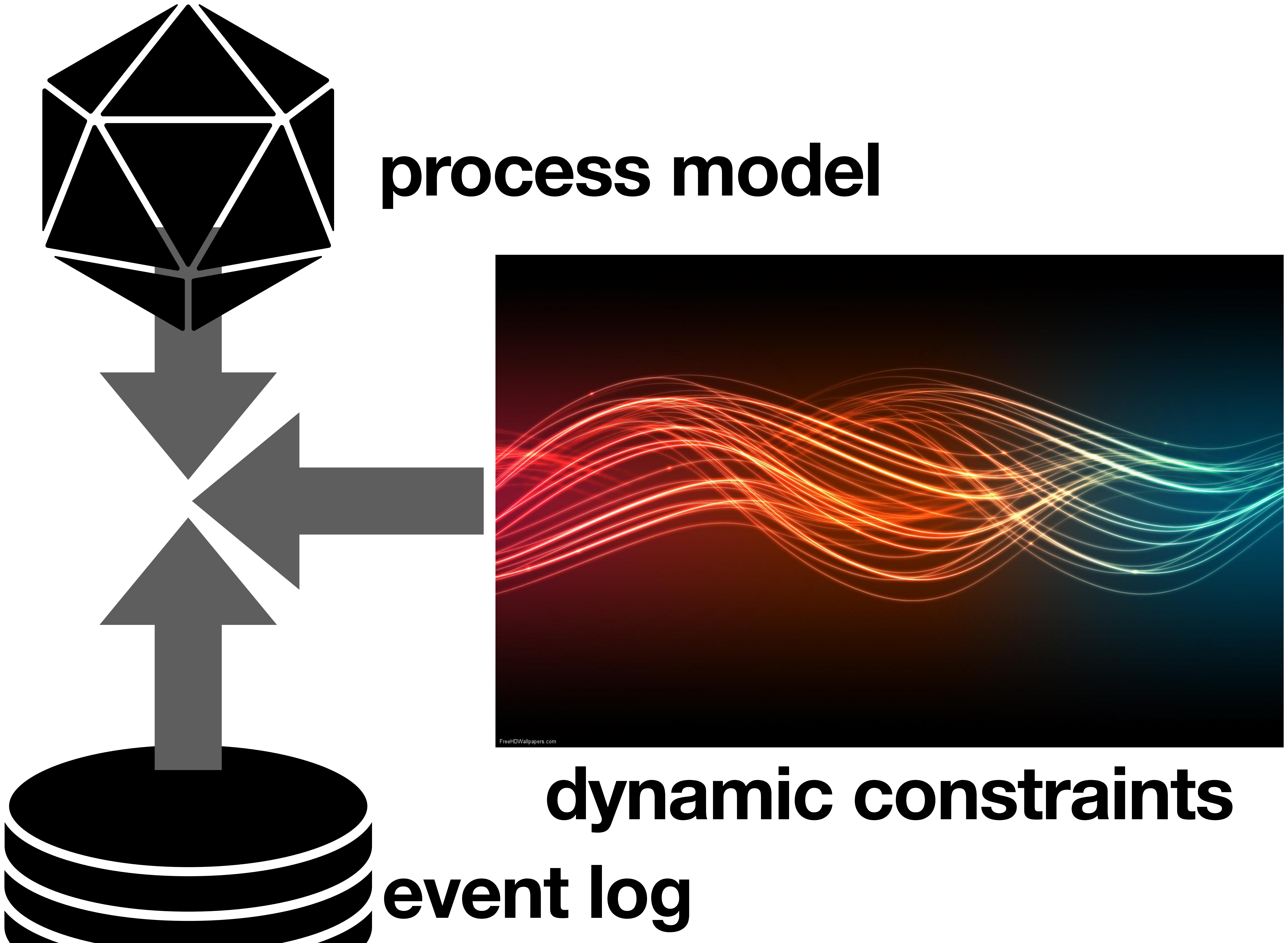
Repairing soundness properties in data-aware processes

Paolo Felli, Marco Montali, Sarah Winkler

ICPM 2023
Rome, Italy

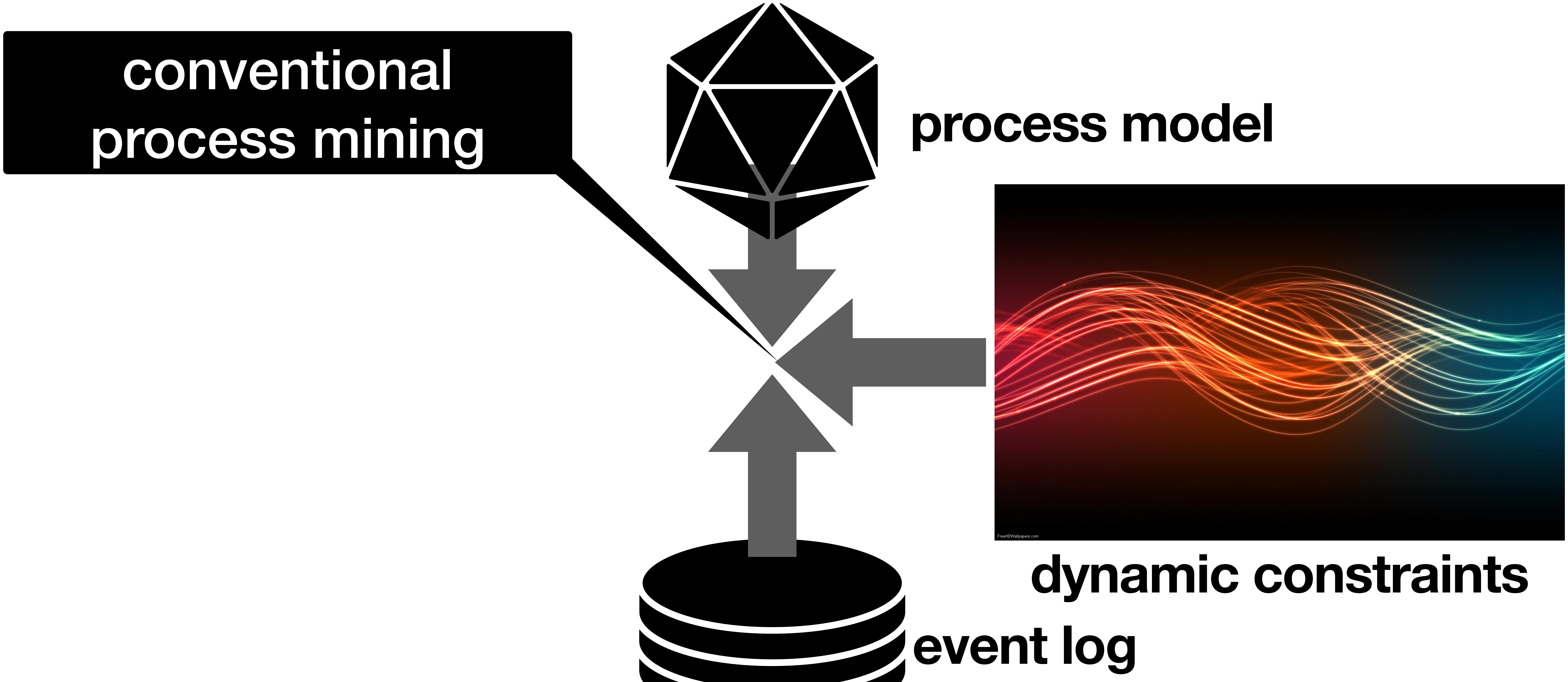
Starting point

A holistic view of information systems



Starting point

A holistic view of information systems



Starting point

A holistic view of information systems

Starting point

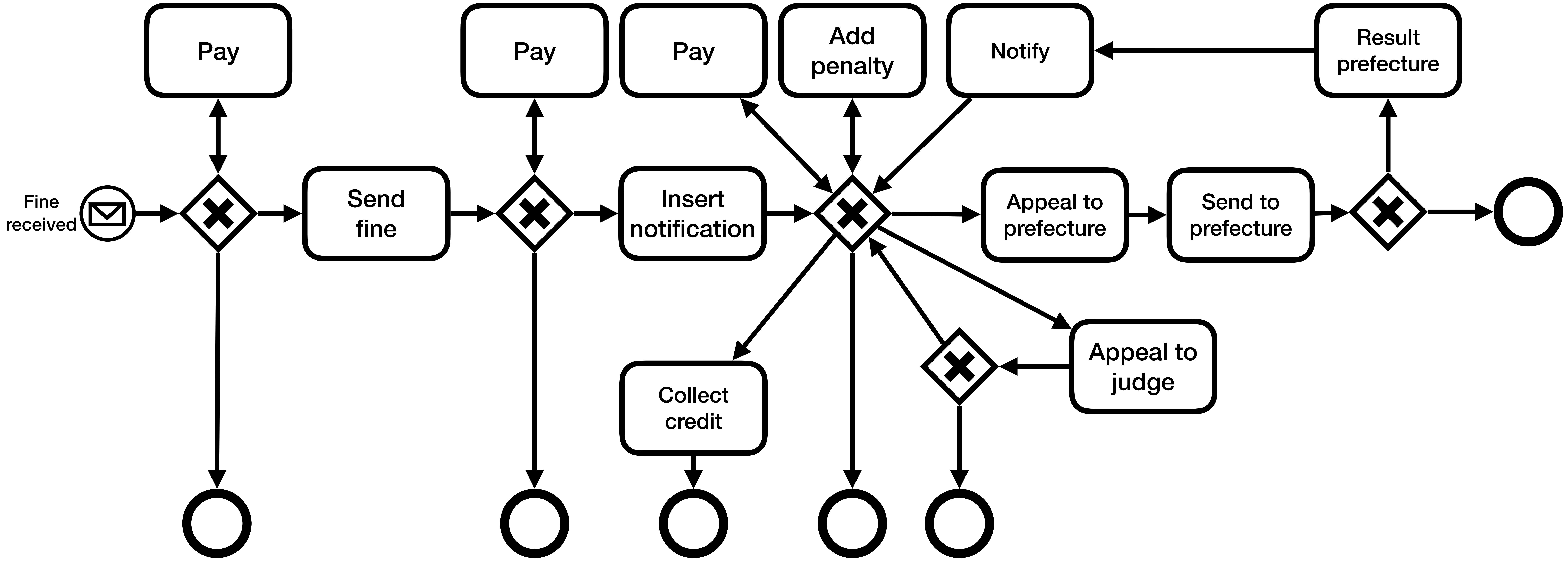
A holistic view of information systems

Starting point

A holistic view of information systems

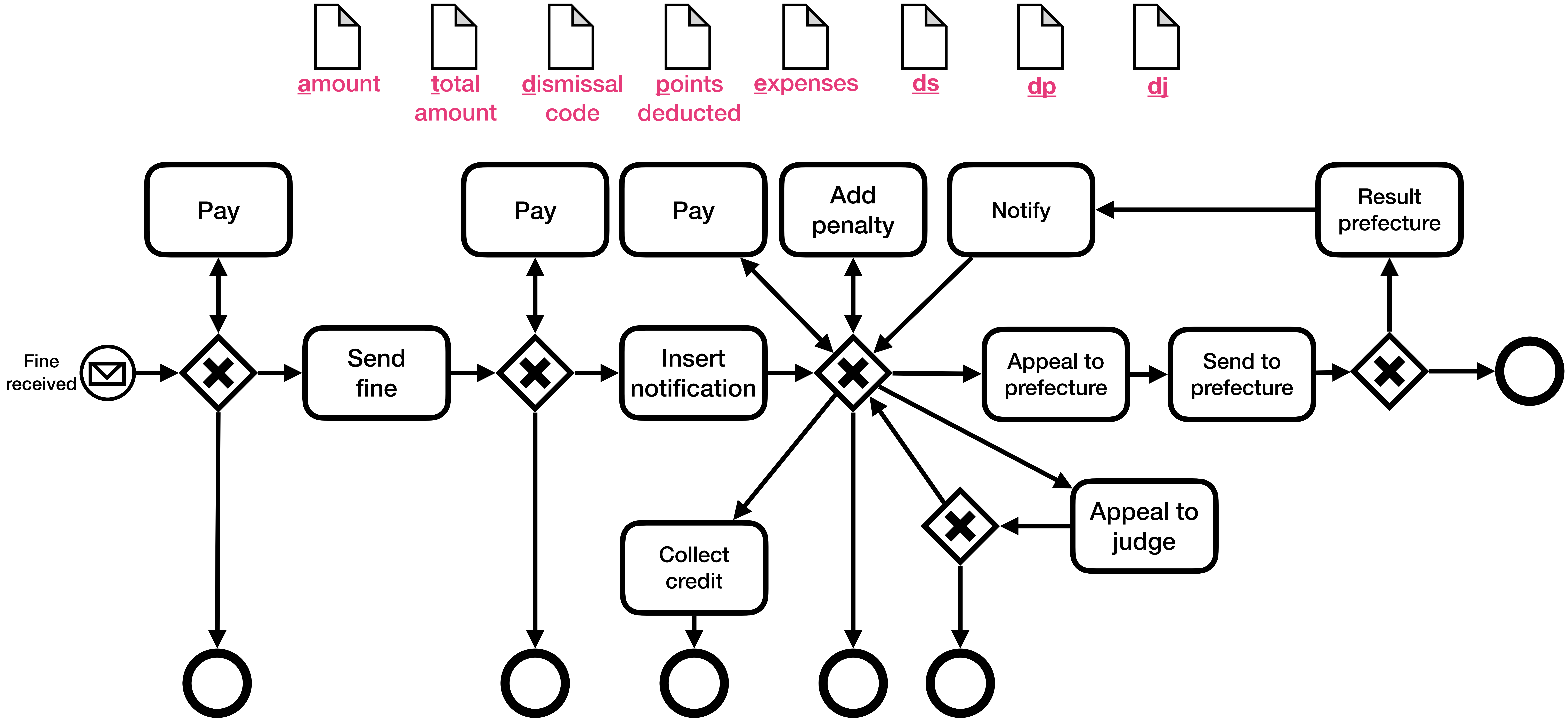
Why reasoning?

Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]



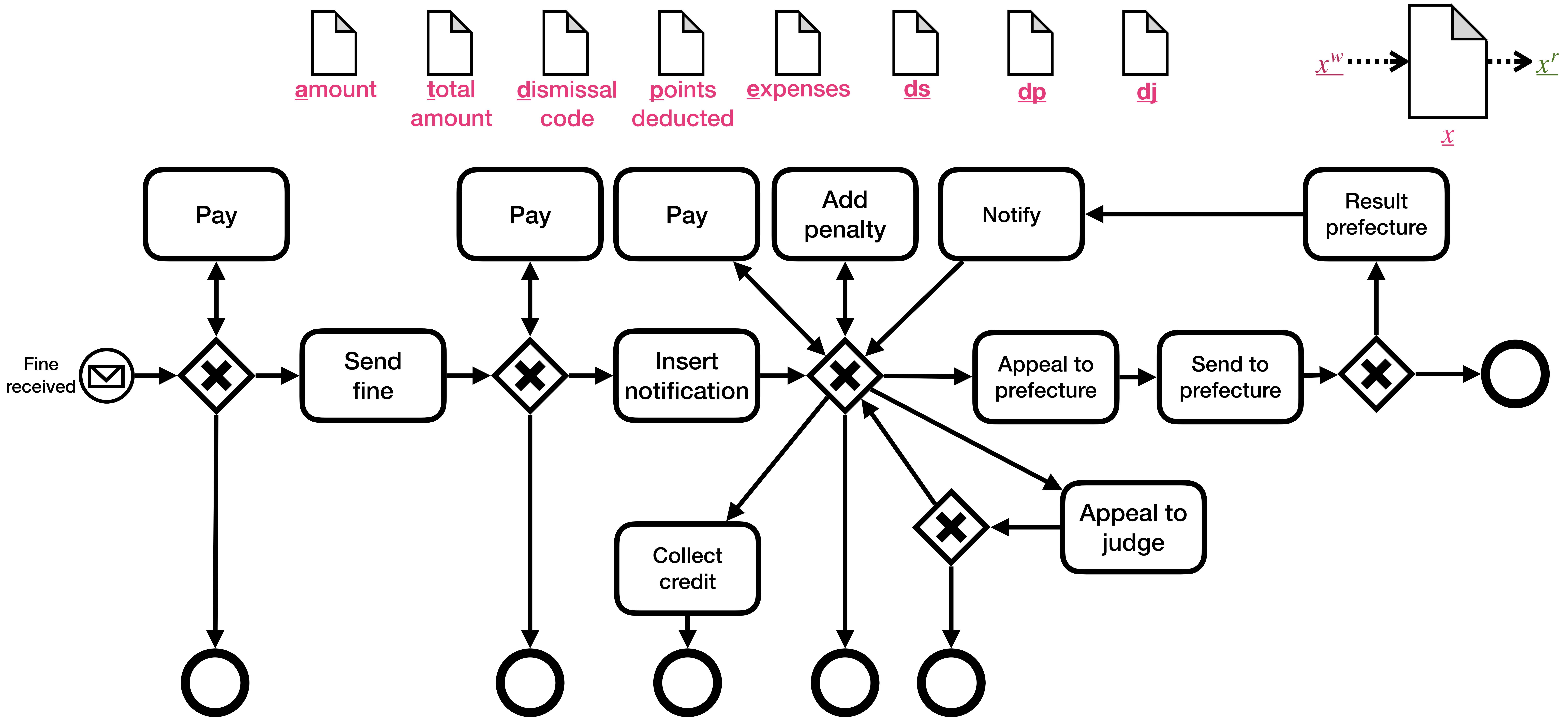
Why reasoning?

Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]



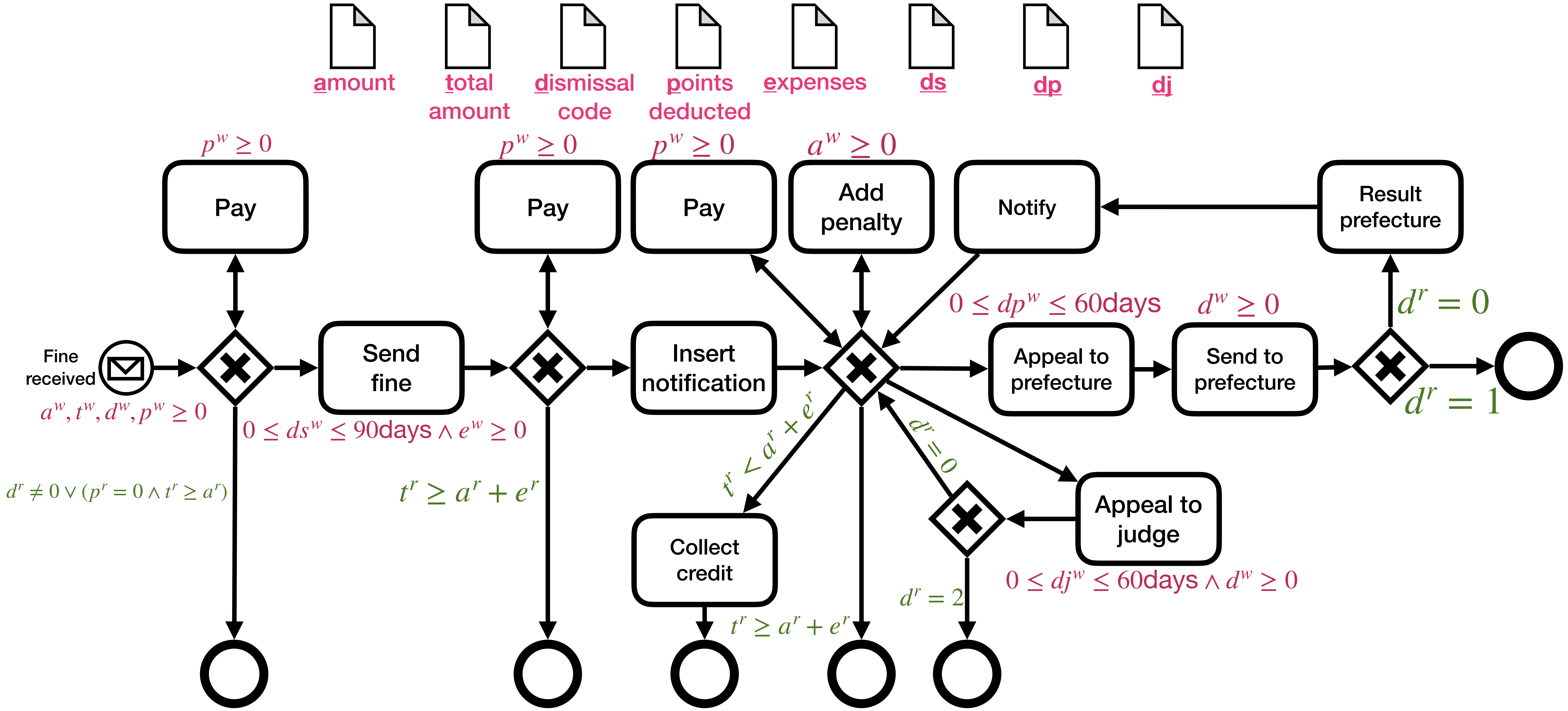
Why reasoning?

Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]



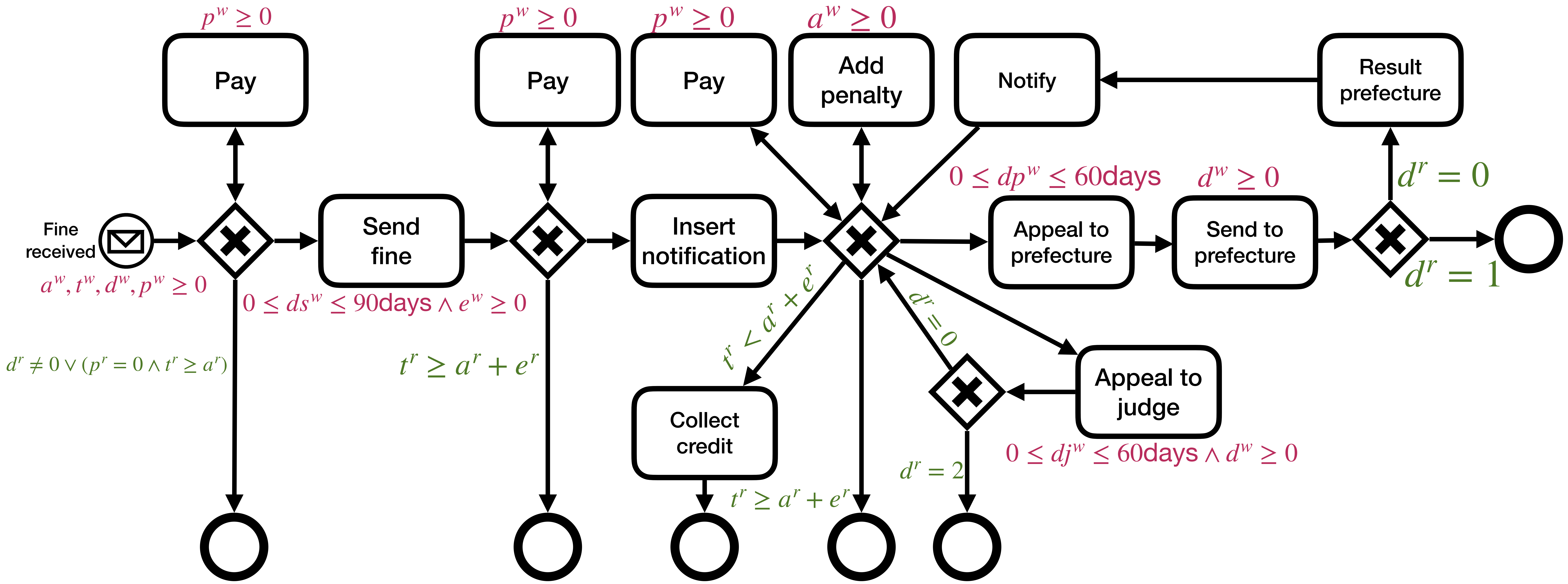
Why reasoning?

