REBS 7.5: Exercises Process Correctness and Compliance

Hugo A. López
Software, Data, People and Society
lopez@di.ku.dk

UNIVERSITY OF COPENHAGEN

Agenda

1. Exercises Structural Correctness (soundness)
2. Exercises Semantic Correctness (compliance)

Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0}\bullet f \mid e\bullet\xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

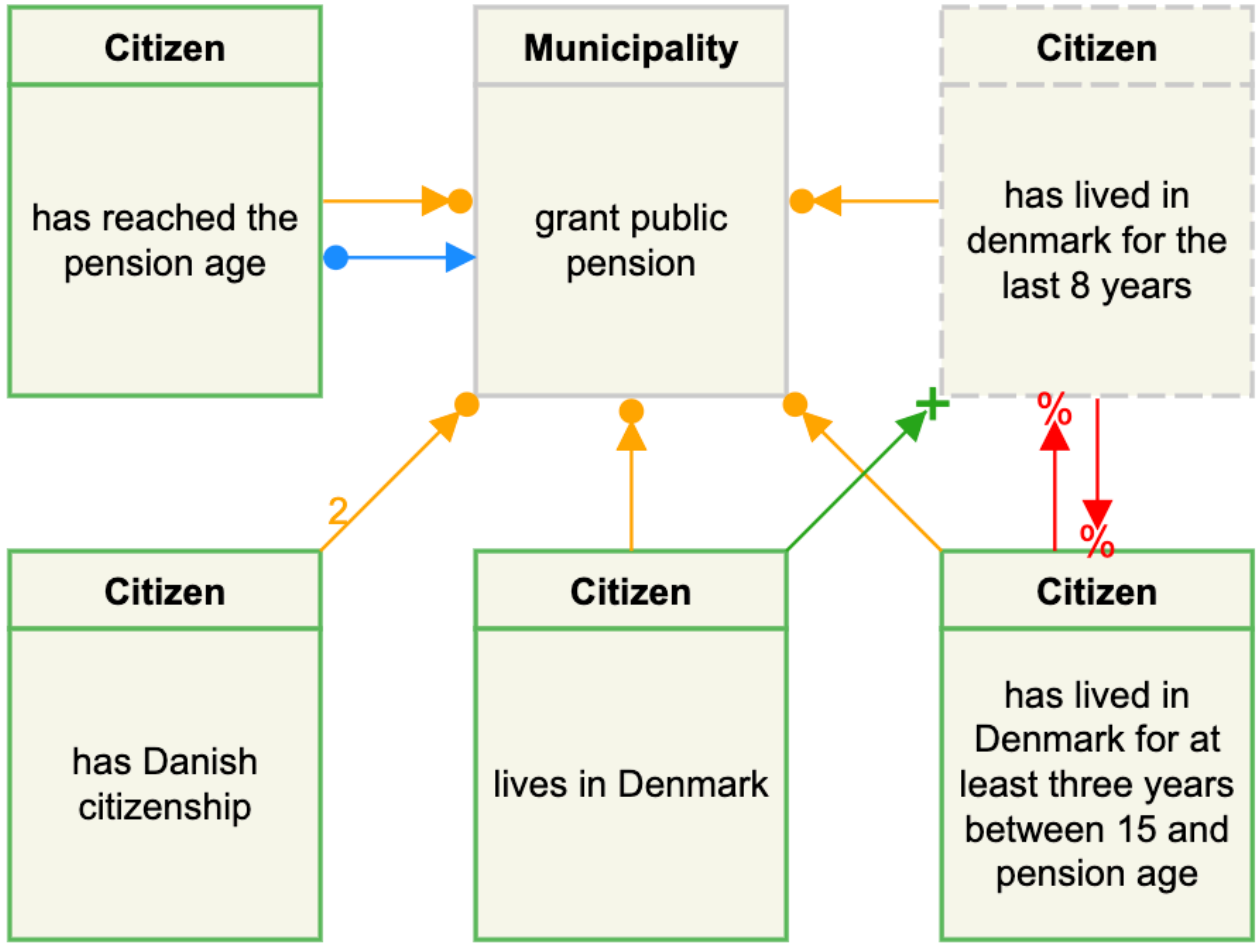
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0}\bullet f \mid e\bullet\xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

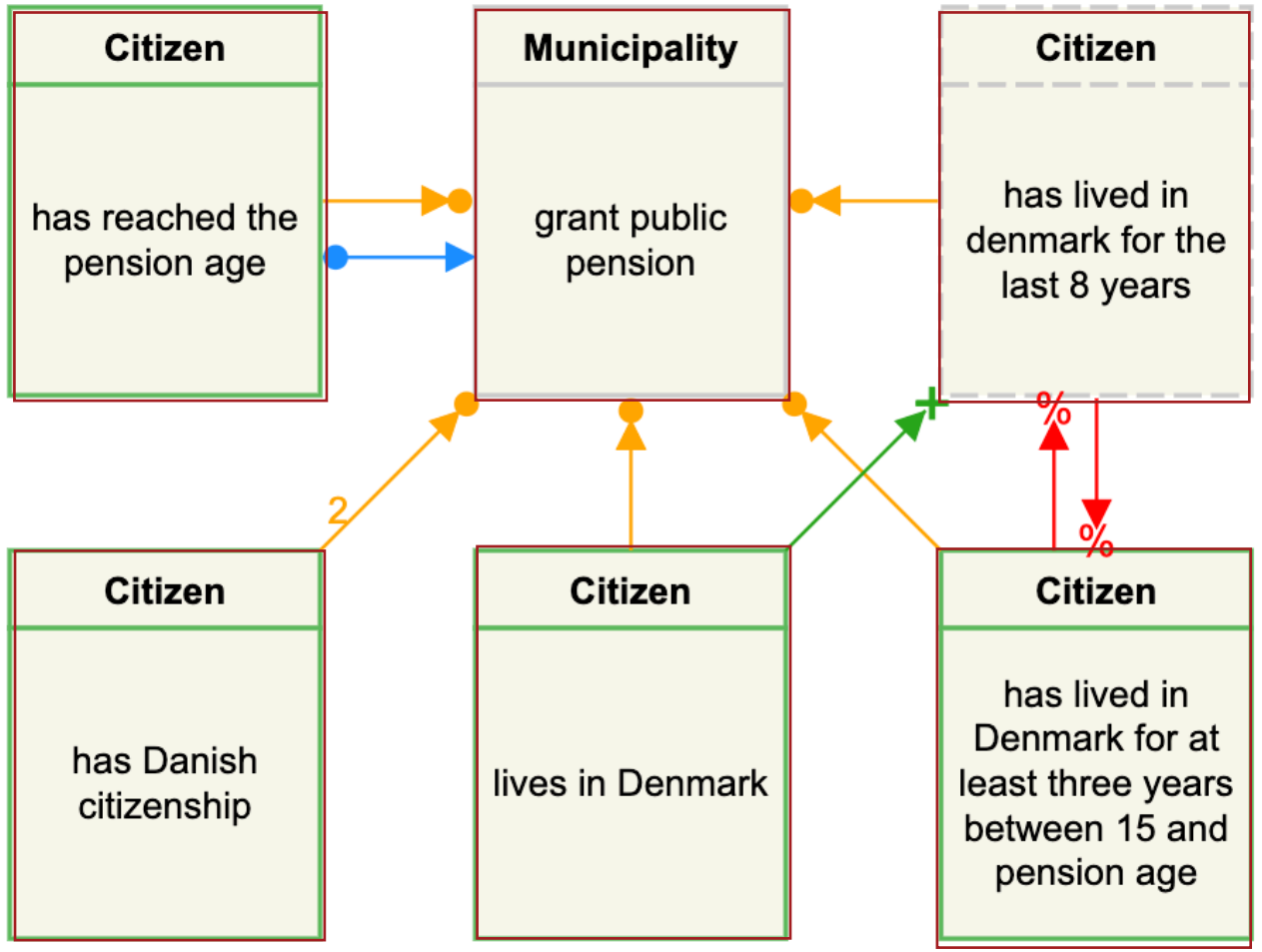
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0}\bullet f \mid e\bullet\xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

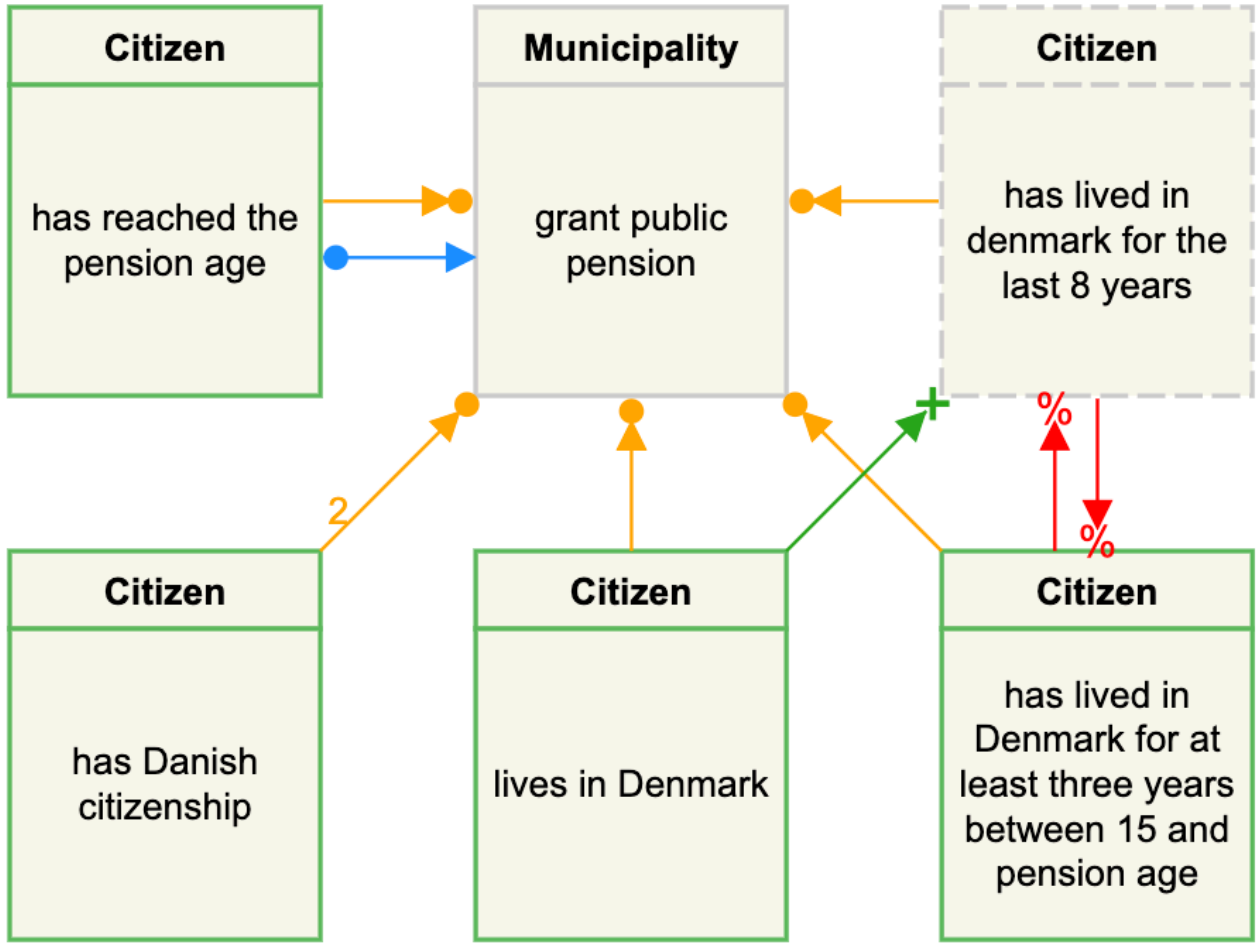
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$T, U ::= e \xrightarrow{t_0}\bullet f \mid e\bullet\xrightarrow{t_\omega} f \mid e \to + f \mid e \to \% f \mid e \to \diamond f \mid T \parallel U \mid 0$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

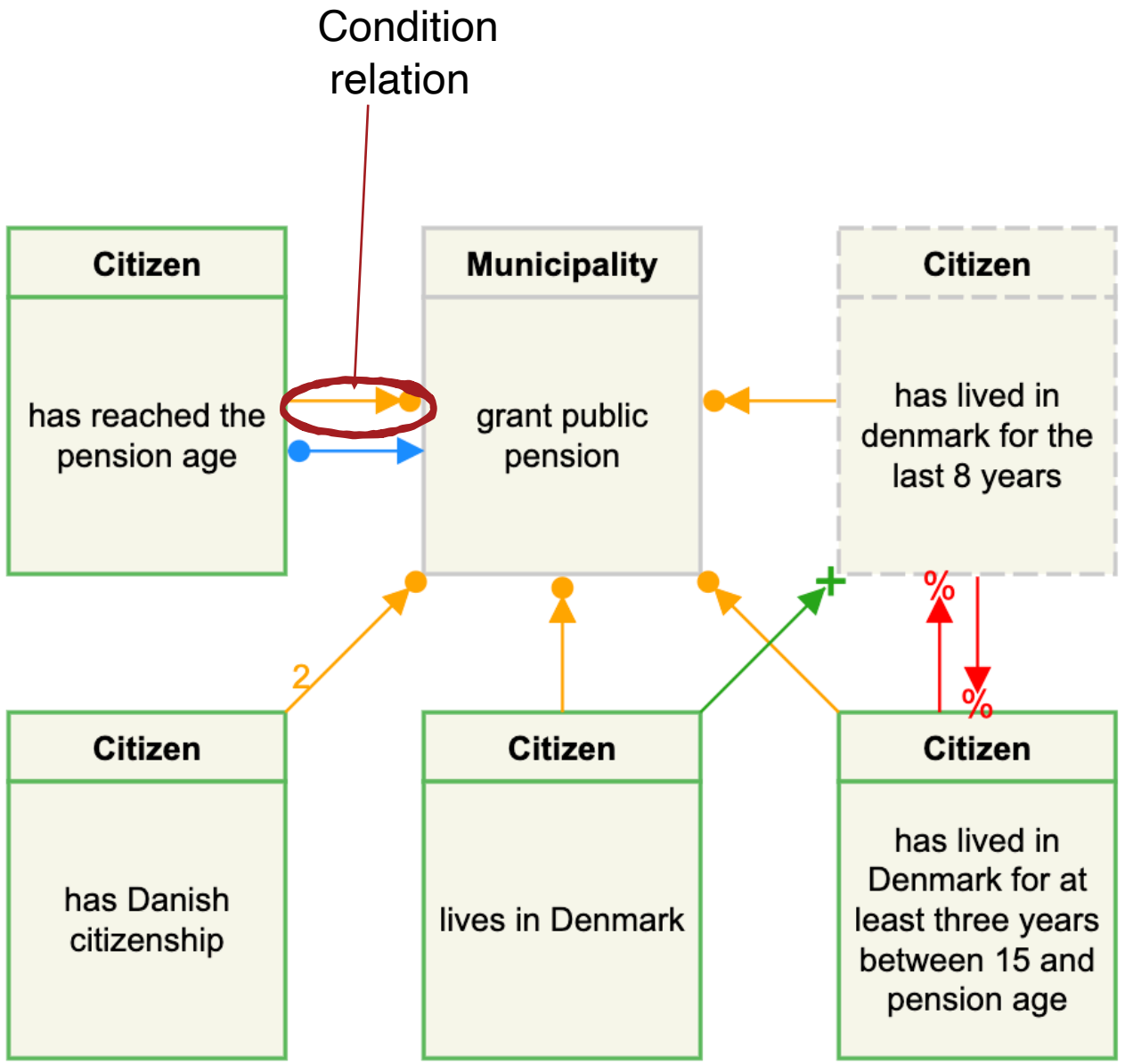
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$T, U ::= e \xrightarrow{t_0} \bullet f \mid e \bullet \xrightarrow{t_\omega} f \mid e \to + f \mid e \to \% f \mid e \to \diamond f \mid T \parallel U \mid 0$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

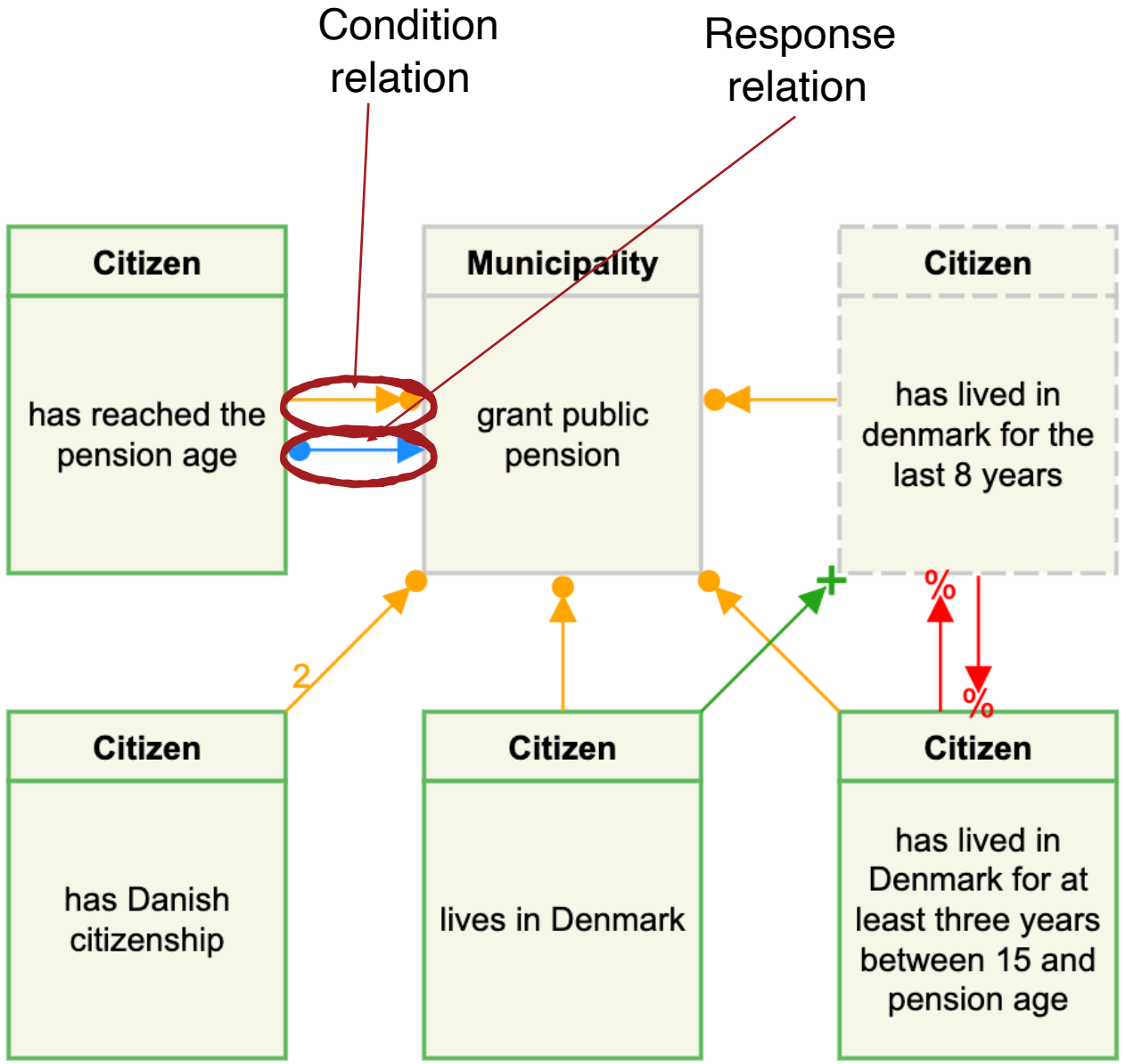
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0} \bullet f \mid e \bullet\!\!\xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

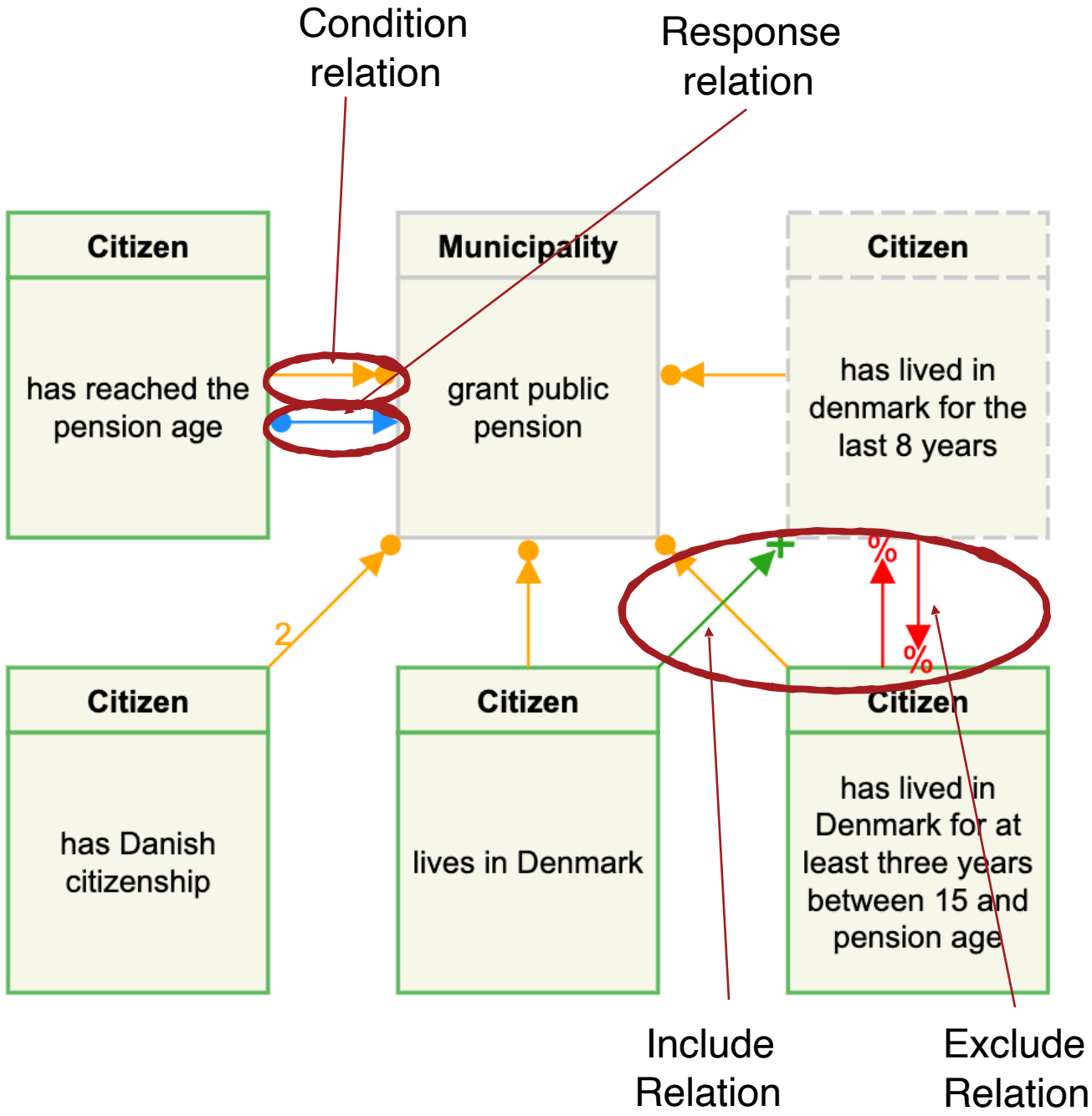
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0} \bullet f \mid e \bullet \xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

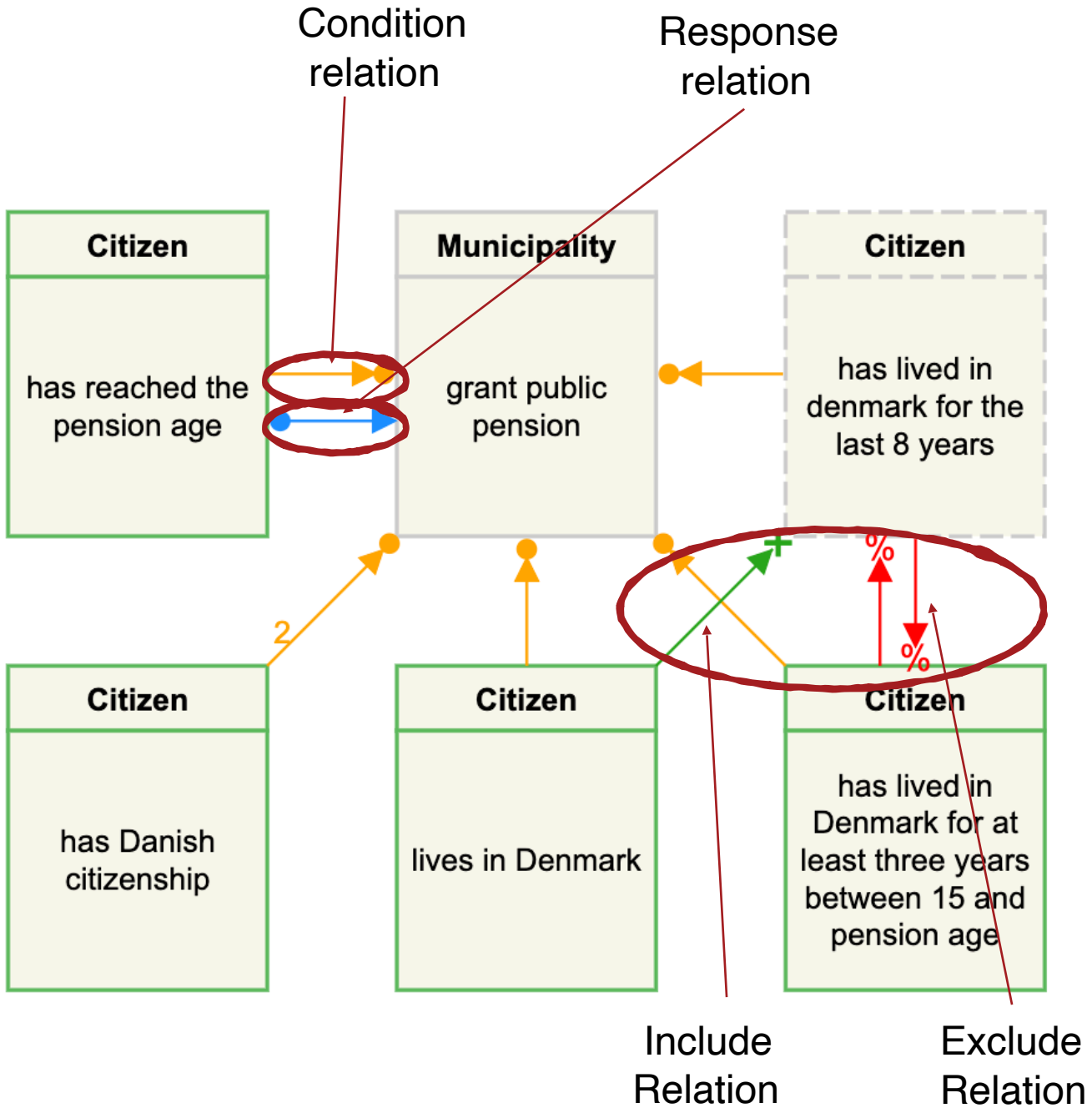
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0} \bullet f \mid e \bullet \xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

