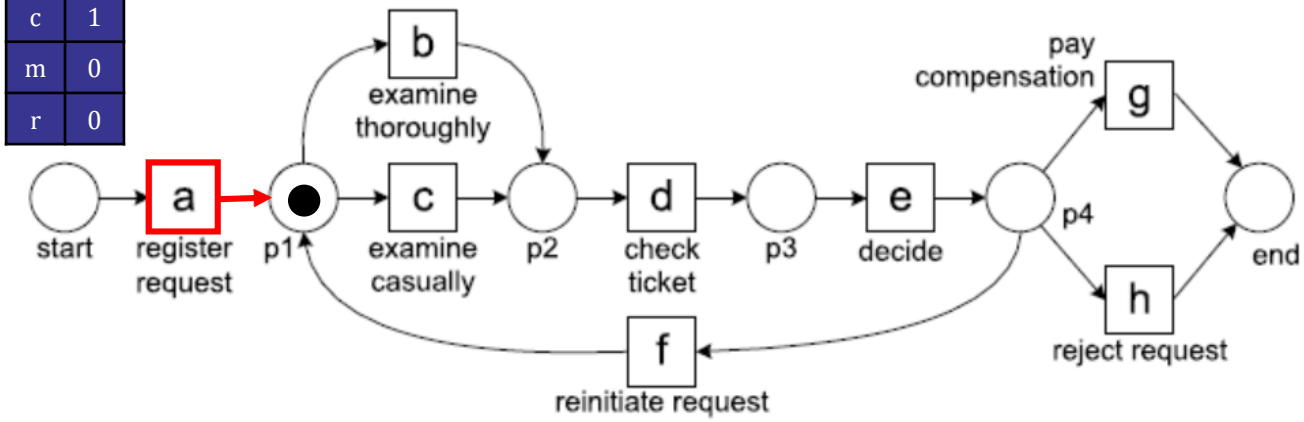
3. Token Based Replay – Advanced

Step 3) Activity a Occurrence

$t = < a, d, c, e, h >$

Token(●)의 움직임에 집중

p	2
c	1
m	0
r	0



- Activity a가 발생하면 start Place에 있던 Token은 Consumed → c + 1
- Activity a가 발생하면 p1 Place에 Token이 Produced → p + 1

예시 Trace

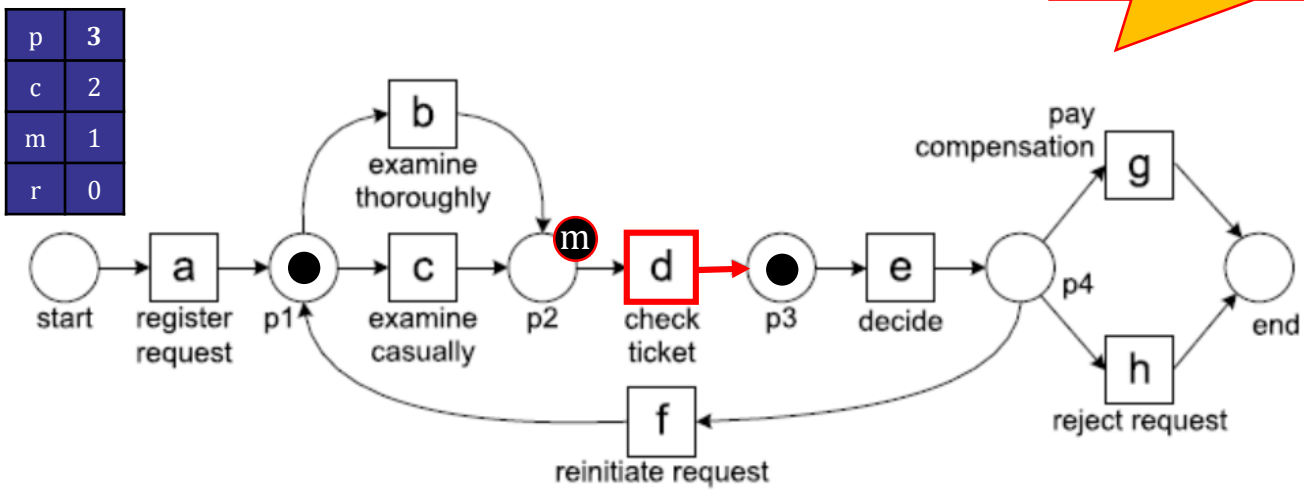
$t = < a, d, c, e, h >$

2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 4) Activity d Occurrence

$t = < a, d, c, e, h >$



Token(●)의 움직임에 집중

예시 Trace

$t = < a, d, c, e, h >$

- Activity d가 발생하려면 p2 Place에 Token이 필요하지만 Missing됨 → m + 1 , Missing Token Produced (ⓜ)
- Missing Token을 Consumed하고 p3 Place에 Token Produced → c + 1 , p + 1

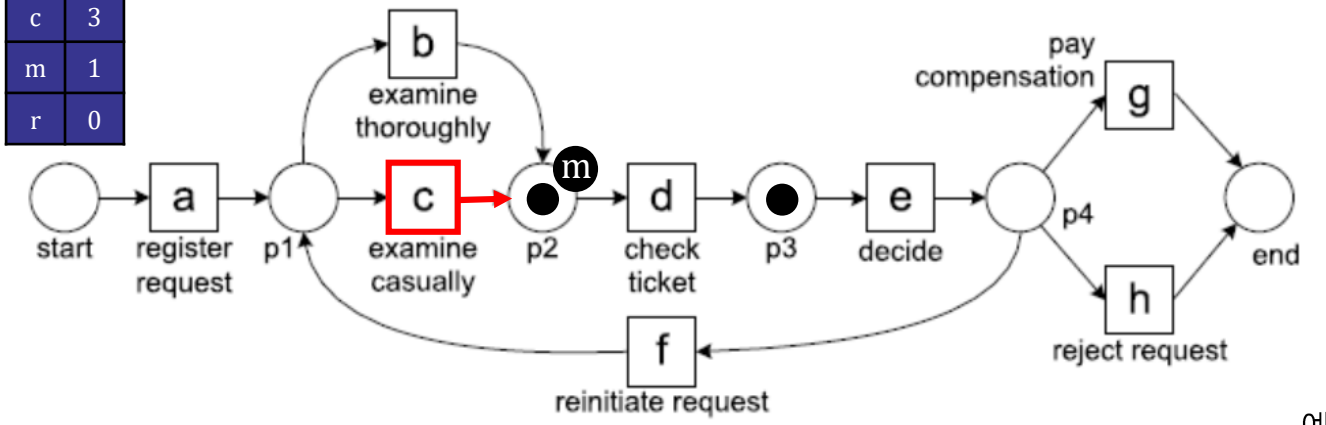
2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 5) Activity c Occurrence

$t = < a, d, c, e, h >$

p	4
c	3
m	1
r	0



Token(●)의 움직임에 집중

예시 Trace

$t = < a, d, c, e, h >$

- Activity c가 발생하면 p1 Place에 Token이 Consumed→ c + 1
- Activity c가 발생하면 p2 Place에 Token이 Produced → p + 1

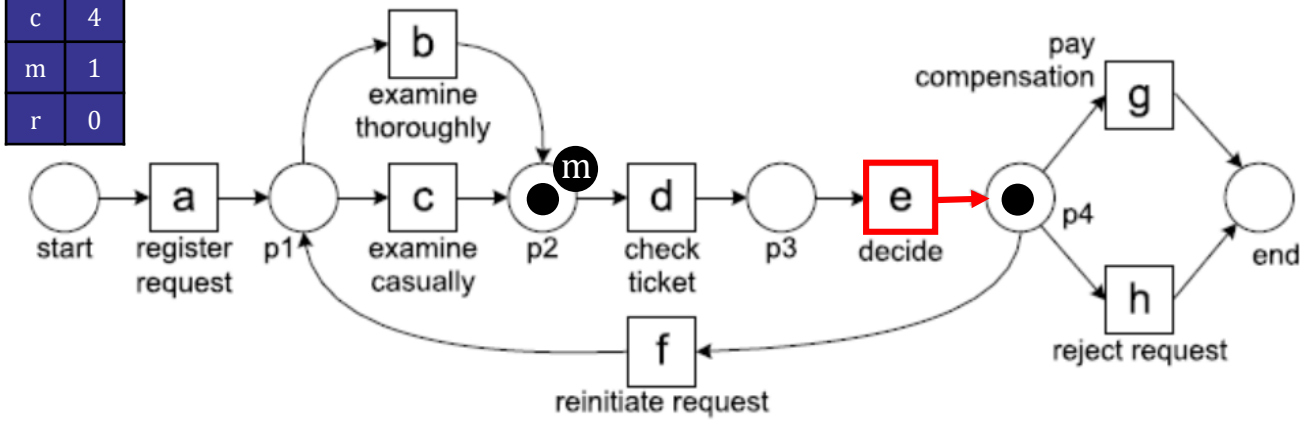
2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 6) Activity e Occurrence

$t = < a, d, c, e, h >$

p	5
c	4
m	1
r	0



Token(●)의 움직임에 집중

예시 Trace

$t = < a, d, c, e, h >$

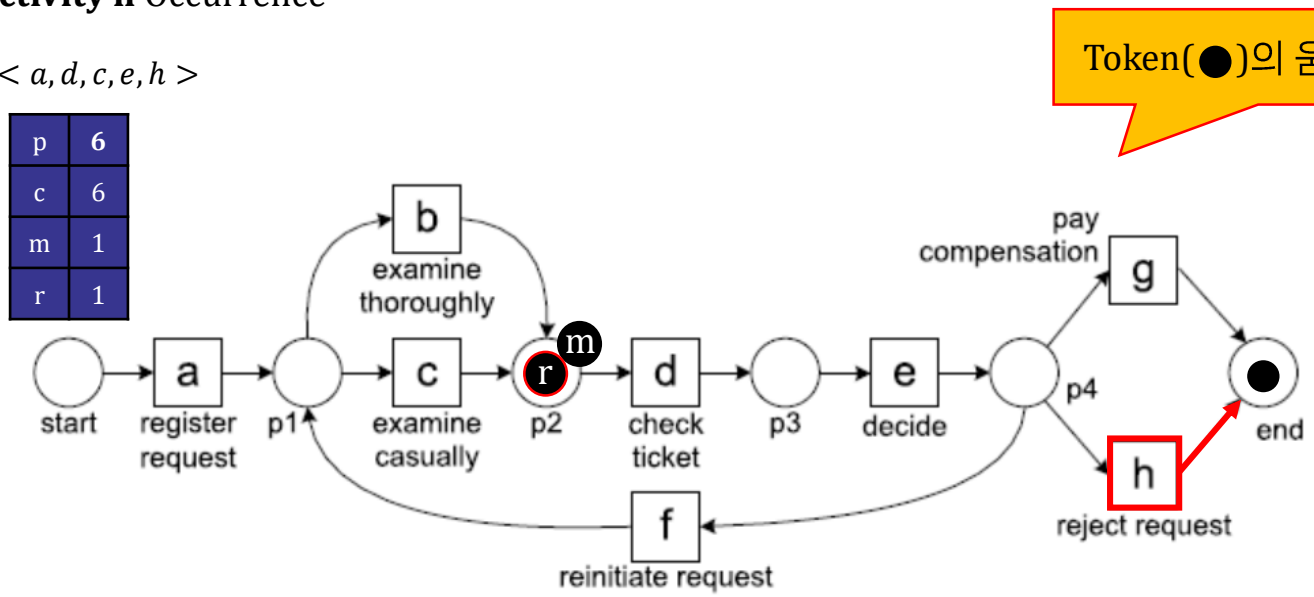
- Activity e가 발생하면 p3 Place에 Token이 Consumed→ c + 1
- Activity e가 발생하면 p4 Place에 Token이 Produced → p + 1

2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 7) Activity h Occurrence

$t = < a, d, c, e, h >$



Token(●)의 움직임에 집중

예시 Trace

$t = < a, d, c, e, h >$

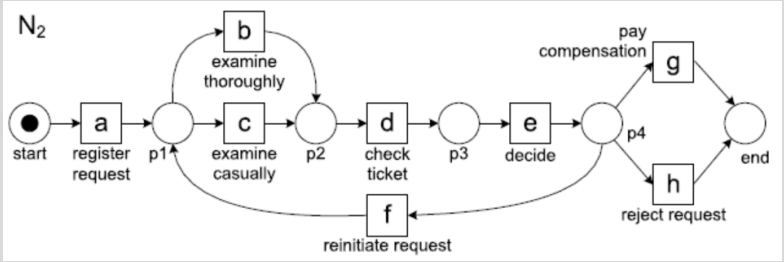
- Activity h가 발생하면 p4 Place에 Token이 Consumed, end Place에 Token Produced→ c + 1, p + 1
- End Place는 마지막 Place이므로 Token이 Consumed → c + 1
- P2 Place를 보게되면, trace가 종료되었음에도 p2에 Token이 Remaining → r + 1

2.3 Conformance Techniques

3. Token Based Replay – Advanced

Step 8) Conformance Checking

- 예시 Petri-Net



예시 Trace

$t = < a, d, c, e, h >$

- 최종 Token Counters

Produced	6
Consumed	6
Missing	1
Remaining	1

$$fitness = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$$

- Consumed된 것 중에 Missing이 아닌 정상적인 Token
- Produced된 것 중에 Remained되지 않고 정상적인 Token

→ Missing과 Remaining의 숫자가 적을 수록 Fitness는 높아짐

$$fitness = \frac{1}{2}\left(1 - \frac{1}{6}\right) + \frac{1}{2}\left(1 - \frac{1}{6}\right) = \frac{5}{6}$$

2.3 Conformance Techniques

3. Token Based Replay – Whole Event Log

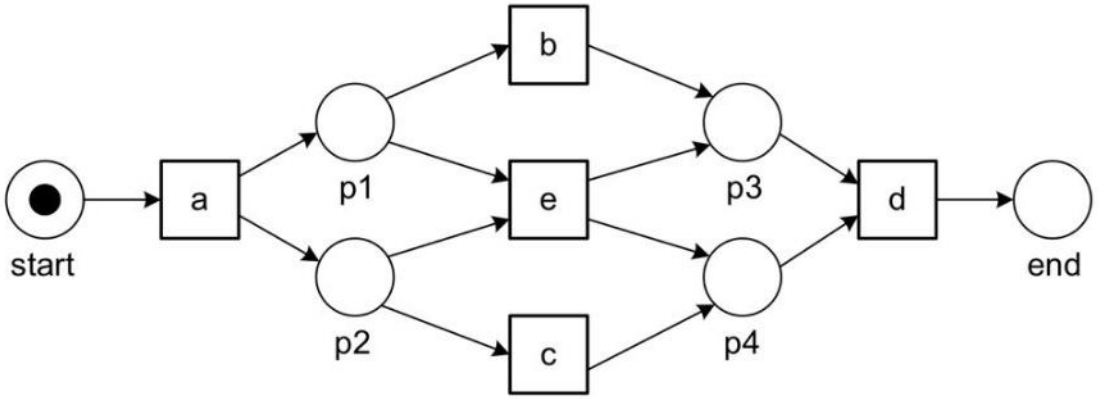
- How to Calculate Fitness for whole Event Log?

→ 앞선 두 예시 모두 각 Trace에 대해서만 p, c, m, r을 계산함.

→ 많은 trace를 포함하고 있는 Event Log에 대해서는 Fitness를 어떻게 계산할까?

→ 아래의 예시 Event Log와 Petri-Net을 통해 설명

$$Event\ Log:\ L = < a,b,c,d >^{10}, < a,c,b,d >^{10}, < a,e,d >^{10},$$
$$< a,b,d >^{2}, < a,c,d >^{1}, < a,d >^{1}, < a,b,b,d >^{1}$$



2.3 Conformance Techniques

3. Token Based Replay – Whole Event Log

- How to Calculate Fitness for whole Event Log?