Adapted from [Mannhardt et al., Comput. 2016], studied in [____,CAiSE2022]



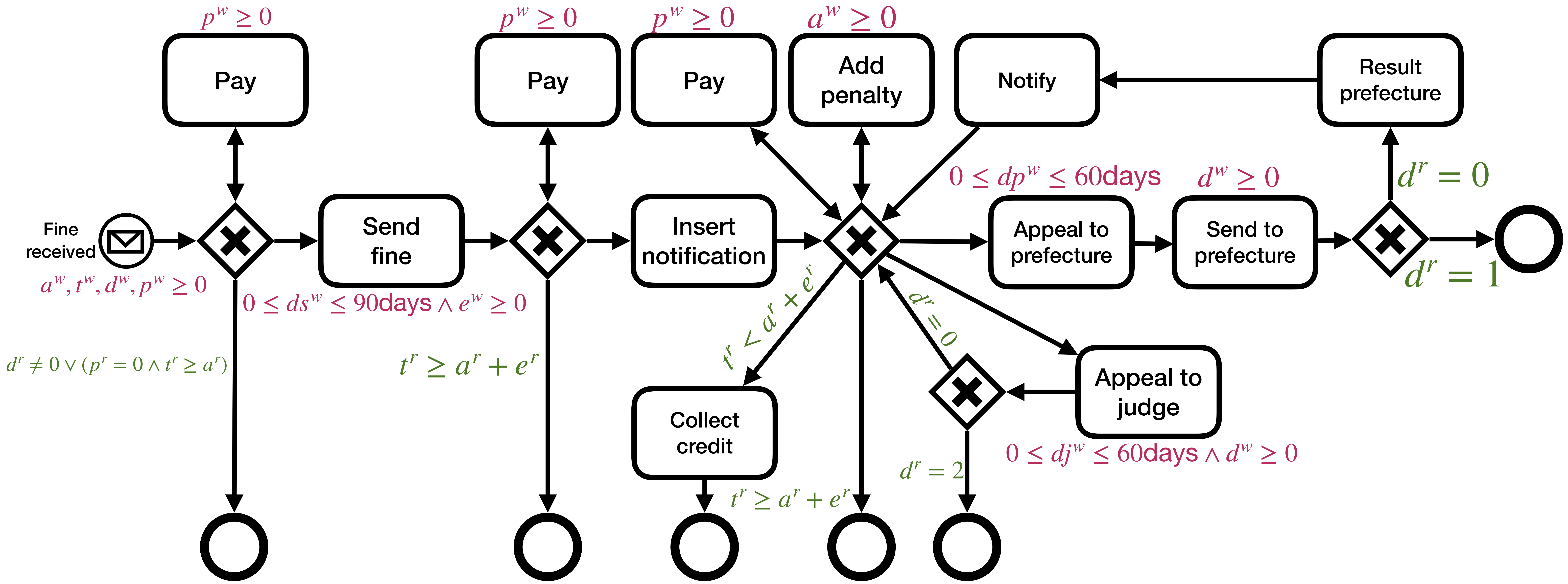
(multi-perspective) **mining** **modelling**

Is the model "correct"?



Is the model “correct”?

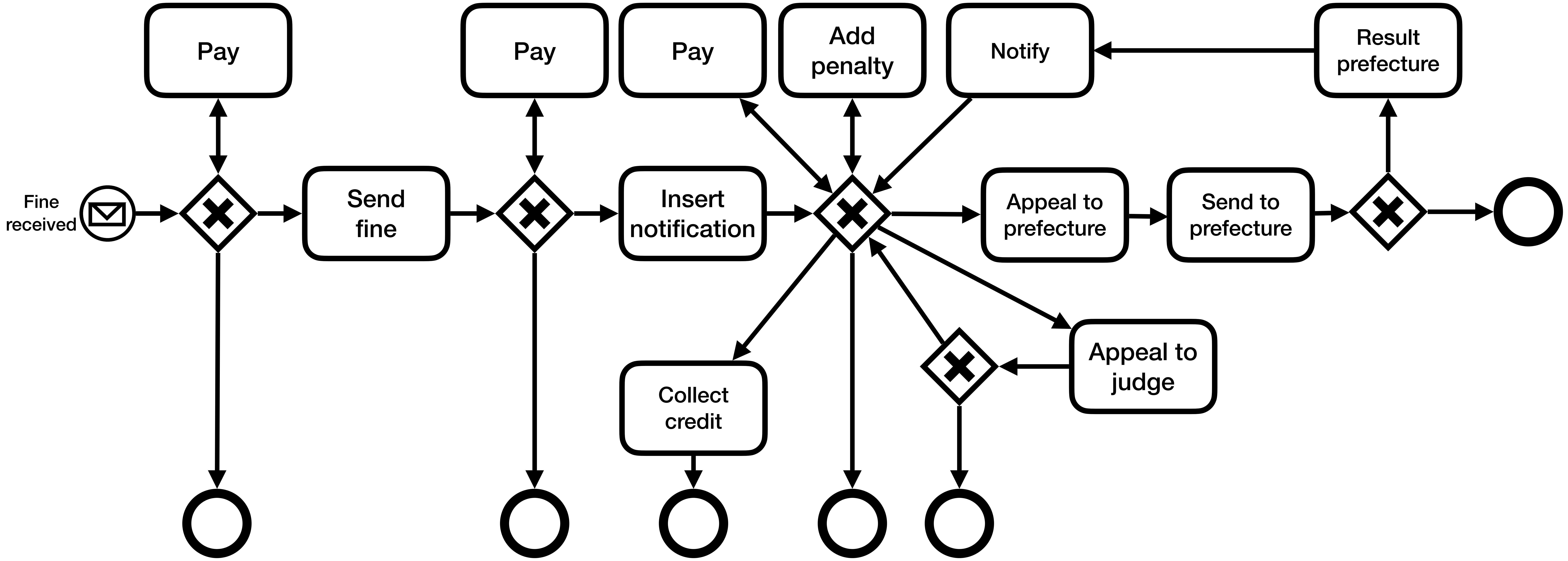
The control-flow way



Is the model “correct”?

The control-flow way

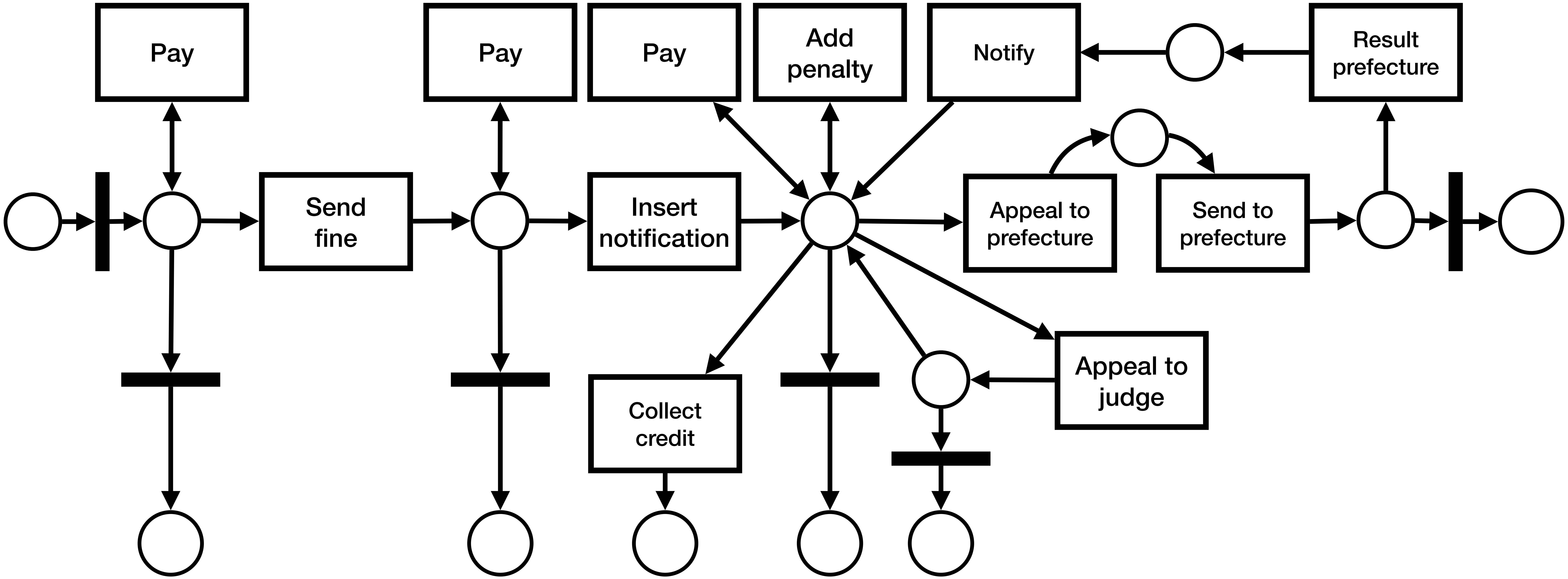
1. Strip-off the data



Is the model "correct"?

The control-flow way

1. Strip-off the data
2. Encode control-flow into bounded Petri net (finite state-space)

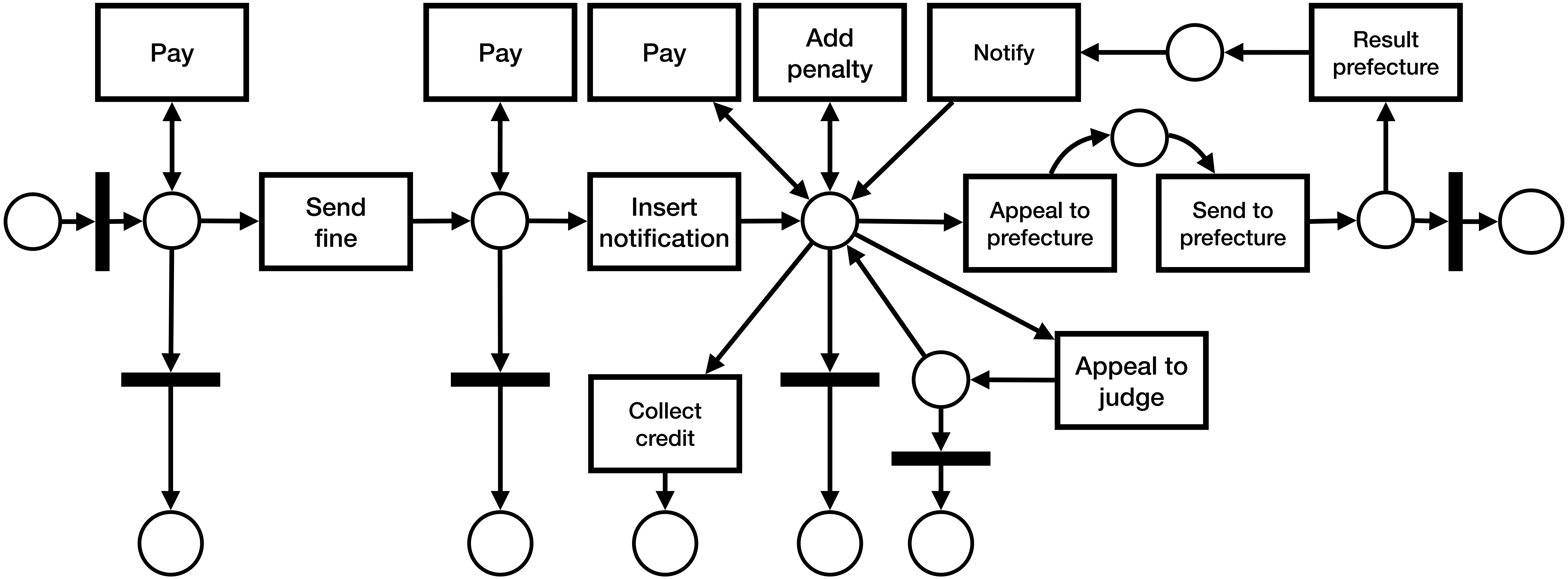


Is the model “correct”?

The control-flow way

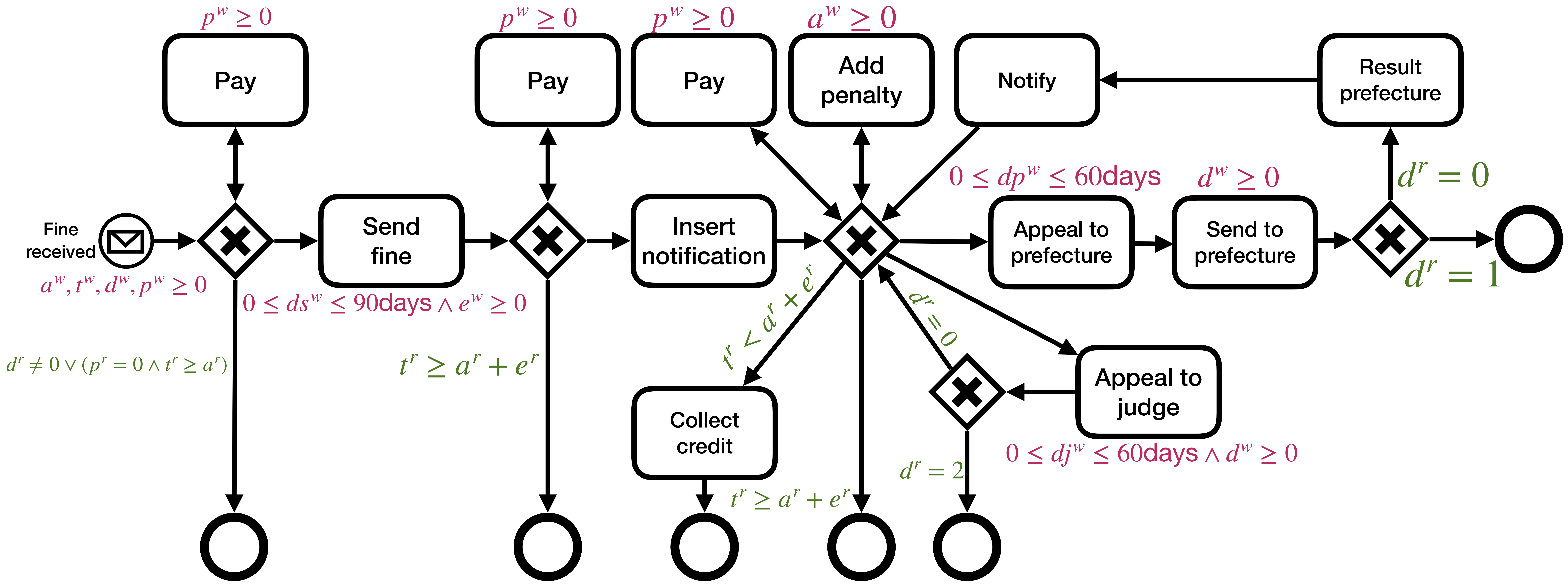
1. Strip-off the data
2. Encode control-flow into bounded Petri net (finite state-space)
3. Explore the state space

Verdict: all good!



Is the model "correct"?

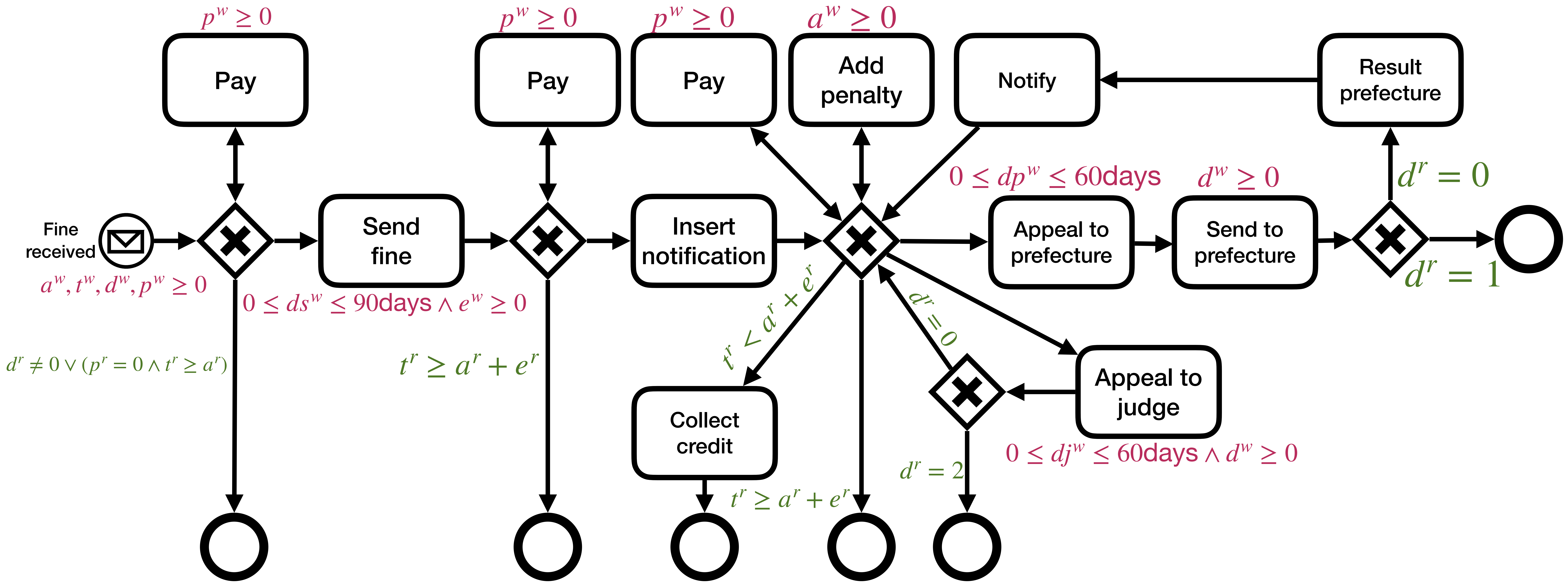
The integrated way



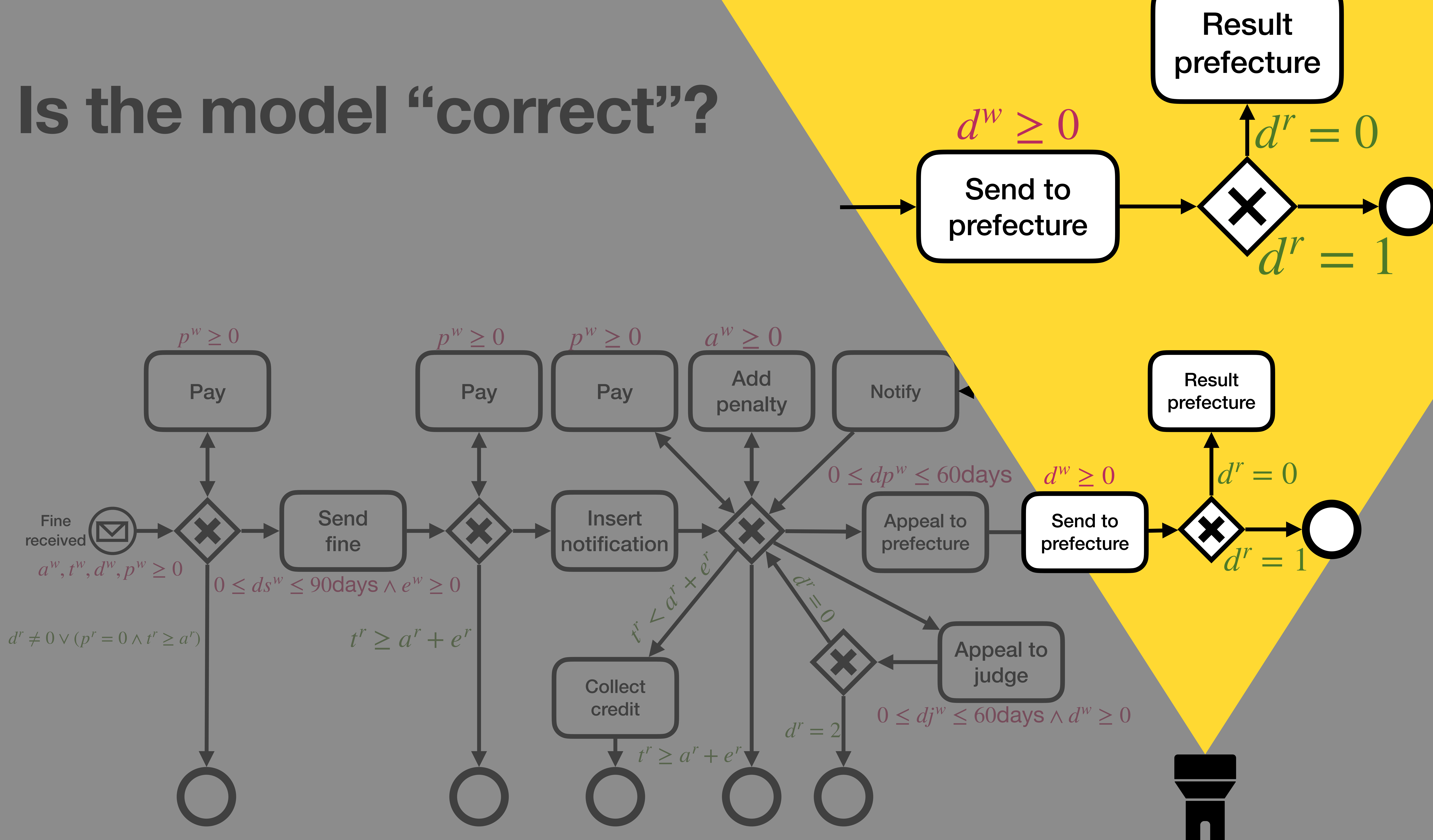
Is the model “correct”?

The integrated way

infinitely many runs with infinitely many distinct variable assignments



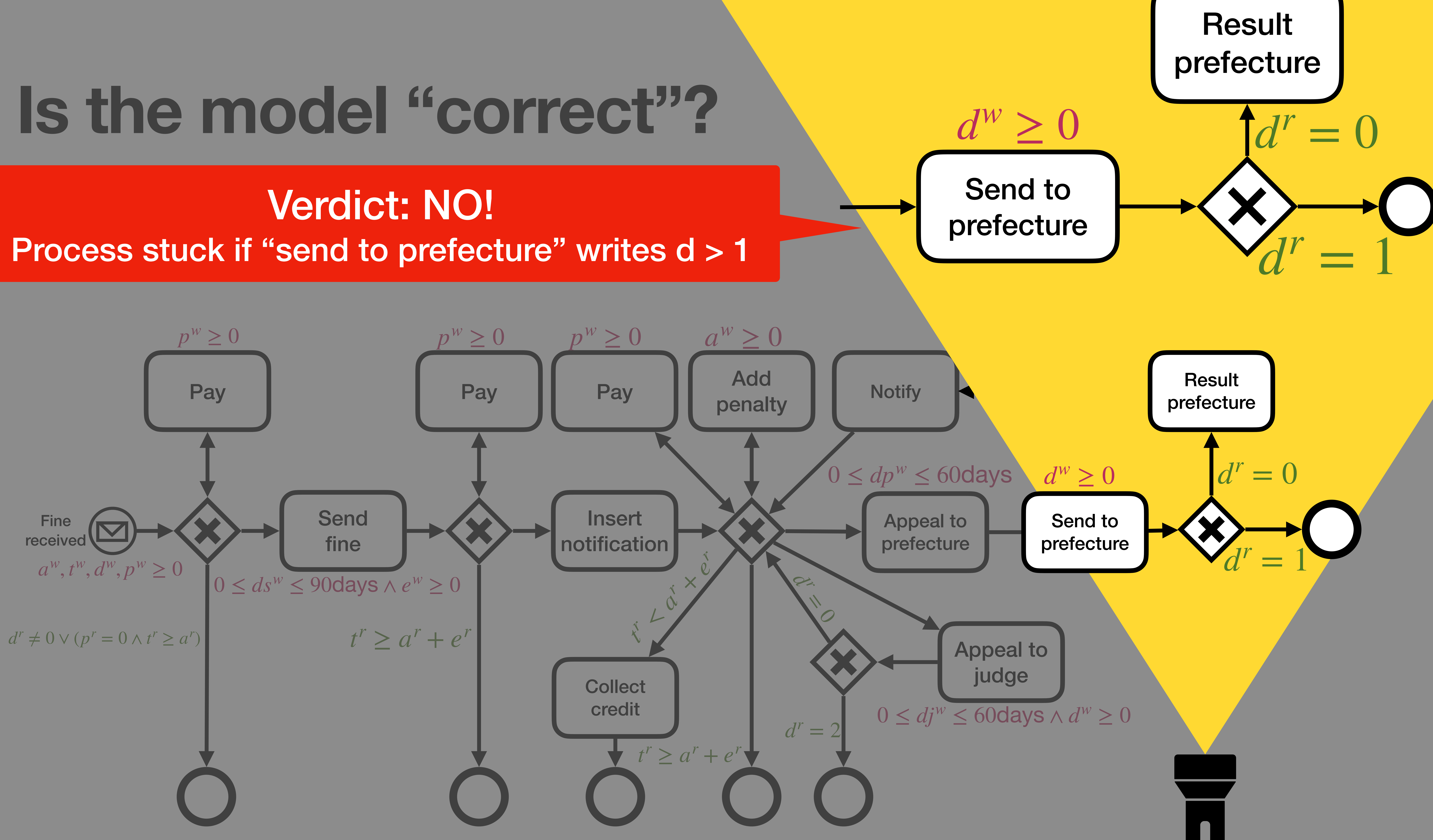
Is the model “correct”?



Is the model "correct"?

Verdict: NO!