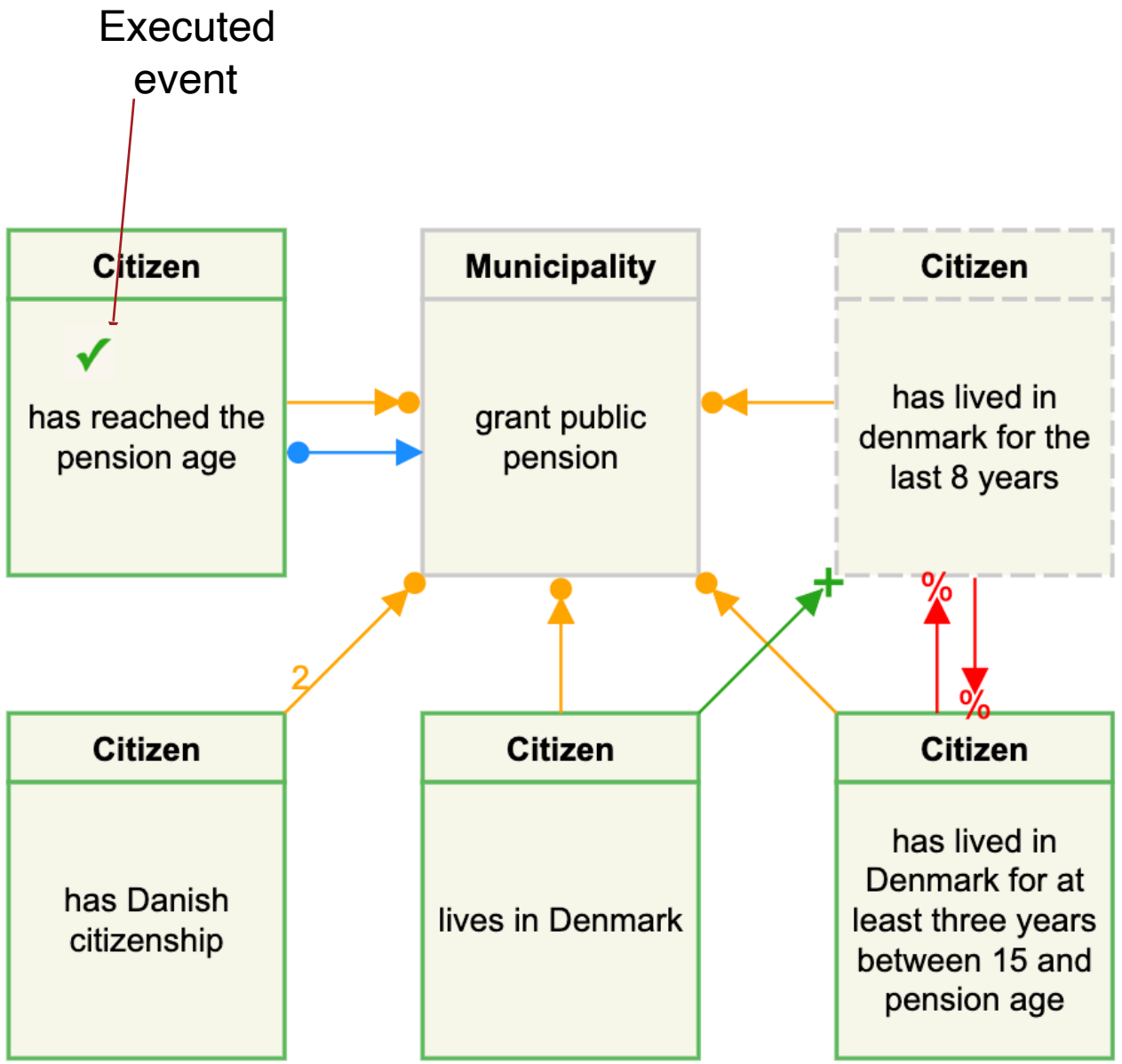
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad \text{0-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0}\bullet f \mid e\bullet\xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

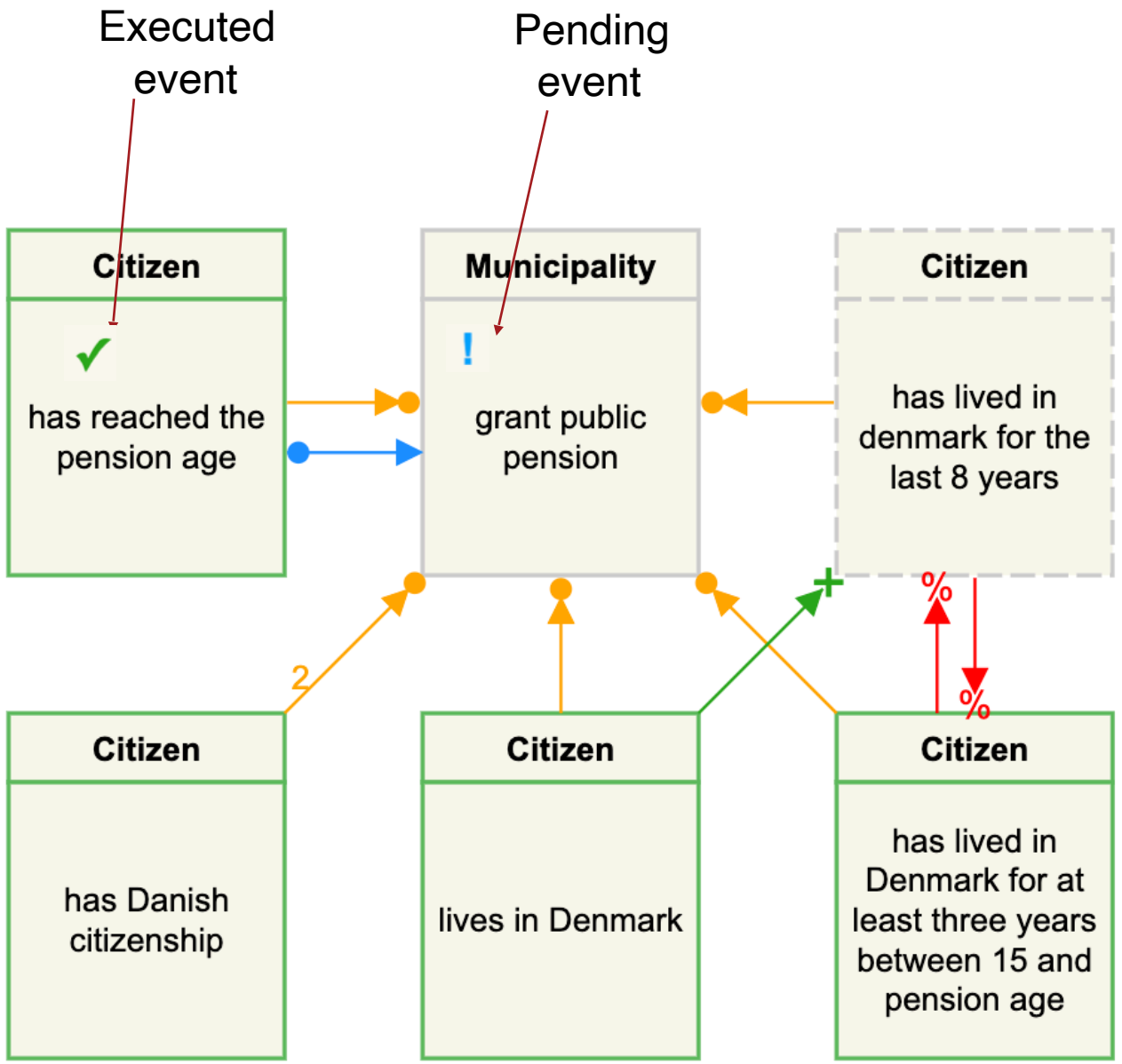
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Timed DCR graphs - briefly

- Assume:
  - A fixed universe of events $\mathcal{E}$ and labels $\mathcal{L}$
  - $e, f \in \mathcal{E}$
  - A mapping λ from events to labels
  - tick $\notin \mathcal{E}$
- The grammar for DCR process is given as:

$$P, Q ::= [M]\ \lambda T \qquad \text{process}$$

$$\begin{aligned} T, U ::= \ & e \xrightarrow{t_0}\bullet f \mid e\bullet\xrightarrow{t_\omega} f \\ & \mid e \rightarrow + f \mid e \rightarrow \% f \\ & \mid e \rightarrow \diamond f \mid T \parallel U \\ & \mid 0 \end{aligned}$$

$$M, N ::= M, e : \Phi \mid \epsilon \qquad \text{marking}$$

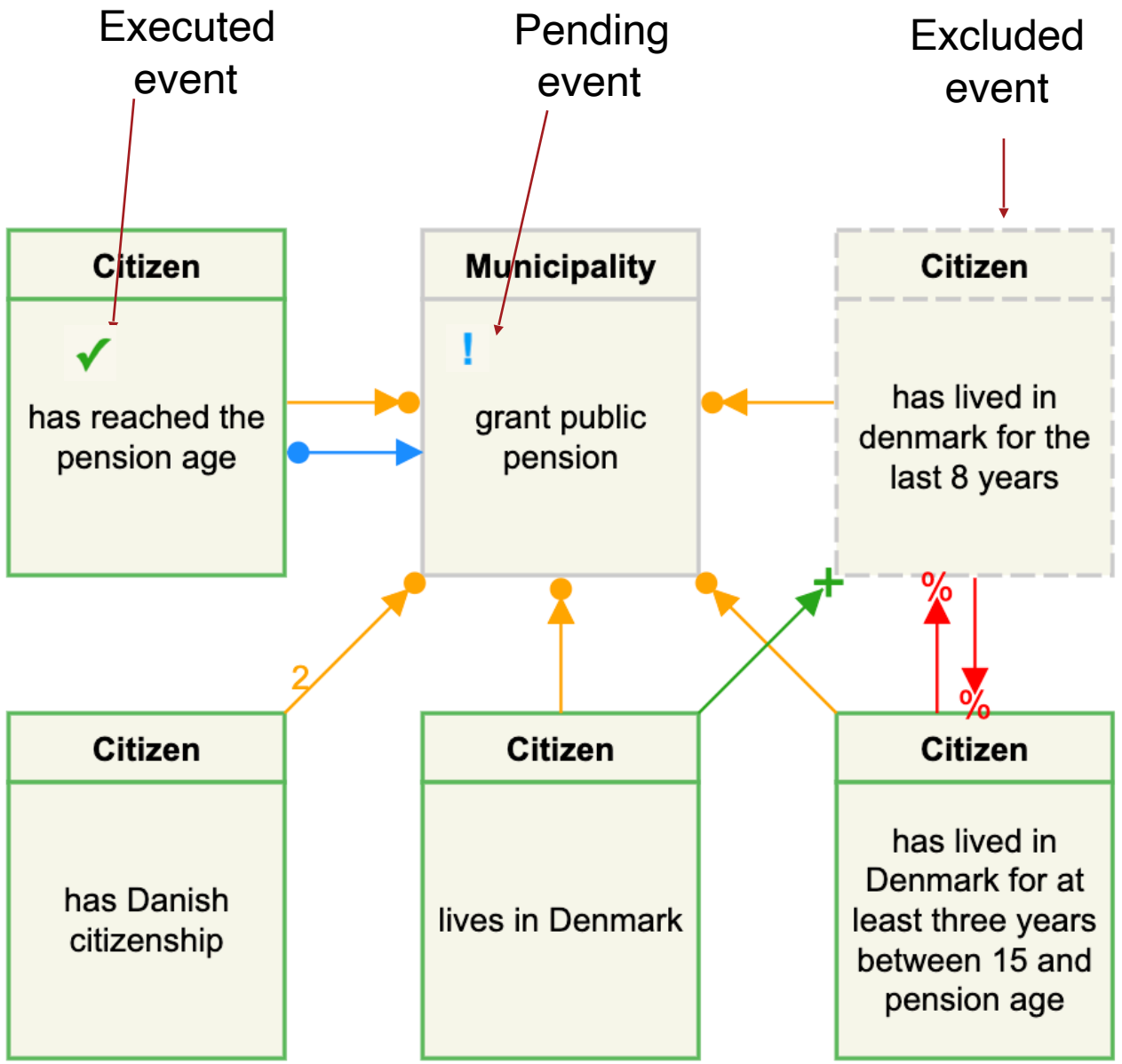
$$\lambda ::= \lambda, e : l \mid \epsilon \qquad \text{labelling}$$

$$\Phi ::= (h, i, p)$$

$$h ::= \mathsf{f} \mid t_0 \qquad t_0 \in \mathbb{N} \cup \{0\} \qquad 0\text{-time}$$

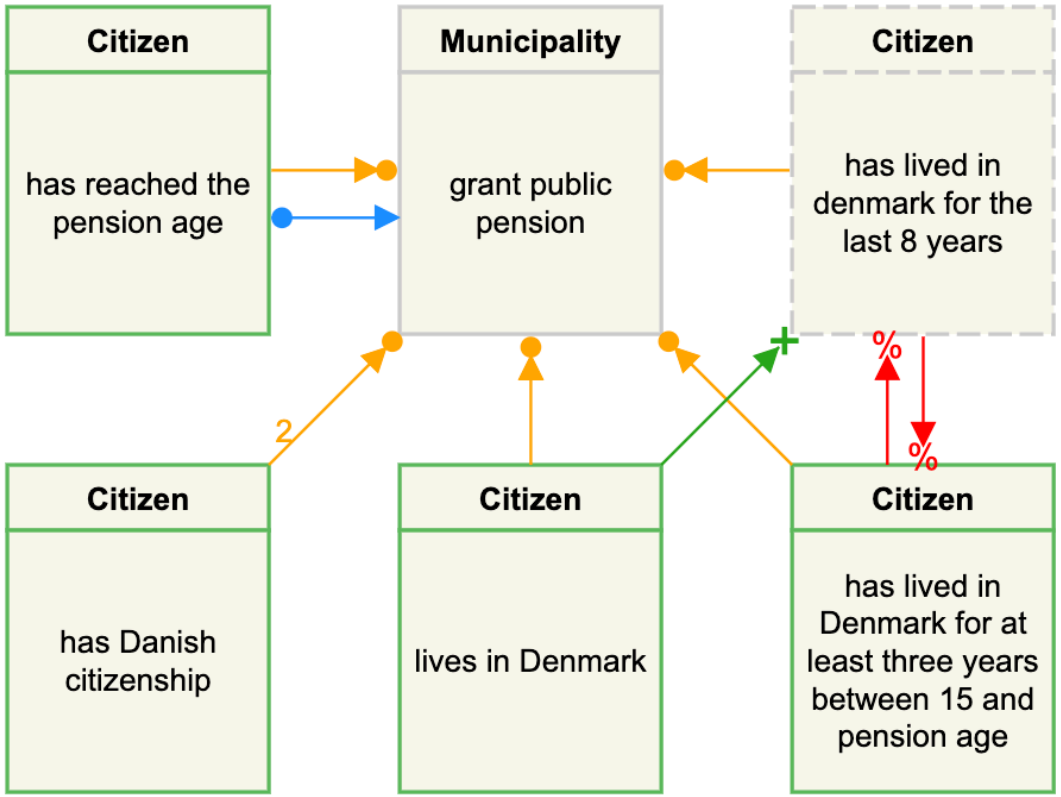
$$i ::= \mathsf{f} \mid \mathsf{t} \qquad t_\omega \in \mathbb{N} \cup \{\omega\} \qquad \omega\text{-time}$$

$$p ::= \mathsf{f} \mid 0 \mid t_\omega$$



Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathrm{f} : (E,I,P)$:

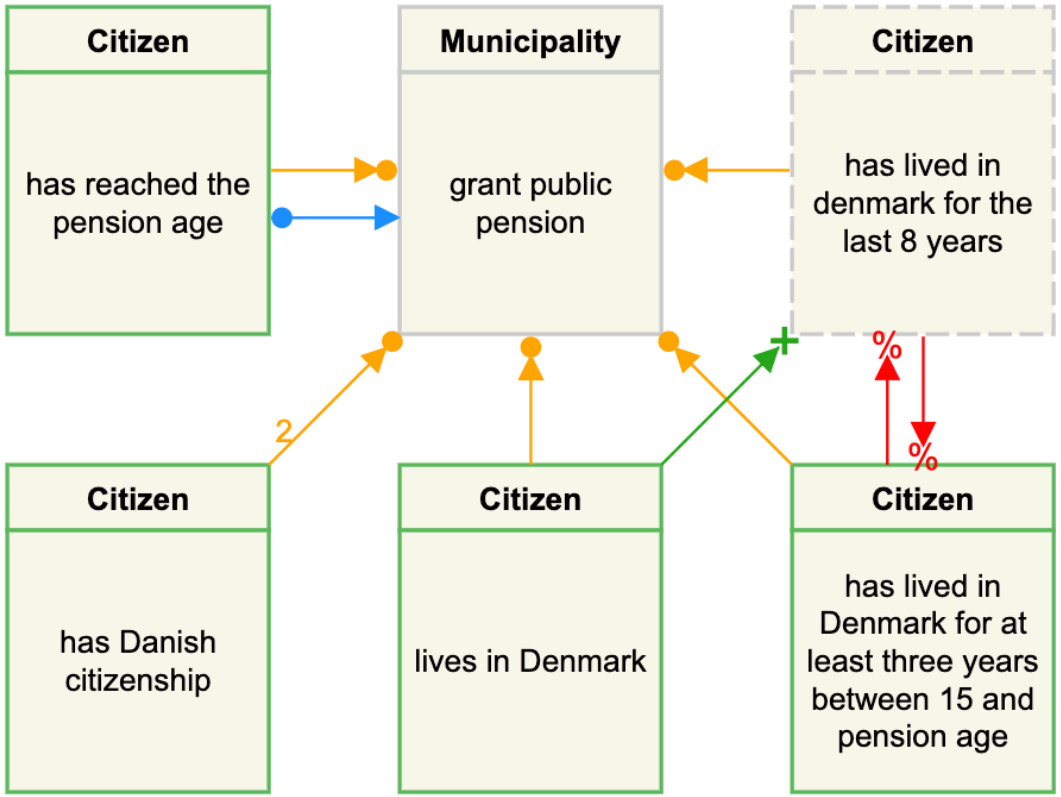


k=0

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E, I, P)$:

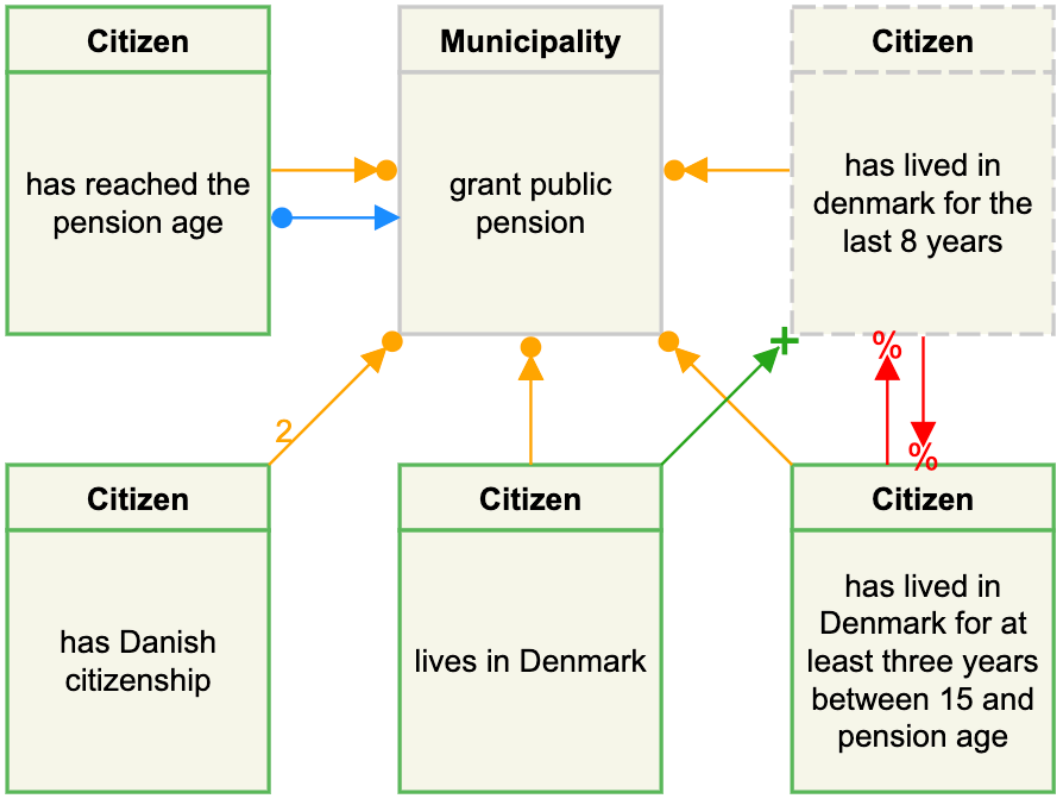
Marking
Term
Event
Excluded
Included
Pending



k=0

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E,I,P)$:



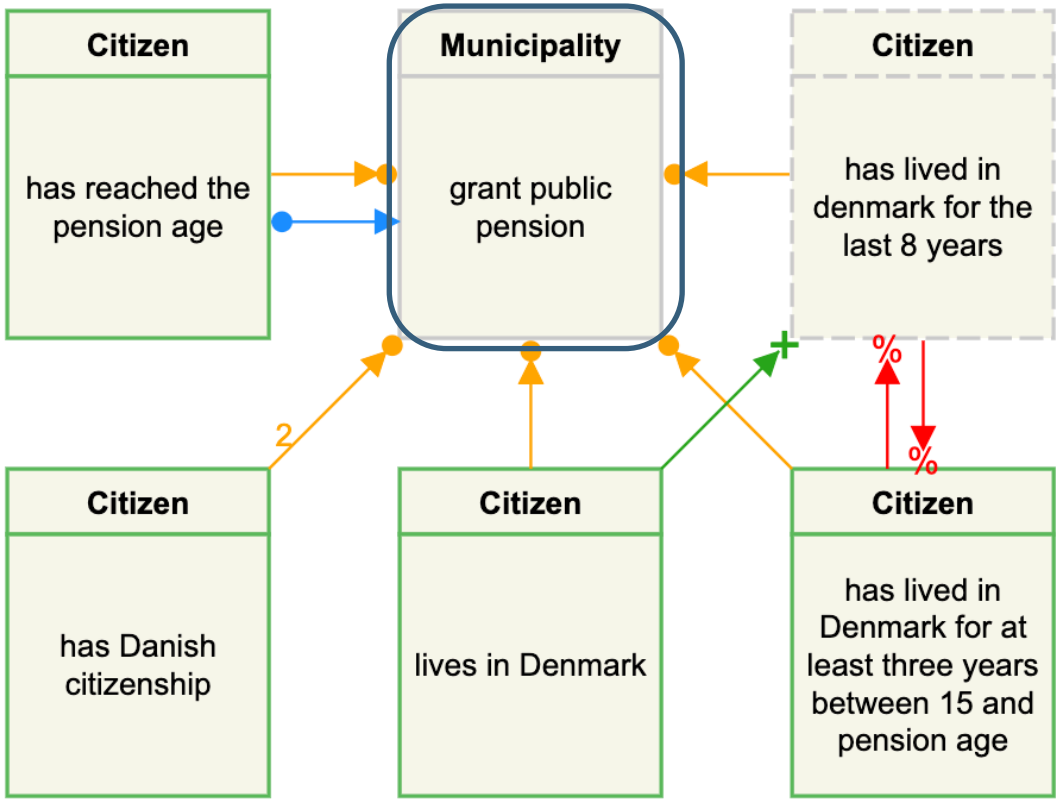
Event *f* is enabled iff

$$\frac{i \Rightarrow h \geq k}{[M, e : (h, i, \_), f : (\_, \mathtt{t}, \_)]\ e \xrightarrow{k} f \vdash f : (\emptyset, \emptyset, \emptyset)}$$

k=0

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E,I,P)$:



Event *f* is enabled iff

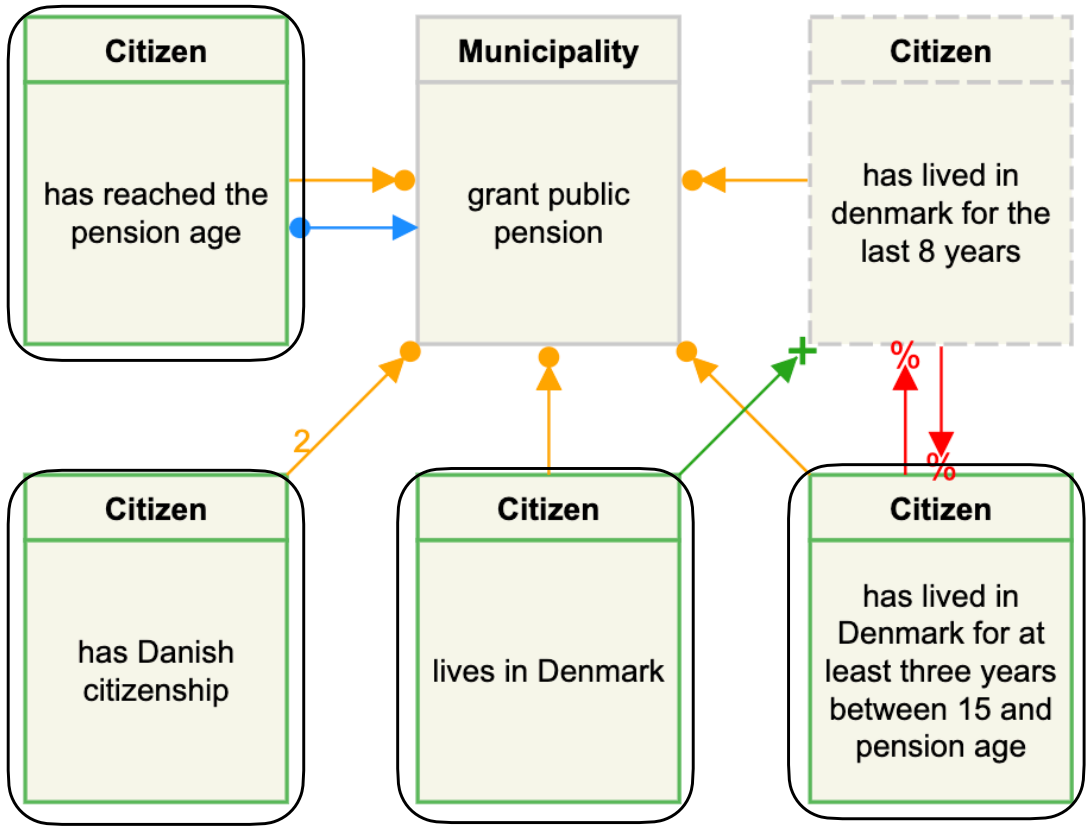
1. *f* is included

$$\frac{i \Rightarrow h \geq k}{[M, e : (h, i, \_), f : (\_, \mathsf{t}, \_)]\ e \xrightarrow{k} f \vdash f : (\emptyset, \emptyset, \emptyset)}$$

k=0

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E,I,P)$:



Event *f* is enabled iff

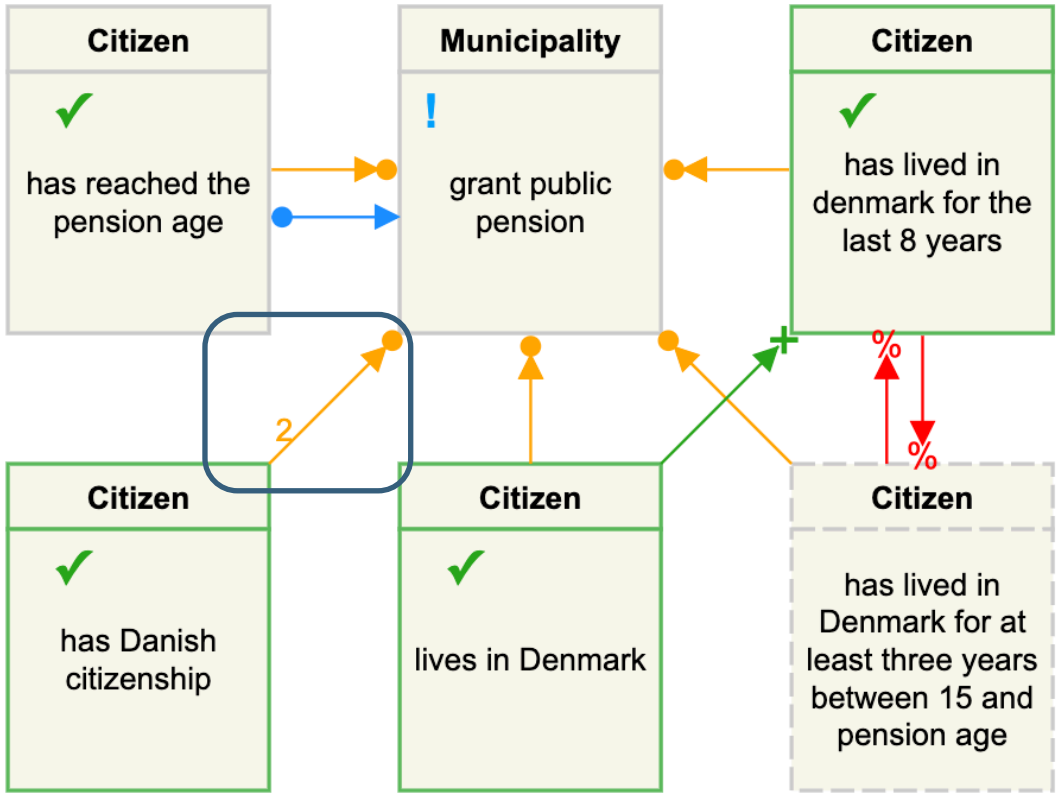
1. *f* is included
2. Its included preconditions have been executed or excluded

$$\frac{i \Rightarrow h \geq k}{[M, e : (h, i, \_), f : (\_, \mathsf{t}, \_)]\ e \xrightarrow{k} f \vdash f : (\emptyset, \emptyset, \emptyset)}$$

k=0

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E,I,P)$:



Event *f* is enabled iff

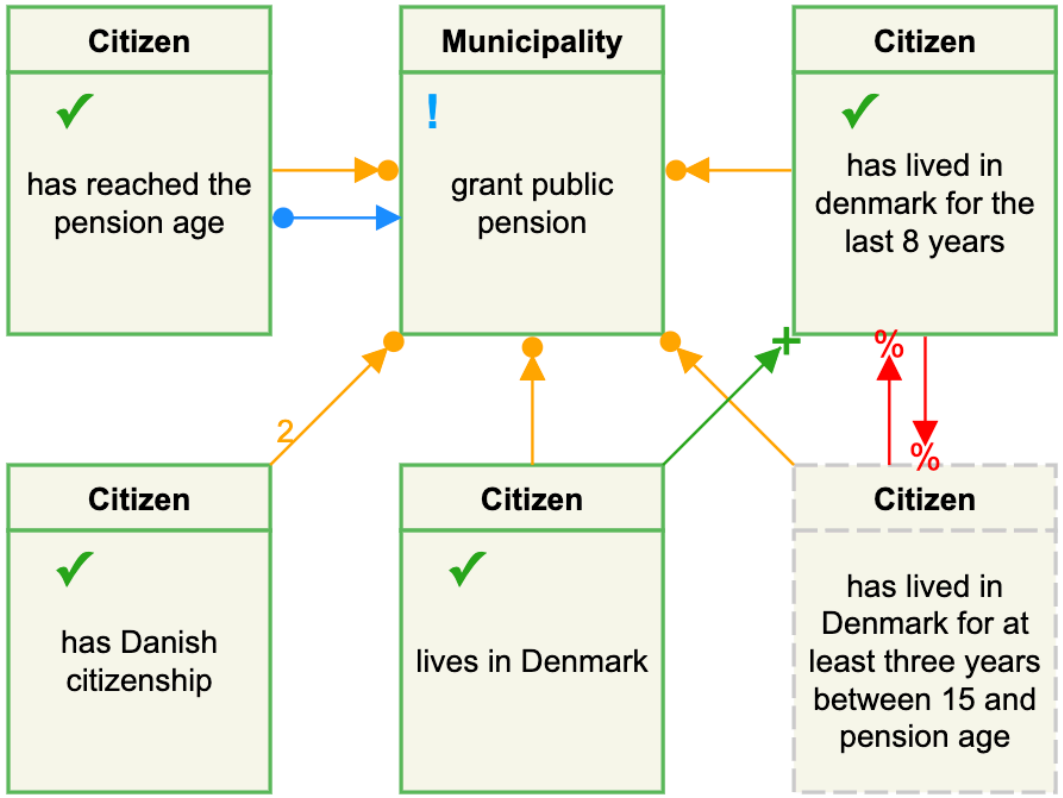
1. *f* is included
2. Its included preconditions have been executed or excluded
3. If it depends on a time condition, this has been executed at least *k* steps ago

$$\frac{i \Rightarrow h \geq k}{[M, e : (h, i, \_), f : (\_, \mathsf{t}, \_)]\ e \xrightarrow{k}\bullet f \vdash f : (\emptyset, \emptyset, \emptyset)}$$

k=0

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E,I,P)$:



Event *f* is enabled iff

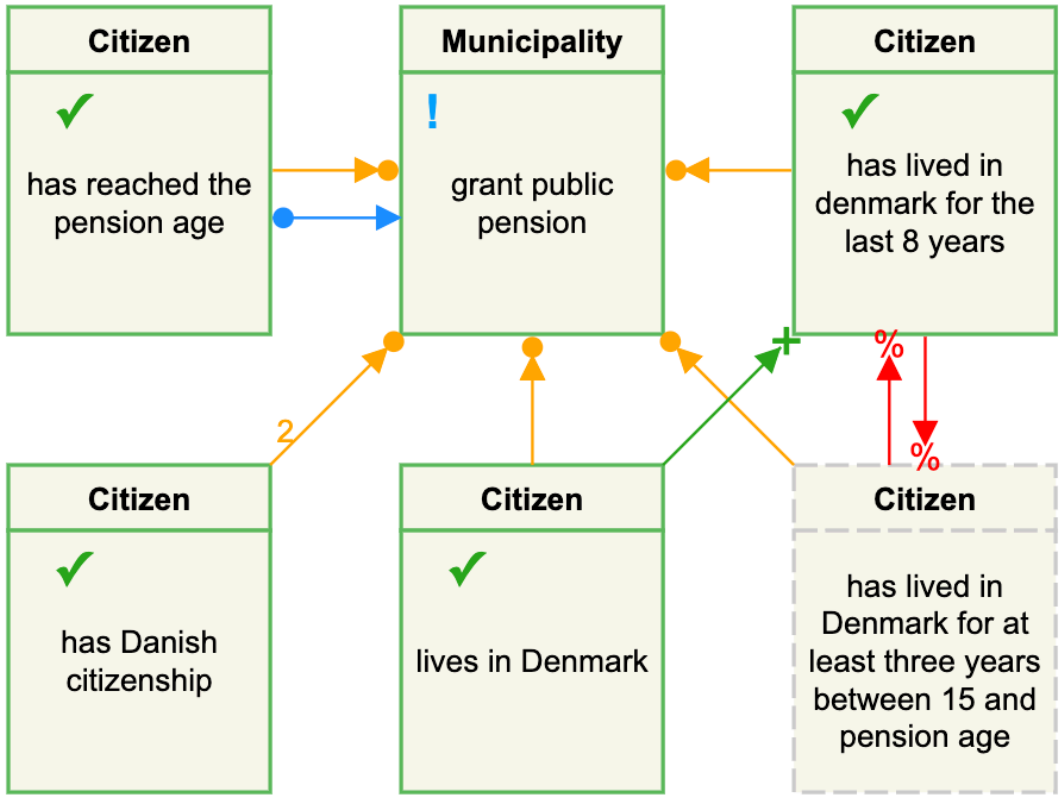
1. *f* is included
2. Its included preconditions have been executed or excluded
3. If it depends on a time condition, this has been executed at least *k* steps ago

$$\frac{i \Rightarrow h \geq k}{[M, e : (h, i, \_), f : (\_, \mathsf{t}, \_)]\ e \xrightarrow{k}\bullet f \vdash f : (\emptyset, \emptyset, \emptyset)}$$

k=2

Operational Semantics: Enabledness and Effects

- Enabled events & effects $[M]\ T \vdash \mathsf{f} : (E,I,P)$:



k=2

Event *f* is enabled iff

1. *f* is included
2. Its included preconditions have been executed or excluded
3. If it depends on a time condition, this has been executed at least *k* steps ago

$$\frac{i \Rightarrow h \geq k}{[M, e : (h, i, \_), f : (\_, \mathsf{t}, \_)]\ e \xrightarrow{k}\bullet f \vdash f : (\emptyset, \emptyset, \emptyset)}$$

$$\frac{}{[M, e : (\_, \mathsf{t}, \_)]\ e\bullet\xrightarrow{k} f \vdash e : (\emptyset, \emptyset, \{f : k\})}$$

Impossed obligations

Operational Semantics: Transitions

$$\frac{[M]\ T \vdash e : \delta}{T \vdash M \xrightarrow{e} \delta\langle e\langle M\rangle\rangle}\ [\textsc{Event}] \qquad \frac{\mathsf{deadline}\langle M\rangle > 0}{T \vdash M \xrightarrow{\mathsf{tick}} \mathsf{tick}\langle M\rangle}\ [\textsc{Time}]$$

- LTS keeps track of the sequence of fired events & time changes
- *e⟨M⟩* : effect of executing e in marking M
- *e: δ = (Ex, In, Pe)*: effects of executing e
- deadline⟨M⟩: modify pending/executed markings.

Soundness

Adapted from [Dumas et. al. Fundamentals of BPM. Chapter 5] to DCR graphs

A process model is **behaviourally correct**, or sound, if and only if it satisfies the following behavioural rules:

1. **Option to complete**: any running process instance must eventually complete,
2. **Proper completion:** at the moment of completion, all pending activities must have been executed, or excluded,
3. **No dead activities:** any activity can be executed in at least one process instance.

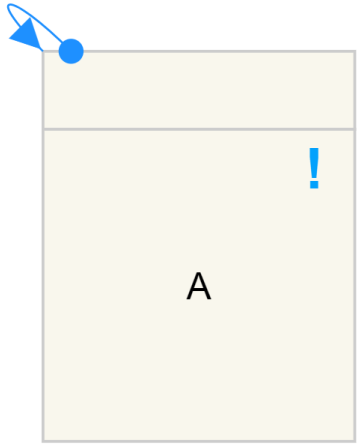
Soundness (I): Termination

- Recall the trace model for DCR graphs:
  - Any enabled activity can be executed
  - An activity can be executed any number of times, unless it is disabled
- A trace is **accepting**, when there are no pending activities to be executed
- Trace **length**: the number of activities executed so far

Trace	Length
<>	0
<Pick Item>	1
<Pick Item, Quit>	2
<Pick Item, Close Order, PayOrder>	3
<Pick Item, Pick Item, Close Order, Pay Order>	4

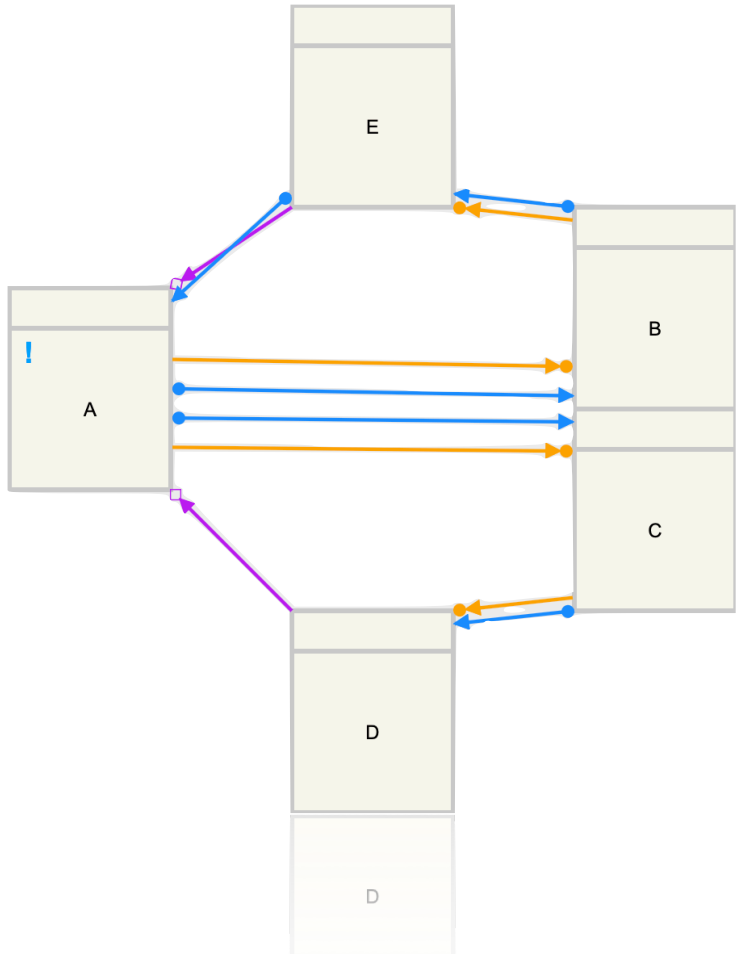
Termination

Do these models generate accepting traces?



Termination

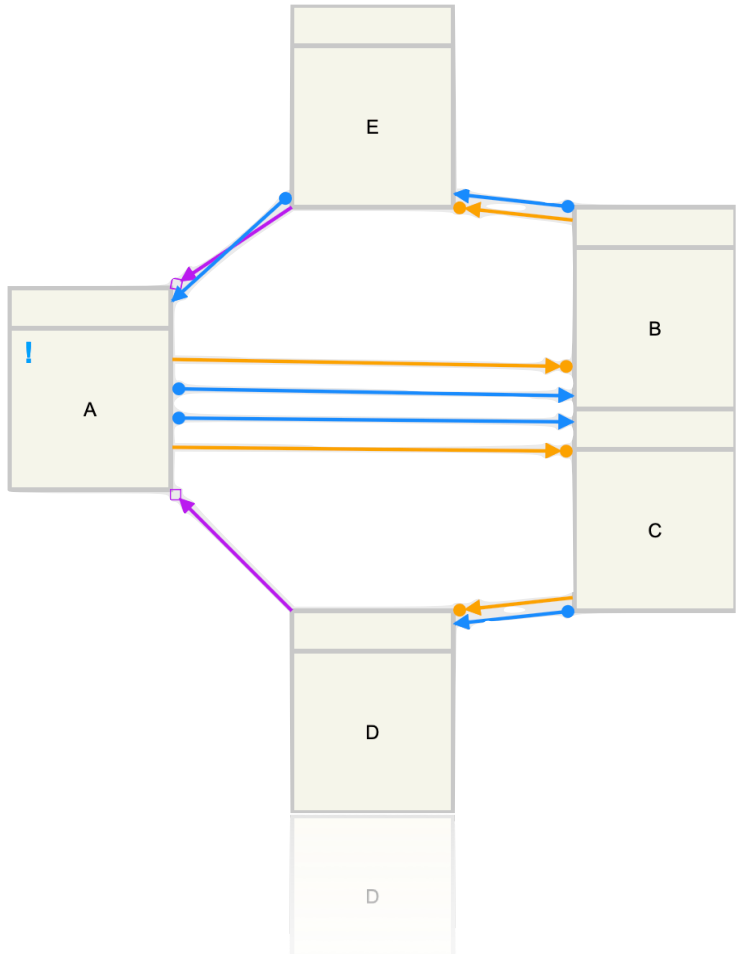
Do these models generate accepting traces?



Termination

Do these models generate accepting traces?

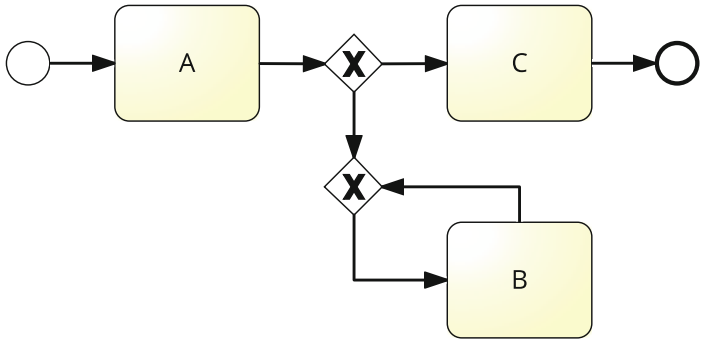
A process is **non-terminating** if for any trace of length N, the execution of an activity generates a non-accepting trace, for any N



Hint: check for circular response/ milestones dependencies in the graph

Exercise 1

- Consider the following BPMN model



- First, provide a model in DCR graphs that exhibits a behaviour similar to that one of the model, and
- Show that such a model is non-terminating

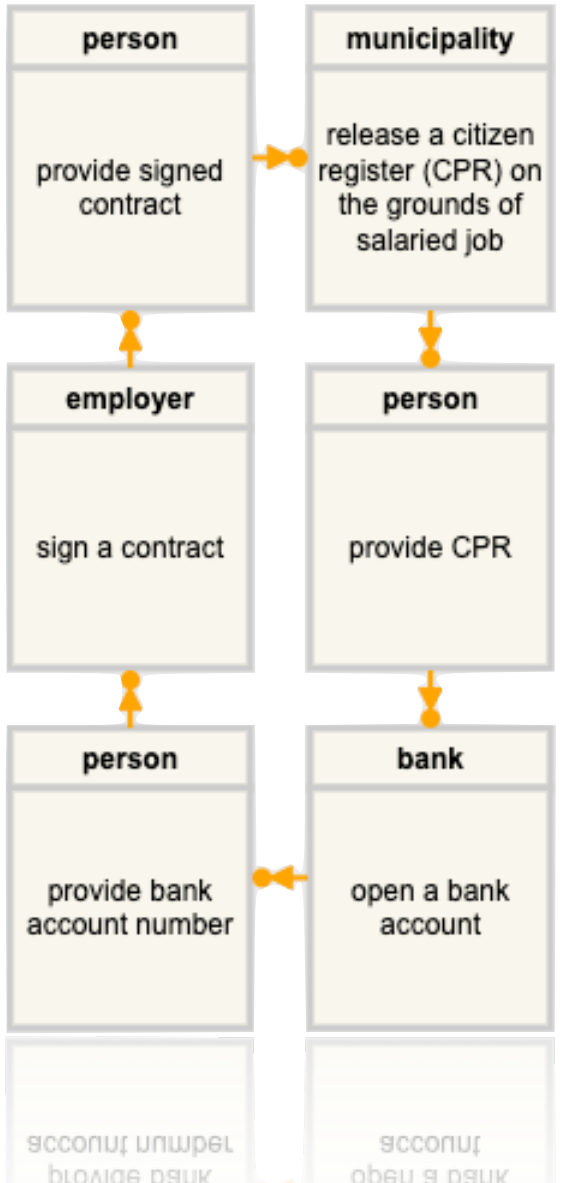
Soundness (II): Dead Activities!

- In order to release a citizen register (CPR) on the grounds of salaried job, the municipality needs the signed contract of the person
- In order to sign a contract, the employer needs to have the bank account of the person
- In order to open a bank account, the bank needs to get the CPR of the person

Soundness (II): Dead Activities!

- In order to release a citizen register (CPR) on the grounds of salaried job, the municipality needs the signed contract of the person
- In order to sign a contract, the employer needs to have the bank account of the person
- In order to open a bank account, the bank needs to get the CPR of the person

https://www.dcrgraphs.net/Tool?id=1017718#



Soundness (II): Dead Activities!

- In order to release a citizen register (CPR) on the grounds of salaried job, the municipality needs the signed contract of the person
- In order to sign a contract, the employer needs to have the bank account of the person
- In order to open a bank account, the bank needs to get the CPR of the person

https://www.dcrgraphs.net/Tool?id=1017718#

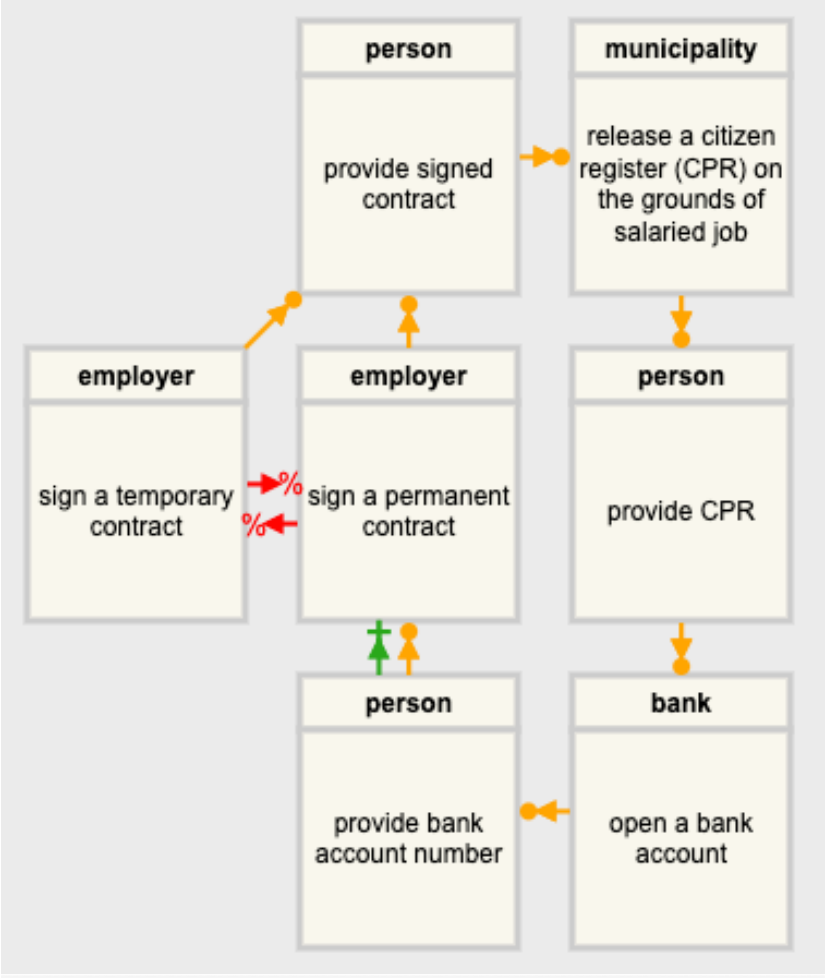
When would you execute this graph?