→ 아래의 공식을 통해 여러 Trace를 포함하고 있는 Event Log에 대해 Fitness를 측정 가능.

$$fitness(L, N) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}}\right)$$

→ 위의 Fitness공식은 복잡해 보이지만 각 Trace에 대해서 p, c, m, r을 계산한 후 각 Trace의 반복 횟수 만큼 곱해서 모두 더하면 됨.

$$Event\ Log: L = < a,b,c,d >^{10}, < a,c,b,d >^{10}, < a,e,d >^{10}, < a,b,d >^{2}, < a,c,d >^{1}, < a,d >^{1}, < a,b,b,d >^{1}$$

Trace	Frequency	p	r	c	m
$< a,b,c,d >$	10	6 * 10	0 * 10	6 * 10	0 * 10
$< a,c,b,d >$	10	6 * 10	0 * 10	6 * 10	0 * 10
$< a,e,d >$	10	6 * 10	0 * 10	6 * 10	0 * 10
$< a,b,d >$	2	5 * 2	1 * 2	5 * 2	1 * 2
$< a,c,d >$	1	5 * 1	1 * 1	5 * 1	1 * 1
$< a,d >$	1	4 * 1	2 * 1	4 * 1	2 * 1
$< a,b,b,d >$	1	6 * 1	2 * 1	6 * 1	2 * 1
Total		205	7	205	7

위의 Fitness 공식에 대입

$$Fitness = \frac{198}{205} = 0.9658$$

2.3 Conformance Techniques

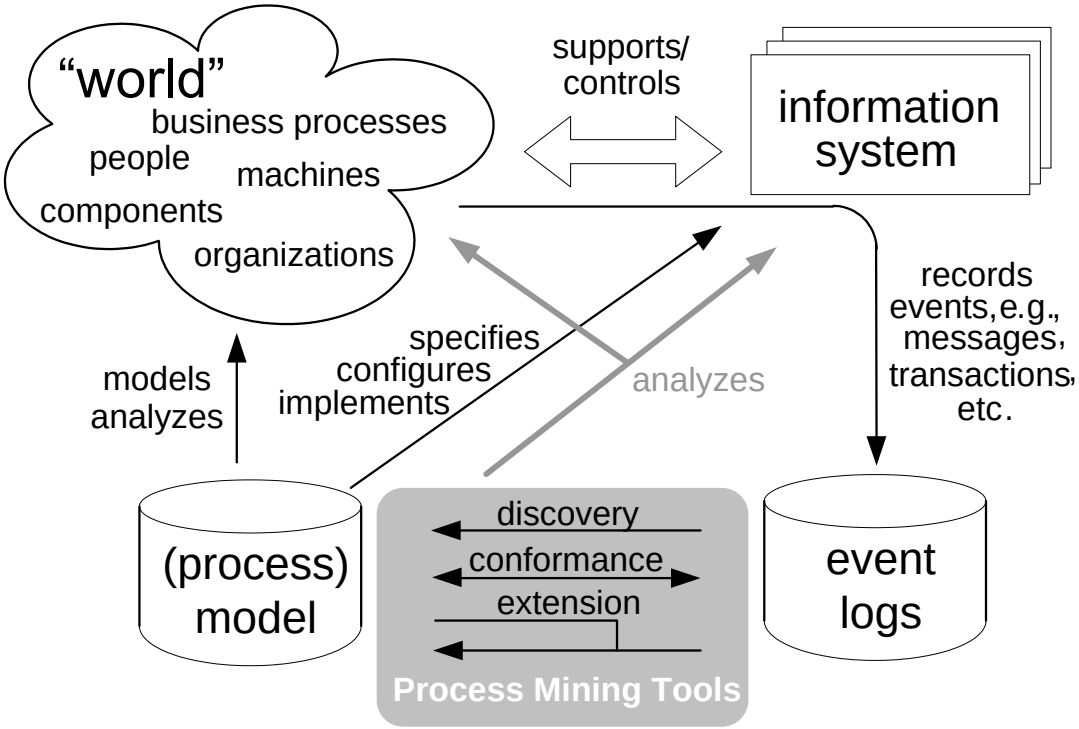
3. Token Based Replay

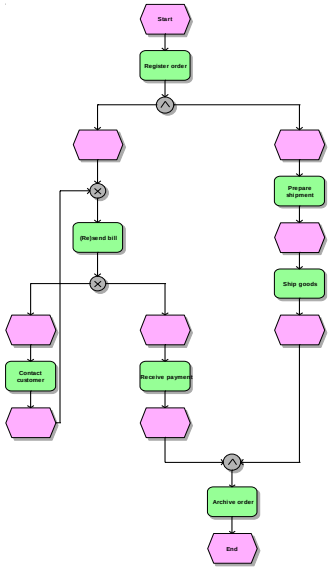
Limitation of Token Based Replay Conformance Checking

1. 모든 Transition이 Unique하다고 가정함.
   - 동일한 이름을 가진 두개의 다른 Transition이 존재한다면 어떤 transition을 fire시켜야 하는지, p,c,m,r을 어떻게 계산해야 할지 알 수 없음
2. 실제 대비 좋은 결과를 보이는 경우가 종종 있음.
   - 하나의 Place에서 여러 Transition으로 반복해서 이동하는 구조를 가진 모델에서 해당 Transition들의 순서에 상관없이 좋은 Fitness 결과를 보임. (일반화 수준, 간결함의 정도, 정확도의 고려 필요)
3. Local Decision 이 존재하는 모델에서 실제 대비 좋은 결과를 보임.
   - 앞에 선행된 Activity가 뒤의 Activity의 행동을 결정하는 Local Decision을 포함하는 모델의 경우, 좋은 Fitness를 보이지만 이것은 잘못된 Fitness임.

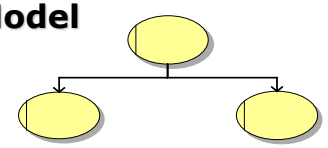
2.4 Extension

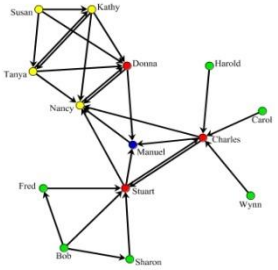
1. Organization Miner





Organizational Model





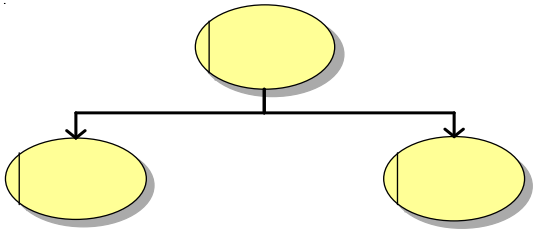
Social Network

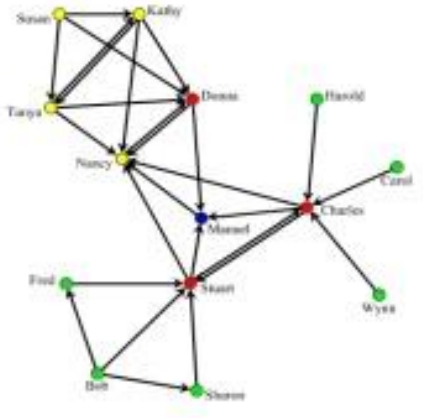
2.4 Extension

1. Organization Miner

Organizational Mining Algorithms

- Aid in understanding and improbing social and organizational structures
- Two types of algorithms
  - Organizational Model
    - Mining of roles and teams in organizations
    - Plug-in : Organizational Miner
  - Social Networks
    - Discovery of relationships among originators
    - Plug-ins : Social Network Miner and Analyze Social Network

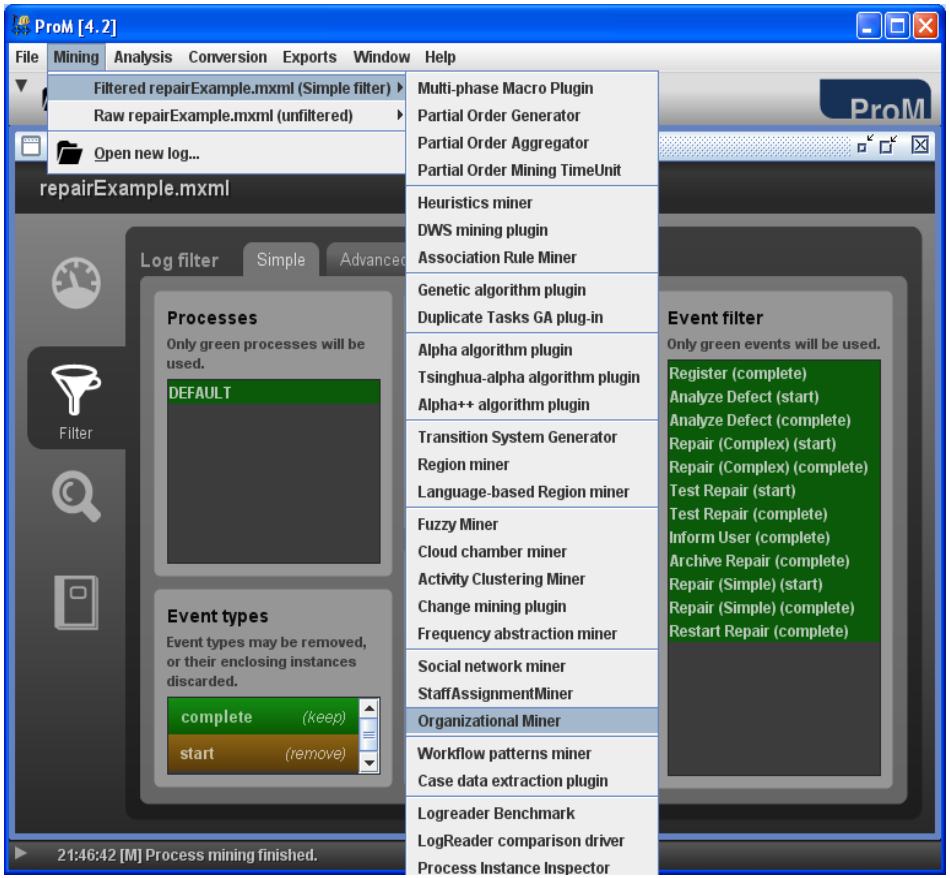




2.4 Extension

1. Organization Miner

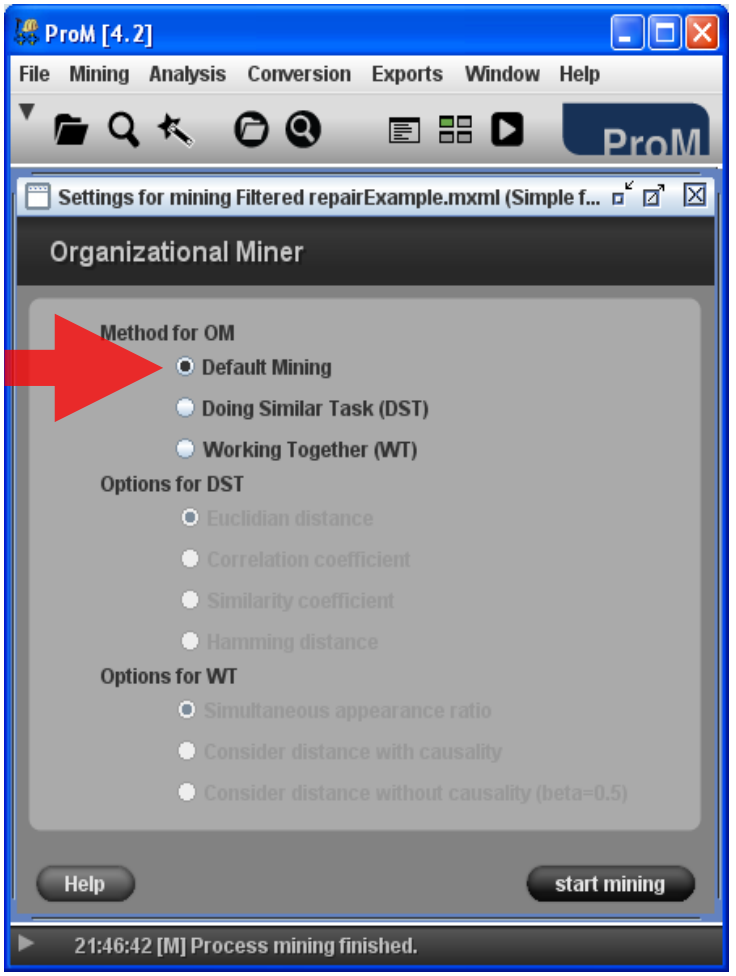
- Main idea : Which orginators are executing which tasks
- Methods to mine roles
  - Default mining
  - Doing Similar Tasks
- Methods to mine teams
  - Working together



2.4 Extension

1. Organization Miner

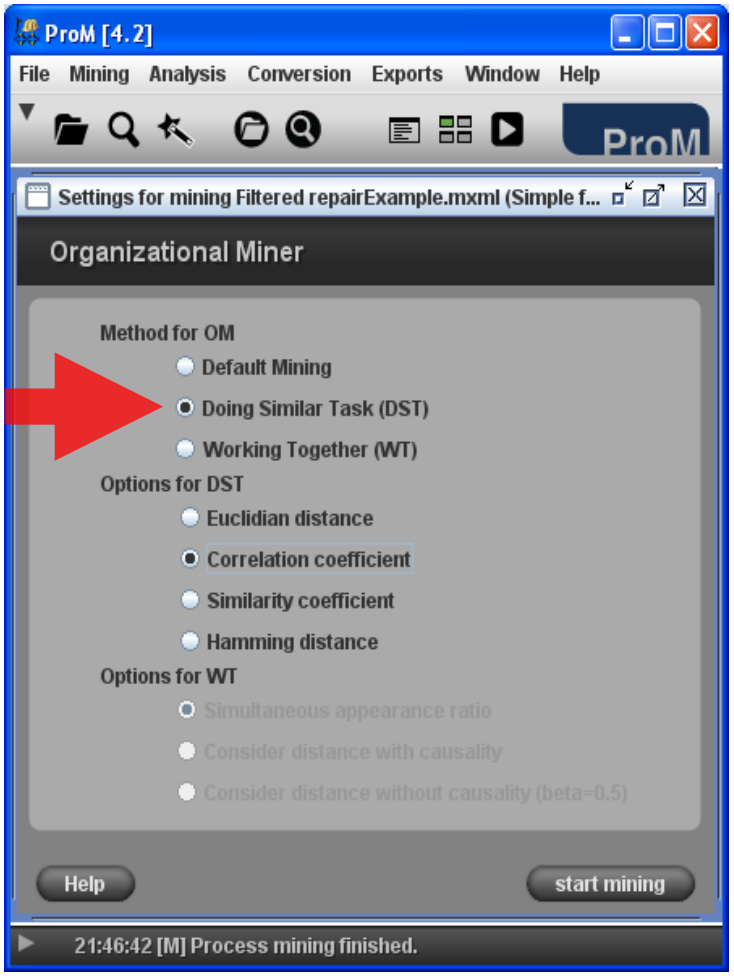
- Main idea : Which orginators are executing which tasks
- Methods to mine roles
  - **Default mining**
  - Doing Similar Tasks
- Methods to mine teams
  - Working together



2.4 Extension

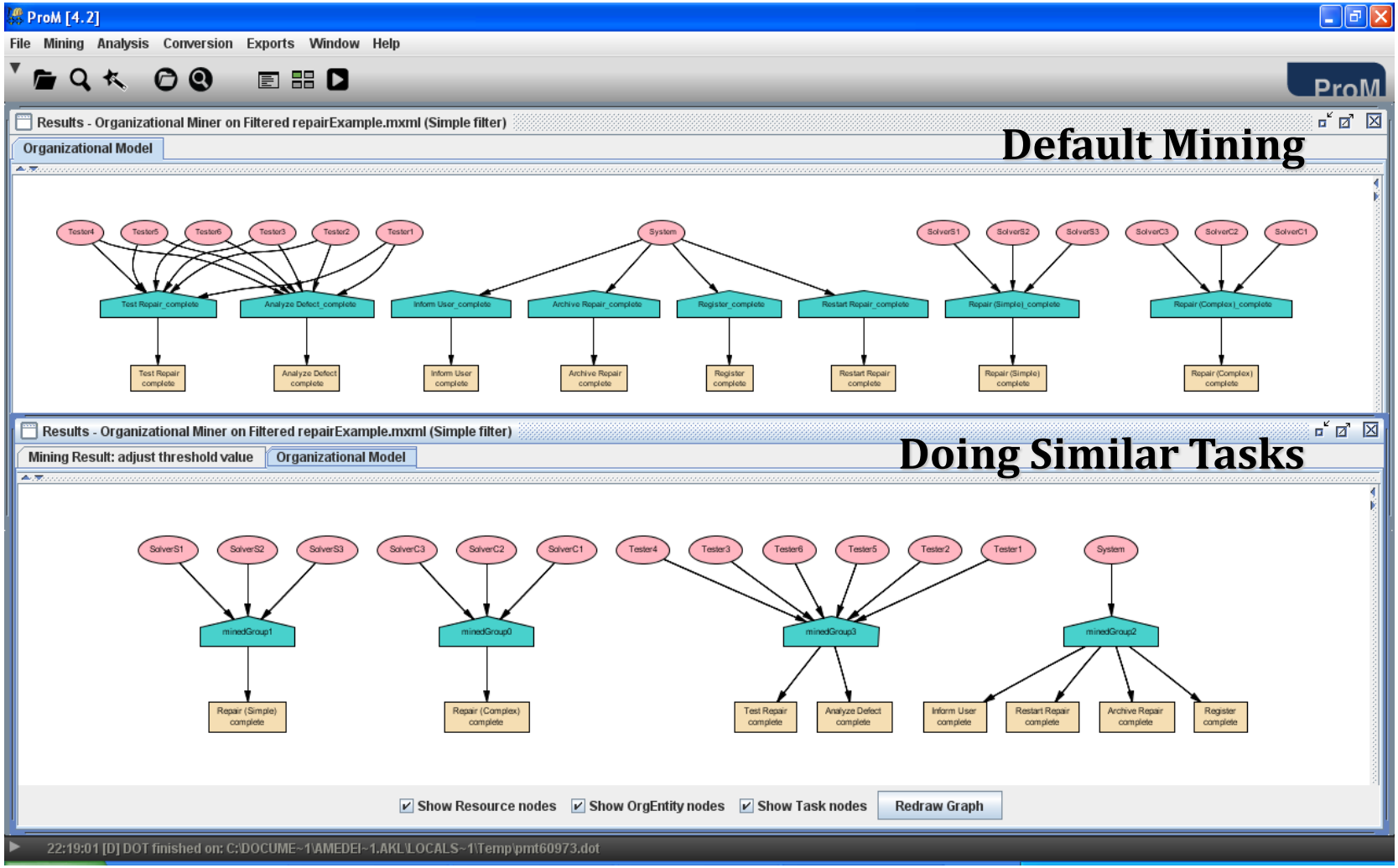
1. Organization Miner

- Main idea : Which orginators are executing which tasks
- Methods to mine roles
  - Default mining
  - **Doing Similar Tasks**
- Methods to mine teams
  - Working together