Process stuck if "send to prefecture" writes d > 1

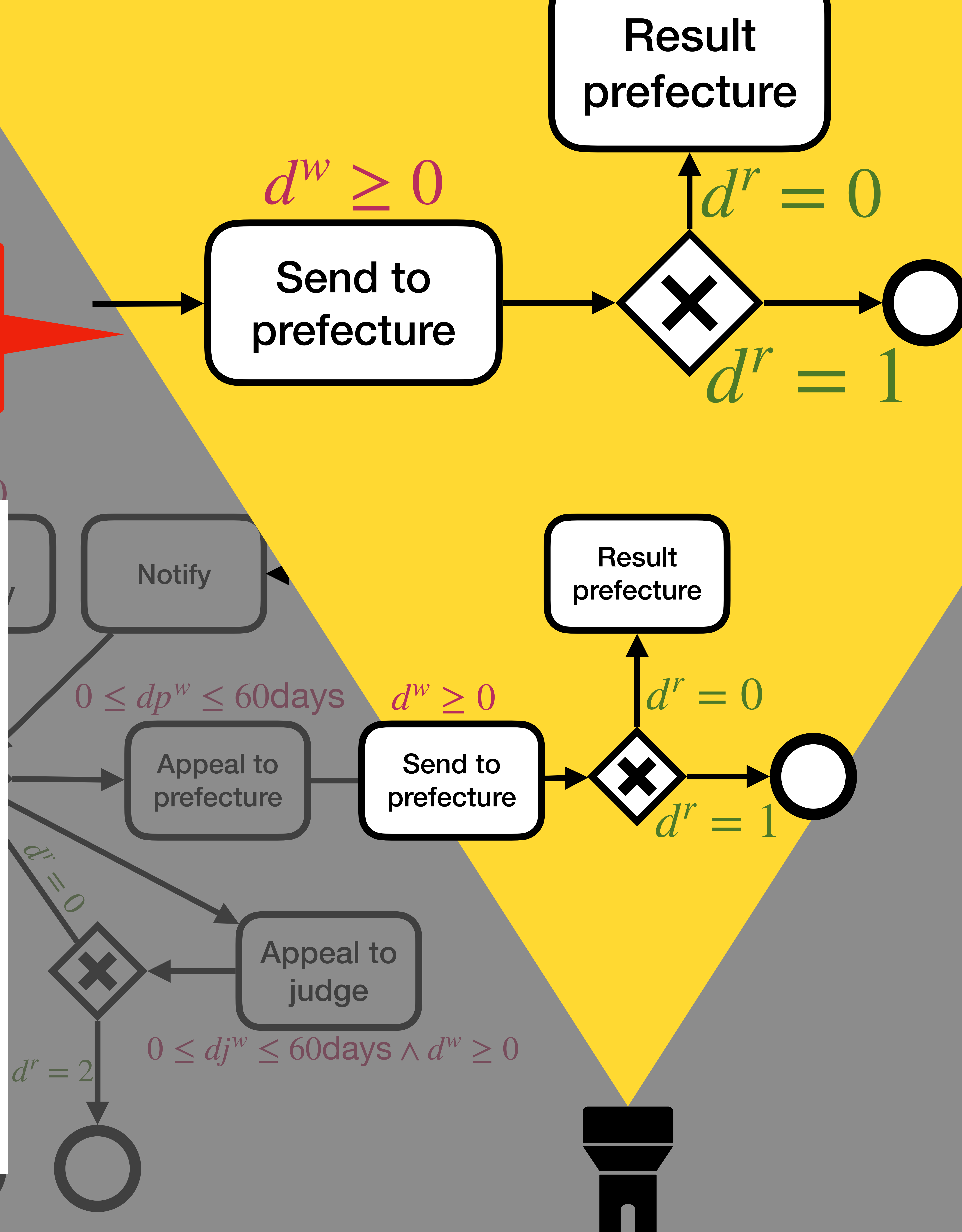


Is the model “correct”?

Verdict: NO!

Process stuck if “send to prefecture” writes d > 1

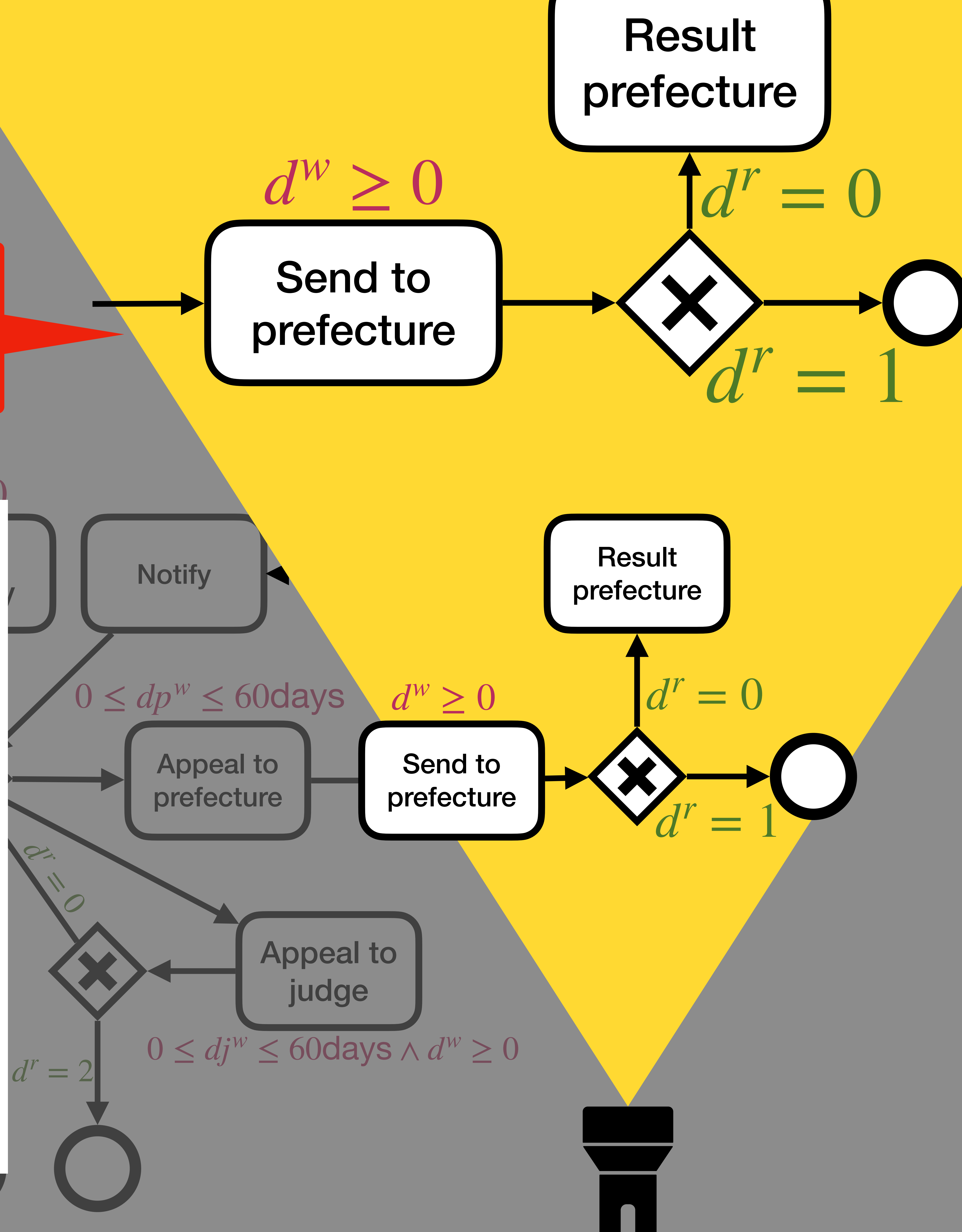
Issue: **blocked state**



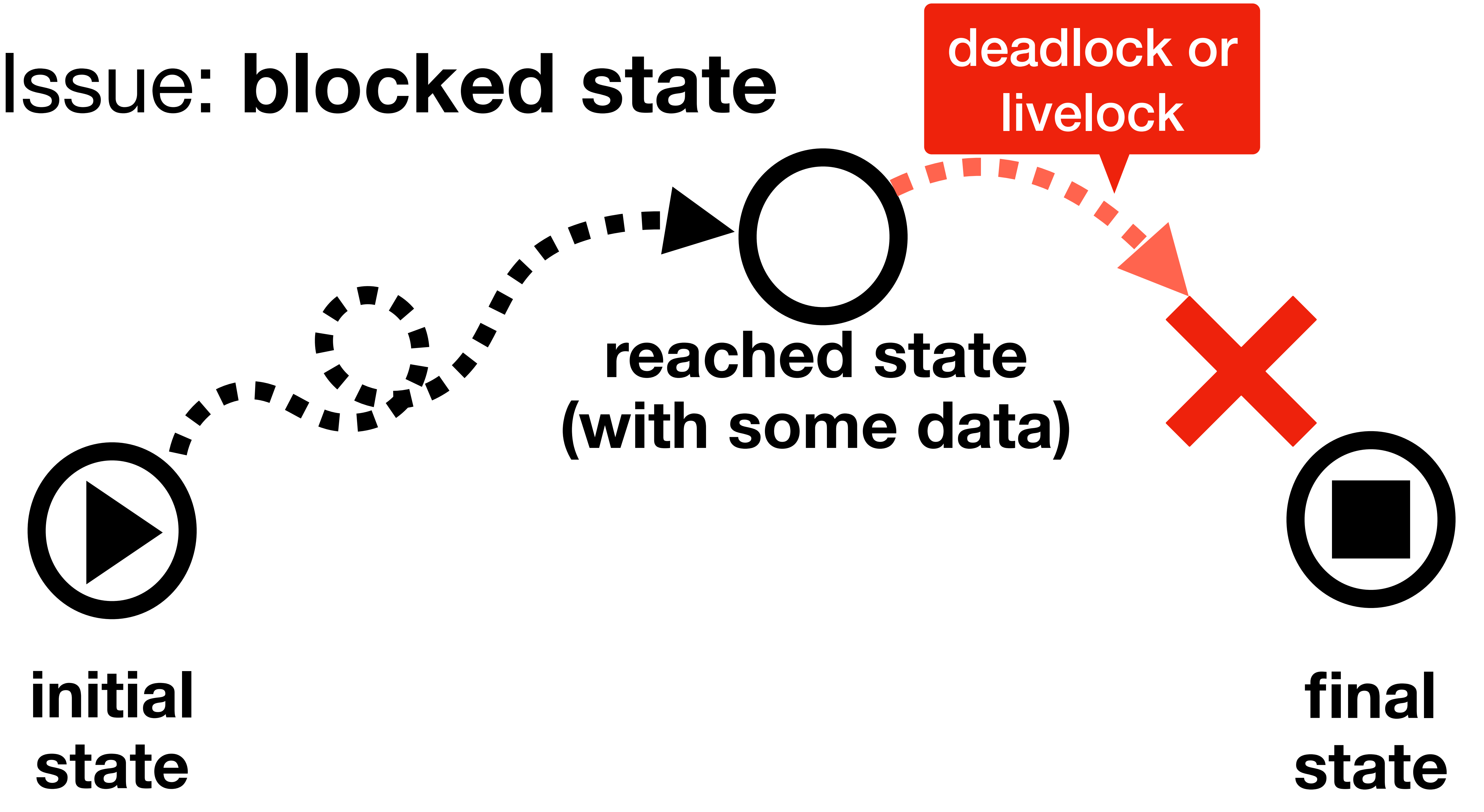
Is the model “correct”?

Verdict: NO!

Process stuck if “send to prefecture” writes d > 1



Issue: **blocked state**



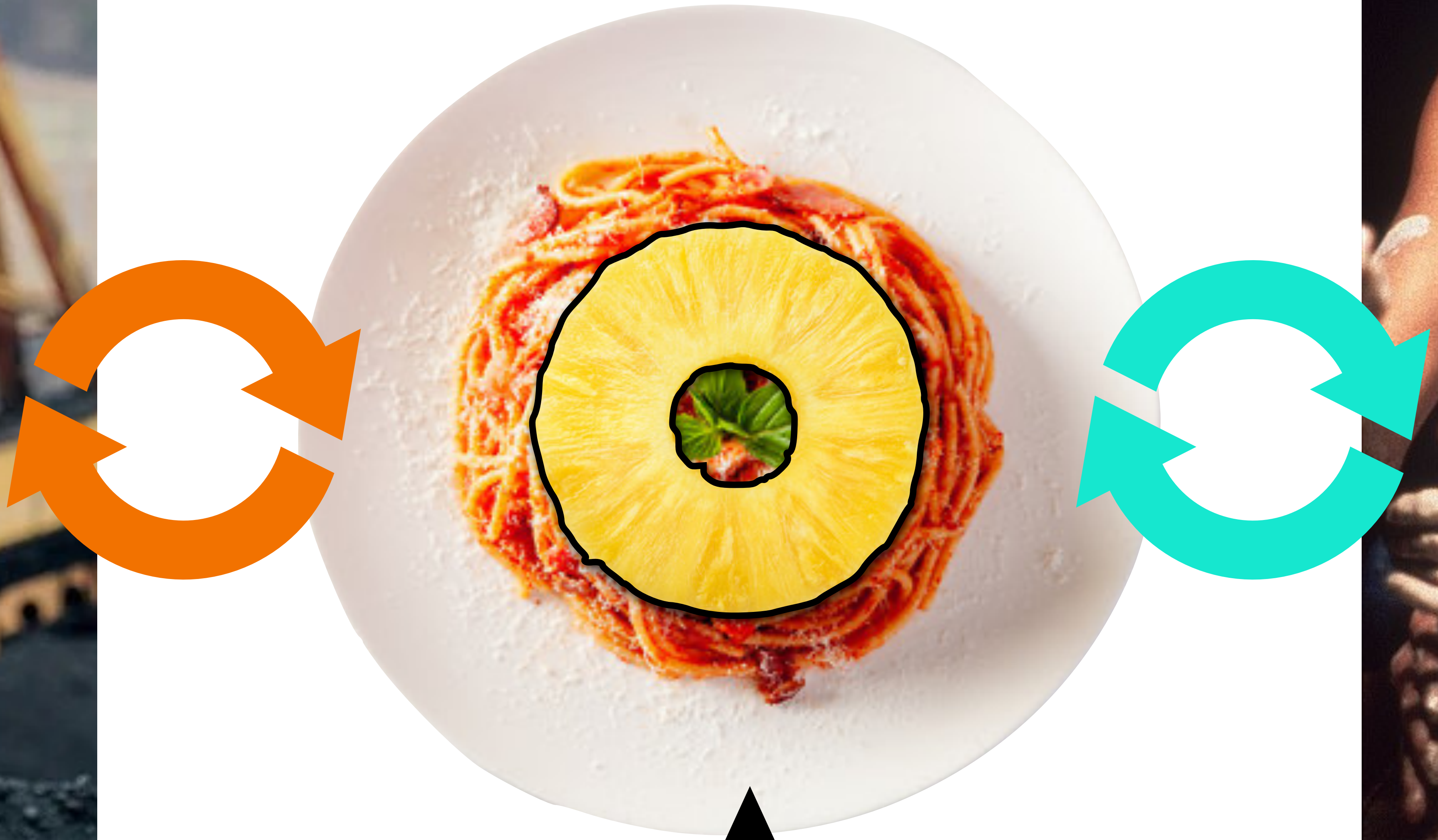
Process mining is an iterative process

Process mining is an iterative process

mining

Process mining is an iterative process

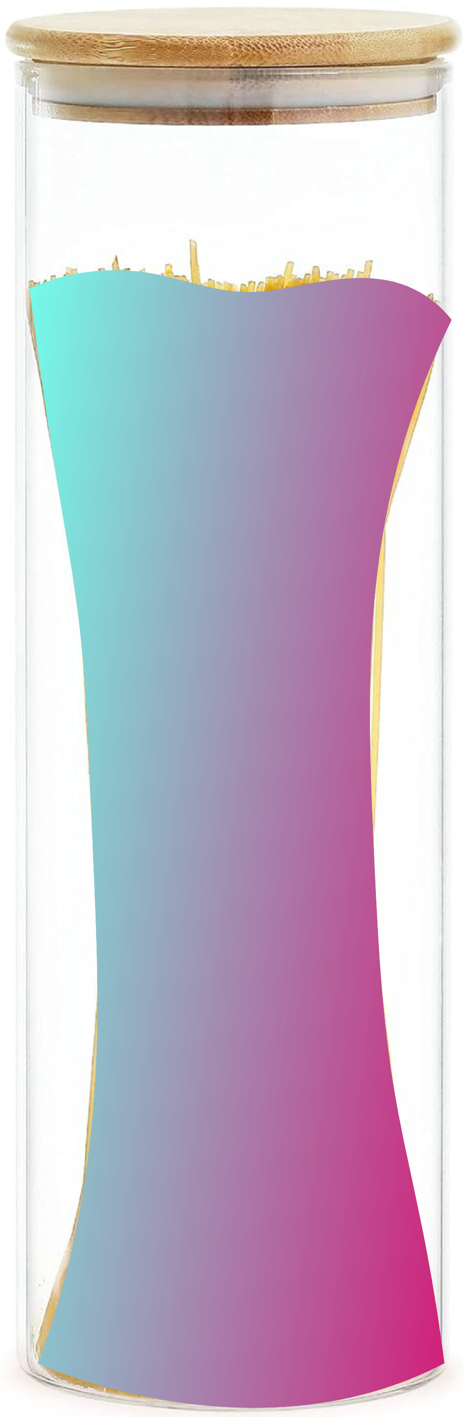
mining



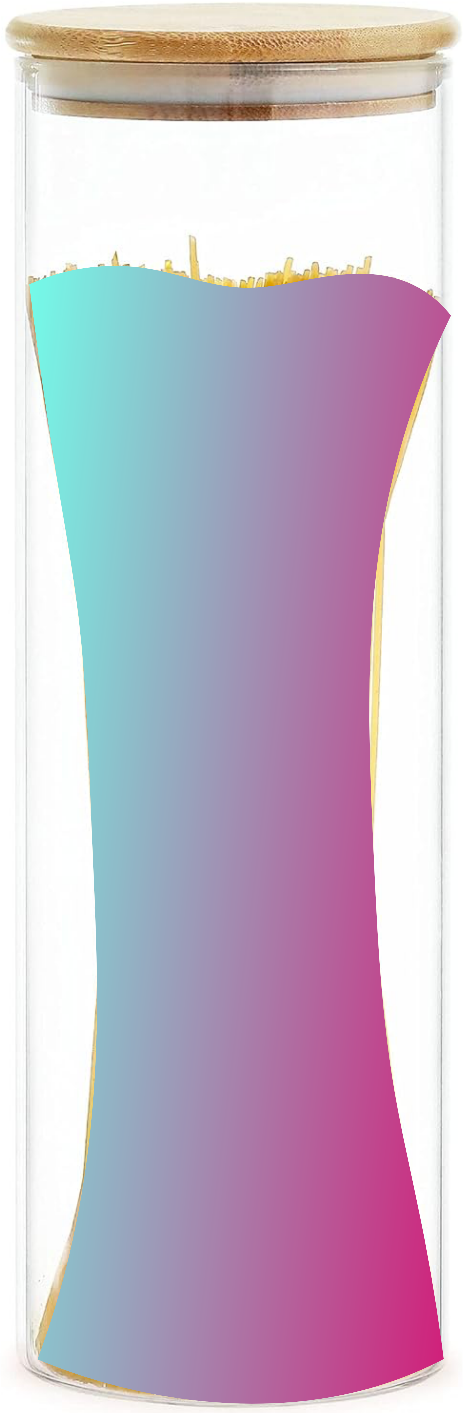
unsound!

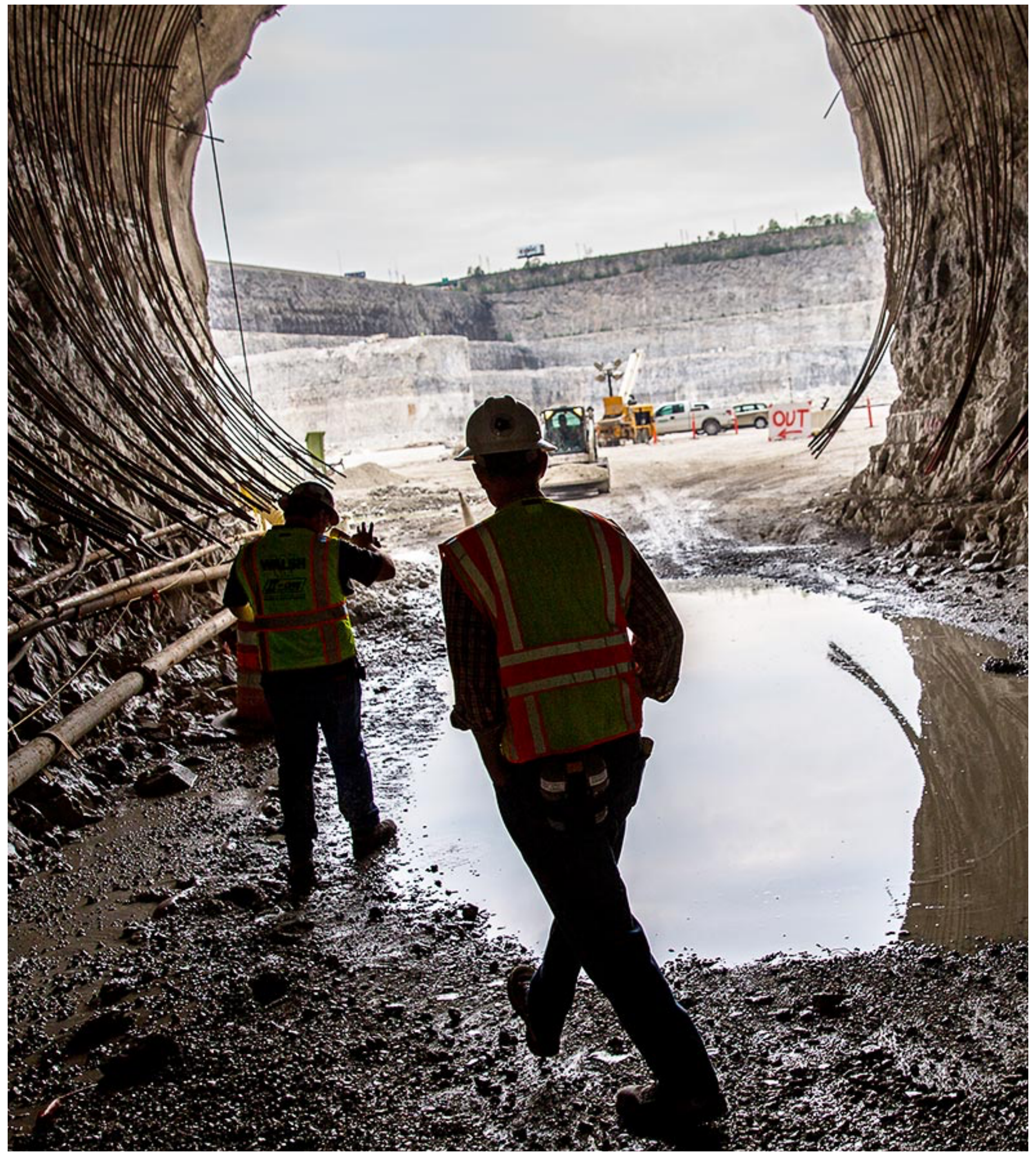
modelling

Data-aware process mining with separate techniques



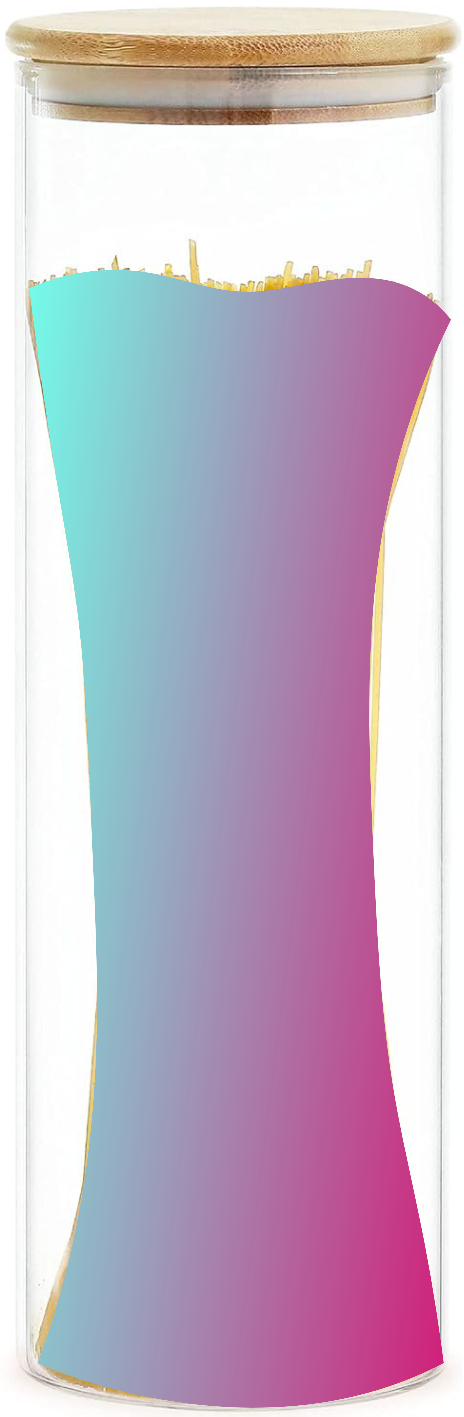
Data-aware process mining with separate techniques