Null Process: Avoiding circular dependencies

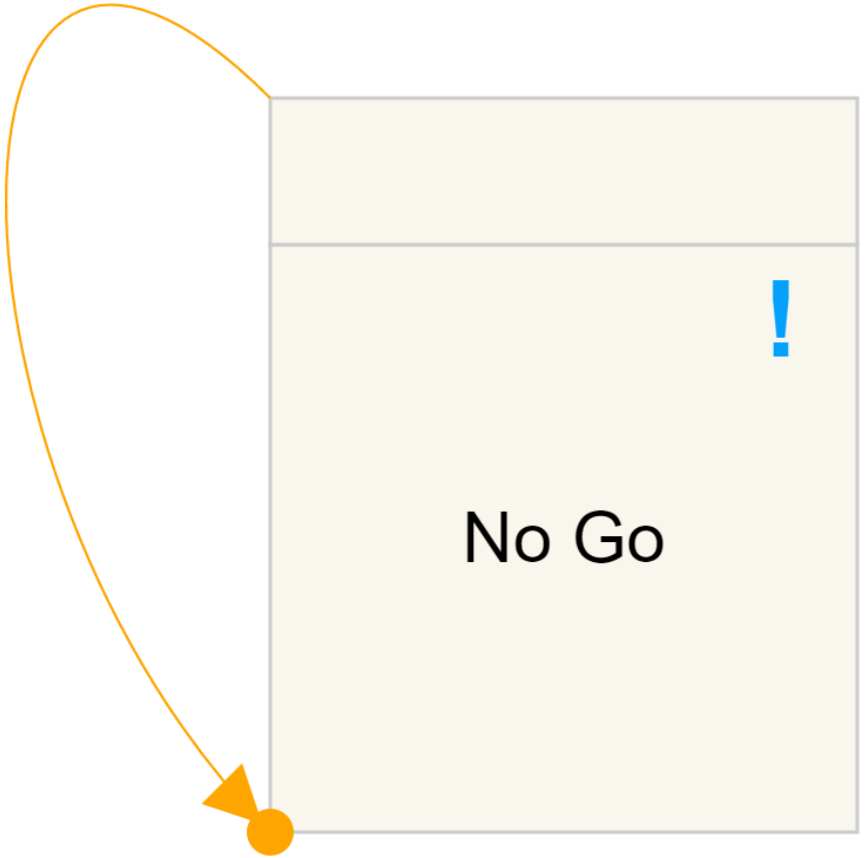
- Break circular dependencies by adding additional activities

Null Process: Avoiding circular dependencies

- Break circular dependencies by adding additional activities

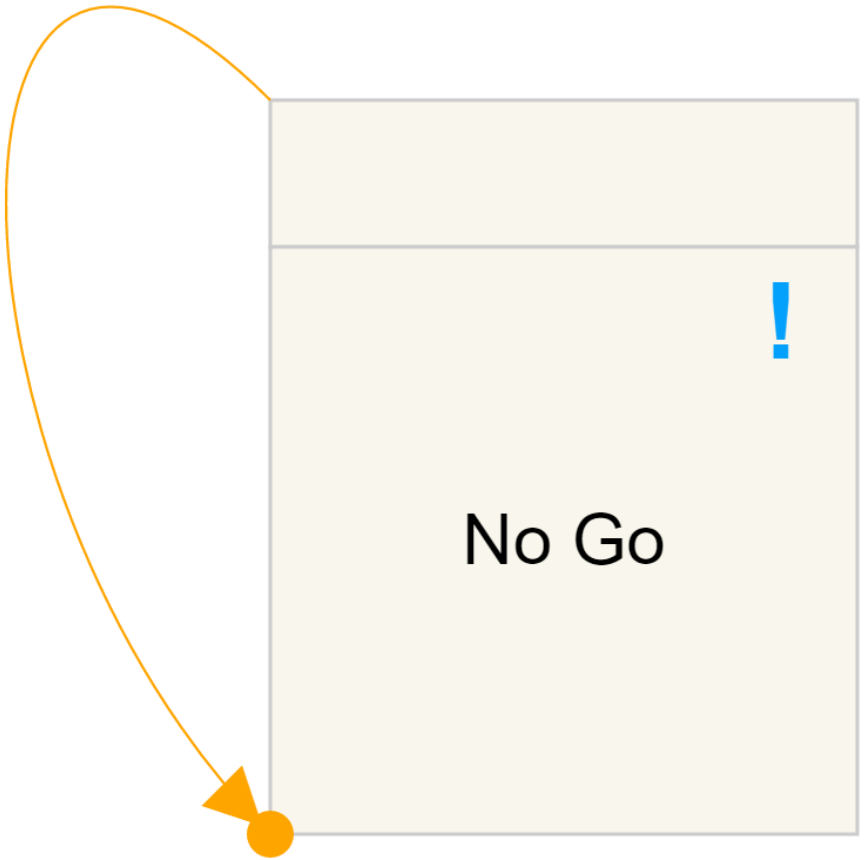


Deadlocks!



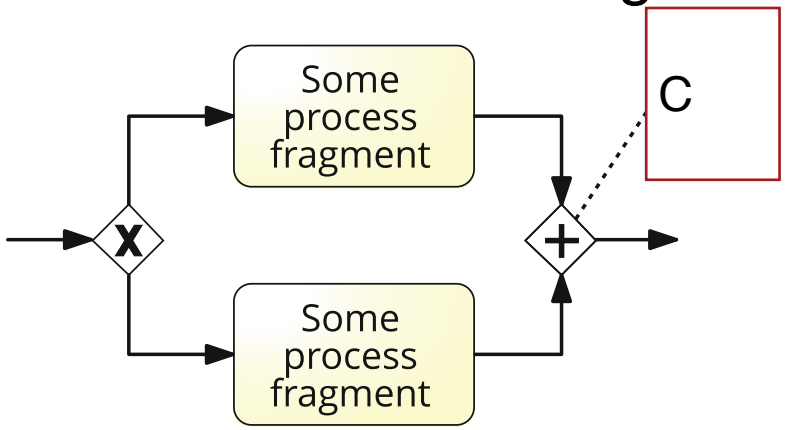
Deadlocks!

- There is a reachable state with no enabled activities, and there is at least one pending activity
- A DCR Graph is **deadlock free** if and only if for any execution, there is either an enabled event or no included required responses.



Exercise

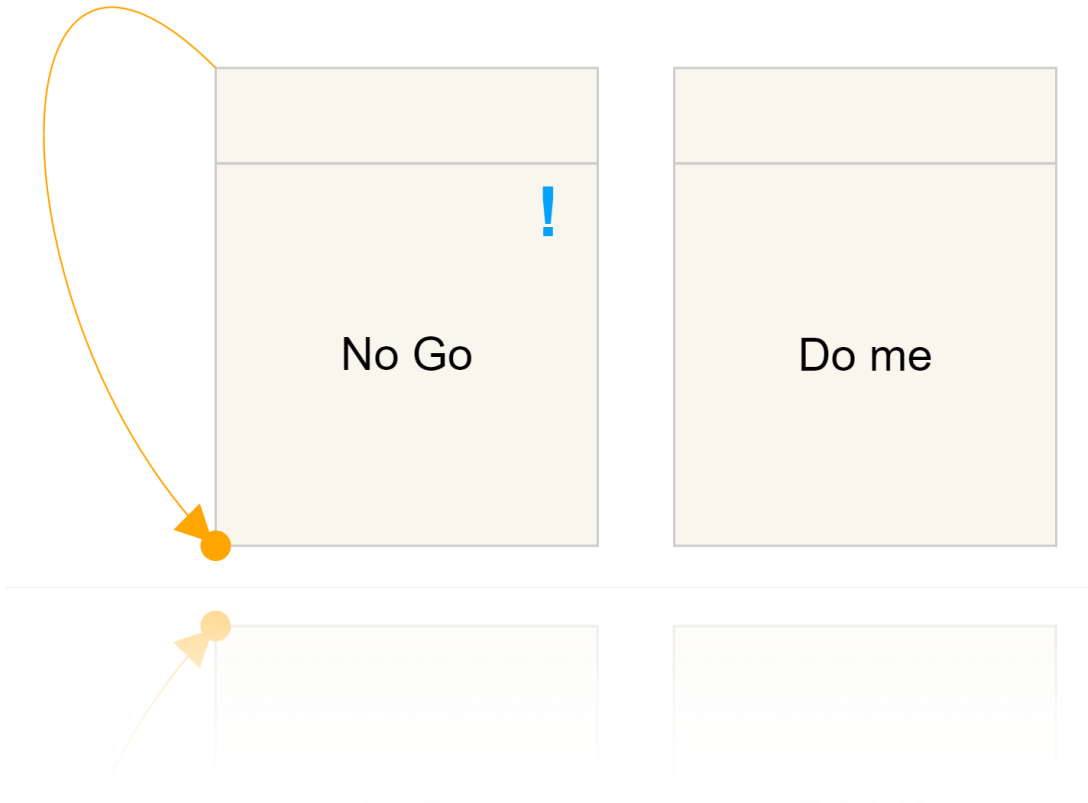
- Consider the following BPMN model



- Provide a DCR graph that describe the same set of traces as the model below
- Show that such a model is not deadlock free

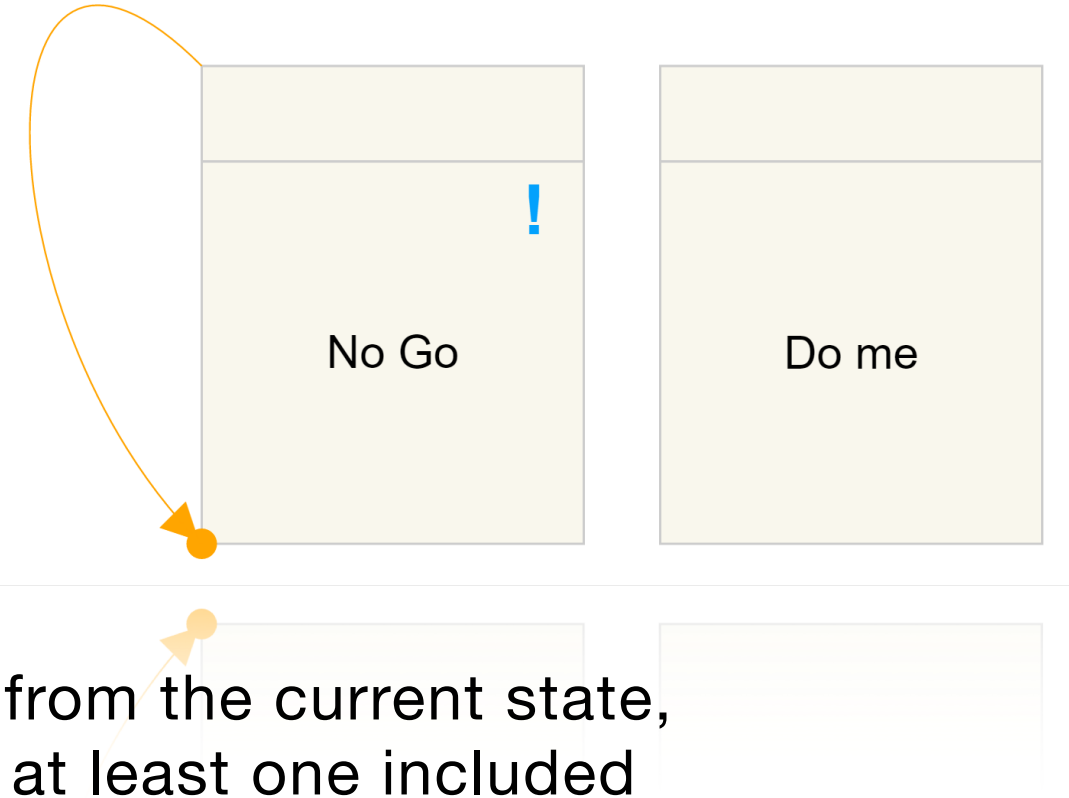
Liveness

- Deadlock-freedom guarantees that we will not get stuck, but it does not say that we will do what we require:



Liveness

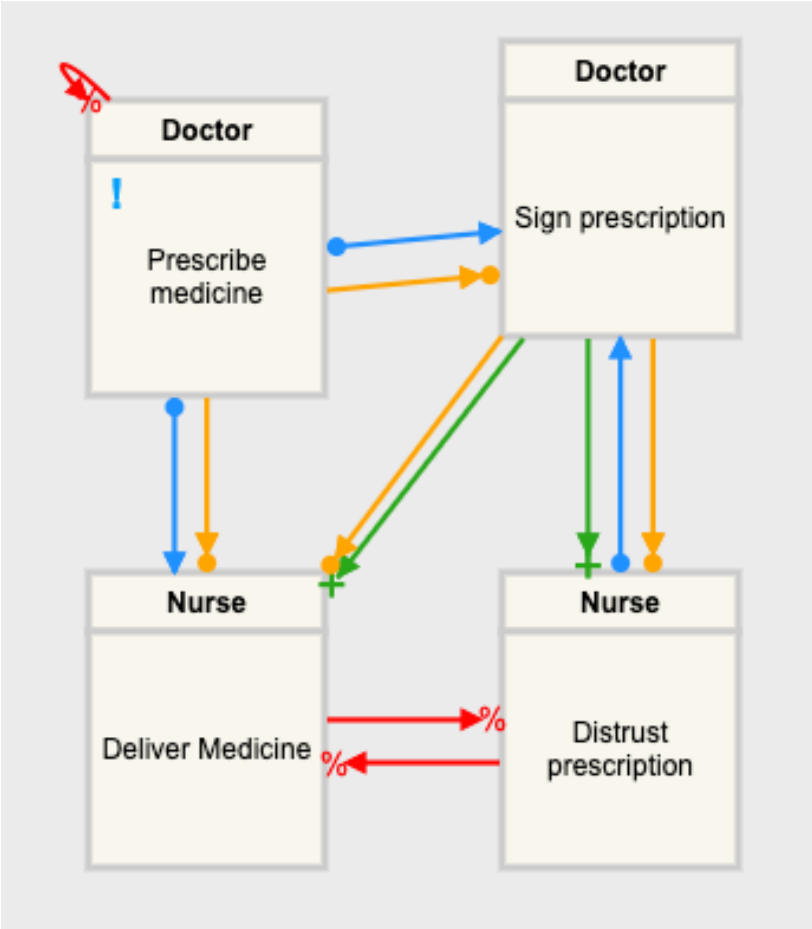
- Deadlock-freedom guarantees that we will not get stuck, but it does not say that we will do what we require:



- **Livelock**: For all states reachable from the current state, there exists an enabled event and at least one included pending event that never gets excluded or enabled

Liveness

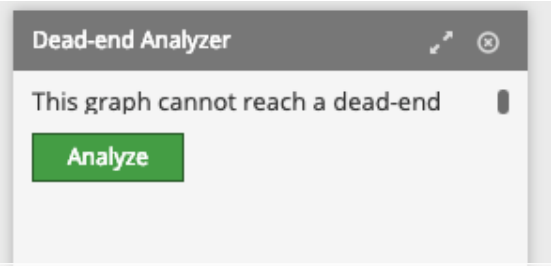
- A DCR Graph is **livelock free** if and only if, in every execution, it is always possible to continue along an accepting run (i.e. eventually execute or exclude any of the pending responses)

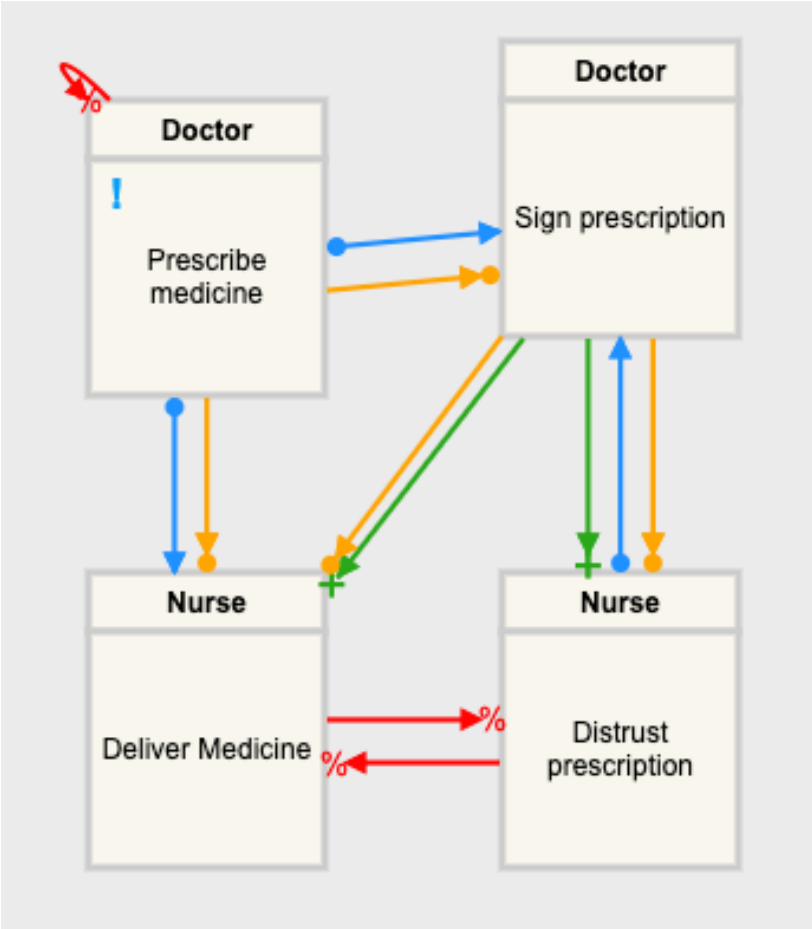


http://www.dcrgraphs.net/Tool?id=9681

Liveness

- A DCR Graph is **livelock free** if and only if, in every execution, it is always possible to continue along an accepting run (i.e. eventually execute or exclude any of the pending responses)

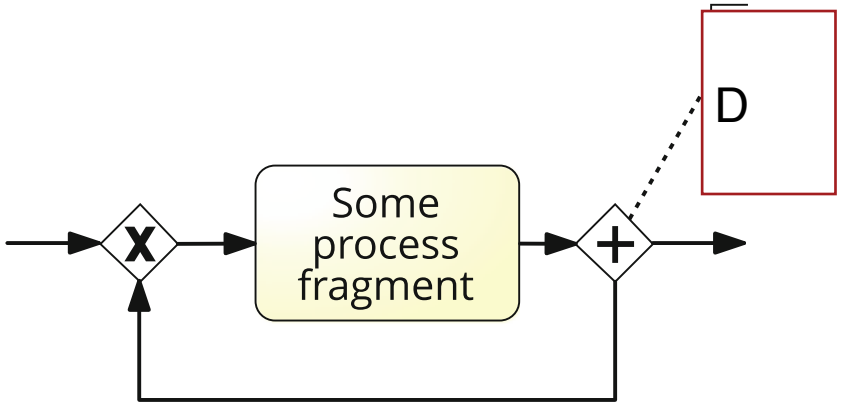




http://www.dcrgraphs.net/Tool?id=9681

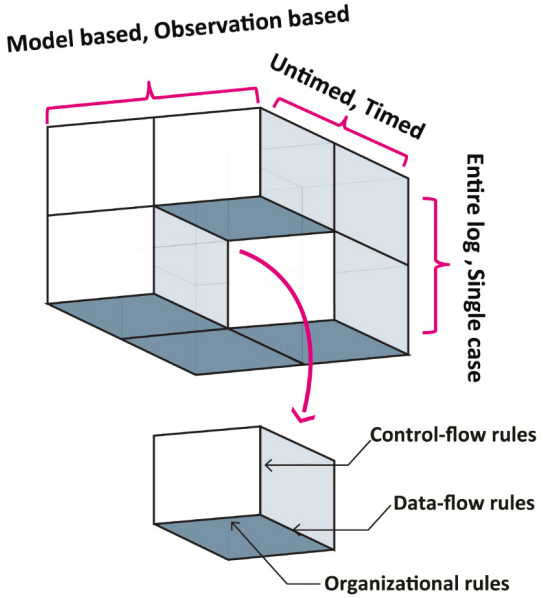
Exercise 3

- Take the following BPMN model:



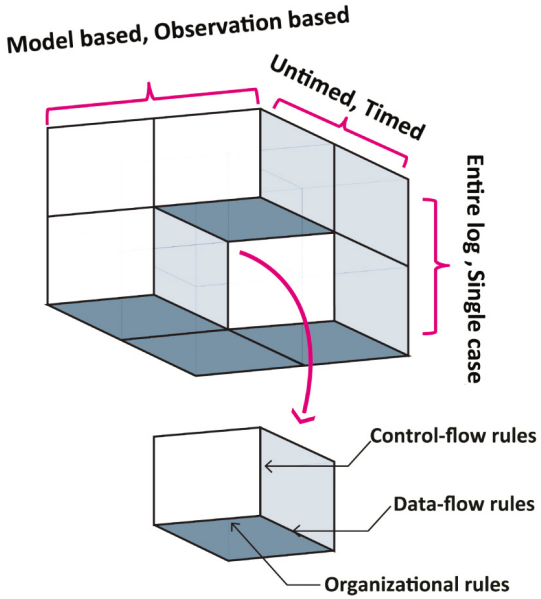
- First, generate a DCR model with equivalent behaviour to the model above
  - Then, show that such a model is in a livelock

Anatomy of a Compliance Rule



[1]
E. Ramezani, D. Fahland, and W. M. P. van der Aalst, "Where Did I Misbehave? Diagnostic Information in Compliance Checking," in *Business Process Management*, 2012, pp. 262–278.

Anatomy of a Compliance Rule



Category (Rules)	Description
Existence (2)	Limits the occurrence or absence of a given event A within a scope. [4],[15],[9], [14],[21],[36],[33]
Bounded Existence (6)	Limits the number of times a given event A must or must not occur within a scope. [15],[14]
Bounded Sequence (5)	Limits the number of times a given sequence of events must or must not occur within a scope. [15],[14]
Parallel (1)	A specific set of events should occur in parallel within a scope. [33]
Precedence (10)	Limits the occurrence of a given event A in precedence over a given event B. [15],[33],[14],[36],[9],[19],[21],[4],[33]
Chain Precedence (4)	Limits the occurrence of a sequence of events $A_1, \ldots, A_n$ over a sequence of events $B_1, \ldots, B_n$. [15],[14],[21]
Response (10)	Limits the occurrence of a given event B in response to a given event A. [33],[14],[21],[15],[37],[9],[19]
Chain Response (4)	Limits the occurrence of a sequence of events $B_1, \ldots, B_n$ in response to a sequence of events $A_1, \ldots, A_n$. [15]
Between (7)	Limits the occurrence of a given event B between a sequence of events A and C. [14]
Exclusive (1)	Presence of a given event A mandates the absence of an event B. [15]
Mutual Exclusive (1)	Either a given event A or event B must exist but not none of them or both. [15],[34]
Inclusive (1)	Presence of a given event A mandates that event B is also present. [15]
Prerequisite (1)	Absence of a given event A mandates that event B is also absent. [15]
Substitute (1)	A given event B substitutes the absence of event A. [15]
Corequisite (1)	Either given events A and B should exist together or to be absent together. [15]

[1]
E. Ramezani, D. Fahland, and W. M. P. van der Aalst, "Where Did I Misbehave? Diagnostic Information in Compliance Checking," in *Business Process Management*, 2012, pp. 262–278.

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false
| a | b | c | …
| ~ Φ
| (Φ)
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …
| **X** Φ
| **F** Φ
| **G** Φ
| Φ **U** Φ’
| Φ **W** Φ’

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false
| a | b | c | … a = atomic proposition, activities in the process
| ~ Φ
| (Φ)
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …
| **X** Φ
| **F** Φ
| **G** Φ
| Φ **U** Φ’
| Φ **W** Φ’

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false
| a | b | c | … a = atomic proposition, activities in the process
| ~ Φ Negation
| (Φ)
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …
| **X** Φ
| **F** Φ
| **G** Φ
| Φ **U** Φ’
| Φ **W** Φ’

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false
| a | b | c | … a = atomic proposition, activities in the process
| ~ Φ Negation
| (Φ) subformula
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …
| **X** Φ
| **F** Φ
| **G** Φ
| Φ **U** Φ’
| Φ **W** Φ’

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false	
| a | b | c | …	a = atomic proposition, activities in the process
| ~ Φ	Negation
| (Φ)	subformula
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …	Logical conjunction, disjunction, consequence
| **X** Φ	
| **F** Φ	
| **G** Φ	
| Φ **U** Φ’	
| Φ **W** Φ’	

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false	
| a | b | c | …	a = atomic proposition, activities in the process
| ~ Φ	Negation
| (Φ)	subformula
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …	Logical conjunction, disjunction, consequence
| **X** Φ	“next”: ϕ is true at next step
| **F** Φ	
| **G** Φ	
| Φ **U** Φ’	
| Φ **W** Φ’	

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false	
| a | b | c | …	a = atomic proposition, activities in the process
| ~ Φ	Negation
| (Φ)	subformula
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …	Logical conjunction, disjunction, consequence
| **X** Φ	“next”: ϕ is true at next step
| **F** Φ	“Eventually”: ϕ will become true at some point in the future
| **G** Φ	
| Φ **U** Φ’	
| Φ **W** Φ’	

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false	
| a | b | c | …	a = atomic proposition, activities in the process
| ~ Φ	Negation
| (Φ)	subformula
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …	Logical conjunction, disjunction, consequence
| **X** Φ	“next”: ϕ is true at next step
| **F** Φ	“Eventually”: ϕ will become true at some point in the future
| **G** Φ	“Globally”: ϕ is true in all the traces.
| Φ **U** Φ’	
| Φ **W** Φ’	

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

“Temporal” refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ’ ::= true | false	
| a | b | c | …	a = atomic proposition, activities in the process
| ~ Φ	Negation
| (Φ)	subformula
| Φ ∧ Φ’ | Φ ∨ Φ’ | Φ → Φ’ | …	Logical conjunction, disjunction, consequence
| **X** Φ	“next”: ϕ is true at next step
| **F** Φ	“Eventually”: ϕ will become true at some point in the future
| **G** Φ	“Globally”: ϕ is true in all the traces.
| Φ **U** Φ’	“until”: ϕ’ is true at some point, ϕ is true until that time
| Φ **W** Φ’	

Linear Temporal Logic (LTL)

Language for describing compliance rules for BPMN processes.