2.4 Extension

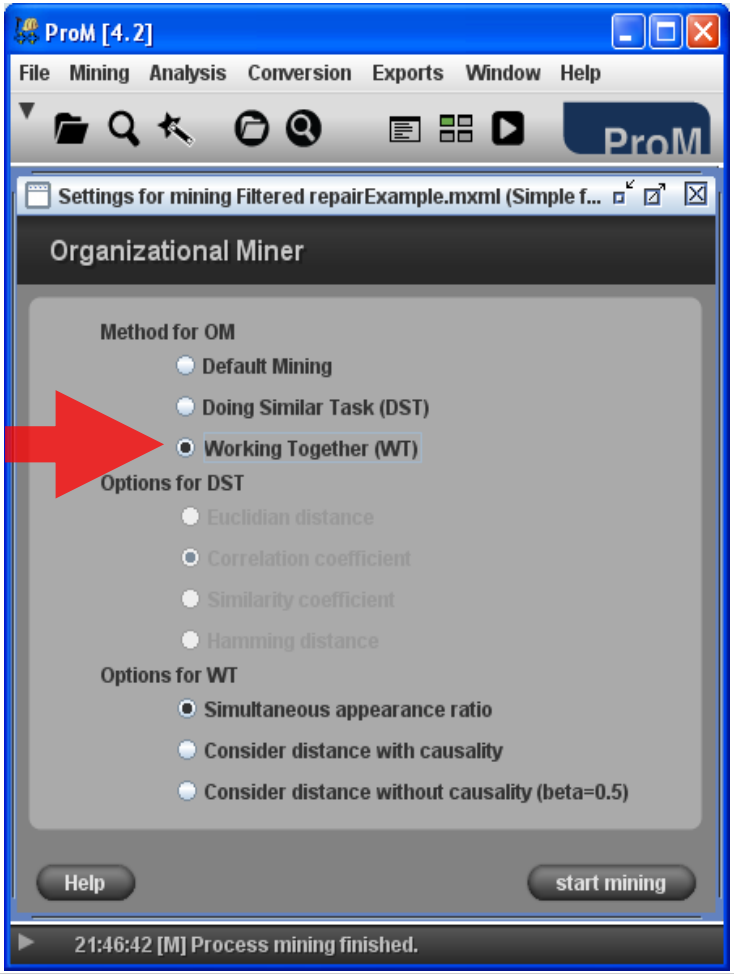
1. Organization Miner



2.4 Extension

1. Organization Miner

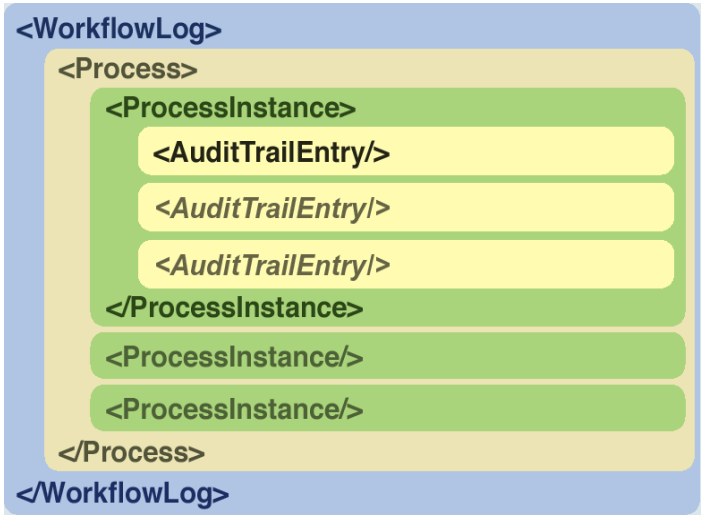
- Main idea : Which orginators are executing which tasks
- Methods to mine roles
  - Default mining
  - Doing Similar Tasks
- Methods to mine teams
  - **Working together**



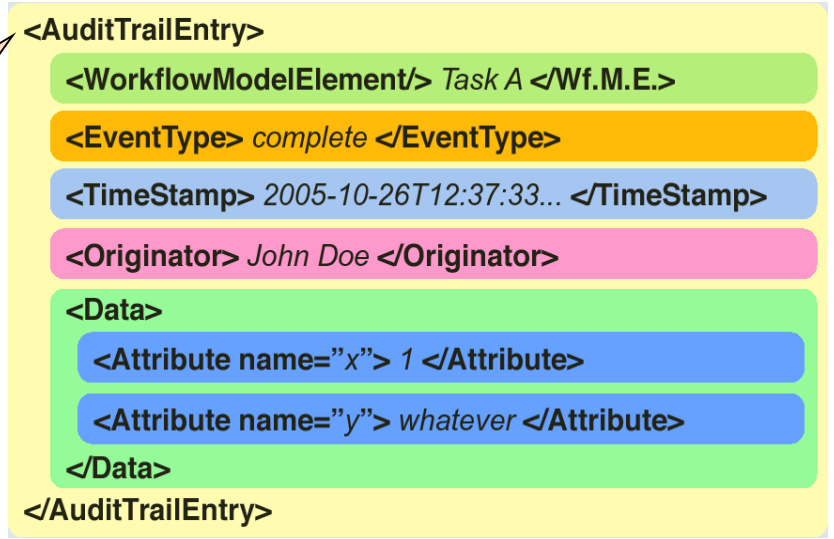
2.4 Extension

1. Organization Miner

Why is the notion of process instances necessary to mine teams but unnecessary to mine roles?



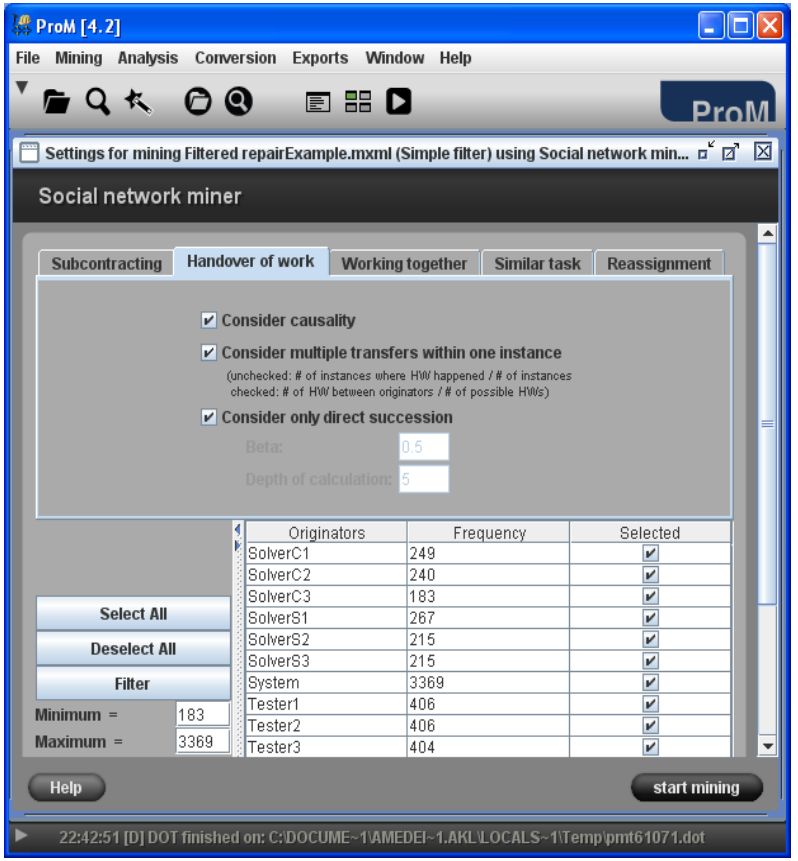
Could you think of an algorithm to detect specialists/generalists for a given process? What is the main idea behind?



2.4 Extension

2. Social Network Miner

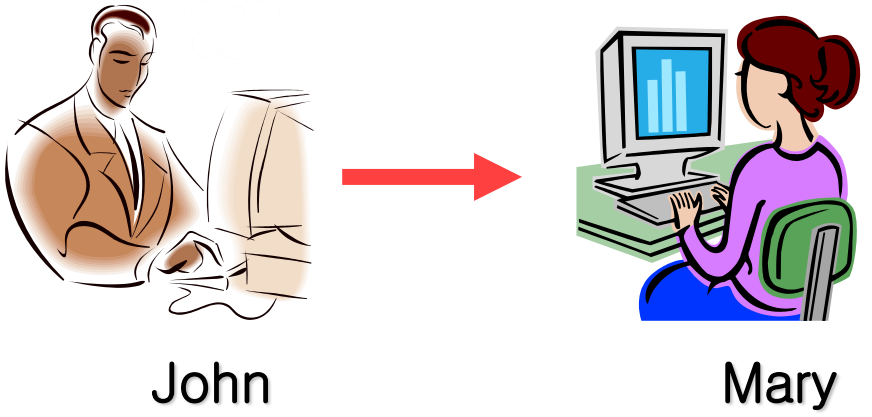
- Aim: Monitor how individual process instances are routed between originators
- Metrics
  - Handover of work
  - Subcontracting
  - Reassignment
  - Working together
  - Similar task



2.4 Extension

2. Social Network Miner

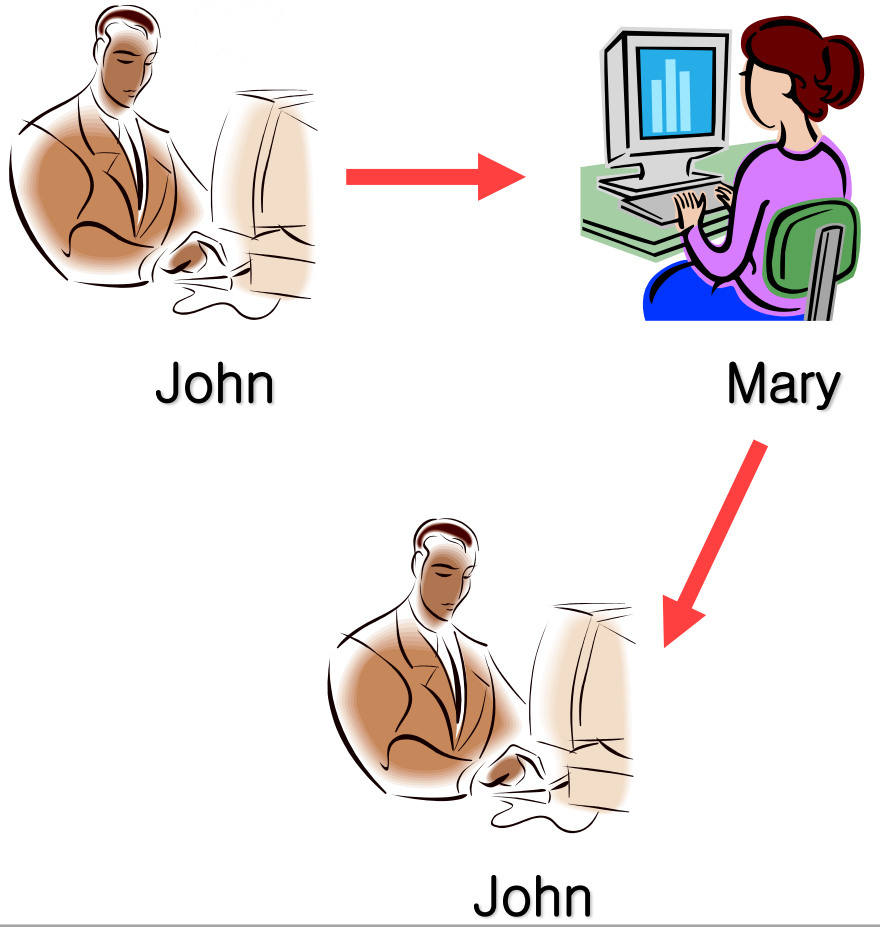
- Aim: Monitor how individual process instances are routed between originators
- Metrics
  - **Handover of work**
  - Subcontracting
  - Reassignment
  - Working together
  - Similar task



2.4 Extension

2. Social Network Miner

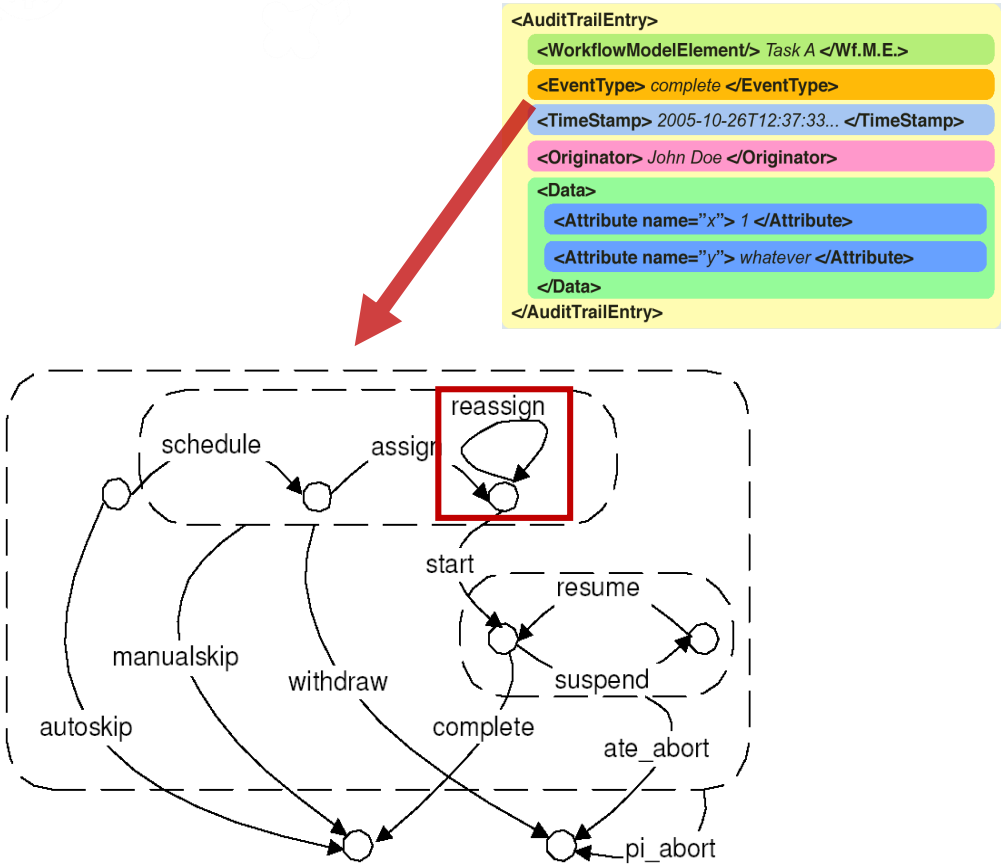
- Aim: Monitor how individual process instances are routed between originators
- Metrics
  - Handover of work
  - **Subcontracting**
  - Reassignment
  - Working together
  - Similar task



2.4 Extension

2. Social Network Miner

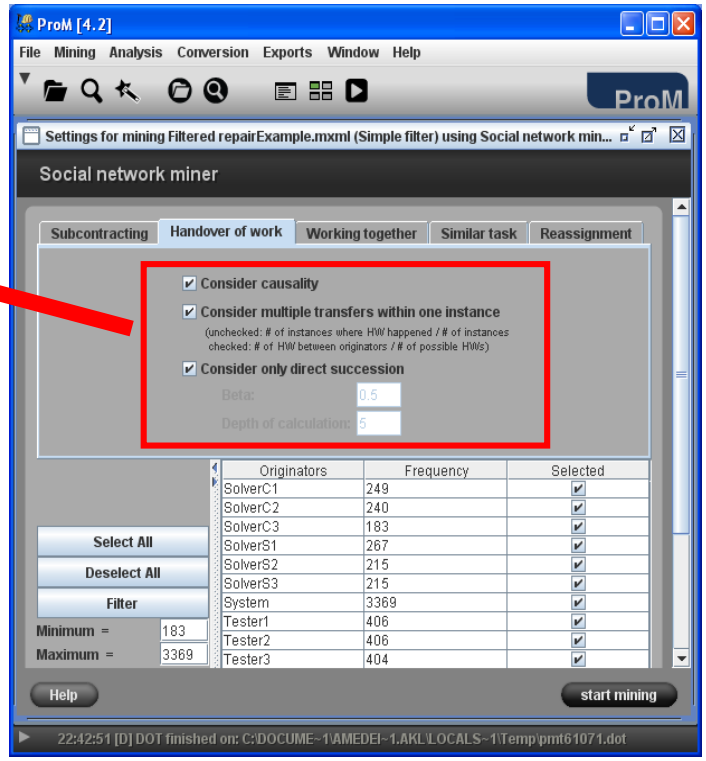
- Aim: Monitor how individual process instances are routed between originators
- Metrics
  - Handover of work
  - Subcontracting
  - **Reassignment**
  - Working together
  - Similar task



2.4 Extension

2. Social Network Miner

- Aim: Monitor how individual process instances are routed between originators
- Metrics
  - Handover of work
  - Subcontracting
  - Reassignment
  - Working together
  - Similar task



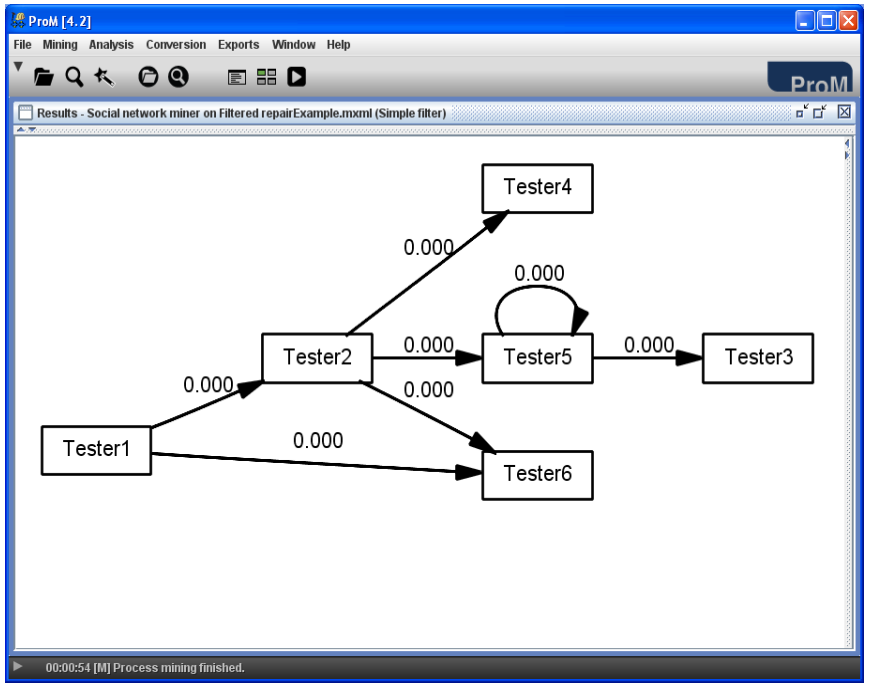
Based on ordering relations derived from a log!