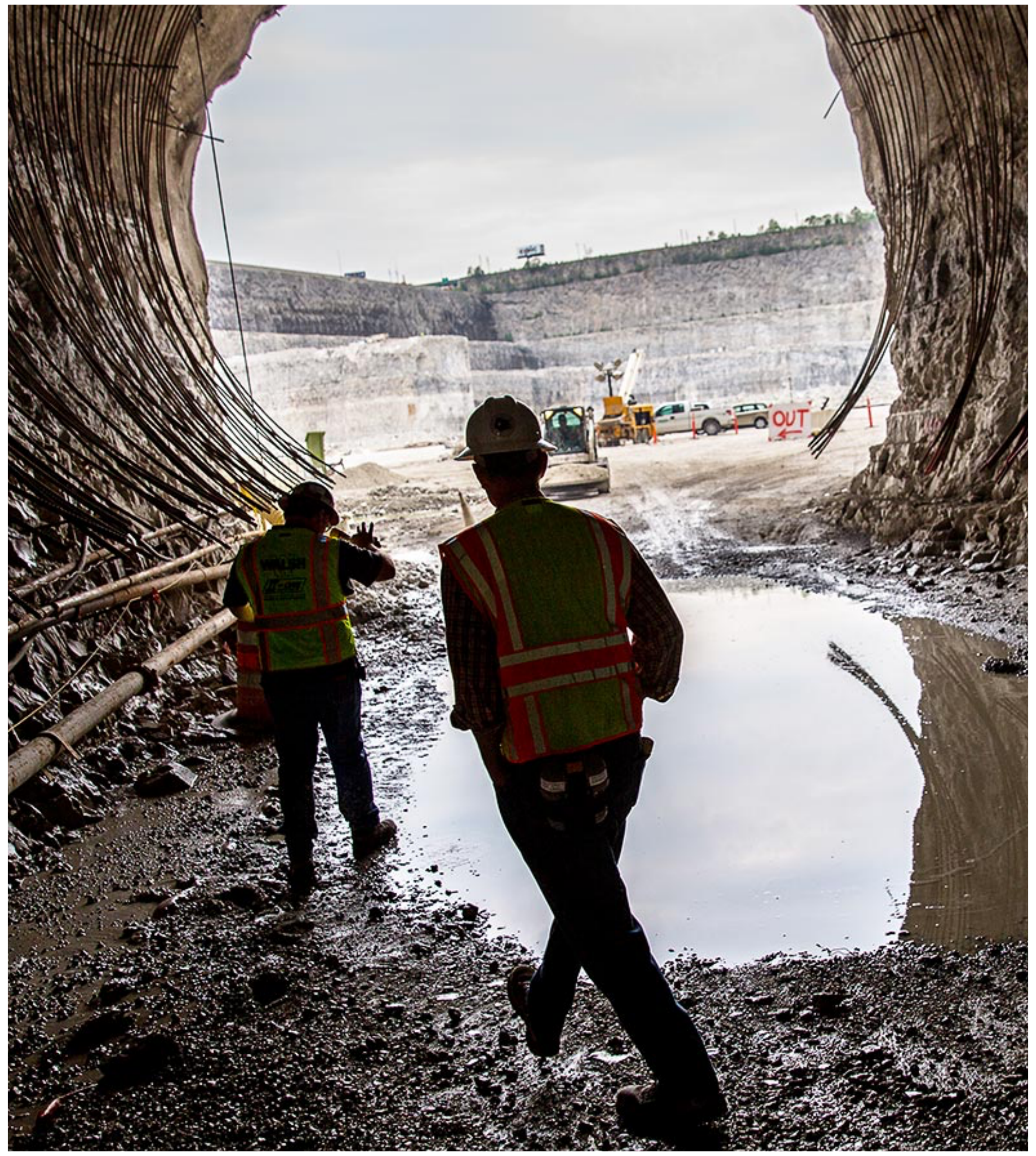




control-flow
mining

Data-aware process mining with separate techniques

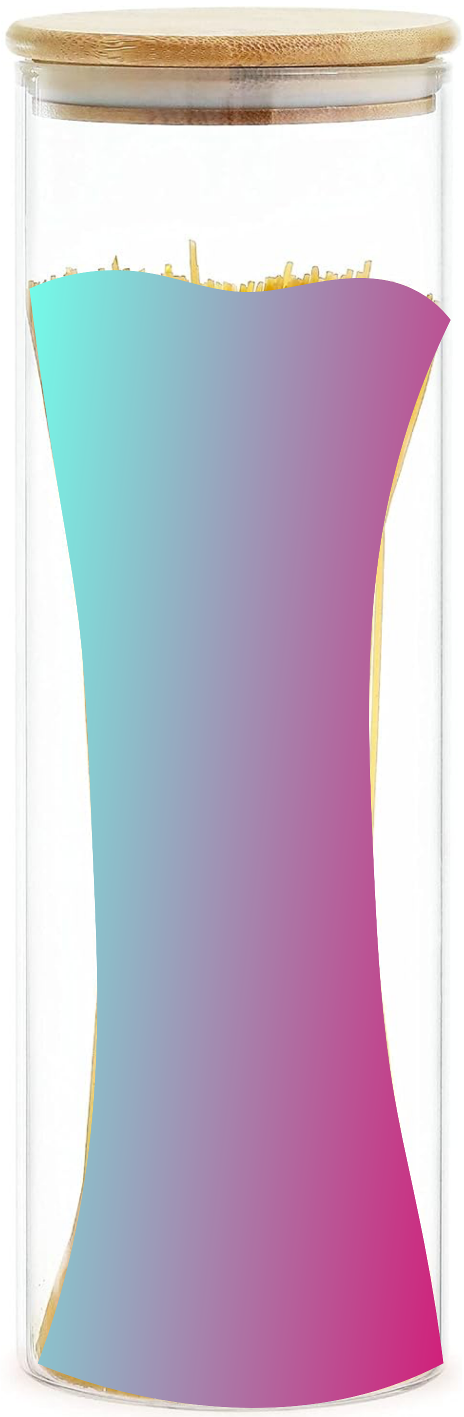


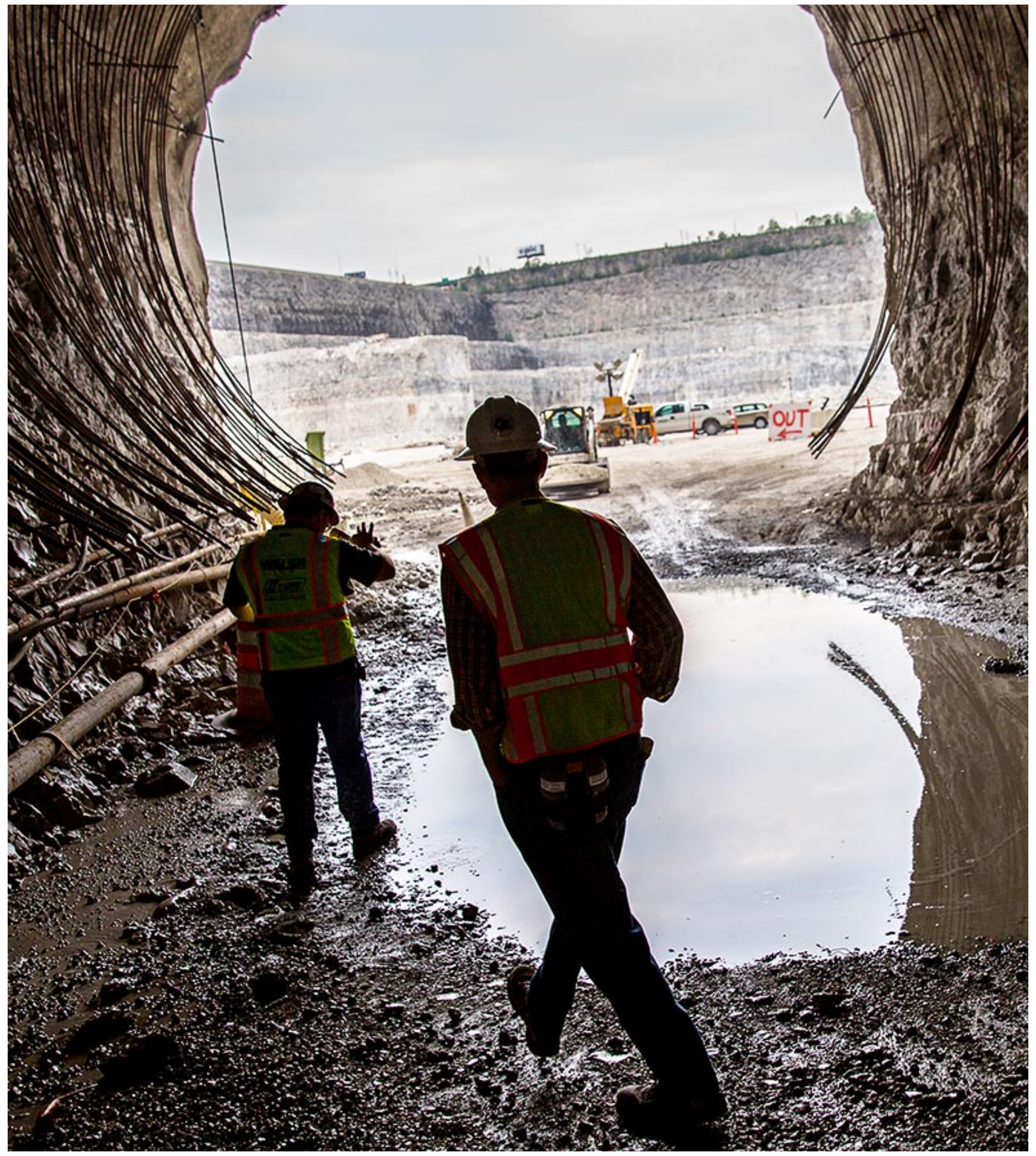


control-flow
mining

guards/decision
mining

Data-aware process mining with separate techniques

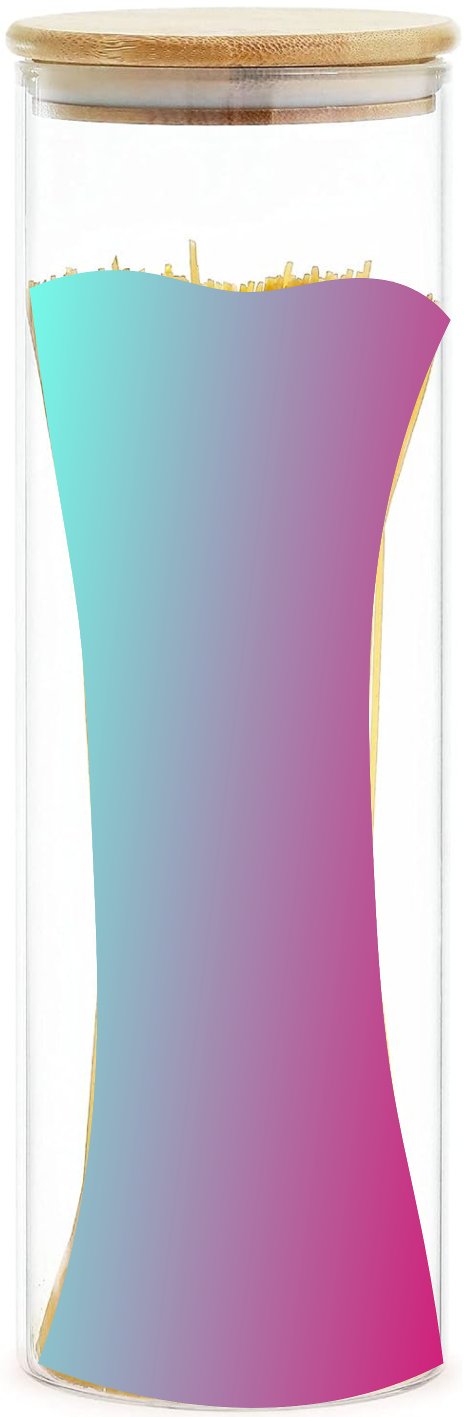


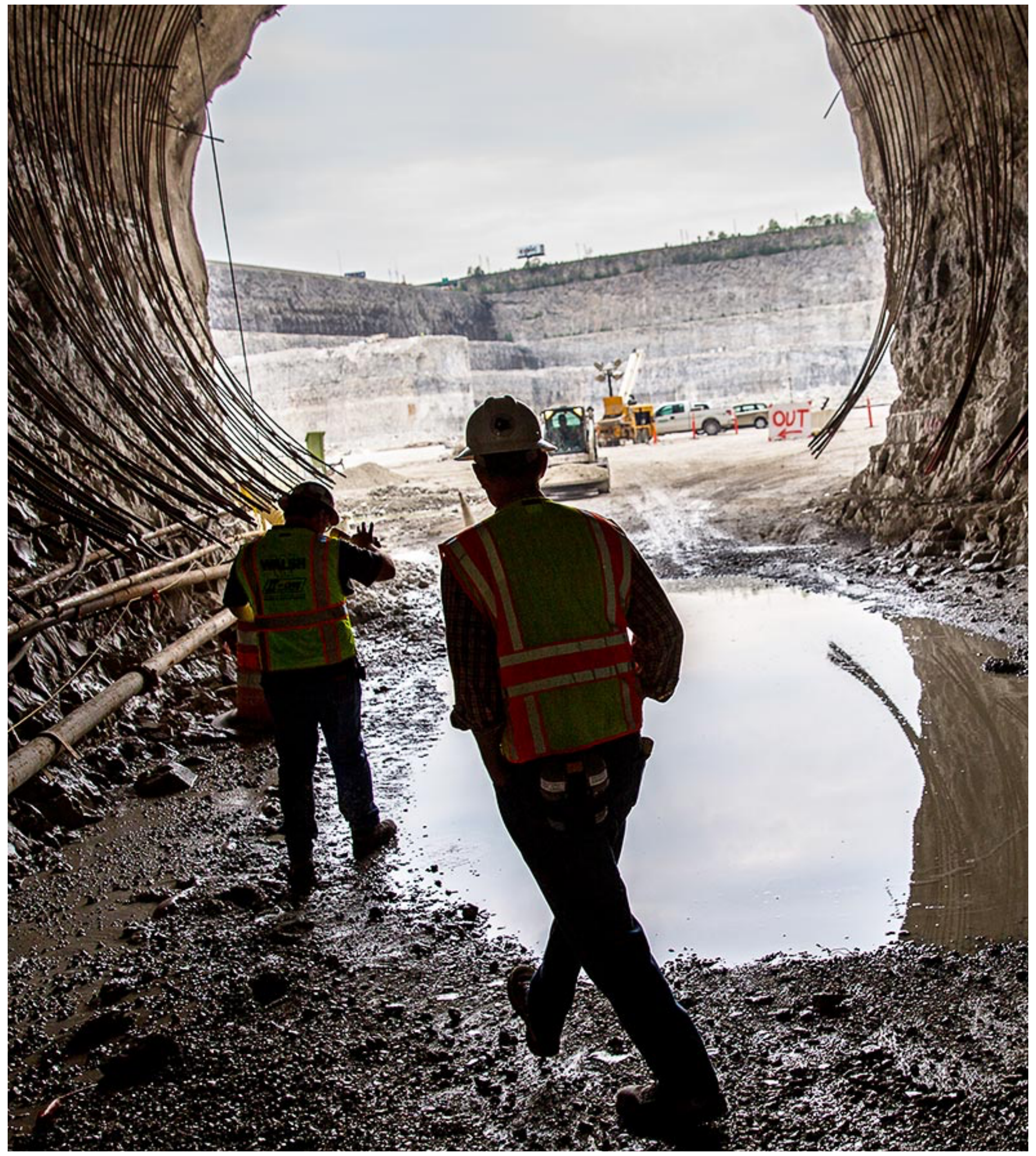


control-flow mining

guards/decision mining

Data-aware process mining with separate techniques





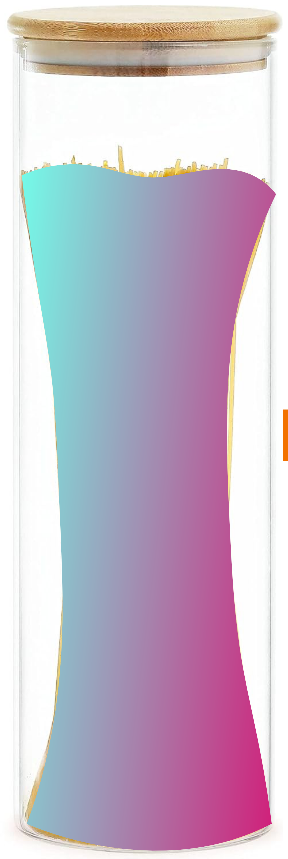
control-flow
mining

unsound!

guards/decision
mining

Data-aware process mining

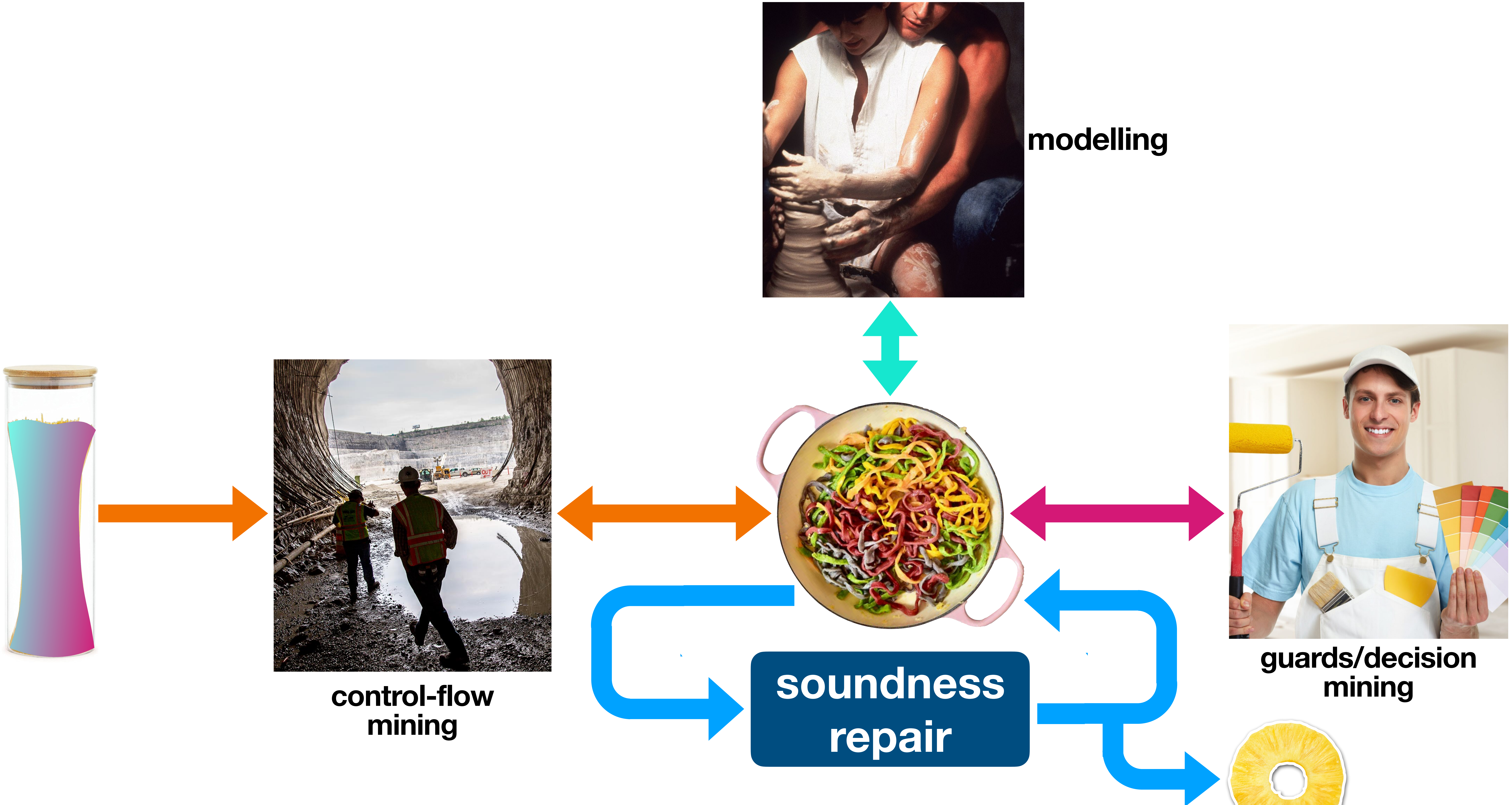
modelling



control-flow
mining

guards/decision
mining

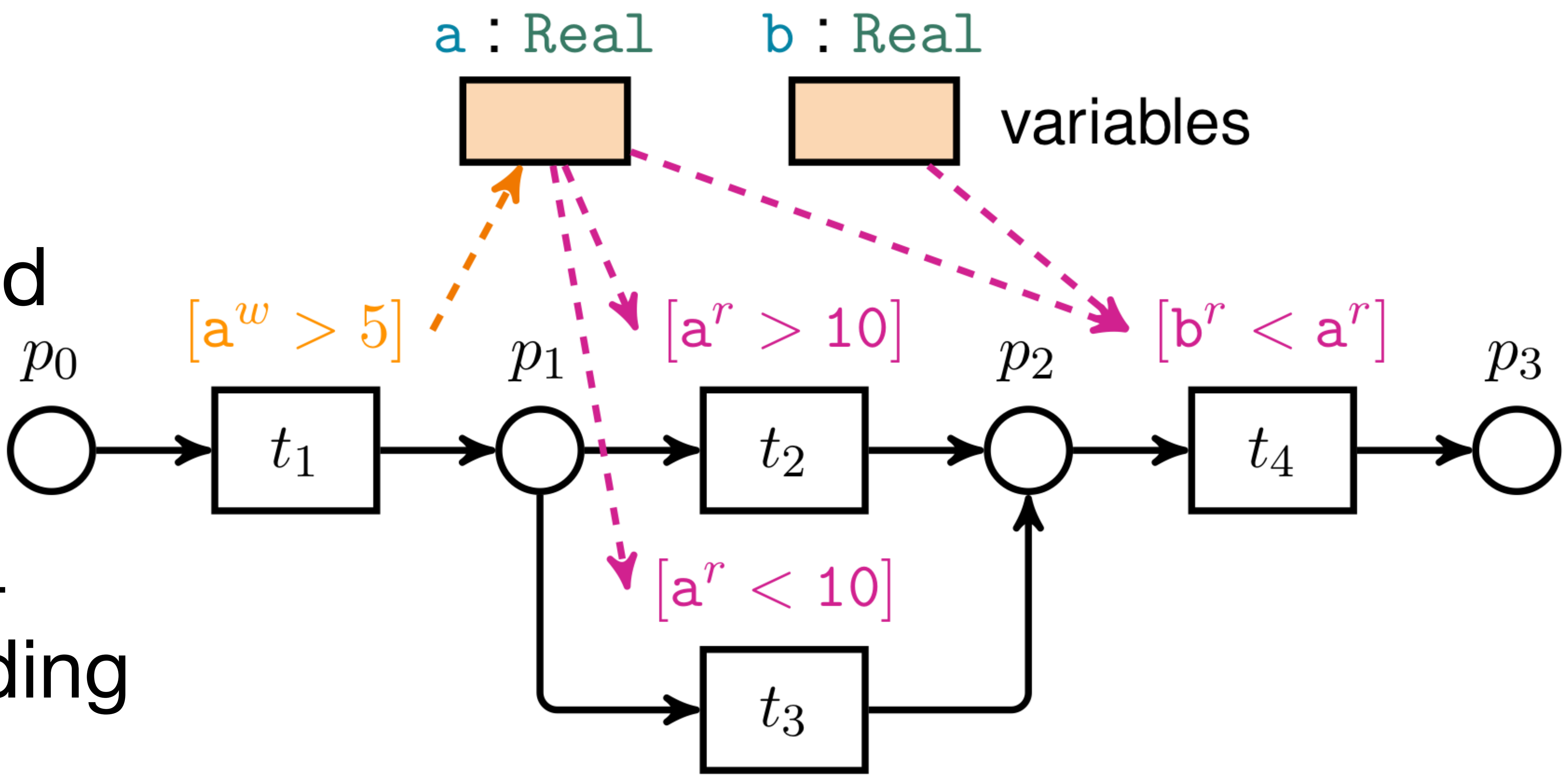
Data-aware process mining with reasoning



Data Petri Nets

[Mannhardt,PhD2018; _____,ER2018; _____,ACSD2019]

- Petri nets enriched with typed variables (ranging over infinite domains)
- Transitions access variables via **read** and **write guards**
- State: **marking + variable assignment**
- Transition firing: usual firing semantics + variable assignment update given a binding for the written variables



Infinite reachability graph even when the net is bounded

Possibility of reasoning depends on the guard language

Fragile setting: undecidability around the corner!