"Temporal" refers to the underlying nature of time: each moment in time has a well-defined successor moment

- Properties are described as formulae in LTL. A formula ϕ can be constructed by the grammar:

Φ, Φ' ::= true | false	
| a | b | c | …	a = atomic proposition, activities in the process
| ~ Φ	Negation
| (Φ)	subformula
| Φ ∧ Φ' | Φ ∨ Φ' | Φ → Φ' | …	Logical conjunction, disjunction, consequence
| **X** Φ	"next": ϕ is true at next step
| **F** Φ	"Eventually": ϕ will become true at some point in the future
| **G** Φ	"Globally": ϕ is true in all the traces.
| Φ **U** Φ'	"until": ϕ' is true at some point, ϕ is true until that time
| Φ **W** Φ'	"weak until": ϕ' is true at some point, ϕ is true until that time, but if ϕ' never happens, ϕ should hold forever

Compliance rules in LTL

Compliance rules in LTL

1. **Direct Precedence of a Task**. “Every time B occurs, it should be directly preceded by A.”

$$(\neg \text{Perform\_surgery}\ W\ \text{Prepare\_patient}) \wedge (\neg \text{Perform\_surgery}\ W\ \text{Send\_patient\_to\_surgical\_suite})$$

Compliance rules in LTL

1. **Direct Precedence of a Task**. “Every time B occurs, it should be directly preceded by A.”

   $(\neg \text{Perform\_surgery}\ W\ \text{Prepare\_patient}) \wedge (\neg \text{Perform\_surgery}\ W\ \text{Send\_patient\_to\_surgical\_suite})$

2. **Direct Precedence or Simultaneous Occurrences of Tasks.** “Task *A* must always be executed simultaneously or directly before task *B*.”

   $(\text{Prepare\_patient} \wedge \text{Send\_patient\_to\_surgical\_suite}) \vee (\text{Prepare\_patient} \wedge X(\text{Send\_patient\_to\_surgical\_suite}))$

Compliance rules in LTL

1. **Direct Precedence of a Task**. "Every time B occurs, it should be directly preceded by A."

   $(\neg \text{Perform\_surgery}\ W\ \text{Prepare\_patient}) \wedge (\neg \text{Perform\_surgery}\ W\ \text{Send\_patient\_to\_surgical\_suite})$

2. **Direct Precedence or Simultaneous Occurrences of Tasks.** "Task *A* must always be executed simultaneously or directly before task *B*."

   $(\text{Prepare\_patient} \wedge \text{Send\_patient\_to\_surgical\_suite}) \vee (\text{Prepare\_patient} \wedge X(\text{Send\_patient\_to\_surgical\_suite}))$

3. **Execution in Between.** "Task B should be performed not before task A has been executed, and not later than C."

Compliance rules in LTL

1. **Direct Precedence of a Task**. "Every time B occurs, it should be directly preceded by A."

   $(\neg \text{Perform\_surgery}\ W\ \text{Prepare\_patient}) \wedge (\neg \text{Perform\_surgery}\ W\ \text{Send\_patient\_to\_surgical\_suite})$

2. **Direct Precedence or Simultaneous Occurrences of Tasks.** "Task *A* must always be executed simultaneously or directly before task *B*."

   $(\text{Prepare\_patient} \wedge \text{Send\_patient\_to\_surgical\_suite}) \vee (\text{Prepare\_patient} \wedge X(\text{Send\_patient\_to\_surgical\_suite}))$

3. **Execution in Between.** "Task B should be performed not before task A has been executed, and not later than C."

- Exercise.

Compliance rules in LTL

1. **Direct Precedence of a Task**. "Every time B occurs, it should be directly preceded by A."

   $(\neg \text{Perform\_surgery}\ W\ \text{Prepare\_patient}) \wedge (\neg \text{Perform\_surgery}\ W\ \text{Send\_patient\_to\_surgical\_suite})$

2. **Direct Precedence or Simultaneous Occurrences of Tasks.** "Task *A* must always be executed simultaneously or directly before task *B*."

   $(\text{Prepare\_patient} \wedge \text{Send\_patient\_to\_surgical\_suite}) \vee (\text{Prepare\_patient} \wedge X(\text{Send\_patient\_to\_surgical\_suite}))$

3. **Execution in Between.** "Task B should be performed not before task A has been executed, and not later than C."

- Exercise.

4. **Mutual Exclusion.** "Either tasks A and B must be executed, but no none of them, or both".

Compliance rules in LTL

1. **Direct Precedence of a Task**. "Every time B occurs, it should be directly preceded by A."

   $(\neg \text{Perform\_surgery}\ W\ \text{Prepare\_patient}) \wedge (\neg \text{Perform\_surgery}\ W\ \text{Send\_patient\_to\_surgical\_suite})$

2. **Direct Precedence or Simultaneous Occurrences of Tasks.** "Task *A* must always be executed simultaneously or directly before task *B*."

   $(\text{Prepare\_patient} \wedge \text{Send\_patient\_to\_surgical\_suite}) \vee (\text{Prepare\_patient} \wedge X(\text{Send\_patient\_to\_surgical\_suite}))$

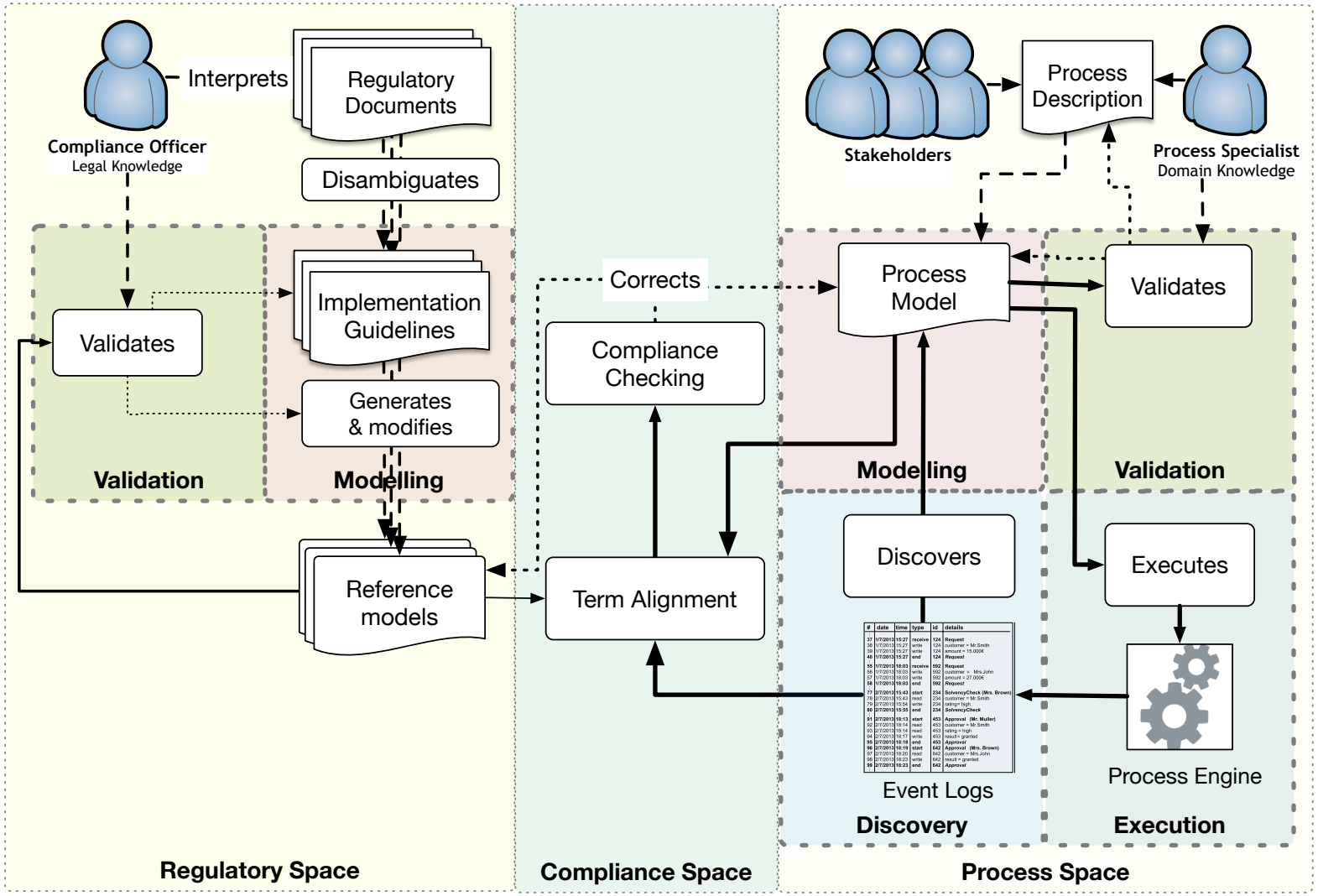
3. **Execution in Between.** "Task B should be performed not before task A has been executed, and not later than C."

- Exercise.

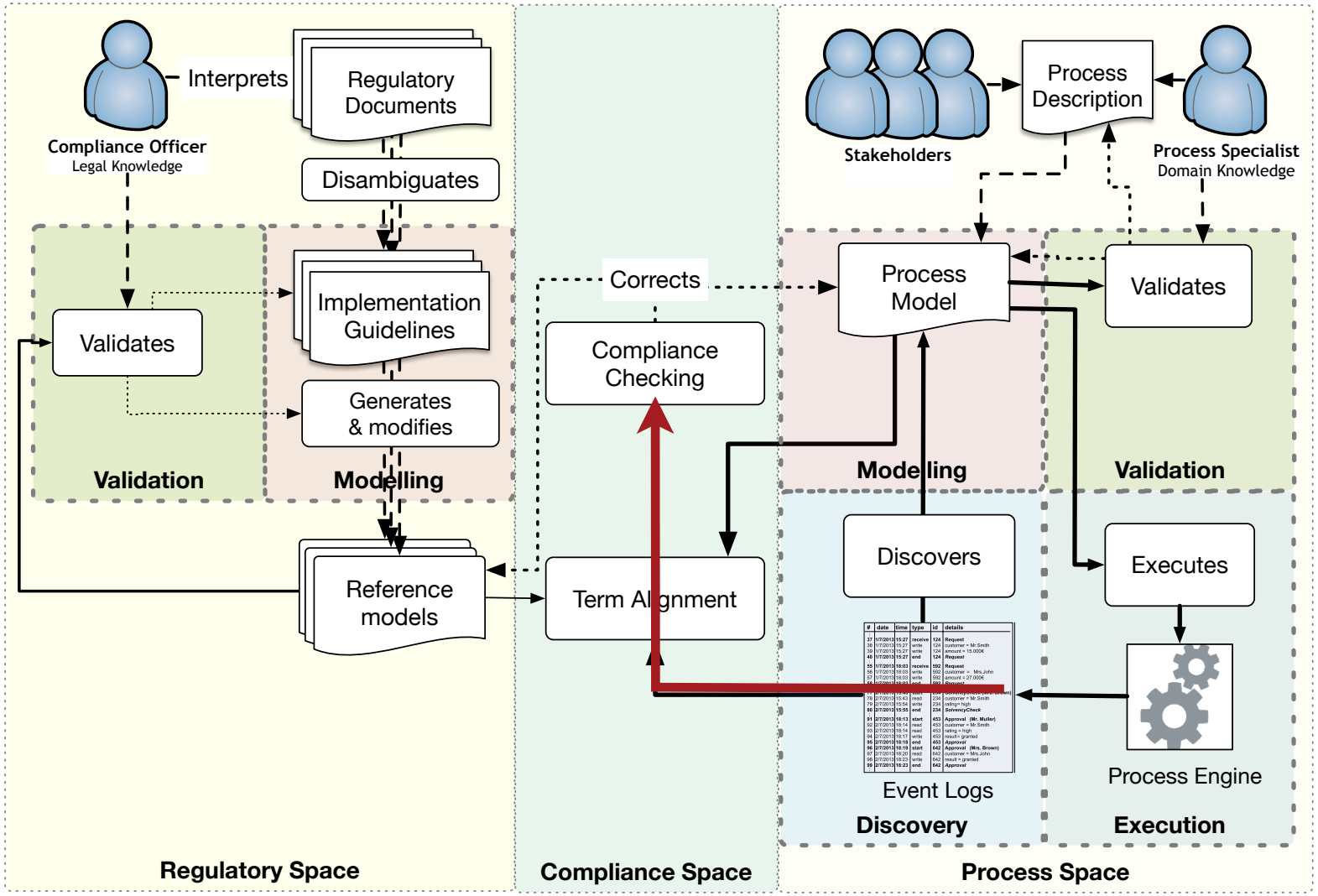
4. **Mutual Exclusion.** "Either tasks A and B must be executed, but no none of them, or both".

- Exercise.

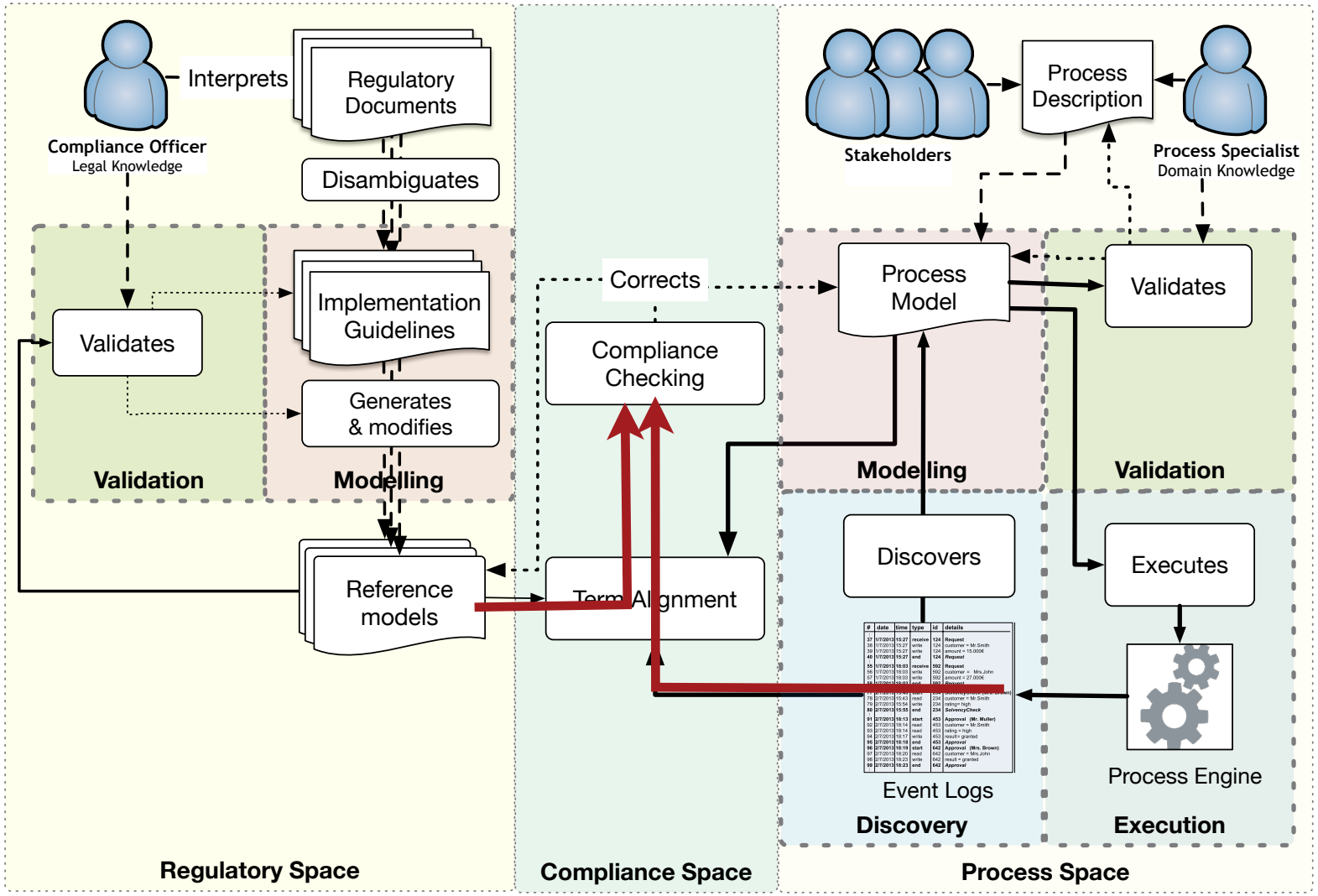
2. Compliance as Conformance Checking



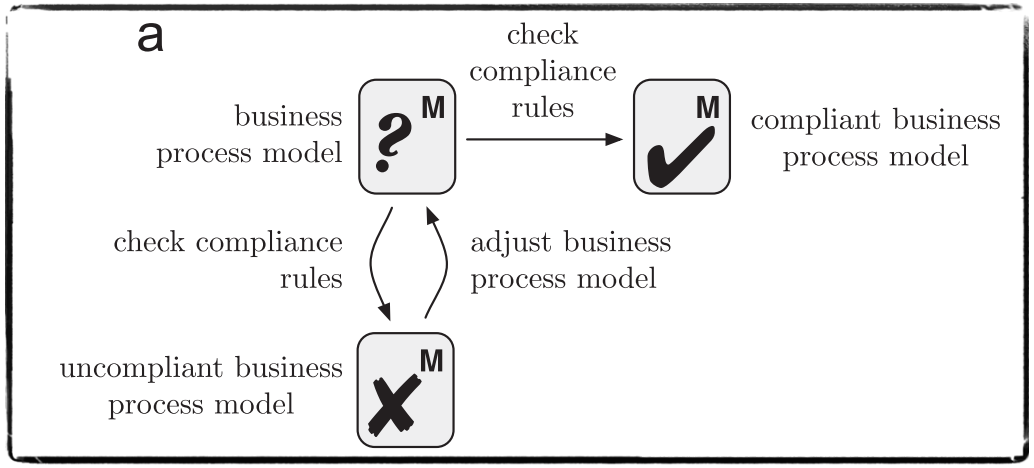
2. Compliance as Conformance Checking



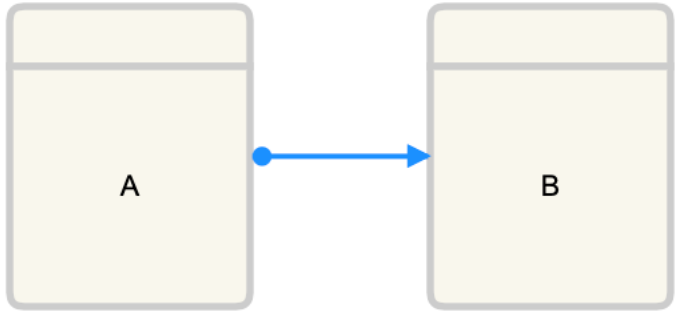
2. Compliance as Conformance Checking



2.Compliance via Conformance Checking



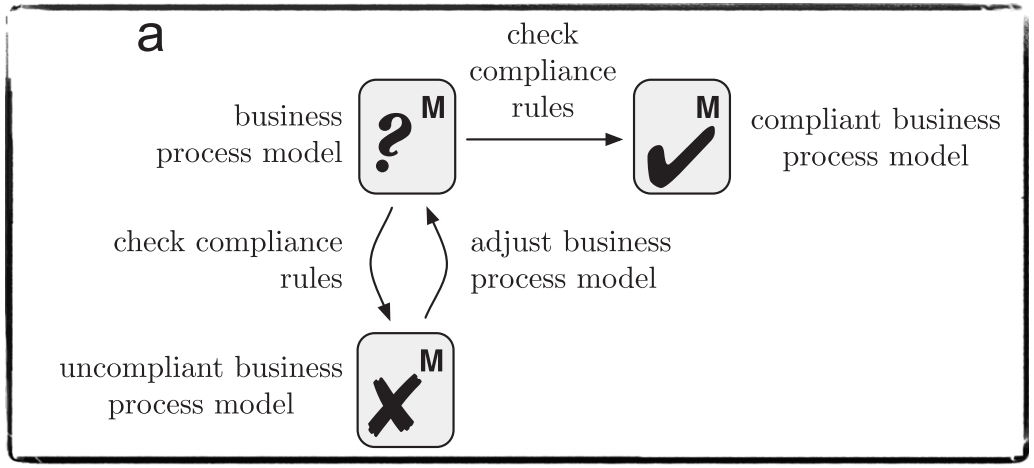
Compliance Rule
(Response of a Task)



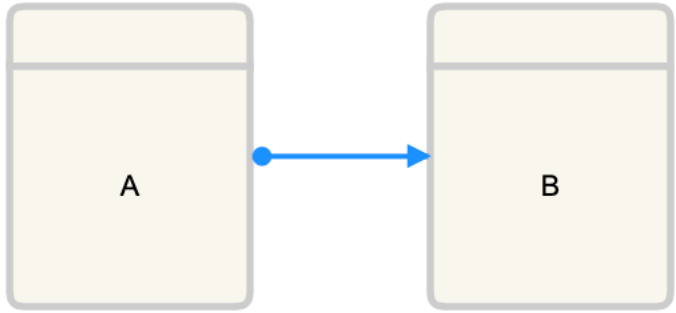
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >
< G, C, F, D, G >
< C, A, B, D, B >
< G, C, B, A, D >
< A, F, A, D, B >
< A, D, B, G, A >

2.Compliance via Conformance Checking



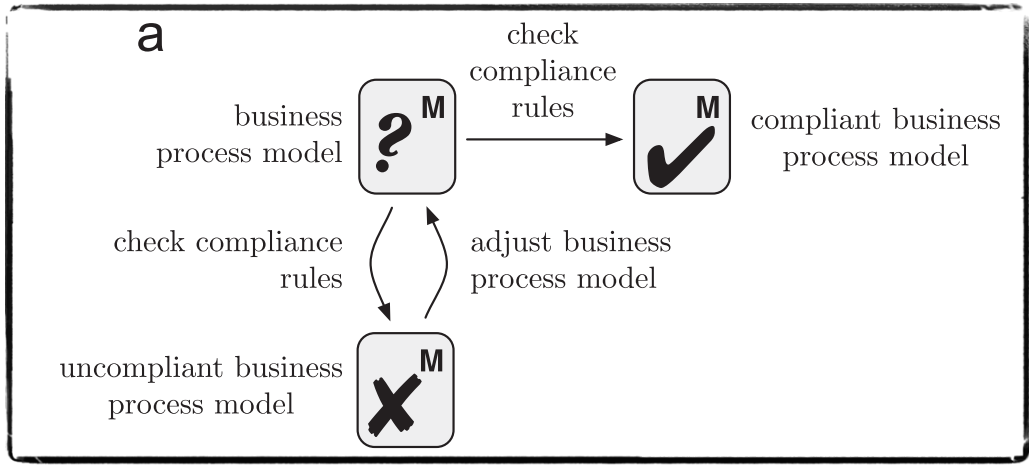
Compliance Rule
(Response of a Task)



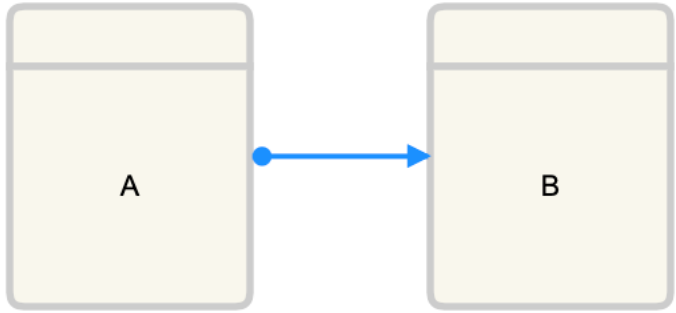
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B > ✓
< G, C, F, D, G >
< C, A, B, D, B >
< G, C, B, A, D >
< A, F, A, D, B >
< A, D, B, G, A >

2.Compliance via Conformance Checking



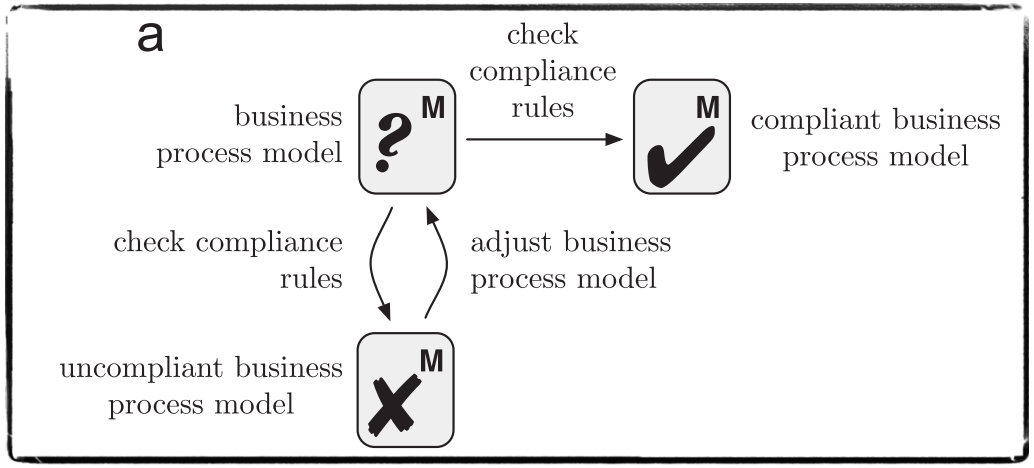
Compliance Rule
(Response of a Task)