2.4 Extension

2. Social Network Miner

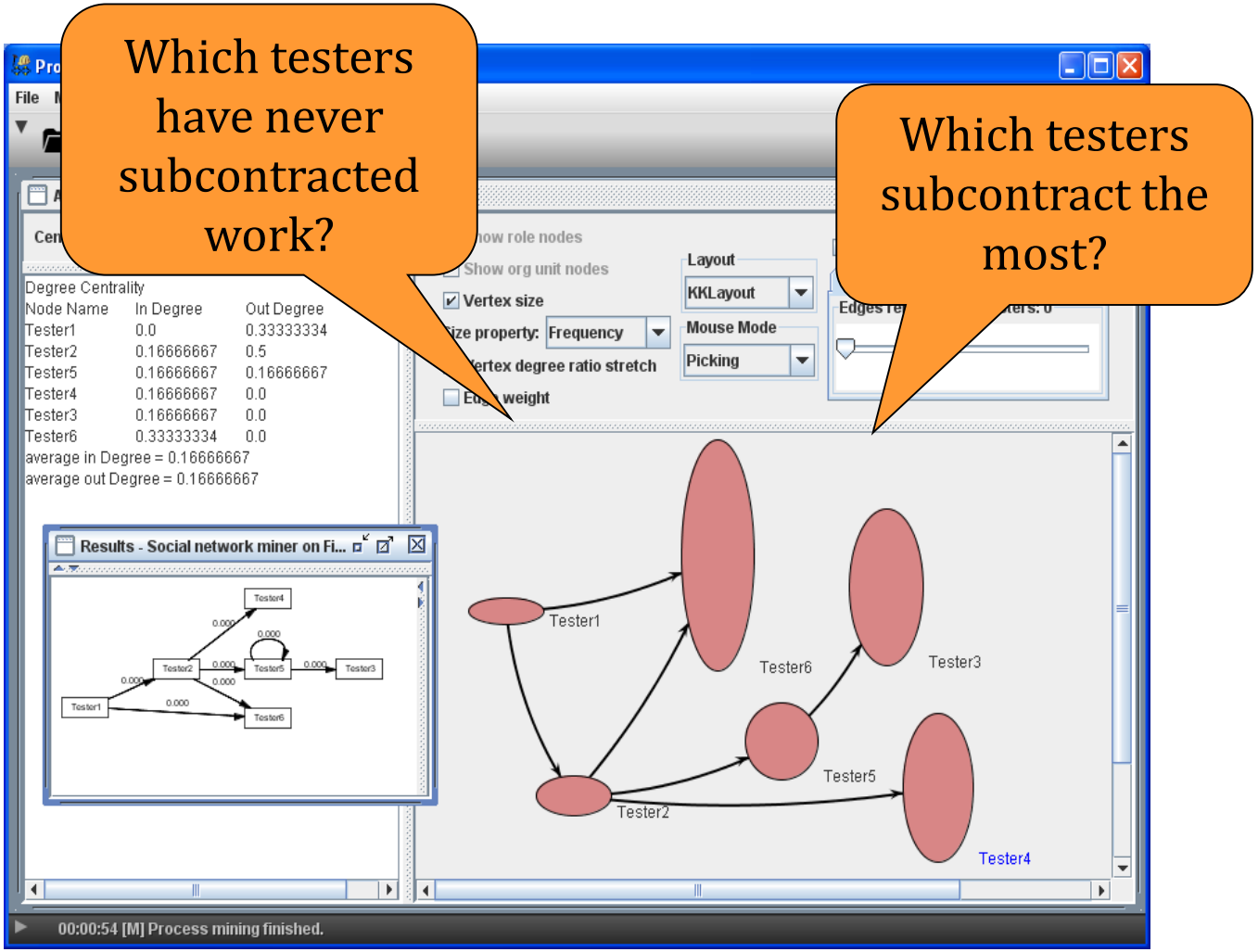
Analyze Social Network

- Better graphical view for the results for the Social Network Miner
- Includes different metrics to measure centrality of nodes
- Example : subcontracting



2.4 Extension

2. Social Network Miner

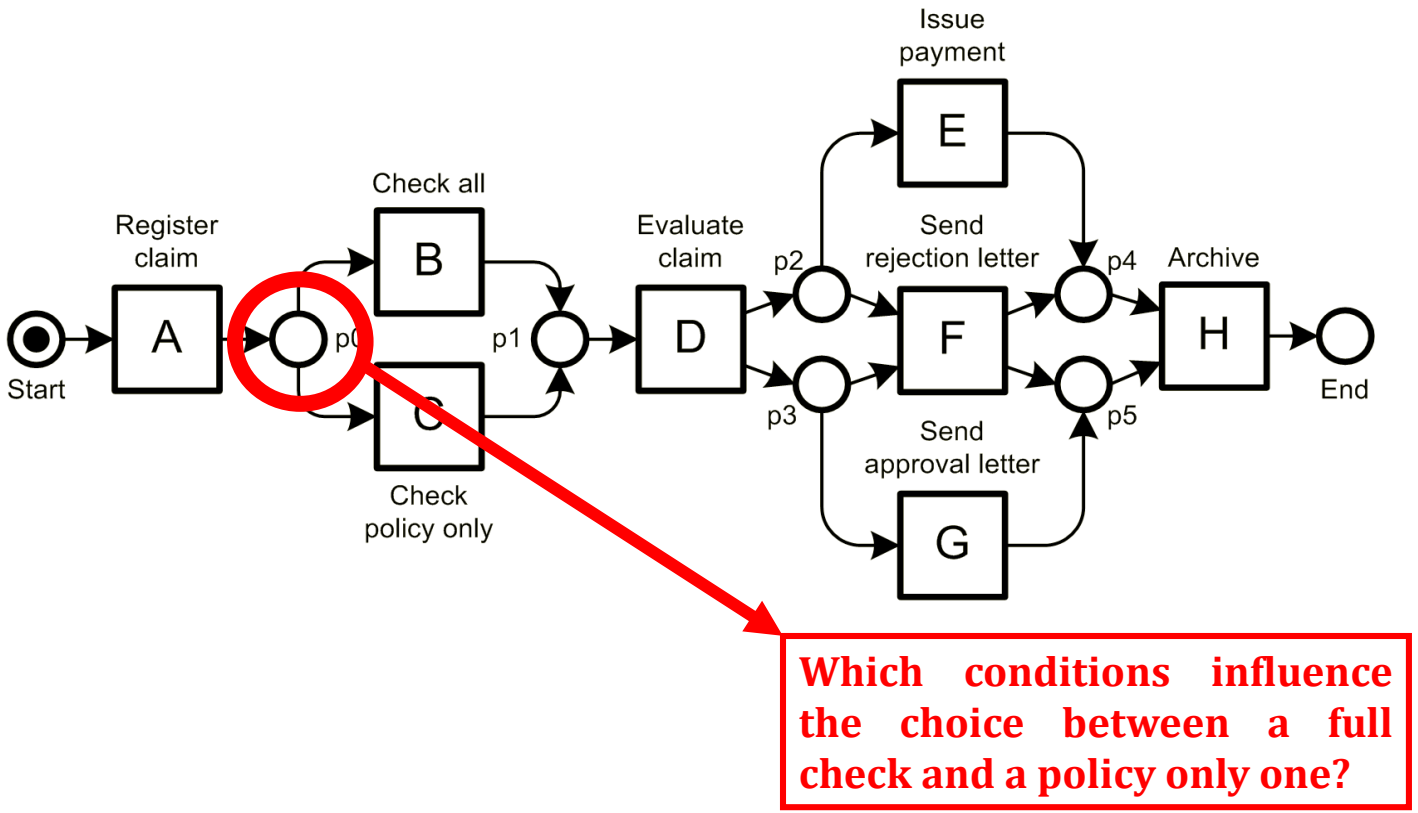


2.4 Extension

3. Decision Point Analysis

Main Idea

- Detection of data dependencies that affect the rounting the routing of process instances

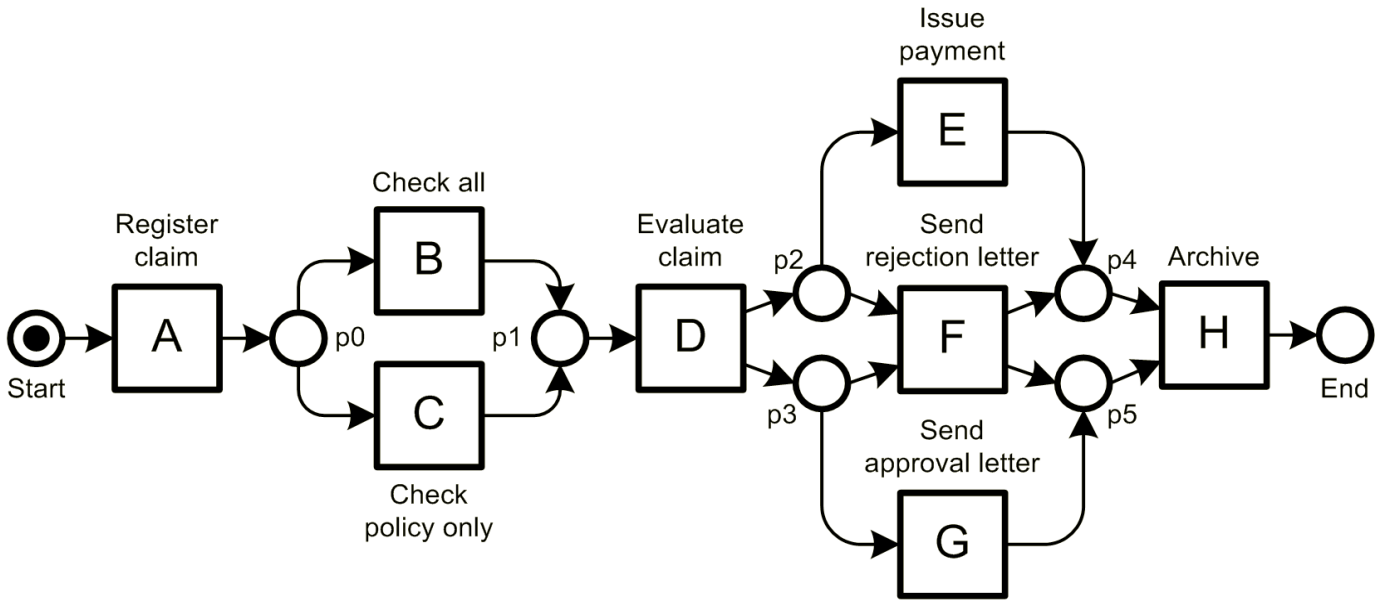


2.4 Extension

3. Decision Point Analysis

Motivation

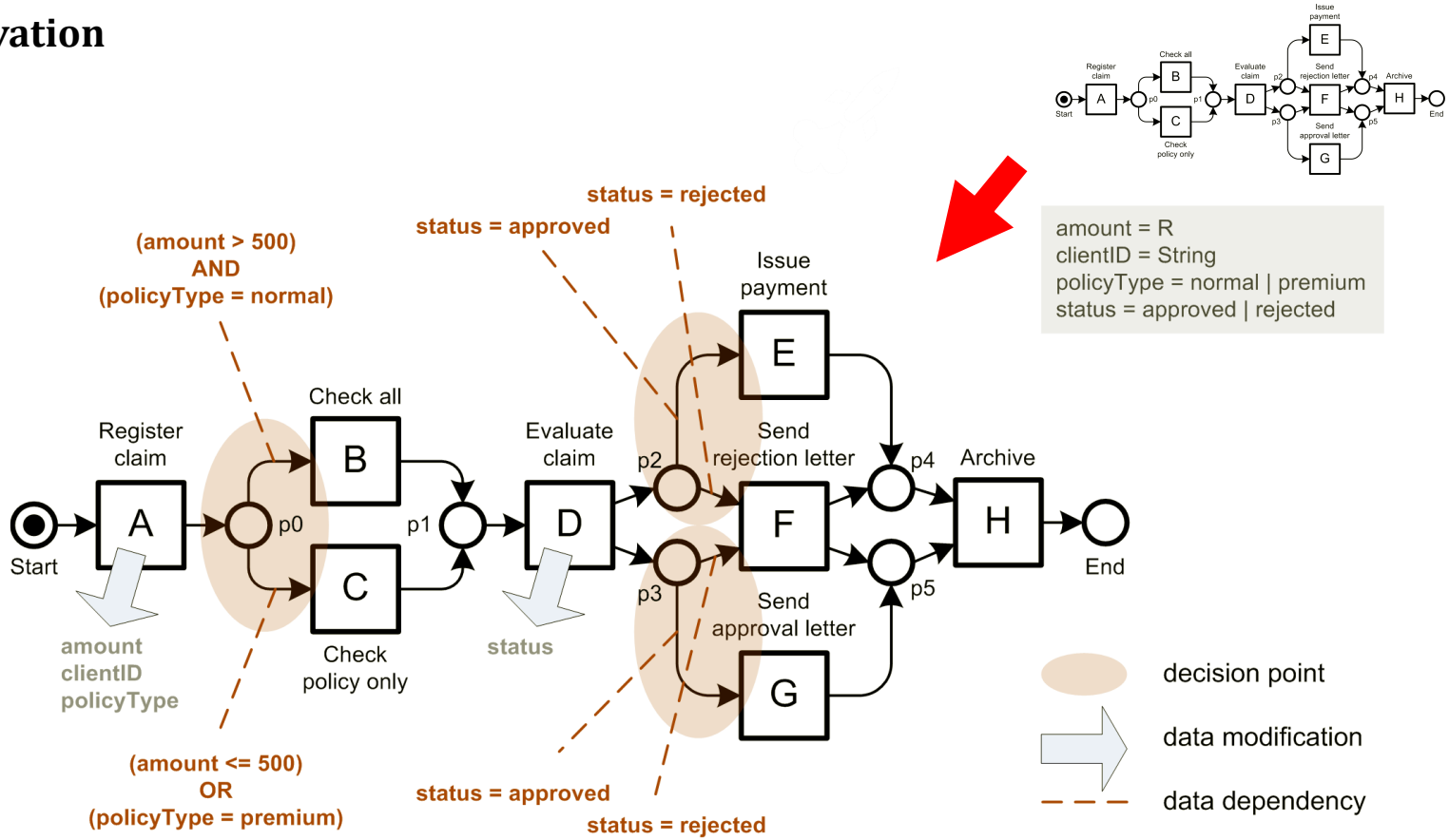
- Make tacit knowledge explicit
- Better understand the process model