Goal

data-aware unsound DPN
N

has blocked states

Data-aware soundness checks: using [____,CAiSE 2022]

soundness repair

data-aware sound DPN
“minimally adapted”
from N

Assumptions

data-aware unsound DPN
N

has blocked states

soundness repair

data-aware sound DPN
“minimally adapted” from N

Assumptions

1.Underlying Petri net
(without data) is sound

data-aware unsound
DPN
N

has blocked
states

soundness
repair

data-aware sound DPN
“minimally adapted”
from N

Assumptions

1.Underlying Petri net (without data) is sound

data-aware unsound DPN
N

has blocked states

soundness repair

data-aware sound DPN "minimally adapted" from N

2.Guard language in a fragment where soundness can be checked [____,CAiSE 2022]
E.g.: variable-to-constant guards

Assumptions

1.Underlying Petri net (without data) is sound

3.Does not modify control structure, only guards

data-aware unsound DPN
N

has blocked states

soundness repair

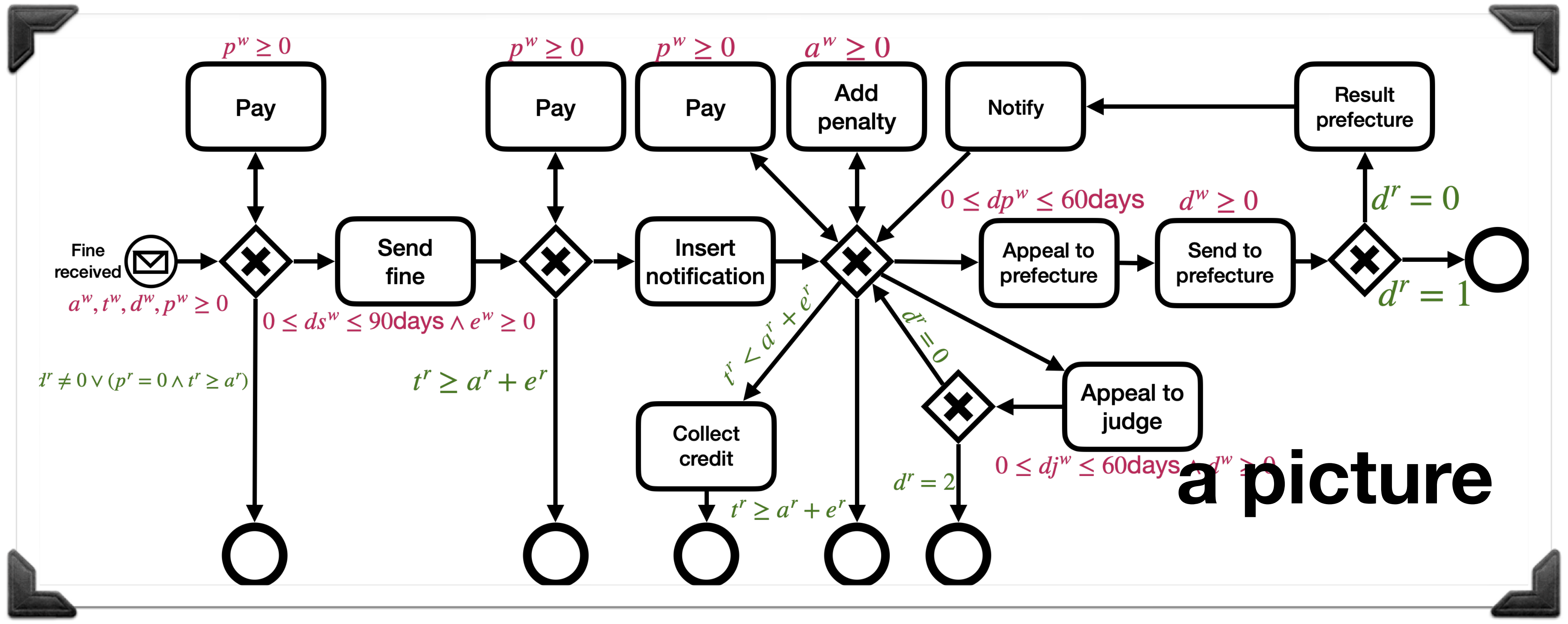
data-aware sound DPN
“minimally adapted” from N

2.Guard language in a fragment where soundness can be checked [____,CAiSE 2022]
E.g.: variable-to-constant guards

The two views of a process model…

… and what "minimality" means!

process representation



a picture

Minimal number of interventions on guards

[Zavatteri et al., FM-BPM 2023]

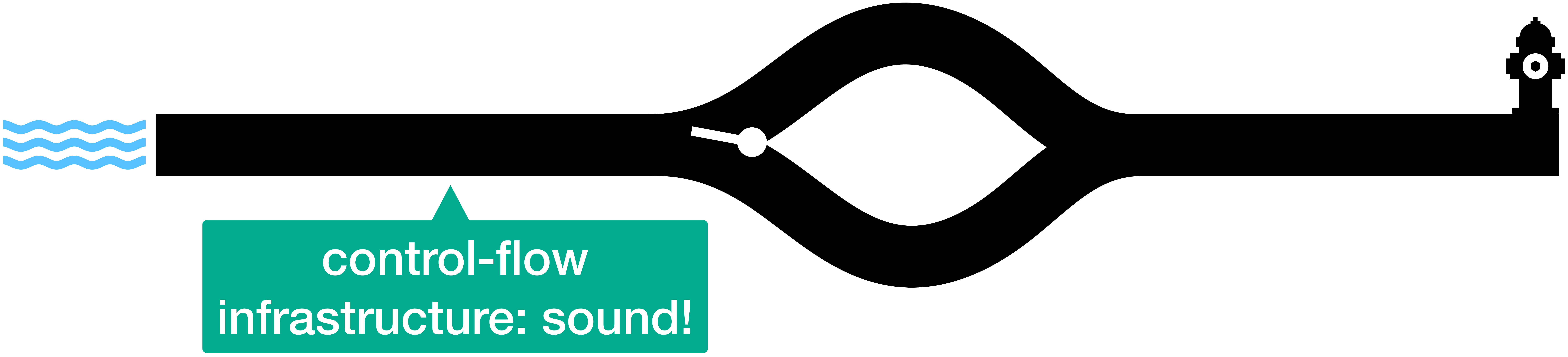
process model

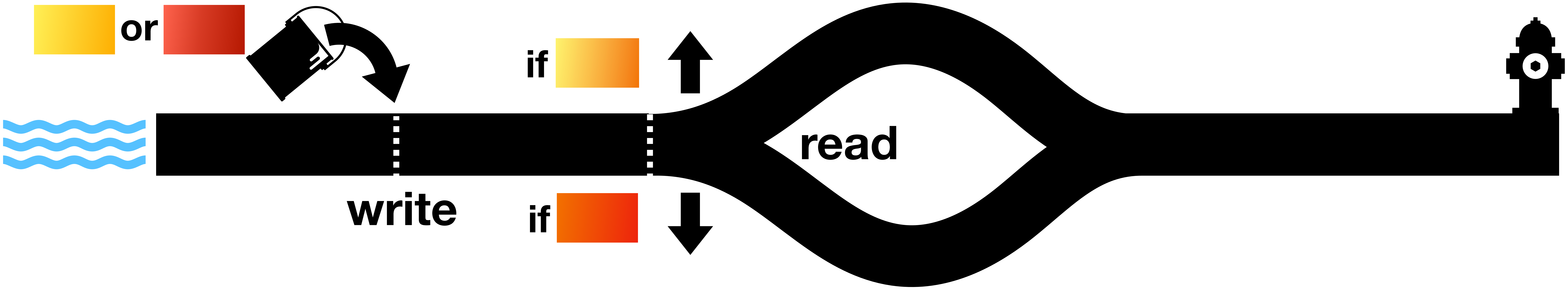
process behaviors

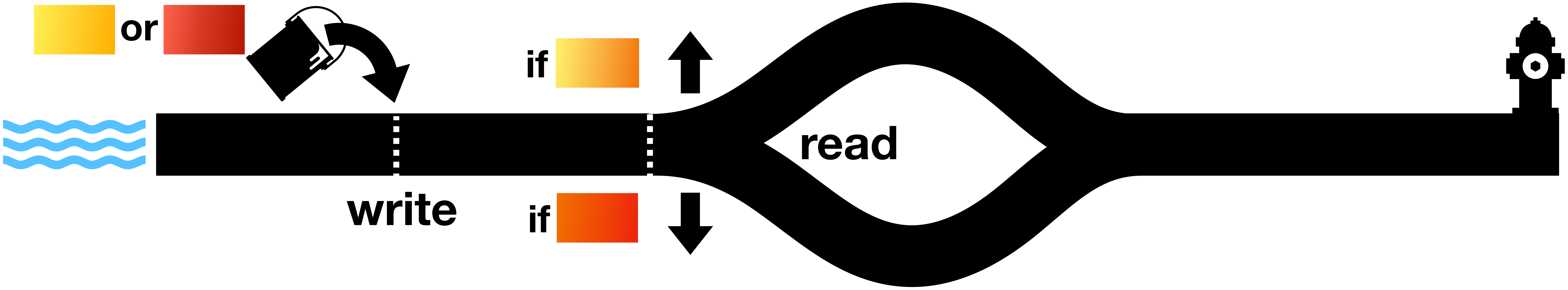
⋮ infinitely many traces

Our approach: minimal impact on behavior

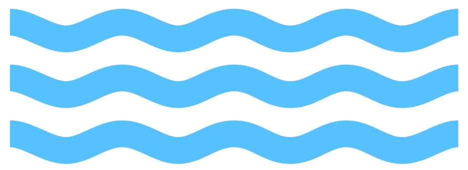
We only impact traces leading to a blocked state

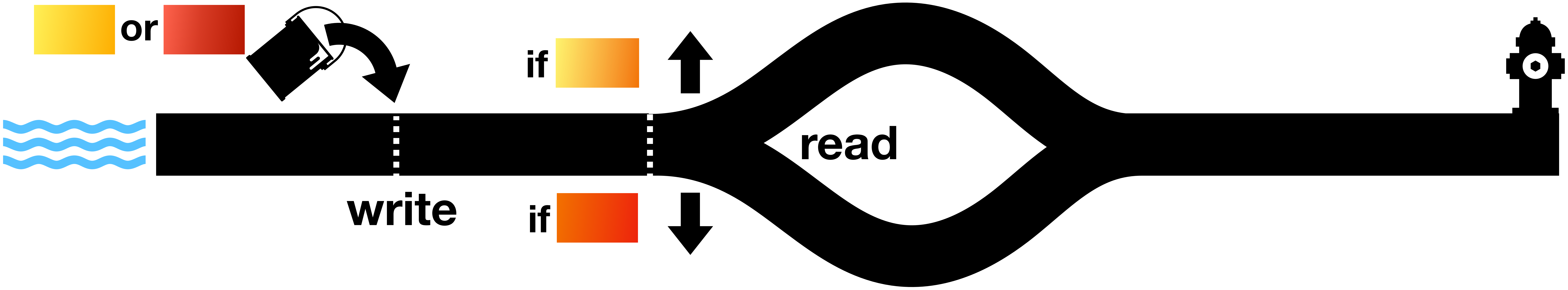

control-flow
infrastructure: sound!