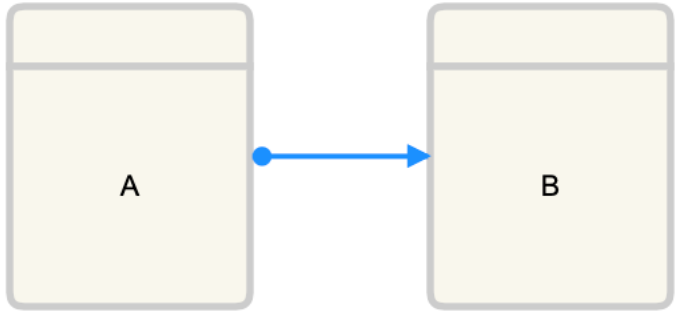
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B > ✓
< G, C, F, D, G > ✓
< C, A, B, D, B >
< G, C, B, A, D >
< A, F, A, D, B >
< A, D, B, G, A >

2.Compliance via Conformance Checking



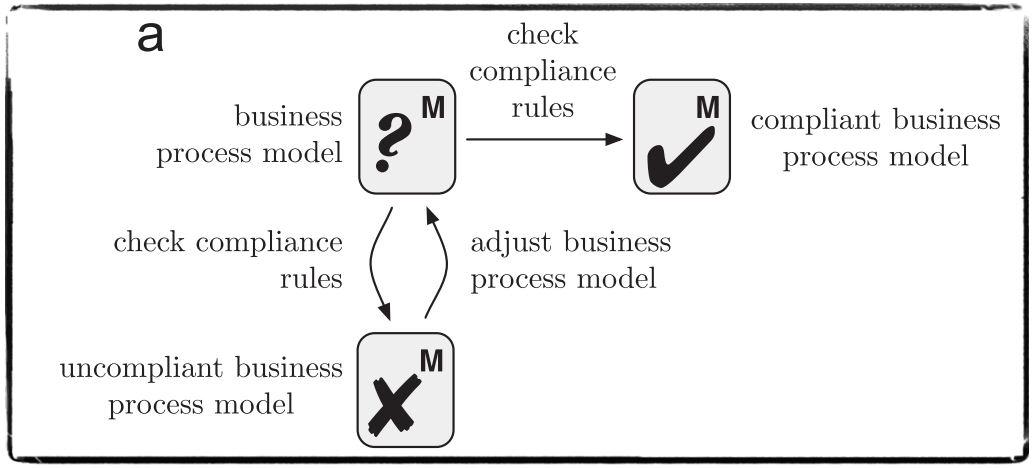
Compliance Rule
(Response of a Task)



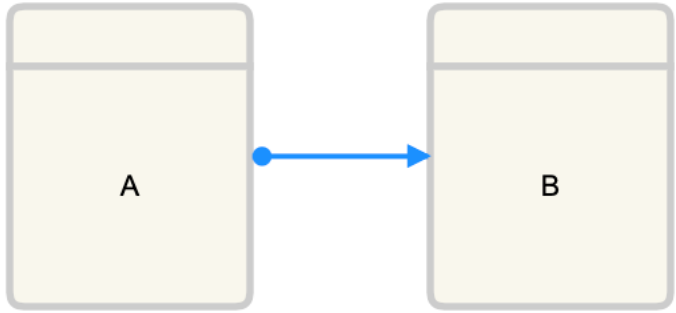
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B > ✓
< G, C, F, D, G > ✓
< C, A, B, D, B > ✓
< G, C, B, A, D >
< A, F, A, D, B >
< A, D, B, G, A >

2.Compliance via Conformance Checking



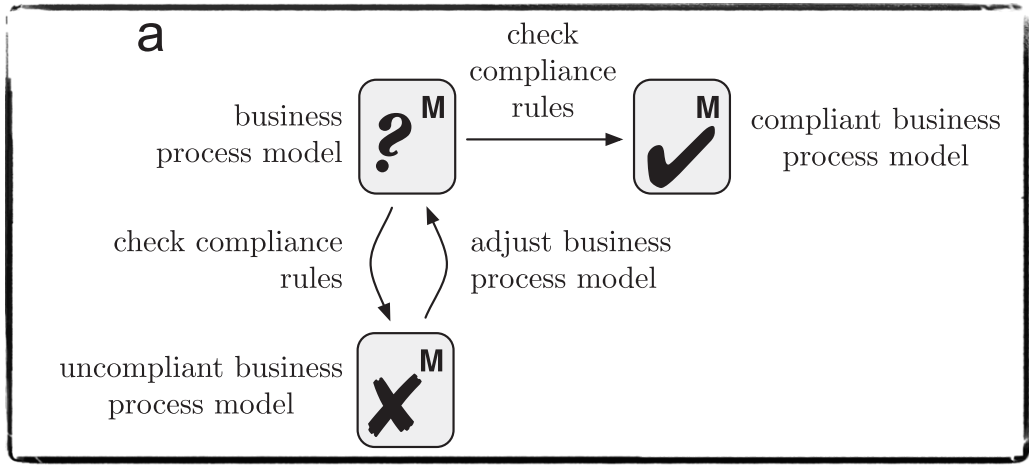
Compliance Rule
(Response of a Task)



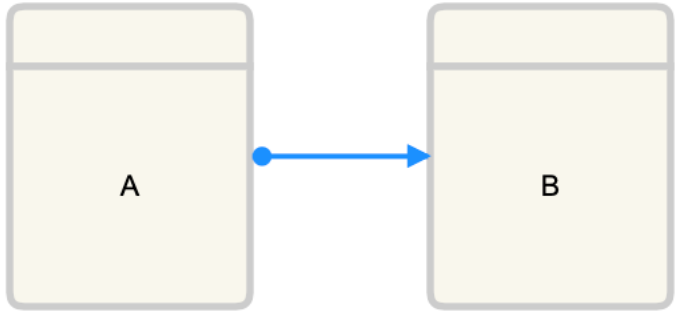
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B > ✓
< G, C, F, D, G > ✓
< C, A, B, D, B > ✓
< G, C, B, A, D > ☹
< A, F, A, D, B >
< A, D, B, G, A >

2.Compliance via Conformance Checking



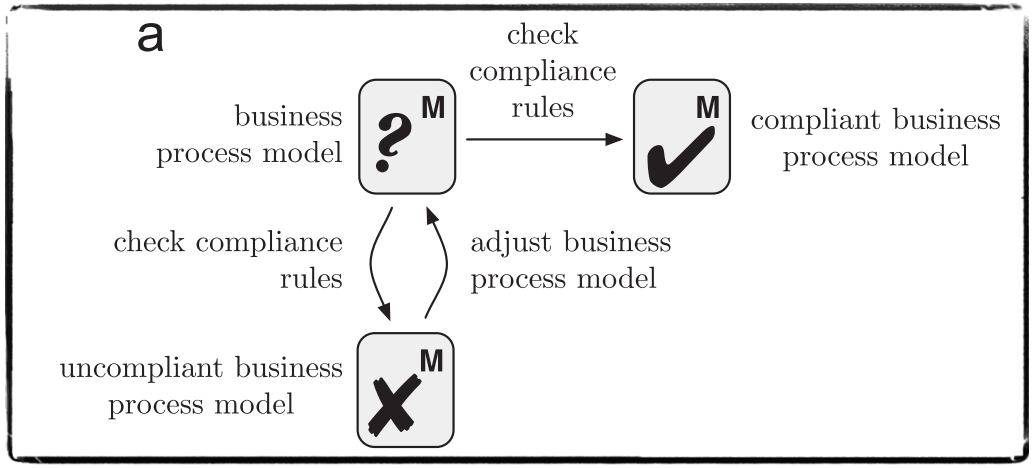
Compliance Rule
(Response of a Task)



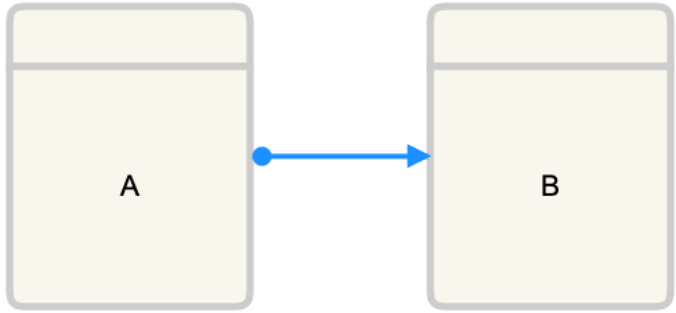
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	

2.Compliance via Conformance Checking



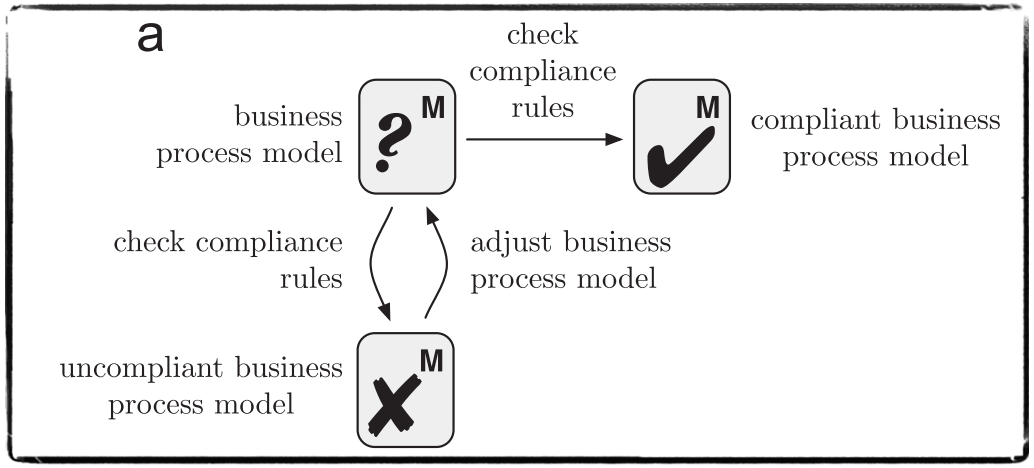
Compliance Rule
(Response of a Task)



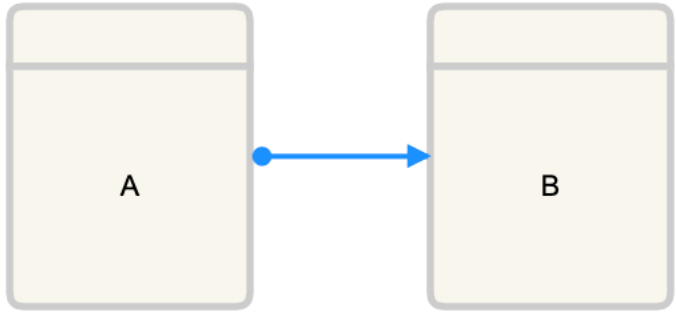
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

2.Compliance via Conformance Checking



Compliance Rule
(Response of a Task)

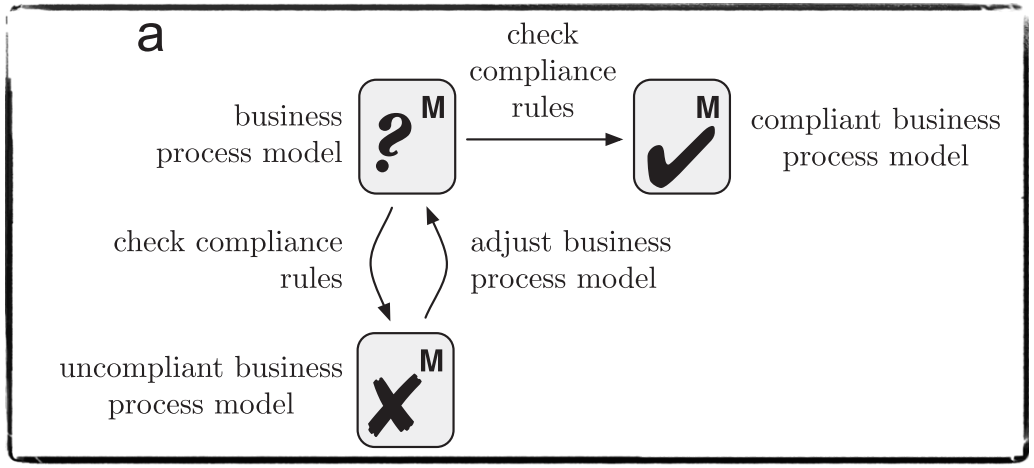


Full Compliant
All traces satisfy the rule

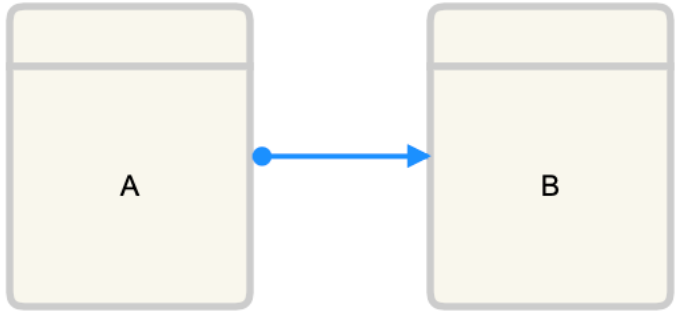
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

2.Compliance via Conformance Checking



Compliance Rule
(Response of a Task)



Full Compliant

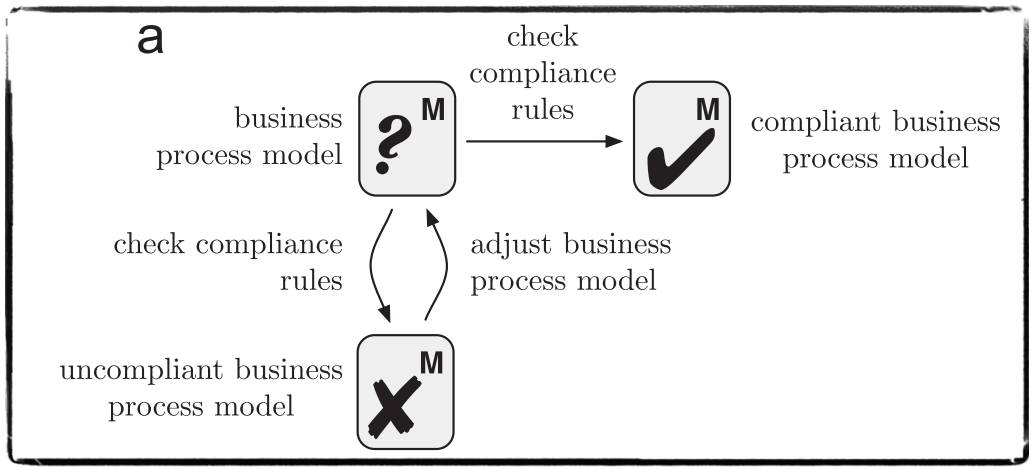
All traces satisfy the rule ☹

Event log

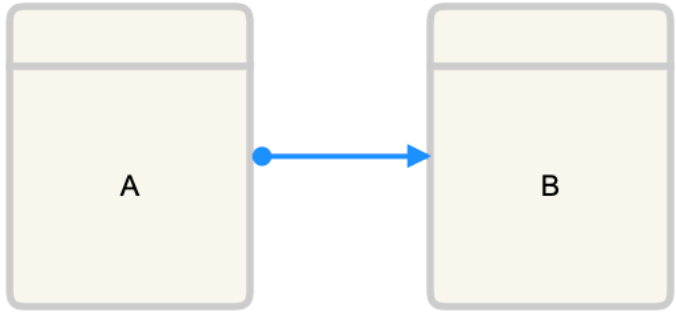
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

2.Compliance via Conformance Checking



Compliance Rule
(Response of a Task)



Full Compliant

All traces satisfy the rule

☹

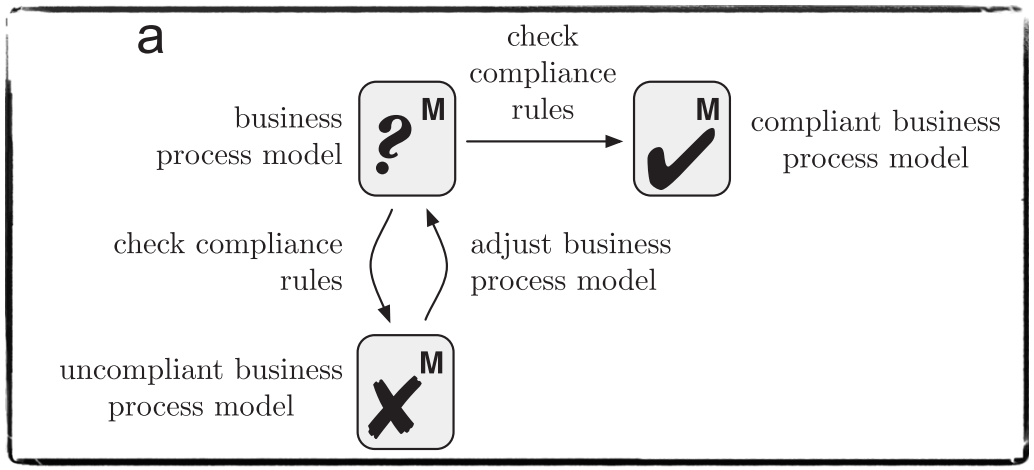
Partially (Weakly) Compliant

At least a trace satisfy the rule

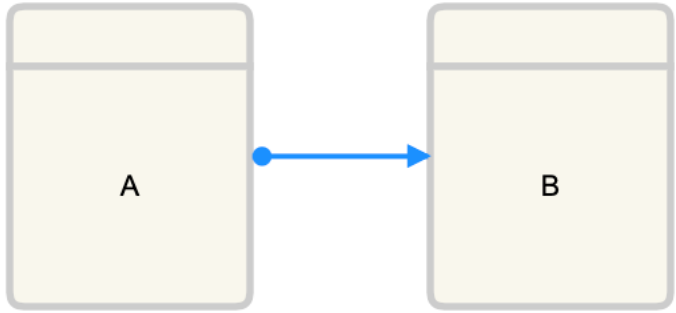
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

2.Compliance via Conformance Checking



Compliance Rule
(Response of a Task)



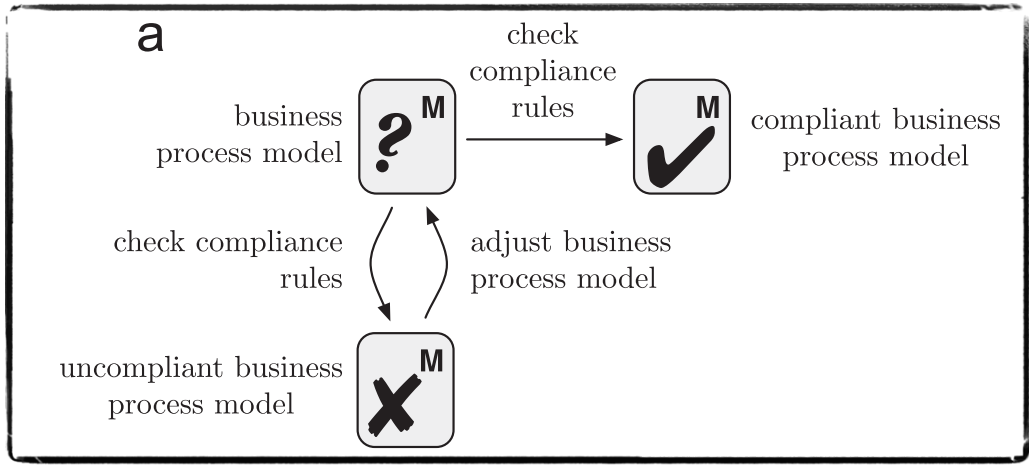
Full Compliant
All traces satisfy the rule ☹

Partially (Weakly) Compliant
At least a trace satisfy the rule ☺

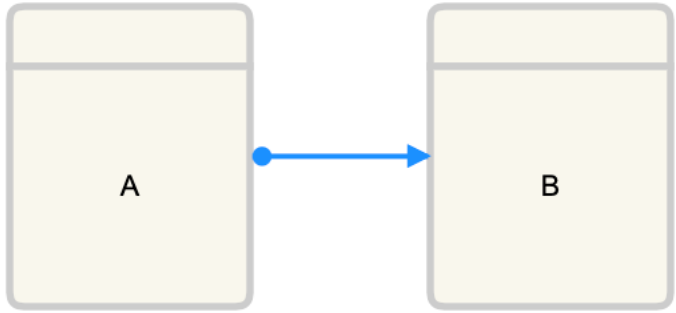
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

2.Compliance via Conformance Checking



Compliance Rule
(Response of a Task)



Full Compliant

All traces satisfy the rule ☹

Partially (Weakly) Compliant

At least a trace satisfy the rule ☺

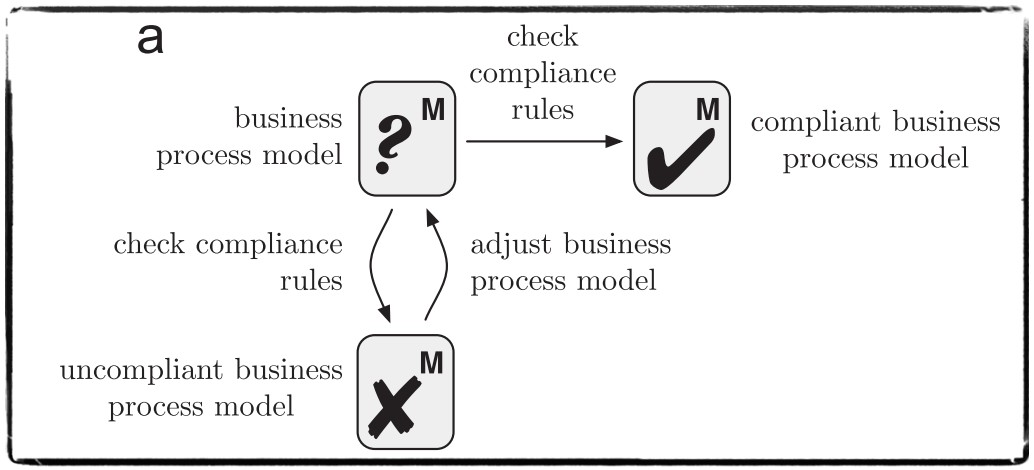
Violated

All traces violate the rule

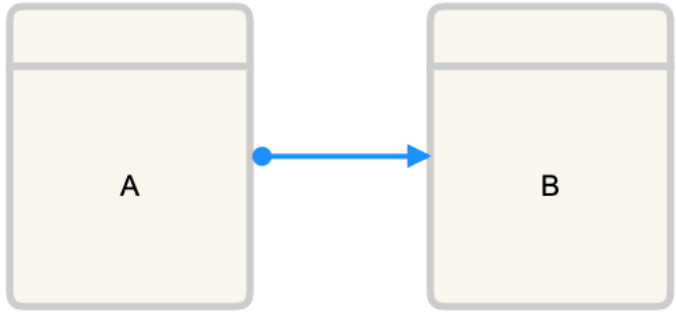
Event log
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

2.Compliance via Conformance Checking



Compliance Rule
(Response of a Task)