2.4 Extension

3. Decision Point Analysis

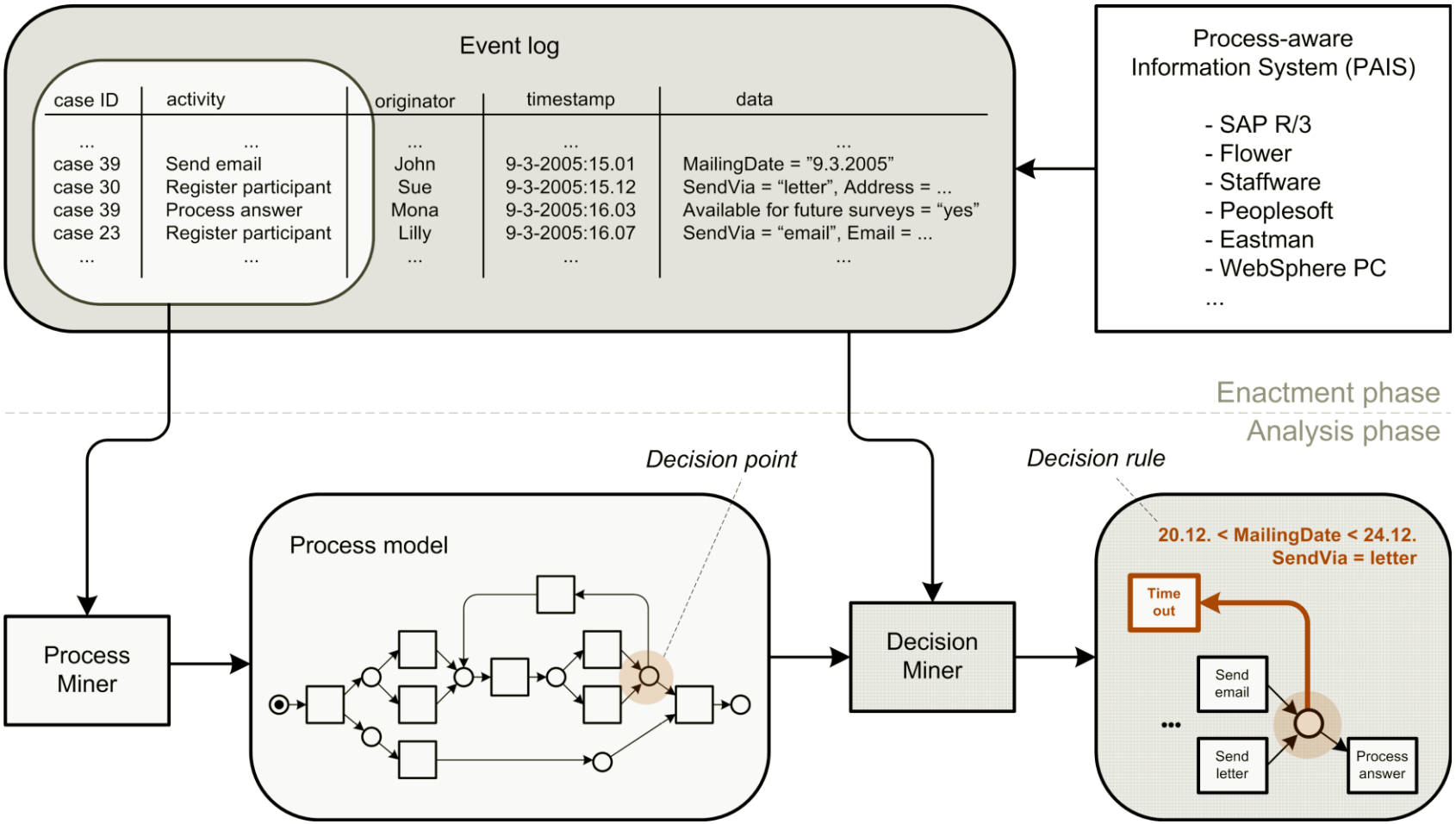
Motivation



2.4 Extension

3. Decision Point Analysis

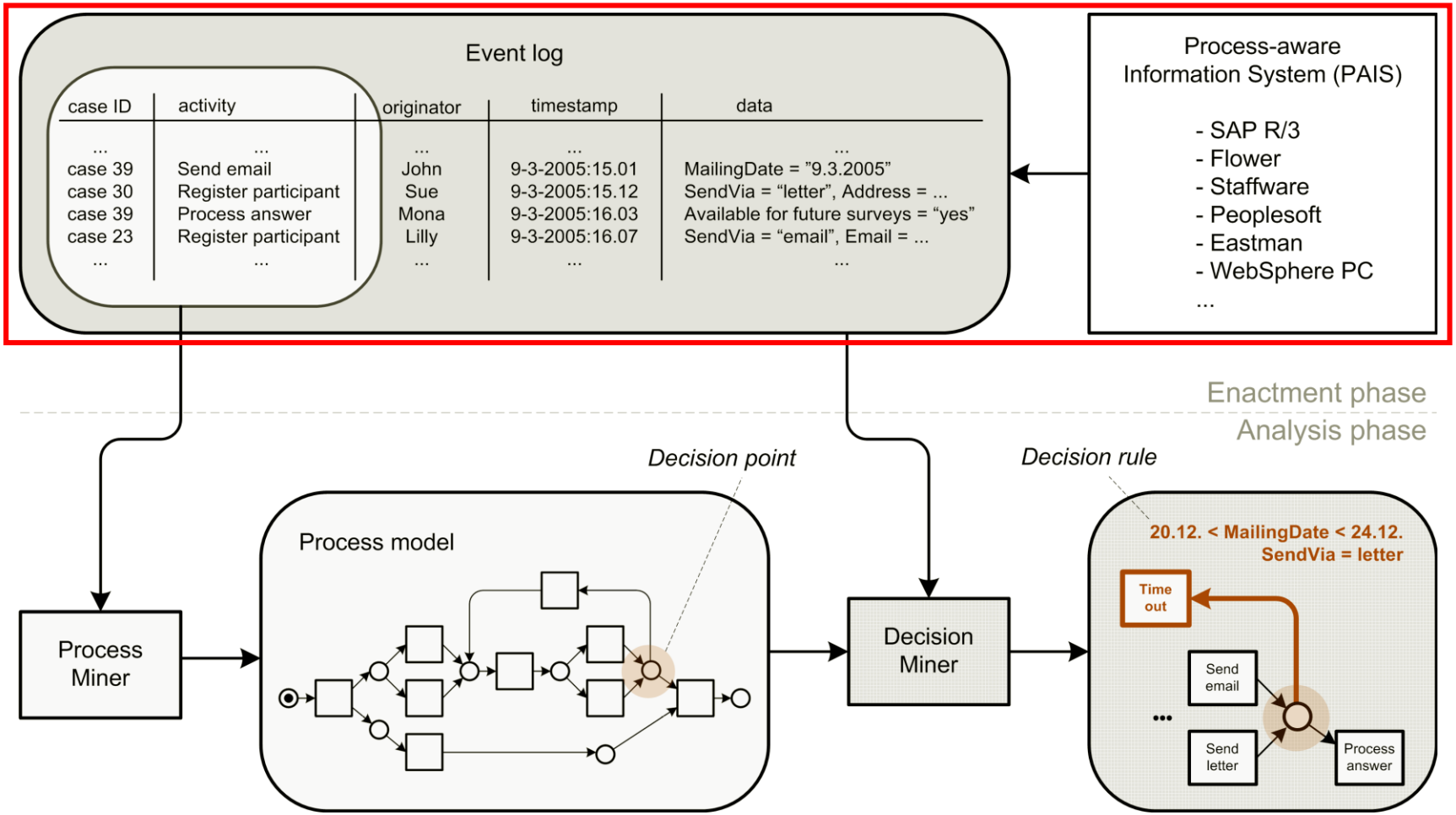
Approach



2.4 Extension

3. Decision Point Analysis

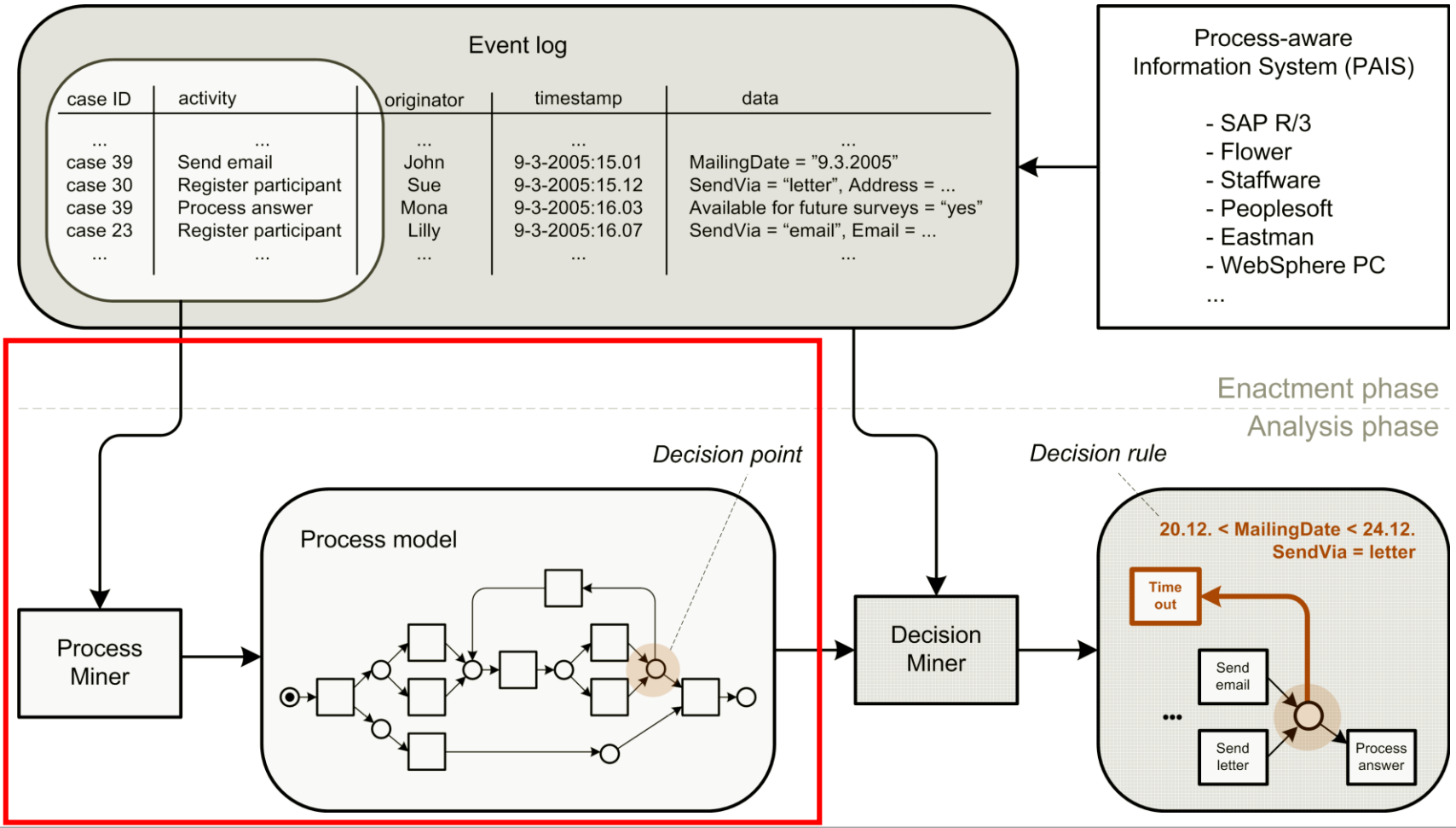
Approach



2.4 Extension

3. Decision Point Analysis

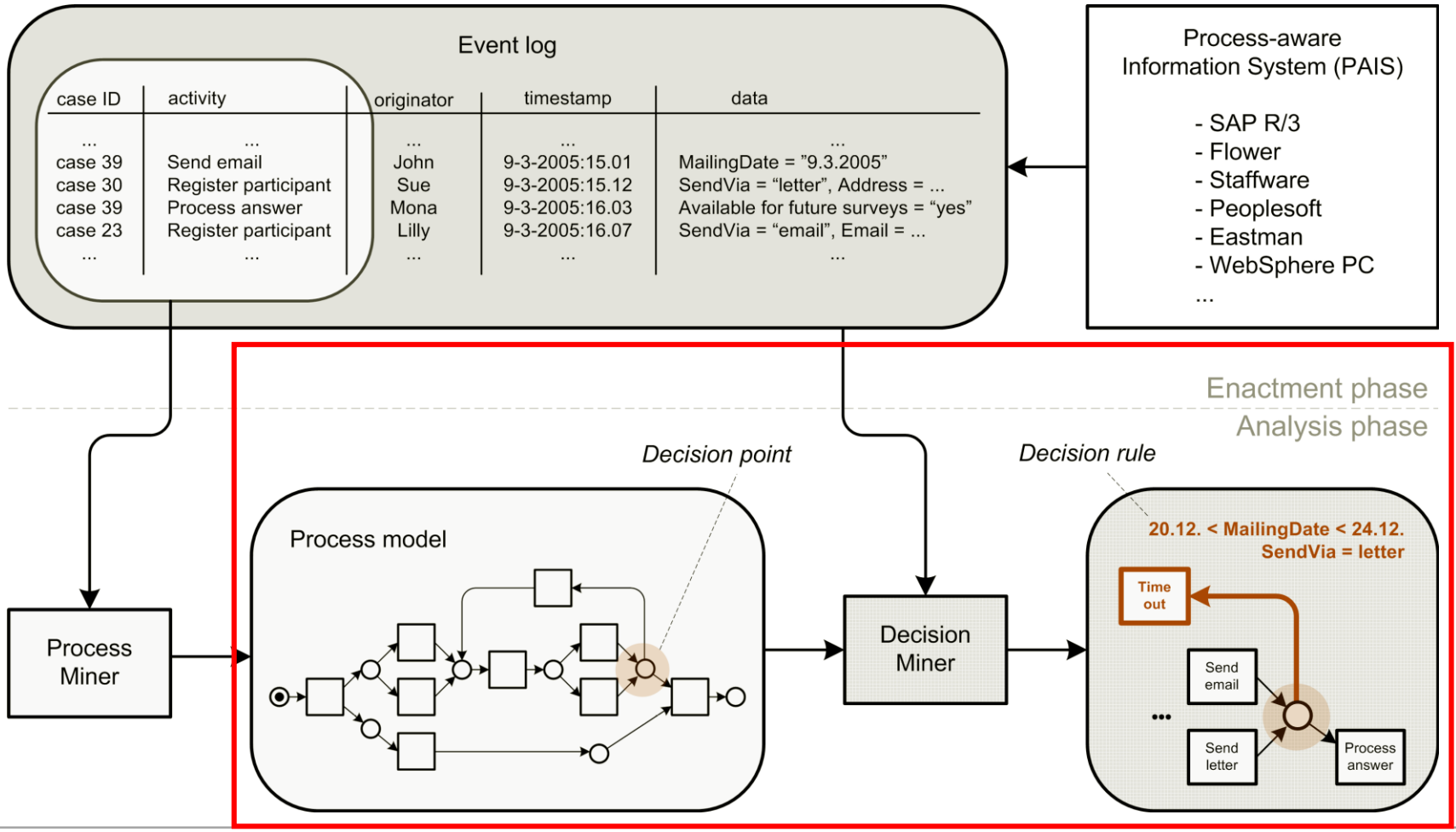
Approach



2.4 Extension

3. Decision Point Analysis

Approach



2.4 Extension

3. Decision Point Analysis

Algorithm's Main Steps

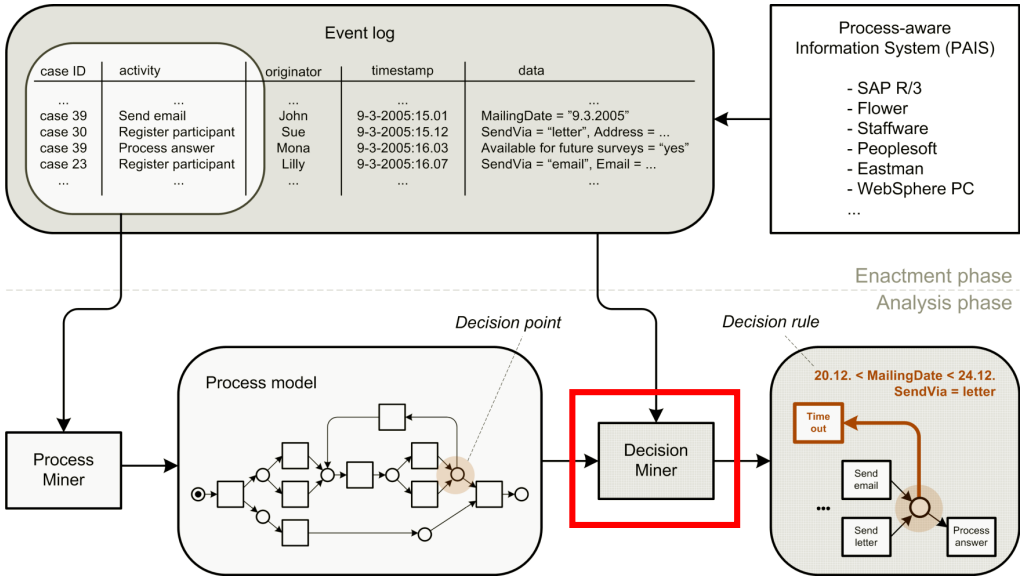
1. Read a log + model
2. Identify the decision points in a model
3. Find out which alternative branch has been taken for a given process instance and decision point
4. Discover the rules for each decision point
5. Return the enhanced model with the discovered rules