or
if
write
read
if

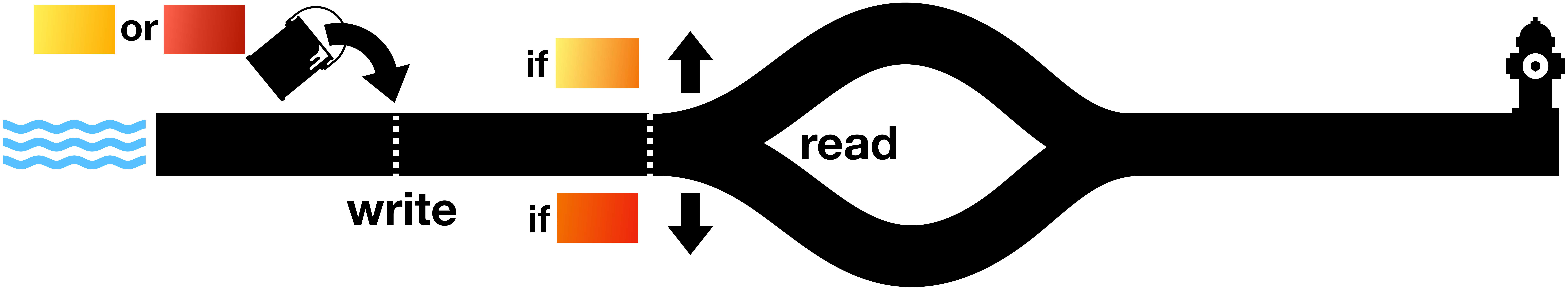


execution

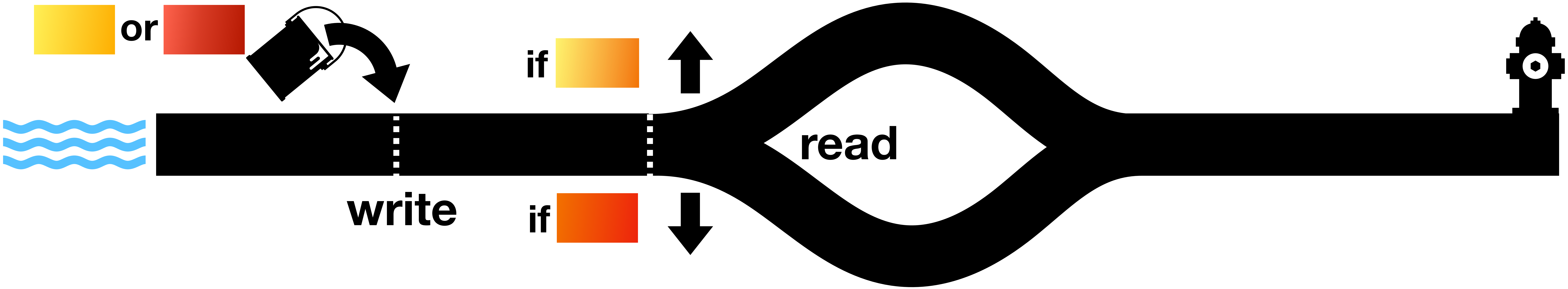




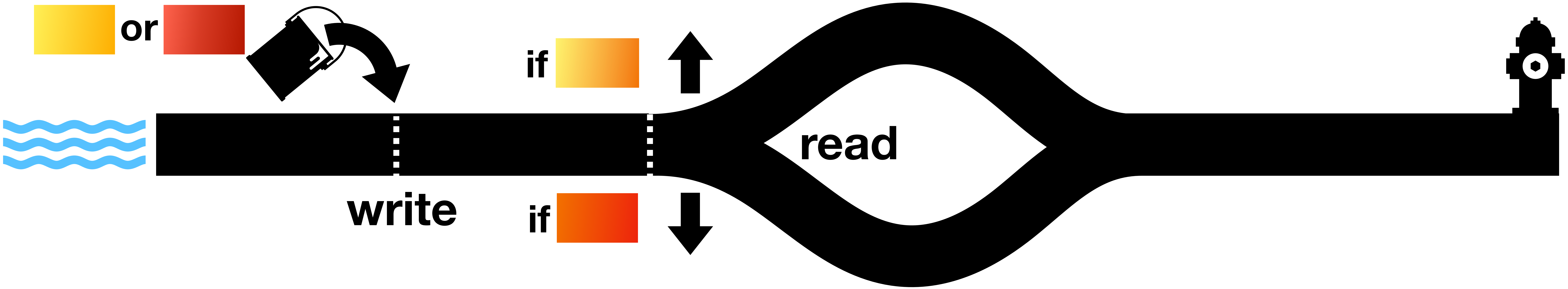
execution


or
write
if
if
read

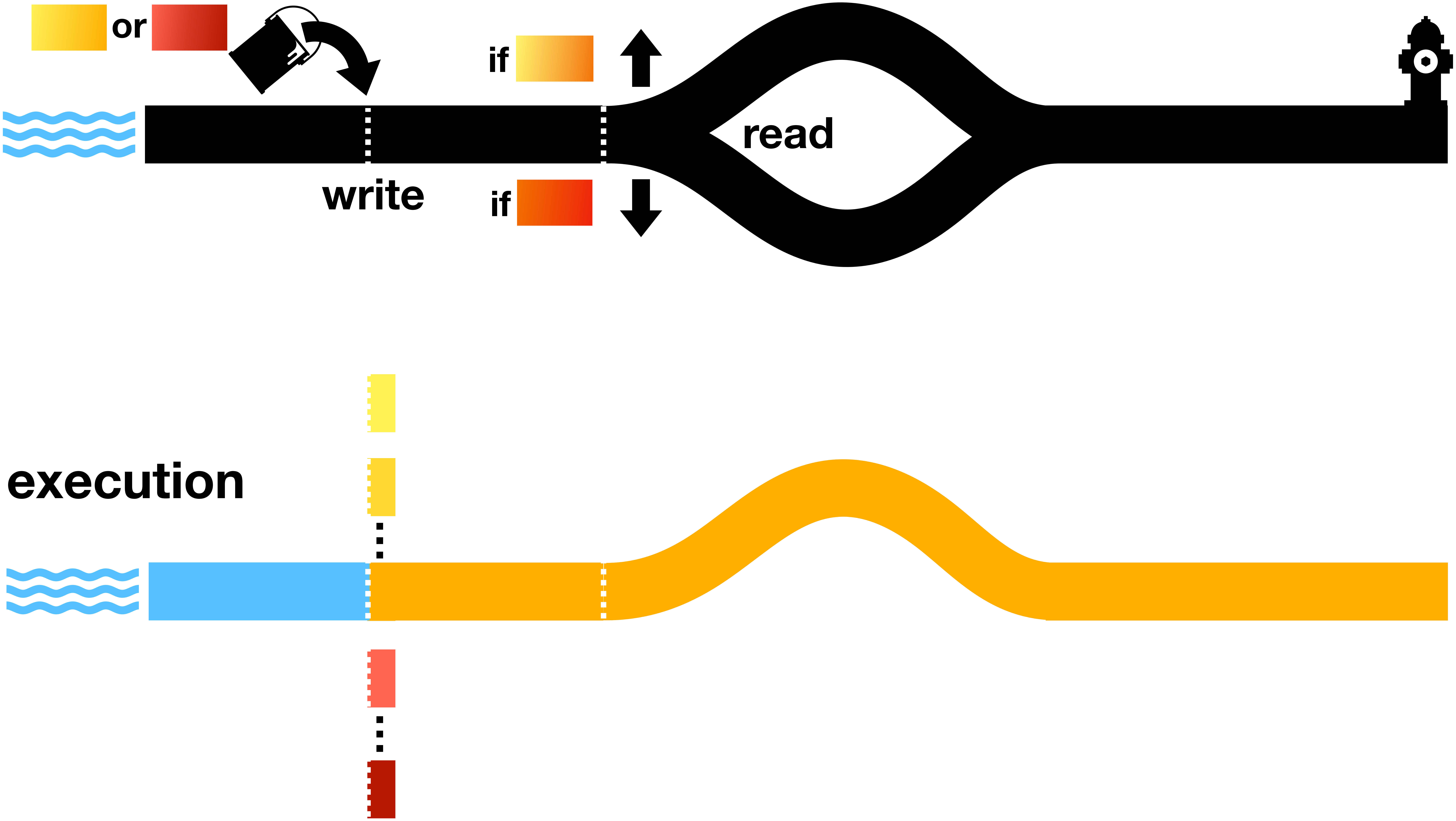

execution
pick

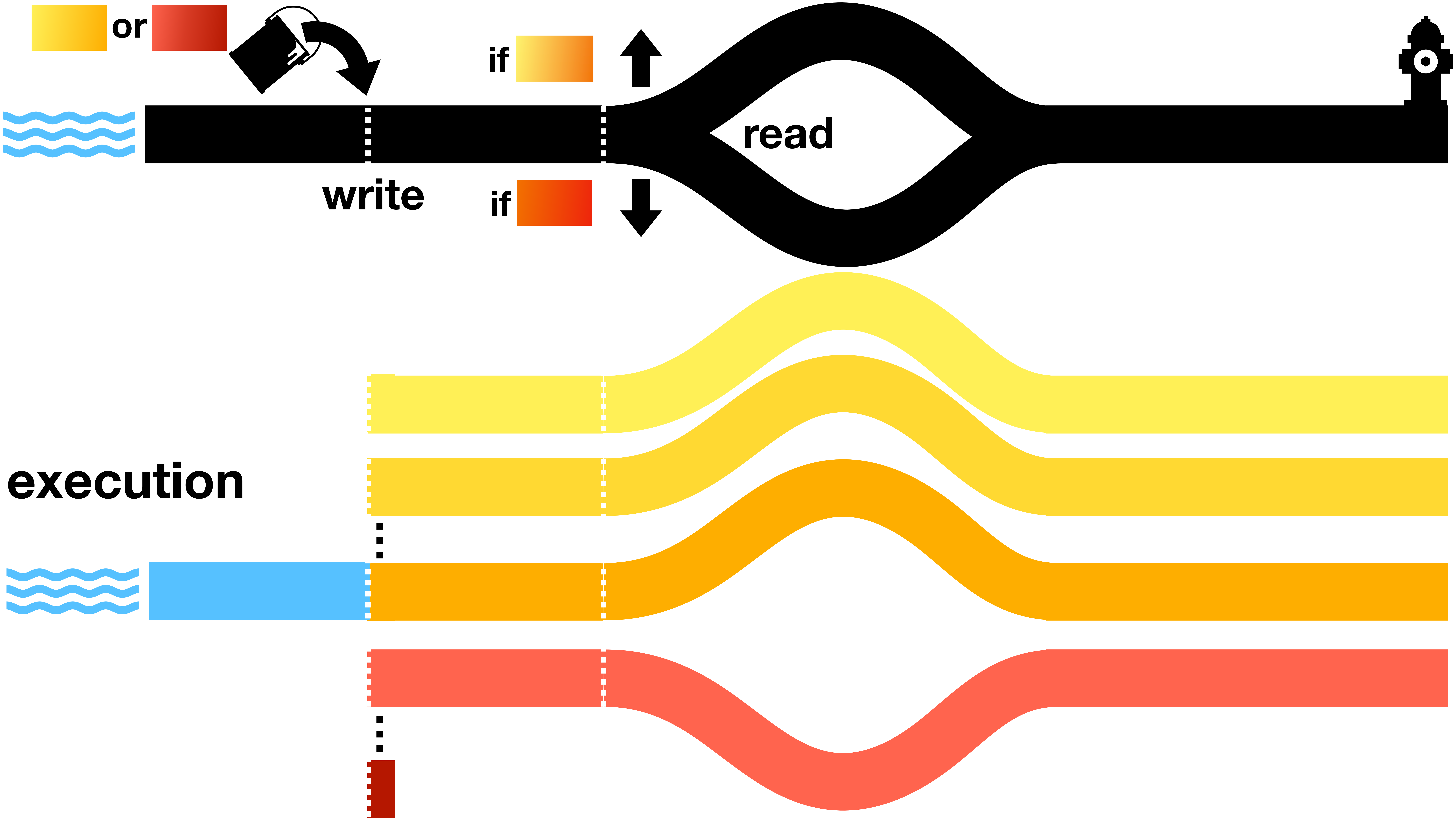


execution

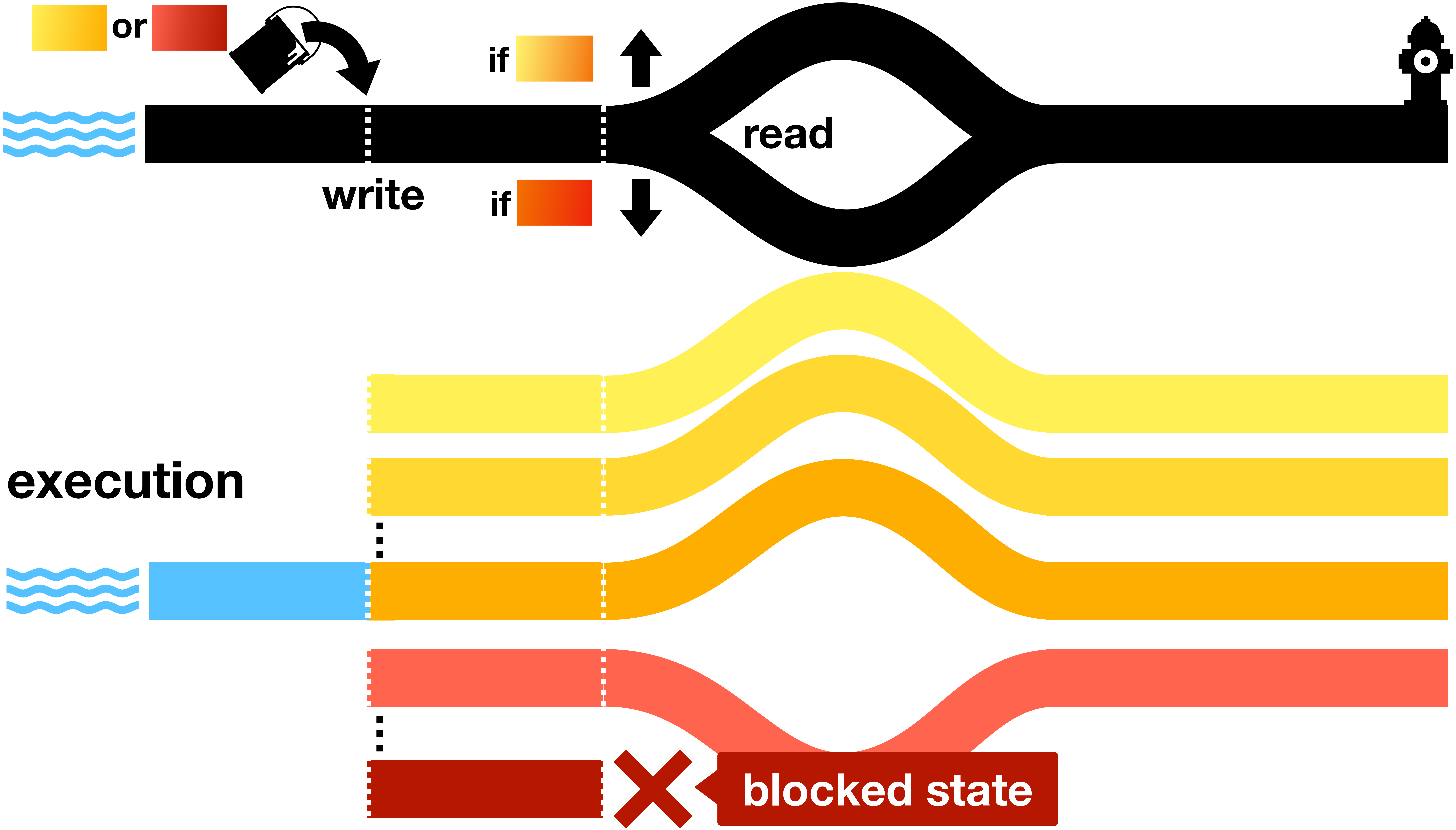


execution

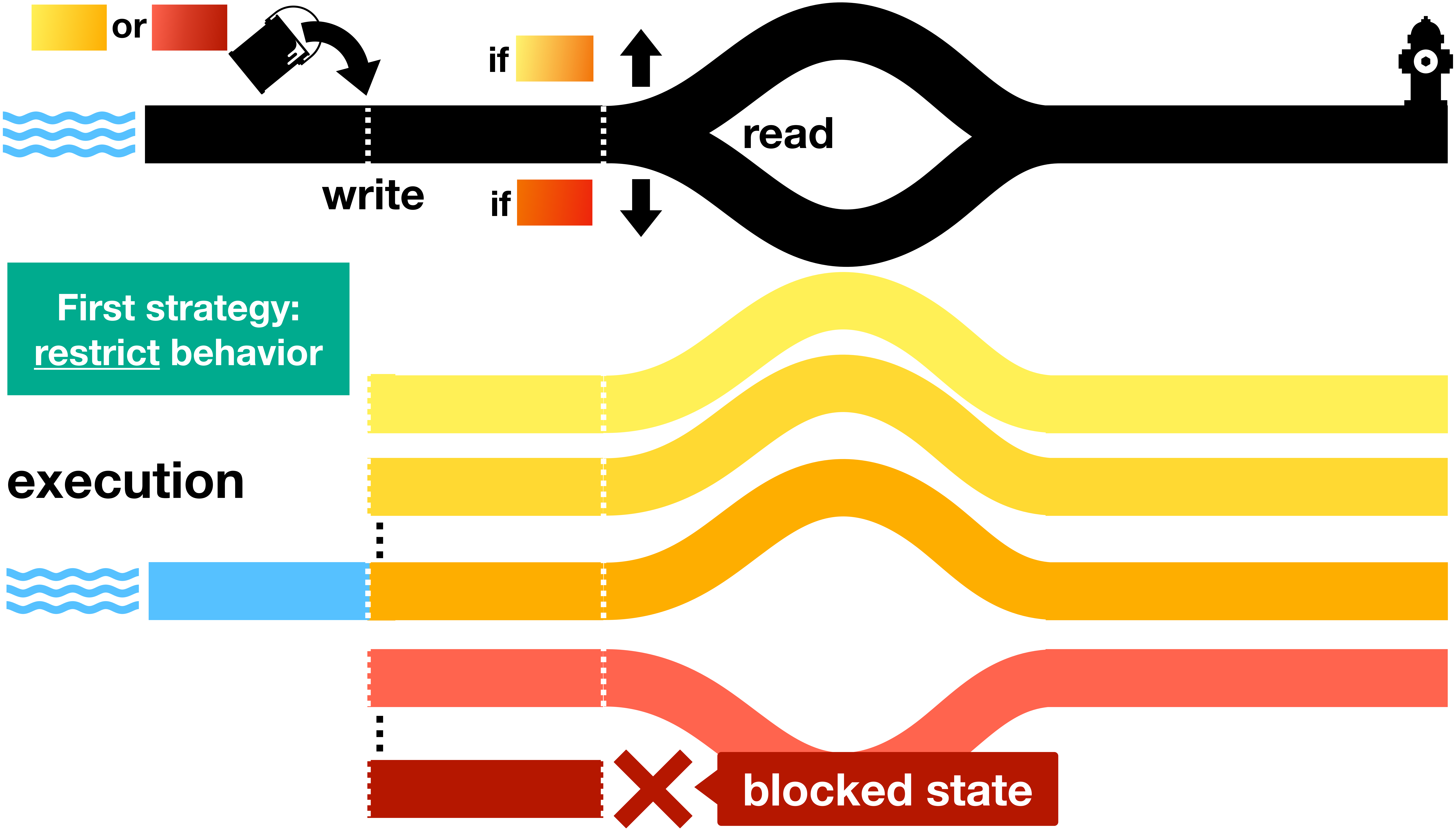

or
if
if
write
read
execution

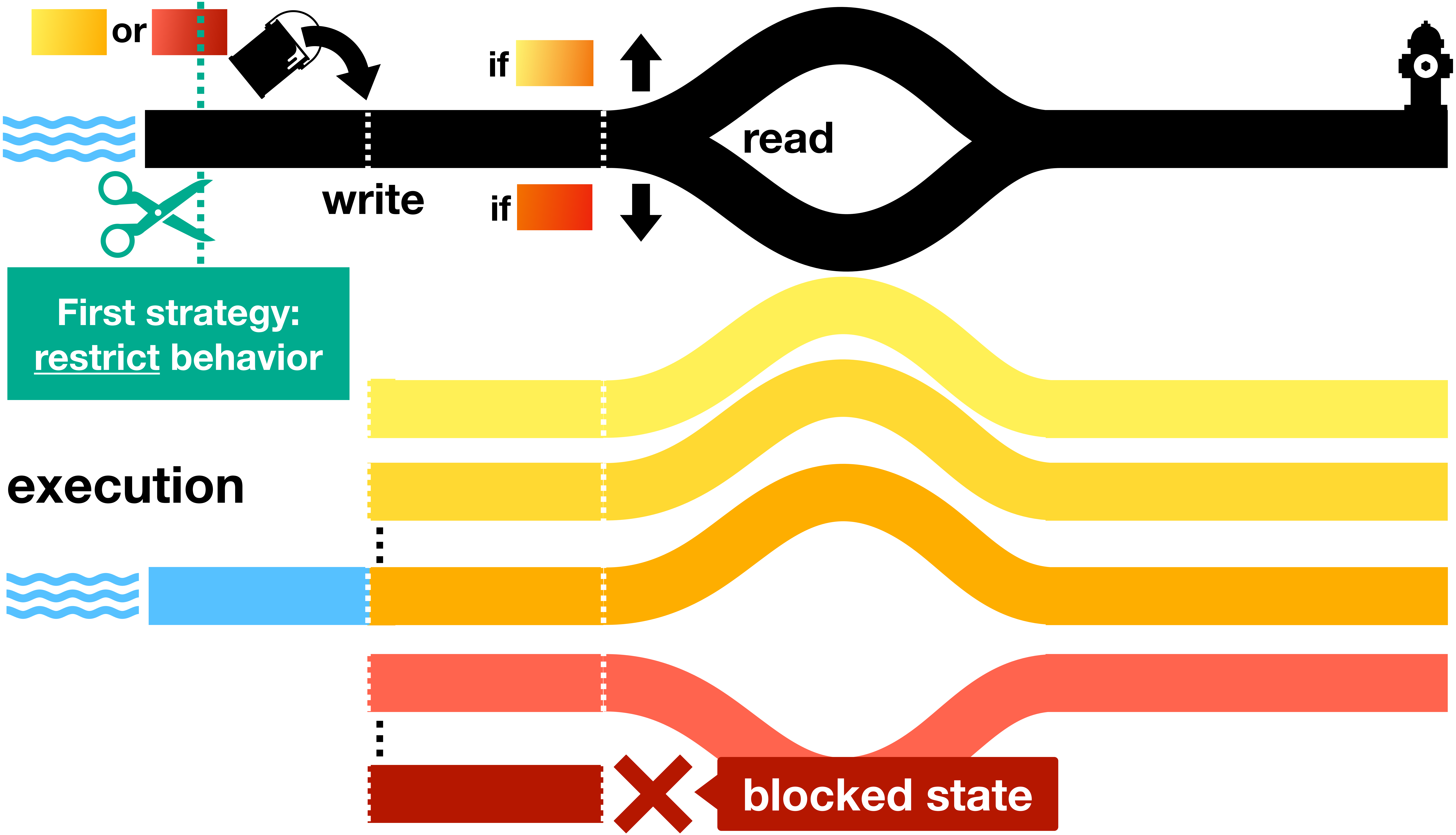


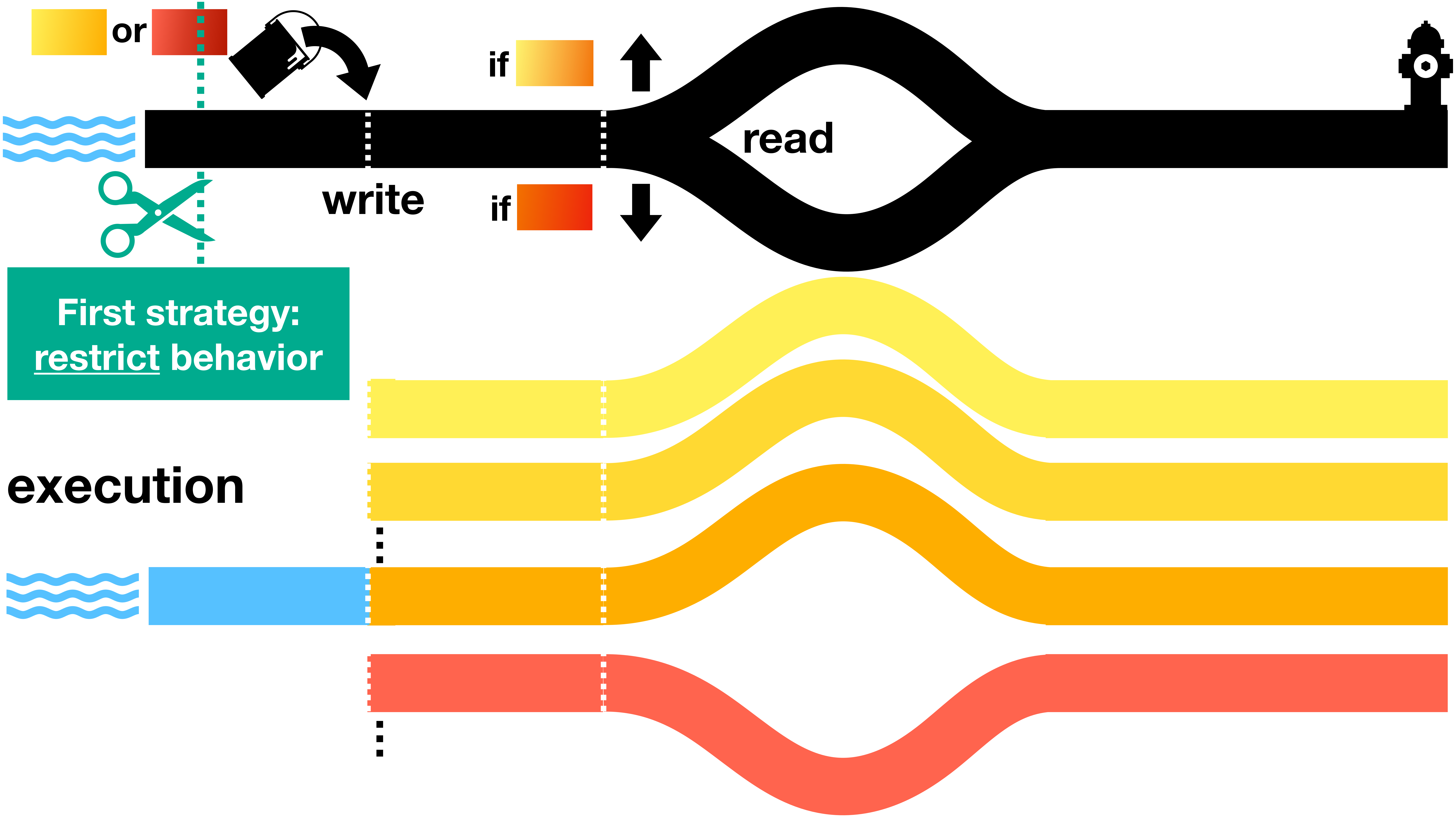
or
if
if
read
write
execution

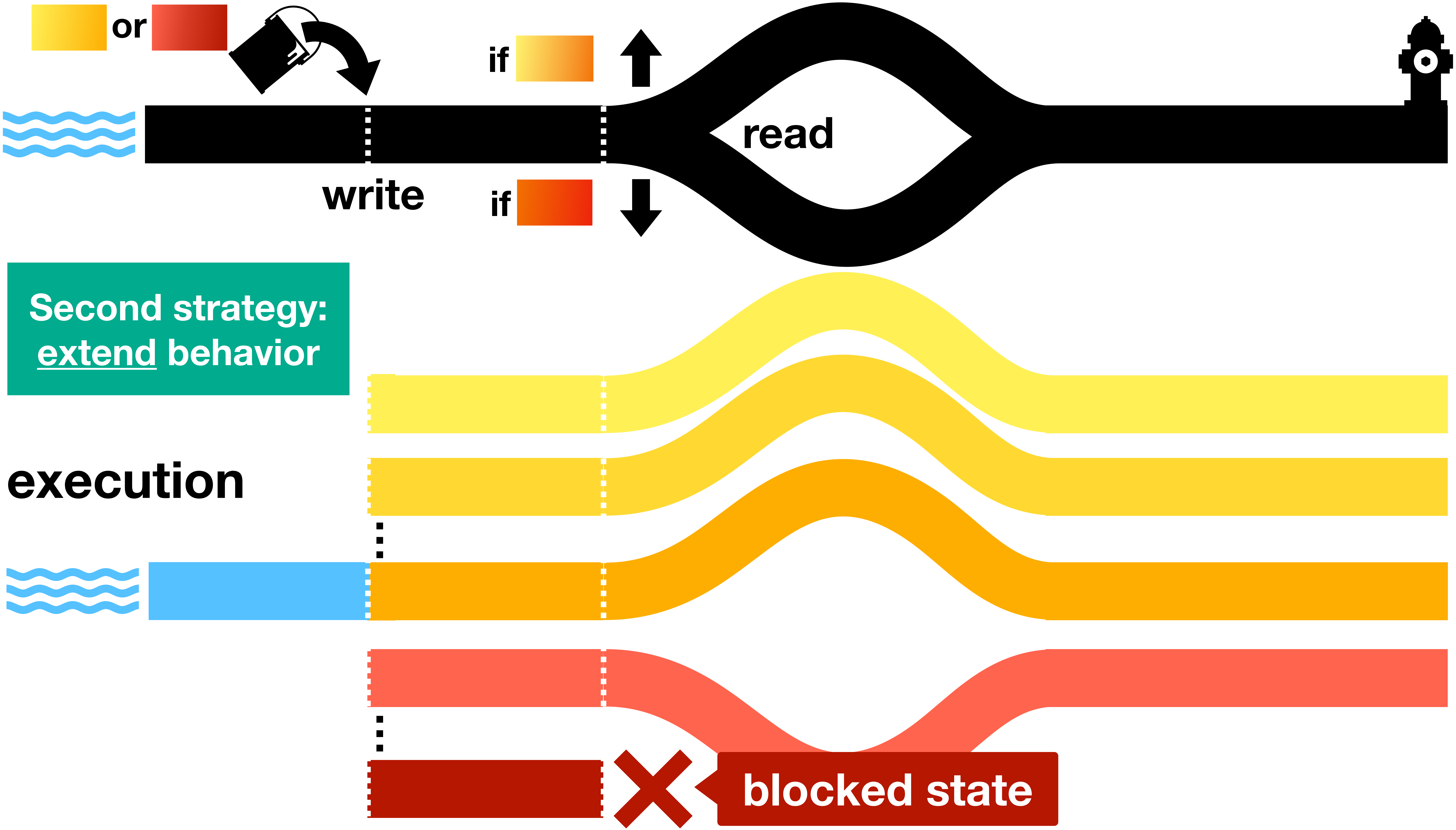


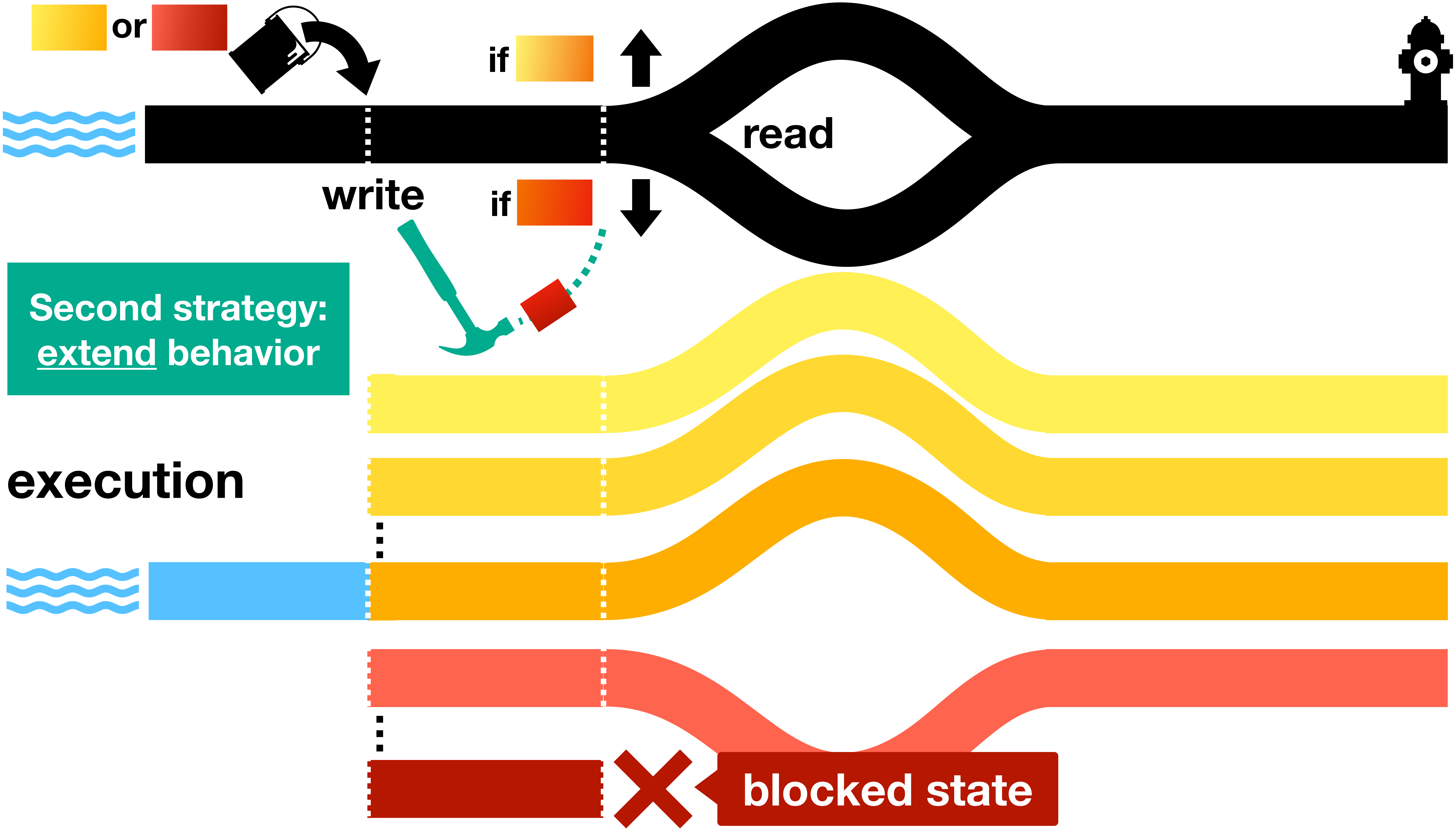
or
if
write
if
read
execution
blocked state