Full Compliant

All traces satisfy the rule ☹

Partially (Weakly) Compliant

At least a trace satisfy the rule ☺

Violated

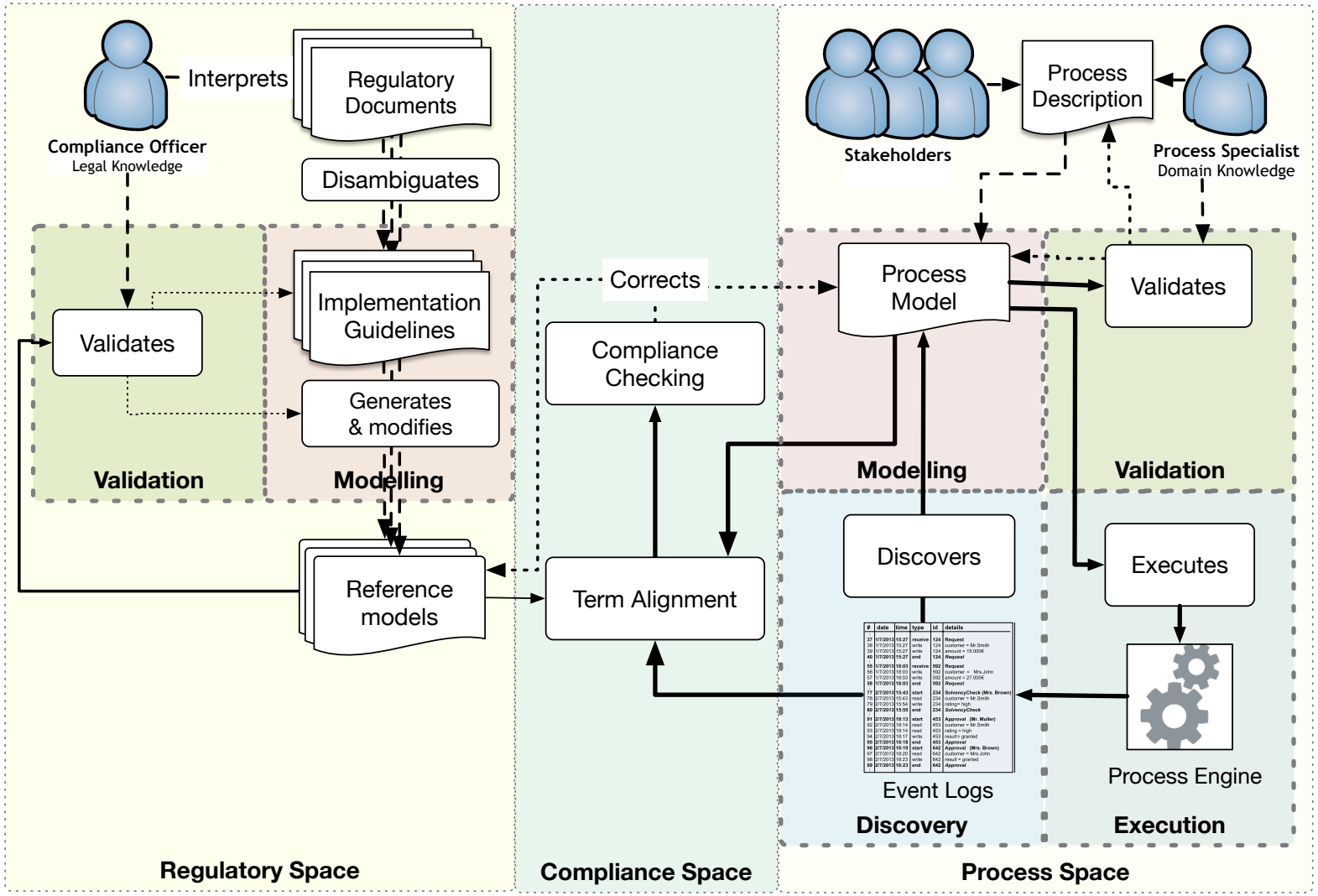
All traces violate the rule ☹

Event log

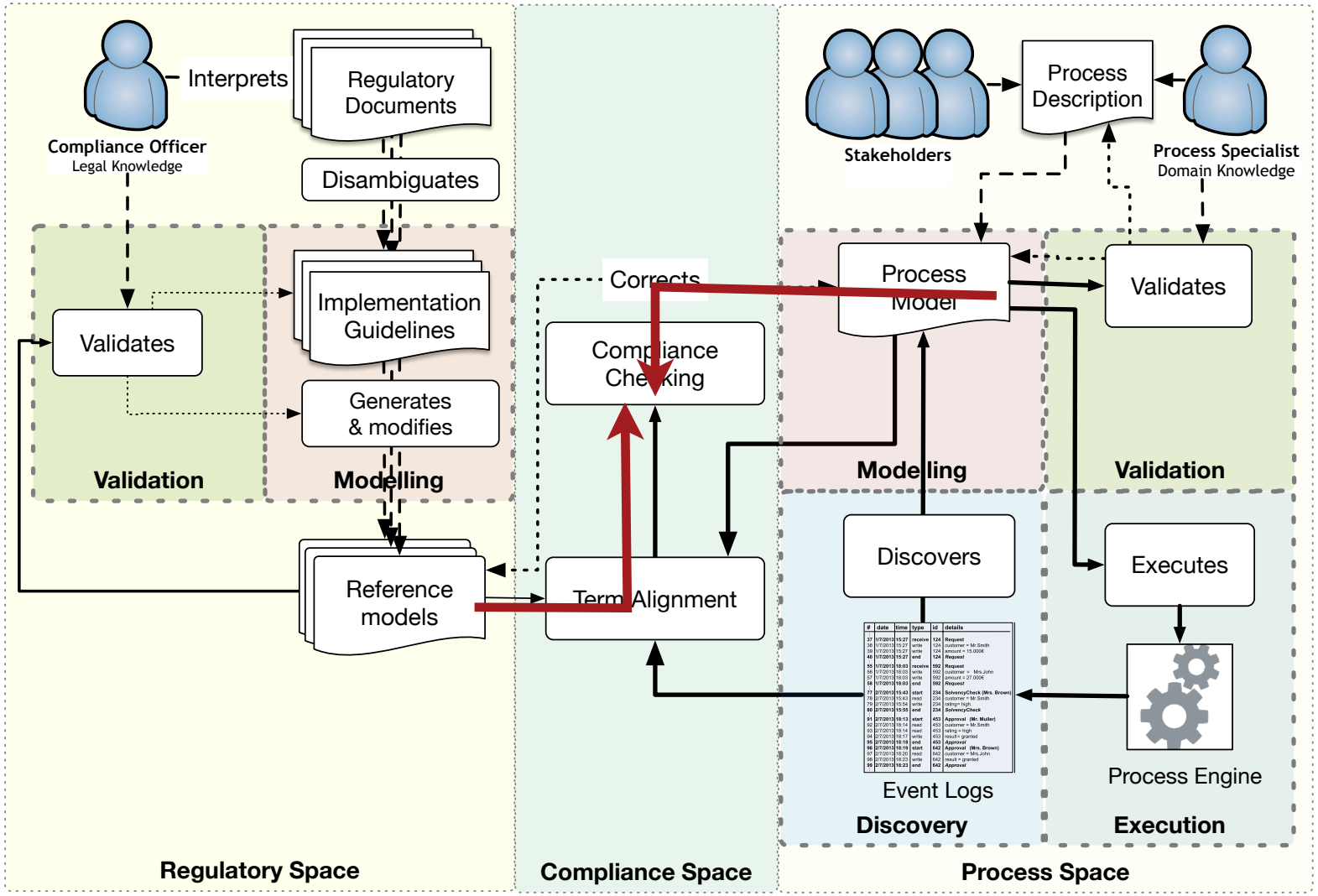
(Might involve events outside the scope of the rule)

< E, D, F, G, B >	✓
< G, C, F, D, G >	✓
< C, A, B, D, B >	✓
< G, C, B, A, D >	☹
< A, F, A, D, B >	✓
< A, D, B, G, A >	☹

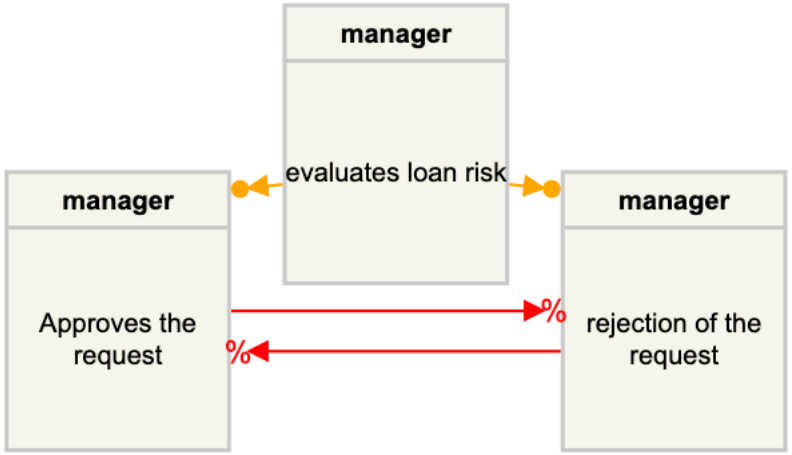
3. Compliance as Process Refinement



3. Compliance as Process Refinement

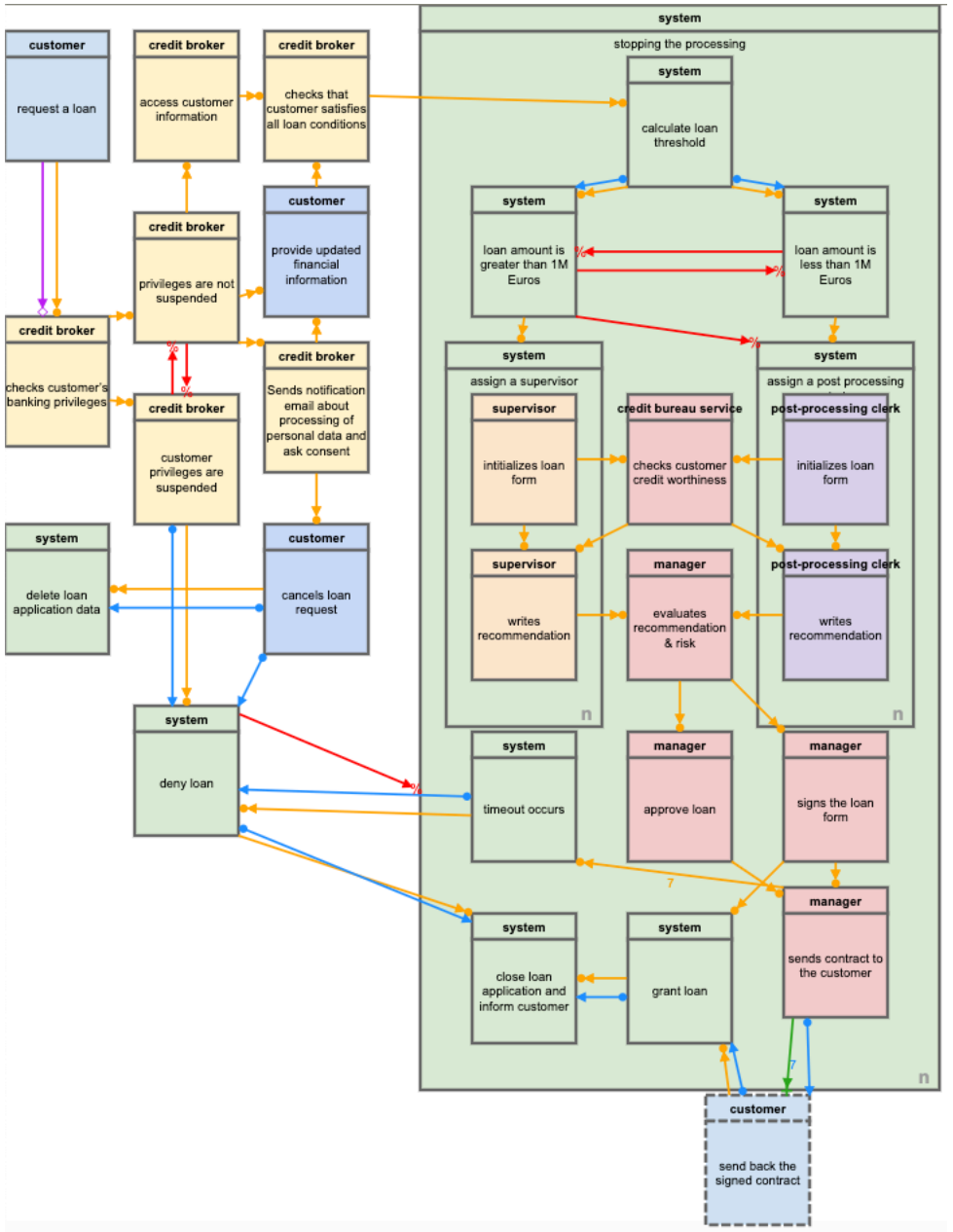


3. Compliance as Process Refinement



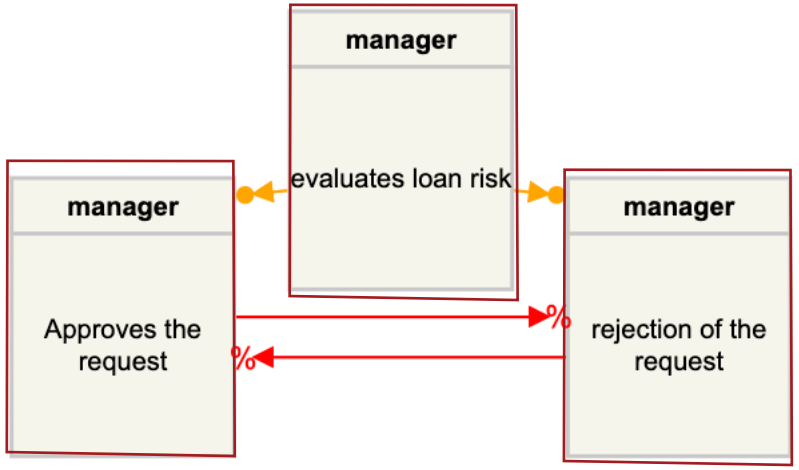
The branch office manager checks whether the risks calculated by the supervisor or the clerk, are acceptable, after which he makes either the final approval or the rejection of the request

Normative Model



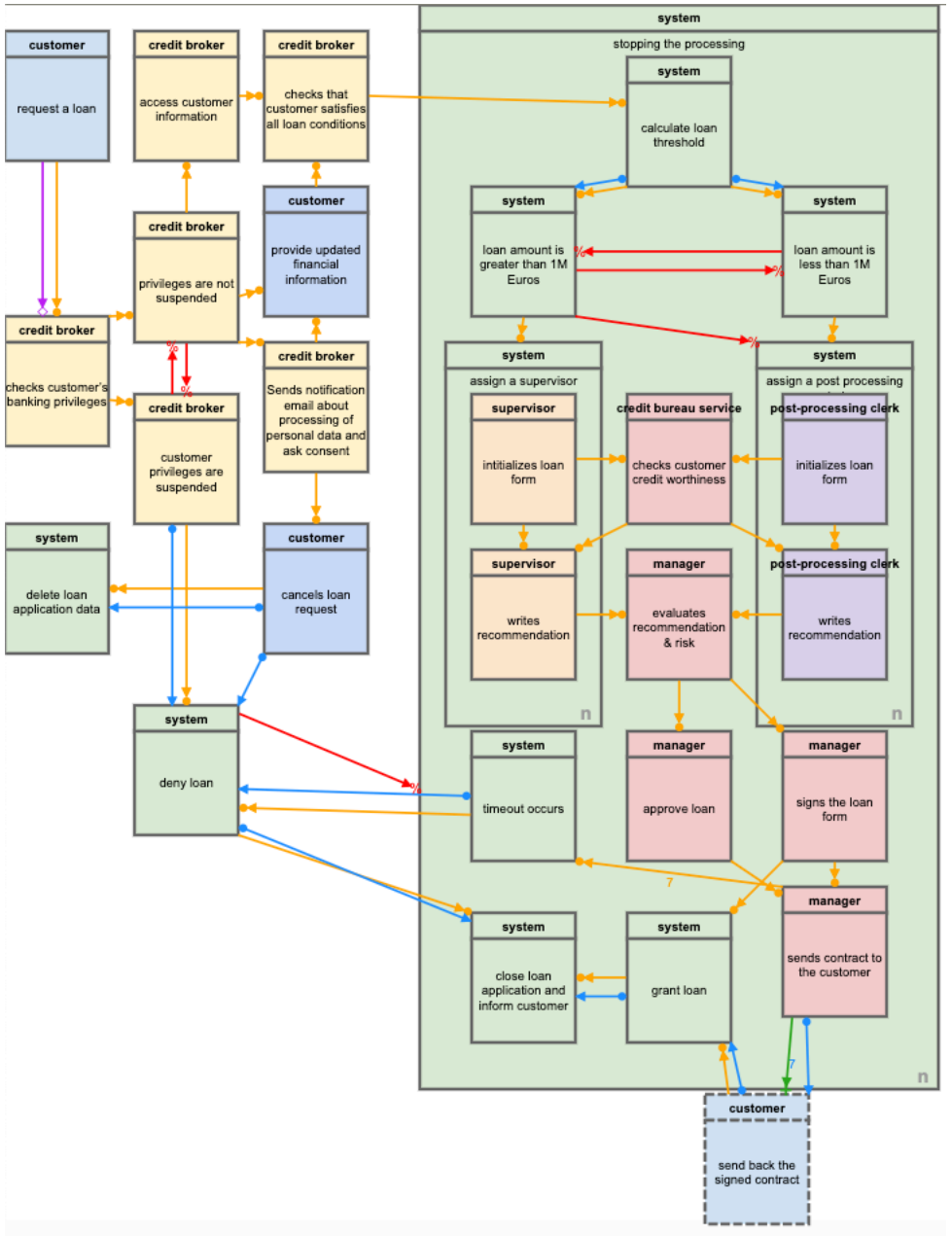
Process Model

3. Compliance as Process Refinement



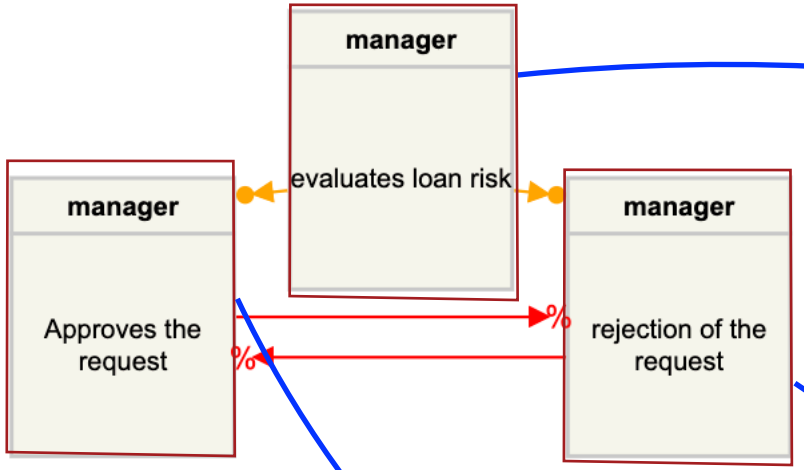
The branch office manager checks whether the risks calculated by the supervisor or the clerk, are acceptable, after which he makes either the final approval or the rejection of the request

Normative Model



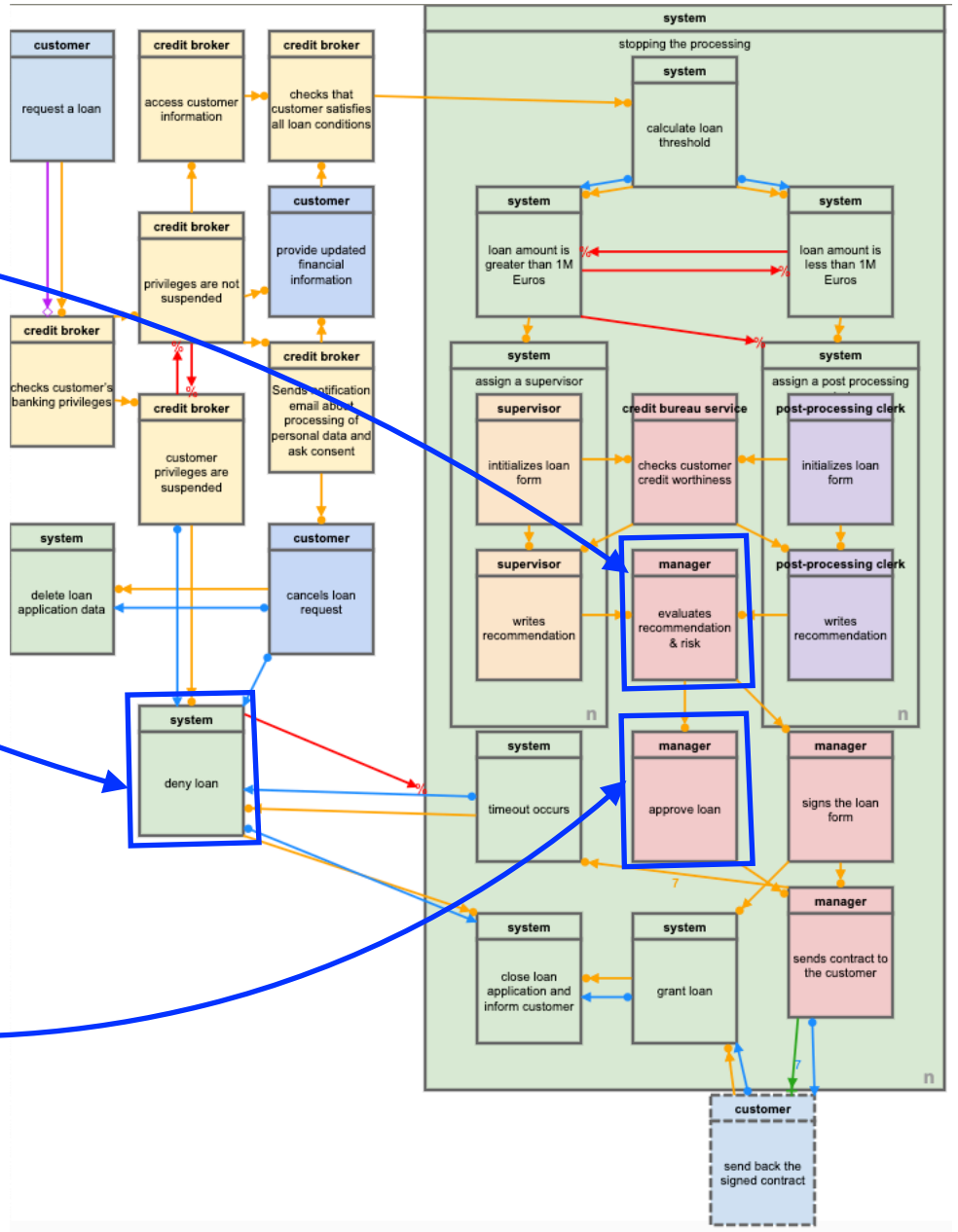
Process Model

3. Compliance as Process Refinement



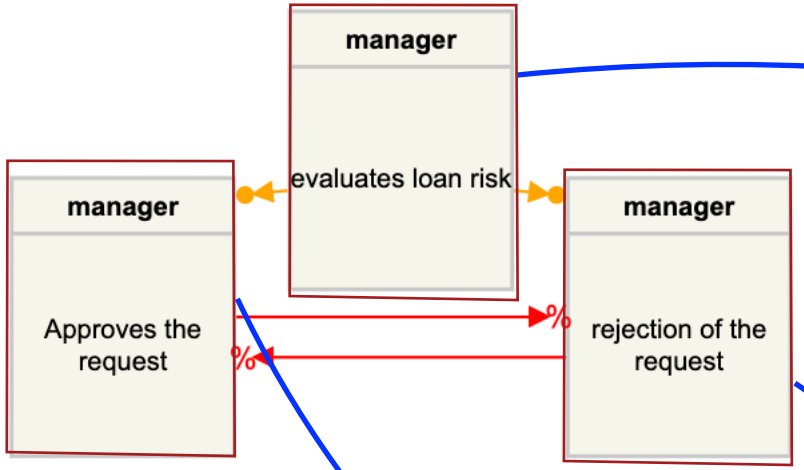
The branch office manager checks whether the risks calculated by the supervisor or the clerk, are acceptable, after which he makes either the final approval or the rejection of the request

Normative Model



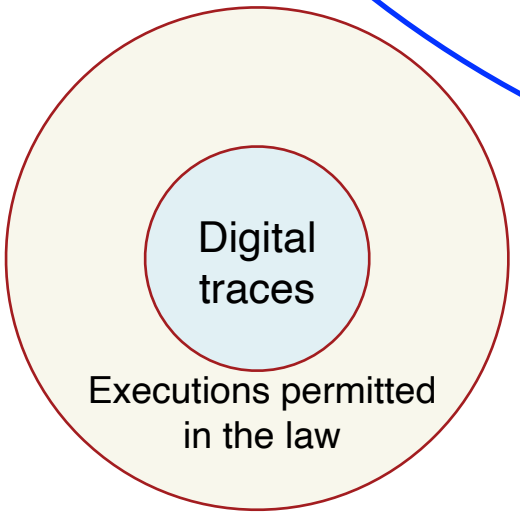
Process Model

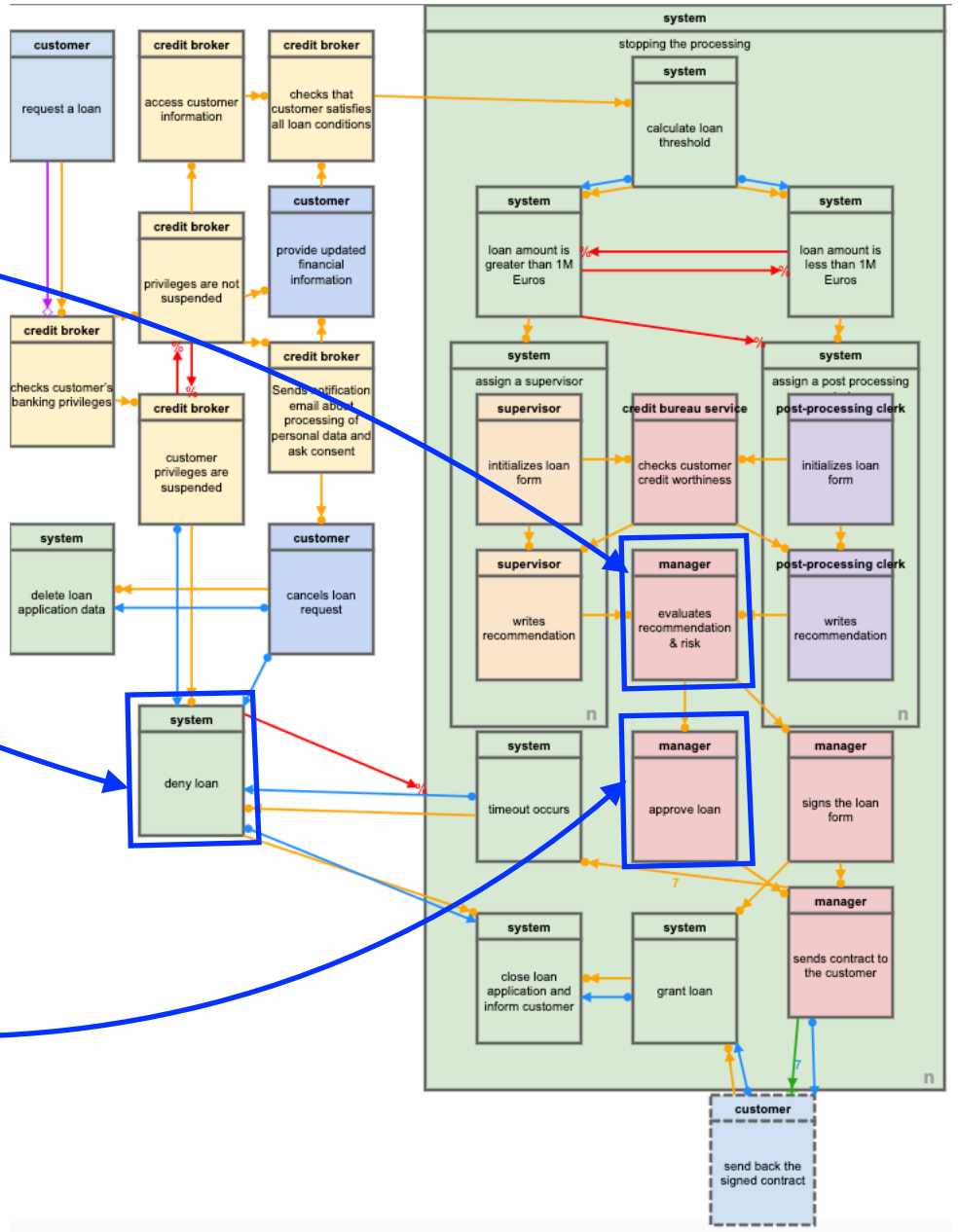
3. Compliance as Process Refinement



The branch office manager checks whether the risks calculated by the supervisor or the clerk, are acceptable, after which he makes either the final approval or the rejection of the request

Normative Model





Process Model

ALIGNING LAWS AND PROCESSES

Definition 3 (Term Alignment & Target events). *Let* $L, L' \subseteq \mathcal{L}$. *A* term alignment *is the function* $g : L \to \mathcal{P}(L')$, *such that* $\forall l \in L.\ g(l) \neq \emptyset$. *If* P, Q *are DCR processes with labels* L, L' *respectively, we say that* g *is a term alignment from* P *to* Q *if* g *is a term alignment from* L *to* L'. *Moreover, we define the target event of* g *for* e *in* P *as* $tg(g, e, P) = \{\lambda^{-1}(g(l)) \mid \lambda(l) \in \mathsf{dom}(g) \wedge e \in \mathsf{fe}(P)\}$.