2.4 Extension

3. Decision Point Analysis

Algorithm's Main Steps

1. Read a log + model
2. **Identify the decision points in a model**
3. Find out which alternative branch has been taken for a given process instance and decision point
4. Discover the rules for each decision point
5. Return the enhanced model with the discovered rules

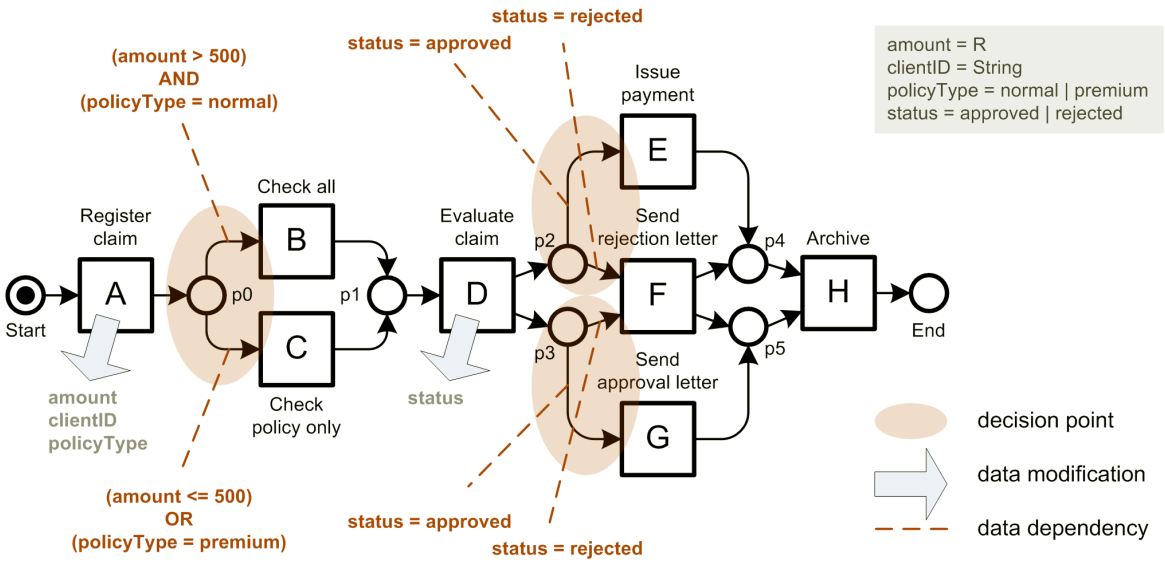


2.4 Extension

3. Decision Point Analysis

Algorithm's Main Steps

1. Read a log + model
2. Identify the decision points in a model
3. Find out which alternative branch has been taken for a given process instance and decision point
4. **Discover the rules for each decision point**
5. Return the enhanced model with the discovered rules

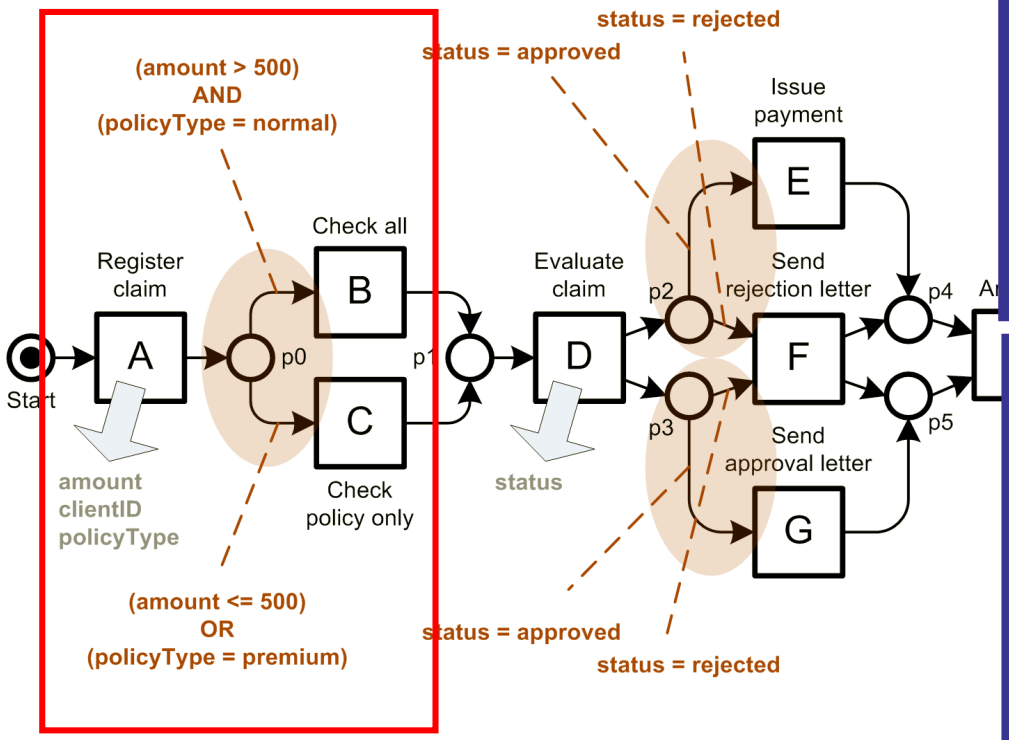


2.4 Extension

3. Decision Point Analysis

Algorithm's Main Steps

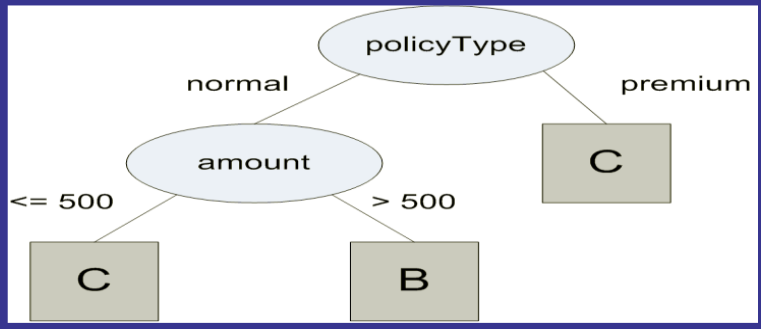
4. **Discover the rules for each decision point**



Training examples for decision point "p0"

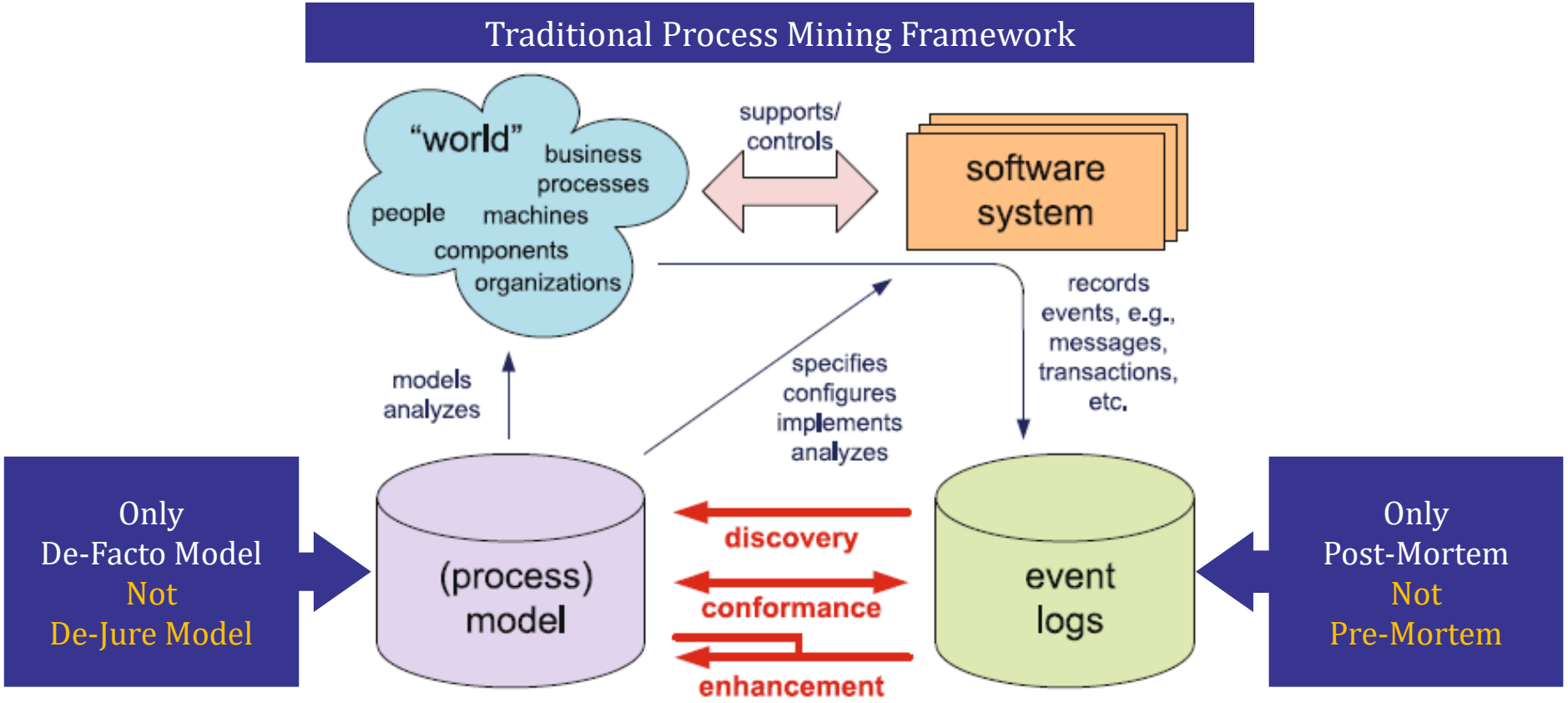
amount	clientID	policyType	class
1000	C567894938	premium	C
700	C938609223	normal	B
550	C135697567	normal	B
500	C568120443	normal	C
50	C493823084	normal	C
200	C945675110	premium	C

Discovered decision tree for point "p0"



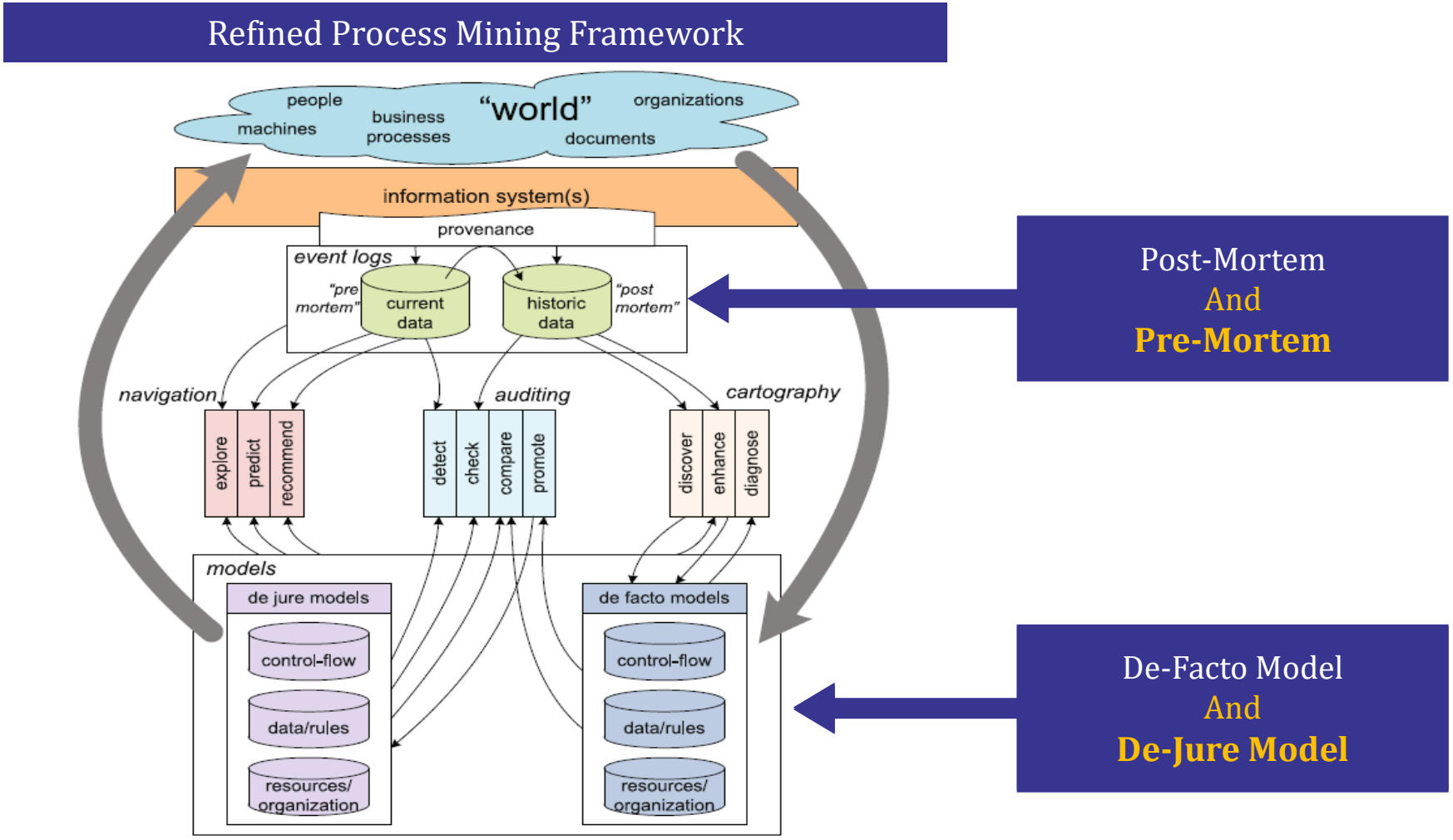
2.5 Refined Process Mining Framework

Bridge between Traditional and Refined Framework



2.5 Refined Process Mining Framework

Bridge between Traditional and Refined Framework



2.5 Refined Process Mining Framework

1. Cartography

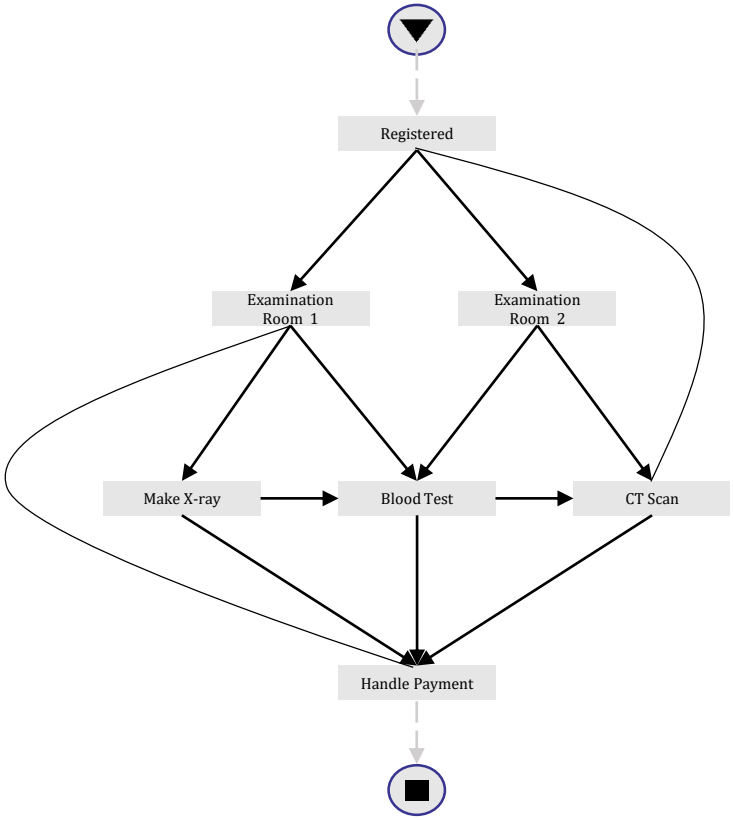
1) Discovery

- 전통적인 프로세스 마이닝의 Discovery와 동일
- Activity 간의 순서와 관계 도식화
- Petri-net, Process Tree등으로 표현

Patient	Activity	Timestamp
5781	Registered	23-1-2023@10.00
5781	Examination Room 1	23-1-2023@10.15
5781	Make X-ray	23-1-2023@10.30
5833	Blood test	23-1-2023@10.27
5781	Blood test	23-1-2023@10.49
5781	CT scan	23-1-2023@11.10
5833	Surgery	23-1-2023@12.34
5781	Handle payment	23-1-2023@12.41