or
if
if
write
read
First strategy:
restrict behavior
execution
blocked state


or
if
if
read
write
First strategy:
restrict behavior
execution
blocked state


or
if
if
read
write
First strategy:
restrict behavior
execution


or
write
if
if
read
Second strategy:
extend behavior
execution
blocked state

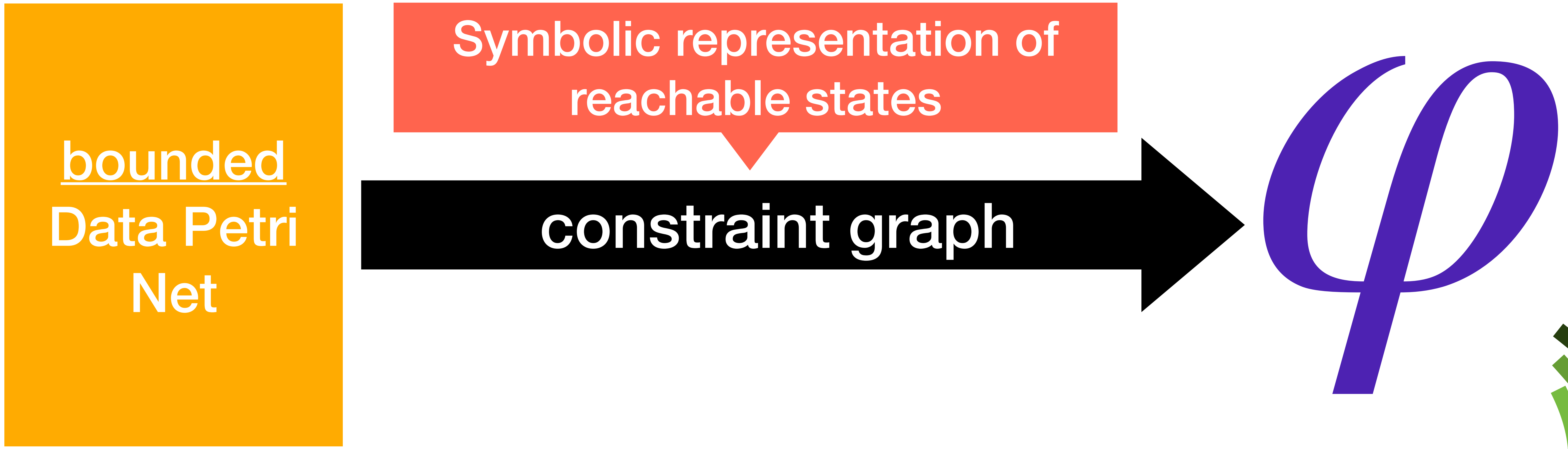


or
if
write
if
read
Second strategy:
extend behavior
execution
blocked state

or
if
write
if
read
Second strategy:
extend behavior
execution

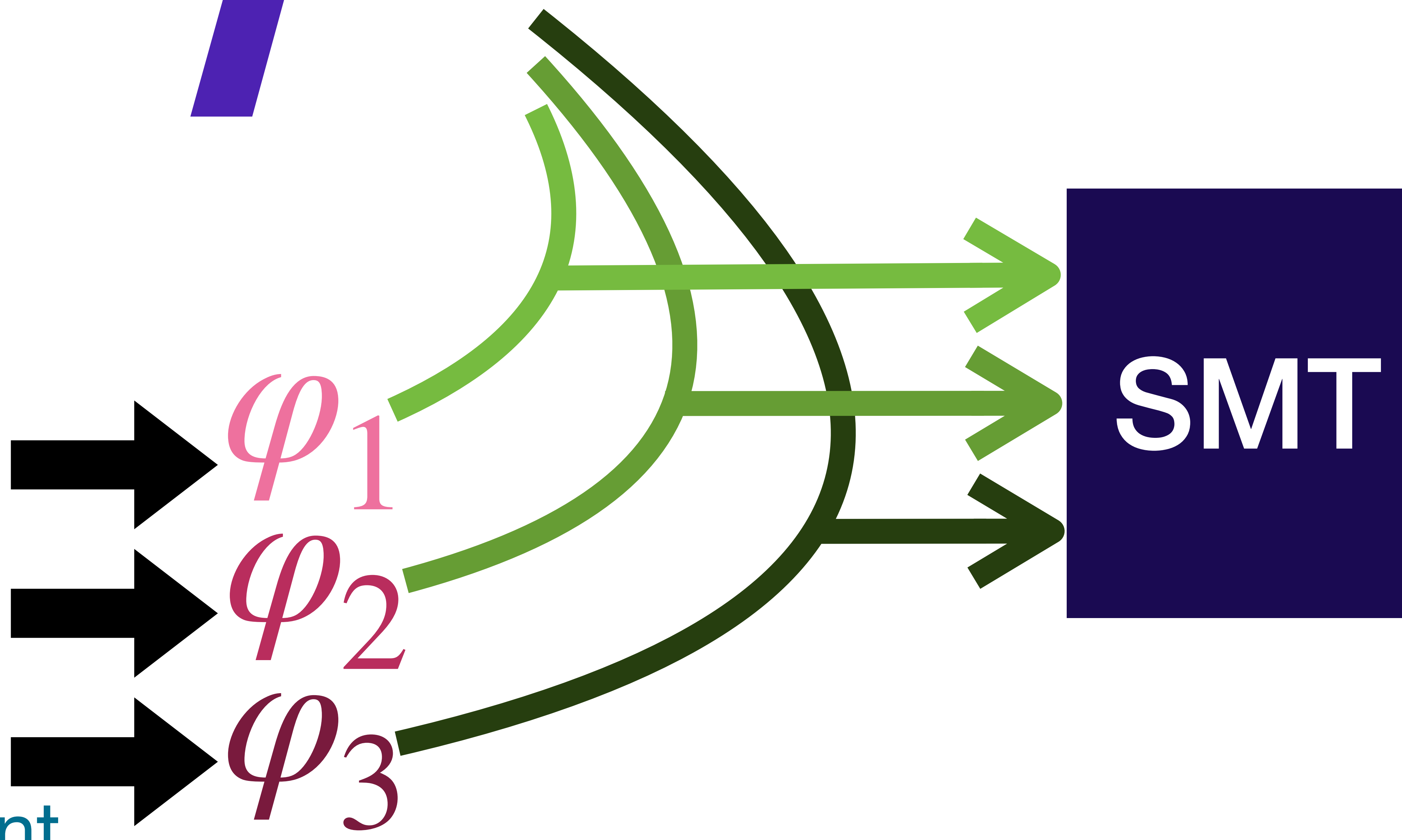
How to?

Step 1. Formula to characterise blocking runs



Data-aware soundness

- There are no dead tasks
- The final marking is only reached in a clean way for some variable assignment
- In every reachable marking, it must be possible to reach the final marking for some variable assignment



How to?

Step 1. Formula to characterise blocking runs

<u>bounded</u> Data Petri Net

Symbolic representation of reachable states

constraint graph

φ

Extract formula $\varphi_{blocked}$ capturing exactly those runs that get stuck

Data-aware soundness

- There are no dead tasks
- The final marking is only reached in a clean way for some variable assignment
- In every reachable marking, it must be possible to reach the final marking for some variable assignment

φ_1

φ_2

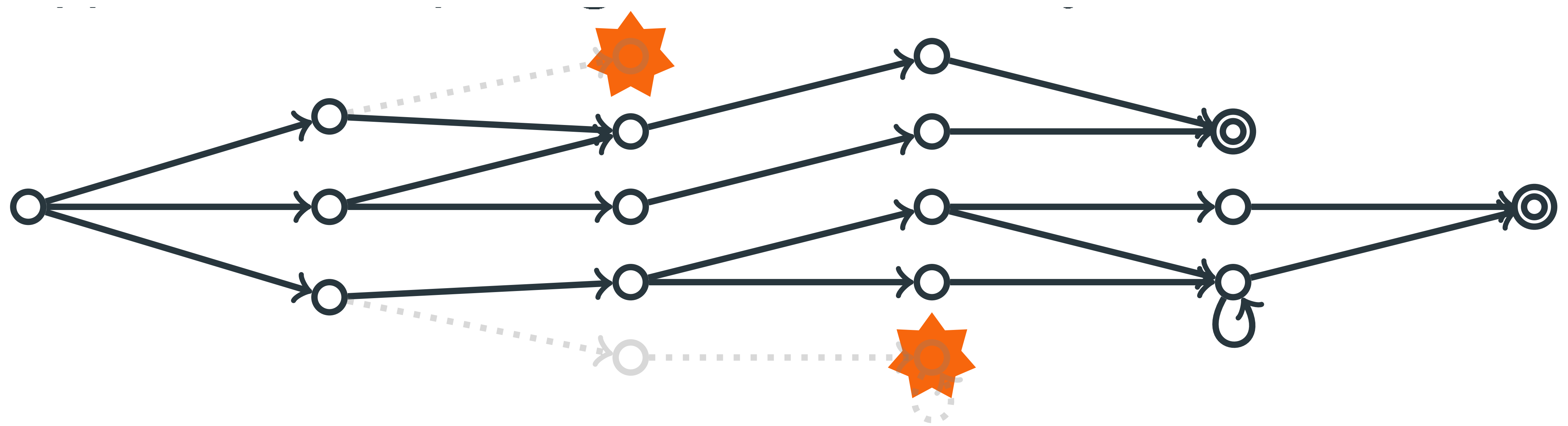
φ_3

SMT

How to?

Step 2. Carefully iterate over blocked states, using $\varphi_{blocked}$ to minimally avoid/unblock them

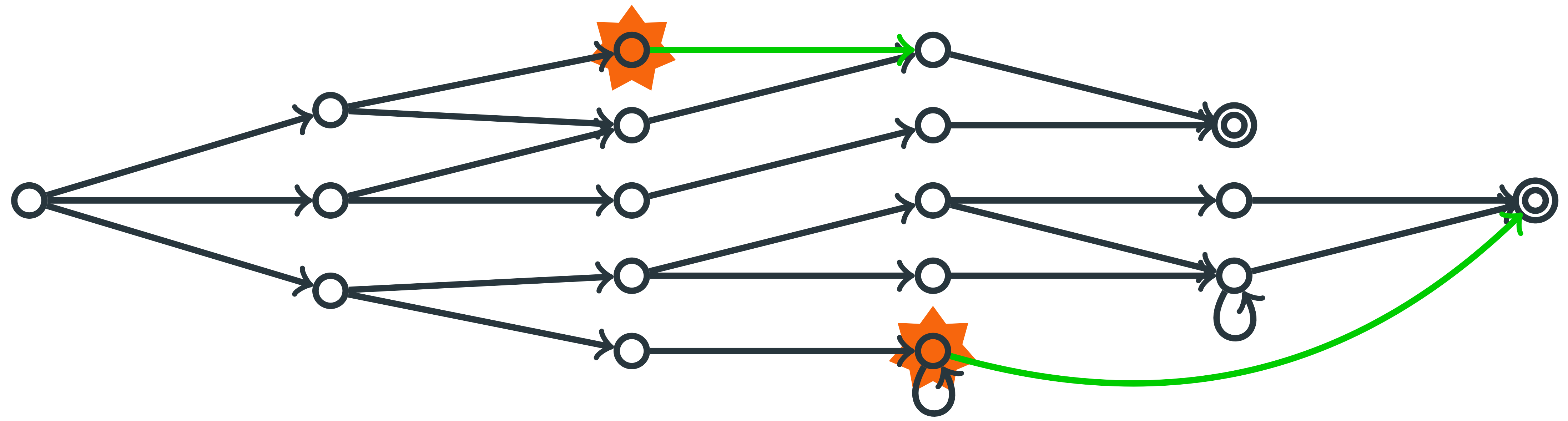
Restriction: avoid blocked states by tightening guards



How to modify guards?

- Retrieve formula $\varphi_{blocked}$
- Let *a* be a transition leading to that blocked state
- **Update** $guard(a) = guard(a) \wedge \neg\varphi_{blocked}$

Extension: let blocked states proceed by relaxing guards



How to modify guards?

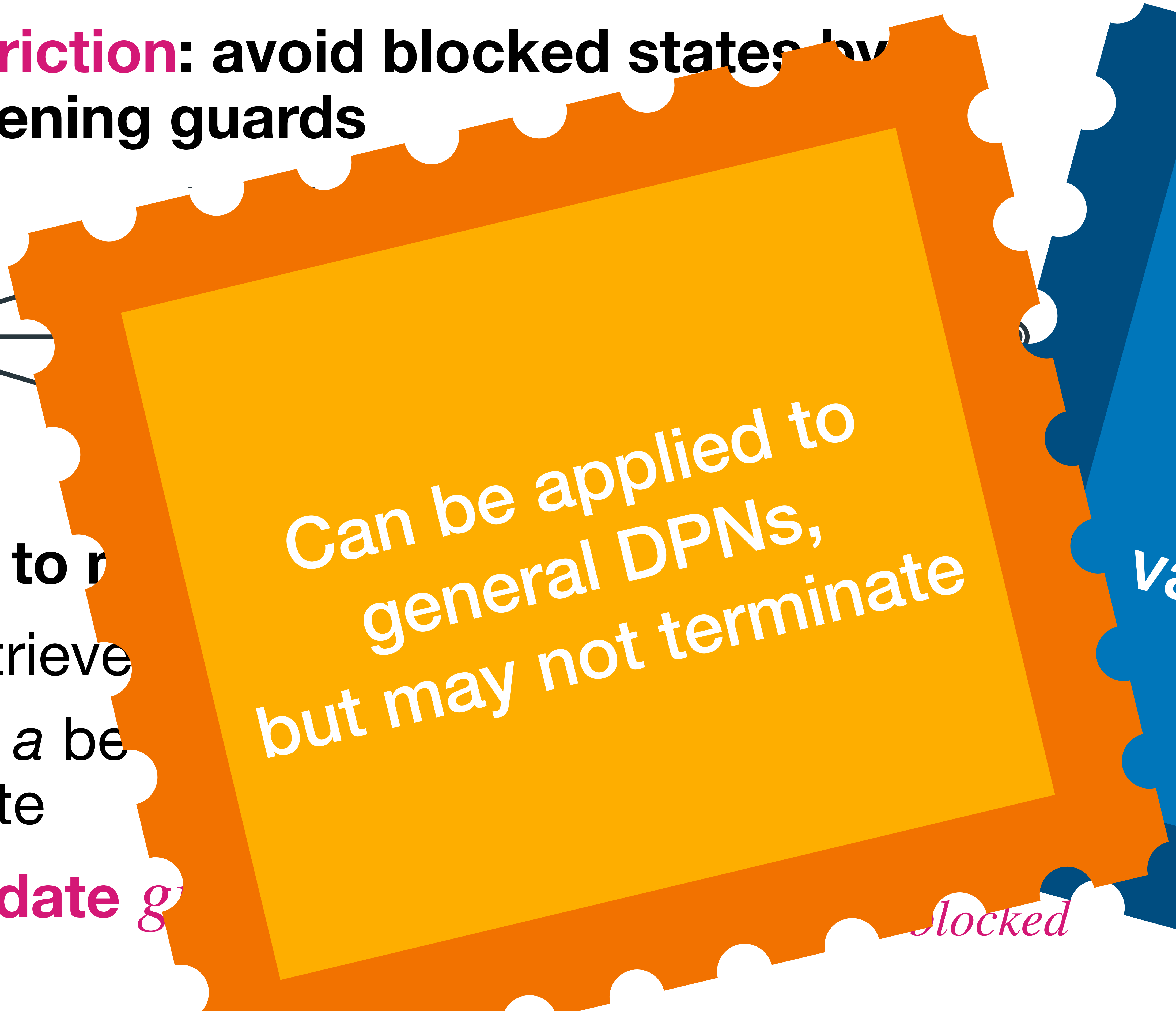
- Retrieve formula $\varphi_{blocked}$
- Let *a* be a transition leading to that blocked state
- **Update** $guard(a) = guard(a) \vee \varphi_{blocked}$

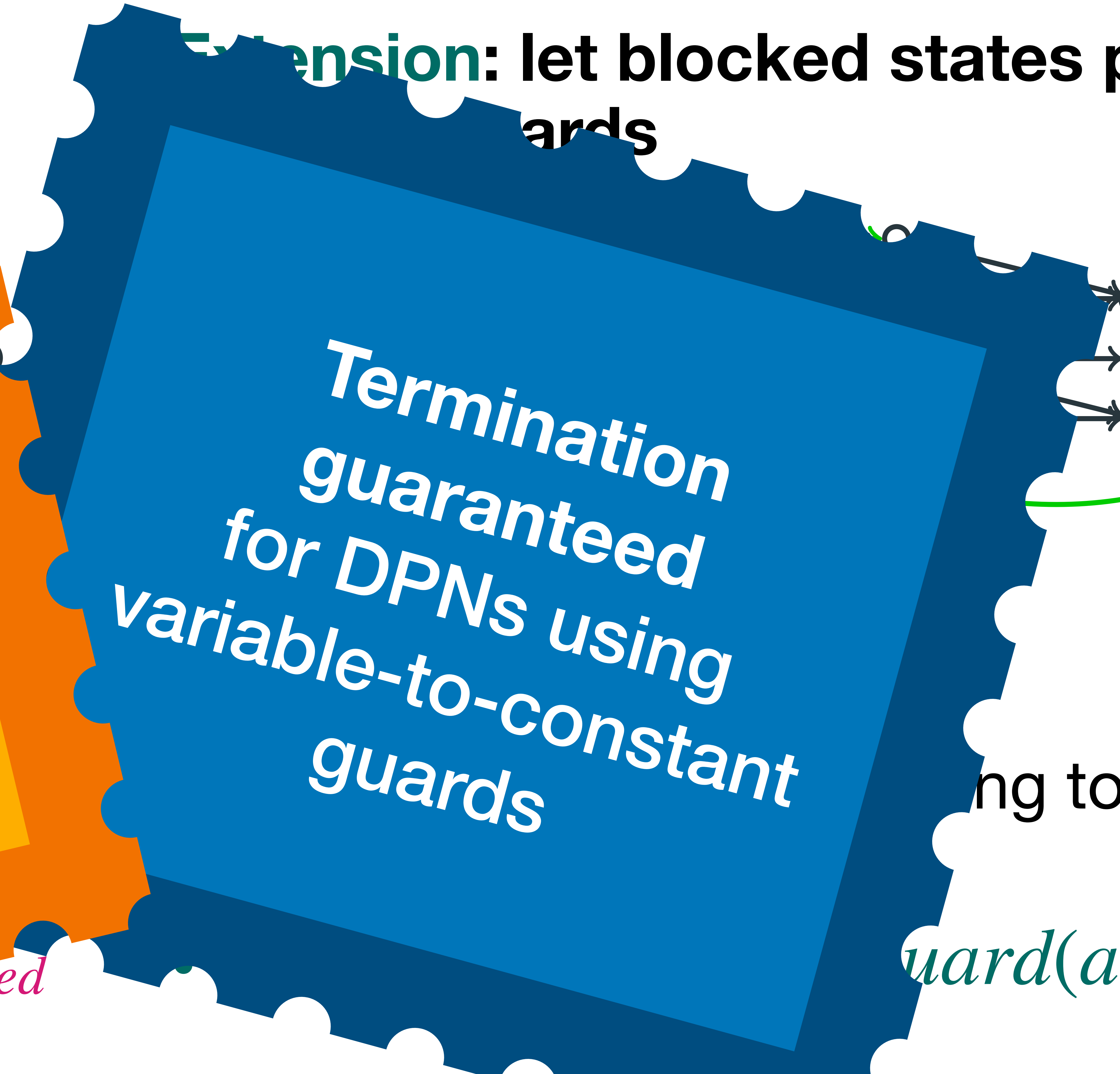
How to?

Step 2. Carefully iterate over blocked states,
using $\varphi_{blocked}$ to minimally avoid/unblock them

Restriction: avoid blocked states by tightening guards

Extension: let blocked states proceed by ...ards

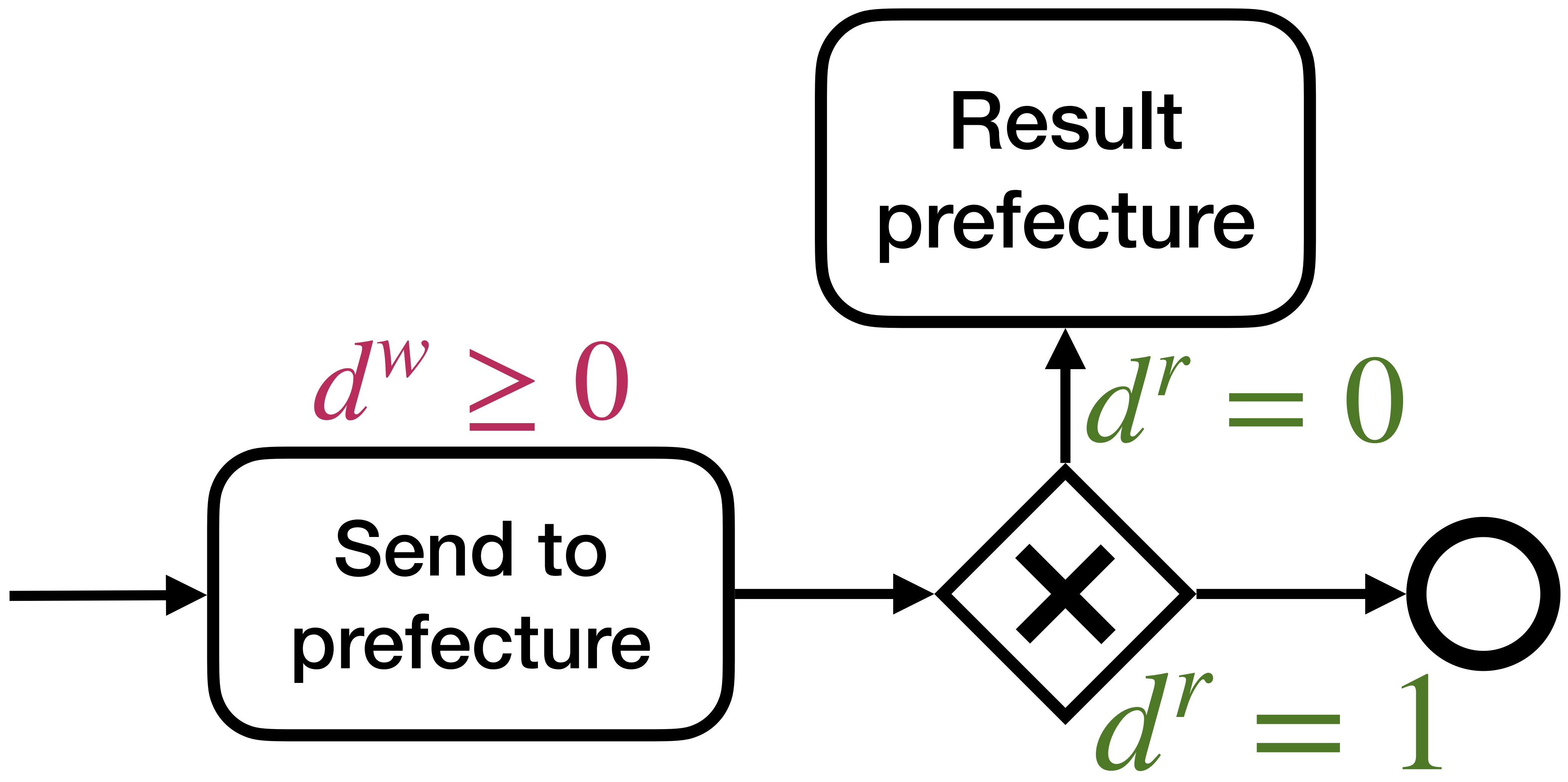




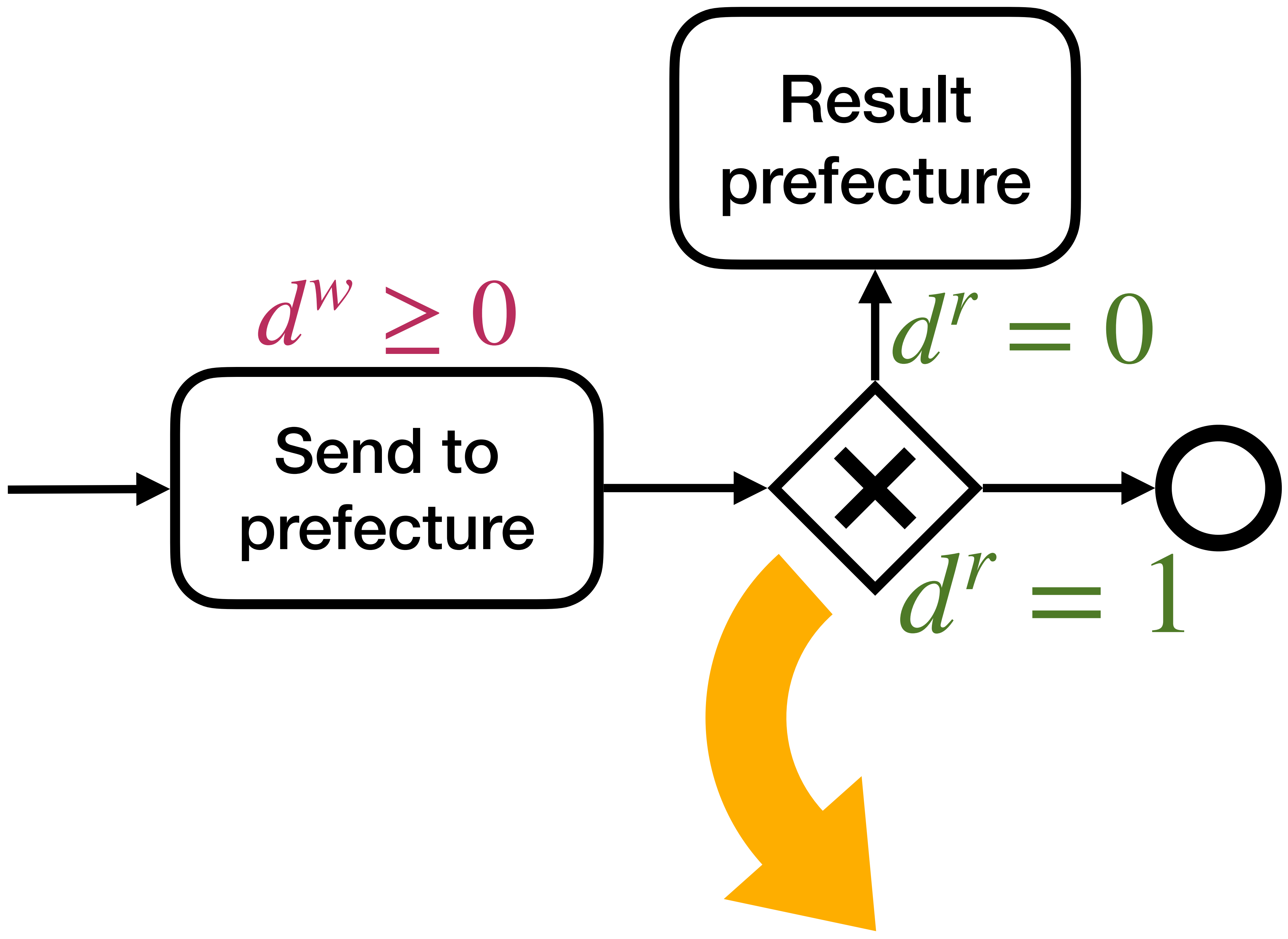
How to r

- Retrieve
- Let *a* be ... ng to that state
- **Update** g ... $\varphi_{blocked}$ ... $guard(a) \vee \varphi_{blocked}$