Event in Legislation	**Activity/event in Process Model**
B_1: Process personal data	A2: Submit change request
B_2: Right to object	A3: Cancel request
B_3: Stop processing	A17: Delete request

ALIGNING LAWS AND PROCESSES

Definition 3 (Term Alignment & Target events). *Let* $L, L' \subseteq \mathcal{L}$. *A* term alignment *is the function* $g : L \to \mathcal{P}(L')$, *such that* $\forall l \in L.\ g(l) \neq \emptyset$. *If* P, Q *are DCR processes with labels* L, L' *respectively, we say that* g *is a term alignment from* P *to* Q *if* g *is a term alignment from* L *to* L'. *Moreover, we define the target event of* g *for* e *in* P *as* $tg(g, e, P) = \{\lambda^{-1}(g(l)) \mid \lambda(l) \in \mathsf{dom}(g) \wedge e \in \mathsf{fe}(P)\}$.

Event in Legislation	**Activity/event in Process Model**
B_1: Process personal data	A2: Submit change request
B_2: Right to object	A3: Cancel request
B_3: Stop processing	A17: Delete request

Definition 4 (Instances of a Compliance Rule). *Let* $G = \{g_1, \ldots, g_n\}$ *be a set of term alignments from* P *to* Q, *an instance of* P *under* g *in* Q, *written* $P{\downarrow_g}\ Q$ *for* $g \in G$, *is* $P\sigma$ *with labelling* $\lambda'(e) = g(\lambda(e))$ *when defined, and* $\lambda(e)$ *otherwise, such that* $\sigma = \{f_1, \ldots, f_n / e_1, \ldots, e_n\}$ *and* $f_i = tg(g, e_i, P), \forall 1 \leq i \leq n$. *The set* $Inst(P, G, Q)$ *denotes the set of all the instances of* P *under* G *in* Q, *that is* $\{R \mid R = P{\downarrow_g}\ Q \ \wedge \ g \in G\}$.

ALIGNING LAWS AND PROCESSES

Definition 3 (Term Alignment & Target events). *Let $L, L' \subseteq \mathcal{L}$. A* term alignment *is the function $g : L \to \mathcal{P}(L')$, such that $\forall l \in L.\ g(l) \neq \emptyset$. If P, Q are DCR processes with labels L, L' respectively, we say that g is a term alignment from P to Q if g is a term alignment from L to L'. Moreover, we define the target event of g for e in P as $tg(g, e, P) = \{\lambda^{-1}(g(l)) \mid \lambda(l) \in \mathsf{dom}(g) \wedge e \in \mathsf{fe}(P)\}$.*

Event in Legislation	**Activity/event in Process Model**
B_1: Process personal data	A2: Submit change request
B_2: Right to object	A3: Cancel request
B_3: Stop processing	A17: Delete request

Definition 4 (Instances of a Compliance Rule). *Let $G = \{g_1, \ldots, g_n\}$ be a set of term alignments from P to Q, an instance of P under g in Q, written $P{\downarrow_g}\ Q$ for $g \in G$, is $P\sigma$ with labelling $\lambda'(e) = g(\lambda(e))$ when defined, and $\lambda(e)$ otherwise, such that $\sigma = \{f_1, \ldots, f_n / e_1, \ldots, e_n\}$ and $f_i = tg(g, e_i, P), \forall 1 \leq i \leq n$. The set $Inst(P, G, Q)$ denotes the set of all the instances of P under G in Q, that is $\{R \mid R = P{\downarrow_g}\ Q\ \wedge\ g \in G\}$.*

Term Alignment	**Label Reference Model**	**Event** P_{spec}	**Label Process Model**
g_3	A2: submit a change request	f_1	A2: submit a change request
	Finish Processing request	f_2	A15: Amend initial contract with approved change
	Cancel Processing	f_3	A3: Delete request
g_4	A2: submit a change request	f_1	A2: submit a change request
	Finish Processing request	f_4	A16: Receive reason for change rejection
	Cancel Processing	f_3	A15: Delete request

Compliance as process refinement

- **Process Refinement**. Let P_{law}, $P_{process}$ be processes. We say that $P_{process}$ *is a refinement* of P_{law} (written $P_{process} \sqsubseteq P_{law}$) iff $lang(P_{process})|_{alph(P_{law})} \subseteq lang(P_{law})$.

Compliance as process refinement

- **Process Refinement**. Let P_{law}, $P_{process}$ be processes. We say that $P_{process}$ *is a refinement* of P_{law} (written $P_{process} \sqsubseteq P_{law}$) iff $lang(P_{process})|_{alph(P_{law})} \subseteq lang(P_{law})$.

Obs: For an arbitrary P, $lang(P)$ can be regular or ω-regular, thus not very practical

Compliance as process refinement

- **Process Refinement**. Let P_{law}, $P_{process}$ be processes. We say that $P_{process}$ *is a refinement* of P_{law} (written $P_{process} \sqsubseteq P_{law}$) iff $lang(P_{process})|_{alph(P_{law})} \subseteq lang(P_{law})$.

Obs: For an arbitrary *P*, *lang(P)* can be regular or ω-regular, thus not very practical

- **Alternative: Refinement as composition (Debois et al 2017)**. Idea: *R, P* are in refinement, iff we can merge the events of *R* and *P* and still behave as *R.*
- Merge ($\oplus$) is defined as the partial relation between markings agreeing on their overlap:

$$(M_1, e:m) \oplus (M_2, e:m) = (M_1 \oplus M_2), e:m$$
$$(M_1, e:m) \oplus M_2 = (M_1 \oplus M_2), e:m \quad \text{when } e \notin \mathsf{dom}(M_2)$$
$$M_1 \oplus (M_2, e:m) = (M_1 \oplus M_2), e:m \quad \text{when } e \notin \mathsf{dom}(M_1).$$

- This extends to processes in the standard way. For $P=[M]\lambda_1 T$ and $Q=[N]\lambda_2 U$, then $P \oplus Q = [M \oplus N](\lambda_1 \cup \lambda_2)(T \| U)$
- **Refines:** *Q refines P iff* $P \oplus Q \sqsubseteq P$

Compliance as refinement

- **Intuition**: when checking compliance between rule *R*, a process *P* and a term alignment mapping labels in *R* to *P*. Compliance requires us
  1. Generate all instances of *R* in *P* and
  2. Check whether the merge of each instance with the *P* is compatible (i.e. refines) the instance.

(Compliance). Let *P,R* be DCR processes, and *G* be a set of term alignments from *R* to P. We say that *P* is compliant with *R* under *G*, written $P \leq_s^G R$ if $\forall R_i \in Inst(R,G,P)$, *P* refines R_i.

Algorithmic Compliance Checking

- Language inclusion is NP-hard for DCR Graphs (Debois et al 2017).
- We give efficient static guarantees for process refinement.
- For DCR graphs *P, R*, an implementation *P* is **non-invasive** to its specification *R* iff their markings are compatible, and *P* does not induce inclusions, exclusions or responses to the free events in *R.*

Definition 5.7 (Non-invasiveness). *Let* $P = [M_P]\ \lambda_P\ T_P$ *and* R *be marking compatible processes. We say that* P *is* non-invasive *for* R *iff for every context* $C[-]$, *such that* $T_P = C[e \rightarrow\!\!\!\!\% f]$, $T_P = C[e \rightarrow\!\!\!\!+ f]$ *or* $T_P = C[e \bullet\!\!\rightarrow f]$, $f \notin \mathsf{fe}(R)$.

- Complexity: an algorithm only needs to check for each inclusion, response and exclusion relation in *P* if the target event exists in *R*. In its worst case, we will have ***n*** relations in T_P. Assuming a standard set membership operation of **O(1)**, then the worst complexity is **O(n)**.

Implementing Compliance Checking

Fig. 8. Non-invasiveness checking, $R \models T$

$$\frac{}{R \models 0}\ [\text{I-Nil}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \to\!\!\!\!\!\% f}\ [\text{I-Excl}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \to\!\!\!\!+ f}\ [\text{I-Incl}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \bullet\!\!\to f}\ [\text{I-Resp}]$$

$$\frac{R \models T \qquad R \models U}{R \models T \parallel U}\ [\text{I-Comp}]$$

Implementing Compliance Checking

Fig. 8. Non-invasiveness checking, $R \models T$

$$\frac{}{R \models 0}\ [\text{I-Nil}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \rightarrow\!\!\!\!\% f}\ [\text{I-Excl}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \rightarrow\!\!\!\!+ f}\ [\text{I-Incl}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \bullet\!\!\rightarrow f}\ [\text{I-Resp}]$$

$$\frac{R \models T \qquad R \models U}{R \models T \parallel U}\ [\text{I-Comp}]$$

Algorithm 1 refines relation

Require: DCR graphs $P = [M_p]\ T_p, Q = [M_q]\ T_q$

```
1: function Refines(P, Q)
2:     try
3:         DCR graph merge ← P ⊕ Q
4:         return merge ⊨ T_p
5:     catch P ⊕ Q undefined
6:         return false
7:     end try
```

Implementing Compliance Checking

Fig. 8. Non-invasiveness checking, $R \models T$

$$\frac{}{R \models 0}\ [\text{I-Nil}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \rightarrow\!\!\!\!\% f}\ [\text{I-Excl}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \rightarrow\!\!\!\!+ f}\ [\text{I-Incl}] \qquad \frac{f \notin \mathsf{fe}(R)}{R \models e \bullet\!\!\rightarrow f}\ [\text{I-Resp}]$$

$$\frac{R \models T \qquad R \models U}{R \models T \parallel U}\ [\text{I-Comp}]$$

Algorithm 1 refines relation

Require: DCR graphs $P = [M_p]\ T_p, Q = [M_q]\ T_q$

```
1: function Refines(P, Q)
2:     try
3:         DCR graph merge ← P ⊕ Q
4:         return merge ⊨ T_p
5:     catch P ⊕ Q undefined
6:         return false
7:     end try
```

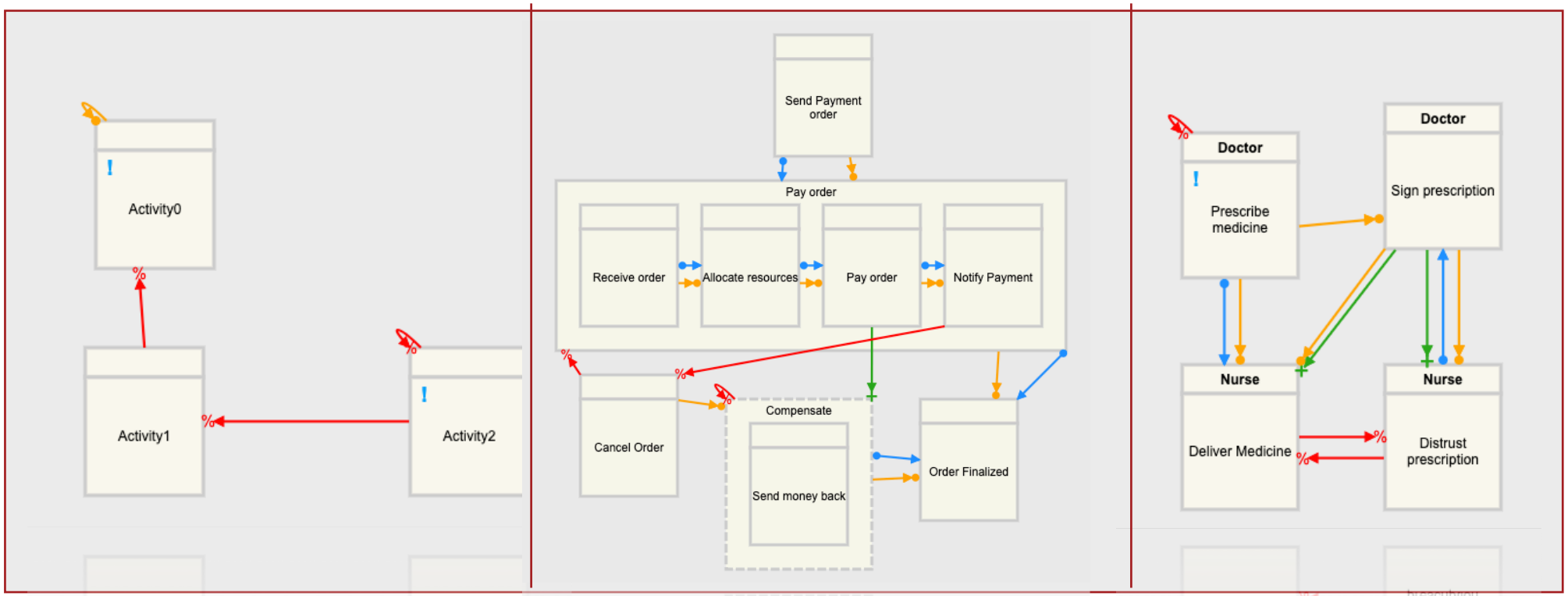
Algorithm 2 Compliance Checking algorithm

Require: DCR graphs P, R, term alignment $G = \{g_i \mid 1 \leq i \leq n\}$
Ensure: $n \in \mathbb{N}$

```
1: function Compliance(P, R, G)
2:     violations ← 0
3:     for each g ∈ G do
4:         DCR graph Q ← P↓_g R
5:         if not refines(P, Q) then
6:             violations ← violations + 1
7:     return (|G| − violations) / |G| * 100%
```

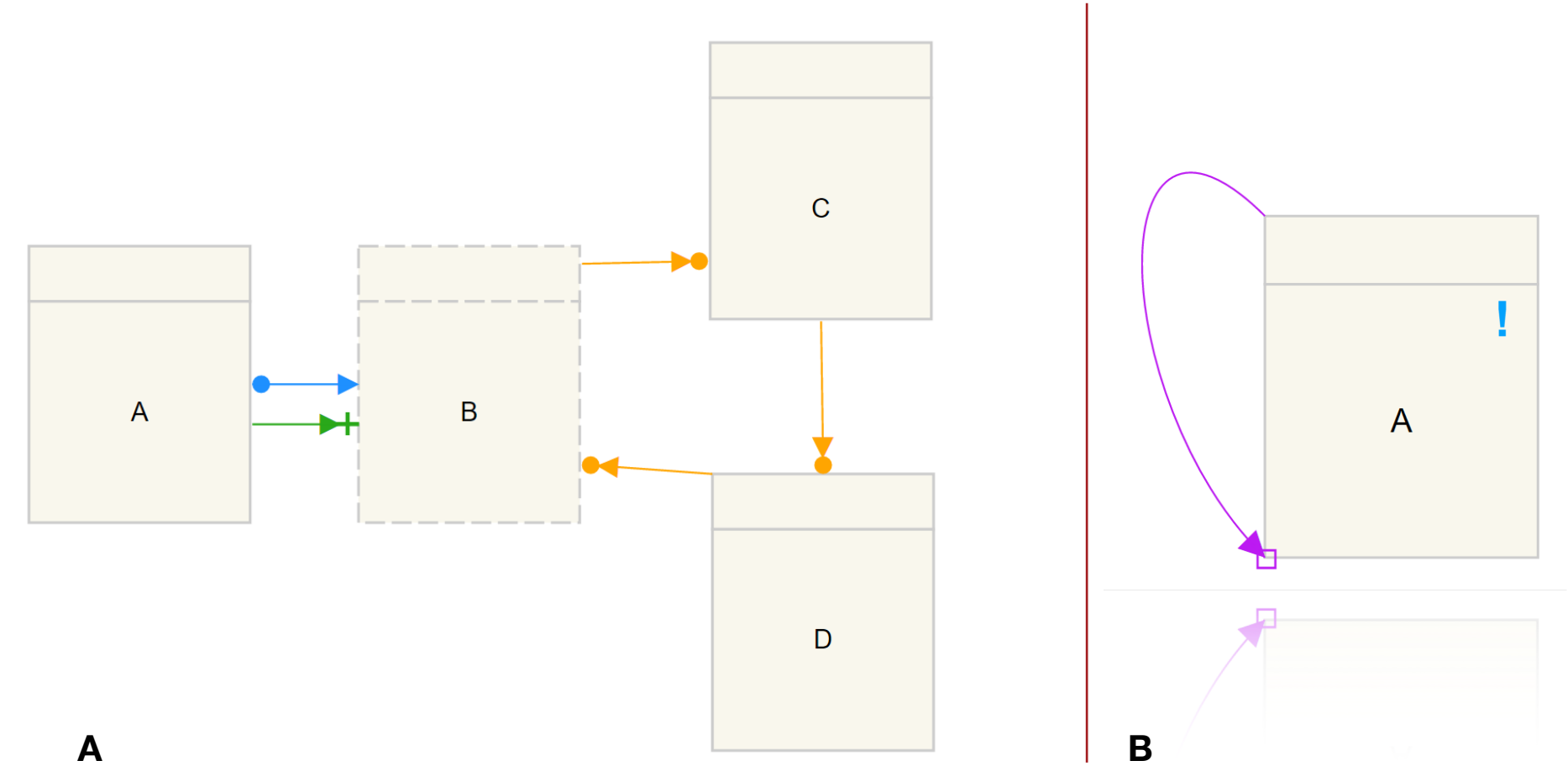
Exercises (Soundness)

1. Give arguments on the soundness or not of the following graphs



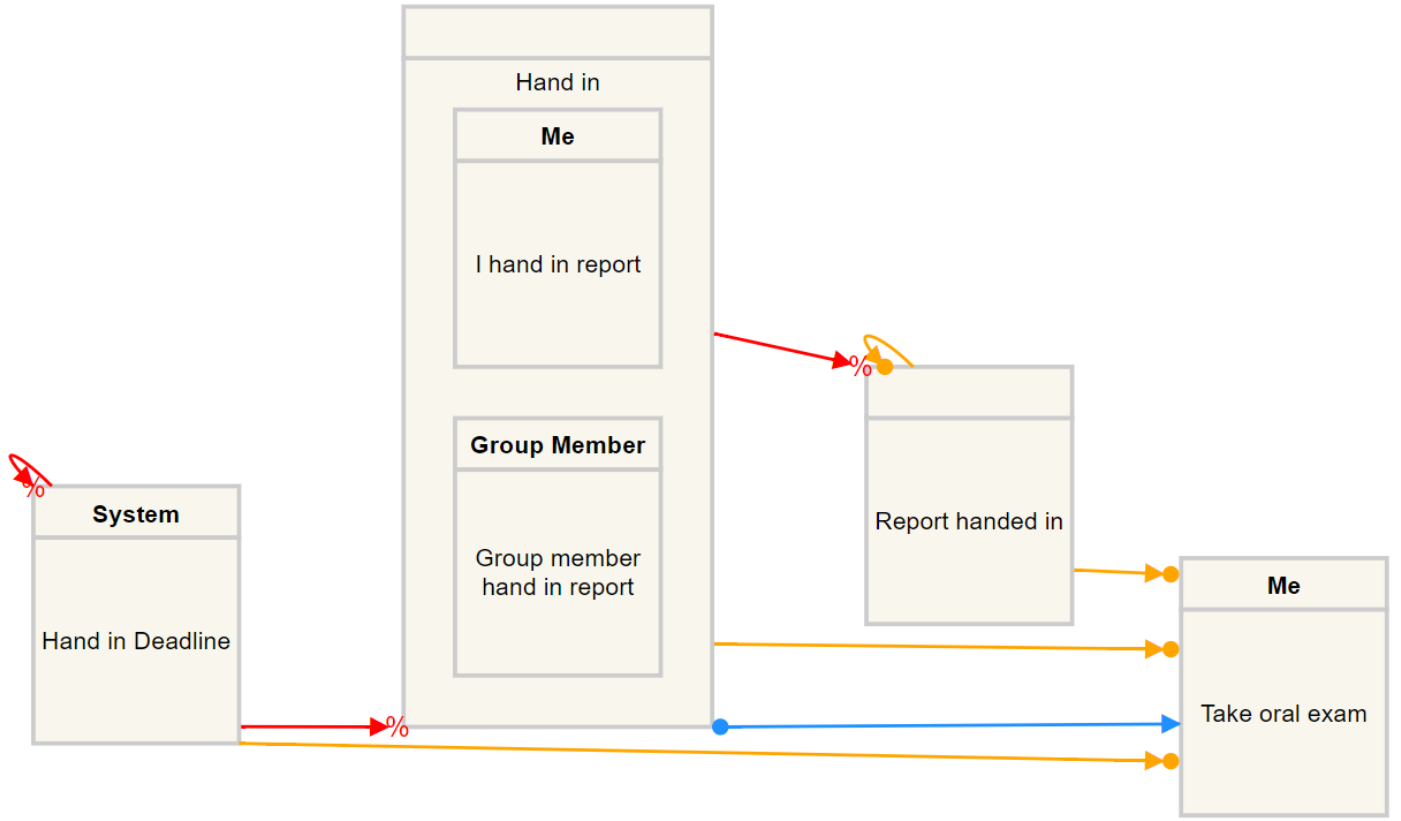
Exercise:

2.Argument whether the following processes are sound:

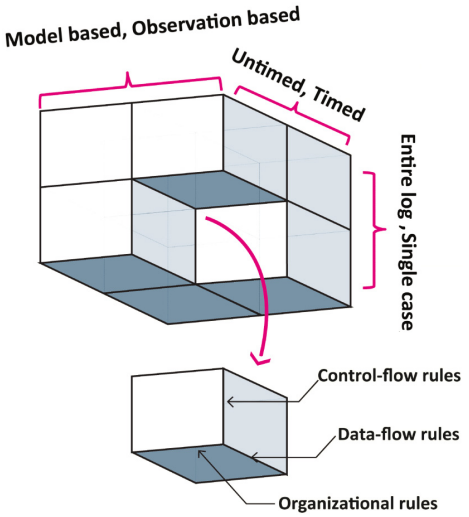


Exercise

3.Argument whether this process is sound:



Exercises (Compliance)



- 4. Discuss in groups the following questions:

A. In which cases will be important to have full compliance, and in which cases will be important to have weak compliance with the rules?

B. Describe one scenario where applying process audit will be more beneficial than applying process monitoring, and vice-versa.

C. According to the dimensions for compliance rules described earlier, give examples of compliance rules describing each dimension.

    A. Can you find examples of multi-dimensional rules?

Third Assignment (Part A)

UNIVERSITY OF COPENHAGEN

Part A: Business Process Compliance

- In this first part of the assignment you will
  - Elicit reference models of laws from real legislations
  - Extend your conformance checking tool in assignment 1 to account for compliance dimensions
  - Implement compliance checking in order to compare models

Exercise A.1

Generate a DCR graph of the policies in §42 of the Danish Consolidation Act for Social Services (Serviceloven). You may use the Process highlighter and the DCR annotation guidelines located in the “Files” section of this course

- *42.–(1) The municipal council shall pay compensation for loss of earnings to persons maintaining a child under 18 in the home whose physical or mental function is substantially and permanently impaired, or who is suffering from serious, chronic or long-term illness. Compensation shall be subject to the condition that the child is cared for at home as a necessary consequence of the impaired function, and that it is most expedient for the mother or father to care for the child.*

- *(2) The requirement in subsection (1) above that the child shall be cared for at home shall not apply to any child mentioned in subsection (1) who has been placed in care under section 52(3) (vii) in connection with the child's hospital visit. It is a condition that the presence of the mother or father at the hospital is a necessary consequence of the child's functional impairment and that such presence is most expedient for the child.*

Exercise A.2

- Construct a set of accepting and non-accepting traces based on activities identified in Exercise A.1.
- Extend the implementation of your conformance checking algorithm so it accounts for the compliance diagnostic criteria in this slide.
- Show via examples how the policies in Exercise A.1 are violated, compliant or weakly compliant.

Exercise A.3.

- Create a compliance checker for DCR graphs. To this aim, you need
  - To create an interface that allows to specify (or upload) a DCR graph of a compliance rule, and a DCR graph of a process model, as well as the term alignments
  - To implement the algorithms for compliance checking given before, providing the diagnostics of whether full or weak compliance has been achieved as a result of the analysis
- You are welcome to reuse modules from previous assignments

Exercise A.4

- Consider the topmost pool of the BPMN model on the right:
- a. Write DCR compliance rules for the following compliance rules:

  a.1. The outpatient department always makes a decision in this process

  a.2. After performing a checkup, either schedule a surgery or write a discharge letter should be executed, but not both in the same trace.

  a.3. After discussing anesthesia and the risk of the patient, the process must follow with a decision.
- Attempt to create an equivalent DCR process model describing the behaviors of the topmost pool, and ensure the process is compliant with rules a.1—a.3 by using the compliance checker developed in Exercise A.3