Event logs

Discovery



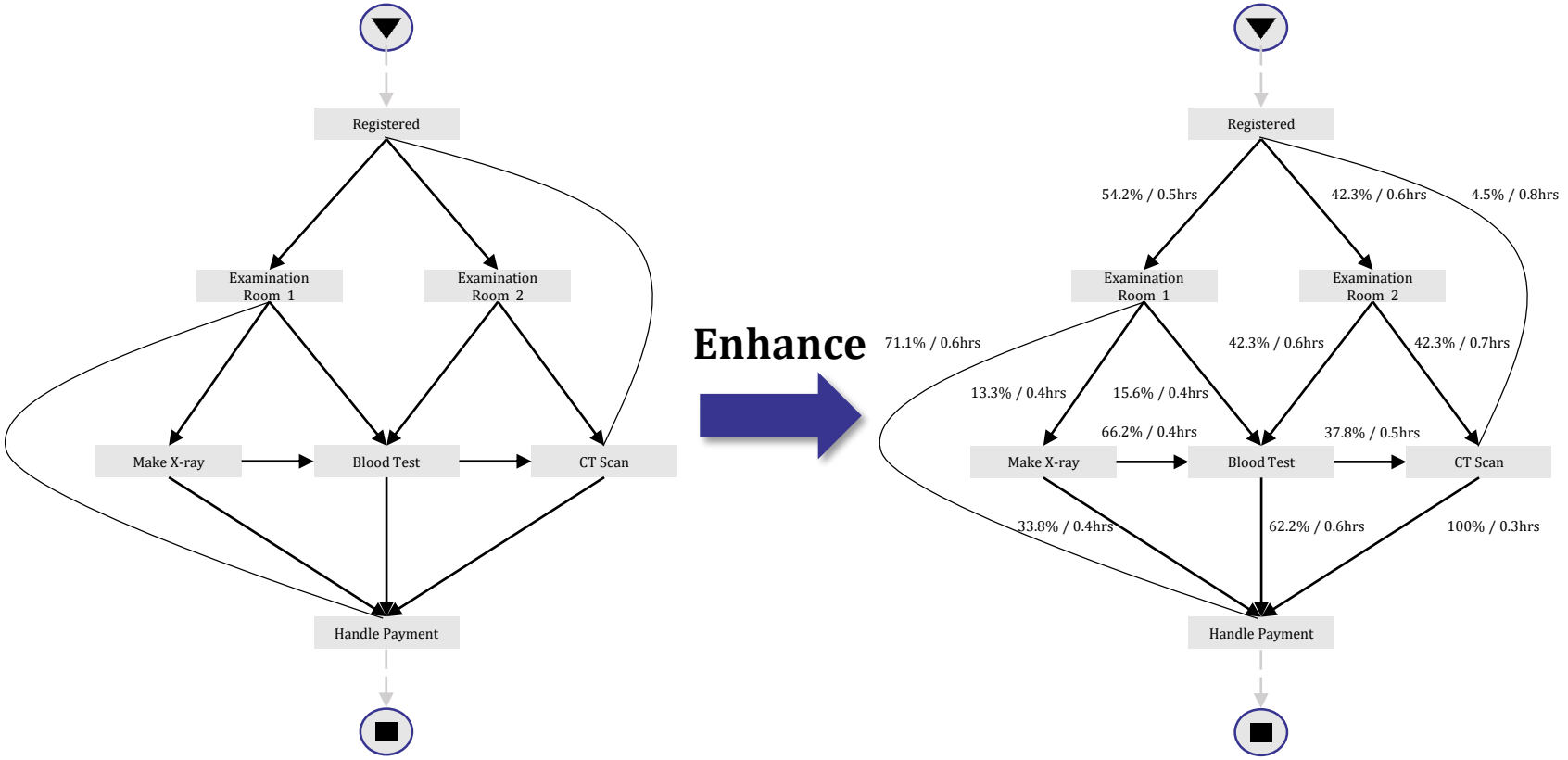
Model

2.5 Refined Process Mining Framework

1. Cartography

2) Enhance

- 기존의 모델을 개선하거나 보완
- 빈도 및 평균소요 시간 등의 정보를 추가하여 프로세스 맵 개선



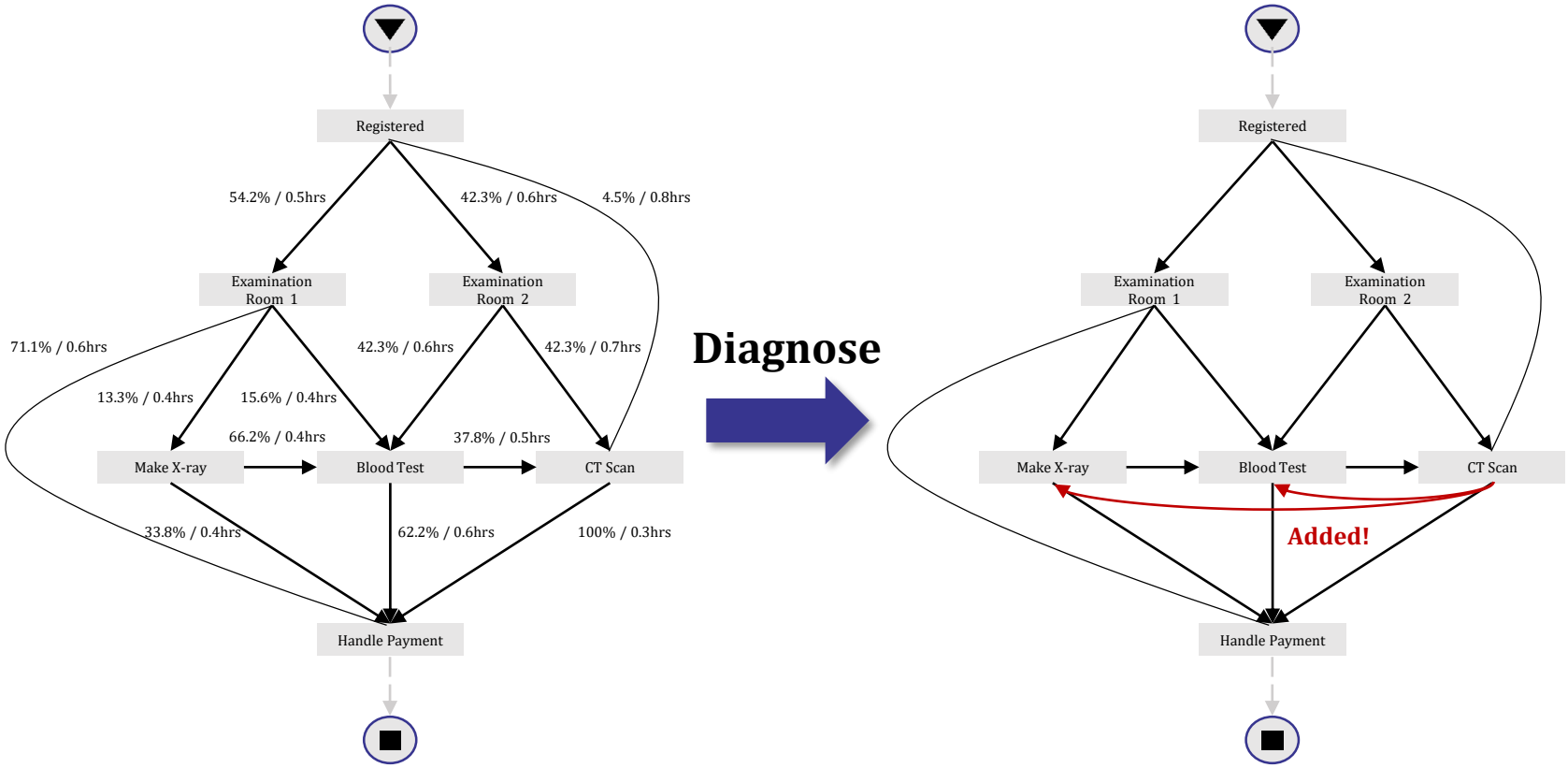
Pre-defined Model **Enhanced Model**

2.5 Refined Process Mining Framework

1. Cartography

3) Diagnose

- Soundness, 대기 시간 분석 등의 모든 모델의 성능 분석
- 모델 자체를 진단하여 분석의 결과를 바탕으로 프로세스 효율 개선



Pre-defined Model **Improved Model**

2.5 Refined Process Mining Framework

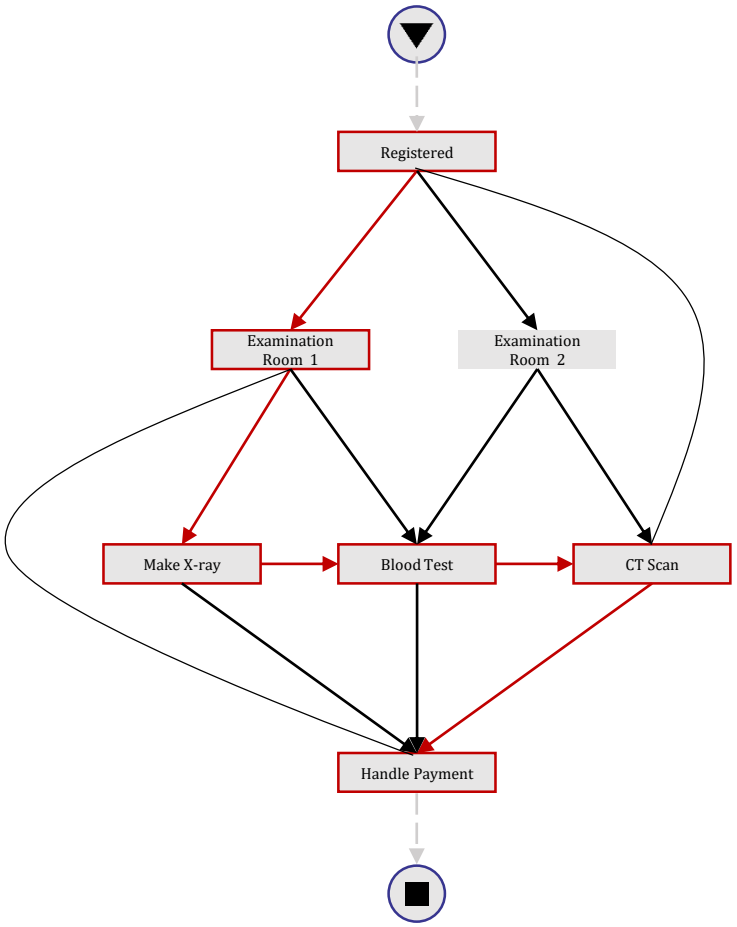
2. Auditing

1) Detect

- 편차를 감지
- Pre-Mortem Data과 De-Jure Model 을 비교
- 규칙 위반하면 경고 생성

Patient	Activity	Timestamp
5781	Registered	23-1-2023@10.00
5781	Examination Room 1	23-1-2023@10.15
5781	Make X-ray	23-1-2023@10.30
5833	Blood test	23-1-2023@10.27
5781	Blood test	23-1-2023@10.49
5781	CT scan	23-1-2023@11.10
5833	Surgery	23-1-2023@12.34
5781	Handle payment	23-1-2023@12.41

No Alert



Pre-Mortem Data

De-Jure Model

2.5 Refined Process Mining Framework

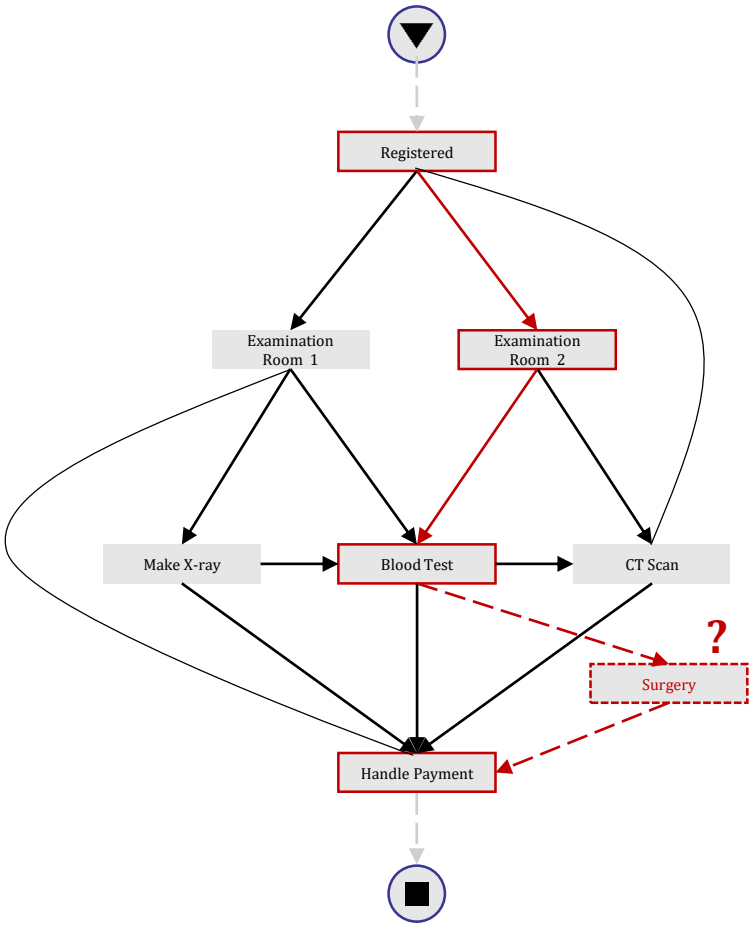
2. Auditing

1) Detect

- 편차를 감지
- Pre-Mortem Data과 De-Jure Model 을 비교
- 규칙 위반하면 경고 생성

Patient	Activity	Timestamp
5733	Registered	23-1-2023@10.00
5733	Examination Room 2	23-1-2023@10.15
5781	Make X-ray	23-1-2023@10.30
5833	Blood test	23-1-2023@10.27
5781	Blood test	23-1-2023@10.49
5781	CT scan	23-1-2023@11.10
5833	Surgery	23-1-2023@12.34
5833	Handle payment	23-1-2023@12.41

Alert!



Pre-Mortem Data

De-Jure Model

2.5 Refined Process Mining Framework

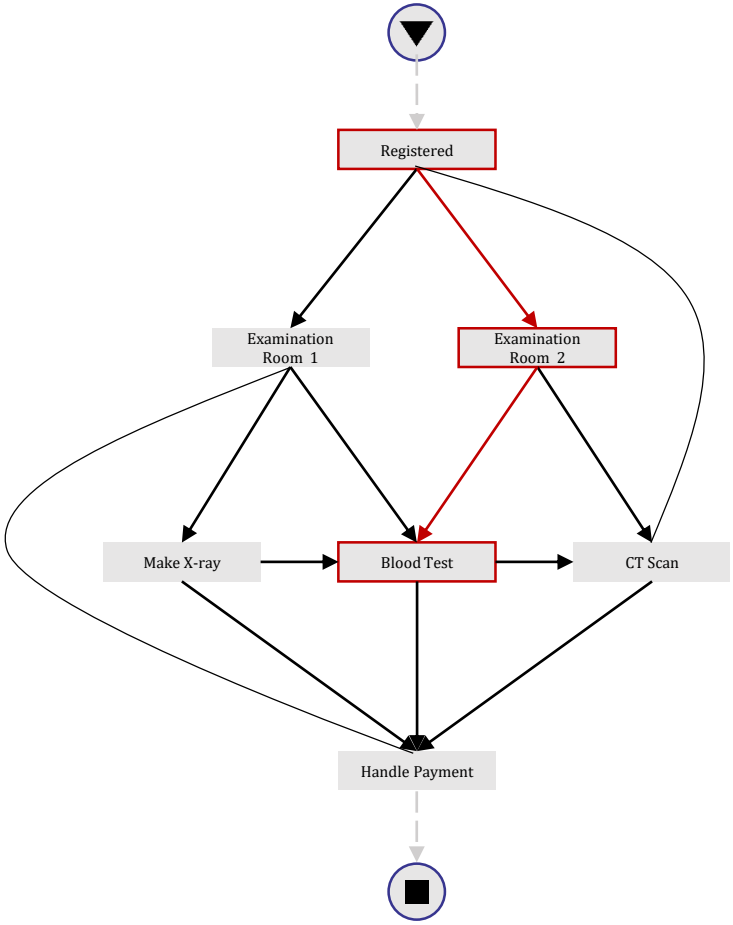
2. Auditing

2) Check

- 편차 찾기, 준수 수준 정량화
- Post-Mortem Data과 De-Jure Model을 비교

Patient	Activity	Timestamp
9255	Registered	25-5-2023@10.00
9255	Examination Room 2	23-5-2023@10.15
9255	Blood test	23-5-2023@10.21