Back to the road fine example



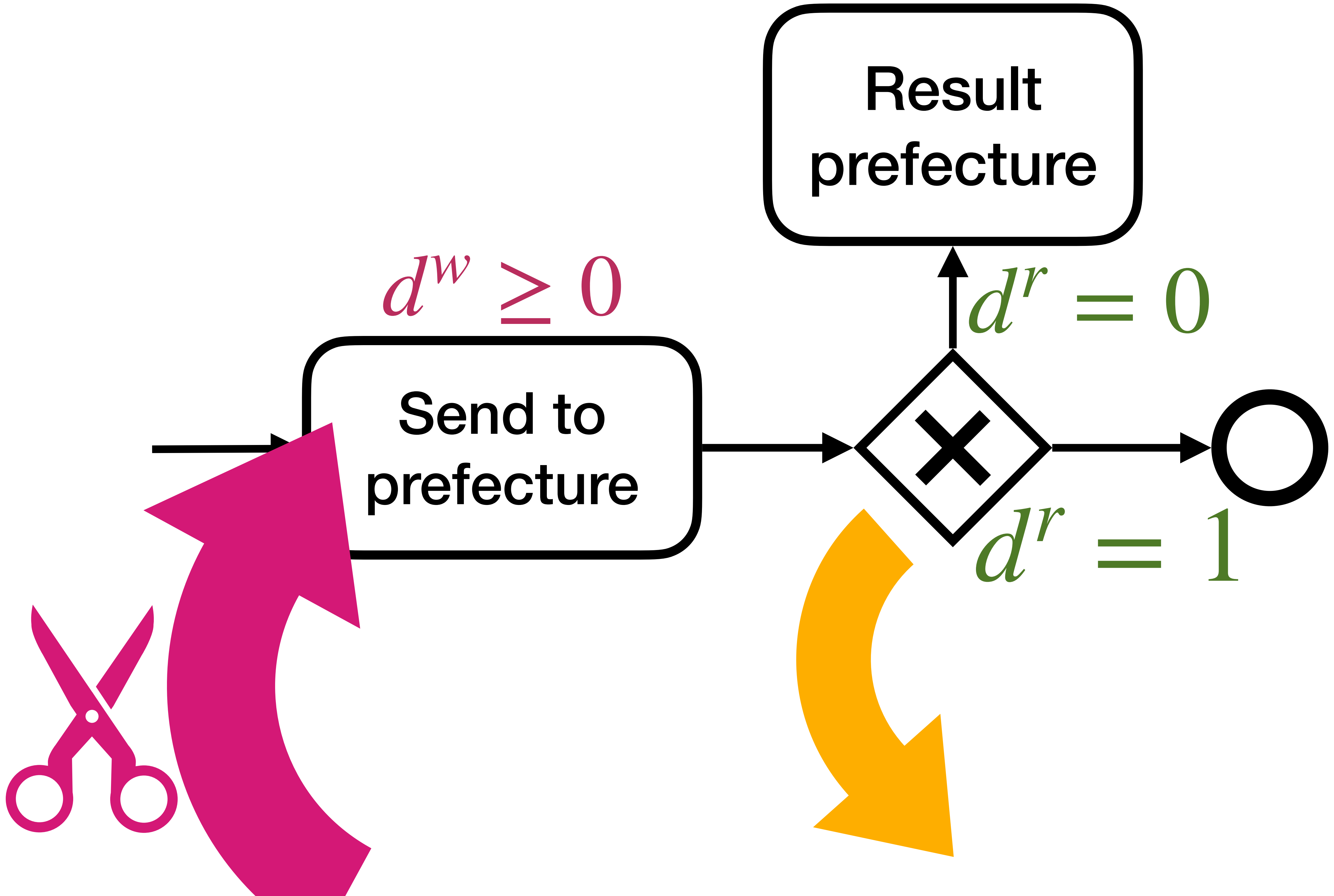
Back to the road fine example



$$\varphi_{blocked} = d > 0 \land d \neq 0 \land d \neq 1$$

Back to the road fine example

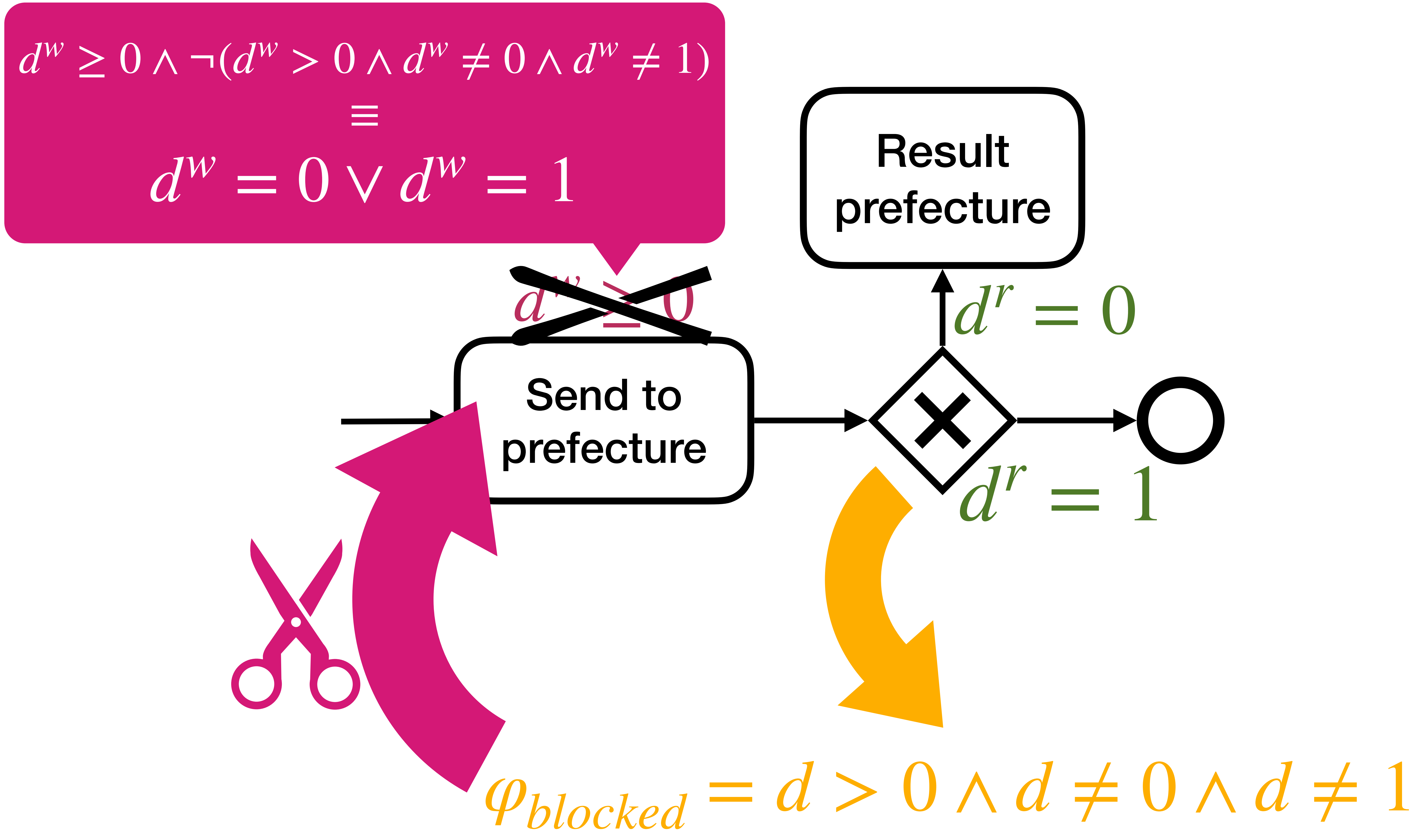
Restriction: modify the write guard on "send to prefecture"



$\varphi_{blocked} = d > 0 \wedge d \neq 0 \wedge d \neq 1$

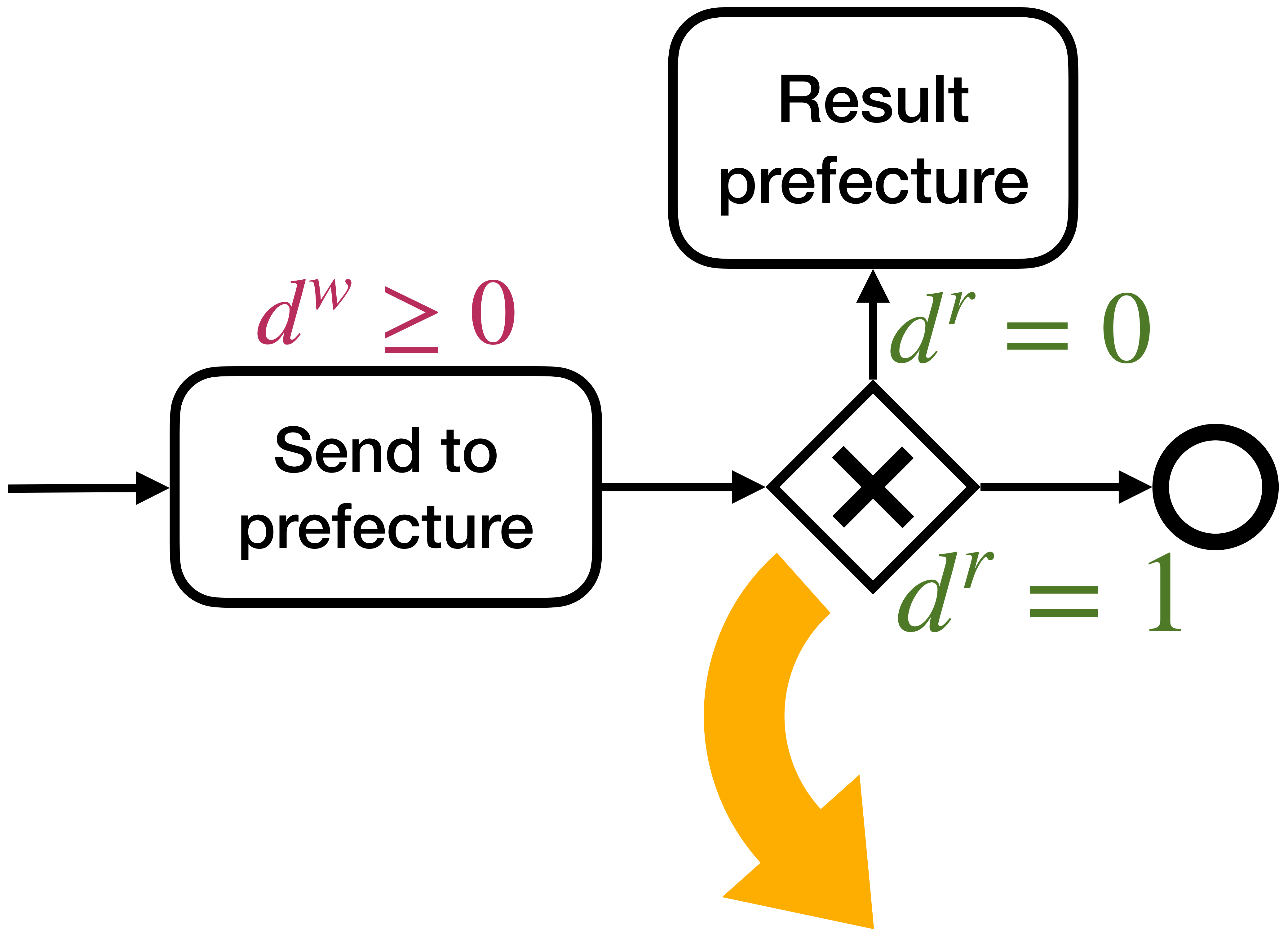
Back to the road fine example

Restriction: modify the write guard on “send to prefecture”



Back to the road fine example

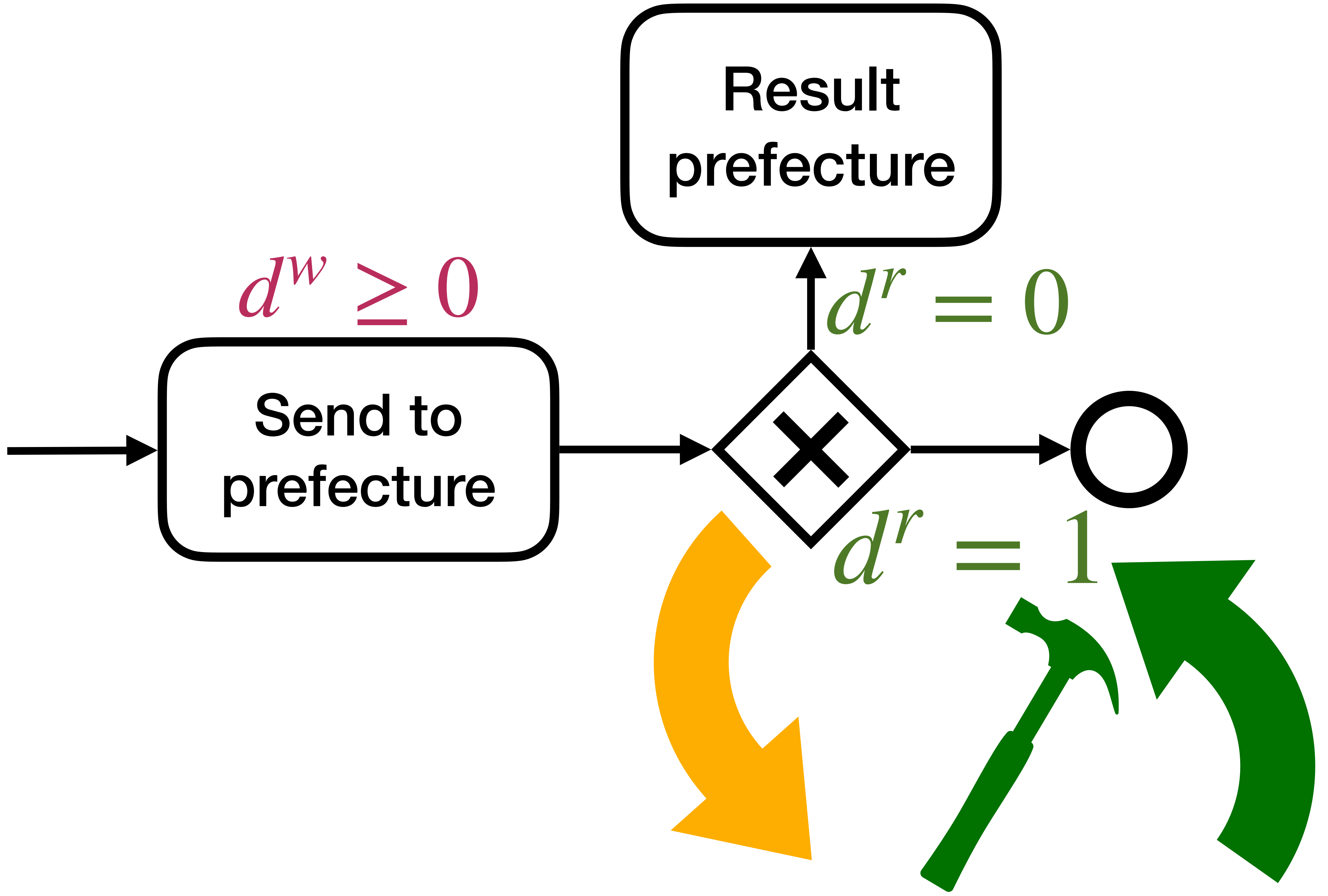
Extension: nondet. pick one of the two choice guards and fix it



$$\varphi_{blocked} = d > 0 \wedge d \neq 0 \wedge d \neq 1$$

Back to the road fine example

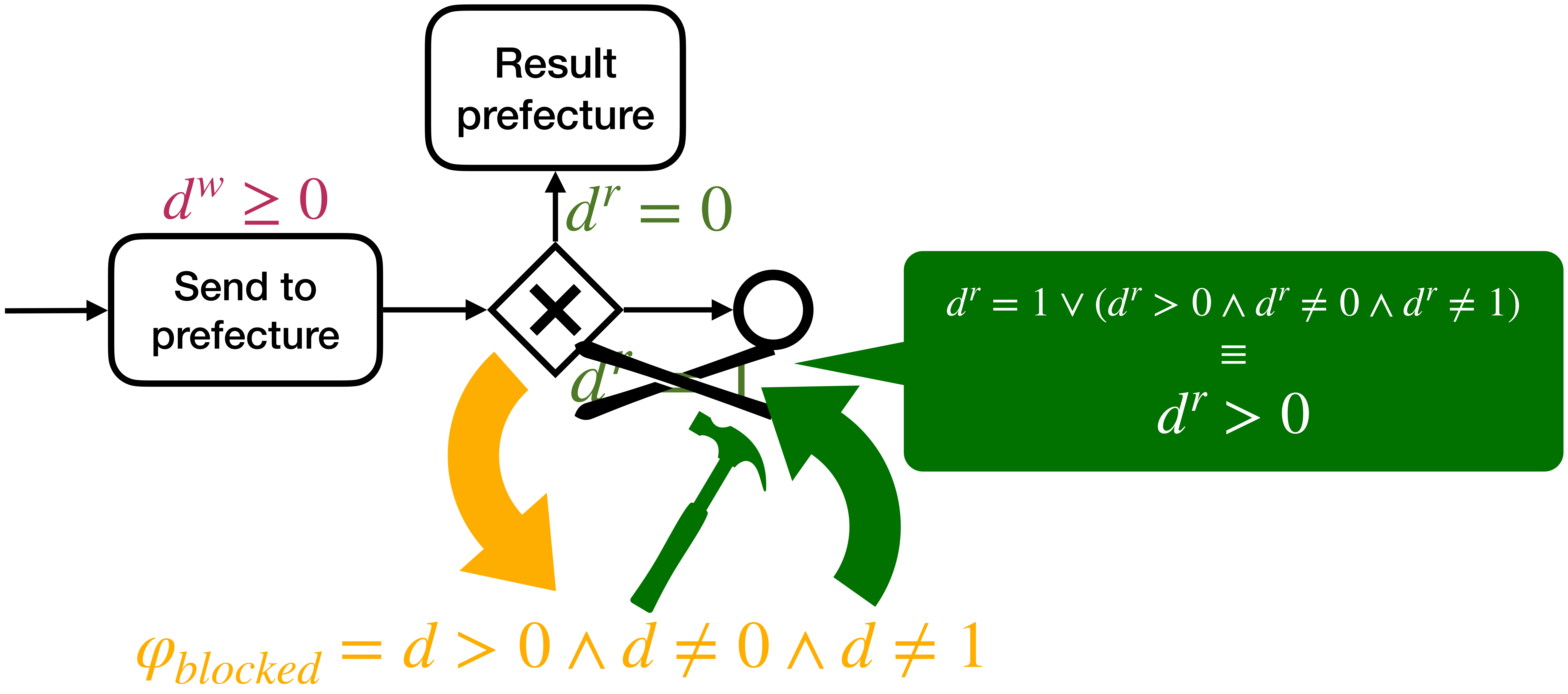
Extension: nondet. pick one of the two choice guards and fix it



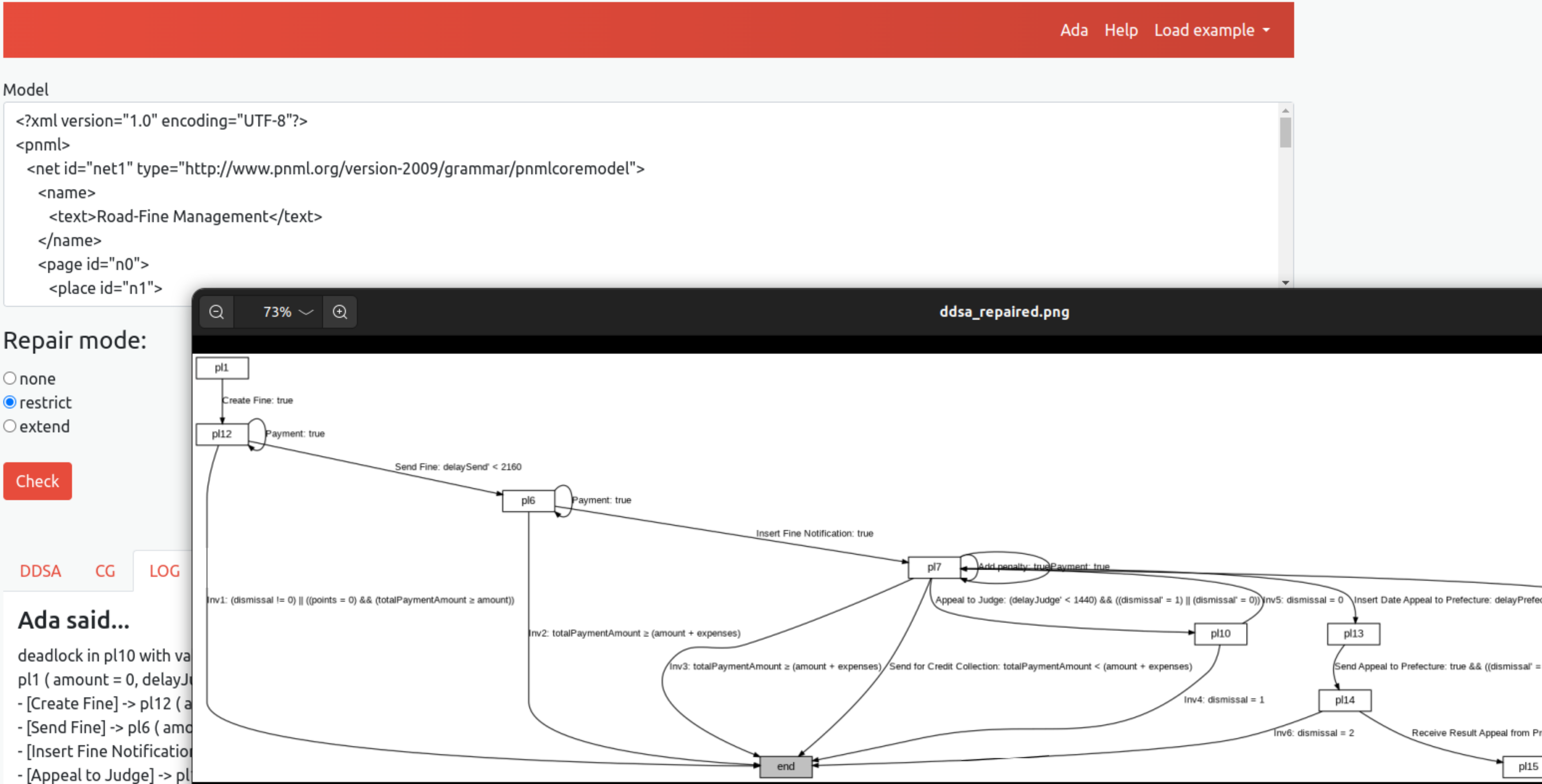
$$\varphi_{blocked} = d > 0 \land d \neq 0 \land d \neq 1$$

Back to the road fine example

Extension: nondet. pick one of the two choice guards and fix it



Fully implemented: soundness.adatool.dev



Experiments

Repair of unsound DPNs

DPN	repair/Restrict		repair/Extend	
	time	*# deadlocks*	*time*	*# deadlocks*
(a) road fines normative [MannhardtLRA16]	50s	2	71s	2
(b) road fines mined [Mannhardt18]	24s	1	22s	1
(c) dig. whiteboard/transfer [Mannhardt18]	2.1s	1	3s	1
(d) package handling [Fig. 5, deLeoniFM21]	6s	0	6s	0
(e) auction [FMW22a]	8s	1	20s	1
(f) auction example	2.5s	1	2.7s	1
(g) livelock example	2.1s	1	2.4s	1

Extension takes more time (larger constraint graphs)

Conformance checking (road fines example)

	original DPN	after repair/Restrict	after repair/Extend
average distance	1.2009	1.2009	1.1305

Repair does not affect conformance negatively

Conclusions

Take home

- Need of **soundness repair** in **data-aware process mining**
- **SMT-based automatic repair** for DPNs that is **conservative on the control-flow** and **on the original behavior**
- Two repair directions: **restrict** or **extend**
- **Fully implemented**

Next steps

- **Blending** of restriction and extension, user-in-the-loop
- **Log-driven data-aware repairs**

Repairing soundness properties in data-aware processes

Paolo Felli, Marco Montali, Sarah Winkler

ICPM 2023
Rome, Italy