Post-Mortem Data



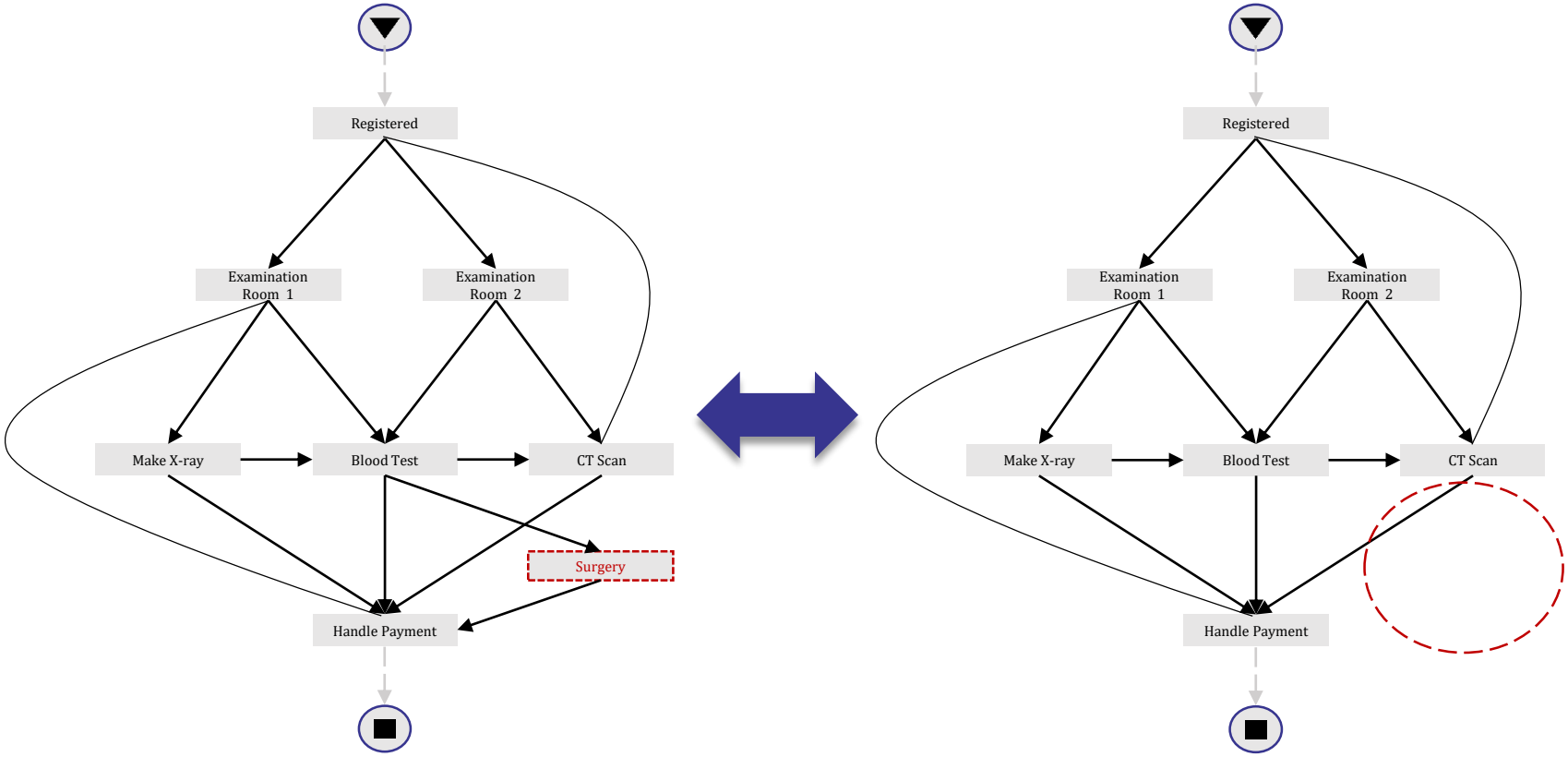
De-Jure Model

2.5 Refined Process Mining Framework

2. Auditing

3) Compare

- De-Facto Model과 De-Jure Model을 비교
- 이상적 모델과 현실의 차이 설명



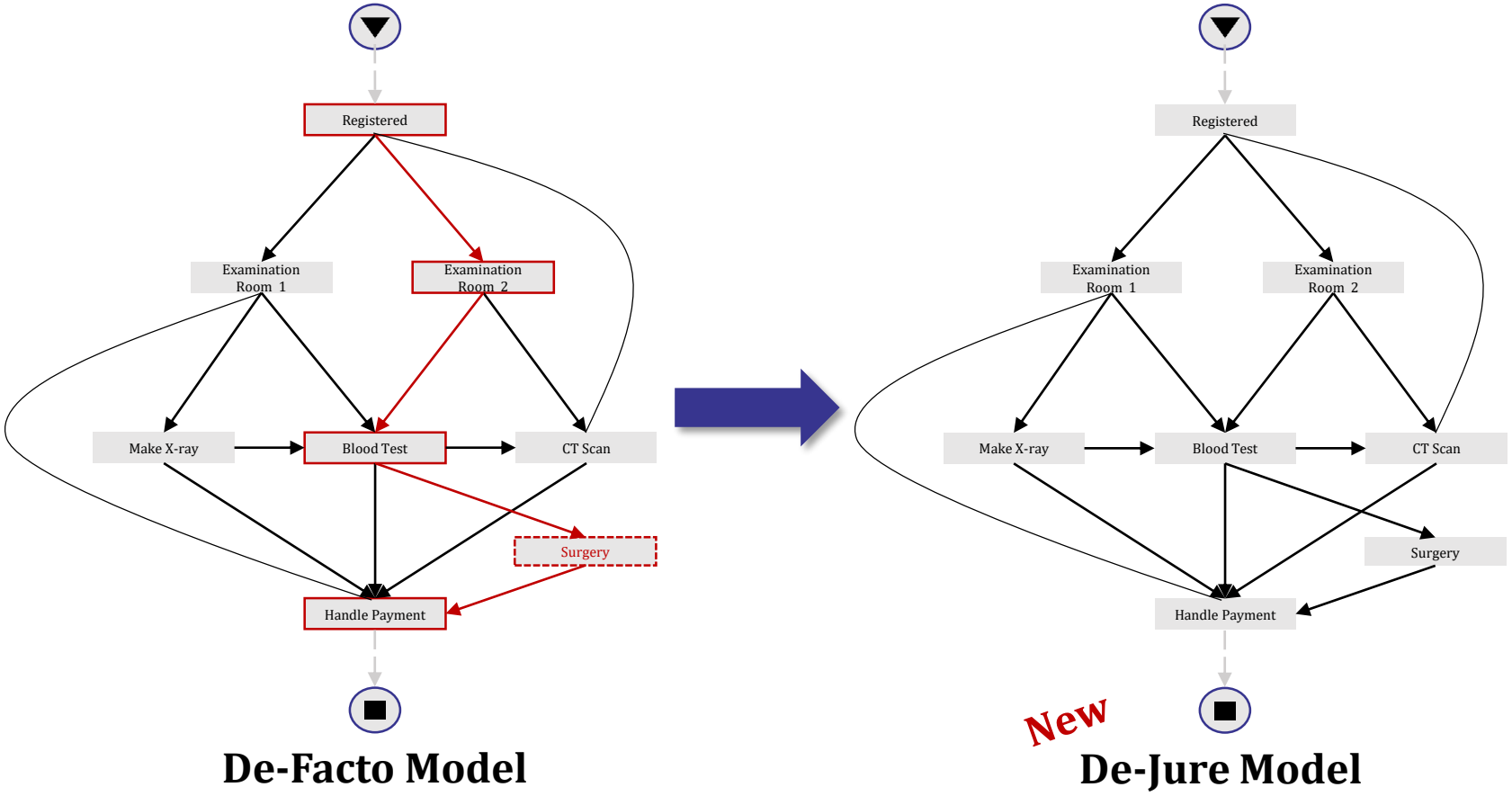
De-Facto Model **De-Jure Model**

2.5 Refined Process Mining Framework

2. Auditing

4) Promote

- De-Facto Model을 De-Jure Model과 비교하여 새로운 De-Jure Model 생성
- 검증된 모범 사례를 De-Jure Model로 승격해, 기존 프로세스 개선



De-Facto Model **De-Jure Model**

2.5 Refined Process Mining Framework

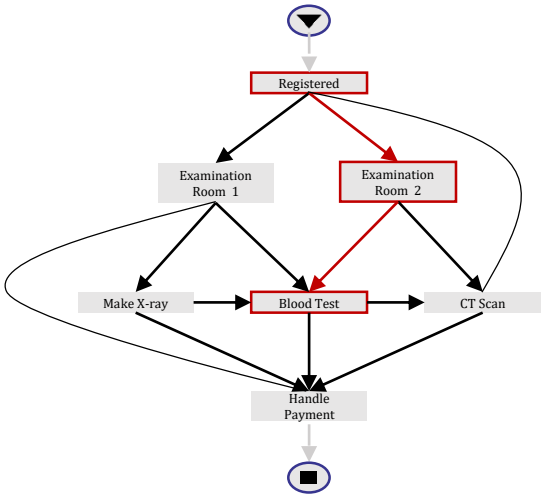
3. Navigation

1) Explore

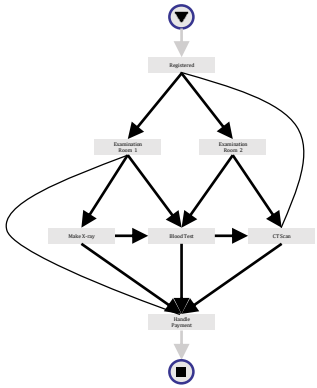
- Event Data와 Model을 이용해 런타임에 비즈니스 프로세스 탐색
- 진행 중인 케이스 시각화, 이전에 처리한 유사 케이스들과 비교

Patient	Activity	Timestamp
9255	Registered	25-5-2023@10.00
9255	Examination Room 2	23-5-2023@10.15
9255	Blood test	23-5-2023@10.21

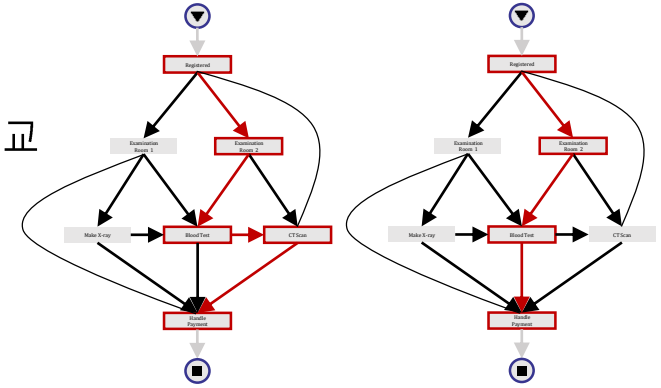
Event Data + Model



프로세스 탐색



처리한 케이스와 비교



2.5 Refined Process Mining Framework

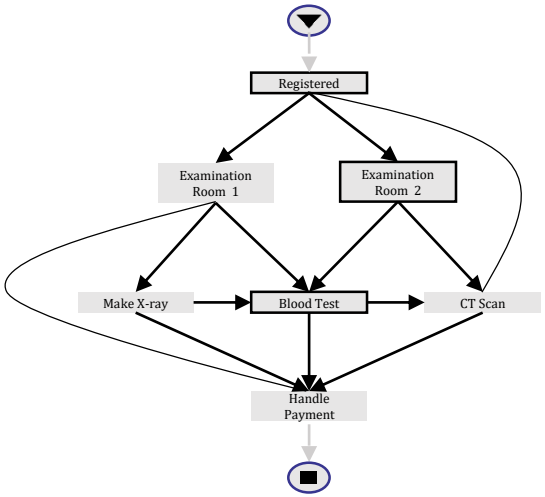
3. Navigation

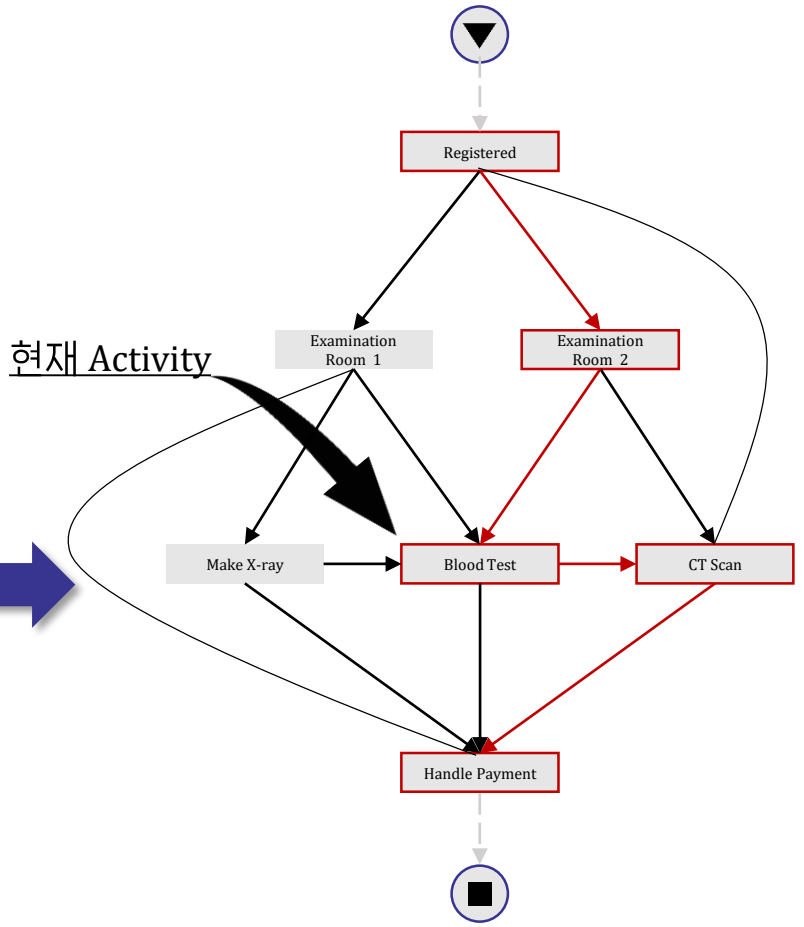
2) Predict

- 진행 중인 케이스의 정보(Running Case)를 모델과 결합
- Remaining-Time, **Next-Activity** 등에 대해 예측

Case	Activity
…	…
H	Registered → CT Scan → Handle Payment
I	Registered → Examination Room 2 → Blood Test → CT Scan → Handle Payment
J	Registered → Examination Room 2 → CT Scan → Handle Payment

Running Case + Model





Next-Activity : CT scan

2.5 Refined Process Mining Framework

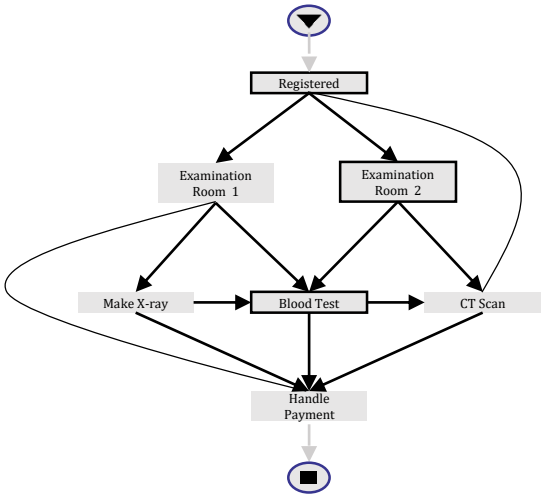
3. Navigation

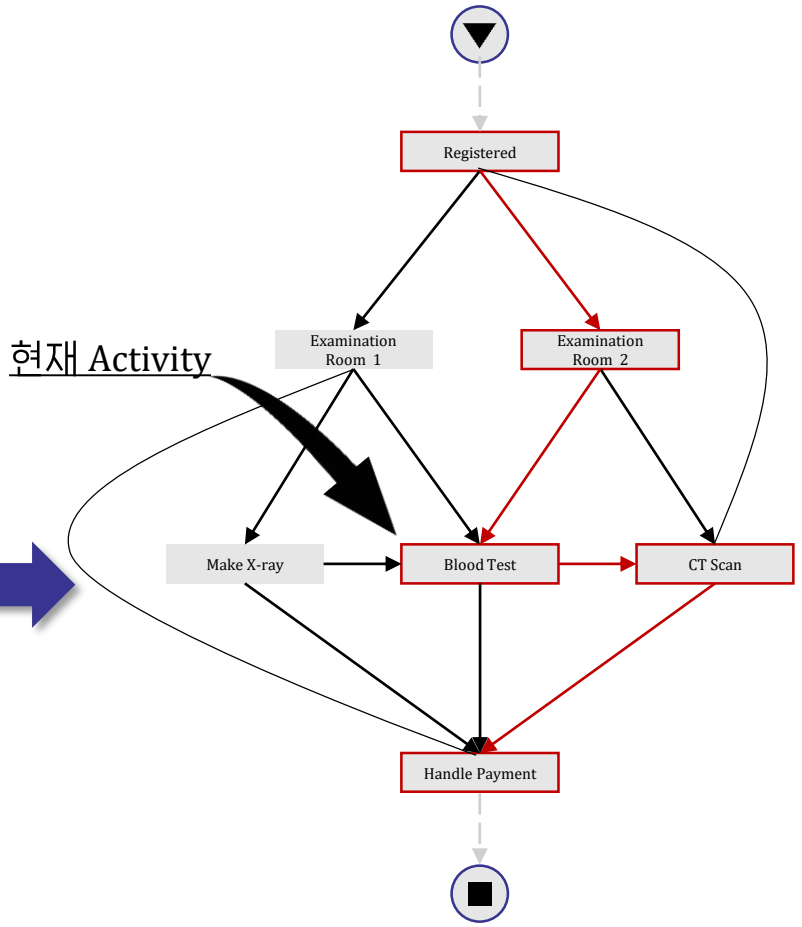
3) Recommend

- 진행 중인 케이스의 정보(Running Case)를 모델과 결합
- Remaining-Time 최소화, 비용 최소화 등을 위한 행동 추천

Case	Activity
…	…
H	Registered → CT Scan → Handle Payment
I	Registered → Examination Room 2 → Blood Test → CT Scan → Handle Payment
J	Registered → Examination Room 2 → CT Scan → Handle Payment

Running Case + Model





Next-Activity : CT scan

Recommend : 비어 있는 CT실로 안내